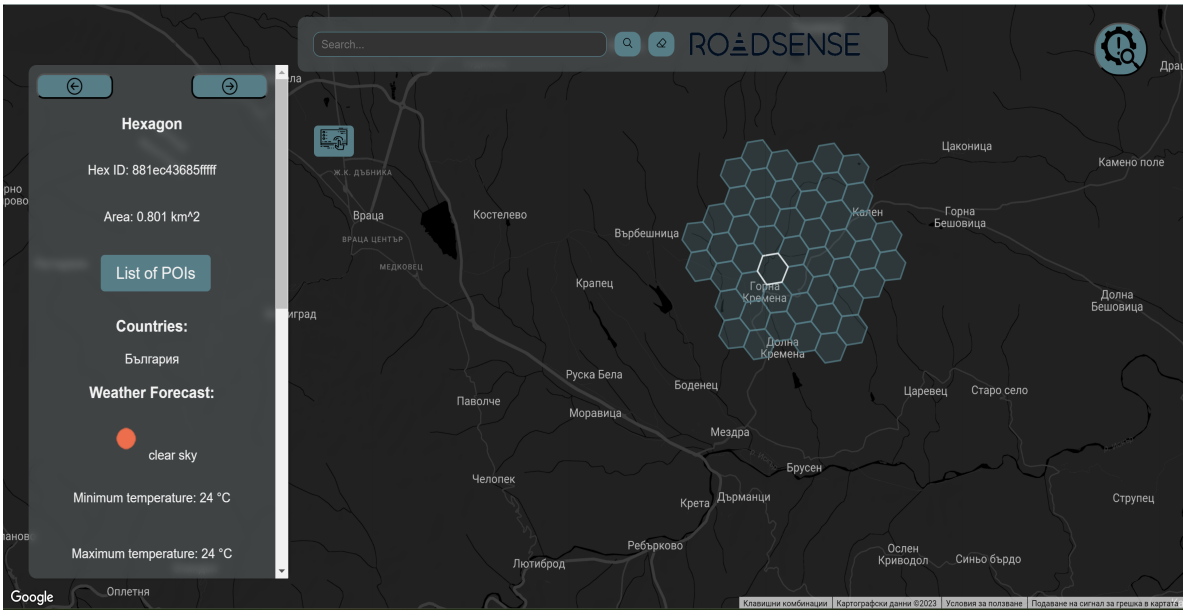
Data Explorer Map

Design and Implementation of a Real-Time Visualization Application



Group 16C
Ivan Virovski
Vladimir Pavlov
Dafni Pandeva
Dimitar Uzunov
Atanas Kichukov

Course: CSE 2000
Delft, June 2023
TA Supervisor: Ruben Backx
TU Coach: Ivo van Kreveld
SP Coordinators: Martin Skrodzki & Thomas Overklift & Otto Visser & Huijuan Wang
Client Supervisors: Viktor Kolev & Atanas Semerdjiev

Preface

This report was written by a group of 5 computer science students at the Delft University of Technology. For the software project in the final quarter of our second year, we developed a web tool named Data Explorer Map for our client, Digital Lights. Data Map Explorer, the subject of this report, is a map application that displays up-to-date road conditions and information about the surrounding environment in a specific geographical region.

For the readers, whether they are fellow computer science students, researchers, professionals or simply people interested in data visualisation and road transportation, we have structured the report in a way that allows them to follow and understand the development process and its steps with ease. While there may be a mention of some technical terminology at times, we have provided explanations and context throughout the report to ensure that no prior knowledge is required to grasp the key concepts of Data Explorer Map.

Readers that are particularly interested in the context of the problem and inspiration behind Data Map Explorer can find this information in Chapter 2. For readers with a technical background who are more interested in our design and implementation choices, Chapter 6 about our project approach provides a sufficient explanation.

Finally, we would like to express our gratitude to our client, Digital Lights, for entrusting us with this project and providing our team with invaluable guidance and feedback. In addition, we are also very thankful to our university advisors, our teacher assistant and project coordinator, for their supervision and advice, which played a role in the successful completion of this project.

Delft, June 2023

Group 16C
Ivan Virovski
Vladimir Pavlov
Dafni Pandeva
Dimitar Uzunov
Atanas Kichukov

Summary

Travel and transportation have always been fundamental to humankind throughout history. It deals with the moving of goods and people and is therefore directly correlated and dependent on a good infrastructure of roads and relevant environmental conditions. That is why it is important to have a good visualisation of up-to-date information on road qualities, incidents, and their effects. Having a tool to visualise the aforementioned data will let users such as the general public or companies that deal in the areas of transportation, travel and/or tourism to be able to make informed decisions when choosing a route.

This report will outline the development process of a real-time traffic, road conditions and hazards visualization web tool, named Data Explorer Map. The functionalities of Data Explorer Map are based on requirements that our team gathered through client meetings, research, and brainstorming, and that were later prioritized using the MoSCoW technique, which ranks based on importance. The tool makes use of Google Maps, an already widespread and detailed map with an open API, which serves as the foundation part that our team builds on. Essentially, Data Map Explorer displays a map over which relevant data about road conditions and other factors is shown in the form of hexagons via the h3 API for JavaScript, which is used to divide a map into hexagonal sectors and show them in different scales.

The first step of the development of Data Explorer Map was doing research of the problem domain. Through meetings with our client, the lack of accurate up-to-date data about road hazards, road conditions, infrastructure, and the surrounding environment was identified as the problem our team is trying. Our research included an in-depth analysis of stakeholders, of whom most important is our client Digital Lights, followed by an examination of potential use cases which are in the context of improved travel, tourism and traffic management. In addition, existing technologies that could be incorporated are Google Maps, Hivemapper and Helium Explorer. Essentially, Data Explorer Map takes inspiration from the above-mentioned popular map applications when it comes to user interface components. However, our tool goes beyond their basic functionality by adding new features that offer better accuracy and much more detailed and relevant information.

The next stage of the development included evaluation of project feasibility and requirement elicitation. The estimation of the project feasibility and its potential risk was done based on analyzing the time constraints, the available resources provided by the client and the need for prior expertise with frameworks and technologies. As a result of this analysis, we deemed the project as complex and potentially privacy-sensitive. Following, we defined the functional requirements, which describe what the tool should do in terms of features and functions, and non-functional requirements, which describe the general properties of the project, through meetings with the client, research, and brainstorming. Next, we employed the MoSCoW prioritisation technique to rank the requirements we came up with based on their importance, the most fundamental requirements being displaying a map with a hexagon layer and allowing the search for a specific hex or area for which various data is displayed.

Afterwards, we analysed the ethical issues related to the environment, health, safety and privacy and moved on to our project approach and implementation. Leveraging the data offered by the application yields numerous benefits, including enhanced travel safety, increased time efficiency, and reduced fuel consumption. These positive outcomes not only contribute to a healthier environment but also promote the well-being of individuals on the road. However, caution needs to be taken when it comes to storing user information to ensure privacy and data security and avoid misuse of the application. Next, we discuss important decisions regarding our project approach and implementation, made after evaluation of various options, in particular our choice of incorporating Angular, Google Maps, and hexagonal layers build with Uber's h3 geo library. To achieve successful implementation and completion of the project we followed the agile approach where we divided the requirements into development timeframes for the most efficient workflow.

In conclusion, our team was able to successfully deliver a fully functioning system that was up to the standards of our client by following the outlined development process. Apart from covering all must-have functionality, such as displaying hexagons within a map and seeing detailed information regarding road conditions, the application provides robust search and filter functionality for a given geographic region. Each step of the process allowed us to come up with design and implementation choices for our purposes while ensuring maximum work efficiency. Based on the project's insights and outcomes, it is advised to adopt a modular and flexible architecture for scalability such as the Angular framework and implement a process of continuous improvement to meet evolving needs.

