Simplifying GIS: Developing web-applications for Atlas

By Group 21 and Group 22 in IS-304:

Group 21:	Lukas Aubert Andersen (lukasaa@uia.no) Marius Evensen (mariue17@uia.no) Sebastian Midtskogen (sebastiam@uia.no) Tom André Slåen Myre (tamyre@uia.no) Markus Nilsen (markusnil@uia.no)
Group 22:	Glenn Joakim Bakklund (gjbakklund@uia.no) Eirik Silseth Bjørdal (eiriksbj@uia.no) Markus Ribe (markusri@uia.no) Lars Henrik Bjørck Råkil (lhraakil@uia.no)

Course ID	IS-304
Course Name	Bachelor Thesis in Information Systems
Course Coordinator	Hallgeir Nilsen, Geir Inge Hausvik
Thesis Supervisor	Lucia Castro Herrera
Due Date	May 16, 2024
Word count	Approx. 18 000

	Yes	**No**
We confirm that we do not cite or use others work unless this is listed, and that all references are listed in the bibliography.	☑	☐
Can the submission be used for teaching purposes?	☑	☐
We confirm that everyone in the group has contributed to the submission.	☑	☐

Abstract

This thesis encapsulates the combined work and effort of group 21 and group 22 on two projects developed for the company Atlas (atlas.co) in order to complete our bachelor's degrees. These two projects involve developing web applications, in an attempt to modernize aspects of traditional Geographic Information Systems (GIS). Specifically, one project attempt to simplify a process called georeferencing, and the other attempt to create an introductory tool for creating maps utilizing AI based on text input.

Utilizing a tech-stack comprised of primarily Python and Next.js, we also explore different methodologies and tools for project management, such as the Scrum framework. We describe how we utilized the various methodologies and tools, and explain the implementation and development of the projects. We describe some more technical aspects of the projects, such as describing deployment pipelines. We further argue on the quality aspects of the projects, and present the final products of each project, as well as deliberating on the projects' journies and results.

In assessing our project goals, we acknowledge the progress made in simplifying GIS functionalities with Image2Map and Text2Map. Although Image2Map streamlines georeferencing, it lacks advanced image manipulation and GeoTIFF modification features. Text2Map succeeds in query-based visualization, but requires refinement in data input and map shareability. Despite these limitations, both applications offer valuable features and lay a foundation for future development. The groups have gained significant experience, meeting personal academic standards. Although not fully meeting initial expectations, a satisfactory note from Atlas, as well as our experiences as a collective group, suggests that we have completed the projects in a respectable, professional, and satisfactory manner.

Preface

Welcome to our thesis detailing our exciting experiences creating applications for Atlas as our Bachelor's project. During this project and our time at the University of Agder, we have encountered many challenges and reached many personal milestones. In addition, we have had the pleasure to work with and meet so many incredible people. We therefore think it is time to properly thank these people:

First, we would like to thank our wonderful thesis supervisor, Lucia Castro Herrera, for the great support during the semester and for helping us with proof reading and giving valuable feedback, booking meeting rooms and showing up for us so often.

Further, we would like to thank our professors in IS-304, Geir Inge Hausvik and Hallgeir Nilsen, and the Atlas team, fronted by the best hype-man – Fredrik Moger, for the opportunity to write such an interesting thesis. We learned so much during our time developing these two projects; about how we work as a group, how we work individually, and our flaws and strengths.

Another thanks goes to Terje Gjøsæter, for letting us conduct user testing during one of his classes (and for being an excellent professor).

As this thesis marks the end of our course of study, we would therefore also like to thank all the other professors we have had the pleasure of meeting at UiA, including (but not limited to) Janis Gailis for his technical knowledge and pure enthusiasm about technology, Øystein Sæbø for engaging and motivating us since the beginning, Polyxeni Vasilakopoulou, Espen Limi, Niels Frederik Garmann-Johnsen, Peter André Busch, Pouria Akbarighatar, Ilias Pappas, Marco Seeber, and the rest. Thank you for teaching us so many things and guiding us through the courses! It has been six great semesters and we thank you for that.

We would also like to thank our families, (four-legged) friends, and lovely partners for the love and support throughout this semester, as well as the other semesters.

Lastly, we thank **you**, the reader, for taking the time to read this thesis!

Glossary

To ease the reading experience throughout the report, we present some terms and definitions which might enlighten the meaning of the report itself.

- **AI (Artificial Intelligence)** has no standardized definition, but is commonly used to refer to systems that can somewhat mimic human behaviour/intelligence (European Commission et al., 2021, p. 9). It is a rather broad term that encompasses many different technologies and systems, such as OpenAIs ChatGPT. In our case it is mostly used to interpret text and generate responses.
- An **API (Application Programming Interface)** can be used to send, get, edit and delete data from an already existing system (Goodwin, 2024). In our case, APIs are used in the projects to do various tasks in the back-end, like conversion or map-manipulation.
- **Branch** (in version control) is a term used for any different line or version of a project that deviates from the main line of development, where work can continue without affecting the main line (Chacon & Straub, 2024, p. 63)
- **Docker** is a platform and tool used to build, run, share and verify an application easily without the major environment setup and management (Docker, 2024). This means that we can build Docker-images that can then be run in a Docker-container, which we can transfer to other machines. When these images and/or containers are transferred, it is almost guaranteed that they behave the same way as they do on the other machine.
- **Figma** is a collaborative design-tool used for modeling and designing websites and user interfaces, and is widely used in the design community for its ease of use, powerful features, and ability to streamline the design process (Figma, 2024).
- **GCPs (Ground Control Points)** are identifiable markers on the Earth's surface with known geospatial coordinates, and serve as reference locations for georeferencing aerial or satellite images with real-world geographic coordinates (Zmejevskis, 2022).
- **Geo-TIFF** is a tiff-image utilized for storing geographical metadata, which means that the image (e.g. satellite imagery or maps) has metadata to inform programs where the image exists in the real world (Sazid Mahammad & Ramakrishnan, 2003, p. 4).
- **Georeferencing** is the process of taking a digital image, such as a scanned photo of a map or a satellite picture, and embedding geographical data. This is achieved by pairing the image's pixels with the latitude and longitude of where it exists in the real world to bind them together (Science Education Resource Center, 2024; USGS, n.d.).

- A **GIS (Geographical Information System)** is a system that is responsible for creating, saving, manipulating and presenting information regarding geographical data, like the surface of the Earth (National Geographic, 2023).

- **Git** is a version control system (VCS) designed to track changes to files over time. One of the key features compared to other version control systems is that Git stores snapshots of files rather than just changes, which makes it efficient and good for large projects. It makes it possible to work on local computers, without the need to constantly be connected to a server. Additionally, it checks files to make sure they have not been changed by accident (Chacon & Straub, 2024, p. 14-17).

- **JSON (JavaScript Object Notation)** is a file type used to store structured information, and is typically used for data-transferring (MDN Web Docs, n.d.). In the project, it is used to transfer and use geographical data about locations (GeoJSON), and sometimes as configuration files.

- **LLMs (Large Language Models)** are defined by IBM (n.d.) as a category of foundation models trained on immense amounts of data, a process that makes them capable of understanding and generating natural language and other types of content to perform a wide range of tasks.

- **Mapbox** supports maps and location services for a wide variety of web, mobile, automotive, and gaming applications (Mapbox, n.d.). Whenever we refer to Mapbox in this project we are specifically referring to Mapbox GL JS React which is a JavaScript library for creating interactive vector maps on the web (Mapbox, 2023).

- **Merge** (in version control) is a term used for the process of joining two or more development histories together into one. (Chacon & Straub, 2024, pp. 53–54).

- **MVP (Minimum Viable Product)** is the initial version of a product with the core set of essential features necessary to achieve the project's primary objectives (ProductPlan, 2022).

- **PNG** is a widely utilized image format, due to its lossless data compression and possibility for transparency, however they are generally larger than formats such as JPG and GIF (Adobe, n.d.).

- **Pull request** (in version control) is a term used for a request that signals the desire to merge a collection of changes from one git-branch in to another. (Chacon & Straub, 2024, p. 144)

- **Repository** is a remote storage location that contains code and files (GitHub, n.d.-b).

- **Tech-stack**, or **stack**, is a collection of different technologies that get used to develop and shape a project; it is used to describe the combination of programming languages, frameworks, the front-end, and the back-end used in a project (Gracilla, 2022; Heap, n.d.).

- **TIFF (Tag Image File Format)** is a flexible image file format that is used for storing high-quality images along with a wide variety of metadata (Adobe, 2023).

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Understanding the world through maps has been an important part of human history for millennia. With the introduction of computers, maps and tools were digitized to take advantage of the digital benefits that computers could bring (Damoor, 2019; Geoapify, 2023). While more and more applications and tools are becoming web-based (such as Google Docs, Figma, etc.), with a large focus placed on ease of use, the world of GIS appear to have been falling slightly behind. GIS tools tend to have a high barrier of entry and difficulty of use, creating a growing need for services that are easy to learn and easy to use (FuseGIS, n.d.; Goldin & Rudahl, 1997; Sheehan, 2018). In the next chapter, we will go over the two projects we are working on, our employer, our two teams, and our goals and ambitions.

2 Background

Understanding what the two projects entails, who our client is, both teams, as well as our goals and ambitions is essential before diving into how we solved both projects.

2.1 About the projects

With the above-mentioned needs in mind, let us introduce our two solutions to help mitigate some of the problems faced with GIS-software:

The first project is named Image2Map, where the main goal is to create an application in which users, even those with minimal experience with GIS, can georeference images quickly and easily. In other words, this means that the user should be able to upload an image of a map or a satellite photo, then georeference it, and lastly receive a file with updated metadata like coordinates.

This should be done by presenting the user with a split-view, where there is an interactive map on one side and the uploaded image on the other. They should then be able to place markers on both the map and the image, to reference real life coordinates to the image's pixels. Additionally, the user should be able to preview the georeferenced file by showing the file overlayed on an interactive map so that the user can check if the result is as desired. Lastly, the user should be able to perform some sort of post-processing after uploading, such as cropping the image, so that the user requires as few additional tools to get a completed image.

The second project, Text2Map, is intended to be a simple-to-use tool for visualizing the placement of locations around the world through the use of a given text or data source. By inputting a piece of text that, for example, explains the history and countries involved in World War II, these countries will be highlighted on an interactive map, along with markers displaying additional information and images related to the country. In addition to this core feature, the application offers users the option to engage with an AI chatbot where users can ask questions about locations, and have locations from the received responses visualized on the interactive map. A user can for example ask where bananas are grown, and have the countries and areas displayed, along with contextual information from the chatbot.

2.2 Client: Atlas

Atlas is an emerging company based in Oslo, Norway. The company was founded in 2021 under the name Enernite by 4 students attending the Norwegian University of Science and Technology (NTNU) (Moger et al., 2024). Since then, the company changed its name from Enernite to Atlas in 2023 (Atlas, 2023). Atlas has received funding both from governmental organizations, such as Innovation Norway and The Research Council of Norway, and from various international investors that believe in Atlas's vision, including the European Space Agency. In total, the company has raised approximately €2.5 million as of February 2024 (Moger, 2024).

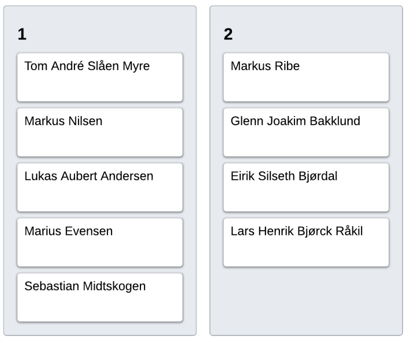
Atlas' main product is their GIS web application, described by them as "The Figma of the GIS-world". This web application aims to revolutionize the way users are able to manipulate and edit maps, by providing an easy to use online tool. Current desktop applications such as QGIS and ArcGIS, while powerful, typically involve complex interfaces that can be cumbersome and challenging for new users, often requiring extensive training to navigate effectively. This gives Atlas the opportunity to address a significant skill gap within the GIS community. As of the writing of this report, Atlas has capitalized on this opportunity by positioning itself within a niche in the market with high potential. Their innovative web application promises ease of use and flexibility, offering a user-friendly alternative to more technically demanding traditional platforms, thus improving accessibility and simplifying the user experience. Atlas' main strength lies within this approach of creating a simple yet powerful web-based GIS tool alternative.

> *There's a revolution happening in business software right now. Notion is changing how we organize, Figma how we design and Slack changed how we communicate. However, maps and geographical information systems (GIS) has remained relatively static. Millions of businesses across the world are using software that were designed for a time when the speed, scale, data and capabilities that businesses require today could not be conceptualized.* (Atlas, 2023)

Our primary contact person within Atlas is Fredrik Moger, CEO and one of the 4 co-founders of Atlas.

2.3 Team structure

As mentioned in previous sections, we are two teams working for the same company on projects that have a lot in common. Given the overlap, it became necessary to explore how we could organize and structure the group as a whole. Our teams, shown in Figure [1], each work on a different project, yet share many common elements, such as client, tech-stack, and goals and ambitions, as elaborated in Section [2.4]. This led us to adopt a collective organizational framework, as can be seen in Figure [2] below. This methodology not only helps coordination, but also encourages cooperation between teams with the aim of enhancing both projects.



1	2
Tom André Slåen Myre	Markus Ribe
Markus Nilsen	Glenn Joakim Bakklund
Lukas Aubert Andersen	Eirik Silseth Bjørdal
Marius Evensen	Lars Henrik Bjørck Råkil
Sebastian Midtskogen	

Figure 1*. Overview of teams, only names*

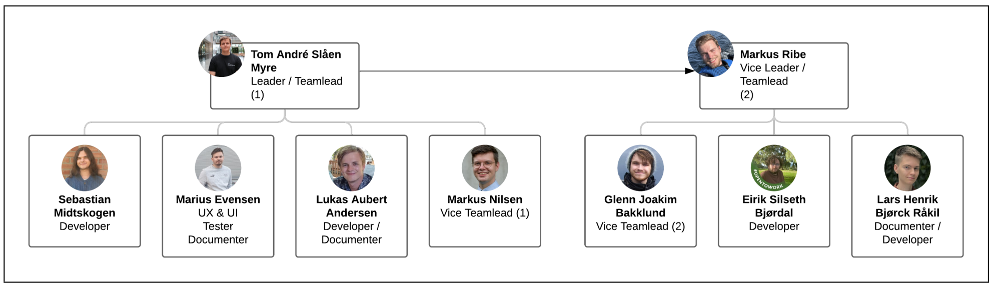


Figure 2*. Organizational chart*

In reality, this combined organizational structure visualizes the assigned roles, as well as what responsibilities within the collective the person has. Table [1] and Table [2] below give a short description of how the roles are defined and the responsibility given to the members of the collective. Each member has, in most cases, more roles, or sub-roles, than given in the organizational chart. The roles given in the organizational chart represent their main tasks. The reason for having more roles than given comes from our agile viewpoint, where we aim to be as flexible as possible and to be able to solve and fix issues as effectively as possible.

Collective level roles:

Role	Is responsible for
Leader	Leading "collective" meetings, or delegate leading in meetings, develop and provide suggestions on management & management structure implementations, and responsible for developing guidelines for the code base, for example branching strategies and how to contribute.
Vice Leader	Taking on leadership responsibilities when the leader is not present, helping the leader fulfill his role, and ensuring that the leader follows a democratic and fair approach to collective decisions.

Table 1*. Descriptions of Collective Roles*

Team level roles:

Role	Is responsible for
Team Leader	Team management, being the Scrum master (read more in Section 3.5.3), and providing guidance when necessary.
Vice Team Leader	Helps the team leader with their tasks, and acts as team leader when the team leader is not present.
Developer	Produces code additions or changes to the software, and enhances the product.
Documenter	Takes notes during meetings, daily stand-ups and more, and makes sure that important verbal instructions or information is written down.
Tester	Uses software testing techniques, such as walk-throughs and peer reviews.
UX/UI	Focuses on the user experience and user interface of the application.

Table 2*. Descriptions of Team Roles*

2.4 Goals and ambitions

Before we start to explore the management and development process of the applications, it is important to have clearly defined goals and ambitions in advance. Hence, having a clear direction in which to work towards, which can hopefully help elevate the final products.

Our overarching goal for the projects is to create GIS applications that improve and simplify elements of existing GIS applications and solutions. The reason for this, as mentioned in Chapter [1], is that GIS applications are often heavy desktop applications that feel outdated and are difficult to learn and master. This overarching goal leads us to the individual goals of the applications.

As previously stated in Section [2.1], Image2Map is intended as an online service to georeference satellite images, maps, etc. The main goal of Image2Map is therefore to act as a tool which both simplifies the georeferencing process over traditional tools, as well as making it accessible to more users by developing it as a web application.

Text2Map is intended as an easy-to-use online tool to visualize data and text as digital maps, also as stated in Section [2.1]. Based on this, the main goal of Text2Map is to be a tool that is easy and effective to use, with minimal preliminary knowledge of GIS tools as possible. As a more concrete goal, the user should be able to provide data, such as a CSV file or text, and get a visualized map of the data in return. In addition, the user should be able to perform queries within the system and have the answers visualized. Both of these tasks should be accomplished utilizing AI, at the request of Atlas.

Academically, our first goal is to learn as much as possible in relation to the separate projects (topics such as GIS, maps, etc.). Our second goal is to complete the projects and the report in a manner that satisfies us, our professors, and external examiners. Our last and arguably most important goal is to learn from the development experience and the final applications; What did we do well, what did we do wrong, and how can we replicate what we did well? Answering these questions will greatly assist us to best prepare for future work within the IT sector, as well as others embarking on similar projects.

With all of these goals in mind, we aim to create services that will be satisfactory to Atlas and Atlas's customers, as well as document any important lessons and findings that may assist others within the GIS and general IT sector.

3 Project Management

In this chapter we will go over our methodology related to how the development of the projects was managed, such as tools for task management, communication, and version control, our usage of the Scrum framework, as well as time estimation and logging.

3.1 Project management tools

Choosing the right tools is essential for proper project management. Expanding within this section, we are going to talk about the various tools we use to manage our project; where we keep track of our tasks, our choice of communication tool, and how we manage and distribute code for the project.

3.1.1 Task management: Trello

Trello is a platform utilized to manage tasks in both projects, which are divided into boards that are heavily based on Kanban, a workflow for assigning tasks with a description and grouping them up in categories (Radigan, 2024). Trello offers collaboration and real-time updates in the Kanban-board, and proved essential for us both in reflection on progression and improvement as well as planning in every phase of the project.

The board is divided into several categories. Tasks that are yet to be done are either in the product backlog or in the sprint backlog, which we will go into further detail later in this chapter. Once someone starts a task, they assign themselves on the card and move it to the "doing" section of the board. Once the task is completed, the card is moved to the "code review" section, where after the completed task has been reviewed, it is moved to "testing". Figures illustrating this can be found in Section [6].

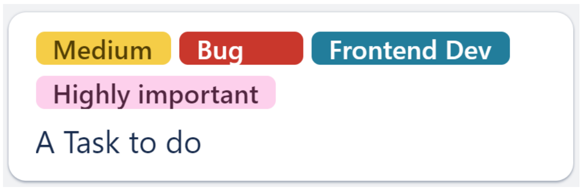


Figure 3*. Example card from the Trello board*

In addition, various cards have varied importance and difficulty. Figure [3] illustrates the different "tags" that a card can utilize; *"Medium"* signifies the estimated difficulty of the task, where *"Easy"* is estimated to be completed in maximum one day, *"Medium"* at worst within the week, and *"Hard"*, which may take more than a week to complete. *"Hard"* tasks may be broken into smaller tasks, where the *"Hard"* card stays as an overarching task to keep track of larger features.

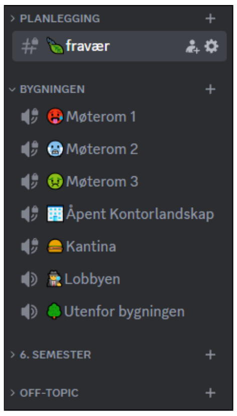
"Bug" is a tag utilized for signifying tasks that are considered bugs, which means that something isn't working as it should, often as a result of new code additions. *"Bug"* tags easily show tasks that require somewhat immediate focus and repair.

"Frontend Dev" is used for showing that a task is centered on work in the front-end section or, by using *"Backend Dev"*, the back-end section of the project. A task can utilize both tags if the task requires making changes in the code base in both sections.

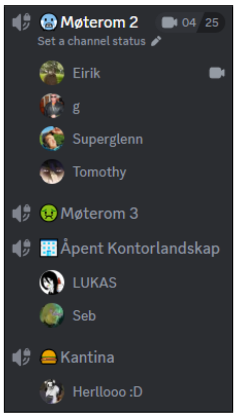
Lastly, *"Highly Important"* signifies perceived importance in regards to completing the project. Additional types of this tag are *"Medium Importance"* or *"Less Important"*. These help us to make sure we focus on the correct tasks so that the main features of the applications are completed. Less important tasks are therefore deprioritized, but may be completed if there is time.

3.1.2 Communication: Discord

Discord is the main platform for communication and the core platform for doing remote work, with key features such as sending messages and IP telephony (Discord, n.d.). It has been heavily used for sending resources to each other, as Discord lets us create several text and voice channels to have different chat rooms for different topics. We tried to simulate an office environment as much as possible by having voice channels such as meeting rooms (Møterom), an open office landscape (Åpent Kontorlandskap) where people work in silence, and a canteen (Kantina) for people to use while having lunch, as can be seen in Figure 4. The main reason for the different rooms has been to be able to contact people as we work, as well as to quickly get an overview of what people are doing. This possibility, along with everyone in the groups already being familiar with Discord, made it easy to choose Discord as our primary communication tool.



(a) Discord channel overview, with categories



(b) Discord in use

Figure 4. *Discord examples*

3.1.3 Version control: GitHub and Git

One of the main benefits of utilizing a version control system, in our case Git through GitHub (see Appendix [A.1]), is the possibility to have different "branches" of code. A repository typically has one "main branch", which serves as the base branch on which all code changes are based. This means that any new "branches" originate from the code on the main branch but are able to be worked on separately; it is then possible to integrate the new branch together, when a code change has been made, with the main branch (or another branch) through a "merge" (GitHub, n.d.-a). In GitHub, we can set rules and requirements that must be met to merge one branch with another. These rules are typically applied to the main branch and can protect and ensure the integrity of the code base from unintended changes.

We took advantage of these rules and requirements in our project's GitHub repositories, Image2Map (Group 21 & The Atlas Repository, 2024) and Text2Map (Group 22 & The Atlas Repository, 2024), to restrict merges and updates to our base branch with the need of a "pull request" with the requirement of needing at least one approved code review from another developer on the team. When the requirement is met, the "pull request" is approved, and the changes can be merged back to the main branch. By this, we ensure that all changes have been peer reviewed and/or tested before becoming an official part of the code base.

3.2 What is Scrum, and why use it?

To understand the Scrum method, and why we utilized it, we first have to understand how Agile and Scrum are different. Drumond (2022) points out that Agile in itself enables the possibility to change course fast by small and frequent releases, or sprints in our case, but lacks the core factor of actually making progress and completing work. Hence it would seem like Agile in itself remains a philosophy, which is most suited as a component to add on to. With this understanding of the uniqueness of both Agile and Scrum, it becomes apparent why Scrum is essential for success in situations like ours (Drumond, 2022).

The essence of Scrum in itself lies in lean thinking and the way knowledge is handled. By lean thinking, we mean the removal of excess and unnecessary fat. In our case, this means removing every component and factor that does not prove useful and focusing on the essentials. Similarly to Agile, we gain knowledge throughout the process, but with Scrum, we make decisions based on observations and the knowledge we have learned. Another term that Drumond introduces that defines Scrum is "heuristic", which means that Scrum is based on continuous learning and adjustments to promote optimal progress (Drumond, 2022).

Now that we understand the core principles of Scrum, it is important to understand the Scrum components, which we have aimed to implement into our own projects. These components include sprint planning, daily stand-up, sprint review, and sprint retrospect with their own unique features and components. These components will be discussed in the chapters within "The sprints" [3.5].

3.2.1 Scrum participants

Scrum usually has three main roles, the product owner, a Scrum master, and the development team (West, 2022). In our case, the product owner will be Atlas, as they set the product vision for the team, as well as track our progress, making sure that we achieve the needs of the product. However, usually the product owner helps to define the project backlog (West, 2022), and this task has become a collaborative effort of the Scrum master and the development team. In both groups, the Scrum master's role is filled by Tom, aided by Markus R. Tom is the collective leader and facilitates the daily Scrum meetings and sprint planning, while also managing any potential obstacles making sure that the team members are unobstructed in their tasks. Finally we have the development team consisting of the rest of the team members, who's role in the Scrum consists of giving their insight into how we can improve the product, helping in sprint planning and goal setting, as well as quality assurance.

3.3 Product backlog

The product backlog is a collection of features or tasks that was set up during the initial planning phase of the project. During the pre-sprint, we dedicated a substantial amount of time to coordinate possible features both projects (more details in Section [3.5.1]). The backlog consists of project cards that can contain several tags, and these cards are explained in detail in Section 3.1.1. By having a product backlog, we can get a good overview of progress and set long-term goals for how the projects can evolve, as well as add new cards as tasks are completed and the applications evolve.

3.4 Sprint backlog

The sprint backlog is a collection of cards and items directly selected from the product backlog. When we place the cards in the sprint backlog, it means that these are the features and ideas that we intend to develop during each sprint. This solves the problem of taking on too many tasks at once, making it seem overwhelming.

At the start of each sprint planning, both teams discuss the current contents of the backlog, as well as adding more cards in sync with our predicted workload. How the sprint backlog was further utilized is discussed in Section [3.5.2].

3.5 The sprints

Throughout both projects, our teams works through several sprints, each planned, executed, and reviewed at its inception. We plan, we execute the plan, review the plan and repeat the process. This process is both unbudgeable process and agile at the same time. The agile part comes from planning and reviewing where we reflect on what we have done and what could improve, whereas the unbudgeable part is reflected in the continual execution of the planing process. We also reflect on; if the progress we have made contributes towards our end goal. By repeating this process, it enables us to keep making progress towards a goal, even if one were to encounter roadblocks or hardships along the way (Rehkopf, 2019). The sprints, which are arguably the essence of project management, are the place where we develop the actual projects. Within the upcoming section, we will discuss what sprints are, as well as what sprints entails.

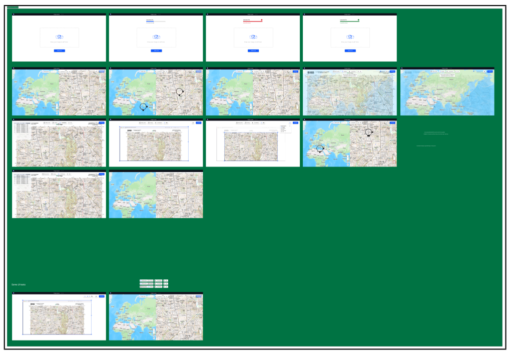
3.5.1 Pre-sprint

Before the pre-sprint took place, we encouraged each member to develop a personal project, without any predetermined ideas or features, before attending an internal hackathon hosted by Atlas at their headquarters in Oslo.

Setting out on these projects was made possible by our mostly confirmed stack components, which were received ahead of the "field trip". This prepared the developers on what to expect from the tech stack and what to learn beforehand, since many of the developers had never coded in React and/or Python before. This improved skills and understanding for those who chose to develop a personal project.

The Pre-Sprint started with a trip to Oslo, to meet our client Atlas, and attend a hackathon there. At this point, we conducted some initial planning for how we were going to structure our two projects, including finalizing the tech stack, which we were planning to utilize for development. This resulted in being the same as the one that Atlas currently utilizes, as planned. Furthermore, we discussed how we were going to manage our two projects, including discussing how we would do our Scrum meetings, setting up our Discord server, and defining the roles in the project.

We learned a lot during the initial pre-sprint; About Atlas as a company, what they do, and what they expected from us in the projects we chose. This was useful information for us as it set a clear definition of expectations from the very beginning.


(a) Image2Map (Full scale, Appendix Figure [B.1])

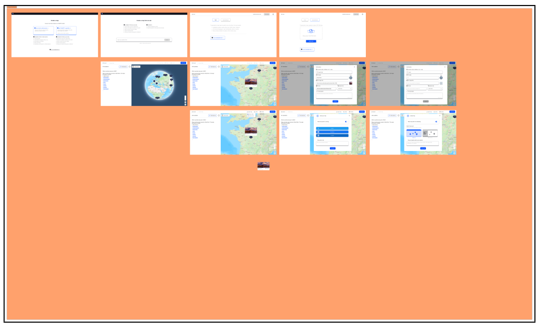

(b) Text2Map (Full scale, Appendix Figure [B.17])

***Figure 5**. Figma overviews*

Furthermore, Atlas had produced a Figma mock-up of each project, shown in the compound Figure [5]. An in-depth look at each page can be found in Appendix [B.1]. These figures visualized Atlas' vision for each project and gave an initial direction for the projects which would evolve during the sprints; this evolution will be explored in detail during later chapters.

In addition, user stories were created as part of our initial planning process for our projects. We designed the user stories around the proposed layout of the applications aiming to create a realistic picture of how the users would navigate the sites. After this, we created an interpreted feature list for each project, utilizing Atlas' Figma mockups as inspiration (see Appendix [B.1]). Using both the mockups and the interpreted feature list, we could create cohesive user stories based on blueprints taught during previous courses at UiA and a video example (Codex Community, 2022, 6:39). In creating these user stories, we followed a simple format. First, we noted the person's "role" who wants or needs a certain "feature". Then, we briefly explained the "reason" behind needing that feature. An example of this can be seen in Appendix [B.3].

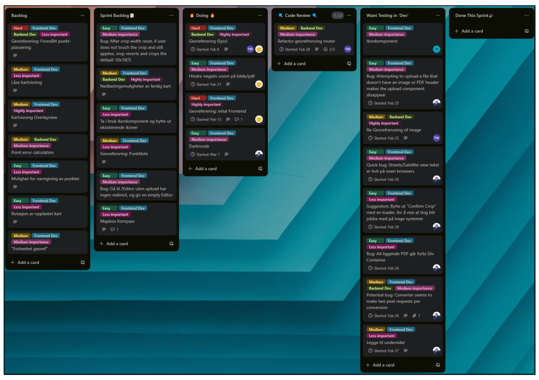
Lastly, while we would normally go through the process of creating initial sketches, wireframes, and mockups for each of the projects. This was not necessary given the Figma mockups Atlas provided. However, we have utilized parts of sketches, wireframes, and mockups when discussing some changes made to UI/UX elements compared to the original mockups (see Appendix [B.4]). These changes will be elaborated on in Section [4.3], and Section [6.2] later in the thesis.

3.5.2 Sprint planning

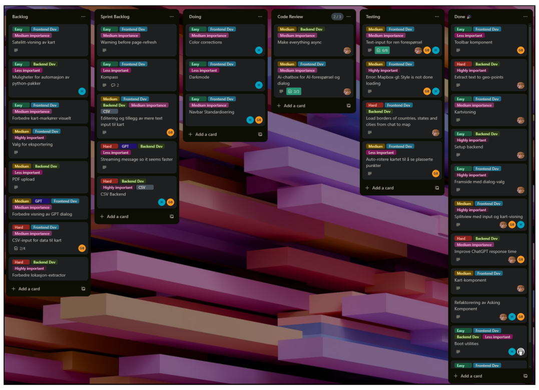
What is prioritized throughout each sprint stems from the repeating sprint planning. During the very first sprint planning, the duration of the sprints was decided to be one week. The crucial part of sprint planning is to create an environment in which all participants are motivated, challenged, and crucially a place where everyone can be successful. The essence of the planning process is to pick the most pressing tasks and predict how many tasks are reasonable to complete during that sprint (West, n.d.).

As the Scrum master and the development team progress through the project, it becomes progressively more important to reflect on past sprint planning meetings. By doing so, both parties can learn, develop and progress as teams and individuals. As we discussed earlier, the ability to learn and adapt is crucial for the success in the use of Scrum and Agile development, hence the importance of reflection.

As discussed in Section [3.1.1], Trello is the management tool of our choice. Rather, where this tool becomes important for sprint planning is the "backlog" and "sprint backlog", where we as a group discuss, reflect, and come up with a plan on what tasks to prioritize and how to tackle them throughout the sprints. In Figure [6] below, you will find two examples of how we set up the Trello boards in the different projects.



(a) *Image2Map Trello board*



(b) *Text2Map Trello board*

***Figure 6**. The projects Trello boards*

In addition to sprint planning, these Trello boards are reviewed and discussed in sprint retrospect, found later in Section [3.5.5]. These boards enables us to visualize the project workflow and highlight what each member is working on. Most critically, it enables us to break down the projects into smaller parts, or cards, which are easier to visualize and start working on. As well as visualizing, these boards help us clarify the importance of each component which leads us towards better prioritization and optimization. These aspects help the teams stay on track to meet goals and deadlines.

3.5.3 Daily-standups

"The daily stand-up is a short, daily meeting to discuss progress and identify blockers" (Radigan, 2019).

One could argue a daily stand-up meeting is like a football teams' "team talk" before a match, where they discuss plans and strategies, as well as potential issues. A daily stand-up meeting within a development team is also just that. In our case, we discuss what we did, what we are going to do, and lastly what potential issues we foresee. Regarding the participants, ideally the product owner, developers, and scrum master should participate in these meetings. In our case, it proved most efficient including solely developers and Scrum master for most meetings , with our product owner instead being included on a larger meeting at the end of each week.

Our standardized set of questions for daily stand-ups was: What did you do yesterday? What are you going to do today? Do you anticipate any issues?

These questions enabled us to keep in touch with all developers, where we quickly located what we had been doing and what we had to do, in order to progress towards our common goal. The last question proved to be essential to ensure proper progress throughout the project. By asking this question every morning during daily stand-up, it enabled us to locate and deal with issues on a very efficient scale.

3.5.4 Meeting with client (Atlas)

In our case, meeting with the client primarily resolves around showcasing current progression to Atlas. The ideal outcome of these meetings is to confirm with the client that we are on the right track, and receive feedback on what to improve during the next sprint.

