
4th Year Applied Project and Minor Dissertation Data Visualisation

Grace Keane

Jina Kim

Shirin Nagle

B.Sc.(Hons) in Software Development

MAY 10, 2021

Final Year Project

Advised by: Dr John French

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)



Contents

1	Introduction	6
1.1	Context	6
1.2	Why Visualisation matters	7
1.3	Project Objectives	9
1.4	Metrics for Success and Failure	9
1.4.1	Applied Project Dissertation	10
1.4.2	Applied Project	10
1.5	Dissertation Summary	11
1.5.1	Methodology	11
1.5.2	Technology Review	11
1.5.3	System Design	11
1.5.4	System Evaluation	11
1.5.5	Conclusion	11
2	Methodology	13
2.1	Project Management and Research	13
2.1.1	Brainstorming and Initial Supervisor Meeting	13
2.2	Methodology Consideration	14
2.2.1	Determining Development Methodology	15
2.3	Waterfall	15
2.4	Agile	16
2.5	Comparison between Agile and Waterfall	17
2.5.1	Agile Methodology	18
2.5.2	GitHub Tasks	19
2.5.3	Scrum	20
3	Technology Review	22
3.1	Version Control	23
3.1.1	Git	23
3.1.2	GitHub	24
3.1.3	Flask	25

3.1.4	React	26
3.1.5	MongoDB Atlas	27
3.2	Languages Used	29
3.2.1	JavaScript	29
3.2.2	JSX	29
3.2.3	Python	30
3.3	Development Environment	30
3.3.1	Visual Studio Code	30
3.4	FireBase	31
3.4.1	User Authentication	31
3.4.2	Realtime database	32
3.5	Cloud Firestore	34
3.6	D3	35
4	System Design	37
4.1	Architecture Overview	37
4.2	Live API Calls	38
4.3	Three tier architecture	39
4.3.1	Data Tier	39
4.4	Logic Tier - Back end	39
4.5	Application Tier - Front end	40
4.6	Heroku - Deployment	41
5	System Evaluation	42
5.1	Chapter Overview	42
5.2	Project Objectives	42
5.3	Testing	42
5.3.1	Unit Testing	42
5.3.2	System Testing	46
5.4	Limitations and areas of Improvement in this project	47
5.4.1	Limitations	47
5.4.2	Improvements	47
5.4.3	System Evaluation Conclusion	47
6	Conclusion	48
6.0.1	Context Objectives	48
6.0.2	Methodology Objectives	48
6.0.3	System Evaluation Findings	49
6.0.4	Opportunities for future Investigation	49
6.0.5	Group Reflections	50

<i>CONTENTS</i>	4
-----------------	---

A Appendix	52
A.1 GitHub Link	52
A.2 Heroku	52
A.3 Screencast Link	52

About this project

Abstract Data visualisation offers powerful and flexible ways of presenting information. It allows ways of bringing together different streams of information and synthesising them into a coherent and meaningful form. Data visualisation has become popular, in part due to the experience of the global Covid-19 pandemic. Information is delivered in easy to understand visual representations. One of the challenges of the pandemic has been finding ways to present information in a public health context and modelling and forecasting. As this is an emerging area, the team decided to investigate different methods of data visualisation, see which performed the best for different scenarios and to develop a data visualisation application. We developed an online application, which utilises the almost real time availability of Covid-19 data and presents this data in clear graphical form. The main focus of the application is Covid-19, The application includes examples of earlier epidemics, SARs, MERs and historic epidemics of smallpox. These are provided for some historical context. Front end development used React, the back end development used Flask to write our own API calls. The data tier used Firebase and MongoDB Atlas to store and retrieve static data. Relating to Covid-19: The web application features maps selectable by country, graphs, and charts. Some of the types of data featured are total worldwide cases, total recovered cases, total deaths, and Irish vaccination progress. Relating to historical diseases some of the data featured are case numbers and mortality numbers.