Table of Contents

1 Introduction

From the earliest human settlements to the present day, roads have played an important role in connecting people and the transportation of goods, and services across vast distances (Stojanovik, 2021). However, the usability of roads is greatly influenced by various factors that can deteriorate their conditions and, in some cases, render them temporarily impassable. In light of these challenges, it becomes imperative to provide users with up-to-date information on road qualities, incidents, and their effects. By gaining insights, users can make informed decisions, optimise their routes, and ensure safe and efficient travel.

The purpose of this report is to describe the development process of the web tool, Data Explorer Map, which provides users with vital information about road conditions. Through visual representations, this tool aims to assist stakeholders in the fields of road infrastructure, commercial travel, and transportation of goods, by enabling them to assess road quality and make informed decisions. The functionalities of Data Explorer Map are based on the requirements of our client, our own research of the problem domain and brainstorming. The most fundamental of these requirements are displaying a map with hexagons and allowing for search of a region or a hexagon and showing detailed information about road conditions. To achieve this, the software utilises technologies to display a world map featuring all the roads, partitioned into interconnected hexagons. Each hexagon contains essential information regarding road quality within its boundaries. This design choice allows users to search specific areas and see further information about recent incidents, road quality and visibility of the selected part of the road, in the selected hexagon.

The report will be presented in the following structure. Chapter 2 delves deeper into the problem at hand, identifies potential stakeholders, and outlines the diverse use cases for the developed software. Chapter 3 deals with the feasibility of the project. Requirement elicitation and the methods used for it are located in Chapter 4. In Chapter 5 we discuss the ethical implications of the use of Data Explorer Map. The research conducted on technologies that will facilitate the implementation of the project is found in Chapter 6. In Chapter 7, we discuss the methodologies adopted by the team for project management, software testing and communication between team members and the company. Finally, the report is concluded with Chapter 8 where we review the key points discussed so far and propose future recommendations.

2 Analysing Problem Domain of Data Explorer Map

This chapter gives an overview of the purpose behind the creation of Data Explorer Map and analyses the related aspects and issues that our team is trying to address with this software system. Section 2.1 outlines the main problem that inspired the project. Following, section 2.2 describes the product stakeholders and their interests. Next, section 2.3 provides several potential use cases of Data Explorer Map. Finally, section 2.4 describes the already existing technologies similar in functionality to the one desired of Data Explorer Map.

2.1 Problem Statement

The problem that our team is aiming to solve with this project is the lack of accurate and up-to-date information about road hazards, road conditions, infrastructure, and surrounding environment within a specified geographical region and timeframe. The absence of accessible current data of these aspects can negatively affect transportation and travel in terms of inefficiency, stress and safety risks. The stated problem is an issue that our client wishes to address with the help of our team and is in accordance with their aspiration that the representation of this information could be helpful in the context of road infrastructure, research, travelling and transportation sectors and could potentially lead to future improvements in the previously mentioned fields.

2.2 Project Stakeholders

Having stated the problem that drives our project, we will now explore the stakeholders involved in the development of Data Explorer Map. Both direct and indirect stakeholders can be identified when it comes to the impact of the existence and functioning of Data Explorer Map. Direct stakeholders are those who are directly involved in the project and have a direct interest in its outcomes, in this case the client and the developers. Indirect stakeholders are those who are affected by the project but are not directly involved in its development or implementation, mainly transportation companies, the general public, data analysts and researchers and the government.

The first direct stakeholder is our client, Digital Lights, which is a software company that is directly involved in the creation of the product in terms of defining requirements and expectations. Their main interest is receiving a functional product that satisfies their needs and solves the problem we previously formulated. Our client expects a user-friendly interface that provides a visualisation of accurate and up-to-date information, which is easy to interact with. As our most important stakeholder, they have a big impact on the functionality and design choices of the app, since our team will strictly stick to our client's requirements and preferences, described in detail later in Chapter 3, to ensure they are being satisfied.

The second direct stakeholder in the project are team members/ software developers (i.e. our team). Our main interest is creating an effective and viable product that satisfies our client since we are directly involved in the development, design, and implementation of the Data Explorer Map tool. Our role is also very significant since the development of the application depends on our expertise, creativity and work ethic. As developers, we value efficiency of functionality and usability and we expect to create a high-quality product. In addition, the development and the maintenance of the application has the potential to open new workplaces in the future.

One of the first indirect stakeholders we can identify are the transportation companies (of both people and goods). They are considered as potential clients and users of the system when deployed, since they are highly likely to be interested in the insights that the Data Explorer Map can provide regarding road conditions such as road surface quality, traffic, safety hazards on the road, weather conditions, road infrastructure, etc. Given this information, the companies can make the best possible choices when it comes to transportation routes in terms of time efficiency and quality of journey. As potential users, they value reliability and usability of the application. It is important to keep these values in mind as well when developing the product, since we want our product to be able to satisfy the intended users of the system, other than our client.

A second indirect stakeholder, we can identify is the general public. It can range from travel enthusiasts to just ordinary people with private means of transportation. Similarly to the transportation companies, the general public would expect to be able to make informed decisions about their travel using the visualised data given by Data Explorer Map. Even though our client is the top priority when

it comes to satisfying their needs, it is useful to consider the general public, since our product has the potential to reach them in the future, so we would like to ensure that our product is easy to use.

A third indirect stakeholder we can identify are the researchers. Data Explorer Map can provide valuable insights to researchers interested in driving conditions and patterns across different regions and timeframes. Due to the variety of parameters captured by the tool and the ability to view the data across numerous timeframes, researchers can greatly benefit from the product when it comes to conducting research in the field of transportation, automotive and road engineering, urban planning, etc. Their needs align with the ones of our client in terms of having access to detailed data on a lot of factors within a specific geographic region and timeframe and being able to interact with it easily.

Another indirect stakeholder we can consider is the government. The government generally has a big interest in road conditions and infrastructure. From their point of view, information regarding road conditions is valuable, since it can aid in making decisions about road infrastructure and improvements. Additionally, as developers, we also have to make sure that our product does not violate any laws and regulations.

Identifying the direct and indirect stakeholders described above and analysing their interests, values and expectations is an important step of the development process of Data Explorer Map. It allows us to make design and functionality decisions that align with their needs and requirements, resulting in a more successful and impactful implementation of Data Explorer Map.

2.3 Use Cases

Data Explorer Map can be used in a variety of cases. The first and most obvious one is in the context of travel and tourism. In this case, the software tool can be used to analyse the road conditions of popular routes and help the travel companies, tourism agencies or individuals, who would like to travel, to choose the best possible route in terms of time efficiency, safety, journey quality, etc. In addition, the authorities could use the tool to improve public safety since they can monitor and identify high-risk areas where accidents occur often, allowing them to take preventative actions to reduce the frequency of accidents.

Another use case of the tool is the management of traffic. In general, there are 3 phases in building traffic management systems: information gathering phase, information processing phase, and the service delivery phase (de Souza et al., 2017). Data Explorer Map can be extensively used in the first phase to gather traffic-related data since it will provide information about a wide range of parameters, which would be the first step to building an effective traffic management system. In this way, the tool can be used by authorities or urban planners to monitor traffic activity and congestion and identify bottlenecks, which can further contribute to making more informed decisions about traffic flow management, road construction, road maintenance, etc.