"The purpose of the sprint review is to inspect the outcome of the sprint and determine future adaptions. The scrum team presents the results of their work to key stakeholders and progress toward the product goal discussed" (scrum.org, n.d.).

In our case, we separated sprint review and sprint retrospect. There were some components that originally remained in sprint review, but worked better for us in sprint retrospect. The first component is the review of the "Trello" board; normally the product owner (Atlas) is present in this review. In our case, it worked out better to move it to sprint retrospect where Atlas is not present. This component was removed from the sprint review to maximize the actual product feedback, whereas the feedback on how we managed the project proved to be less important. By doing this, we could optimize the development process and deliver products that our client would be satisfied with (Radigan, n.d.).

3.5.5 Sprint retrospect

The sprint retrospective concludes the sprint, and is an event for reflection and learning, the purpose of which is to create a plan that increases quality and effectiveness (“Scrum Guide | Scrum Guides”, n.d.).

The idea behind a sprint retrospective is to reflect on how the sprint went, both positive and negative (“Scrum Guide | Scrum Guides”, n.d.). In our sprint retrospectives, we included reflections on ourselves as individuals, our processes, and our tools. It was important to assess whether we were using the right tools for the task and, if not, to find a solution. We also reflected on our meeting structure and how we conducted our sprints to uncover potential inefficiencies and issues.

As mentioned in Section [3.5.4], we decided to move the review and discussion of the sprint backlog to the sprint retrospective meeting. This meeting took place every week after our meeting with Atlas, at the end of our sprints, and smoothly transitioned into sprint planning. Moving the sprint backlog discussion to the retrospective meeting allowed us to more clearly determine future tasks and make the process more efficient. During the sprint backlog discussion, we split into the two respective teams to focus on the respective project’s backlogs.

3.5.6 Boilerplate

For us to understand the boilerplate concept, we have to understand "Scrum of Scrums". "The Scrum of Scrum has a chief purpose: to synchronize the work coming from different teams who are working on various parts of the same project" (scrumexpert, 2020). In other words, Scrum of Scrums is a meeting in which selected members of the teams, the product owner, and the Scrum master participate. The goal is to plan ahead and manage the work of different teams in the most efficient way. Boilerplate, on the other hand, is a set of questions that could be included in a meeting like Scrum of Scrums (scrumexpert, 2020). Such questions could be:

- What has your team accomplished since our last meeting?
- What problems occurred, if any, that negatively affected your team?
- What does your team want to accomplish before we meet again?
- What output from your team in future sprints, do you see as possibly interfering with the work of other teams?
- Does your team see any interference problems coming from the work of other teams?

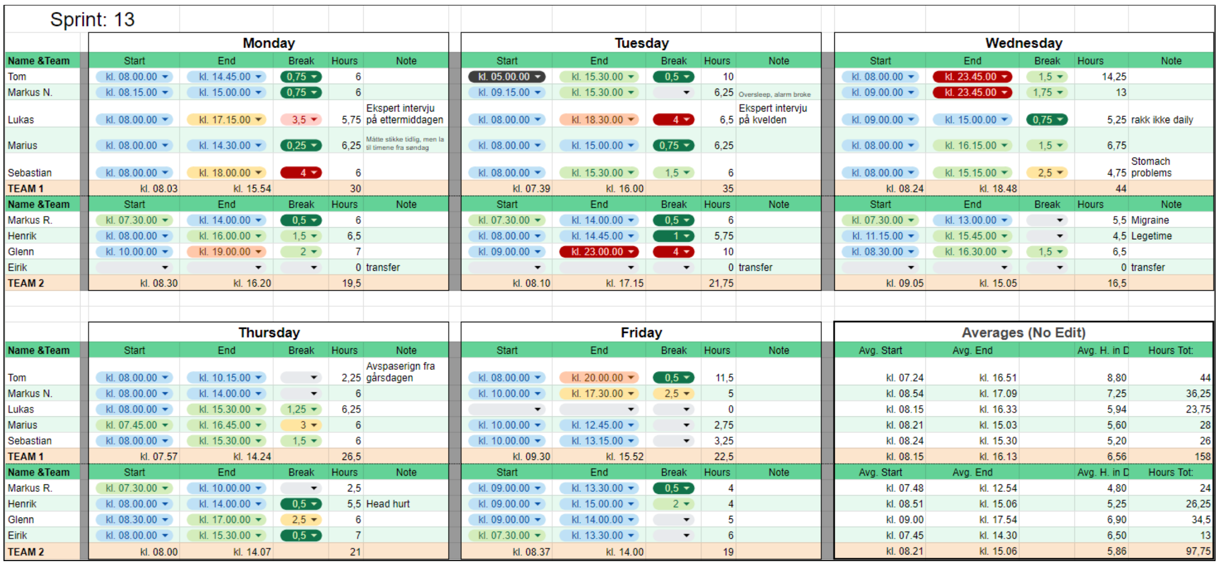
(scrumexpert, 2020)

With our understanding of Scrum of Scrums and boilerplate questions, we predicted that such questions would prove beneficial in terms of managing the two teams. At first we tried with the questions listed above, by asking them while sitting together, which created some disorder. Most of the members lacked context, which resulted in poor performance with regard to the results of the questions. To improve performance, we tried Google Forms, as well as optimizing the questions. The idea behind Google Forms was to improve accessibility and efficiency, as team members could fill them out when they had time. Even with these improvements, we experienced poor performance. Most critically, these questions were too similar to our sprint retrospective and sprint planning. As mentioned, sprint retrospect and sprint planning take place on the same day, which proved the boilerplate questions to be insignificant. This experience resulted in increased clarity in our project management and increased understanding of why we follow through with sprint retrospect and sprint planning.

3.6 Time estimation

Time estimation is a common practice within tools such as Trello, where teams estimate how much work they can finish in a sprint. Normally, each task on the board should have its own estimated time, showing how long it is expected to take to finish. We have instead utilized a timetable where we explain why we have spent the time we did, and on what tasks.

Below, as seen in Figure [7], is an illustration of our timetable for sprint 13, which highlights the start and end times, as well as the breaks for each person each day. The spreadsheet then calculates the average hours worked in each sprint. This number helps us gauge how much effort we are putting in and how hard the tasks are. If we spend the same amount of time in two sprints but finish a different amount of tasks, it suggests that the tasks in the "less productive" sprint were harder.


Sprint: 13

	Monday					Tuesday					Wednesday				
Name &Team	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note
Tom	kl. 08.00.00	kl. 14.45.00	0,75	6		kl. 05.00.00	kl. 15.30.00	0,5	10		kl. 08.00.00	kl. 23.45.00	1,5	14,25	
Markus N.	kl. 08.15.00	kl. 15.00.00	0,75	6		kl. 09.15.00	kl. 15.30.00		6,25	Oversleep, alarm broke	kl. 09.00.00	kl. 23.45.00	1,75	13	
Lukas	kl. 08.00.00	kl. 17.15.00	3,5	5,75	Ekspert intervju på ettermiddagen	kl. 08.00.00	kl. 18.30.00	4	6,5	Ekspert intervju på kvelden	kl. 09.00.00	kl. 15.00.00	0,75	5,25	rakk ikke daily
Marius	kl. 08.00.00	kl. 14.30.00	0,25	6,25	Måtte stikke tidlig, men la til timene fra søndag	kl. 08.00.00	kl. 15.00.00	0,75	6,25		kl. 08.00.00	kl. 16.15.00	1,5	6,75	
Sebastian	kl. 08.00.00	kl. 18.00.00	4	6		kl. 08.00.00	kl. 15.30.00	1,5	6		kl. 08.00.00	kl. 15.15.00	2,5	4,75	Stomach problems
TEAM 1	kl. 08.03	kl. 15.54		30		kl. 07.39	kl. 16.00		35		kl. 08.24	kl. 18.48		44	
Name &Team	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note
Markus R.	kl. 07.30.00	kl. 14.00.00	0,5	6		kl. 07.30.00	kl. 14.00.00	0,5	6		kl. 07.30.00	kl. 13.00.00		5,5	Migraine
Henrik	kl. 08.00.00	kl. 16.00.00	1,5	6,5		kl. 08.00.00	kl. 14.45.00	1	5,75		kl. 11.15.00	kl. 15.45.00		4,5	Legetime
Glenn	kl. 10.00.00	kl. 19.00.00	2	7		kl. 09.00.00	kl. 23.00.00	4	10		kl. 08.30.00	kl. 16.30.00	1,5	6,5	
Eirik				0	transfer				0	transfer				0	transfer
TEAM 2	kl. 08.30	kl. 16.20		19,5		kl. 08.10	kl. 17.15		21,75		kl. 09.05	kl. 15.05		16,5	

	Thursday					Friday					Averages (No Edit)				
Name &Team	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note	Avg. Start	Avg. End		Avg. H. in D	Hours Tot:
Tom	kl. 08.00.00	kl. 10.15.00		2,25	Avspaserign fra gårsdagen	kl. 08.00.00	kl. 20.00.00	0,5	11,5		kl. 07.24	kl. 16.51		8,80	44
Markus N.	kl. 08.00.00	kl. 14.00.00		6		kl. 10.00.00	kl. 17.30.00	2,5	5		kl. 08.54	kl. 17.09		7,25	36,25
Lukas	kl. 08.00.00	kl. 15.30.00	1,25	6,25					0		kl. 08.15	kl. 16.33		5,94	23,75
Marius	kl. 07.45.00	kl. 16.45.00	3	6		kl. 10.00.00	kl. 12.45.00		2,75		kl. 08.21	kl. 15.03		5,60	28
Sebastian	kl. 08.00.00	kl. 15.30.00	1,5	6		kl. 10.00.00	kl. 13.15.00		3,25		kl. 08.24	kl. 15.30		5,20	26
TEAM 1	kl. 07.57	kl. 14.24		26,5		kl. 09.30	kl. 15.52		22,5		kl. 08.15	kl. 16.13		6,56	158
Name &Team	Start	End	Break	Hours	Note	Start	End	Break	Hours	Note	Avg. Start	Avg. End		Avg. H. in D	Hours Tot:
Markus R.	kl. 07.30.00	kl. 10.00.00		2,5		kl. 09.00.00	kl. 13.30.00	0,5	4		kl. 07.48	kl. 12.54		4,80	24
Henrik	kl. 08.00.00	kl. 14.00.00	0,5	5,5	Head hurt	kl. 09.00.00	kl. 15.00.00	2	4		kl. 08.51	kl. 15.06		5,25	26,25
Glenn	kl. 08.30.00	kl. 17.00.00	2,5	6		kl. 09.00.00	kl. 14.00.00		5		kl. 09.00	kl. 17.54		6,90	34,5
Eirik	kl. 08.00.00	kl. 15.30.00	0,5	7		kl. 07.30.00	kl. 13.30.00		6		kl. 07.45	kl. 14.30		6,50	13
TEAM 2	kl. 08.00	kl. 14.07		21		kl. 08.37	kl. 14.00		19		kl. 08.21	kl. 15.06		5,86	97,75

***Figure 7**. Timetable for Sprint 13*

4 Project Implementation

As mentioned, the central goal of these two geospatial projects is to significantly improve workflows for GIS users, spanning from single individuals to large businesses. This chapter explores the process of implementing our two projects; going from concept to functional program, it covers the steps and decisions that were made, the challenges we encountered, and solutions we found throughout the development process.

4.1 System requirements

To maximize accessibility, we have designed our software with minimal system requirements. The guiding principle is that any machine capable of running a modern web browser possesses sufficient computational power to execute our tools. Furthermore, we have optimized the interface for optimal usability on any device with a large display.

4.2 System definitions

This section covers the criteria that specify what our two products have to achieve in order to meet the requirements set for Minimum Viable Product (MVP). An MVP represents the most pared-down version of a product that can still be released to early users and therefore may not include all the functions we were aiming for in the final products.

The MVP for Image2Map requires the ability for users to be able to: Upload a picture (Image or PDF) to the website, then easily compare and georeference the uploaded image with a live map, and lastly download the georeferenced picture in some format.

The MVP for Text2Map requires the ability for users to be able to: Input text or a question to the application, then have relevant locations be extracted and displayed on a map, and the ability to download the displayed data or save it in some way.

4.2.1 Prioritization of system criteria

The initial focus of our development lies in the development of the core functionality of the two products we are producing, which is image georeferencing for Image2Map and text-to-map generation for Text2Map. This strategic decision aligns with the MVP concept, ensuring that we deliver a baseline product that directly addresses our primary goals. Although, there is ample potential for future feature expansion, prioritizing the core functionalities could drive user adoption and establish a strong foundation for future iterations.

4.3 Graphical design principles

In developing the user interface for Image2Map and Text2Map, we adhered to various established graphical design principles to ensure that our tools are functional, user-friendly, and engaging. We made sure to maintain simplicity, having a consistent graphical layout, and focusing on creating visually consistent products. In both tools, it was important to use simple color schemes with uniform buttons and icons (Benyon, 2019, p. 291-294, 308).

To break down our choices further down, we made sure to have a clear hierarchy in our tools in order to guide the user's eyes to where we want them. For example, it was important to make the map the central focus with toolbars or legends positioned as secondary visual elements, as well as the use of alignment and repetition to ensure that the tools are as easy and intuitive as possible for the user (Benyon, 2019, p. 296-298, 302, 596).

4.3.1 Design inspiration

The user interface for our two geospatial tools draws inspiration from the design principles exemplified by the Atlas.co website. Atlas.co leverages a clean and intuitive interface that prioritizes user-friendliness.

4.4 User interface

The designing of a user-friendly and intuitive interface is incredibly important for creating successful web tools. This chapter outlines key principles, layout decisions, and special features that guided the development of the interfaces in order to ensure that we meet our project objectives.

4.4.1 Layout and Navigation

We chose to adopt relatively minimalist layouts for both Image2Map and Text2Map in order to prioritize the core functions that we are developing for both projects. We also drew some inspiration from the layout of Atlas's existing web application (app.atlas.co). For instance, Image2Map replicated the Atlas navbar almost one-to-one in order to create higher cohesion between Image2Map and the Atlas App, thereby creating familiarity for users when switching between the two (*After we finished developing our tools, there have since been some major changes on the Atlas Application, meaning the aspects mentioned in this paragraph might not be accurate anymore, but they were at the time of development and initial writing of this. For example, the aforementioned navbar being replaced by a sidebar.*). We also based our layouts heavily on Figma models (Appendix [B.1]) provided to us, courtesy of Atlas CEO Fredrik Moger.

With our minimalist and intuitive layout, we attempted to lay the groundwork for straightforward, uncomplicated navigation across both tools. The tasks the user is supposed to be able to perform using our tools are also quite limited by design, further aligning with making sure that it is easy to navigate the tools.

4.5 System development life cycle

In this section, we will discuss the system development life cycle, which entails details on how we plan to succeed with both projects. We have divided our life cycle into a set of stages: Where first stage is the pre-planning, second stage is the development, and the last stage is deployment and publishing.

A good start for any big software project is to strategize. In our case, this involved creating a plan that made room for agile thinking, which was done in the pre-sprint

[3.5.1]. This ensured that we had a structure to follow, which gave us a very general idea of what to expect in the future. This was important as it ensured that we were all on the same page when we started developing in the second stage.

The second stage was the longest stage in our development life cycle, as it is where we developed our software. This stage, due to agile thinking, is broken down into smaller, iterative loops. In our case, these loops consisted of Scrum sprint, plan, do, review, reflect, and repeat (seen in next Section [4.5.1]). As previously stated, this is done multiple times throughout the stage.

Transitioning to the last stage is relatively seamless, as the project develops and gradually meets the requirements for deployment and initial release or pre-release, while sprints continue in the background. The requirements for the last stage transposition to start are that the projects have implemented an MVP with its core user workflow established.

4.5.1 Key Elements of Our Scrum Implementation

Our Scrum implementation entails a set of key components:
Sprint Planning: At the end of each Sprint, we conduct a planning meeting to define the Sprint backlog (a prioritized list of tasks) and establish the Sprint goal (as explained in Section [3.5.2]).
Daily Standups: We held brief daily meetings to synchronize progress, identify any blockers, and align the team towards the Sprint goal (as explained in Section [3.5.3]) (see Appendix [C] for notes).
Sprint Review: At the end of each Sprint, we reviewed the completed work, gathered feedback, and demonstrated the functionality to stakeholders (as explained in Section [3.5.5]).
Sprint Retrospective: We conducted retrospective meetings to reflect on the Sprint process, identify areas for improvement, and adapt our practices continuously (as explained in Section [3.5.5]) (see Appendix [D] for notes).

4.5.2 Advantages of Scrum for Our Project

Scrum creates flexibility by accommodating evolving requirements and user feedback throughout the development process. Additionally, the daily meetings facilitate constant communication, providing collaboration and transparency. Lastly, the focus on working software in each sprint provides tangible progress and allows for quick course correction if needed, providing early value delivery.

4.6 System architecture

Our web applications both employ a well-structured front- and back-end architecture, ensuring a clear separation between data processing and the user interface. This is also beneficial for our development process, as it is easier for the developers to work on different things at the same time without having to worry about fixing frequent merging issues. We will continue this section by looking at the architecture of Image2Map, followed by Text2Map.

4.6.1 Image2Map

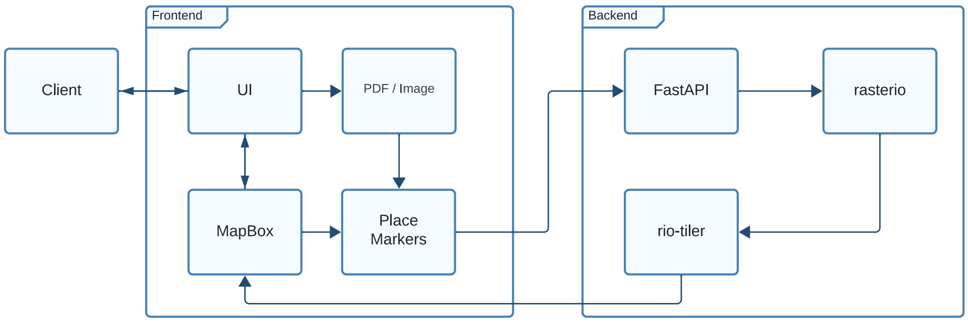


Figure 8*. Initial draft of Image2Map's system architecture*

Figure [8] is a representative system diagram that was developed during the pre-sprint hackathon (mentioned in Section [3.5.1]). In this architecture, the back-end is a self-contained unit responsible for the incoming requests from the front-end, which then processes and delivers the results back. The front-end user interface (UI) manages user interaction, including the processing of new Image/PDF through a sub unit that checks what the image file type is, before making a request to the back-end for final processing and necessary converting. The front-end also takes the processed information from the back-end and visually displays it for the client/user.

This architecture, as well as the initial backlog, served as a useful initial starting point for our development process. However, during our sprints, we discovered in sprint three that Image2Map needed data storage between requests. A change in the system architecture proved necessary in order to implement this. This process started to be implemented by the end of sprint three by pull-request #29 (Group 21 & The Atlas Repository, 2024).

The new architecture can be seen in Figure [9]. As this was a redesign and developed further into the project's life-cycle, the figure has some level of added precision and more outlined core components in the diagram. The new outer Heroku layer will be explored in more detail in Section [4.10.2]. The external and semi-internal details, described in the figure as Add-ons, are outlined in Appendix [A.4.7] and [A.4.8]. These are third-party services that were linked with their respective connections.

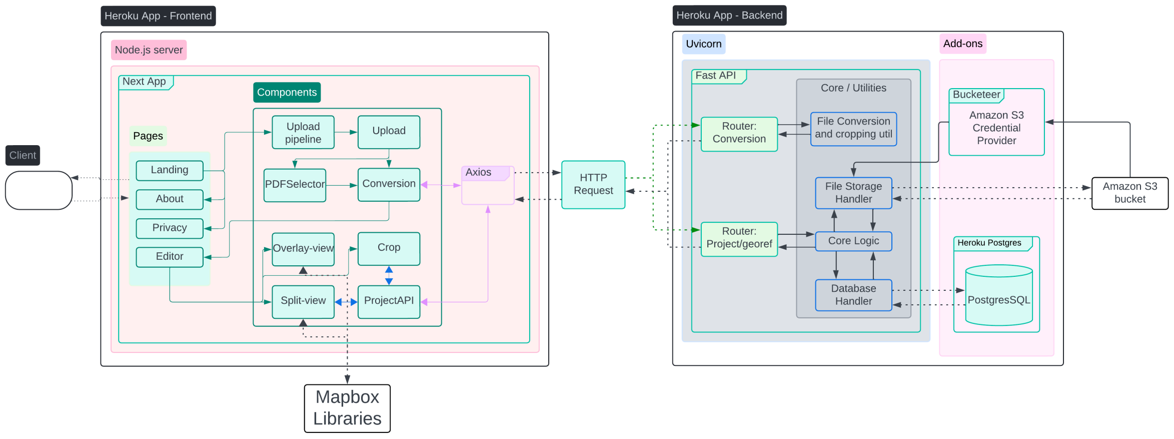


***Figure 9**. Final Image2Map system architecture*

Another benefit of the new architecture, Figure [9], is how it clearly defines all pages in the Next app front-end. It also highlights how the application navigates and utilizes different components, how these components are dependent on each other, and which components communicate with external sources or our FastAPI back-end. Overall, this new system architecture model is more aligned with the current and final state of the system, and display in detail how the project is setup as a whole, including all its integral parts.

4.6.2 Text2Map

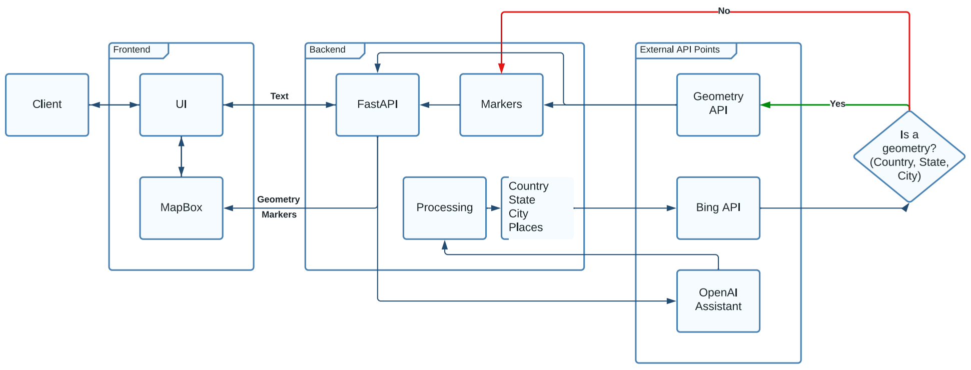


***Figure 10**. Final Text2Map system architecture*

Figure [10] depicts the final architecture of the Text2Map application, as well as highlighting how the different parts and processes are connected. The figure also illustrates the flow of connections through the application. Similarly to the Image2Map application, the rough structure of the architecture was developed during the pre-sprint planning, and was later updated throughout the development process as different requirements and limitations were made apparent. As mentioned in the previous Section [2.1], the indented functionality of the Text2Map application is to visualize the placement of locations around the world through text or data that a user inputs. To accomplish this, the application is required to process the user input in an efficient and consistent way, and gather the necessary geodata for displaying the requested information. This is done through communication with external API's as seen in the Figure [10]. These technologies and processes will be explored in further detail throughout the chapter.

4.7 Back-end

Now that we have gotten a general understanding of how both projects are built up by looking at their system architecture, we will in this section delve deeper into the technicalities of the back-ends, starting with the similarities between Image2Map and Text2Map.

Both projects utilize the Python programming language as the foundation for the back-ends. Python is known for its simplicity, readability, and extensive libraries (Python Software Foundation, 2023), which makes it suitable for various applications, such as data analysis, machine learning, and working with geospatial data. Due to the large amount of third-party packages available (such as one for interacting with the OpenAI API for example), it became possible for both teams to develop prototypes quickly, which then slowly developed into streamlined and efficient back-ends.

In order to utilize the back-end as an API, it proved necessary to operate it as a server. We chose to utilize Uvicorn (defined further in Appendix [A.3.2.3]), which is a server interface for Python that implements an important specification named ASGI (Asynchronous Server Gateway Interface). By implementing ASGI, we can run an application on the server asynchronously ("Uvicorn", n.d.), allowing us to receive multiple requests simultaneously. This in turn will speed up the processing of a request and will provide a better safety net in case of a slow process.

For interaction across the back-end and front-end, both projects utilize the FastAPI framework (defined further in Appendix [A.3.1]). Utilizing FastAPI, the individual projects can create specialized "endpoints", which acts as locations the front-end can request, either with or without data, and get something in return. In other words, the API is responsible for taking requests from the front-end and routing them to code in the back-end. An example can be Text2Map's "/newChat" endpoint, which instructs the back-end to send a message to a custom ChatGPT assistant, which then gets a JSON response in return and passes it back through the API to the front-end.

Another common utility is Docker, which is utilized for containerized deployment and service orchestration. This ensures that all necessary components run smoothly together. Docker containers isolate the application, making it easier to deploy in different environments without conflicts (Docker, 2023). Additionally, Docker assists in the orchestration of multiple containers, which can be essential for running complex systems composed of multiple services (Docker, 2023).

Now that we have had a quick look at the common technologies of the back-ends, let us delve deeper into the specifics of each project, starting with Image2Map.

4.7.1 Image2Map

The Image2Map back-end is responsible for managing project data, storing and interacting with files, georeferencing processes, and general image processing. However, as mentioned in Section [4.6.1], it was originally only intended to request, process, and respond. Due to the necessity of handling and storing files in the back-end, the architecture was redesigned to store and keep track of files, as well as keeping track of and storing data related to user projects.

This means that the back-end is composed of several modules that utilize various Python libraries and packages. The arguably most important module, the "core" module, is responsible for managing the projects, the GCPs of the projects, and the georeferencing process itself. This module is heavily dependent on two external packages. The first package is named *rasterio* (defined further in Appendix [A.3.2.4]), which is a package specifically made for working with geospatial raster data and is the package that allows us to georeference an image using 3 or more GCPs. The second package is named *rio_tiler* (Appendix [A.3.2.7]), and is utilized to split georeferenced images into image tiles. The tiles are then returned to the front-end to be displayed within the Overlay-section of the application.

Additionally, this core module also interacts with other modules for file storage and data storage. These modules can be seen as "connecting modules", which means that they are used to allow storing data or files on various services. For example, the back-end, as it is currently, allows for a local database for projects and local storage for files, as well as third-party services such as *PostgreSQL* (Appendix [A.4.8]) for data and *Amazon S3 Buckets* (Appendix [A.4.7]) for files, or any combination of these.

Lastly, another important module is used to convert and modify files. This module mainly provides functions for converting PDF files and various image types to PNG files, as well as for cropping the converted PNG files. This module mostly utilizes two packages; Pillow (Appendix [A.3.2.2]), which is used for converting and modifying image files, and pdf2image (Appendix [A.3.2.6]), which is utilized to convert a page from a PDF file to an image. These are used in preparation for the georeferencing process, as the rest of the back-end expects to only interact with PNG files pre-reference and TIFF files post-reference.

4.7.2 Text2Map

The Text2Map back-end depends on a small amount of essential components to run. One of these is the OpenAI Assistant called ChatGPT (defined in Appendix [A.4.4]), which the application is currently heavily dependent on. Text2Map currently uses two custom OpenAI Assistants, one for answering questions and extracting mentioned locations, and another that only extracts locations. The custom assistants were created to maximize consistency and reliability in the responses they provide, which are returned back to the back-end as formatted JSON files containing the necessary information and the locations.

The location extraction process was previously left to a Python add-on named Spacy. However, this solution was found to be inconsistent at times, in addition to being a large package that made the file size of the back-end application noticeably larger. Spacy was therefore dropped in favor of modifying the existing assistant to replace this functionality, as well as creating a secondary assistant with the sole purpose of returning location data in a specific format.

Another essential component on which the application relies is the Bing Maps REST Services (defined in Appendix [A.4.5]), which is used to translate text or addresses into coordinates. By converting the locations to coordinates, it was possible to place exact markers on the map. Similarly, the application also utilizes a custom geoboundaries API that delivers the boundaries, or outlines, of a given location and is used to better highlight and display the boundaries of locations on the map. More about this API can be read in Appendix [A.4.2].

4.8 Front-end

In this front-end section, we will discuss the solutions we chose to implement for both projects, looking at the functionalities and extensions, including their connected components.

4.8.1 Image2Map

The front-end provides a user-friendly interface for interacting with our georeferencing system. It lets users easily upload an image or PDF to the app, thanks to a custom component created to select a specific page in a PDF, that uses **react-pdf** (see Appendix [A.2.6]). In the main part of the application provide interfaces for different toolbars, live maps, and a viewer of the uploaded image or PDF. The team has also developed an intuitive way to accurately place points on both the map and image/PDF.

We rely on few, but important, extensions to make our interface more intuitive and reliable. We utilize Mapbox to supply the interactive map that is displayed on the interface. The interactive map is utilized to manipulate and show your georeferenced image or PDF. These images and PDFs can also be cropped using the react-image-crop extension. Additionally, since the PDF is converted to an image when uploaded, we only require this extension for this specific task. (Example Figure illustration [11])

We also wanted our system to be as modular as possible, letting users manage the area and/or size of some of our popup windows, such as the coordinate list. We make this possible by utilizing a mix of the allotment and react-draggable packages for React (Appendix [A.2.5] & [A.2.7]).

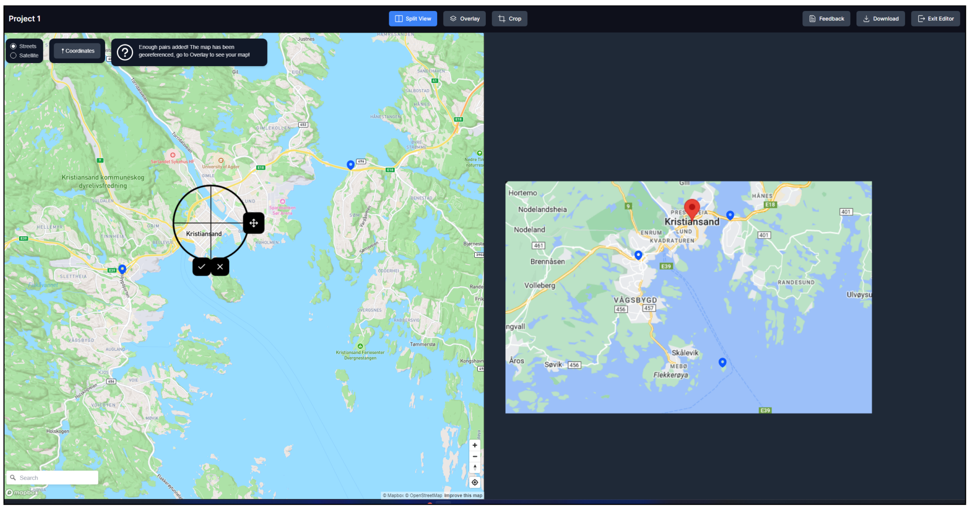
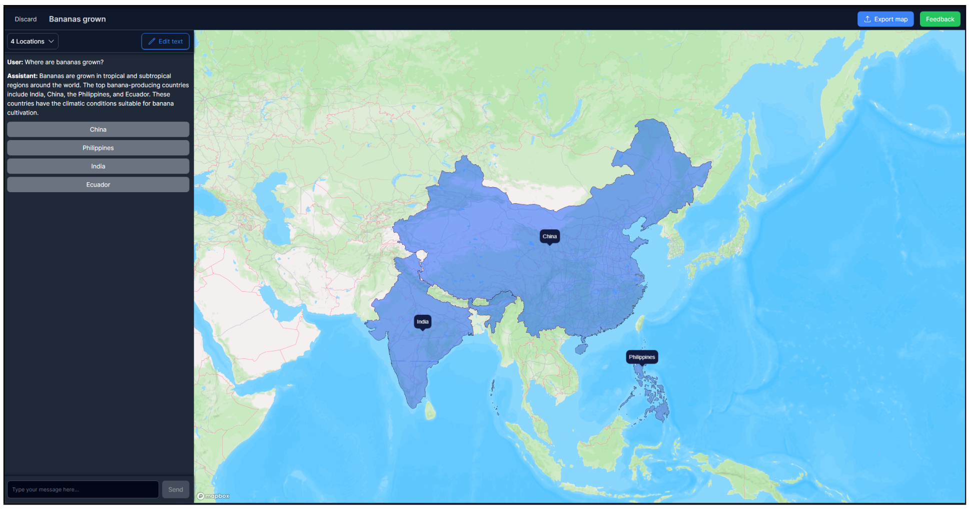


Figure 11. *Screen with uploaded photo and some markers placed*

4.8.2 Text2Map

The Text2Map interface is designed to be easily understandable. It has interfaces ranging from our initial view, which is a page that lets users decide whether they want to speak with an assistant or whether they want to input data/text. From this point, users are presented our main interfaces, which include a chat view on the left and an interactive map on the right (An example figure is illustrated at [12]).

Our front-end relies on the same MapBox dependencies as Image2Map, as well as **allotment**, but does not depend on additional front-end dependencies.



User: Where are bananas grown?

Assistant: Bananas are grown in tropical and subtropical regions around the world. The top banana-producing countries include India, China, the Philippines, and Ecuador. These countries have the climatic conditions suitable for banana cultivation.

Figure 12. *Screen with generated map and some text information*

4.9 API

Application Programming Interfaces (APIs) serve as foundational components in modern software architecture. APIs function as structured communication contracts, enabling disparate software components and services to seamlessly exchange data and functionalities (Appendix [A.3.1]). This modular approach offers several benefits within the context of software development and research. In the next sections, we will discuss what APIs entail and our choice in terms of implementing APIs.