Authors This project was developed as part of a fifteen credit module by Grace Keane, Jina Kim and Shirin Nagle, fourth year students of Galway Mayo Institute of Technology.

Acknowledgements The authors would like to acknowledge the help and advice provided by the project supervisor Dr. John French.

Chapter 1

Introduction

This chapter will outline the context of the project, why the subject matter was chosen and its relevance. We will discuss the project objectives and metrics for success and failure. We will summarise the project by giving a brief introduction into each chapter heading of the dissertation.

1.1 Context

Why choose data visualisation as a subject for our Applied Project and Minor Dissertation? Towards the end of our third year in college, Ireland was shut down and effectively put into a lockdown. All colleges, schools, and areas where large amounts of people gather were closed or off limits. The reason for the lockdown was the outbreak of a disease which became known as Covid-19, a global pandemic. The first pandemic that the majority of the world's population will have ever experienced. The last global pandemic was the Spanish flu, which lasted from approximately 1918 to 1920, this pandemic occurred in three waves, though not simultaneously around the globe.[1]

The fact that our on-campus learning was curtailed due to the Covid-19 outbreak has had an impact on our learning. Being software students, it was fascinating to see how much real time information was available coming from all directions. Sites like John Hopkins Dashboard tracked Covid-19's progress round the world on a daily basis. Starting off with what looked like small numbers at first, the amount of cases quickly exploded in countries like Italy and Iran, an example of exponential disease growth. If measures we not taken to halt the spread of the virus the rest of the world would eventually experience exponential growth of cases.

1.2 Why Visualisation matters

There are two useful functions for data visualisation.

- a) The dissemination of public health information, with purpose of giving the clearest information possible to the general public.
- b) The effective modelling, prediction and forecasting of disease trends for scientists and policy makers.

Exponential growth is a mathematical concept that is not properly understood by the majority of people. "People mistakenly perceive the coronavirus to grow in a linear manner, underestimating its actual potential for exponential growth." [2]

"The complex data are extremely challenging for our cognitive abilities. Visual Analytics solutions may support knowledge discovery and problem solving if the needs of the different stakeholders, such as epidemiologists and environmental health specialists, are adequately addressed." [3]

It struck the team that the visual representation of data is a good explanatory tool to show how fast exponential growth can happen. As well as disseminating public health information to the general public and allows experts to perform predictions and modelling based on current trends. This information can help policy makers make the best decisions possible given current conditions. "It can help researchers and policy makers to identify trends that could be overlooked if the data were reviewed in tabular form". [4]

With this background, the project seemed timely, relevant and interesting. It offered the potential to be data rich and ideally suited to a team project. The team felt it was important that the project encouraged the use of existing skills as well as allow for the natural development of new and relevant techniques and processes while also being worthy of the scope of the project.

Data visualisation and analysis techniques have been central in the efforts to communicate the statistics as well as the science around the COVID-19 virus. "Data visualization discussions can surface key issues, jump start conversations, and set the stage for further inquiry" [5]. Many websites have Incorporated interactive dashboards as well as charts and graphs to to visualize complex and overwhelming pandemic data sets. "Visualization

plays an important role in epidemic time series analysis and forecasting.”[6]

The team felt it would be beneficial as well as informative to create a web application that would take large COVID-19 data sets as well as other disease data and create a series of data visualisations from it. To compare past and present viruses and distinguish similarities along with differences. This would provide a clear view of COVID-19 and how similar viruses have spread.

Once the project area had been decided on, each team member focused on a pandemic or epidemic, investigated where to source relevant information for the pandemic or epidemic, and investigated different technologies for displaying data in a visual manner.

Data Visualisation Research

- John Hopkins Dashboard

An interactive web-based dashboard to track COVID-19 in real time. This dashboard is the leading example of Covid-19 dashboard information. It had come to public awareness at the beginning of the pandemic and its data visualisations shaped the way people began to think about the pandemic.