2.4 Existing Technologies

There are a few existing systems that cover some of the use cases of Data Explorer Map. The first and probably the most widely-used one, especially by the general public, is Google Maps. In fact, according to the latest statistics, Google Maps is the most popular navigation app in the world (Popa, 2022). Google Maps covers to a large extent the functionality of providing information about accidents and traffic conditions and this information is usually real-time. Google Maps can also indicate possible generalised roads hazards such as slippery roads or other special weather statements that can affect the journey, but it does not provide specific road conditions (Johnson, 2020). Google Maps also shows the quality of the road in terms of the existence of bumps, but this sort of data is very general. In contrast, Data Explorer Map aims to give a very detailed view of problematic areas and aims to visualise road issues with far greater precision. Additionally, the main focus of the tool is not navigation, but rather the visualisation of data with a number of different parameters. However, as a whole, Google Maps is worth considering as an existing technology that can be incorporated in the same context as it can be very useful and helpful for the implementation of the tool since it uses map and geographic information and collects data from users in real time, which is a very valuable feature (Mehta et al., 2019). Essentially, Data Explorer Map will build on the functionalities of Google Maps in terms of providing and visualising much more detailed data as well as more active involvement of the user in the process.

Two other existing technologies that closely cover the functionality of Data Explorer Map are Hivemapper and Helium Explorer. Both Hivemapper and Helium Explorer have a concept similar to our project, namely, they deal with a visual representation of data and can be used to monitor and analyze location-based data[1]. This is also one of the main ideas behind Data Explorer Map, but the difference here is that Data Explorer Map will strive to visualize data in much more detail, with greater number of parameters regarding road conditions and the surrounding environment. So as a whole, the functionalities of both Hivemapper and Helium Explorer will be very useful for the creation of Data Explorer Map. The next sections of the project plan describe in more detail what and how they will be incorporated.

[1]Helium and Hivemapper are both mapping technologies which model points of interest on a map and use the hexagonal architecture to emphasize the intensity of regions.

3 Feasibility Study & Risk Analysis

This chapter consists of feasibility study and risk analysis of the development of Data Explorer Map. Section 3.1 will discuss the feasibility of the project in terms of realisation within the given time constraints, the available resources provided by the company, the prior preparation required for the project and a possible fallback plan in case there is no outlook for entirely finishing the project. Section 3.2 will analyse the factors that pose risks for the completion of the project or for its users.

3.1 Feasibility Study

Firstly, in terms of feasibility, we deem the project as generally doable with proper time management and a well-structured plan. We consider the project completed when all the must-have and should-have requirements have been implemented before the deadline. The development of the web tool required by our client is a complex task as it demands knowledge and a sufficient level of proficiency in unfamiliar libraries and frameworks such as H3 and the Google Maps API. In addition, the Angular framework, which will be used for creating a good and functional user interface, also requires a thorough understanding. On top of that, the project will be written in Typescript which is unfamiliar to our team and would also take time to learn. With this in mind, the learning curve will be quite steep and time-consuming in the beginning. However, we are confident about successfully finishing the project with the right time management and a well-formulated work plan.

Secondly, regarding the resources at our disposal, our client being an IT company offers technical support. They have already conducted research, have a well-defined concept of the product and offer guidance when requested. For example, the client provided a lot of clarification when we were discussing the implementation of the hexagon structure on the map. Additionally, we have carefully reviewed the data necessary for testing our project's functionality. We have reached an agreement with the company, under which they will provide us with sample JSON files. These files will be instrumental in testing and will also serve as mock data, enabling the visualization of the points of interest on the map. This arrangement is crucial in ensuring that our project is effectively validated and refined. Overall, under these favourable circumstances, we deem that we have the resources and support we need to complete the project.

Thirdly, in case the project proves to be infeasible at some point, we have come up with a fall-back plan to adapt and adjust our approach. This plan involves reducing the UI components based on their MoSCoW prioritisation. If a further reduction is required, we plan to further relieve the load of the project by relinquishing features based on their weights assigned in Gitlab. Implementing these strategies during the planning and/or development will lighten the workload and allow us to adapt and keep intact the very core functionality and features of the project.

3.2 Risk Analysis

Regarding the risks when approaching such a project, one of the most common risks is how responsive a client is in terms of product details and demands or feedback on development. However, our concerns on that matter are not significant since during the first week of working together, we have had several meetings with the client and have been assigned an employee of the company to supervise the project and to be our connection with the client. So far, they have responded quickly and seem readily available to answer questions and schedule additional meetings if needed. In addition, during the meetings with the company, we have been presented with different possibilities of getting the data needed for testing, namely getting access to the backend and manually sending instructions to the front end or being provided with sample files to simulate the communication with the backend. This can also be deemed as a potential risk and pose a problem if this data is not provided on time as it is essential to the development, visual representation of progress and testing.

Privacy remains one of the main risks and ethical issues, even though the only user information we have access to is the user ID. Our client has made sure that the user ID remains anonymous, however, there's still a possibility of de-anonymization. Malicious agents might monitor and analyse the data on the map. They can extract various patterns and possibly deanonymize certain users if they identify people's travel habits. Another potential risk concerning privacy is that sensitive data concerning clients might unknowingly be shared through the project and spread around.

4 Requirements

This chapter will outline the methodologies employed to elicit the project requirements, as well as the strategies used to prioritise them effectively within the given timeframe. In section 4.1, we will elaborate on the process of deriving our own requirements through research and collaboration with the client. Subsequently, in section 4.2, we will delve into the implementation of the MoSCoW prioritisation technique, which facilitated the ranking of requirements based on their significance to the client and feasibility within the designated project time frame. For a comprehensive overview of the MoSCoW prioritisation list, please refer to the Appendix.

4.1 Requirement Elicitation

To gather the requirements for our project, we conducted meetings with the company to gain a deep understanding of the project's objectives and determine its essential features. Through collaborative efforts, we successfully identified the core and non-functional requirements and enhanced our comprehension of the project's overall purpose. This improved understanding enables us to further elicit requirements and propose them to our client.

Through collaborative brainstorming and drawing insights from similar projects, our team has identified additional requirements for the project and enhanced the core features previously elicited in collaboration with the client. Initially, our team and the client agreed to visualize hexagons based on the map's zoom level, aiming for increased detail as the user zoomed in, and rendering less hexagons when viewing further away, to save on resources. However, we discovered that this approach was visually unappealing and inefficient. Parsing a single Point of Interest to multiple hexagon resolutions posed challenges in terms of resource consumption and the constant changes of the hexagons when zooming in and out proved to be irritating for the eyes. The past implementation of the visualization for hexagons can be observed in Figure 1, which includes two images illustrating different zoom levels. On the left side, we have the representation of our implementation when zoomed out, while the image on the right showcases the visualization when zoomed in.

After thoughtful deliberation and consultation with the client, we proposed an alternative solution for hexagon and Point of Interest visualization, aiming to improve the user experience and optimize resource utilization. Our team decided to display a fixed hexagon resolution that remains consistent across different zoom levels. We carefully selected a resolution that captures a small area while accommodating multiple Points of Interest. This adjustment provides our web tool with a more visually pleasing appearance and reduces the time required for hexagon visualization since they are processed at a single level. The final improved version can be observed in Figure 2, which shows all the hexagons that contain a Point of Interest. The hexagons in this implementation maintain their relative positions regardless of the zoom level of the map.

Through several productive brainstorming session, our team identified several functionalities that align with the project's goals. One crucial feature we devised is the search functionality, enabling users to search for hexagons, Points of Interest, and users by their respective IDs. Another significant feature we elicited is the filter functionality, allowing users to narrow down displayed hexagons on the map based on the selected Point of Interest type. These two features are considered essential and categorized as Must Have for the project.

In addition, our team generated Should Have and Could Have requirements. These include coloring hexagons differently based on the number of Points of Interest contained within them, implementing region-based search functionality, and displaying a heat-map over the geographical map when zooming out significantly. We carefully evaluated all elicited requirements and assigned priorities using the MoSCoW prioritization method.