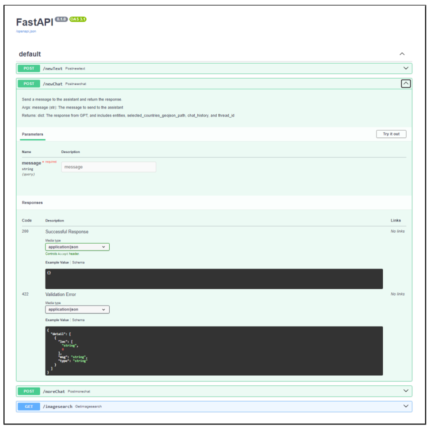
4.9.1 Encapsulation

Encapsulation is a big reason as to why we decided to develop using APIs. By encapsulating functions behind a set of interface specifications, APIs allow us to treat components or services as black boxes, which enhances re-usability in systems. It also makes individual functions much more scaleable, as the back-end can be independently modified from our front-end as long as the API contract remains the same (Tuychiev, 2024). APIs provide a layer of abstraction over implementation details. Developers can consume services offered by existing APIs without having to understand their underlying complexities.

4.9.2 FastAPI & Performance

FastAPI is a modern web framework tailor-made for the creation of APIs in Python and is built with performance in mind (FastAPI, 2024). It leverages asynchronous models and the efficiency of tools such as Starlette and Pydantic. These are both tools for building asynchronous web services and the most widely used data validation tool for Python (Appendix [A.3.1]).

This makes the performance comparable to frameworks like Node.js and Go. It is also user-friendly as its emphasis is on developer experience, where it aims to minimize the time required to build and deploy APIs. Additionally, the API is very robust and easily maintainable as it has built-in type hints and automatic data validation, which contribute to reducing overall errors and maintaining code integrity.


FastAPI 0.1.0 OAS 3.1
/openapi.json

default

POST /newText Postnewtext

POST /newChat Postnewchat

Send a message to the assistant and return the response.

Args: message (str): The message to send to the assistant

Returns: dict: The response from GPT, and includes entities, selected_countries_geojson_path, chat_history, and thread_id

Parameters — Try it out

Name	Description
message * required string (query)	message

Responses

Code	Description	Links
200	Successful Response Media type: application/json Controls Accept header. Example Value | Schema `{}`	No links
422	Validation Error Media type: application/json Example Value | Schema	No links

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

POST /moreChat Postmorechat

GET /imagesearch Getimagesearch

Figure 13. Documentation Page for Text2Map with one of the drop down opened

Additionally, the FastAPI framework has the ability to automatically create a documentation page (Figure [13]) for all API points present in the back-end, through the use of an included web user interface called Swagger UI (Ramírez, n.d.).

These are some of the reasons we chose to write our whole back-end in FastAPI. More importantly, these are also the same libraries that Atlas uses, hence why it was preferred by both teams. Using something familiar to Atlas could prove beneficial in the long run.

4.9.3 External APIs

Text2Map relied on external APIs outside of the application in order to function, one of these is the Geo-coding API that Bing offers [A.4.5]. We utilized this to acquire coordinates and additional information on search locations. As an example, if the text "New York" is provided, the Bing API will return the point coordinates for New York in the United States, in addition to defining values to determine other aspects of the location. This API is crucial, as we need this information for marking the point on the generated map, in addition to aid in an additional process. We also have a custom built external API called GeoB-API. This API works by taking in a country's ISO3 code by itself to return the boundaries of a country. Alternatively, the API can be passed a State or City, along with the country's ISO3 code, and return the boundaries for these as well. All boundaries are returned as a geoJson and the boundaries are created and maintained by geoboundaries.org.

4.10 Continuous integration and delivery

Adding on to our understanding of how the projects are set up, we are going to discuss how we applied continuous integration and delivery in the following section. The term continuous integration, also known as CI, is a development practice that, in essence, is rapid updates of code to a central branch or branches in a shared source code repository. With each of these updates, an automatic process of automated tests is started; furthermore, kicking off the build process (GitLab, 2024b). Using CI increases code quality; the scope of quality is explored in further detail in Chapter [5.2.3].

Continuous delivery, or CD, is a practice that builds on CI in the practical term. It continues after CI is completed, where it implements automation in the infrastructure provisioning and application release process. Using CD, the software is built in a way that enables deployment to production at any time. With this in mind, implementing continuous deployment would be ideal, which is the automated process of deploying applications (GitLab, 2024b).

Encompassing CI, CD and continuous deployment in a streamlined process, utilizing DevOps or a site reliability engineering approach, we have the concept of a CI/CD pipeline (GitLab, 2024a). In the next subsection, we are going to explore how we have integrated these into our projects.

4.10.1 Use of pipelines in the projects

Both projects implemented the practice of CI/CD pipelines through use of GitHub Actions, also known as actions, which is a platform for continuous CI/CD that can be set up to be a pipeline or just CI/CD. Expanding on this fact, it enables the automation of other repository processes (GitHub, 2024). Our reason for choosing actions is that we use GitHub as our cloud source control (Section [3.1.3]), with an action's features covering our need in terms of automation and deployment. The option of using a third party tool, with less integration to the platform, was not desirable.

Using actions, we have been able to implement our pipeline; from code checks, to test-building the application, and then building the application for production, before finally deploying. In the following sections, we will explore the action(s) and highlight the small differences in the action Image2Map uses contrary to Text2Map's action, hence their similarities.

The action start parameters are set with triggers that comply with CI / CD. After defining the triggers, we define the jobs, which are the work, or rather the pipeline itself. Furthermore, each job is divided into steps. A step is generally where stages of the CI/CD process are defined, for example, a build test. Everything related to this stage would either be one or two steps.

Given the projects Mono-repository approach, which means that the projects have both the front-end and back-end in the same repository, the Action is set up with two jobs; one for each of the apps. These are run in parallel and do not interfere with the success of each other.

In both projects' actions, the jobs start by checking out the repository from the state in which the parameters were triggered. The next step is the creation of an ".env". This file contains certain values that the projects need to run. These values are stored in GitHub secrets, where actions retrieve and utilize these securely. An example of a value is the access token for MapBox used in the front-end. In the step of creating the ".env" file, there are project- and app-specific values which are securely created.

The final step in the jobs is a more complex process. GitHub actions provides the opportunity to import an action, then utilize it in a step, as well as providing it with configurable settings. We have used this feature to import and use a third-party action; this action builds a Docker container, pushes the container to the deployed environment, and tells it to release.

What we have access to configure with this third-party action is what it builds (the application), the back/front-end, the target location of deployment, and the contents of the Docker-build file (known as Dockerfile). The building process of the Docker container is a central part to the pipeline, where we have the ability to use command-line interface tools (CLI-tools). During the build process in the Dockerfile, we utilize these tools to automate code checks for stylistic and programmatic errors. This process is known as linting and makes it a part of the CI practice in the pipeline.

In terms of completion of pipeline integration, the projects have utilized nearly identical CI/CD pipelines. The differences are app and project-specific environment values and app differences for building with the Dockerfile. Next we are going to look at our choice of deployment target, release, and hosting platform.

4.10.2 Deployed application, Heroku

When deciding on a deployment-target/hosting of the applications, a meeting with a representative from Atlas.co was arranged during sprint eight (see Appendix [C]). During this meeting, Atlas suggested the utilization of Heroku regarding hosting, including some examples.

Heroku is a hosting platform, and, according to Heroku themselves, is "a cloud platform that lets companies build, deliver, monitor and scale apps — we're the fastest way to go from idea to URL, bypassing all those infrastructure headaches." (Heroku, 2024a).

When we first started looking into hosting the application, we encountered difficulties in implementing and utilizing Heroku. This caused us to look for other alternatives that would still fit our needs. It needed to be economically viable, easily integrable, and able to host a Docker container. Examples include Fly.io, Google Cloud Run, and Amazon Cloud. Additionally, we looked into exchanging Docker for Kubernetes to have more alternatives, but implementing this proved to be even more complex and was quickly discarded. The other alternatives were discarded after some testing and changes. Through substantial testing, we managed to make it operational on Heroku, which had the added benefit of being familiar to Atlas.

On Heroku, both projects have been divided into two applications (as previously illustrated in Figures [9] & [10]); A back-end application and a front-end application, which made the total number of applications on Heroku four. When deploying on Heroku and releasing the apps, they are running on "dynos", which on Heroku are separated, virtually run containers. This provides an optimal environment for an application (Heroku, 2023). There are different tiers of dynos, where our apps are running on the tier's Basic and Standard-1x/2x. The first difference between the tiers is the larger RAM capacity and computing power that the dyno is able to provide. The second is the auto-scalability feature, which we currently do not utilize. Other support features were not needed or prioritized in terms of research in the current stage of the projects.

5 Quality Control

Implementing and managing a project does not automatically cause a project to be of high quality and standard; a strategy has to be in place. In the following sections, we will define and argue for how we secured a project of high quality.

5.1 Defining quality

With our understanding of both how we managed the projects and how we implemented the projects, we are going to explore why and how we achieved good quality in both aspects. Ensuring good quality could make or break a project, hence the importance of this specific section.

An important distinction to make is the difference between Quality Assurance and Quality Control. In its simplest form, Quality Control is the *act* of securing good quality, and Quality Assurance is the *plan* of securing good quality (Indeed Editorial Team, 2023b).

5.1.1 Process Quality / Quality assurance

As mentioned, Quality Assurance is the strategy for ensuring high quality. This involves various tools used in planning and the processes that determine the product's quality (Indeed Editorial Team, 2024b). Quality also involves establishing standards for processes, which are applied throughout the project and can be continuously measured against these standards (March, 2022). In our case, we utilized different tools and processes to assure optimal quality throughout the project.

5.1.2 Quality control

As mentioned, quality control is the inspection phase following quality assurance. Its essence is a set of testing procedures that verify or control the product. March suggests that such procedures include *batch inspection, product sampling, validation testing, laboratory testing, and software testing* (March, 2022).

Product inspection and product sampling proved less significant in terms of our project. Instead, software testing would prove to be crucial. Software testing, both internally and externally, provided a control factor that allowed us to validate the progress of the projects (March, 2022).

5.1.3 Code Quality

Code quality is the measurement of code, program, or software compared to a set of predetermined values. Essentially, high-quality code can be recognized by its ease of interpretation and the amount of documentation. In addition, high-quality code fulfills a set of parameters, where such parameters could include: *Functional, Consistent, Easy* *to understand, Meets clients' needs, Testable, Reusable, Free of bugs and errors, Secure, Well documented* (Indeed Editorial Team, 2024a).

Following these parameters results in various positive affects. Readability eases new developers' experience when building on to the code, program, or software. In other words, it makes it easier to improve the existing code (Indeed Editorial Team, 2024a). Transferability entails the development of code that can be easily transferred to a new product owner where minimal to no changes must be made for the code to be utilized (Indeed Editorial Team, 2024a). Transferability is an essential parameter in a project where it's goal is to be handed over or transferred.

5.1.4 Product Quality

Product quality in its essence is determined by how well customers' needs are satisfied. In addition to serving its intended purpose and meeting industry requirements (Indeed Editorial Team, 2023a). There are many factors that determine the success of a product, both in the customer aspects and in the products themselves. The Indeed Editorial Team highlights a set of perspectives that can determine product quality, and these perspectives are: *Performance and intended function, Reliability of the product within a specific time frame, Conformity to product specifications, Product durability and lifespan, Product serviceability, Physical features of the product, Customers' perception of the product* (Indeed Editorial Team, 2023a).

Fulfilling these perspectives secures a project of high quality. In its simplest form, if a customer is not satisfied and does not use the product, its very existence disappears. An easily determined factor is the intended purpose of the product; if a product satisfies its intended purpose in every aspect, it is a sign of a high quality product (Indeed Editorial Team, 2023a).

5.2 Implementing quality

With our newfound understanding of quality definitions, we will argue how we secured quality within our projects in the following sections.

5.2.1 Process Quality / Quality assurance

We decided early on to adopt Trello for planning [3.1.1]. This led us to be extremely well-structured and has aides us in keeping the quality of the development process up to modern standards. Since we run such a fast-paced Agile work environment, it has been crucial for us to have tools to complement this, and Trello has been more than satisfactory for our needs.

Working in an Agile environment and having daily Scrum meetings in the morning has helped greatly in assuring that all the developers and managers are keeping up the quality of our projects. Upon completing the daily meetings, both two teams decide whether they want to work for themselves or together. This assures that both teams can easily ask each other help, if needed.

As we have an online work environment, we use Discord (see Figure [14]) to emulate a real-life work environment. Simulating this environment ensures that we can always reach each other if needed. Quality can suffer if we do not let team members have some options in working conditions, and as mentioned in Section [3.1.2] we have several rooms for different needs. This is shown in practice in Figure [14].

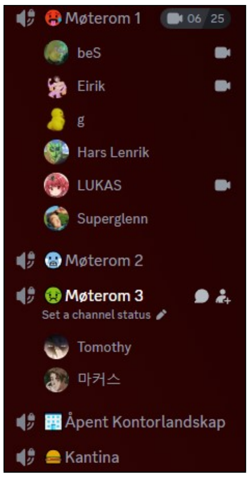


***Figure 14**. Discord voice channels*

5.2.2 Quality control

With the proper groundwork through quality assurance and process, it is essential to act on the plans and strategies laid down. We decided early on that continuous manual testing internally in the groups was an essential factor for our success, hence why we underwent testing of the projects whenever a change or new feature was added. By doing so, we believed we would save time later in the process by having already validated large parts of the code, where only new parts needed to undergo testing. This type of control and validation is what we believe contributes to good quality.

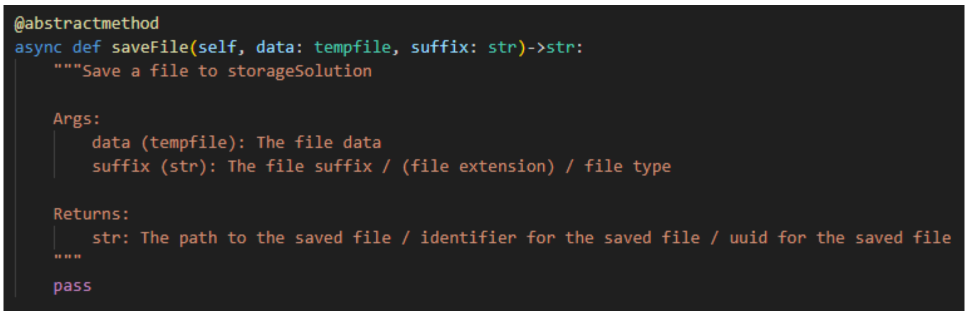
External testing is a factor we acknowledged early on that is crucial for good quality within a project. Though, time did not allow this to happen as early as we intended. In the later stages, we underwent external testing by utilizing our advisors connections, and this testing highlighted unforeseen complications that required addressing. Even with the late external testing, we managed to implement and/or fix smaller features and bugs. In an ideal world, we would have started external testing sooner to maximize the input which could lead to a better product, hence enhancing quality.

5.2.3 Code Quality

Quality coding involves various important aspects to guarantee the software's dependability, ease of maintenance, and effectiveness.

Readability is an important aspect as it ensures that other developers who wish to further develop the software can clearly understand the code base. This includes clear variable names, consistent formatting, and proper documentation.

Additionally, it is important that the code base is easily expandable, enabling it to handle feature expansions and larger user loads. We did this by making our code as adaptable and reusable as possible. An example of this could be the use of abstract classes used for storage in Image2Map; the abstract classes made it easy to implement various types of storage handlers with minimal effort.



```
@abstractmethod
async def saveFile(self, data: tempfile, suffix: str)->str:
    """Save a file to storageSolution

    Args:
        data (tempfile): The file data
        suffix (str): The file suffix / (file extension) / file type

    Returns:
        str: The path to the saved file / identifier for the saved file / uuid for the saved file
    """
    pass
```

Figure 15. *An abstract class method named "saveFile"*

All complex and large functions are developed with this in mind, as it is important that we keep things modular and well-structured. In turn, easing the development environment for future developers to add new features and functionalities. This is also crucial for the long-term success of the projects and its later potential implementation within Atlas's main app. Creating reusable functions and components also greatly help in minimizing redundancy and promoting efficiency.

As both projects have been developed with the intent of learning, we have not created any automated unit tests for our projects, as an attempt to become even more familiar with our code and how it is coupled. However, we have been actively testing out the products internally by trying to break them, stress testing, and some smaller user tests. Although, since we have focused on making readable and well-structured functions, adding such unit tests later on would be relatively easy for any future developers to integrate.

Utilizing a version control system, such as Git, has proven essential for tracking changes, collaborating between developers, and reverting to previous versions if necessary, as detailed in Section [3.1.3]. In turn, this assists us in ensuring code integrity and a smooth development cycle through being able to continually compare development branches, as well as providing a feedback loop where Atlas can continually monitor progress. This is crucial for our work, especially since we are working in a remote environment with occasionally different time zones and several developers working on the same project in real time. More importantly, this tool allows us to secure quality control through secure repository hosting and proper validation of our code through peer reviews.

We also implemented Github Actions, as mentioned in Section [4.10.1], which easily enabled us to test and publish our code base to the cloud utilizing Heroku. This ensured functional code when it reached production, as well as preventing broken branches being published.

5.2.4 Product Quality

We ensured good product quality on both projects by ensuring our functions and third-party APIs perform well. This includes researching and making sure we are using the fastest and most reliable external APIs. Our own back-end also has to be reliable and optimized so that the product the users experience feel snappy and reliable. We accomplished this by testing our project frequently, and always looking for better ways of ensuring the efficiency and reliability of our projects.

The teams have been following the few specifications Atlas requested since the beginning, as well as ensured our projects kept up with their specifications by doing weekly meetings with the Atlas team. This ensured that even if we added features that seemed fit, it would always get the final approval from Atlas before we finalized the features.

Our two projects are built on a modern tech stack, which should withstand having to be rebuilt or maintained often. As we have made two web applications that are fundamentally different, both have different durability and lifespan ratings. Image2Map has the advantage of depending on few external APIs compared to Text2Map, which would not be operational without its main contributor OpenAI. However, even though this is a possibility to be a lifespan limit, we do not see an end to the external API anytime soon.

6 Final Product

This chapter focuses on the finalized versions of the products both groups worked on, and covers the functionality that was added, the architecture of both projects, as well as ending with suggestions and thoughts for future developments on both projects. However, before we dive into the final products, let us recap the process of how they transpired, from the start of the project to the end, by summarizing the development timeline.

6.1 Summary of development timeline

This section encompasses a summary of the development timeline during the development of Image2Map and Text2Map, and is based on our Scrum daily notes (as can be seen in Appendix [C]).

Sprints 1 to 4 were characterized by slow developments, due to a high focus on individual learning of the technologies, frameworks, and programming languages utilized. Progress on the applications was still made; both applications got an early UI design based on the Figma sketches provided, a rough back-end running, and a common-designed startup script. Image2Map managed the first file conversions and started the research process in terms of implementing georeferencing for the back-end, while Text2Map got the first coordinates showing on the map, and after started using an API to retrieve GeoJSONs for retrieving and displaying outlines of countries on the map.

After this, things started to speed up. During sprints 5 to 7, it was clear that the teams were getting more and more comfortable with the technologies, the work hours, and the collaboration. The way code was produced, reviewed, and added to the "main" code-base was clearly defined at this point, and both applications were clearly mimicking their original mockups. Many functions were written and implemented in both projects, and the finished parts were further improved and polished. Already at sprint 5, the back-end portion of the initial georeferencing functionality of Image2Map was implemented, and Text2Map was further refining displaying information on the map, such as adding states and cities.

From there, the development process continued to progress at a high pace. Image2Map's group redesigned landing pages, cropping, map markers, improved image transformation, and a major refactor of the back-end, as well as a rudimentary version of the georeferencing working in the front-end. Text2Map further refined their AI inputs/outputs, and how and what data was displayed on the map, with a focus on making sure that the data was accurate (such as Victoria being either the Australian state or the Canadian city based on context). Additionally, during this time-frame, both applications got dark modes and a lot of quality-of-life updates and bug fixes.

Sprints 8 through 10 was the period in which both tools functionality was nearing a testable state and getting ready to be hosted. During this time, both projects created Docker files and planned deployment to Heroku. For Image2Map the overlay view was introduced using tile-serving, a "sniperscope" added to help users get exact marker placements, and some mobile work was done. Text2Map got some fundamental features such as updating and deleting markers for cities, states, etc. In addition, several quality-of-life features were introduced, such as better markers and better responses.

At the beginning of sprint 11 both projects went live for testing, and both projects received feedback from a testing session at Atlas. This means that the last sprints, 12 to 14, were defined by heavy bug fixing, quality-of-life updates, implementation of feedback features, and preparation for ending and handing over the project.

For Image2Map, this included adding a coordinates table where the user can delete markers, fixing a pesky bug (images would duplicate in the back-end and take up a lot of space), a user guide, and making the application generally stable and less prone to crash. Additionally, multiple storage alternatives were added and tested (database provider and storage provider alternatives to local storage), which in turn required additional bugfixes that were previously undiscovered.

Text2Map also had some larger changes during this time, and a decision was made to change the way locations were interpreted; rather than having the AI assistant respond with text and then use another LLM to interpret locations, the AI assistant would provide answers in a specific format. This greatly increased accuracy and made sure that problems like the "Victoria-conundrum" were avoided.

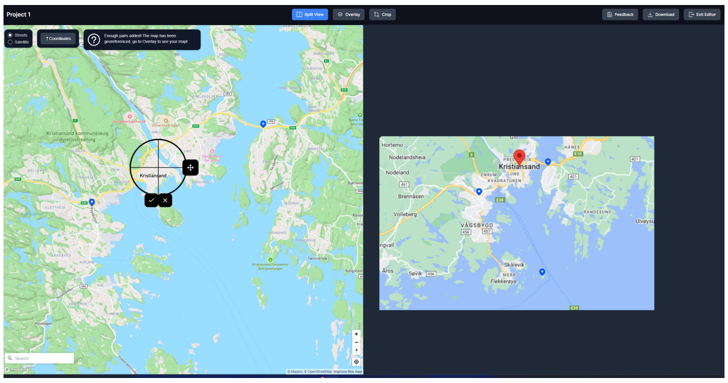
The rest of the available time before the code freeze date was spent on minimizing bugs, writing documentation, and generally preparing for handing over the projects to Atlas, such as making sure that the code that was being handed over would be understandable, readable, and usable. Sprint 14 was completed with the projects being successfully handed over to Atlas over a digital meeting, giving a demo and answering any questions they had.

Now that we have described the development process of the applications, we can move on to exploring the final products.

6.2 Functionality

Functionality in its essence entails specifically how an application functions. In this section, we will discuss the final functionalities within both projects.

6.2.1 Image2Map



Enough pairs added! The map has been georeferenced, go to Overlay to see your map!

***Figure 16**. Split View after uploading and placed a marker on the map*

When the user first visits Image2Map, they are greeted with a landing page. It states some information about what you can do with the web application, with a button where you can upload a picture to be used for georeferencing. If a PDF file is selected when uploading, the user will be redirected to a page where they can select a specific page of the document to be turned into a picture for further georeferencing. After the image has been uploaded, the user is redirected to the "editor". Here, the user can do several things, such as being able to crop the image further if they need to use a more narrowed version of the picture, examples being cropping out all the text from a page of a PDF document.

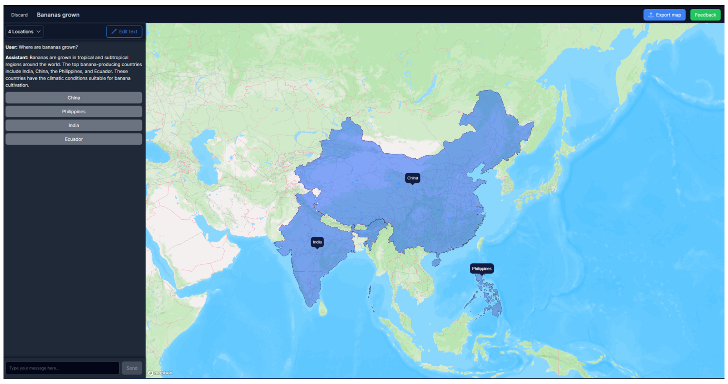
Once the user is satisfied with the selected picture, they can place a pair of markers with one placed on the map (example figure [16]) and the other on the uploaded picture. The pair is then added to the coordinate list, and the next pair is ready to be added. In order to georeference the image, there have to be a minimum of three pairs. Once there are at least three pairs, the image is georeferenced and makes the overlay view accessible.

In the overlay view, you can view the georeferenced image and see how it has been skewed and transformed according to the map (but not warped or stretched). The user is able to toggle between street maps and satellite maps, and adjust the opacity with the slider at the top.

In addition, the user can download the map as a Geo-TIFF when finished. This file can be used further with either Atlas's systems or with the use of another GIS application. Unfortunately, it is not possible to download a copy of the GCPs.

Further figures detailing the sections of Image2Map described above can be found in Appendix [B.6.1].

6.2.2 Text2Map



User: Where are bananas grown?

Assistant: Bananas are grown in tropical and subtropical regions around the world. The top banana-producing countries include India, China, the Philippines, and Ecuador. These countries have the climatic conditions suitable for banana cultivation.

Figure 17. *Split View after asking Text2Map Chat Where bananas are grown.*

When the user first visits Text2Map, they are introduced to two options. They can either choose to use a query, utilizing AI to answer, or a CSV-styled list to generate a map. These two are quite different, despite resulting in a map with highlighted locations.

A user can type in text in their own dedicated variations of the Text2Map application. The user input is then transformed into an interactive map. For example, if the user inputs "London", it will generate a map highlighting London in a split view, where the text you input on the left, and a map on the right. The places mentioned will also be shown as buttons in the text view, where the user can easily click on it and go to the location.

Utilizing ChatGPT lets users type in short questions, such as "Where are bananas grown?" (example Figure [17]), where it will generate text, then pass the response through the application processing. Once the processing is done, the user is greeted by the same split view as when providing CSV-styled lists, but with an additional AI generated response in the text view.

The user can then choose to ask the AI more questions to enhance or extend the response or add more locations through editing the text box in the sidebar. When done, the user can then download the generated map's geographical data in GeoJSON format, either with included boundaries or boundaries and markers.

Further figures detailing the sections of Text2Map described above can be found in Appendix [B.6.2].

6.3 Architecture

The final project architecture in both projects have had minimal to no changes from their respective figures, Figure [9] and Figure [10], seen in Section [4.6].

6.4 Further development

While the final product has been reached in our case, a project is "never" done. In this section, we will reflect and discuss on the groundwork that has been laid and the potential for further development.

6.4.1 Image2Map

Throughout the project, we succeeded in implementing most of the features Atlas intended us to implement; Mainly creating a solution which simplifies the process of georeferencing images and PDFs. The complete lists of features, with additional details for further development, can be found in Appendix [E]. Some of these features that were left out of the final product due to time constraints are the following:

Suggested points would make it so whenever an image is initially georeferenced, the web application would recommend the placement for the next pair of markers. This would greatly improve the user experience and the learning curve for new users of this kind of software. However, this might prove to be more challenging than anticipated, since it would either require the use of artificial intelligence or advanced algorithms to achieve this at a level we aim to achieve.

Multiple georeferenced images would provide the opportunity to have, or work on, several georeferenced maps. This would contribute to increasing the utility of the tool, by letting the user upload several images and display them in overlay view simultaneously. While not as difficult, it would still require a large rework of both the front- and back-end.

Point error calculation calculates how far off your original points were from their real positions. This was supposed to be done when a user adds more points to a map for improved accuracy, and they would be able to see a calculation of how far off their original marker placements were, compared to the re-referenced points on the map. This would let users improve the accuracy of their georeferencing even further.

Image rotation would simply provide the ability to rotate the uploaded image, which may assist with georeferencing if the image is at a weird rotation. This should be a relatively easy feature to add and should not require a lot of rework.

6.4.2 Text2Map

Text2Map is built on utilizing some of the cutting edge technology when it comes to Generative AI. We have built a Web App that is easily, hopefully, understandable for anyone where they can pick this up and use it as it is today. An extensive list of future improvements can be found in Appendix [F]. Regardless, there are several features both the user and developer can benefit from implementing in the future:

Change of Generative AI Model, as this software heavily relies on Generative AI and which is always changing and under development, hence the importance of continuous research around the the idea of using different Generative AI models like Gemini, Llama or a different versions of GPT. This is difficult and time consuming to modify and implement, as it requires a lot of research, in turn, rising expenses. However *Prompt Engineering* is cheaper. Further research into Assistant prompts needs to be constantly reevaluated. Prompt engineering is just as important as developing the software and will lead most likely lead to better responses from the AI, which would directly improve the general user experience and the reliability of the answers it provides. Comparing to changing the generative AI model, it does require a lot of time, trial and error.

CSV Feature is the lacking part of the project, whereas the ChatGPT part of the app is the most fully fledged one. CSV needs to be better implemented, judging by its infant stage. Where as of now, it is limited to extracting locations from the text without giving any further explanation or generating any extra information related to it. We want the CSV2M to be able to give the user more then just the locations. However, this requires extensive work, where we acknowledge its difficulty.

Embed Maps is also a feature both Atlas and group 22 aimed to add to the final product. Though, this ended up being difficult, where we lacked the time to implement it. This feature would let users easily embed their generated maps into their own websites or applications.

Further explanation on how to achieve these candidates can be found at the GitHub Repository (Group 22 & The Atlas Repository, 2024).

7 Reflection

Adding onto our final product, we are going to to reflect on both positive and negative aspects of both projects. Particularly important was the research and testing of the various tools and extensions with which we experimented.

7.1 Summary of Project Goals and Objectives

The two groups, and on a larger scale the team as a whole, set out to develop two tools with the aim of shaking up the GIS world by developing simple-to-use web-based tools for georeferencing (Image2Map) and visualizing geographic data from textual inputs (Text2Map). We also strived to manage the projects in a professional manner, mirroring what we might face in future work-life. Reflecting on our main goals, we can affirm that we have met most of our initial goals and objectives, found in Section [2.4].

Image2Map has somewhat successfully been developed into a quick and easy-to-use web tool to georeference quickly and easily, with the functions it was originally intended to have. However, there are some missing functions that would have been desired, such as retaining geodata when uploading a GeoTIFF, additional transformation methods, as well as downloading of GCPs.

With Text2Map the goal of creating an online tool for visualizing data and text as a digital map was mostly accomplished as well, with the caveat of not being able to upload files (CSV being the main one). Instead, the user always has to type or paste in their data/query. This was due to time constraints and prioritizing optimizing the "Ask ChatGPT" function. Next, we reflect on the methodology we used throughout the project.

7.2 Methodology and Tools

In our project, we used a mix of Trello, Scrum, and Discord to manage and assign tasks within our two groups. This combination greatly improved our productivity by keeping us all on track with a common goal.

Trello helped manage our work by dividing it into smaller tasks, making it easy for us to take a card and assign it to one another, as expanded upon in Section [3.1]. Each card could be assigned to a member with short or long descriptions of the tasks and comments if necessary. Categorizing tasks with different labels helped determine the type of task and estimate the time required. Over time, however, the two groups began to show minor differences in how they used the Trello board. To avoid inconsistencies, restructuring could have ensured that the boards remained similar.

Scrum, as detailed in Section [3.2], kept everyone up to date with the activities of the teams and individual members. It facilitated easier requests for help when tasks proved more difficult than initially planned. These meetings promoted communication within the teams, especially when faced with development challenges.

Discord was an excellent solution for remote work. Since Atlas is based in Oslo, the group's institution is in Kristiansand, and some members live in other places in Agder, we needed a reliable solution for daily Scrum meetings and general communication. We configured our server to reflect real-life offices, providing communication channels for open discussions and small talk. This setup contributed greatly to our project's success by promoting a healthy work cycle and individual work ethic. Our setup was also tested when one developer worked from Australia, proving that our remote work setup was effective without a major impact on our productivity.

Although this approach was largely successful, we did encounter some challenges. Initially, many group members showed up tired for early morning meetings, wasting valuable time getting ready. Our solution was to have everyone wake up an hour earlier, giving people time to wake up properly and ensuring everyone was ready to start at 9 AM.

Overall, by effectively utilizing Trello, Scrum, and Discord, we improved productivity, maintained clear communication, and ensured the successful management of our projects. Next, we will reflect on our software development life cycle as a whole.

7.3 Software Development Process

Our software development life cycle began with a pre-sprint hackathon at Atlas HQ in Oslo in January, detailed in Section [3.5.1], where our initial expectations for rest of the project where set.

Both groups followed a similar development process and kept up with daily stand-up meetings. Additionally, we shared similar GitHub rulesets which leads to an easier development cycle. If one team had problems, the other team would come and assist when necessary.

The initial stages of actual development included getting everyone comfortable in the new development environment as several of our developers had little to no experience with neither Python, TypeScript nor React. Our solution was to explain and discuss to developers with experience and go through their initial thoughts on how the app would be structured.

After the initial stages, we started working on the solutions Atlas wanted. Originally, our plan was to start off with Image2Map and then move to Text2Map, but this was quickly changed to us splitting up to two groups. This reflected our initial plan, where we developed our two projects separately. In this stage, we also set up most of the frameworks and a Github repository with the rules that we agreed on, as described in Section [3.1.3]. This marked the transition from planning to implementation.

The implementation stage was challenging and required extensive research. Despite heavily focusing on implementation, planning proved to be an essential part in every aspect. While implementing all the functions we would require, we also had to test our software while developing it. This consisted of mostly local testing. For Text2Map, prompt engineering played a huge role in getting the desired AI outputs.

The deployment of our applications differed between the projects. The front-end for both was deployed using Heroku. However, an older version of Text2Map's back-end was a 10GB Docker container, which faced hosting issues on Heroku. After major changes, it was successfully deployed and ran fully on Heroku, as detailed in Section [7.5.4].

Next, we will compare the two software solutions we developed to evaluate their respective strengths and areas for improvement.

7.4 Comparison of GIS software solutions

Though the projects were developed from a similar starting point, they have their own differences and uniqueness. In the following section, we are going to compare the two projects. Image2Map and Text2Map are quite different and both are helping two different aspects of the GIS world. Text2Map is very simple, with an extremely low learning curve. Image2Map on the other hand, while still very user friendly compared to the competition, has a slightly steeper learning curve if you have never attempted georeferencing before. This makes Image2Map more of a professional tool than Text2Map, which can almost be viewed as a playground rather than a tool.