- Gap Minder

A company founded in Sweden in 2005, this company developed several innovative data visualizations, including Trendalyzer a bubble chart software, which was acquired by Google.[7] We were interested in this company as a leading force in data visualisation technology. They have changed how statistics are presented.

- EpiViewer

This is an example that is close to the subject matter and technological scope of our project which involved developing a web application too. It was a large collaborative research project involving nine researchers. They worked on a web application that provides a framework for exploring, comparing, and organizing temporal data sets. To date the website does not seem to be publicly available.

During the initial research stage, the team were aware of the Spanish Flu as being the most recent pandemic. We thought that it would be interesting to include data for the Spanish flu and other epidemics like smallpox, SARs and MERs. Covid-19 data is very easy to source and the main problem was narrowing down which sources to use. Conversely data for the other diseases

mentioned above was not so easy to source. The main reason for this is the age of the data, the Spanish Flu pandemic took place over one hundred years ago. Smallpox was finally eradicated in 1980. "Smallpox eradication was one of the greatest scientific and humanitarian achievements of the 20th century." [8] The inclusion of the older disease data provides an interesting counterpoint to the current data rich modelled pandemic.

Data visualisations can give a seemingly very precise picture but it is always reliant on the veracity of the data and may run the risk of all data being equal. "An exponential growth model is usually assumed to characterise the early phase of epidemics. But, this assumption can lead to failure to appropriately capture the profile of the epidemic growth, eventually giving rise to non realistic epidemic forecasts" [9].

1.3 Project Objectives

The objective of the project is to create a web application that would process different virus data and create various data visualisations to ensure a simple way for users to view large complex pandemic data. To ensure the objectives were reached, the team outlined the following:

- Investigate the field of interest and scope for the project. What is currently being done in the field, to understand some its challenges and potential development.
- Research and evaluate appropriate frameworks and tools available for development. Shortlist technologies which have the potential to be fit for purpose.
- Incorporate state of the art technologies, frameworks, databases and tools that will allow users to view, add, update, post and retrieve data. Evaluate which might be most suitable in practical terms for the scope of this project.
- The web application will, allow users to sign-up, login, view data visualizations as well as incorporate user interaction features, create unique user visuals and post user data to databases.

1.4 Metrics for Success and Failure

In order for the project aims to stay on track the team felt it was important to identify measures for success and failure. Each project outlined objectives

listed in *Section 1.3* will be reviewed at the end of the project and evaluated in the conclusion section. The initial success and failure objectives defined are outlined in *Section 1.4.1* and *Section 1.4.2*

1.4.1 Applied Project Dissertation

For this dissertation we are following the guidelines set out by the module, which are listed below:

- *The final dissertation should be easily understood, laid out in a structured and informative manner. Allowing readers without knowledge or familiarity of data visualisation to gain a general understanding of the concept and areas discussed.* We measured this metric by asking for input from family members. Their feedback was very useful as they are not software students and did not have as much exposure to data visualisation applications. This helped us identify areas for exploration that had not occurred to us before consulting our family members.
- *The dissertation should be in order of 10,000 - 15,000 words, excluding appendices as well as approximately 35 pages (excluding diagrams, abstract, TOCs, references, appendices etc.)*

1.4.2 Applied Project

A selection of metrics were outlined specifically for the development of the applied project, they were as follows.

- *The applied project must be easy to use, navigate and attempt to deal with a task of problem deemed to be of sufficient technical challenge and depth.* At frequent stages of development feedback from team members and Dr. John French was evaluated and taken on board relating to the application. Still relying on family members to check ease of use and navigation particularly in the early stages of development when the application was not deployed.
- *Collaboration between team members and supervisors should be consistent to ensure a smooth work flow throughout the project planning and development.)* In order for this metric to be adhered to, the team maintained continuous communication, keeping in touch using Microsoft Teams to manage meetings. Maintaining a tasks list in GitHub, daily communications through Whats App to discuss project ideas or issues.