Having thoroughly assessed these requirements, we presented them to the client for their consideration and feedback. Based on the feedback received from the client, we carefully reviewed and refined the elicited requirements, incorporating their suggestions and modifications. This feedback served as valuable guidance, enabling us to reevaluate the requirements and propose new features that align with the project's goals and objectives.

Additionally, during the product development phase, our team identified several valuable features. One notable example is the implementation of a back and forward button functionality. This feature allows users to easily navigate between previously searched type of data by their ID and view the corresponding information in the infotainment panel.

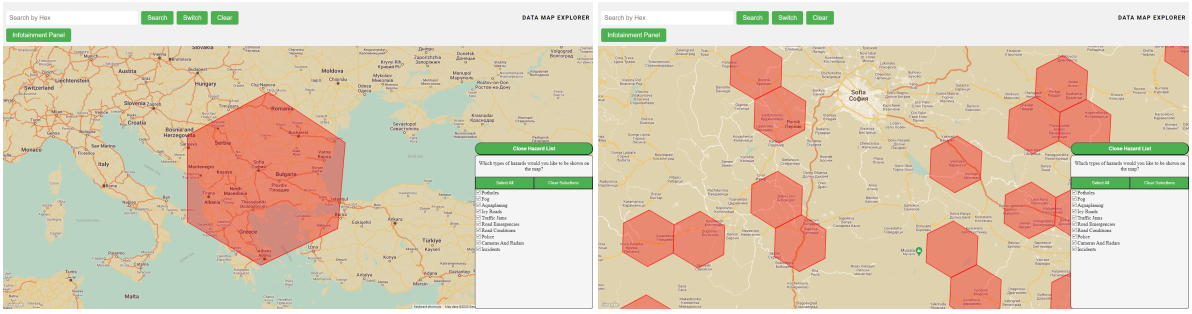


Figure 1: Past implementation of Hexagon visualization

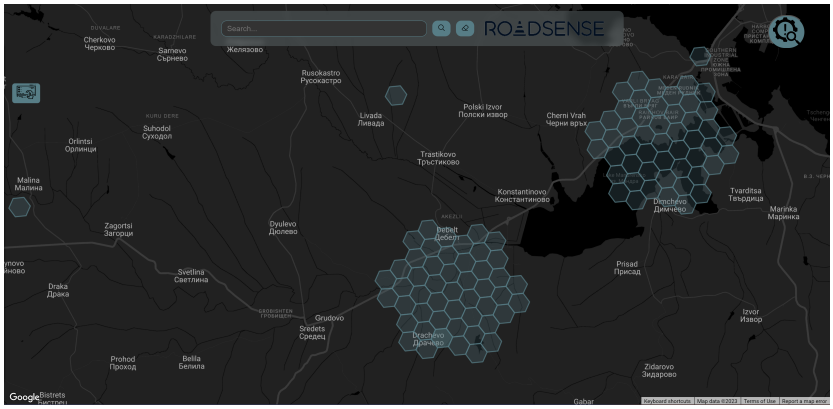


Figure 2: Final implementation of Hexagon visualization

4.2 MoSCoW prioritisation

Using the MoSCoW prioritisation technique, we ranked the requirements in order of importance and feasibility for the given time frame. The "Must-Have" requirements were those that were absolutely necessary for the project to be considered successful and were given the highest priority. The "Should-Have" requirements are important features that are not absolutely necessary but would greatly enhance the project if implemented. The "Could-Have" requirements were desirable features that could be implemented if time allowed. Lastly, during our requirement elicitation process, we identified certain features as "Won't-Have" requirements. These are functionalities that we determined to be outside the scope of the project's goals or beyond the feasible timeframe for implementation. While these features may have potential value, we made a conscious decision not to include them in the current project to ensure a focused and timely delivery of the core objectives.

Non-Functional requirements were derived from meeting with the client and researching technologies, frameworks and libraries that would benefit the development of this application. As described in the Project approach chapter, we reached the conclusion to use Angular, which makes use of TypeScript, HTML and CSS, to create the web-tool project. We chose to use Google Maps as our map provider, to implement the map in which we will display all the hexagons, which will be implemented with the H3 library.

After deriving all the requirements for the application and ordering them in the MoSCoW principle, we consulted with the company and reached a consensus for the requirements list and the priority of the tasks, so that the client is satisfied and the most important functionalities like the "Must-Haves" and "Should-Haves" can be implemented in a timely and efficient manner. However, some functionalities could be changed mid production, since the company itself stated they have not fully decided on some features and we might have to change a feature's functionality, if the company tests it and decides a different way to implement a feature will be more beneficial to the project's goal.

5 Values

In this chapter, we will explore the ethical considerations associated with the design and implementation of a software application that visualizes traffic conditions and the surrounding environment. In section 5.1, we explore the ethics in regard to the environment, health and safety, and in section 5.2 we discuss any issues that might occur when processing personal data.

5.1 Environment, Health and Safety

The design and implementation of a software application that visualizes traffic conditions and surrounding environment introduces several ethical issues related to the environment, health and safety of its users. From an environmental perspective, our application plays a significant role in promoting more sustainable practices for road navigation. One of them is letting the users have access to real time traffic and road conditions. This way they can considerably reduce their fuel consumption and emissions by avoiding traffic jams and choosing more optimal routes.

The application also has many benefits for the health and safety of its users. For example, it has a feature that alerts users about road hazards, like potholes, accidents, fog and icy roads. By providing users with this crucial information, they can make informed decisions and not face the same issues. In this way they can optimize their routes and ensure safe and efficient travel, which will lead to fewer accidents and injuries. However, this can pose a heavy ethical obligation, as strong measures should be taken in order to prevent misinformation that could lead to harm and in order to ensure the accuracy and timeliness of the data as a whole.

Another health and safety issue that we have considered is the problem of getting distracted while driving. Since our application has a lot of UI elements, it may require a lot of attention to make sense of the data, which can make the use of the application while driving a potential road safety hazard. Because of that we strongly advise our users to not use our application while operating a vehicle. The whole premise of the project is to display thorough information and ensure nothing is missed when considering road conditions. With this said, mobile usage is not a relevant use case in our sight and we make sure that this is transparent to our users.

5.2 Personal Data

There are various ethical issues that need to be taken into account when implementing this front-end application. Privacy remains one of the main ethical issues, even though the only user information we have access to is the user ID. Our client has made sure that the user ID remains anonymous; however, there is still a possibility of de-anonymization. Malicious agents might monitor and analyse the data on the map. They can extract various patterns and possibly deanonymize certain users if they identify people's travel habits (Wheatley et al., 2016).

Additionally, it is possible that certain users might be unfairly subjected to discrimination using the system. For instance, if insurance firms or law enforcement exploited the app's data, it may result in higher insurance premiums or targeted policing. One might wonder if any recorded information may include traffic bystanders who do not comply with the terms and conditions of the application. However, a user can only report information that is directly extracted from his or her vehicle and not from other participants in the traffic. Any reported traffic condition, such as ice, road hazards, police, etc., is too general to obtain any information about an individual. Another important factor is data security. The stored data is not considered sensitive; however, any unauthorised breach or data leak might still be used to harm users, the company, or third parties. We need to clearly state how the data will be stored, used, and shared and make sure that this is clearly disclosed to the users so that we can mitigate any potential problems in the future.

The best way to deal with personal data issues is to include information regarding possible uses, for example, in the "Terms and Conditions" section. By doing this, the company does not need to reduce any of the functionality, and any responsibility for using the application lies solely with the end user. Principles such as maintaining privacy, guaranteeing data security, and adhering to transparency are all crucial factors in developing an ethical product.