Image2Map features a simple interface which lets anyone georeference either a PDF or Image with a intuitive UI. It lets anyone place three or more markers on a interactive map and your picture accordingly and the show this image on top of the interactive map after getting georeferenced. This fact is a great strength of our app, as it is significantly easier than anything that is out on the market right now, and especially better then QGIS. Performance is significantly better than previously predicted. Our app is efficient and georeferences quickly when three points have been marked. The page where you can view the image over the map is heavily optimised using raster tiles to render the map.

Weaknesses to this approach is mostly reflected though the learning curve, while very small, its still there. Rather, someone with experience using tools like QGIS will have a predictably easier time doing it than someone who has never georeferenced before.

Text2Map is a simple tool, utilized to easily make maps from simple prompts or text / CSV-styled input. It greatly reduces the amount of GIS knowledge needed to be able to create a map from scratch. It uses advanced AI to answer the prompt and give a meaningful and relatable map to the assistant output. Its strengths include its ease of use, since the only user requirement is the ability to read and write.

The weaknesses of this application appears to be its dependency on AI. While this is mostly a weakness it can also be seen as a strength. While the AI is mostly reliable and gives decent answers, it can fail, which leads to the user having to repeat the process.

It can also be quite slow at times, especially with prompts that lead to extensive outputs from the assistant. While it is positive that the AI can return such detailed and big responses, the back-end processing which retrieves the boundaries and be quite slow and even crash if it takes too long.

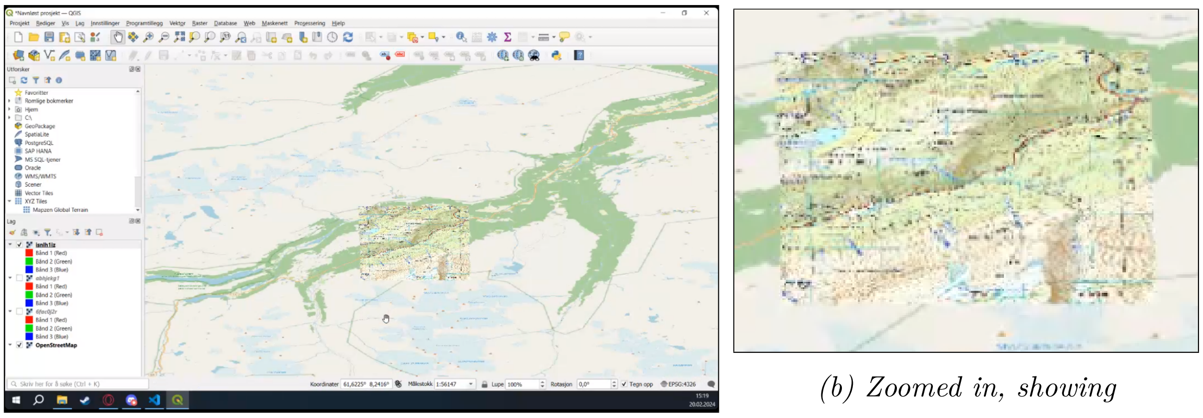
We will now delve deeper into specific achievements and challenges for Image2Map and Text2Map respectively.

7.5 Achievements and challenges

This section will detail the achievements and challenges we made and encountered during our development process.

7.5.1 Achievements and milestones in Image2Map

In Image2Map, we had three major milestones related to the task of georeferencing an image using a map. Here we will go over each major milestone in our development.



(a) Confirmed in QGIS (GIS-desktop app).

(b) Zoomed in, showing transformation match.

***Figure 18**. Confirmation of geographic metadata on image*

In Sprint five, we created the first georeferenced image in the back-end, as mentioned in Section [6.1]. By inputting mock data, including a picture and coordinates, we produced a GeoTIFF. When this GeoTIFF was used in another GIS application, the image was correctly georeferenced, as shown in Figures [18a] and [18b].

In Sprint seven, we managed to implement all the essential features for georeferencing, such as marker placement, which sent the coordinates to the back-end and returned the georeferenced image with the API-endpoint [6.1]. This allowed us to georeference images using only the application's UI and return a TIFF file containing geographic information.

In Sprint eight, we completed the tile-serving API route, which served the TIFF as tiles to the MapBox component, allowing us to visualize the referenced map for the user in the overlay view of our Image2Map application [6.1].

7.5.2 Challenges and difficulties in Image2Map

During development of the Image2Map project, the team encountered several significant challenges where our problem-solving skills were tested. Here, we will delve into specific difficulties we faced during development.

MapBox React

Poor documentation for using MapBox with React was a recurring issue. MapBox is officially made for regular JavaScript, so we had to use an external version that supports React. This made searching for relevant solutions more difficult.

Tiling service

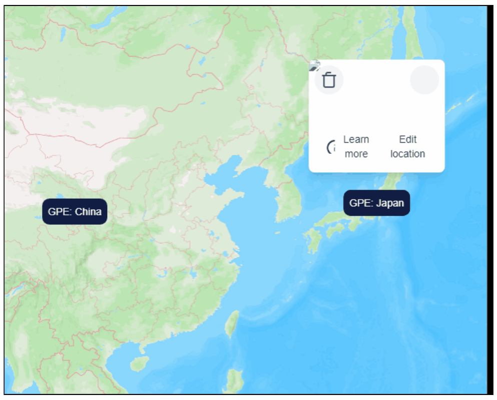
Displaying the georeferenced image on a world map was more complex than anticipated. The industry standard is to feed the image into a tiling service, which turns it into smaller tiles that are served to the map when requested. Implementing this in our FastAPI back-end using Python proved more challenging than initially expected.

The biggest challenge was finding the right Python tool for this task. Initially, we considered a static tile-serving solution that generated all tiles beforehand, but this required significant storage and processing power. Instead, we used the Python library rio-tiler (see Appendix [A.3.2.7]), which dynamically generates tiles only when needed, reducing potential storage requirements.

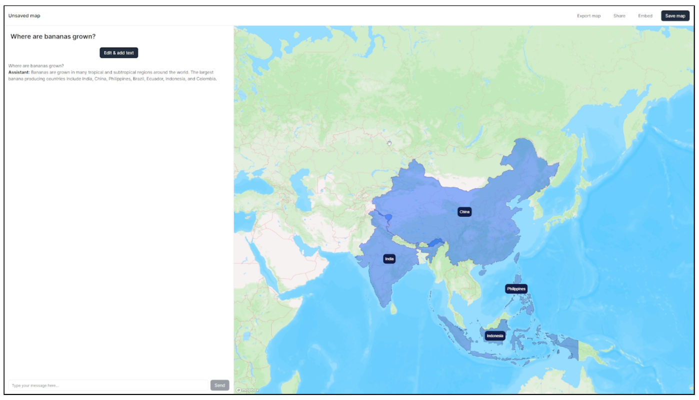
The next section will go over the achievements, milestones, and challenges in Text2Map.

7.5.3 Achievements and milestones in Text2Map

The first major achievement in Text2Map was the first response received and the marker placed on the map, as seen in Figure [19a]. This led to several smaller breakthroughs, including finding an API that was reliable and inexpensive. Bing API ended up being the best solution for our project.



(a) First Time getting a response with markers placed.



Where are bananas grown?

Where are bananas grown?
Assistant: Bananas are grown in many tropical and subtropical regions around the world. The largest banana producing countries include India, China, Philippines, Brazil, Ecuador, Indonesia, and Colombia.

(b) First Time getting a response with geometry and markers.

***Figure 19**. Achievements for Text2Map development*

The second major milestone for Text2Map was displaying boundaries on the map, as seen in Figure [19b].

After achieving these features, the focus was shifted to improving the speed and efficiency of the application. This led to a significant milestone: switching from OpenAI Chat to OpenAI Assistant. This change improved performance and reliability, allowing continuous conversation without resending chat history by letting us save each chat in a thread run by OpenAI.

During this period, we began hosting our application live. The front-end was hosted on Heroku, but the back-end was initially too large for Heroku, so we used Google Cloud Run instead.

The final major milestone was switching from using Spacy for entity extraction to having the OpenAI Assistant return a structured JSON with the mentioned places. This change improved performance and, most importantly, reduced the Docker container size from 10GB to 1GB, allowing both the front- and back-end to be hosted on Heroku.

7.5.4 Challenges and difficulties in Text2Map

Chat API Text2Map, as discussed previously, is a cutting-edge app that uses new technology such as artificial intelligence. Initially, we used OpenAI's Chat API (OpenAI, 2024), which was used about halfway through the development. This was slow, unreliable, and lost context after two messages. This was because we had to re-send the entire chat history for the AI to remember the context, which was inefficient in terms of resources, token usage, and performance.

Entity extraction Entity extraction involves any kind of extraction of countries, states, cities, and places. These are the items that are processed to be placed on the MapBox map. Extracting place entities using Spacy was initially acceptable, but had limitations. It missed some places and identified non-existent ones. Hosting the back-end with Spacy resulted in a 10GB Docker container, necessitating Google Cloud Run. We later found a better solution with the OpenAI Assistant, which significantly reduced the container size.

Geocoding Text2Map relied heavily on geocoding APIs. We tested various APIs (Google, Geoapify, ArcGIS, and OpenStreetMap), but the Bing API proved to be the most reliable and cost-effective.

Boundaries Finding a solution to obtain all the boundaries of every country, state, and city was an important, but very difficult task. We started off by using a library that used the geoBoundaries.org APIs to retrieve these boundaries, but lacked certain features and would only extract countries, not other administrative levels, without heavy modification on our part. Other APIs either had usage limits or incomplete data. Eventually, one of our developers created a custom API, discussed further in Section [7.9.3].

Our next section reflects on the progress made throughout the project.

7.6 Progress

Making actual progress throughout a project is essential; it is its core purpose. With that in mind, we are going to reflect on the progress as a whole in this section.

Overall, our progress has been good, although we have experienced a few ups and downs. Establishing a good working routine and setting core hours were prioritized as early as possible. While our initial structure of working from 10 AM to 3 PM was changed within a month to 8 AM to 3 PM, which felt more in line with real-life work schedules.

We had a few rounds of illness, where many of the team members were ill and could not work. Although progress was affected in some sprints, the teams still remained strong and the end products speak for themselves.

The two teams have known each other for years, so the dynamic was already somewhat established. There were, however, some individual expectations that were different. Although the first couple of months went by quickly with great progress, the general prioritization of the two teams has always been the products we were developing for Atlas. Since the products had a steeper learning curve than initially expected, more challenges arose as we continued the development process. Progress, from the perspective of application development, has rarely been compromised, which came at the cost of how much time we spent as a whole group on the project documentation during this time. Looking back at this, we could have delegated more resources to maintaining a steady pace on the thesis, which is a key component of documenting the development process. While few, we did experience some conflicts due to differences in individual expectations, especially as the report submission date approached. This was organized in a way that for most seemed fine at first, but in retrospect, the compromising solution favored the project development more than documenting the project progress. As a result of this prioritization, we experienced a sense of rush in terms of completing the project documentation. While experiencing these difficult situations, there were attempted compromising solutions that did not go through as the majority still wanted to keep developing. What we have learned from this experience dealing with individual expectations is how important it is to understand all perspectives of a group for it to work. While the majority saw the development as a priority, we also needed to prioritize writing.

Overall, we think our progress throughout the whole project, while heavily focused on development, has left us with two projects that we are proud of as a big team. In the next section we will reflect on the quality of the final product.

7.7 Quality

Quality has been a core focus from the beginning, throughout the development process of these two applications. Upon completion, it is time to reflect on the quality.

While both projects had quality in mind, we would argue quality translated to different things for the two projects. Obviously, the end goal for both projects is to create individually good and accurate GIS tools. Although Image2Map's end product is simply how well one can georeference, or more specifically, its accuracy and how good its tools improve the user experience, Text2Map has a very different goal in terms of quality. Its end goal is to create a good map utilizing AI. This means we rely on generative AI, which is difficult to really reflect on, as it is being constantly developed and improved upon. While we think that the current implementation is the final product, we also know that this will change in the future.

We are uncertain if anything could have been done differently for Image2Map, as its goal is relatively static in nature. We believe that what we have made a solid baseline product that holds up both quality and functionality wise to what Atlas expected. There are features that would improve our final product significantly, but were not prioritized due to time constraints. We believe that was the right decision to make, as a lot of them were based on expanding the core functionality of the product.

Text2Map could have been improved if the team had more experience in the AI space. Regardless, judging by the time-frame at our disposal, we believe we arrived at a satisfactory product. An observation is that if we had initially started with OpenAI Assistant, rather than Chat, we would have been able to develop more on the CSV part of the application. More research in the pre-sprints would have been beneficial. Nevertheless, we think our final product, and the quality of it, is still up to par with what Atlas expected from us.

Both projects implemented pipeline automation using Github Actions. This is a pipeline the Image2Map group developed, and while time consuming, we do think it contributed significantly in getting the app up and running on the cloud whenever we did a feature merge. While the pipeline could have been implemented with more internal tests before getting published, we believe the time spent on this and the pipeline that was developed was sufficient. This also goes for our branching rules and guidelines. While fundamentally simple, it made the process of merging, as well as developing. very simple while keeping the quality of the code up.

In the next section, we will go more into detail about the limitations that we observed at project completion.

7.8 Limitations

Even the most successful projects have their limitations. For future improvements, it is important to acknowledge the limitations, whereas in the following section we will reflect on the limitations we have discovered. In our development process, we have been committed to delivering quality features in line with Atlas' initial expectations for the web applications. Despite us successfully developing applications that achieve these expectations, the time constraints imposed on us during the final sprints required us to strategically focus on the more essential functionalities at the expense of more time consuming, and less important features. This prioritization was vital for us to ensure stability of the core components of the two applications.

Image2Map's limitations are mostly due to time constraints. Currently, the main usage limitations come down to our referencing process that does not do any warping of the image, and is only limited to skewing and transforming. This restricts the application's ability to accurately reference images that require non-linear adjustments. Point error calculation is a feature we had to deprioritize, explained in section [6.4.1], which could be crucial for assessing the accuracy of the referencing process. This would be especially useful for users who need to verify the precision of the spatial data, and not having this implemented will potentially result in less accurate referencing overall. Another limitation of Image2Map is that we have only implemented support for certain file types, potentially excluding other potential map formats. Additionally, uploading a GeoTIFF will result in it not retaining any geographic metadata, and only retain the image content. Next, we will address the limitations of the T2M web application.

Text2Map is based on GPT 3.5, which is limited to information up to 2021. This could easily be improved by changing the model to 4 or 4 Turbo. While removing some limitations, it adds new ones. Examples being limitation of showing boundaries implemented in 2024, like the new states in Norway. While both of these are out of our bounds to fix, they are still considered limitations of our application.

7.9 Lessons learned

Throughout working on this project, we encountered a variety of challenges and opportunities that changed our understanding of the development process. In this section, we aim to reflect on this, sharing the lessons learned during our development, important decisions made that impacted the project, things that we felt went well, and finally what we could have done differently.

7.9.1 Lessons learned along the way

This whole semester has been a great learning experience. We know that the best way of learning is often to make some mistakes along the way. In the following section, we will highlight some of the lessons we have learned during our development process which influenced future decisions.

In the Image2Map project, we encountered instances where we figured out that the solution we were looking at did not work as intended, or were unable to get working easily. Initially, attempting to create custom components for specific functionalities, such as image manipulation or dragging logic, seems like a good idea. Since this was an early feature we made, we had not yet done proper research on how well some of the external components would work in our use case. This ended up creating many bugs and issues later in development that we needed to work out. We started looking into using an external component for this feature, which could streamline development, reduce bugs, and also improve code maintainability. However, refactoring this would have required us to completely change certain sections of the code that relied on the current solution, which made this unrealistic for us at that stage in the process. This made us more aware of the need to conduct proper research of different existing solutions and possibilities when beginning development on other features. One clear example is when we wanted to add a movable scope for precision placement of markers, we instead used an existing component called React-draggable, found in Appendix [A.2.7], which overall let us more quickly develop a feature while still having high-quality clean code.

Text2Map underwent several complete refactors since the inception of the first working prototype in February. Through this, we gained a better understanding of OpenAI and its different APIs (Appendix A.4.3), and which to use for different projects. Throughout the development process, we did change from one to another. We also learned along the way that external APIs are not free, including the fact that some were better made by our team than outsourced.

7.9.2 Important decisions

Most of our important decisions were set during our initial pre-sprint. This included setting workdays and not working in weekends, core hours, and flex hours. This was a positive solution and remained mostly unchanged throughout the semester, besides the adjustment of core hours. After sprint two, we started meeting in person during Friday meetings, which also worked as the end of every sprint. The decision of collective scrum meetings ended up working great. As our two projects were similar in style and tech stack, it helped us to all get better by learning from each other.

In the next section, we will talk about what went well in our project.

7.9.3 What went well?

We are really proud of both final products, and it appears that we have succeeded in meeting Atlas' expectations, as seen in Appendix [G]. Furthermore, the issue of not having a physical office to go to worked well, using our virtual office solution (which we previously mentioned in Section 3.1.2). Daily scrum meetings went well; we saw a lot of great benefits of this from both teams, which was written about in Section 4.5.1. However, to contrast this, we will now go over what could have done differently.

7.9.4 What could have been done differently?

Looking back at our approach on both projects, we generally think a lot was achieved, although there were some elements that could have been done differently. Specifically, when it comes to prioritizing development over project documentation. Additionally, while our research seemed sufficient at the start of our semester, in retrospect, we would have like to do more research before starting to create both back-ends and the Image2Map front-end.

The tools we decided to use seemed to fit, and we believe we would have used the same tools if the projects were to be restarted from scratch. Additionally, we believe we will use several of the same tools in the future. What we think would have benefit us significantly, at least for our Bachelors thesis, would be documenting in more detail what we did through the daily sprint notes. This is something we realized quite late when we were attempting to recall sprints during the development processes of the applications.

In retrospect, we could have prioritized differently when choosing features. An example from Text2Map is enabling users to receive a link or tutorial which details how to embed their own custom maps on their own web page. This could have potentially been added if we prioritized this over further developing the assistant and making it more efficient. Regardless, both teams believe they prioritized the right functionalities as the applications progressed.

8 Conclusion

In reviewing our goals described in Section [2.4], we find that while our projects have improved and simplified aspects of existing GIS solutions, there are still areas for improvement.

The Image2Map application has made georeferencing simpler and more accessible, but lacks features such as extended image-manipulation and GCP exportation found in traditional GIS tools. Additionally, it lacks the feature of modifying GeoTIFFs, which are stripped of metadata on upload. It shows promise as a simpler alternative, but requires further development to fully replace traditional options.

The Text2Map application has succeeded in query-based visualization, but falls short in data input and visualization, in addition to the shareability of the maps. Despite this, the application offers useful features, such as AI-based formatting and GeoJSON exportation. Text2Map serves as an introductory GIS tool, but needs some refinement to fully fulfill Atlas' requirements.

Our teams have gained valuable knowledge and teamwork experience throughout this project, despite facing challenges. The learning experience and the completion of the projects meet our personal academic standards. Although our applications may not fully meet Atlas' and their customers' expectations, they provide a solid foundation for further development. Despite not fully meeting Atlas' original expectations, Atlas is satisfied with our work, and our applications show potential for future enhancements. This statement from Atlas, as well as our experiences as a collective group, suggests that we have completed the projects in a respectable, professional, and satisfactory manner.

Postface

Regardless of our achievements through both applications, we set out to follow through with an exciting project in the most ideal scenario imaginable; where we rejoice in our successful completion of two projects in a respectable and professional way, in which we can look back upon and reminisce in satisfaction. Not only, creating two applications which would leave a positive experience for both Atlas and future users, but also, two applications which emanate a polished and feature rich experience.

As a final note, we once again thank everyone we have had the pleasure of working with during these exciting projects.

References

Adobe. (n.d.). Png files. *Adobe*. Retrieved May 15, 2024, from https://www.adobe.com/creativecloud/file-types/image/raster/png-file.html

Adobe. (2023). What are TIFF files and how do you open them? *Adobe*. https://www.adobe.com/creativecloud/file-types/image/raster/tiff-file.html

Atlas. (2023). Enernite is becoming Atlas. Retrieved February 19, 2024, from https://www.linkedin.com/posts/atlasmapshq_maps-activity-7122570117095301120-UPxg

Author), A. C. (F. (n.d.). Pillow: Python imaging library (fork). *PyPI*. https://pypi.org/project/Pillow/

AWS. (2023). Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service. *Amazon Web Services, Inc.* https://aws.amazon.com/s3/

aws. (2024, April). Boto3 documentation. *aws*. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Axios. (2023). Getting Started | Axios docs. *axios-http.com*. https://axios-http.com/docs/intro

Belval. (2024, April). Pdf2image. *Belval*. https://github.com/Belval/pdf2image

Benyon, D. (2019). *Designing user experience* (4th ed.). Pearson Education.

Bjørdal, E. S. (2024). GeoB-Rust-API. https://github.com/Eiriksb/GeoB-Rust-API

Chacon, S., & Straub, B. (2024, May). *Pro Git* (2.1.428). Apress.

Codecademy. (n.d.). What Is REST? *Codecademy*. https://www.codecademy.com/article/what-is-rest

Codex Community. (2022, July). *Scrum in 20 mins... (with examples)*. YouTube. Retrieved May 9, 2024, from https://www.youtube.com/watch?v=SWDhGSZNF9M

Damoor, P. (2019, November). History of Digital Maps. *Medium*. https://medium.com/@prem.damoor19/history-of-digital-maps-b45e94ca45fa

Discord. (n.d.). Create Space For Everyone To Find Belonging. *Discord*. Retrieved May 13, 2024, from https://discord.com/company

Docker. (2023). What is a container? *Docker Inc.* Retrieved May 14, 2024, from https://www.docker.com/resources/what-container/

Docker. (2024). What is Docker? *Docker Inc.* Retrieved May 7, 2024, from https://www.docker.com/

Drumond, C. (2022). Scrum - what it is, how it works, and why it's awesome. *Atlassian*. https://www.atlassian.com/agile/scrum

European Commission, Joint Research Centre, Samoili, S., López Cobo, M., Delipetrev, B., Martínez-Plumed, F., Gómez, E., & De Prato, G. (2021). AI watch, defining artificial intelligence 2.0 : towards an operational definition and taxonomy for the AI landscape. *Publications Office of the European Union*. https://doi.org/10.2760/019901

FastAPI. (2024, May). FastAPI. *FastAPI*. https://fastapi.tiangolo.com/

Figma. (2024). Figma: the Collaborative Interface Design tool. *Figma*. https://www.figma.com/

Flanagan, D. (2006). *Javascript: The definitive guide* (5th ed.). O'Reilly Media, Inc.

FuseGIS. (n.d.). 5 Challenges in GIS Implementation — and How to Overcome Them. *FuseGIS*. Retrieved May 8, 2024, from https://www.fusegis.com/5-challenges

GeeksForGeeks. (2022, July). Static websites. *GeeksForGeeks*. https://www.geeksforgeeks.org/static-websites/

GeeksForGeeks contributors. (2020, March). Component Based Software Engineering. *GeeksforGeeks*. https://www.geeksforgeeks.org/component-based-software-engineering/

Geoapify. (2023, March). The Importance of GIS: 5 Key Benefits. *Geoapify*. https://www.geoapify.com/gis-importance

GitHub. (n.d.-a). About GitHub and Git. *GitHub*. Retrieved May 13, 2024, from https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

GitHub. (n.d.-b). About repositories. *GitHub, Inc.*

GitHub. (2024). Understanding GitHub Actions. *GitHub*. Retrieved May 6, 2024, from https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

GitLab. (2024a). What is a CI/CD pipeline? *gitlab*. Retrieved May 6, 2024, from https://about.gitlab.com/topics/ci-cd/cicd-pipeline/#why-should-it-leaders-use-ci-cd-pipelines

GitLab. (2024b). What is CI/CD? *GitLab*. Retrieved May 6, 2024, from https://about.gitlab.com/topics/ci-cd/

Goldin, S. E., & Rudahl, K. T. (1997). Why is GIS difficult. *Proceedings of the 23rd Asian Conference on Remote Sensing. Kuala Lumpur, Malaysia.*

Goodwin, M. (2024, April). What is an Application Programming Interface (API). *IBM*. https://www.ibm.com/topics/api

Gracilla, N. (2022, February). Explain it to me like i'm 5: What is a tech stack? *Medium*. Retrieved May 6, 2024, from https://medium.com/@ngracilla/explain-it-to-me-like-im-5-what-is-a-tech-stack-5e06c02e3023

Group 21 & The Atlas Repository. (2024). ImgPDF2Map. *GitHub*. https://github.com/TheAtlasRepository/ImgPDF2Map

Group 22 & The Atlas Repository. (2024). Text2Map. *GitHub*. https://github.com/TheAtlasRepository/Text2Map

Heap. (n.d.). What is a Tech Stack: Examples, Components, and Diagrams. *Heap*. Retrieved May 7, 2024, from https://www.heap.io/topics/what-is-a-tech-stack

Hellum, S. (2019, November). JesterOrNot/pymath. *GitHub*. Retrieved March 6, 2024, from https://github.com/JesterOrNot/pymath

Heroku. (2023). Running applications on dynos. *Salesforce.com*. Retrieved May 8, 2024, from https://devcenter.heroku.com/articles/how-heroku-works#running-applications-on-dynos

Heroku. (2024a). What is Heroku? *Salesforce.com*. Retrieved May 7, 2024, from https://www.heroku.com/what

Heroku. (2024b, March). Bucketeer - Add-ons - Heroku Elements. *Heroku*. Retrieved May 11, 2024, from https://elements.heroku.com/addons/bucketeer

Heroku Dev Center. (2024, April). Heroku Postgres | Heroku Dev Center. *devcenter.heroku.com*. https://devcenter.heroku.com/articles/heroku-postgresql

IBM. (n.d.). What are large language models (LLMs)? *IBM*. Retrieved May 12, 2024, from https://www.ibm.com/topics/large-language-models

Indeed Editorial Team. (2023a, March). Understanding Product Quality: What It Is and Why It Matters. *Indeed Career Guide*. Retrieved April 30, 2024, from https://www.indeed.com/career-advice/career-development/product-quality

Indeed Editorial Team. (2023b, December). Quality Control vs. Quality Assurance: Key Differences. *Indeed Career Guide*. Retrieved April 29, 2024, from https://www.indeed.com/career-advice/career-development/quality-control-vs-quality-assurance

Indeed Editorial Team. (2024a, February). Code Quality: What It Is and How To Measure It. *Indeed Career Guide*. Retrieved April 29, 2024, from https://www.indeed.com/career-advice/career-development/what-is-code-quality

Indeed Editorial Team. (2024b, February). What Is Quality Assurance? (And How To Improve Your Process). *Indeed Career Guide*. Retrieved April 29, 2024, from https://www.indeed.com/career-advice/career-development/what-is-quality-assurance

IONOS. (2023, July). Document Object Model (DOM). *IONOS Digital Guide*. https://www.ionos.com/digitalguide/websites/web-development/an-introduction-to-the-document-object-model-dom/

Maj, W. (2024, March). Wojtekmaj/react-pdf. *GitHub*. Retrieved March 6, 2024, from https://github.com/wojtekmaj/react-pdf

Mapbox. (n.d.). Getting Started. *Mapbox*. Retrieved May 14, 2024, from https://docs.mapbox.com/help/getting-started/

Mapbox. (2023, January). Mapbox GL JS. *Mapbox*. https://github.com/mapbox/mapbox-gl-js/

Mapbox. (2024). Mapbox. *Mapbox*. https://www.mapbox.com/

March, J. (2022, February). What is Quality Assurance vs. Quality Control? [5 key differences]. *www.qualio.com*. Retrieved May 1, 2024, from https://www.qualio.com/blog/quality-assurance-vs-quality-control#:

MDN contributors. (2023a, September). Asynchronous. *MDN*. https://developer.mozilla.org/en-US/docs/Glossary/Asynchronous

MDN contributors. (2023b, November). Document Object Model (DOM). *MDN*. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

MDN Web Docs. (n.d.). Working with JSON. Retrieved May 3, 2024, from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Meta Open Source. (n.d.). React. *react.dev*. https://react.dev/

Moger, F. (2024, February). Private conversation regarding funding for Atlas ["€1.5m fra investorer og €1m fra Innovasjon Norge, Forskningsrådet og Europeiske Romfartssenteret"].

Moger, F., Hofgaard, M., Rieber, H. T., & Løwe, V. J. (2024, January). Information presented during introductory hackathon at Atlas's headquarters.

Munk, S., French, C., & Brundritt, R. (2022, July). Bing Maps REST Services - Bing Maps. *learn.microsoft.com*. Retrieved May 14, 2024, from https://learn.microsoft.com/en-us/bingmaps/rest-services/

National Geographic. (2023). GIS (Geographic Information System). *National Geographic Education*. https://education.nationalgeographic.org/resource/geographic-information-system-gis/

ncbi. (2024, April). Libpq-dev. *ncbi*. https://github.com/ncbi/python-libpq-dev

Omoyeni, T. (2020). The Differences Between Static Generated Sites And Server-Side Rendered Apps [Accessed: 2024-05-08]. *Smashing Magazine*. https://www.smashingmagazine.com/2020/07/differences-static-generated-sites-server-side-rendered-apps/

OpenAI. (n.d.). Openai charter. *OpenAI*. https://openai.com/charter/

OpenAI. (2023). About openai. *OpenAI*. https://openai.com/about/

OpenAI. (2024, April). OpenAI. *OpenAI*. https://openai.com/chatgpt

ProductPlan. (2022). What is a Minimum Viable Product (MVP)? | A Product Mgmt Definition. *ProductPlan*. https://www.productplan.com/glossary/minimum-viable-product/

psycopg. (2024, April). Psycopg. *psycopg*. https://www.psycopg.org

Python Software Foundation. (2023). What Is Python? Executive Summary. *Python*. https://www.python.org/doc/essays/blurb/

python-dotenv. (2024, April). Python-dotenv. *python-dotenv*. https://github.com/theskumar/python-dotenv

Quentin F. Stout. (2018, August). What is Parallel Computing? *University of Michigan*. https://web.eecs.umich.edu/~qstout/parallel.html

Radigan, D. (n.d.). Three Steps To Better Sprint Reviews. *Atlassian*. https://www.atlassian.com/agile/scrum/sprint-reviews

Radigan, D. (2019). Agile Daily Standup. *Atlassian*. https://www.atlassian.com/agile/scrum/standups

Radigan, D. (2024). What is kanban? *Atlassian*. https://www.atlassian.com/agile/kanban

Ramírez, S. (n.d.). Features - FastAPI. *FastAPI*. Retrieved May 9, 2024, from https://fastapi.tiangolo.com/features/

rasterio. (2024, April). Rasterio. *rasterio*. https://rasterio.readthedocs.io/en/stable/

React Bootstrap. (n.d.). Spinners | React Bootstrap. *react-bootstrap.netlify.app*. Retrieved March 6, 2024, from https://react-bootstrap.netlify.app/docs/components/spinners/

Reed, S. (2024, April). React-Draggable. *github*. https://github.com/react-grid-layout/react-draggable

Rehkopf, M. (2019). Scrum Sprints. *Atlassian*. https://www.atlassian.com/agile/scrum/sprints

rio-tiler. (2024, April). Rio-tiler. *rio-tiler*. https://cogeotiff.github.io/rio-tiler/

Sazid Mahammad, S., & Ramakrishnan, R. (2003). GeoTIFF -A standard image file format for GIS applications. *Geospatial World*. https://www.geospatialworld.net/wp-content/uploads/images/pdf/117.pdf

Science Education Resource Center. (2024, February 27). *Science Education Resource Center at Carleton College*. Retrieved March 21, 2024, from https://serc.carleton.edu/research_education/geopad/georeferencing.html

Scrum Guide | Scrum Guides. (n.d.). *scrumguides.org*. https://scrumguides.org/scrum-guide.html#sprint-retrospective

scrumexpert. (2020, August). Scrum of Scrums: How to Succeed in 4 Simple Steps. *Scrum Agile Project Management Expert*. Retrieved March 13, 2024, from https://www.scrumexpert.com/knowledge/scrum-of-scrums-how-to-succeed-in-4-simple-steps/

scrum.org. (n.d.). What is a Sprint Review? *Scrum.org*. https://www.scrum.org/resources/what-is-a-sprint-review

Sheehan, M. (2018, September). Why is GIS Simple ... so Difficult? *Linkedin*. Retrieved May 8, 2024, from https://www.linkedin.com/pulse/keep-gis-simple-stupid-matt-sheehan/

SPI inc. (2024). Package: Libpq-dev (16.2-2 and others). *ncbi*. Retrieved May 8, 2024, from https://packages.debian.org/sid/libpq-dev

The PostgreSQL Global Development Group. (2019). PostgreSQL: The world's most advanced open source database. *Postgresql.org*. https://www.postgresql.org/

Trylesinski, M. (2024, May). Kludex/python-multipart. *GitHub*. Retrieved May 7, 2024, from https://github.com/Kludex/python-multipart

Tuychiev, B. (2024). Encapsulation in Python Object-Oriented Programming: A Comprehensive Guide. *datacamp*. Retrieved May 6, 2024, from https://www.datacamp.com/tutorial/encapsulation-in-python-object-oriented-programming

USGS. (n.d.). What does "georeferenced" mean? | U.S. Geological Survey. *USGS*. Retrieved April 30, 2024, from https://www.usgs.gov/faqs/what-does-georeferenced-mean

Uvicorn. (n.d.). *Uvicorn.org*. Retrieved May 7, 2024, from https://www.uvicorn.org

Vercel. (2024, May). Server-side scripting. *Vercel*. https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering

vinayakmehta. (2024, April). Poppler-utils. *vinayakmehta*. https://pypi.org/project/poppler-utils/

Walley, J. (2022, October). Johnwalley/allotment. *GitHub*. https://github.com/johnwalley/allotment

West, D. (n.d.). Sprint Planning. *Atlassian*. https://www.atlassian.com/agile/scrum/sprint-planning

West, D. (2022). Agile scrum roles and responsibilities. *Atlassian*. https://www.atlassian.com/agile/scrum/roles

Wickramasinghe, S. (2021, October 1). GitHub, GitLab, Bitbucket & Azure DevOps: What's The Difference? *BMC Software*. https://www.bmc.com/blogs/github-vs-gitlab-vs-bitbucket/

Zmejevskis, L. (2022, February 21). Ground control points. the cornerstone of accuracy. *Pixpro*. https://www.pix-pro.com/blog/ground-control-points-accuracy

Appendix A Extended information on tools etc.