6 Project Approach

This chapter introduces the solution to the problem identified in Chapter 1, regarding the absence of precise and current information regarding road conditions and the surrounding environment, available to the public. In section 6.1, we discuss the development of a web application which tries to solve the problem in question. In section 6.2, we elaborate on the technologies we have used and provide an explanation of why we have chosen this exact set of frameworks and libraries.

6.1 Problem Solution

The solution to the problem mentioned in Chapter 1, namely the lack of accurate and up-to-date information about road conditions and the surrounding environment, can be realised through the creation of a web tool, named Data Explorer Map. Data Explorer Map can be used for visual representation of data collected by moving vehicles regarding the above-mentioned road conditions and surrounding environment. The data collected should be analysed, processed and presented as a specific field or parameter of a region on the map and it should be easy to update, navigate and search according to location and timeframe.

6.2 Choice of Software Technologies

The first design decision our team made was to decide which front-end framework to use when building the foundation of the web-tool. Selecting a suitable framework is an essential task that requires a thorough analysis of a few critical features. The first key aspect to consider is the learning curve associated with each framework. The level of difficulty in mastering a new technology can significantly affect the development timeline and overall project success. Second, the scalability of the framework is crucial. As the project grows, the chosen framework should be capable of meeting its needs. The third consideration is the framework's popularity and community support. A community contributes significantly to more rapid troubleshooting, availability of more abundant resources, and facilitated learning as more documentation and tutorials are available. The fourth feature to consider is the performance capabilities of the framework. Performance optimization is crucial to ensure user satisfaction. Finally, the flexibility and ease of integration with other libraries and frameworks can also influence the choice of a front-end framework (Xing et al., 2019).

Given these key considerations, Angular stands out as an ideal choice for several reasons. First, Angular relies on component architecture, which encourages reusability and modularity. This simplifies the project and increases scalability when dealing with big projects. This means that components can be treated as standalone programs, each with a clearly defined purpose, and each of the team members can work on one without being overly dependent on each other. Moreover, Angular has two-way data binding, which drastically cuts down on the boilerplate code required to synchronise the display and model (Bucea-Manea-Tonis, 2016). Being one of the biggest frameworks and backed up by Google, this framework includes a lot of libraries that speed up development. Tasks such as code generation, properly setting up the project, and deployment are facilitated. If the client wanted to deploy or scale the project, it would be easier for them to do so (Sahani et al., 2020).

Compared to React, the main alternative, Angular provides a more structured approach to how your application should be designed. For example, Angular allows dependency injection while React does not, as components in the latter have their own global state. React relies heavily on external libraries in order for its components to be easily tested and decoupled (Pandit, 2021). Also, Angular is written in Typescript, while React in Javascript. Since variables in Javascript are dynamically typed, they can have any type of value assigned to them at runtime, regardless of their preset type. In contrast, Typescript is a statically typed language that provides static type functionality. Explicit type annotations, static type checking, and enhanced tool support, all of which are possible with Typescript, can aid in the early detection of mistakes and increase the maintainability of code. Finally, one disadvantage might be the steep learning curve of Angular compared to React or Vue.js, but our team will put in the necessary effort to overcome it.

Another key decision our team had to make is which map provider should we choose. The main options were Google Maps, Mapbox and OpenStreetMap. Again, we assessed each one of them, and in the end, we decided to use Google Maps. To begin with, it provided the most up-to-date mapping data. This includes features, such as address validation, address resolver and geocoding services.

Google offers in-depth documentation for its Maps API, which would facilitate the maps integration. Last but not least, Google Maps has the most familiar interface compared to the other options. Users are already acquainted with the way this map looks and would be more natural for them to use it.

The last fundamental decision we had to make is what is the best way to visualise points of interest on a map and express the intensity of a specific segment. All this should be achieved without making the map too overwhelming for the user. Also, this structure improves spatial analysis, as hexagonal grids tend to have less distortion in area, shape, and distance compared to other grid types. In addition, all adjacent hexes are at an equal distance from each other. This property ensures that interaction between neighbouring cells is consistent and eliminates the need for diagonal movements or distances, simplifying calculations (Mahdavi-Amiri et al., 2014). We researched this problem and found similar projects and looked into how they solved this problem. This is how we decided to use the hexagonal layer on top of the map. The next question was which framework to use so that we can build this hexagonal structure. Here our choices were more limited to several open-source javascript libraries, such as S2 geometry and Uber's h3 geo library. The latter turned out to be the best option as it provided complete documentation, the best performance, in terms of computing POI's (point of interest) intensity within hexagons and calculating area. For additional minor functionalities such as weather API and chart visualisation, we have decided to use respectively OpenWeather and Chart.js. The first one provides free and reliable API for the weather and the second one is the most famous Javascript library that facilitates chart visualisation.

We rely on well-known technologies which have proven their potency in time. In the course of the project, we will remain open to any suggestions from the university staff and client as to how we might improve the application.

6.3 Software Architecture

The design choices in developing an Angular application with distinct components for the homepage, map visualization, search bar, infotainment panel, and others reflected a pragmatic approach. By organizing the application into modular components, we followed Angular's best practices, which promoted maintainability and scalability.

The choice of a dedicated homepage component acted as the entry point for the application. This was a core component that contained all other components and managed them based on the user's interaction with the application. This was a middleware[2] component that facilitated communication between various components in order to make changes on the map. By isolating the homepage as a separate component, we allowed it to scale independently as the application grows (Garg, 2019).

For the map visualization component, this choice indicated a need to present geospatial data or location-based services. By separating it into its own component, we ensured that the complexities of dealing with maps and geospatial data were isolated. This makes it easier to integrate specialized libraries and optimize performance, which is often a challenge with map-heavy applications.

The search bar component likely serves as a critical utility across various parts of the application. By designing it as a distinct component, we can ensure that it is reusable, consistent, and maintainable. This design choice also allows for the encapsulation of searching logic, making it easier to upgrade or swap out underlying search functionality without impacting the rest of the application. Similarly, the filter component is responsible for enhancing the user experience by allowing users to tailor the content displayed according to their preferences. In an application with a vast dataset, such as points of interest, filtering is a valuable feature that can be further expanded with additional functionalities.

An infotainment panel component suggested a focus on user engagement. Separating this functionality into its own component was aimed at creating a dynamic, informative area of the application. This component had other components integrated into it, so it could easily change its content based on the users' needs. Due to this architectural decision, it contained details about points of interest, users, hexagons, and other elements of interest, such as geographic regions. This design also allowed for the integration of additional media in future developments.

Another vital part of our application was the Point of Interest (POI) service. It served as a connection between the backend and the front-end components, ensuring data flow. As an intermediary, it was responsible for fetching the necessary POI data, such as locations, descriptions, and any associated

[2]Software that enables one or more kinds of communication or connectivity between applications or application components

metadata, from the backend. Once the data is retrieved, the service processed and structures it as needed. It then disseminated this information to the relevant components, such as the map visualization for plotting locations or the infotainment panel for displaying details. By utilizing Angular's services, the POI service could efficiently manage data retrieval and communication in an asynchronous manner. This ensured that the user interface remained responsive and up-to-date, enhancing the overall user experience (Gautam, 2022). Through this service, the application could centralize the handling of POI data, making it easier to implement caching, error handling, and other optimizations.

7 Development Methodology

This chapter focuses on the development methodology we chose and used throughout the project. We recognized the importance of a structured approach that allows for flexibility and adaptability while delivering value to our client. By adopting an agile methodology, we aimed to focus on collaboration, efficiency, and responsiveness throughout the project life cycle. In this chapter, we discuss our project management methodology in section 7.1, guidelines for development in section 7.2, software testing in section 7.3 and communication strategies within our team and the company we are working with in section 7.4.

7.1 Project Management Methodology

Our team decided to implement the Agile methodology in developing our application for several reasons. To begin with, it provided us with the flexibility we needed to deal with uncertainties in software development. With Agile, we could respond to changes, even late in the development cycle, ensuring that we were not compromising the code quality. This methodology allowed us to break down our project into manageable units, known as sprints, which facilitated a structured development approach. This distribution not only made the process more manageable but also enabled us to prioritize tasks efficiently and deliver functional features in each iteration (Molina Ríos and Pedreira-Souto, 2019).

Moreover, the regular feedback and active involvement of the stakeholders, an integral aspect of Agile, helped us match our development process with the expectations of our client. Through frequent meetings with our TA and client, we could continually assess our progress and make necessary adjustments promptly. Agile also allowed for a collaborative environment within our team. Regular communication led to mutual understanding and easier resolution of conflicts.

During the app development process, we utilized GitLab as our version control system to streamline the workflow. GitLab, being a powerful collaboration tool, allowed the team to work on different features simultaneously through branching. Team members could commit code independently without affecting the main codebase. Merge requests facilitated the review of code changes, ensuring that only high-quality code was merged into the main branch. We utilized Gitlab's issues to divide the tasks between team members and assign labels and weights, so that we could monitor the significance and workload required to finish each task. We utilized labels to indicate the specific project components that an issue will target, as well as its rank in the MoSCoW hierarchical order. An example of issue division with labelling and assigning weights can be seen in Figure 3.

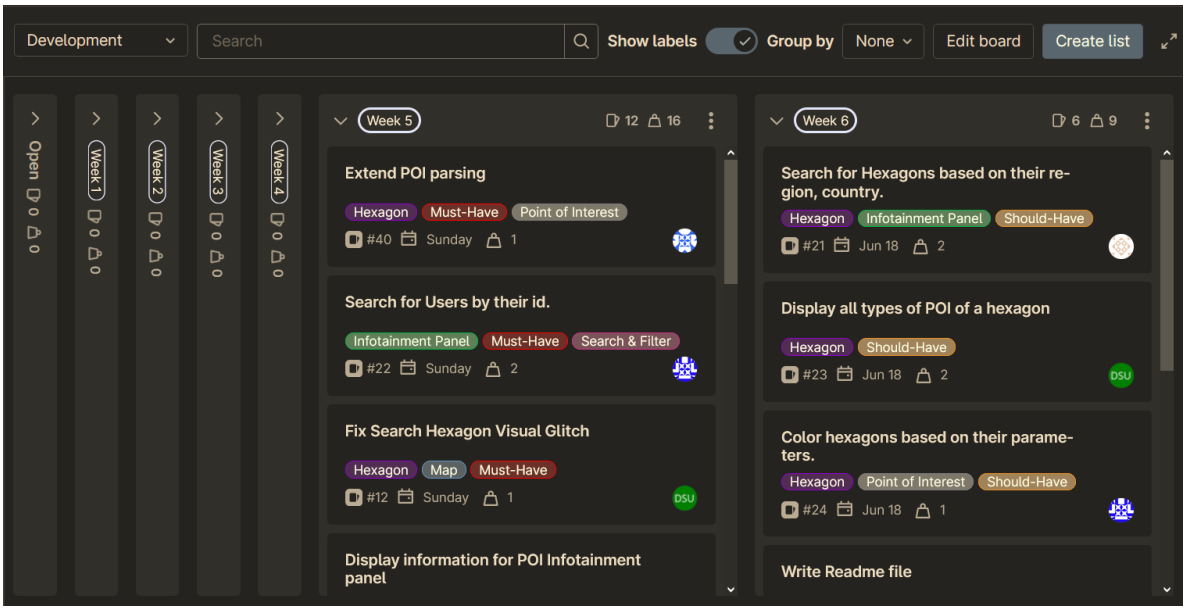


Figure 3: Issue board with all our current tasks, each with their corresponding assignee.

7.2 Guidelines for Development

Our development process placed a strong emphasis on our testing policy, which played a pivotal role in ensuring the quality and reliability of our application. We adopted a rigorous approach to testing, primarily focusing on unit tests as a means of breaking down our application into its smallest testable parts. By isolating and thoroughly testing each component, we were able to identify and address potential issues at the earliest stages of development. This not only ensured the integrity of our individual components but also laid a solid foundation for the entire application.

In addition to unit tests, we also implemented component tests, which provided a broader perspective by validating the interactions between different components. Through component tests, we ensured that each section of our application worked effectively in isolation and exhibited the desired behavior when interacting with its own view (Hamilton, 2023)..

Our team dedicated significant effort to writing comprehensive unit tests, aiming to achieve a minimum coverage of 70% for lines, branches, and functions before the project deadline, with a target of 80-90%. However, we faced challenges when it came to mocking Angular Google Maps, a crucial part of our project that required extensive research and consumed substantial time. Despite these obstacles, we recognized the importance of addressing this issue since a significant portion of our application relied on it.

To facilitate our testing process, we utilized Jasmine and Karma, which proved to be effective testing tools and enabled us to set up a robust testing pipeline. Our pipeline consisted of three stages. The first stage, the Build stage, verified if the code compiled successfully, allowing it to be merged into the codebase. The second stage focused on testing, ensuring that all tests passed before approving the code. The final stage of our pipeline was the checkstyle stage, where we enforced regulations and coding standards, such as adhering to naming conventions and eliminating unused imports.

Overall, our testing policy and pipeline played a critical role in maintaining the quality and reliability of our project, allowing us to deliver a well-tested and dependable application.

7.3 Application Testing

Our development process placed a strong emphasis on our testing policy, which played a pivotal role in ensuring the quality and reliability of our application. We adopted a rigorous approach to testing, primarily focusing on unit tests as a means of breaking down our application into its smallest testable parts. By isolating and thoroughly testing each component, we were able to identify and address potential issues at the earliest stages of development. This not only ensured the integrity of our individual components but also laid a solid foundation for the entire application.

In addition to unit tests, we also implemented component tests, which provided a broader perspective by validating the interactions between different components. Through component tests, we ensured that each section of our application worked effectively in isolation and exhibited the desired behavior when interacting with its own view (Hamilton, 2023).

Our team dedicated significant effort to writing comprehensive unit tests, aiming to achieve a minimum coverage of 70% for lines, branches, and functions before the project deadline, with a target of 80-90%. However, we faced challenges when it came to mocking Google Maps, a crucial part of our project that required extensive research and consumed substantial time. Despite these obstacles, we recognized the importance of addressing this issue since a significant portion of our application relied on it.

To facilitate our testing process, we utilized Jasmine and Karma, which proved to be effective testing tools and enabled us to set up a robust testing pipeline. Our pipeline consisted of three stages. The first stage, the Build stage, verified if the code compiled successfully, allowing it to be merged into the codebase. The second stage focused on testing, ensuring that all tests passed before approving the code. The final stage of our pipeline was the checkstyle stage, where we enforced regulations and coding standards, such as adhering to naming conventions and eliminating unused imports.

Overall, our testing policy and pipeline played a critical role in maintaining the quality and reliability of our project, allowing us to deliver a well-tested and dependable application.

7.4 Communication with Company and Team

When it comes to communication we decided on several platforms. Together with the company we have chosen MS Teams as the main platform for communication with them. For collaborative work on assignments and documents we used Google Docs to work on them simultaneously. For coding and sharing code, we used GitLab so that we can review each other's code and keep the code quality high. Among ourselves, we communicated through Discord and Messenger which we used for sharing relevant documents, scheduling meetings, etc.