This appendix contains extended information on various tools, packages, etc. that have been using during this project.

A.1 Version control: GitHub and Git

GitHub is a cloud-based platform for storing, sharing, and collaborating on code. GitHub is built on the open-source version control system Git. When using GitHub, Git is integrated into the GitHub repository, which is a central storage location for a project's code. When developing locally, one would have a local Git repository for a project, with GitHub set as a "remote" location. This enables synchronizing of Git tracked changes to GitHub where another developer can link their local repository and get the same code version on their system (GitHub, n.d.-a).

There exist several other alternatives to GitHub, such as Azure DevOps and GitLab, which also focus on version control. The difference in these services is in the additional functionality they provide for tracking and managing versions, code changes ("Commits"), and their other project management features (Wickramasinghe, 2021). We chose to utilize GitHub for version control mainly because the tool is already used by Atlas, and our group already has pre-existing familiarity with using it as well.

A.2 Front-end development

A.2.1 React

React itself is an open source front-end JavaScript library, with the purpose of building user interfaces based on components (Meta Open Source, n.d.).

Component-based development can be defined by the term "loosely coupled". What we mean by that is that each "part" or component of the system is not dependent on the rest. In other terms, remain working even when other components are replaced or changed, hence the term "loosely coupled". A byproduct of this development style is that components could, and should, be reusable. This results in a more efficient code and development process (GeeksForGeeks contributors, 2020).

Upon what can be developed through React: This library can be used to develop single-page or server-rendered applications, the latter with NextJS. More about NextJS in the next chapter.

A single-page application (SPA) is a web application or website that continually rewrites the current web page that is in use. Traditionally, your web browser would load a whole new page whenever a change was made to the interface the user is using. SPA on the other hand would never load a completely new page, but rather only rewrite and update the data that changes. By doing so, the whole process is significantly more efficient (Flanagan, 2006, p. 497).

Now that we understand SPA, it is also important to understand Document Object Model, or DOM for short, which is also an important factor in terms of Reacts functionality. DOM is a cross-platform and language-independent interface. The key characteristic of DOM is that it recognizes HTML and XML documents as tree structures, thus creating a hierarchy (IONOS, 2023).

A feature of React is the use of a virtual DOM model to create an in-memory data-structure cache, as well as computing the differences, and then updating the Virtual DOM model when needed. What this process results in is a more efficient development process where only the changes made are updated. The Virtual DOM model enables the developer to receive live updates of the page, where, in reality, only the components that are worked on are updated (MDN contributors, 2023b).

The concern of React is only user interface and its connected components, hence using such models. We can recognize the key advantage React has, which is that it only renders the parts that changes while using. This avoids unnecessary rendering of elements that remain unchanged, hence the DOM model. The server-rendered application will be explained in the next chapter about NextJS.

A.2.2 NextJS

As mentioned, NextJS is a web development framework. What NextJS does is extend React, by providing web applications based on React. Such applications provide server-side rendering as mentioned earlier, as well as static website generation (Omoyeni, 2020).

First, to understand server-side rendering, we have to understand how traditional web pages made through JavaScript libraries work. Traditionally web pages are rendered in the users browser, which gathers JavaScript's from the server to be able to render the web page. The limitations with this process are, for example, the limitation of users. What we mean by this is that only users with access to JavaScript can utilize the web pages developed this way. Additionally, users would experience increased page load times, as well as hurting the search engine optimization.

Server-side rendering, on the other hand, executes this process on the server side, instead of in the browser. The user request a web page in the browser, but instead of receiving JavaScript's that render the page, the user receives a fully rendered page that is ready to be used. A limitation of client-side rendering (browser rendering) is that the scripts themselves are bound to JavaScript. Server-side rendering accepts any and all scripting languages, hence the increased flexibility. To add on to the flexibility, by accepting more scripting languages, it also opens up the possibility for more customized interfaces for the user. In our case with large amounts of map data, it might prove to be a more efficient and smoother experience for the user on the client side when the advanced map data pages are rendered on the server side rather than on the client side (Vercel, 2024).

By implementing NextJS, it also opens up the possibility of using static web pages. A static web page is a page that is stored on the server and delivered to the web browser exactly as stored. The difference static web pages have from server-side rendering is that static web pages already exist on the server. Server-side rendering on the other hand creates new pages on the server and then delivers them to the web browser. By using static web pages, you could significantly reduce the load on the server. Hence removing live rendering of web pages wherever it is possible (GeeksForGeeks, 2022).

A.2.3 Axios

Axios is a promise-based HTTP client that works in both the browser and node.js, offering an isomorphic design for seamless operation across both client and server environments. It leverages node.js' native http module on the server and XMLHttpRequests in the browser. Axios supports features like request and response interception, data transformation, request cancellation, and automatic handling of JSON data, among others (Axios, 2023).

A.2.4 React Spinners

React spinners is a React package that supplies some "spinners", which are different variations, shapes and sizes of loading bars or circles. By using React spinners we aim to visualize the loading process as well as engaging the user (React Bootstrap, n.d.).

A.2.5 Allotment

Allotment is a React package. It's purpose is enabling resizable window panes. It has the possibility to create resizable panes both horizontally and vertically, as well as creating resizable containers within containers (Walley, 2022).

A.2.6 React-pdf

React-pdf is a React package which lets us display a pdf document on our page. In Image2Map it is used for the pdf page selector page to display which page the user wants to select for editing (Maj, 2024).

A.2.7 React-draggable

React-draggable is used to add dragging functinality for precision placement of markers and movement of the coordinate list used in the application (Reed, 2024). In Image2Map it is primarily responsible for the moving of the "sniper-scope" and the coordinates table.

A.3 Back-end development

A.3.1 FastAPI

FastAPI is a relative new web framework, intended for building RESTful APIs by utilizing Python. One of the key characteristics of FastAPI is its support for asynchronous programming. To be able to fully understand what this means, we will dive into what RESTful APIs, APIs and asynchronous programming actually mean (FastAPI, 2024).

RESTful or REST is what we call "state transfer", or an architectural style of state transfer to be specific. What REST does is to provide a standard for the communication between computer systems and the web. One way to characterize REST, is by how stateless and separate the principals are. REST separate the concerns of client and server. What this enables is changes made in client will not affect the sever, and vice versa. In other words, one could make lots of development changes in one or the other without the other knowing, hence stateless and separate. REST is the standard of communication, the communication remains the same, but content can change without problems (Codecademy, n.d.).

With our understanding of REST, we can understand what APIs are. In short terms, API stand for "application programming interface" and is an interface created for computers to be able to communicate securely with each other (Goodwin, 2024). For end users, we have dedicated user interfaces.

As we mentioned earlier, a key characteristic of FastAPI is its support for asynchronous programming. What this means is that the program is able to handle events that might happen outside the main flow of the program. It might be the new arrival of information while the main program is running, where it is able to handle such events alongside the main flow. One could call it parallelism, which stands for multiple events happening and running at the same time (MDN contributors, 2023a; Quentin F. Stout, 2018).

A.3.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics, meaning the expressions can change at runtime, often depending on the current state of the program's execution. Python's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together (Python Software Foundation, 2023).

A.3.2.1 Pymath

Pymath is an integrated library which is designed to extend Python's already existing math module, offering additional functionalities and tools for more complex mathematical computations and operations. It's aim is to provide a more comprehensive suite of mathematical resources to developers, facilitating easier and more efficient implementation of mathematical algorithms within Python applications (Hellum, 2019).

A.3.2.2 Pillow

Pillow is Python library that provides an extensive expansion in terms of file format support and image processing capabilities (Author), n.d.). In our case, it is utilized to crop, modify, and convert image file format.

A.3.2.3 Uvicorn

Uvicorn is a low-level server interface for Python that implements the ASGI (Asynchronous Server Gateway Interface) specification. This makes it possible to run servers for async frameworks such as FastAPI ("Uvicorn", n.d.). In the projects, it is used to run the FastAPI backend code, and allows the writing and usage of async functions.

A.3.2.4 Rasterio

Rasterio is a Python package which gives access to geospatial raster data; this enables us to read and write geo-specific formats such as GeoTIFF and provides Python APIs (rasterio, 2024). In Image2Map, we utilize this to mostly write GeoTIFF, but we use the reading functionality provided by the package.

A.3.2.5 Python-multipart

Python-multipart is a "streaming multipart parser" for Python that makes it possible to stream large files in multiple parts (Trylesinski, 2024). This is utilized in the Image2Map project whenever larger files are sent back and forth.

A.3.2.6 pdf2image

pdf2image converts PDFs to image objects, and is a small python util package for python version 3.7+ wrapping two other packages; pdftoppm and pdftocairo (Belval, 2024). Image2Map uses this package for its conversion tools.

A.3.2.7 rio-tiler

rio-tiler is a wrapper around rasterio and GDAL, able to map tiles compatible with web-maps. Because of wrapping, it has built-in support for rasterio datatypes(rio-tiler, 2024). Image2Map uses this to create tiles and stream them to the MapBox map to overlay a tiled GeoTiff on the map.

A.3.2.8 python-dotenv

python-dotenv is used to allow python to read key-value pairs from a .env file(python-dotenv, 2024). It is used to keep and read defined API secret, and because it is a file, separated it from the version control system (In our case, Git).

A.3.2.9 poppler-utils

poppler-utils is a pre-compliled commandline utility for manipulating PDF files and converting them to other files, based on poppler (vinayakmehta, 2024). This is a bigger dependency for other packages that Image2Map uses.

A.3.2.10 psycopg2-binary

psycopg2-binary is a PostgreSQL database adapter for Python, the core of psycopg completely implements Python DB API 2.0 (https://peps.python.org/pep-0249/) specifications (psycopg, 2024). It is used in Image2map to "talk" to a PostgreSQL database.

A.3.2.11 boto3

boto3 is an AWS (Amazon Web Services) SDK (software development kit) for Python. It enables creating, configuring, and managing AWS services, like AWS S3 (aws, 2024). We use this in Image2map to manipulate and get data from our AWS S3 buckets.

A.3.2.12 libpq-dev

"The libpq-dev library is built following Debian's libpq-dev package idea: it contains a minimal set of PostgreSQL binaries and headers required for building 3rd-party applications for PostgreSQL." (ncbi, 2024). Debian libpq-dev is in addition to compile C programs to link with the library (SPI inc., 2024). This is a necessary dependency for the usage of psycopg2 in Image2map.

A.4 Dependencies

A.4.1 MapBox

MapBox provides an API for accessing and creating custom maps of the world for use in for web, mobile and automotive applications. MapBox supplies online maps for many other popular applications and companies such as Toyota, Voi, Samsung, T Mobile and many more (Mapbox, 2024).

A.4.2 GeoB Rust API

The GeoB Rust API is a custom-made API point made and serviced separately from the main application. This API point simply returns a singular geojson file that the user searches for. This can be either countries by themselves or cities / states inside those countries. The API is written in Rust and is completely open source (Bjørdal, 2024).

A.4.3 OpenAI

OpenAI, is a research organization focusing on artificial intelligence (AI). The organization's goal is to develop "safe and beneficial" artificial intelligence (OpenAI, n.d., 2023).

A.4.4 ChatGPT

ChatGPT is an LLM developed by OpenAI that can assist with many different tasks, for example writing, learning, and brainstorming. ChatGPT is able to answer questions as well as render images based on requirements. (OpenAI, 2024).

A.4.5 Bing Maps REST Services

The Bing Maps REST Services Application Programming Interface (API) is a web service provided by Microsoft and provides a Representational State Transfer (REST) interface that allows for the integration of various mapping and location-based services into different applications. Some useful functionalities provided by the API are the geocoding of addresses (converting addresses to geographic coordinates), reverse geocoding (converting coordinates to addresses), retrieving imagery metadata, and traffic information, to name some examples (Munk et al., 2022).

A.4.6 Encapsulation

"Encapsulation is a fundamental object-oriented principle in Python. It protects your classes from accidental changes or deletions and promotes code reusability and maintainability" (Tuychiev, 2024).

A.4.7 Amazon S3

Amazon Simple Storage Service (Amazon S3) is a cost-effective object storage service delivered by Amazon. The service offers industry-leading scalability, data availability, security, and performance, and is equipped with many easy-to-use management features that help customers control and configure the service to meet their specific needs and requirements (AWS, 2023).

A.4.7.1 Heroku Bucketeer

Heroku Bucketeer is an add-on for Heroku that allows you to use Amazon S3 directly from a Heroku application. The add-on makes setup quick and simple, along with additional control and tools for convenience and quality of life (Heroku, 2024b).

A.4.8 PostgreSQL

"PostgreSQL is a powerful, open source object-relational database system with more than 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance" (The PostgreSQL Global Development Group, 2019).

A.4.8.1 Heroku PostgreSQL

Heroku Postgres is a PostgresSQL database service provided directly by Heroku, allowing for easier setup and use with applications running on Heroku. Heroku Postgres also provides a web dashboard for easier overview and control, in addition to several other useful features (Heroku Dev Center, 2024).

Appendix B Additional Figures

This appendix contains Additional figures of the final product of each of the projects, Image2map and Text2map.

B.1 Atlas Figma

B.1.1 Figma Image2Map

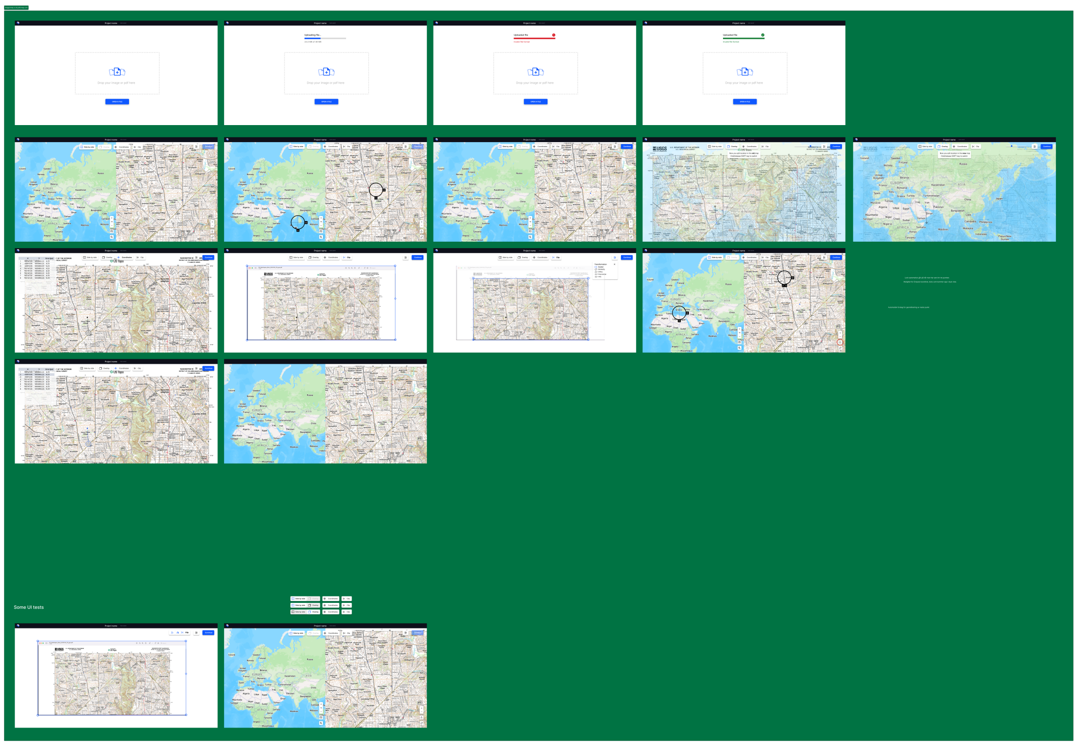


Figure B.1*. Figma Image2Map Overview*

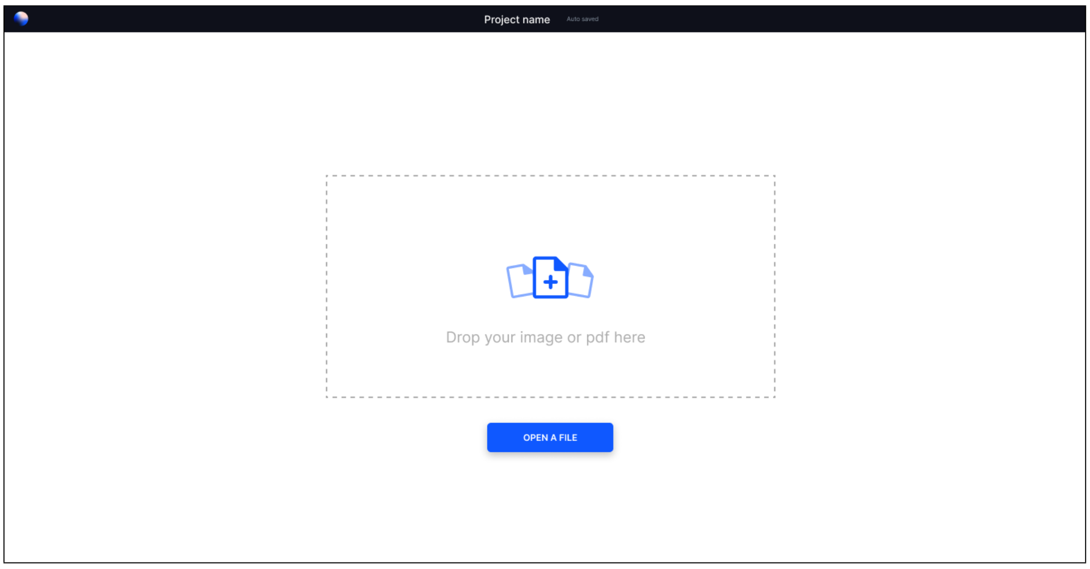


Figure B.2*. Figma Image2Map Landing Page*

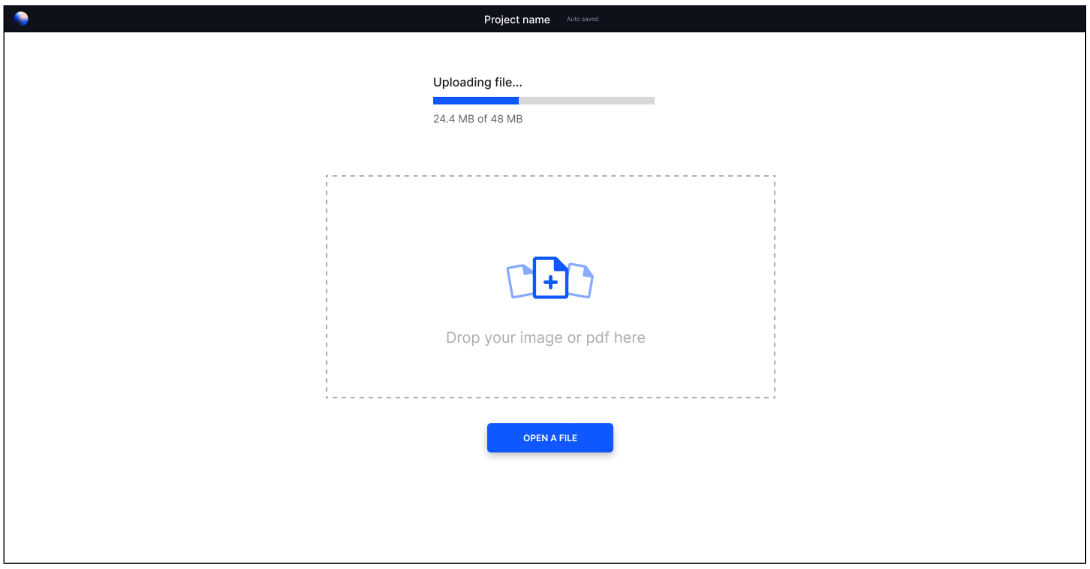


Figure B.3. *Figma Image2Map file upload*

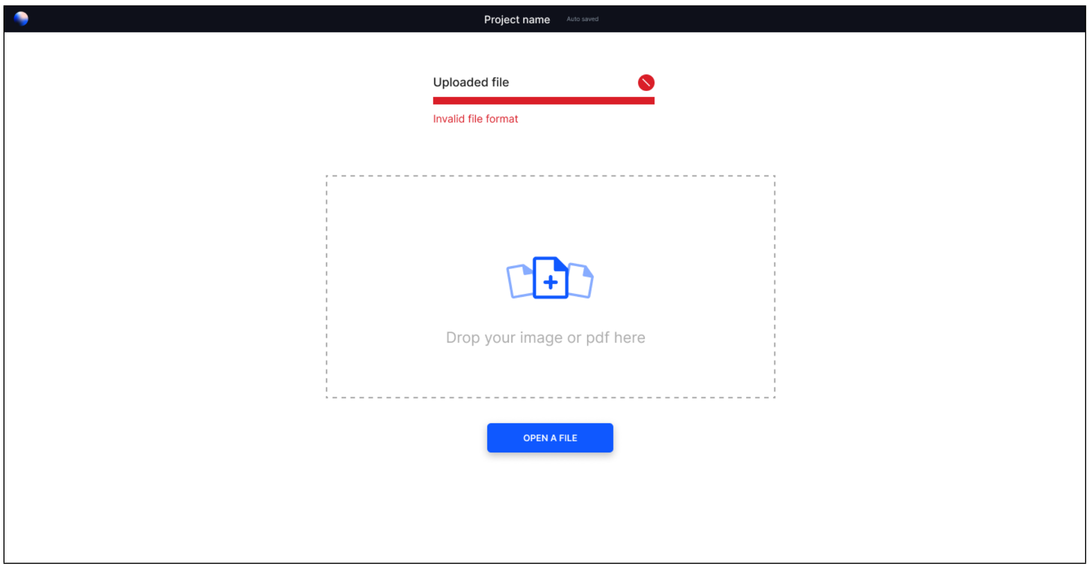


Figure B.4. *Figma Image2Map file upload fail*

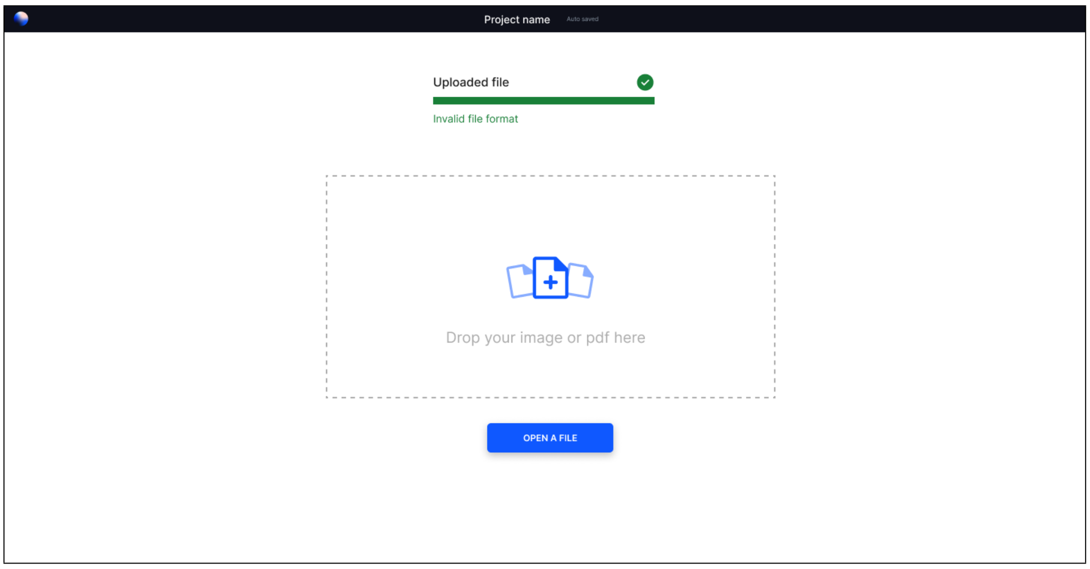


Figure B.5. *Figma Image2Map file upload success*

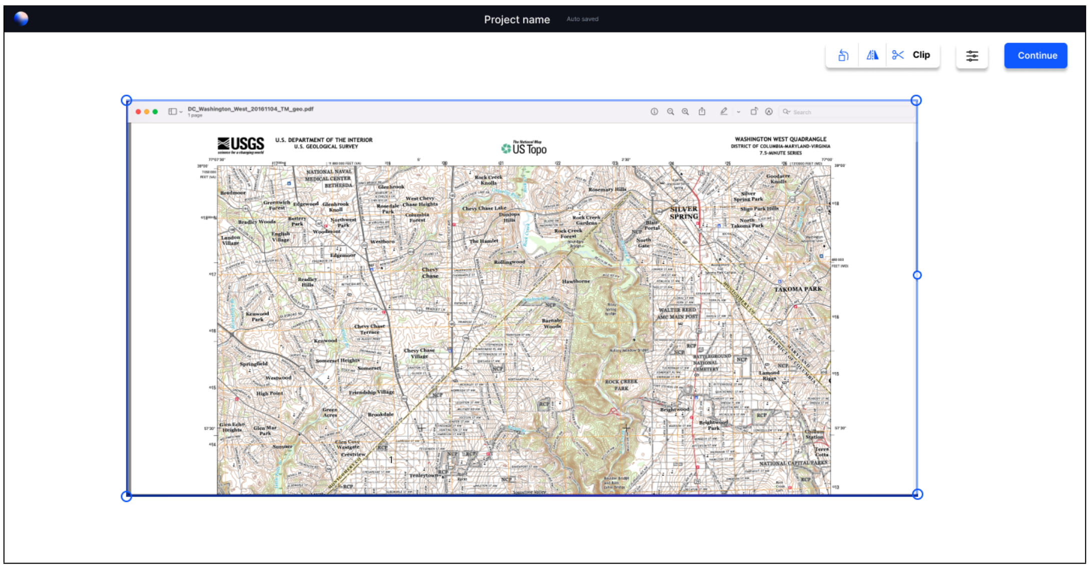

Project name
Auto saved
Clip
Continue
DC_Washington_West_20161104_TM_geo.pdf
USGS
U.S. DEPARTMENT OF THE INTERIOR
U.S. GEOLOGICAL SURVEY
US Topo
WASHINGTON WEST QUADRANGLE
DISTRICT OF COLUMBIA-MARYLAND-VIRGINIA
7.5-MINUTE SERIES
SILVER SPRING
TAKOMA PARK
ROCK CREEK PARK

Figure B.6. *Figma Image2Map Pre Edit of file*

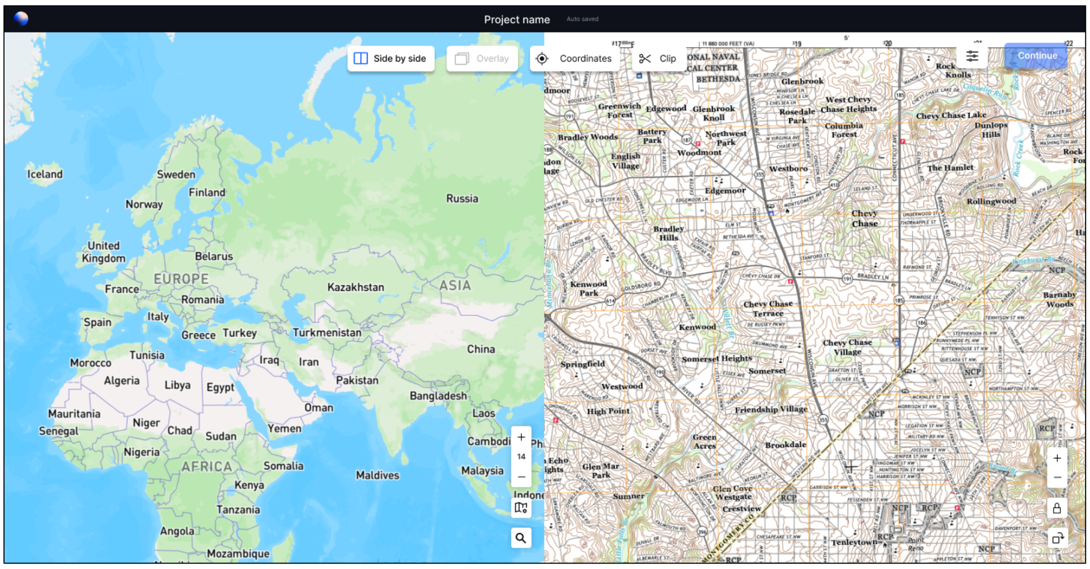


Figure B.7. *Figma Image2Map Side by Side*

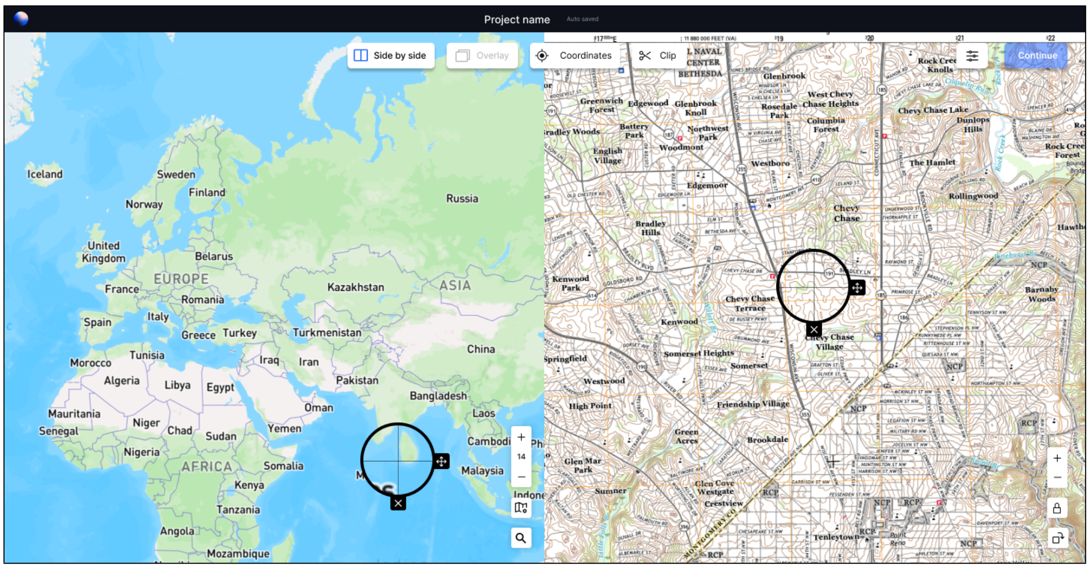


Figure B.8. *Figma Image2Map Selecting point 1*

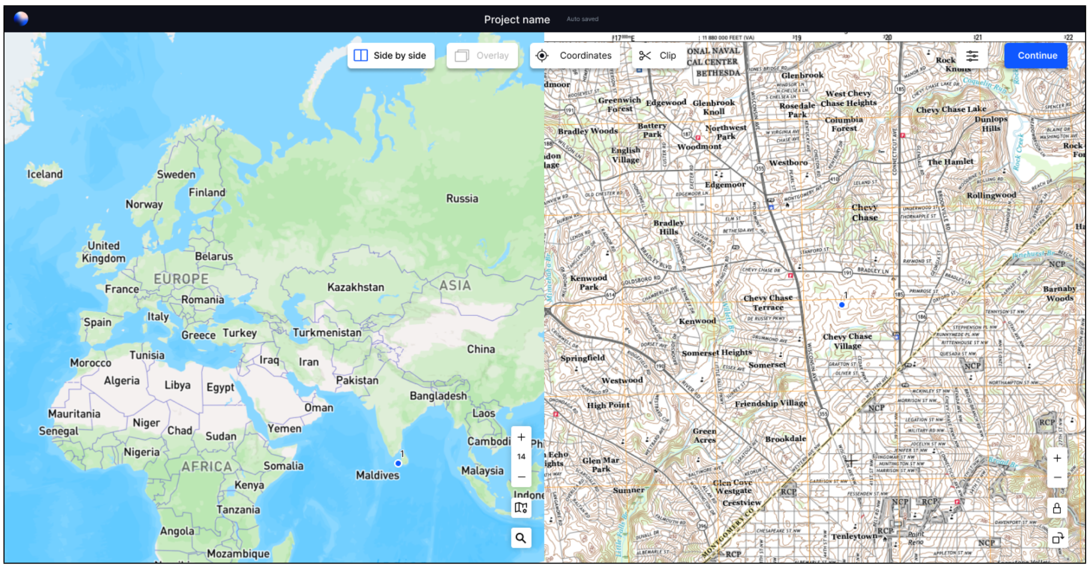


Figure B.9. *Figma Image2Map finished point 1*

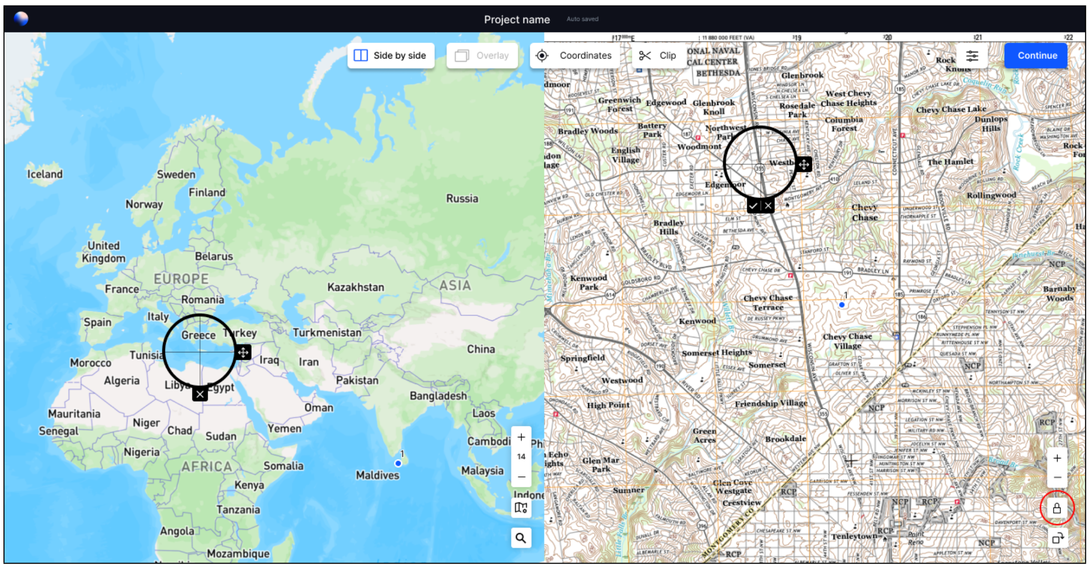
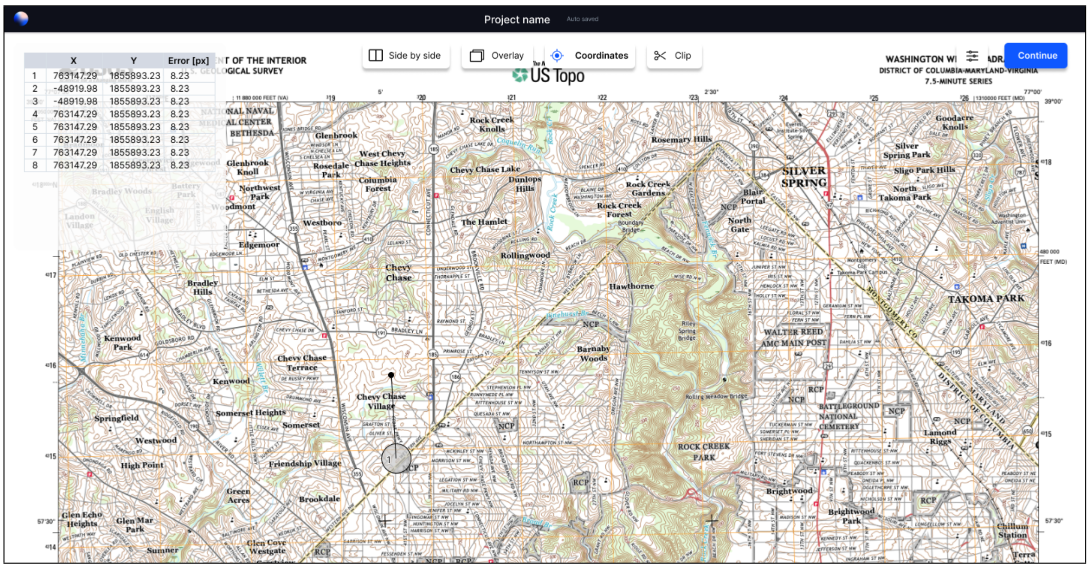
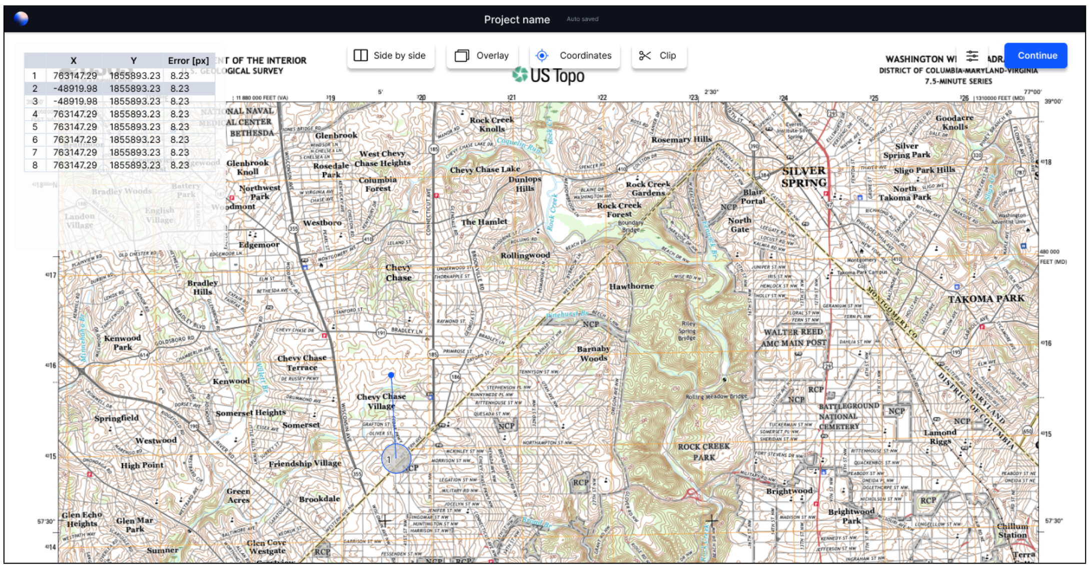


Figure B.10. *Figma Image2Map Auto-suggest point*



	X	Y	Error [px]
1	763147.29	1855893.23	8.23
2	-48919.98	1855893.23	8.23
3	-48919.98	1855893.23	8.23
4	763147.29	1855893.23	8.23
5	763147.29	1855893.23	8.23
6	763147.29	1855893.23	8.23
7	763147.29	1855893.23	8.23
8	763147.29	1855893.23	8.23

***Figure B.11**. Figma Image2Map Coordinates table*



	X	Y	Error [px]
1	763147.29	1855893.23	8.23
2	-48919.98	1855893.23	8.23
3	-48919.98	1855893.23	8.23
4	763147.29	1855893.23	8.23
5	763147.29	1855893.23	8.23
6	763147.29	1855893.23	8.23
7	763147.29	1855893.23	8.23
8	763147.29	1855893.23	8.23

***Figure B.12**. Figma Image2Map Hover on coordinates*

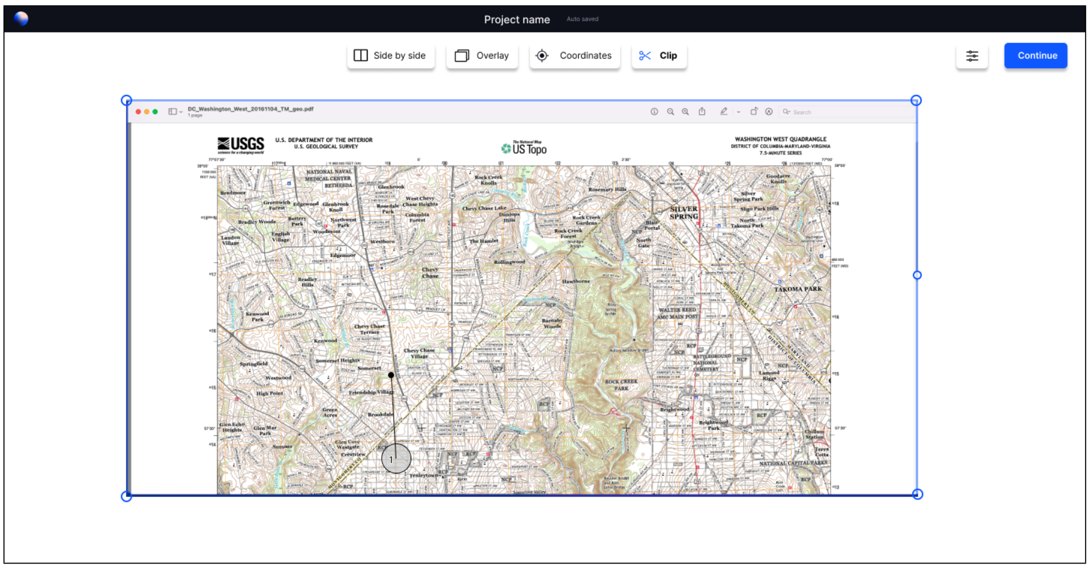


***Figure B.13**. Figma Image2Map clipping*

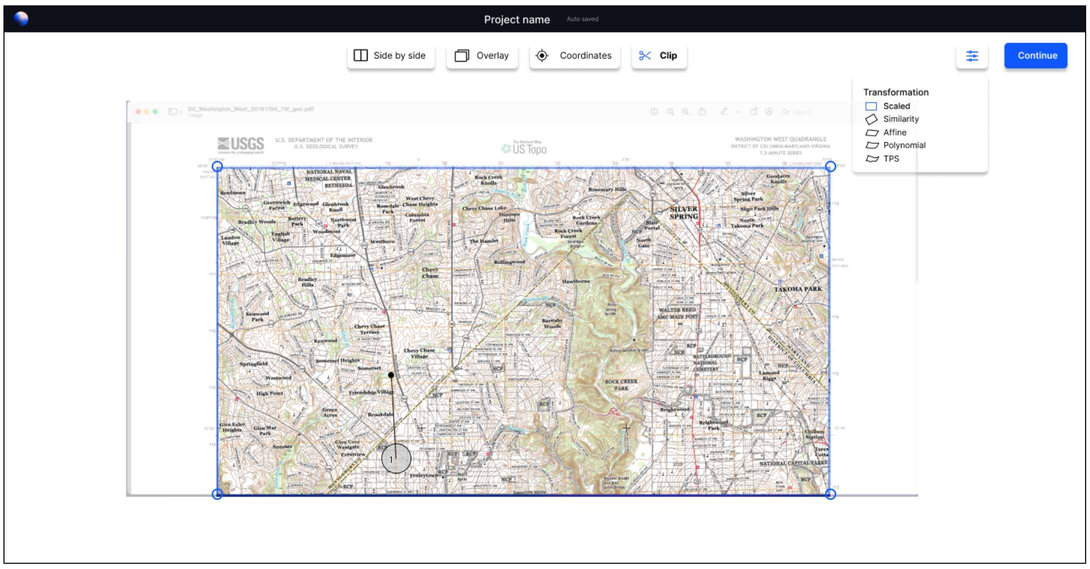


***Figure B.14**. Figma Image2Map choosing Transform*

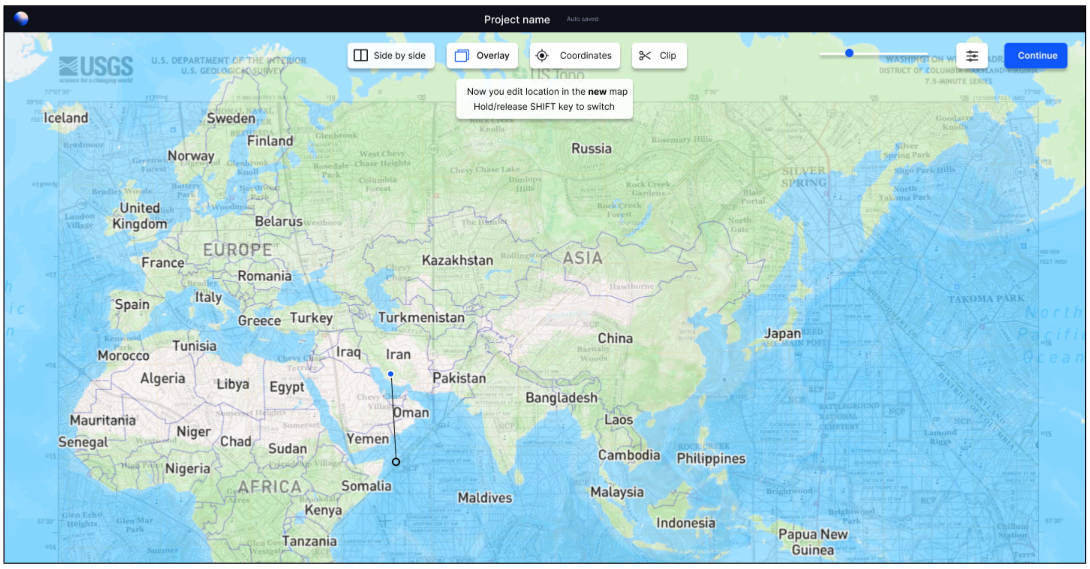

Project name
Auto saved
Side by side
Overlay
Coordinates
Clip
Continue
Now you edit location in the **new** map
Hold/release SHIFT key to switch

Figure B.15. *Figma Image2Map Map view overlay 1*

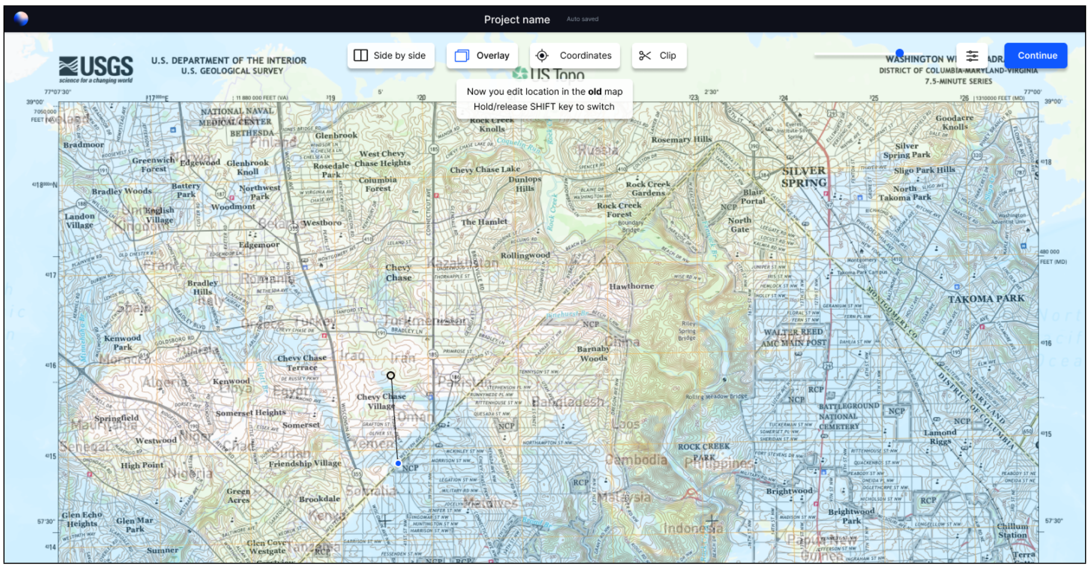

Project name
Auto saved
Side by side
Overlay
Coordinates
Clip
Continue
Now you edit location in the **old** map
Hold/release SHIFT key to switch

Figure B.16. *Figma Image2Map Map view overlay 2*

B.2 Figma Text2Map

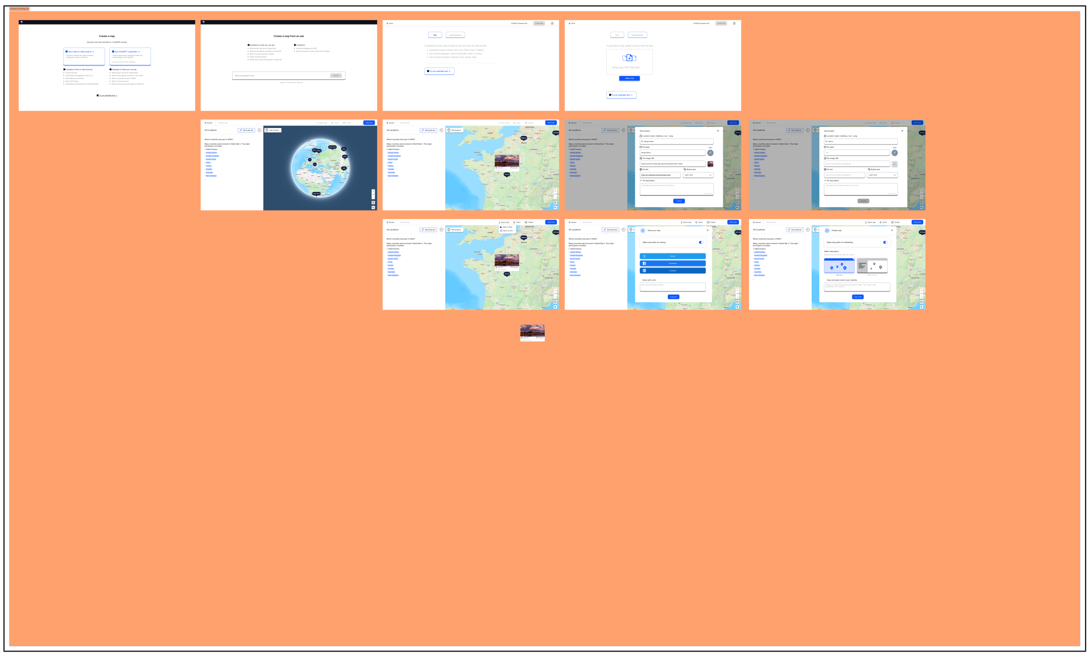
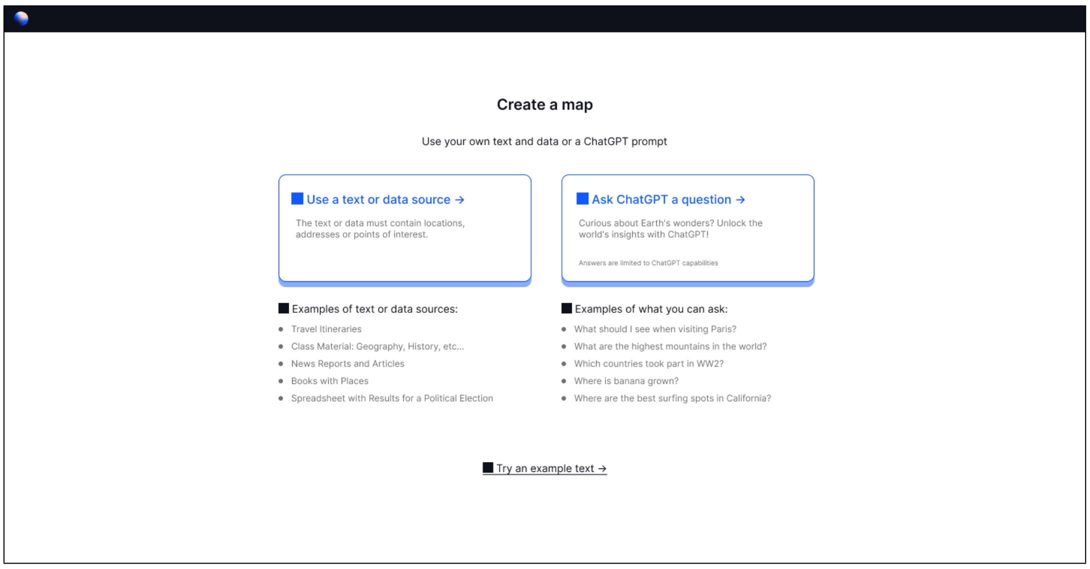


Figure B.17*. Figma Text2Map Overview*



Create a map

Use your own text and data or a ChatGPT prompt

Use a text or data source →

The text or data must contain locations, addresses or points of interest.

Ask ChatGPT a question →

Curious about Earth's wonders? Unlock the world's insights with ChatGPT!

Answers are limited to ChatGPT capabilities

Examples of text or data sources:

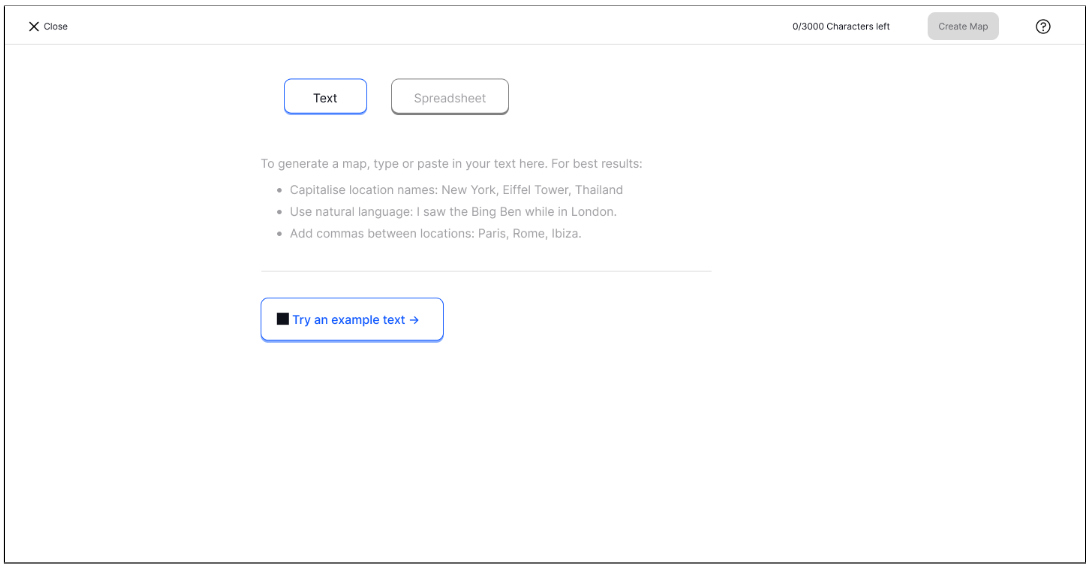
- Travel Itineraries
- Class Material: Geography, History, etc...
- News Reports and Articles
- Books with Places
- Spreadsheet with Results for a Political Election

Examples of what you can ask:

- What should I see when visiting Paris?
- What are the highest mountains in the world?
- Which countries took part in WW2?
- Where is banana grown?
- Where are the best surfing spots in California?

Try an example text →

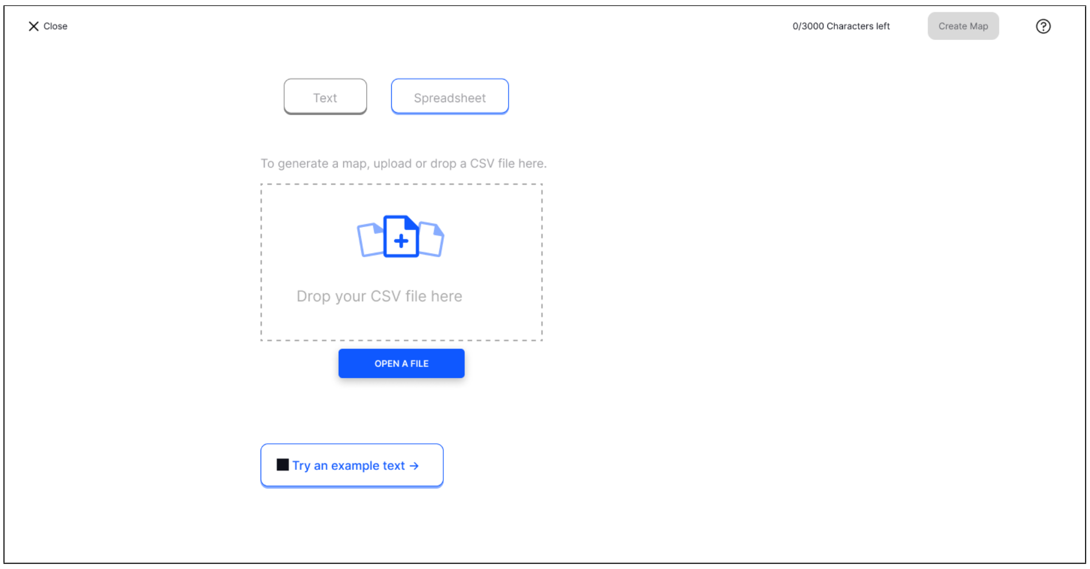
Figure B.18*. Figma Text2Map Landing page*



To generate a map, type or paste in your text here. For best results:

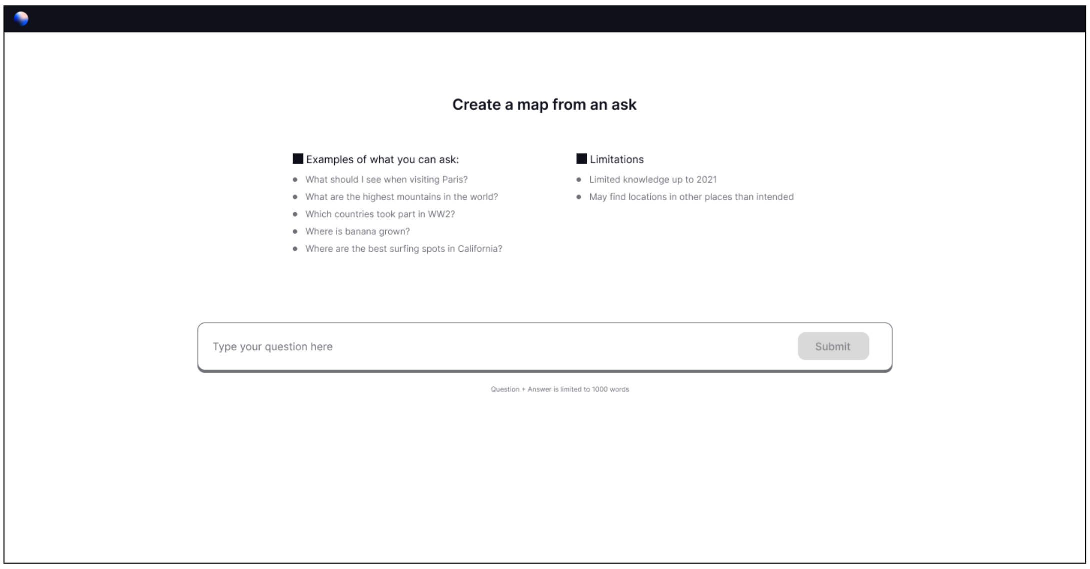
- Capitalise location names: New York, Eiffel Tower, Thailand
- Use natural language: I saw the Bing Ben while in London.
- Add commas between locations: Paris, Rome, Ibiza.

Figure B.19*. Figma Text2Map Clean text Input page*



To generate a map, upload or drop a CSV file here.

Figure B.20*. Figma Text2Map Spreadsheet input*



Create a map from an ask

Examples of what you can ask:

- What should I see when visiting Paris?
- What are the highest mountains in the world?
- Which countries took part in WW2?
- Where is banana grown?
- Where are the best surfing spots in California?

Limitations

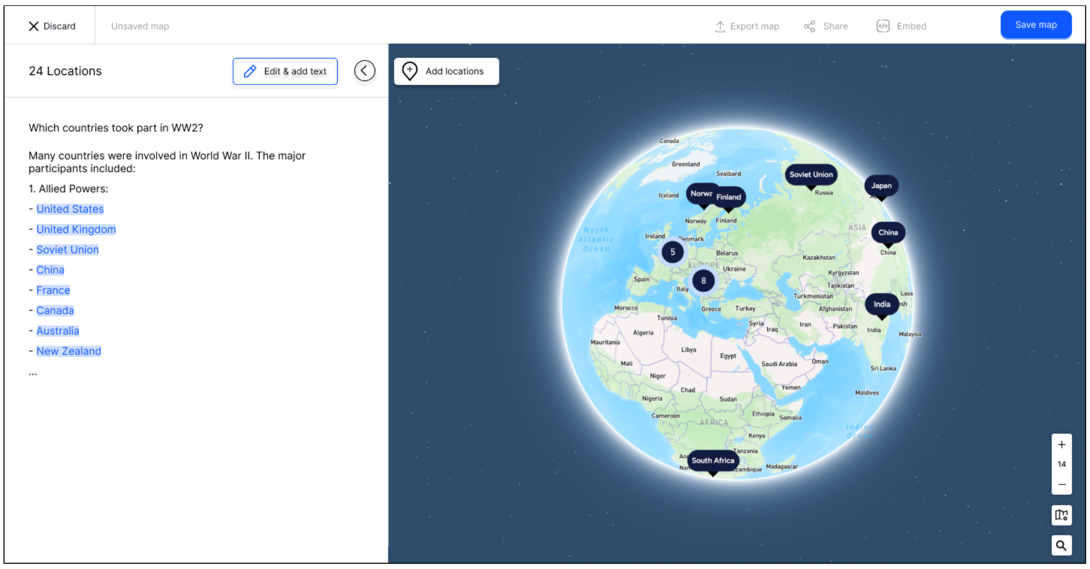
- Limited knowledge up to 2021
- May find locations in other places than intended

Type your question here

Submit

Question + Answer is limited to 1000 words

Figure B.21. *Figma Text2Map Asking AI*



Discard

Unsaved map

Export map

Share

Embed

Save map

24 Locations

Edit & add text

Add locations

Which countries took part in WW2?

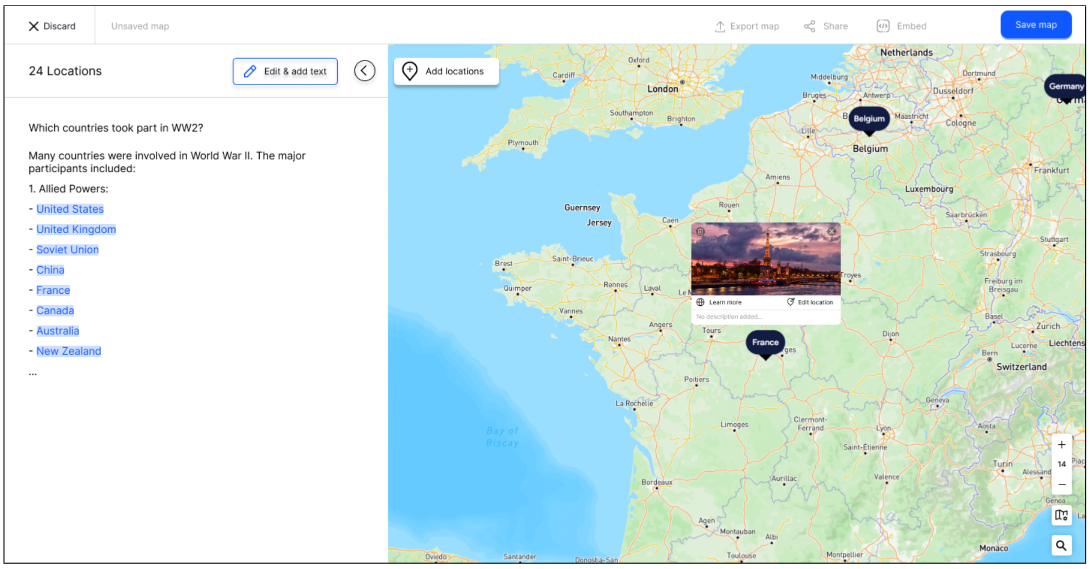
Many countries were involved in World War II. The major participants included:

1. Allied Powers:

- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand

...

Figure B.22. *Figma Text2Map Initial AI response*



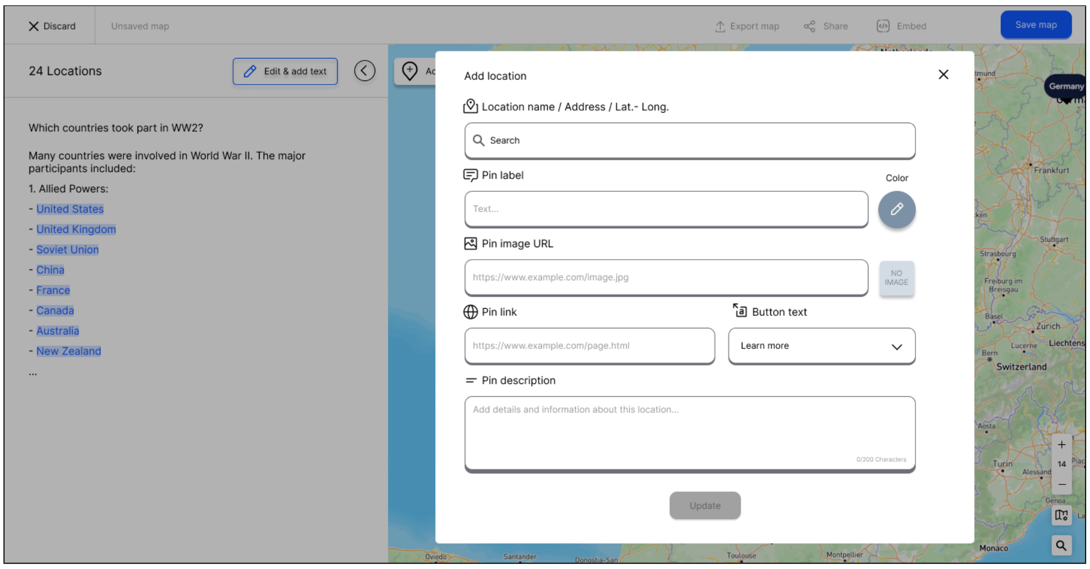
Which countries took part in WW2?

Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand

...

Figure B.23. Figma Text2Map Centered on a location



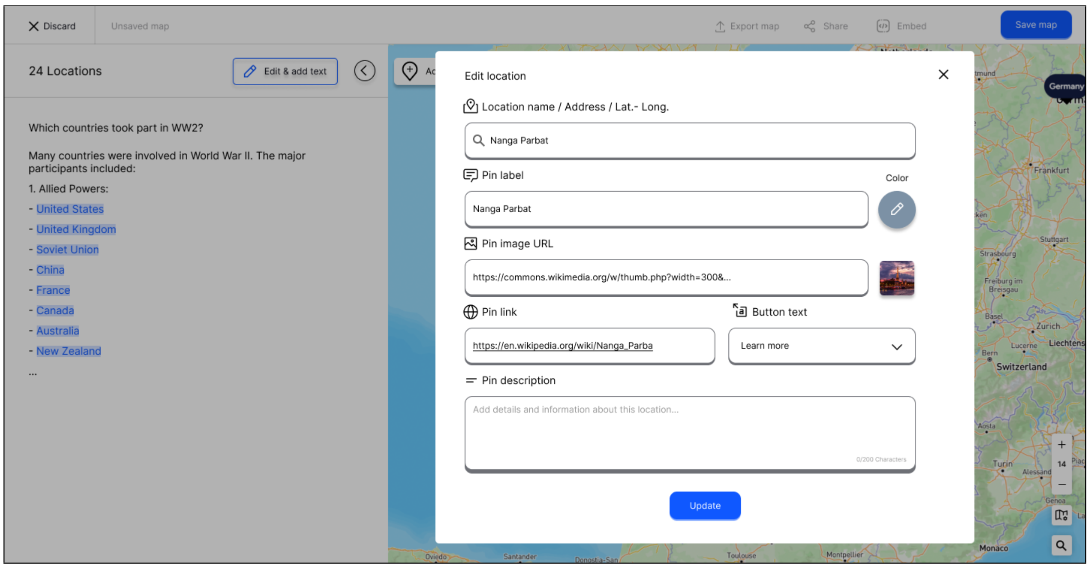
Which countries took part in WW2?

Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand

...

Figure B.24. Figma Text2Map Adding new location



Which countries took part in WW2?

Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand
...

Location name / Address / Lat.- Long.: Nanga Parbat

Pin label: Nanga Parbat

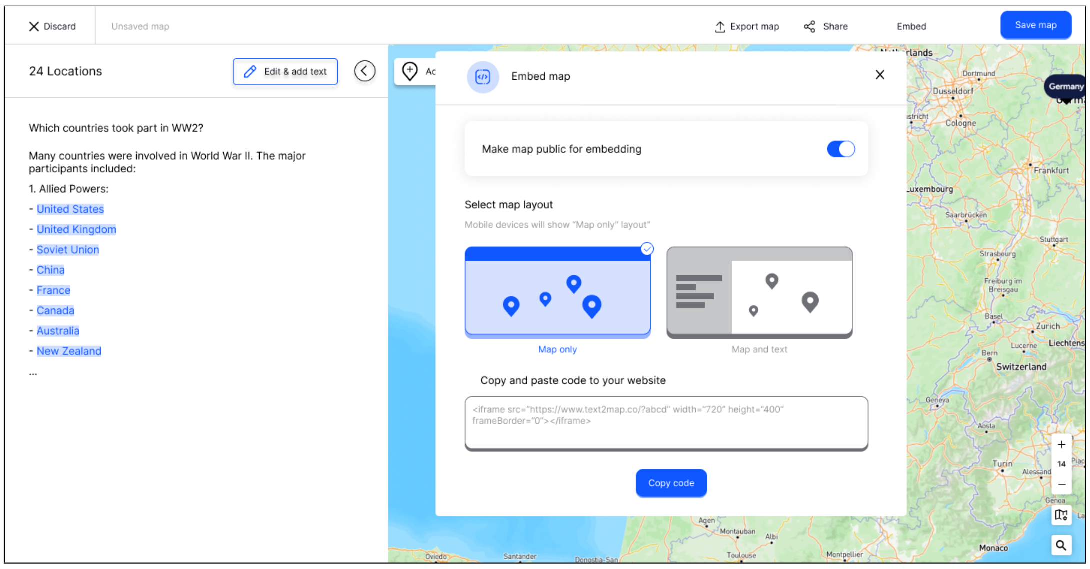
Pin image URL: https://commons.wikimedia.org/w/thumb.php?width=300&...

Pin link: https://en.wikipedia.org/wiki/Nanga_Parba

Button text: Learn more

Pin description: Add details and information about this location... 0/200 Characters

Figure B.25. Figma Text2Map Edit location



Which countries took part in WW2?

Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand
...

Make map public for embedding

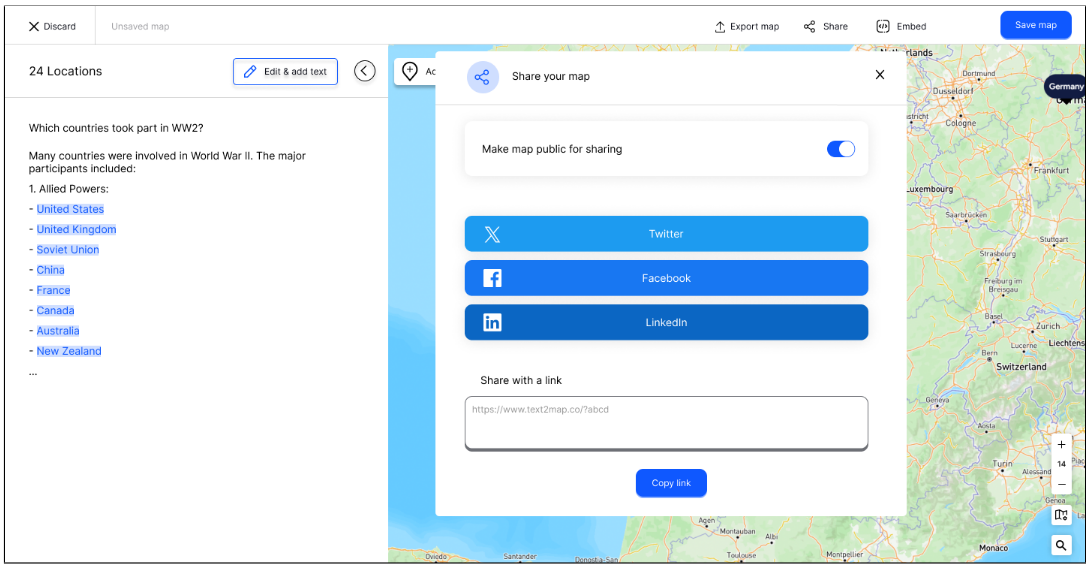
Select map layout

Mobile devices will show "Map only" layout

Copy and paste code to your website

```
<iframe src="https://www.text2map.co/?abcd" width="720" height="400"
frameBorder="0"></iframe>
```

Figure B.26. Figma Text2Map Embed map



Which countries took part in WW2?

Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand

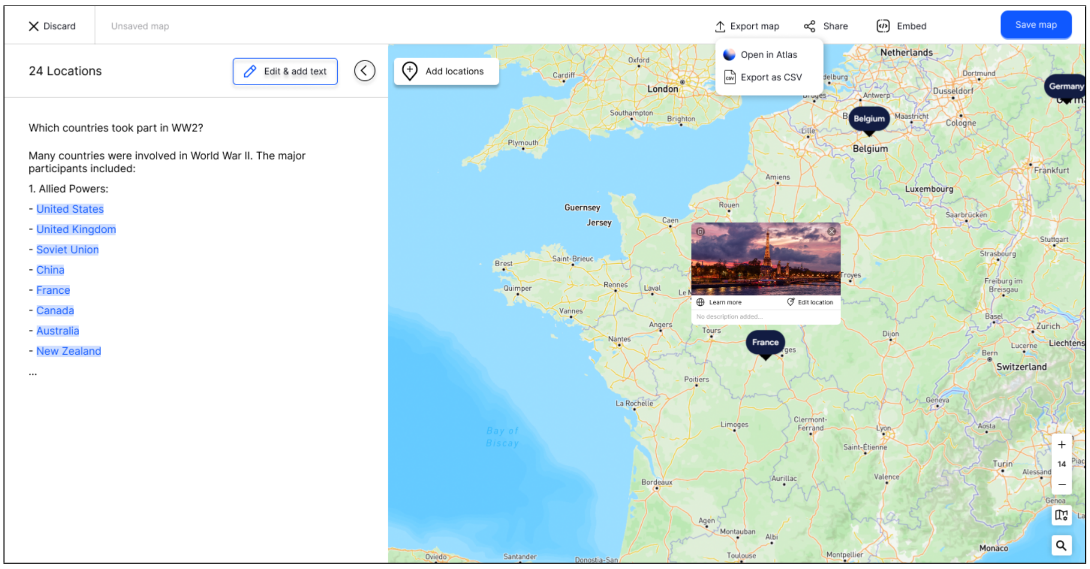
...

Make map public for sharing

Share with a link

https://www.text2map.co/?abcd

Figure B.27*. Figma Text2Map Sharing Map*



Which countries took part in WW2?

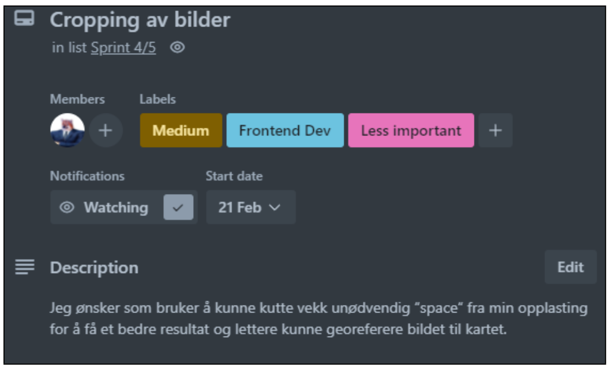
Many countries were involved in World War II. The major participants included:

1. Allied Powers:
- United States
- United Kingdom
- Soviet Union
- China
- France
- Canada
- Australia
- New Zealand

...

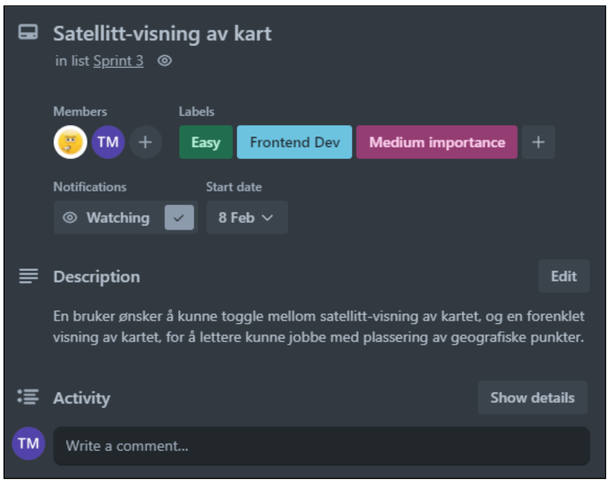
Figure B.28*. Figma Text2Map Exporting Map*

B.3 Examples from initial Trello cards


Cropping av bilder
in list Sprint 4/5

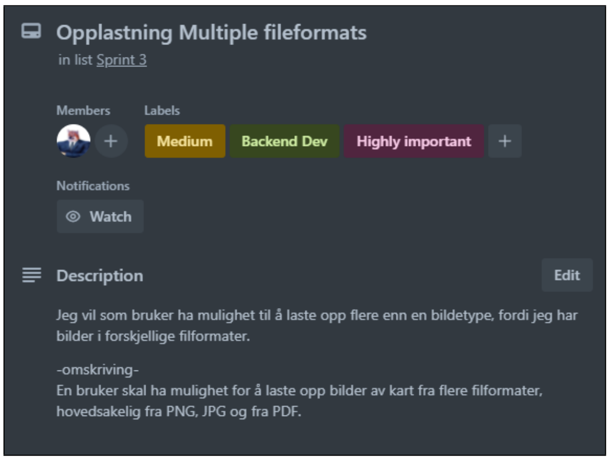
Jeg ønsker som bruker å kunne kutte vekk unødvendig "space" fra min opplasting for å få et bedre resultat og lettere kunne georeferere bildet til kartet.

Figure B.29. *Card in Norwegian, description is a User Story of Cropping*


Satellitt-visning av kart
in list Sprint 3

En bruker ønsker å kunne toggle mellom satellitt-visning av kartet, og en forenklet visning av kartet, for å lettere kunne jobbe med plassering av geografiske punkter.

Figure B.30. *Card in Norwegian, description is a User Story of Satellite-View*


Opplastning Multiple fileformats
in list Sprint 3

Jeg vil som bruker ha mulighet til å laste opp flere enn en bildetype, fordi jeg har bilder i forskjellige filformater.

-omskriving-
En bruker skal ha mulighet for å laste opp bilder av kart fra flere filformater, hovedsakelig fra PNG, JPG og fra PDF.

Figure B.31. *Card in Norwegian, description is a User Story of Uploading Formats*

B.4 Sketches & Mockups

B.4.1 Image2Map

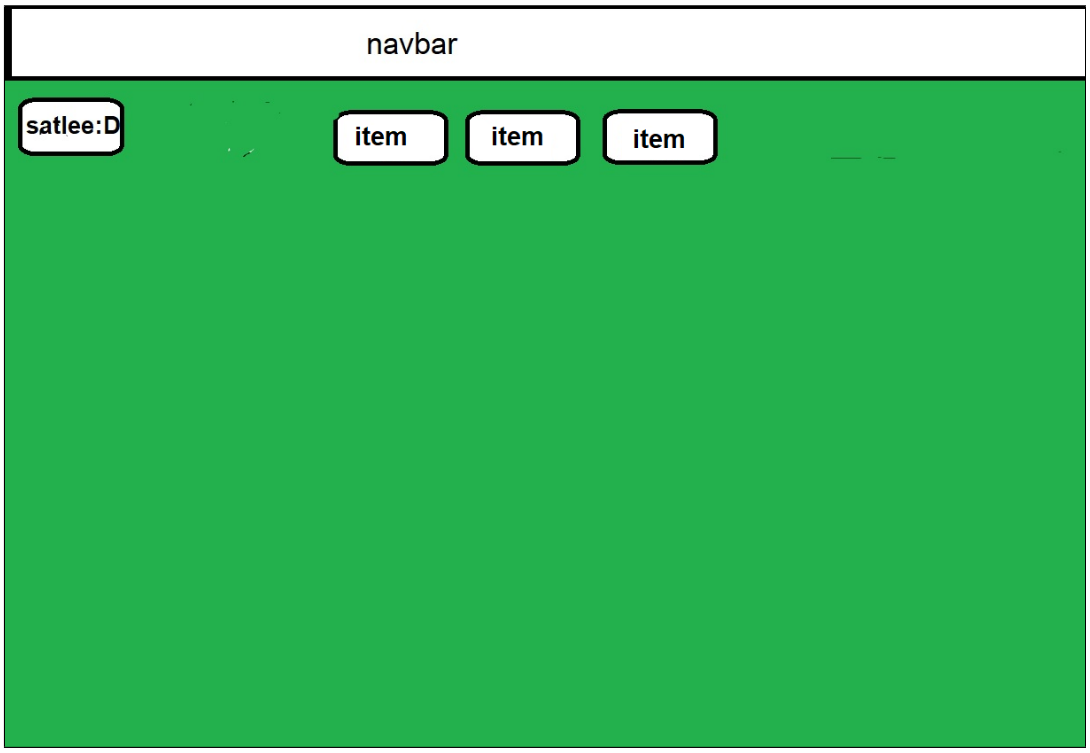


Figure B.32. *Simple sketch of the "map toolbar"*

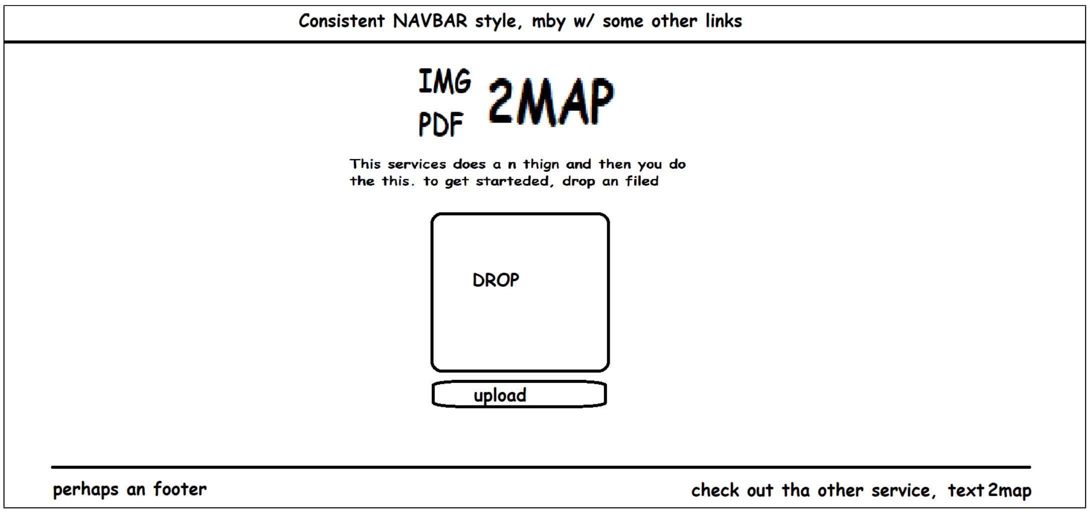
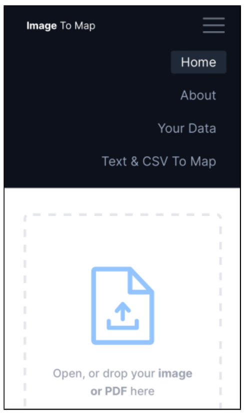
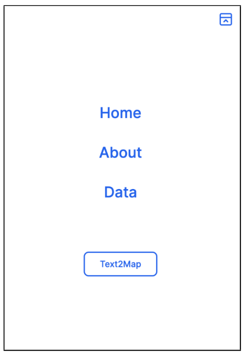

Consistent NAVBAR style, mby w/ some other links
IMG
PDF
2MAP
This services does a n thign and then you do the this. to get starteded, drop an filed
DROP
upload
perhaps an footer
check out tha other service, text2map

Figure B.33. *Simple sketch of the landing page*



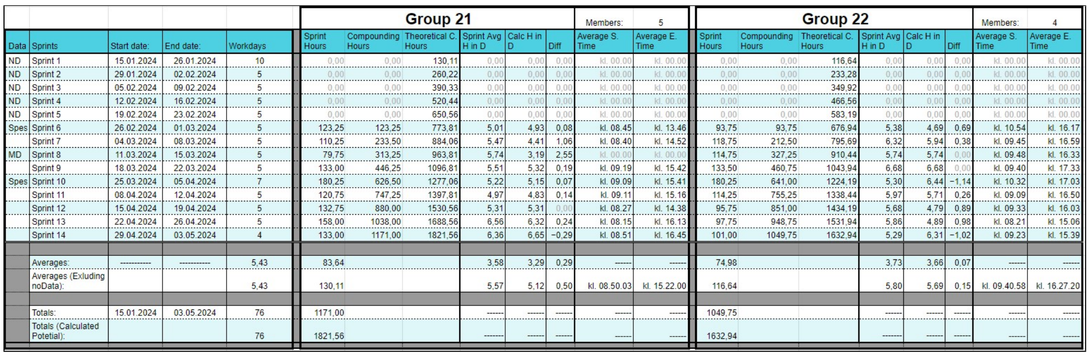
(a) First mockup of dropdown menu



(b) Second mockup of dropdown menu

Figure B.34. *Mockups of dropdown menus used on mobile devices*

B.5 Work Hours collective Data



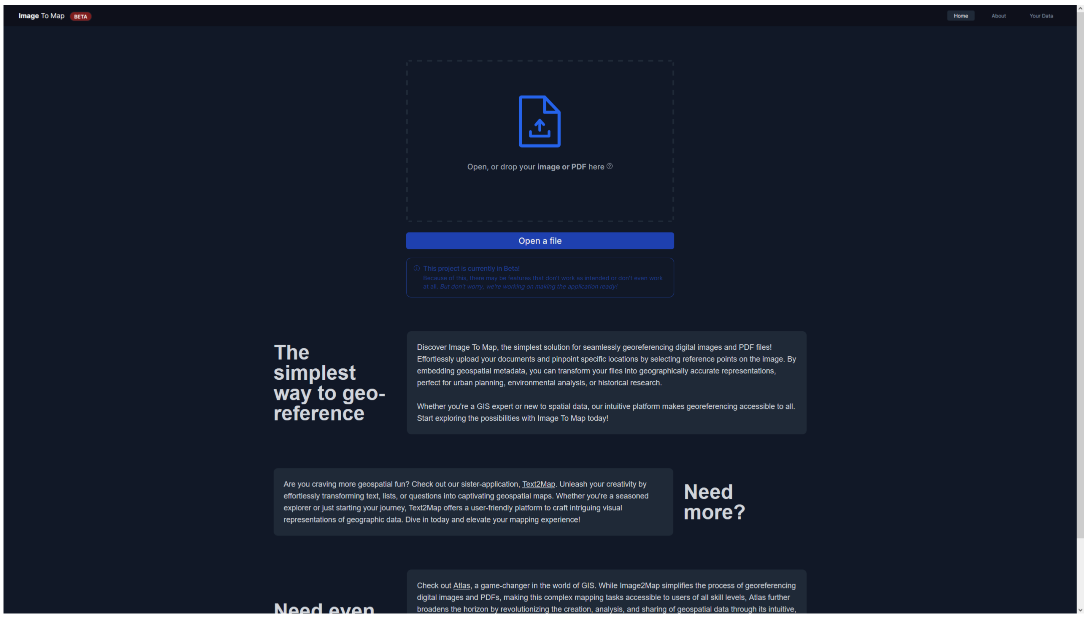
					Group 21						Members:	5
Data	Sprints	Start date:	End date:	Workdays	Sprint Hours	Compounding Hours	Theoretical C. Hours	Sprint Avg H in D	Calc H in D	Diff	Average S. Time	Average E. Time
ND	Sprint 1	15.01.2024	26.01.2024	10	0,00	0,00	130,11	0,00	0,00	0,00	kl. 00.00	kl. 00.00
ND	Sprint 2	29.01.2024	02.02.2024	5	0,00	0,00	260,22	0,00	0,00	0,00	kl. 00.00	kl. 00.00
ND	Sprint 3	05.02.2024	09.02.2024	5	0,00	0,00	390,33	0,00	0,00	0,00	kl. 00.00	kl. 00.00
ND	Sprint 4	12.02.2024	16.02.2024	5	0,00	0,00	520,44	0,00	0,00	0,00	kl. 00.00	kl. 00.00
ND	Sprint 5	19.02.2024	23.02.2024	5	0,00	0,00	650,56	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Spes	Sprint 6	26.02.2024	01.03.2024	5	123,25	123,25	773,81	5,01	4,93	0,08	kl. 08.45	kl. 13.46
	Sprint 7	04.03.2024	08.03.2024	5	110,25	233,50	884,06	5,47	4,41	1,06	kl. 08.40	kl. 14.52
MD	Sprint 8	11.03.2024	15.03.2024	5	79,75	313,25	963,81	5,74	3,19	2,55	kl. 00.00	kl. 00.00
	Sprint 9	18.03.2024	22.03.2024	5	133,00	446,25	1096,81	5,51	5,32	0,19	kl. 09.19	kl. 15.42
Spes	Sprint 10	25.03.2024	05.04.2024	7	180,25	626,50	1277,06	5,22	5,15	0,07	kl. 09.09	kl. 15.41
	Sprint 11	08.04.2024	12.04.2024	5	120,75	747,25	1397,81	4,97	4,83	0,14	kl. 09.11	kl. 15.16
	Sprint 12	15.04.2024	19.04.2024	5	132,75	880,00	1530,56	5,31	5,31	0,00	kl. 08.27	kl. 14.38
	Sprint 13	22.04.2024	26.04.2024	5	158,00	1038,00	1688,56	6,56	6,32	0,24	kl. 08.15	kl. 16.13
	Sprint 14	29.04.2024	03.05.2024	4	133,00	1171,00	1821,56	6,36	6,65	−0,29	kl. 08.51	kl. 16.45
	Averages:	----------	----------	5,43	83,64			3,58	3,29	0,29	------	------
	Averages (Exluding noData):			5,43	130,11			5,57	5,12	0,50	kl. 08.50.03	kl. 15.22.00
	Totals:	15.01.2024	03.05.2024	76	1171,00			------	------	------	------	------
	Totals (Calculated Potetial):			76	1821,56			------	------	------	------	------

	Group 22						Members:	4
Sprints	Sprint Hours	Compounding Hours	Theoretical C. Hours	Sprint Avg H in D	Calc H in D	Diff	Average S. Time	Average E. Time
Sprint 1	0,00	0,00	116,64	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Sprint 2	0,00	0,00	233,28	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Sprint 3	0,00	0,00	349,92	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Sprint 4	0,00	0,00	466,56	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Sprint 5	0,00	0,00	583,19	0,00	0,00	0,00	kl. 00.00	kl. 00.00
Sprint 6	93,75	93,75	676,94	5,38	4,69	0,69	kl. 10.54	kl. 16.17
Sprint 7	118,75	212,50	795,69	6,32	5,94	0,38	kl. 09.45	kl. 16.59
Sprint 8	114,75	327,25	910,44	5,74	5,74	0,00	kl. 09.48	kl. 16.33
Sprint 9	133,50	460,75	1043,94	6,68	6,68	0,00	kl. 09.40	kl. 17.33
Sprint 10	180,25	641,00	1224,19	5,30	6,44	−1,14	kl. 10.32	kl. 17.03
Sprint 11	114,25	755,25	1338,44	5,97	5,71	0,26	kl. 09.09	kl. 16.50
Sprint 12	95,75	851,00	1434,19	5,68	4,79	0,89	kl. 09.33	kl. 16.03
Sprint 13	97,75	948,75	1531,94	5,86	4,89	0,98	kl. 08.21	kl. 15.06
Sprint 14	101,00	1049,75	1632,94	5,29	6,31	−1,02	kl. 09.23	kl. 15.39
Averages:	74,98			3,73	3,66	0,07	------	------
Averages (Exluding noData):	116,64			5,80	5,69	0,15	kl. 09.40.58	kl. 16.27.20
Totals:	1049,75			------	------	------	------	------
Totals (Calculated Potetial):	1632,94			------	------	------	------	------

Figure B.35. *Collective data of work hours*

B.6 Final product images

B.6.1 Image2Map



Open, or drop your **image or PDF** here

This project is currently in Beta!
Because of this, there may be features that don't work as intended or don't even work at all. *But don't worry, we're working on making the application ready!*

The simplest way to geo-reference

Discover Image To Map, the simplest solution for seamlessly georeferencing digital images and PDF files! Effortlessly upload your documents and pinpoint specific locations by selecting reference points on the image. By embedding geospatial metadata, you can transform your files into geographically accurate representations, perfect for urban planning, environmental analysis, or historical research.

Whether you're a GIS expert or new to spatial data, our intuitive platform makes georeferencing accessible to all. Start exploring the possibilities with Image To Map today!

Are you craving more geospatial fun? Check out our sister-application, Text2Map. Unleash your creativity by effortlessly transforming text, lists, or questions into captivating geospatial maps. Whether you're a seasoned explorer or just starting your journey, Text2Map offers a user-friendly platform to craft intriguing visual representations of geographic data. Dive in today and elevate your mapping experience!

Need more?

Need even

Check out Atlas, a game-changer in the world of GIS. While Image2Map simplifies the process of georeferencing digital images and PDFs, making this complex mapping tasks accessible to users of all skill levels, Atlas further broadens the horizon by revolutionizing the creation, analysis, and sharing of geospatial data through its intuitive,

Figure B.36. *Landing Page of Image2Map*

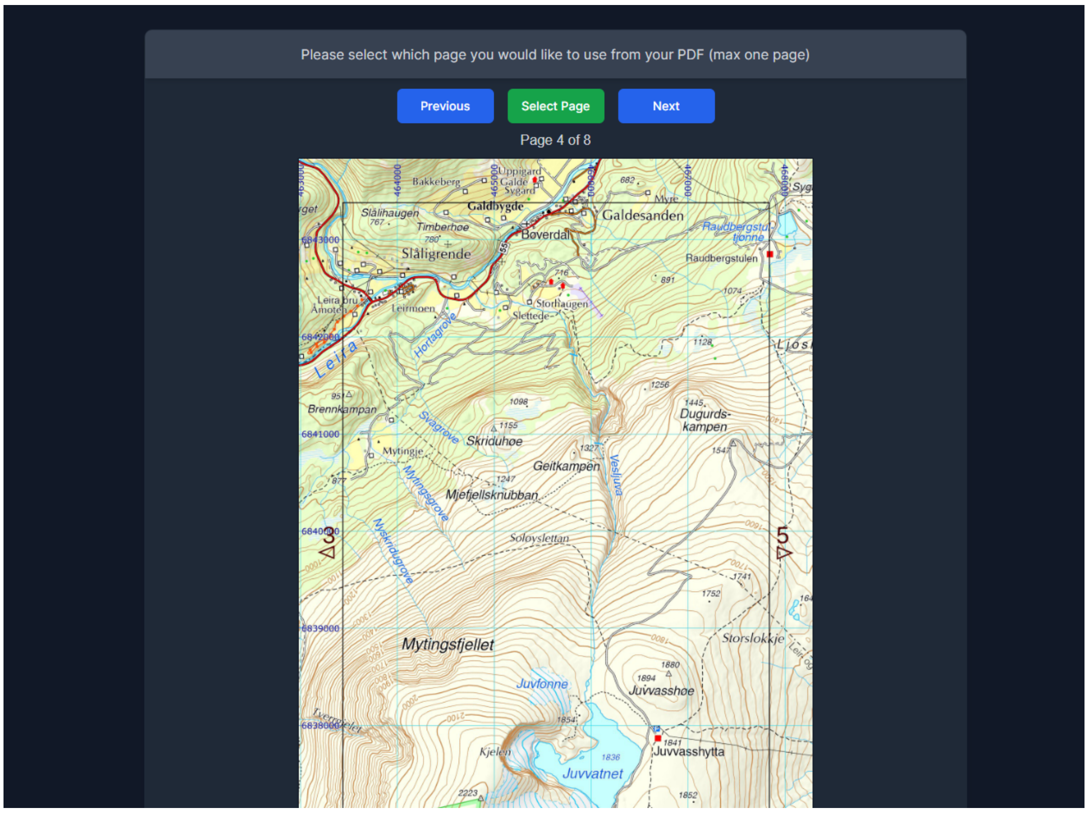
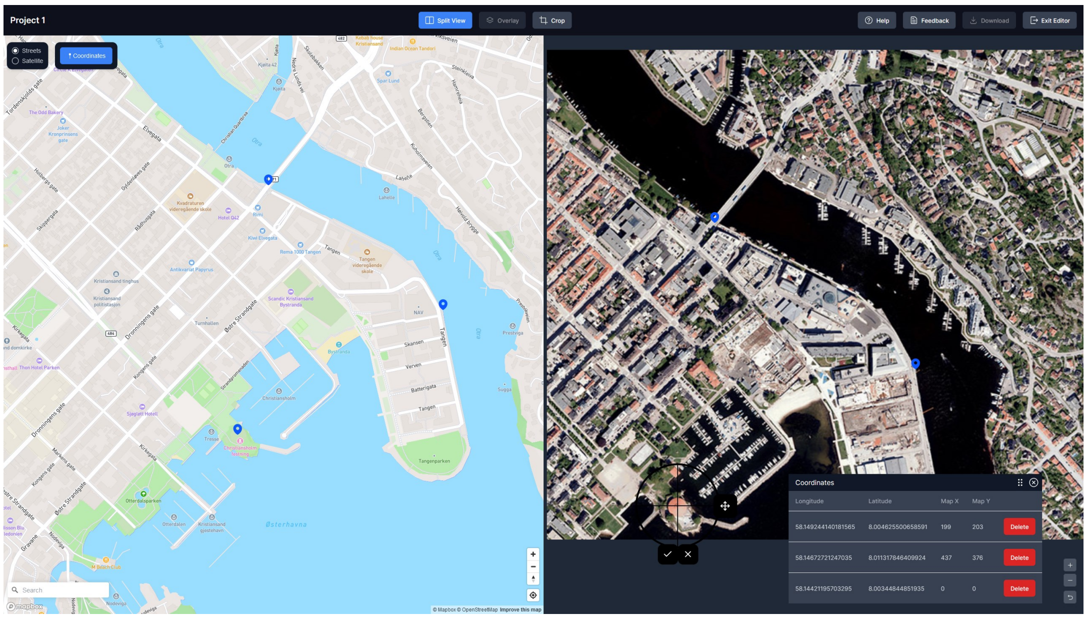

Please select which page you would like to use from your PDF (max one page)
Previous
Select Page
Next
Page 4 of 8

Figure B.37. *PDF Selector for Image2Map*



Coordinates

Longitude	Latitude	Map X	Map Y	
58.149244140181565	8.004625500658591	199	203	Delete
58.14672721247035	8.011317846409924	437	376	Delete
58.14421195703295	8.00344844851935	0	0	Delete

Figure B.38. *Split-View in Image2Map, One of the essential user view's for the app to function.*