To assure our communication went smoothly, we held meetings each week, in which everyone shared what they had worked on, and discussed our progress and potential issues we had encountered. We worked closely together in an office each week and tried to collaborate as much as possible. Our team believed that collaboration of this kind would be highly beneficial for the final product.

8 Conclusion

This report gave a detailed overview of the development process for a web-based tool called Data Explorer Map, which is designed to visualize real-time traffic, road conditions, and hazards. Following the process described in this report, we ensured that Data Explorer Map effectively provides users with detailed information about various road factors affecting travel and transportation and meeting the requirements and expectations of our client, Digital Lights.

The development process started with a research phase. The idea and main function of Data Explorer Map was conceived through thoroughly analysing the problem that was posed to our team, namely the lack of easily accessible and accurate data about road hazards, road conditions, infrastructure, and surrounding environment. This was followed by further research of stakeholders, of whom most important is our client, use cases in the field of travel, transportation and traffic management and already existing technologies, whose features could be incorporated in the application such as Google Maps, Hivemapper and Helium Explorer.

The next stages in the development included conducting feasibility study, risk analysis, requirement elicitation and implementation approach. We acknowledged the project as complex, knowledge-intensive and potentially privacy-sensitive. The approach we used to obtain functional and non-functional requirements was through meetings with the client, research, and brainstorming, and then employing the MoSCoW prioritisation technique to rank them based on importance, the most essential requirements being displaying a map with a hexagon layer and allowing search for a specific hex or area for which various data is displayed. Having defined the requirements, we proceeded to discuss our project approach and implementation decisions, made after thorough research and evaluations of various options, more specifically our choice of incorporating Angular, Google Maps, and hexagonal layer build with Uber's h3 geo library. Finally, the report went over the development methodology and the way our workflow was structured based on the agile methodology, which we chose to adopt.

In conclusion, we have successfully completed the development process and delivered the web-tool as per the client's request. The resulting product is a fully functional front-end map application with a unique hexagonal layer display. It enables users to view and interact with data collected by moving vehicles on the road. The application effectively addresses the initial problem and meets the requirements of our client.

Based on the insights and outcome of the project, our team can make the following recommendations that could ensure good scalability, data protection and efficient workflow. Firstly, we recommend choosing a modular and flexible architecture which can make integration of features and functionalities much easier, making the application much more scalable. For this purpose, we recommend using the Angular framework, since it relies on component architecture, which encourages reusability and modularity. Secondly, since this kind of application is rather privacy-sensitive due to the collection of a lot of sensitive data, we recommend incorporating robust security measures and privacy protections that should be prioritised when the application reaches a stage of deployment for the general public in the future. Finally, we recommend implementing a process of continuous improvement to ensure the application remains up-to-date, easy to use and aligns with the evolving needs of the client or user.

References

Bucea-Manea-Tonis, R. (2016). Angular js – the newest technology in creating web applications. *Annals of Spiru Haret University Economic Series*, *16*, 103. https://doi.org/10.26458/1638

de Souza, A. M., Brennand, C. A., Yokoyama, R. S., Donato, E. A., Madeira, E. R., & Villas, L. A. (2017). Traffic management systems: A classification, review, challenges, and future perspectives. *International Journal of Distributed Sensor Networks*, *13*(4), 1550147716683612. https://doi.org/10.1177/1550147716683612

Garg, B. (2019). *Angular Architecture Overview*. https://medium.com/@bhavikagarg8/angular-architecture-overview-1e7cc7483a0

Gautam, D. (2022). *Exploring Angular Services*. https://www.dotnettricks.com/learn/angular/services-in-angular-example

Hamilton, T. (2023). *What Is Component Testing? Techniques, Example Test Cases*. https://www.guru99.com/component-testing.html

Johnson, D. (2020). *How to Check the Traffic Around You on Google Maps in 2 Ways, so That You Know Which Routes to Avoid*. https://www.businessinsider.com/guides/tech/how-to-check-traffic-on-google-maps

Mahdavi-Amiri, A., Harrison, E., & Samavati, F. (2014). Hexagonal connectivity maps for digital earth. *International Journal of Digital Earth*, *8*(9), 750–769. https://doi.org/10.1080/17538947.2014.927597

Mehta, H., Kanani, P., & Lande, P. (2019). Google maps. *International Journal of Computer Applications*, *178*(8), 41–46. https://doi.org/10.5120/ijca2019918791

Molina Ríos, J., & Pedreira-Souto, N. (2019). Approach of agile methodologies in the development of web-based software. *Information*, *10*, 314. https://doi.org/10.3390/info10100314

Pandit, N. (2021). *What and Why React.js*. https://www.c-sharpcorner.com/article/what-and-why-reactjs/

Popa, B. (2022). *Google Maps and Waze Are Still the Most Popular Navigation Apps*. https://www.autoevolution.com/news/google-maps-and-waze-are-still-the-most-popular-navigation-apps-188811.html

Sahani, A., Singh, P., & Jeyamani, V. (2020). Web development using angular: A case study. *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, *1*(2), 1–7. https://doi.org/10.54060/JIEEE/001.02.005

Stojanovik, M. (2021). *Seven Ancient Roads That Connected the World*. https://www.odysseytraveller.com/articles/seven-ancient-roads-that-connected-the-world/

Wheatley, S., Maillart, T., & Sornette, D. (2016). The extreme risk of personal data breaches and the erosion of privacy. *The European Physical Journal B*, *89*(1). https://doi.org/10.1140/epjb/e2015-60754-4

Xing, Y., Huang, J., & Lai, Y. (2019). Research and analysis of the front-end frameworks and libraries in e-business development. *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, 68–72. https://doi.org/10.1145/3313991.3314021

Appendix 1: Requirements

Functional Requirements:

Must Have:

- Map
  - System must display a map. ✓
  - Users must be able to move around in the map. ✓
  - Users must be able to zoom in and out on the map. ✓
  - Hexagons on resolution 8 must be displayed on the map. ✓
  - All Point of Interest received as mock data must be parsed to hexagon resolution 8. ✓
  - Hexagons must be displayed on the map only if a Point of Interest has been reported inside the hexagon's boundaries. ✓
  - Multiple hexagons must be displayed for a Point of Interests that is smaller than the hexagon resolution 8. ✓
  - A single hexagon must be displayed for a Point of Interest that is bigger or equal to the hexagon resolution 8. ✓
- Structures
  - Hexagons must have an ID. ✓
  - Hexagons must have a list of Point of Interests that are inside the hexagon. ✓
  - Points of interest must have an ID. ✓
  - Point of interest must have the ID of the user that has reported it. ✓
  - A Single Point of Interest can you be reported by one user. ✓
  - Point of Interest must have a type. ✓
  - Point of Interest must have a date of creation. ✓
  - Point of Interest must have a date of expiration. ✓
  - Point of Interest must have a status that indicates if the Point of Interest is still active. ✓
  - Point of Interest must have the ID of the hexagon to which it belongs to. ✓
  - Point of Interest types must be: Potholes, Fog, Aqua-planning, Icy roads, Traffic Jams, Road Emergencies, Road Conditions, Police, Cameras and Radars, Incidents. ✓
- Filter
  - System must have a button that opens and closes a window for the filter. ✓
  - The filter window must contain a toggle switch for each type of Point of Interest. ✓
  - The filter window must contain a button for switching on all toggle switches. ✓
  - The filter window must contain a button for switching off all toggle switches. ✓
  - System must show only hexagons containing the Point of Interest Types, that are selected in the filter window. ✓
- Search
  - System must have a search field. ✓
  - Users must be able to search for hexagons by their ID. ✓
  - Searched Hexagon must be shown on the map. ✓
  - Searched Hexagon information must be shown on the infotainment panel. ✓
  - Map must move and zoom to the hexagon's location. ✓
  - System must indicate when a hexagon was not found by its ID. ✓

  - Users must be able to search for Points of Interest by their ID. ✓
  - The hexagon for the searched Point of Interest must be displayed. ✓
  - Map must move and zoom to the hexagon's location. ✓
  - Searched Point of Interest information must be shown on the infotainment panel. ✓
  - System must indicate when a point of interest was not found by its ID. ✓
  - Users must be able to search for a user by their ID. ✓
  - Searched user information must be shown on the infotainment panel. ✓
  - All Hexagons with the searched user must be displayed on the map. ✓
  - Map must move and zoom so that all found hexagons are visible. ✓
  - System must indicate when a user was not found by its ID. ✓

- Infotainment Panel
  - System must display an Infotainment panel. ✓
  - Users must be able to hide the Infotainment panel with a button when it is open. ✓
  - Users must be able to open the Infotainment panel with a button when it is hidden. ✓
  - The Infotainment Panel must have a back button, which shows the previously searched item and its Infotainment Panel. ✓
  - If there is no previous search item, the system must display an error. ✓
  - The Infotainment Panel must have a forward button, which shows the next searched item and its Infotainment Panel. ✓
  - If there is no next search item, the system must display an error. ✓

- Infotainment Panel for selected Hexagon
  - Clicking on a hexagon must display the Infotainment Panel for that Hexagon. ✓
  - Searching for a hexagon must display the Infotainment Panel for that Hexagon. ✓
  - Clicking on the hexagon ID in other infotainment panels must display the Infotainment Panel for that Hexagon. ✓
  - Infotainment panel must display the ID for the selected hexagon. ✓
  - Infotainment panel must display the name of the countries inside the hexagon. ✓
  - Infotainment panel must display the area in square kilometers for the selected hexagon. ✓
  - Infotainment panel must display a list of Point of Interests for the selected hexagon, containing all the Point of Interest that have been reported inside that hexagon. ✓
  - The list of Point of Interests must contain the ID of Point of Interest, Type of Point of Interest and ID of the user that has reported it for each Point of Interest in the list. ✓
  - When the user ID in the list is selected, it must search that user and display all hexagons of the user and open an Infotainment panel for the user. ✓
  - When the Point of Interest ID in the list is selected, it must search for the Point of Interest and display its hexagon and open an Infotainment panel for the Point of Interest. ✓
  - Infotainment panel must display the weather forecast of the selected hexagon. ✓
  - Weather forecast must contain the minimum and max temperature, description of the weather, wind speed, amount of rain the how the temperature feels for the selected hexagon. ✓
  - Infotainment panel must display an activity pie chart for a period. ✓
  - Users must be able to specify the period of an activity chart (week, month, year). ✓
  - The pie chart must consists only of the Point of Interests that have been reported in the last week, month or year, depending on what has been selected by the user. ✓

- Infotainment Panel for selected Point of Interest
  - Searching for a Point of Interest must display the Infotainment Panel for that Point of Interest. ✓
  - Clicking on the Point of Interest ID in other infotainment panels must display the Infotainment Panel for that Point of Interest. ✓
  - Infotainment panel must display the ID for the Point of Interest. ✓
  - Infotainment panel must display the type for the Point of Interest. ✓
  - Infotainment panel must display the date of creation for the point of Interest. ✓
  - Infotainment panel must display the expiration date for the point of Interest. ✓
  - Infotainment panel must display the status for the point of Interest. ✓
  - Infotainment panel must display the hexagon ID in resolution 8 to which the Point of Interest is parsed. ✓
  - When the hexagon ID is selected, it must search that hexagon and display it and open it's Infotainment panel. ✓
  - Infotainment panel must display the ID of the user that has reported the Point of Interest. ✓
  - When the user ID is selected, it must search that user and display all hexagons of the user and open an Infotainment panel for the user. ✓
- Infotainment Panel for selected User
  - Infotainment panel must display the ID of the user. ✓
  - Infotainment panel must display a list with all the points of interest recorded by the user. ✓
  - The list must contain the ID of reported Point of Interest and it's type. ✓
  - When the user ID in the list is selected, it must search for the user and display all the hexagons and open an Infotainment panel for the Point of Interest. ✓

Should Have:

- Search by Region and Region Infotainment Panel
  - Users should be able to search based on region. ✓
  - Region should be a Continent, Country, City, Town, Road, District or any other geographical region. ✓
  - All hexagons inside the searched region should be displayed on the map. ✓
  - Map should move and zoom so that all the found hexagons are visible. ✓
  - An infotainment panel for the searched region should be visible. ✓
  - The infotainment panel for the search region should contain the name of the region. ✓
  - The infotainment panel for the search region should contain a list with all the IDs of the hexagons inside the region. ✓
  - Clicking on the ID of a hexagon should open the infotainment panel for that hexagon. ✓
  - Clicking on the ID of a hexagon should visualize the hexagon. ✓
  - Clicking on the ID of a hex should move and zoom the map to the hexagon location. ✓
- Hexagons should be colored differently, depending on the number of Point of Interests reported inside them. ✓
- Hovering over a hexagon should display a pop-up with the all the type of Point of Interest inside the hexagon. ✓

- System should search for the corresponding item, without the users requiring to the specify the type of the item they are searching for. ✓

Could Have:

- Users could search for hexagons based on geographical coordinates (Latitude, Longitude).
- System could have auto-complete for the search field and show suggestions for what to search.
- System could have a button that switches between dark and light themes.
- System could show a heat map instead of hexagons, when zooming out too much on the map.

Won't Have:

- The system won't show the children hexagons of a hexagon in the Infotainment panel in General Info.
- Users won't be able to create new types of points of interests and report them.
- Users won't be able to set the time of expiration for a point of interest.
- The system won't show personal and sensitive information for a user in the infotainment panel for the user.

Non-Functional Requirements:

- System uses Google Maps as a map provider.
- System uses the H3 Geo library.
- System is implemented with the Angular framework.
- System is written in TypeScript, HTML, CSS.
- System uses OpenWeatherMap for the weather forecast.
- System uses ChartJs for the chart generation.

Appendix 2: Work Division

- Ivan Virovski

  Implementation - Documentation, Testing, Map component and visualization, Hexagon visualization, Pipeline, Homepage Component, Infotainment panel structure, Hexagon Infotainment panel business logic, Back and Forward buttons UI and logic, search by region, Region infotainment panel

  Report - Requirements chapter, Table of Contents, Appendix 1 - MoSCoW, Page numbers, Introduction

- Vladimir Pavlov

  Implementation - Poi Service, Poi parsing, User Infotainment panel component, Hover functionality over hexagon, Testing + Documentation, User Interface improvements, Add mock users to mock data, Hexagon infotainment panel pie chart logic

  Report - References list, Project Approach chapter, Personal Data from Values chapter, Development Methodology Rework, Title page

- Dafni Pandeva

  Implementation - Documentation, Testing, Top Bar UI, POI infotainment panel component, Infotainment panel UI, Clicking on hexagon logic, Hexagon infotainment panel business logic, Color hexagons based on parameters

  Report - Summary, Conclusion, Analysing Problem Domain of Data Explorer Map chapter

- Dimitar Uzunov

  Implementation - Documentation, Testing, Search by hexagon ID, Search by user ID, Search by POI ID, Pipeline, Search logic without switch button, UI small fixes

  Report - Environment, Health and Safety from Values chapter, Development Methodology chapter, Weekly agendas

- Atanas Kichukov

  Implementation - Documentation, Testing, Pipeline most work, Filter business logic and UI, Improvements to hexagon parsing and visualization

  Report - Feasibility Study & Risk Analysis chapter, Preface