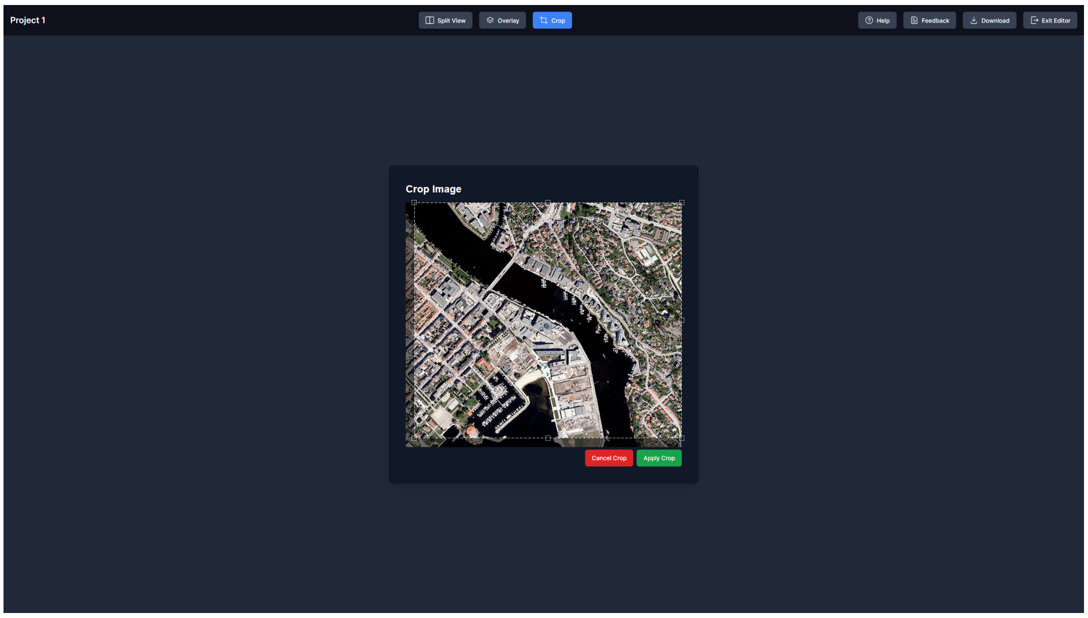


***Figure B.39**. Crop in Image2Map, Side tool for a user to adjust the image by cropping*

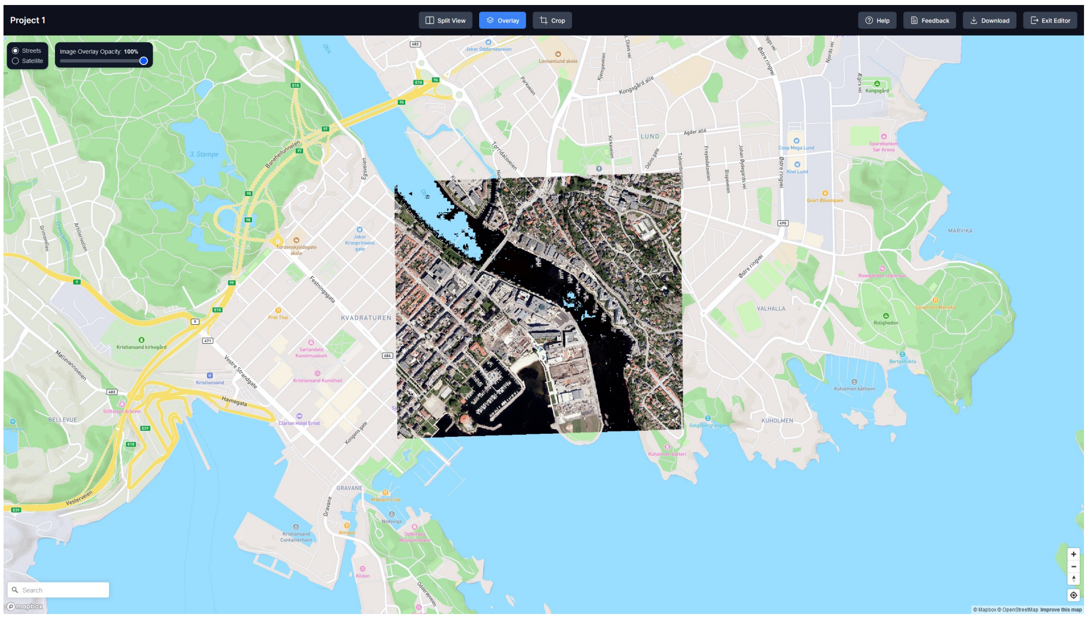
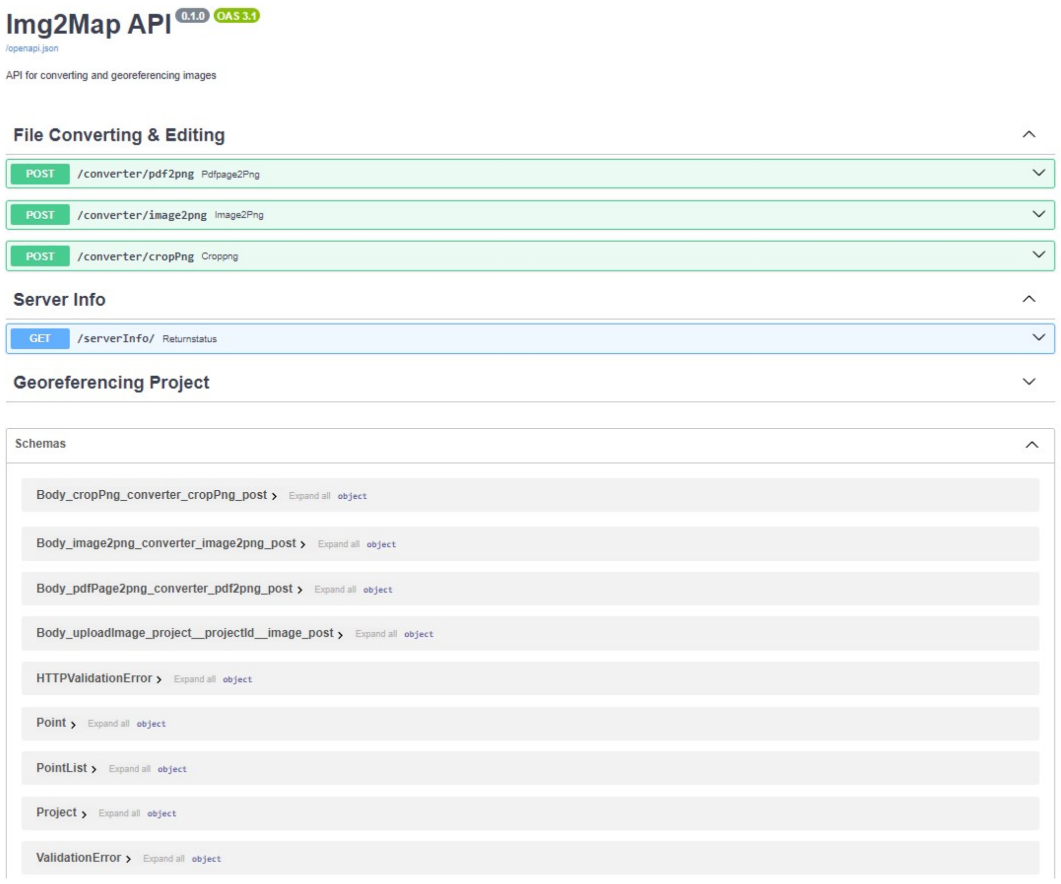


***Figure B.40**. Overlay-view in Image2Map, One of the essential user view's, where the user is able to validate the result.*



Img2Map API 0.1.0 OAS 3.1
/openapi.json
API for converting and georeferencing images

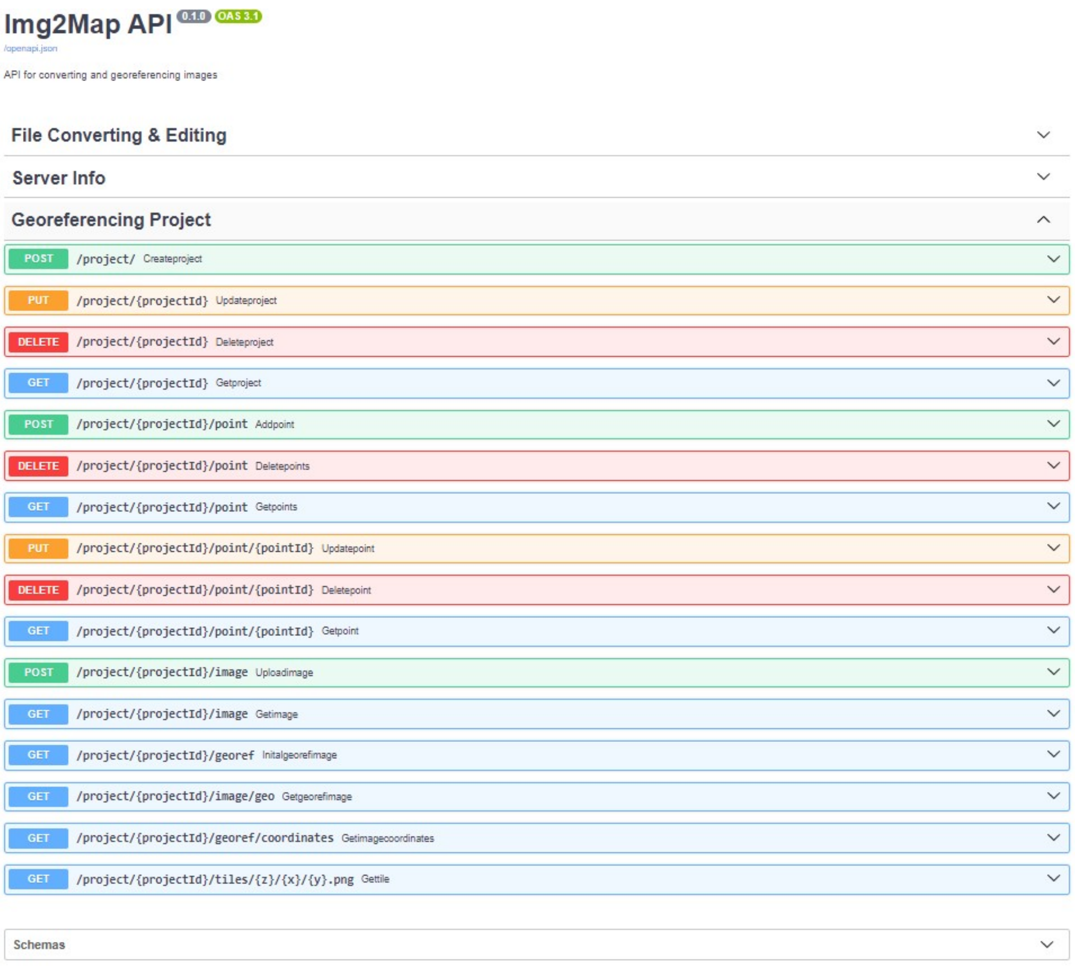
File Converting & Editing
POST /converter/pdf2png Pdfpage2Png
POST /converter/image2png Image2Png
POST /converter/cropPng Croppng

Server Info
GET /serverInfo/ Returnstatus

Georeferencing Project

Schemas
Body_cropPng_converter_cropPng_post > Expand all object
Body_image2png_converter_image2png_post > Expand all object
Body_pdfPage2png_converter_pdf2png_post > Expand all object
Body_uploadImage_project__projectId__image_post > Expand all object
HTTPValidationError > Expand all object
Point > Expand all object
PointList > Expand all object
Project > Expand all object
ValidationError > Expand all object

Figure B.41. *Docs (Swagger docs) of back-end endpoints in Image2Map, Number 1*



Img2Map API 0.1.0 OAS 3.1

/openapi.json

API for converting and georeferencing images

File Converting & Editing

Server Info

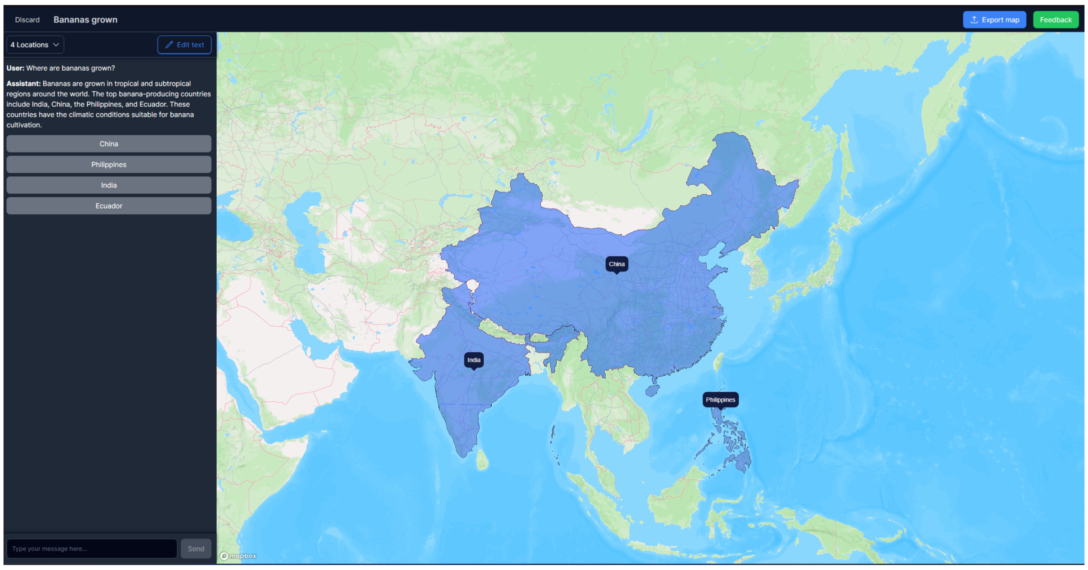
Georeferencing Project

```
POST    /project/  Createproject
PUT     /project/{projectId}  Updateproject
DELETE  /project/{projectId}  Deleteproject
GET     /project/{projectId}  Getproject
POST    /project/{projectId}/point  Addpoint
DELETE  /project/{projectId}/point  Deletepoints
GET     /project/{projectId}/point  Getpoints
PUT     /project/{projectId}/point/{pointId}  Updatepoint
DELETE  /project/{projectId}/point/{pointId}  Deletepoint
GET     /project/{projectId}/point/{pointId}  Getpoint
POST    /project/{projectId}/image  Uploadimage
GET     /project/{projectId}/image  Getimage
GET     /project/{projectId}/georef  Initalgeorefimage
GET     /project/{projectId}/image/geo  Getgeorefimage
GET     /project/{projectId}/georef/coordinates  Getimagecoordinates
GET     /project/{projectId}/tiles/{z}/{x}/{y}.png  Gettile
```

Schemas

Figure B.42. *Docs (Swagger docs) of back-end endpoints in Image2Map, Number 2*

B.6.2 Text2Map



User: Where are bananas grown?

Assistant: Bananas are grown in tropical and subtropical regions around the world. The top banana-producing countries include India, China, the Philippines, and Ecuador. These countries have the climatic conditions suitable for banana cultivation.

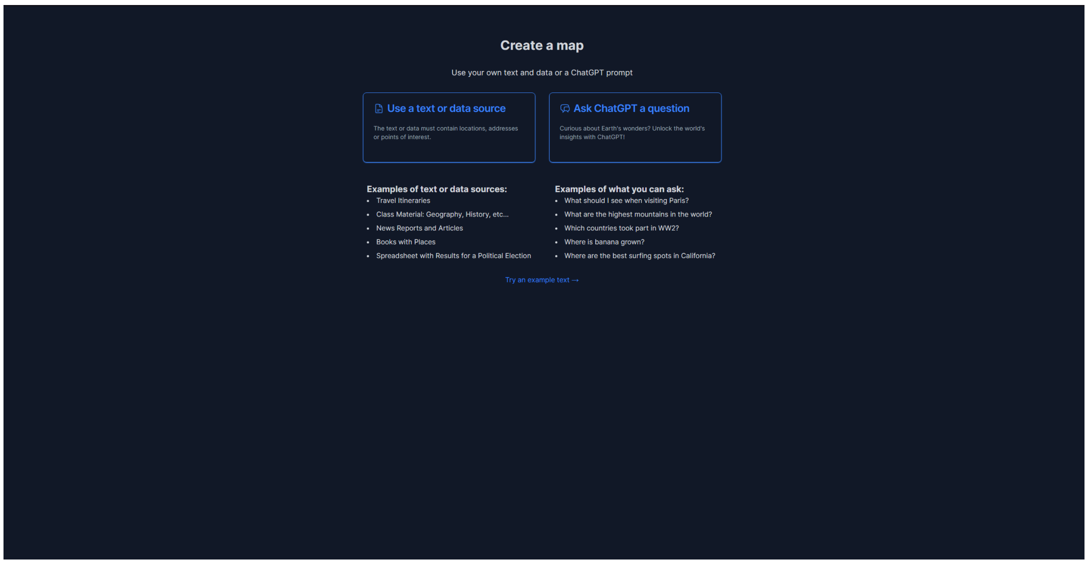
China

Philippines

India

Ecuador

***Figure B.43**. Split View after asking the GPT about where bananas are grown.*



Create a map

Use your own text and data or a ChatGPT prompt

Use a text or data source

The text or data must contain locations, addresses or points of interest.

Ask ChatGPT a question

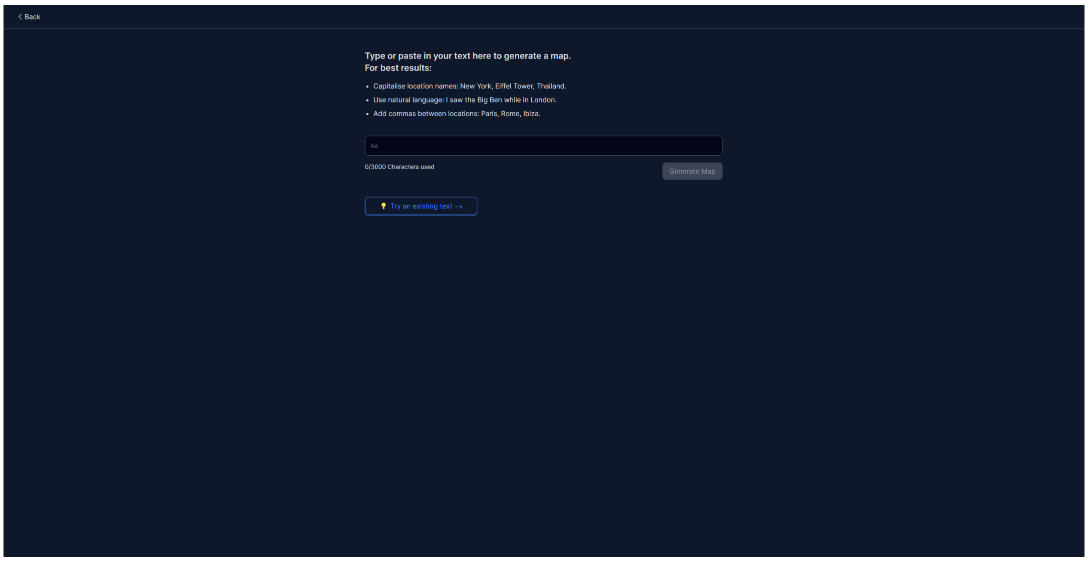
Curious about Earth's wonders? Unlock the world's insights with ChatGPT!

Examples of text or data sources:
- Travel Itineraries
- Class Material: Geography, History, etc...
- News Reports and Articles
- Books with Places
- Spreadsheet with Results for a Political Election

Examples of what you can ask:
- What should I see when visiting Paris?
- What are the highest mountains in the world?
- Which countries took part in WW2?
- Where is banana grown?
- Where are the best surfing spots in California?

Try an example text →

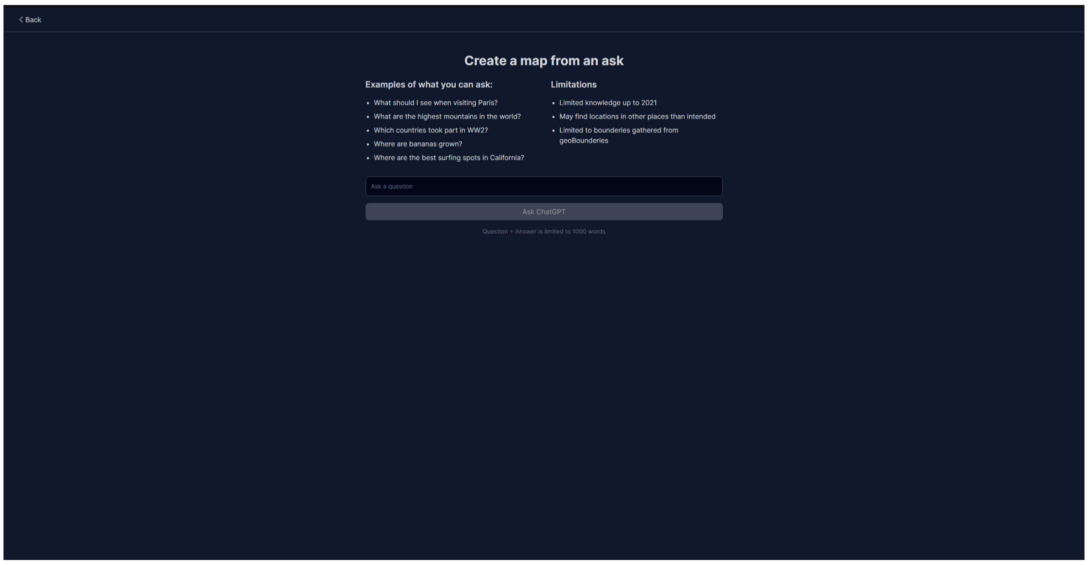
***Figure B.44**. Landing page for Text2Map*



Type or paste in your text here to generate a map.
For best results:

- Capitalise location names: New York, Eiffel Tower, Thailand.
- Use natural language: I saw the Big Ben while in London.
- Add commas between locations: Paris, Rome, Ibiza.

Figure B.45*. Landing page for the CSV part of Text2Map*



Create a map from an ask

Examples of what you can ask:

- What should I see when visiting Paris?
- What are the highest mountains in the world?
- Which countries took part in WW2?
- Where are bananas grown?
- Where are the best surfing spots in California?

Limitations

- Limited knowledge up to 2021
- May find locations in other places than intended
- Limited to bounderies gathered from geoBounderies

Question + Answer is limited to 1000 words

Figure B.46*. Landing page for the Text part of Text2Map*

Appendix C Daily Notes

This appendix has been removed from the published thesis due to privacy concerns. The important content from this Appendix can be found summarized in Chapter [6.1].

Appendix D Scrum retrospectives

This appendix has been removed from the published thesis due to privacy concerns. The important content from this Appendix can be found summarized in Chapter [6.1].

D.1 Sprint 1, 2, & 3

Appendix E Image2Map: Future development

This appendix contains future suggestions for further developing Image2Map, taken from the documentation in the projects code-base on GitHub (Group 21 & The Atlas Repository, 2024). This is a prioritized list of suggestions for improvements and enhancements to the application.

Authors: *Tom A. S. Myre, Markus Nilsen, Marius Evensen, Sebastian Midtskogen & Lukas A. Andersen* **Date:** 02.05.2024

This document is intended to provide information on the further suggested development after our handover to Atlas. As a guide, we will provide suggestions for general improvements, new features, and expansions.

Note: There are also enhancements listed in the GitHub-repo Issues, which are not mentioned here.

Defining the general structure of the suggestion, the top line will consist of a Title and a Type Category. The next line will have two fields/keywords that describe the estimated difficulty and our amount of already completed research & testing. Below is a list of the different categories and a short note on what these encompass, as well as an example.

1. Type:
   - **Front-end** *(The Next.js application / website)*
   - **Back-end** *(The FastAPI / processing / backend connected Services)*
   - **Fullstack** *(Encompassing both)*
   - **unknown** *(Unsure of where to place, or the current types are not descriptive)*
2. Difficulty:
   - **Low** *(Relatively easy, considering context)*
   - **Medium** *(Some time/difficulty expected)*
   - **High** *(Expected high difficulty and high time consumption)*
   - **unknown** *(We are not sure, and can not estimate time & difficulty)*
3. Researched:
   - **None** *(Only based on intuition, given our work with the project)*
   - **Low** *(Some testing done, and/or started researching)*
   - **Medium** *(Decent amount of testing done, and/or researched somewhat)*
   - **High** *(Good amount of testing done, and/or researched a lot)*

Example, Front-End:

- `Difficulty: Medium`
- `Research: Medium`

Description: What we suggest to implement, why we suggest this implementation, & possibly a theoretical suggested solution.

E.1 High priority

Point error calculation, Fullstack:

- `Difficulty: High`
- `Research: Medium`

Description: Calculate the amount of deviation from their real positions that the user's original points are. This is supposed to be done when users add more than the minimum amount of points to a map for georeferencing. This will then improve accuracy, and they would be able to see how far off their original marker placements were compared to the re-georeferenced points on the map.

A suggestion on this is to create an algorithm that creates an affine transformation matrix, to have more control of the calculation parameters. In this case, it must be accepted by Rasterio.

Another possible solution is to convert to the base use of GDAL for transformation, but this option has not been explored as thoroughly.

E.1.1 Good Starter

Logging Middleware, Front-End:

- `Difficulty: Medium`
- `Research: Low`

Description: Implement a standardized logging middleware to log crucial information during runtime, and generate better metrics. It would also be beneficial to have different log levels in the code to differentiate good development information and staged runtime logs. This is also a good way to get familiar with the codebase.

E.2 Medium priority

Rotation of Image with map, Fullstack:

- `Difficulty: Low`
- `Research: Low`

Description: This suggestion includes being able to synchronously rotate the image in split-view, along with the map after it has been georeferenced.

This is to make the image behave more like the map, and keep the image and map in a similar state while working.

Theoretically, this can be done by calculating the deviation from the real positions, after georeferencing, by determining the image's corner bounds. Then, it is possible to calculate the line towards the north from the center and adjust the starting point by rotating it to align with the north. Lastly, extract the rotation from MapBox and replicate it in the image view.

Subfeature: To be able to lock the image upright after triggering the minimum requirements for the feature above.

Refactor ProjectHandler, Back-End:

- `Difficulty: Medium`
- `Research: Low`

Description: Separate the ProjectHandler into smaller classes. The reason for this is to improve the current readability of the codebase, and that the ProjectHandler-class currently has too much responsibility.

A suggestion on our part would be to split the handling of points to one class and the handling of georeferencing to another.

A subsuggestion to the previous is to have the new GeoreferencingHandler know of the ProjectHandler, which in turn uses the PointHandler. The API-router will use all three.

Suggested points, Fullstack:

- `Difficulty: High`
- `Research: Medium`

Description: After the inital georeference, the software could suggest points on the map to be placed. This could greatly improve the user experience and the learning curve for new users of this kind of software. However, this might be a quite difficult task, as it would probably have to include some kind of Artificial Intelligence or advanced algorithm to achieve this at a good level.

In terms of an algorithm, this would then most likely depend on different code-changes and improvements, such as the error point calculations.

E.2.1 Good starters

A General HTTPExceptionHandler, Back-End:

- `Difficulty: Low`
- `Research: Medium`

Description: Create a common exception class to respond with the appropriate HTTP-exceptions. This is to consolidate the error response for the routers, which would improve readability. A possible way to do this is with Python decorators, a FastAPI dependency injection, or a combination of both.

Rotate Image, Front-End:

- `Difficulty: Low`
- `Research: Medium`

Description: Giving the user the possibility of rotating the uploaded image. This can be useful in cases such as when a user has a PDF where the image is rotated sideways, or when in general when images are not facing the classic north used in maps we are familiar with. This is not a feature we have previously prioritized, but is more prevalent now, and it should be relatively easy to implement.

Automated Testing, Back-end:

- `Difficulty: Medium`
- `Research: Low`

Description: The project currently lacks tests, so code-breaking changes are hard to identify. What would help mitigate this is to create unit tests for the core logic classes in the back-end API. This is a bit of work, but will make you familiar with the existing code. This also has the added benefit of a more thoroughly checked code, which can give more confidence to making changes and ensuring that the application works as intended.

E.3 Low priority

Clock to run on Heroku, Back-End:

- `Difficulty: Medium`
- `Research: Medium`

Description: The clock script's intention is to run periodic tasks on the database, specifically cleaning the storage of stale projects and files.

This would be a full implementation of projectSelfDestruct, where the clock would fetch from the database the projects that are over the time limit. For example, the clock can fetch every 10 minutes and delete projects that are above the limit. We also recommend setting a new self-destruct time each time a project is updated to prevent active projects from being deleted. Here is a source on Heroku clock setup.

Workers, Back-End:

- `Difficulty: High`
- `Research: High`

Description: Creation of a worker to offload more processing-heavy tasks from FastAPI. This is to prepare the application for more scalability and making it more redundant. A worker should handle all of the file saving, and creation and manipulation of projects. For communication between the worker and the API it is recommended to use a Message Broker.

A suggestion is to use a combination of RabbitMQ (Message Broker) and Celery to implement this. RabbitMQ might be time-consuming and complex to setup, but it includes built-in redundancy in the queues and several other features, hence the complexity. Additionally, there exists a Heroku Addon for a RabbitMq server, which has built-in metrics.

If Rabbit is too complex, one of the alternatives, Redis Queue, is more bare-bones and easier to setup. It has fewer features, but it is possible to set up with the features Rabbit has, but it is a very manual process to do this; this in turn could make the implementation of RQ more complex than implementing Rabbit.

Editing placed Marker Pairs (Points), Front-End:

- `Difficulty: Low`
- `Research: Medium`

Description: It could be beneficial for the user to be able to select a marker pair from the coordinates table, adjust them, and have them updated. This could increase usability, as the user might have confirmed a pair before properly checking the location, and might want to adjust either the coordinates (map), or the pixel placement (image). This is prioritized as low due since the user has the possibility to delete points.

The back-end endpoint for updating a Marker Pair already exists, and the data to do so is in the front-end, but the user interaction and flow are missing.

Multiple Image Georeferencing, Fullstack:

- `Difficulty: High`

- `Research: None`

Description: A suggestion that might be beneficial for the user is to have the possibility of uploading multiple images at once and georeference them together, and see them on the Overlay together. Another suggestion is for the user to upload multiple images at once and then work through them separately, like a backlog, before showing them all on the Overlay. This is to improve the workflow in case the user has multiple files they need to work with - rather than going back, uploading, georeferencing, checking, then downloading, the user could instead get a more seamless user-experience, starting and ending the process only once.

Georeferencing multiple images would require a more substantial change in the way the back-end stores file paths and how it handles file paths. Additionally, the front-end would also require substantial rework in how it manages and displays images.

Appendix F Text2Map: Future development

This appendix contains future suggestions for further developing Text2Map. This is a prioritized list of suggestions for improvements and enhancements to the application.

This document provides some thoughts and suggestions for future improvements for the application.

F.1 Change of Generative AI Model

As this software heavily relies on Generative AI and this always is changing and under development: It is important to constantly do research around the the idea of using different Generative AI models like Gemini, Lambda or a different versions of GPT. Gemini would probably be the best bet as it can be toggled to only return a JSON file, which is different from how the Assistant currently works as it is only asked nicely to return it as one.

AI Model	Pros	Cons
Gemini	Very good and natural replies. Can return directly as a JSON.	Expensive and currently very limited in how many countries it works in and how many times you can use it in a day.
OpenAI	Decent answers and cheap to run. Has no limits on how many times you can request a day.	Can provide unreliable answers, especially in formatting the JSON correctly. The assistant has fewer features than Gemini.
Meta Llama	Open Source and can be run locally.	While it can be run locally and is pretty quick, it gives more unreliable answers than OpenAI and fails most of the time to get the JSON structure right, which is crucial for the app.
Grok	Open Source and can be run locally.	Isn't meant for this kind of use case. Can be very political. Not suitable for the app.
Mixtral-8x7B	Open Source and can be ran locally. Has technically no limits if ran locally other then costs of runnning such a thing. Really good answer from testing.	Might be expensive to run if done locally. Haven't done to much research into it but doesn't seem to have an Assistant like OpenAI has

F.2 Prompt Engineering

More research into what to tell the Assistant also needs to be looked at further. Prompt engineering is just as important as developing the software and would and will lead to better responses from the AI, which would directly improve the general user experience and the reliability of the answers it gives. Currently this prompt seems to give the most reliable and best answers while maintaining the JSON Structure.

```
You are a text interpreter tasked with providing informative responses. You also do h

Final Output Requirement:

{
"Information: "Information",
"locations": [
    {
      "country": "ISO3",
      "state": "State or Region",
      "city": "City Name",
      "place": "Specific Place"
    }
  ]
}
Please don’t write anything in your reply outside of this JSON and keep every comma a
```

But I belive this can be further engineered to give even better answers.

F.3 CSV Feature

Currently the ChatGPT part of the App is the most fully fledged one. CSV need to be better implemented as the current solution is quite bare bones, and only really extracts locations from the text without giving much extra information or further explanations related to them.

This can be improved by changing the Assistant used for this part of the project (There are different assistants for CSV and Text2Map), in addition to changing the general formatting of the inputted prompt from the user if they decide to upload a CSV files, as the current solution for formatting might not look good.

F.4 Embed Map

This feature was requested so that users could easily embed their generated maps into their own websites or applications. This functionality has not yet been implemented, but to do so would in our opinion require two things:

1. For data to be stored somewere persistently.

2. Implement two dedicated page-displays for displaying stored data:

    - One with the text at the side of the map,
    - One with only the map.
    - Additionally without options for editing, and without the navigational bar at the top.

Appendix G Client statement from Atlas

This appendix contains a statement from the client, written in Norwegian, by Fredrik Moger. Additionally, it has been translated using AI into English.

G.1 Orignial Statement in Norwegian

PDF2Map og Text2Map har vært to utfordrende prosjekter med høy grad av teknisk usikkerhet og kun grove produktskisser ved prosjektoppstart. Til tross for dette har studentene levert to utrolig bra produkter ved prosjektslutt. Som oppdragsgiver har vi særlig observert tre egenskaper ved gruppene, som vi tror har vært sentrale faktorer bak prosjektresultatene.

Nysgjerrig: til tross for at studentene måtte ta i bruk ny teknologi som man tidligere ikke hadde brukt, var tilnærmelsen alltid preget av nysgjerrighet. De var uredde for å oppsøke kunnskap, spørre om hjelp, prøve og feile. Som programvareutvikler rir man en teknologisk bølge med høy fart, og det er umulig å allltid være oppdatert på ny teknologi. Evnen til å være uredd og nysgjerrig tror vi har vært sentral for å lykkes med prosjektene.

Teamarbeid: det har vært en tydelig rolleinndeling fra start, og det oppleves som kommunikasjonen internt har vært veldig god. Kommunikasjon med oppdragsgiver har vært presis og strukturert. På de ukentlige demoene har det alltid vært uttrykt positivitet og entusiasme fra teamet.

Teknisk kompetanse: selv om prosjektene har vært teknisk krevende, er det ikke tvil om at studentene innehar høy teknisk kompetanse. På mange måter er dette et biprodukt av det å være nysgjerrig og uredd. Vi er sterkt imponert over hva studentene har levert fra et teknisk perspektiv.

Alt i alt, stiller vi oss veldig positive til leveransen som har blitt gjort, og gleder oss til å følge studentene i tiden som kommer

G.2 AI translated statement

$EnglishChatGPTTranslate$

PDF2Map and Text2Map have been two challenging projects with a high degree of technical uncertainty and only rough product sketches at the project start. Despite this, the students have delivered two incredibly good products at the end of the project. As the client, we have particularly observed three qualities in the groups that we believe have been key factors behind the project results.

Curiosity: Despite having to use new technology that they had not previously used, the approach was always characterized by curiosity. They were unafraid to seek knowledge, ask for help, try, and fail. As a software developer, one rides a technological wave at high speed, and it is impossible to always be updated on new technology. The ability to be fearless and curious, we believe, has been crucial for succeeding in the projects.

Teamwork: There has been a clear division of roles from the start, and internal communication has been very good. Communication with the client has been precise and structured. In the weekly demos, there has always been expressed positivity and enthusiasm from the team.

Technical competence: Even though the projects have been technically demanding, there is no doubt that the students possess high technical competence. In many ways, this is a byproduct of being curious and fearless. We are deeply impressed by what the students have delivered from a technical perspective.

Overall, we are very positive about the delivery that has been made and look forward to following the students in the future.