The team had weekly meetings with Dr John French throughout the development and planning of the applied project.

1.5 Dissertation Summary

This section contains a brief overview of the dissertation structure and provides a short description of each chapter.

1.5.1 Methodology

This chapter will discuss the investigation process of prospective development methodologies. The decision making process and factors leading up to the decisions and design implementations will also be discussed. Comparing the software development methodologies Waterfall and Agile investigated during the research stage. The results of the investigation and how the chosen methodology shaped the development of the project.

1.5.2 Technology Review

The technological review will outline the necessity for version control. Discuss the technologies included in the project, why they were chosen, what role they played in the project. The benefits of the selected technologies will be evaluated and compared with similar available alternatives.

1.5.3 System Design

A detailed explanation of the overall system architecture will be provided. We created an illustration of how the system architecture works, to explain in a visual way of how each technology interacts in the the architecture.

1.5.4 System Evaluation

This chapter will cover an evaluation of the final applied project against the initial project objectives. The final result of the project will be evaluated including an analysis of areas of software quality, improvements or changes to the overall project.

1.5.5 Conclusion

To conclude, a summary of the context and objectives to remind the reader about the overall rationale and goals of the project. Key insights will be

identified and reflected on. A final analysis will describe the teams overall experience and what the team learned while working on a project similar to one encountered in the software industry.

Chapter 2

Methodology

2.1 Project Management and Research

This chapter will explore the pre-development process, the approach the team took with development and testing, how the team dealt with various problems faced, and the influence of regular supervisor meetings along with a conclusion based on what impact pre-development research had on the overall project direction.

The first team project meeting took place in the final week of September 2020. This meeting consisted of analysing the applied project requirements that had been defined and outlined. The team agreed to choose a topic that would be informative yet beneficial to the end user. It was concluded that the idea and architecture of the solution should be finalised as soon as possible to allow for necessary research and planned development.

2.1.1 Brainstorming and Initial Supervisor Meeting

Before development began and after the initial project topic was chosen each team member researched various technologies and concepts that could be potentially incorporated into the project. We had a few brainstorming meetings and talked through various options. After our brainstorming meetings we considered each technology researched and the team decided whether each would be a good fit.

The team met prior to the initial supervisor meeting to produce effective ideas and questions to be asked relating to the overall architecture of the solution and the ultimate goals and objectives, these included:

- **What technologies would be of best fit for the project?**
The team investigated many possible technologies for the project. Chapter three will fully outline our choices and reasoning for selecting these technologies.
- **What benefits would certain methodologies possess compared to others?**
The advantages and disadvantages of the methodologies, such as Agile and Waterfall, were considered by the team and are outlined in full in this chapter.

The first meeting with our supervisor Dr. John French took place in the first week of October. We outlined our basic project idea, potential research, development technologies, development methodology and collaboration approaches. Dr French advised the team to spend the next few weeks to investigate and research the overall picture of how the applied project will be designed and implemented.

After the supervisor meeting the team decided to take this advice on board. Each member focused on researching the What? Why? How? of the project technology, idea and the overall end user experience as well as the relevance of the project. Once research was carried out the team had a meeting to discuss and analyse the findings. Some questions asked were as follows:

- What development methodology and collaboration technique would be best to adopt?
- Would the chosen project idea be of relevance to the end user?
- Would the chosen technologies be of a professional standard?
- How would the chosen technologies fit together as a whole?
- Is the scope an appropriate size for a three person project?

These questions will be fully discussed further in the dissertation.

2.2 Methodology Consideration

There were numerous possible methodologies to consider. After discussions with team members, the team decided it would be worthwhile to analyse

and compare **Waterfall** and **Agile** to determine which methodology would be best suited to the applied project. These methodologies are amongst the most popular used by software development teams worldwide.

2.2.1 Determining Development Methodology

Following the decision of Agile and Waterfall as possible appropriate approaches to software development, the team began to research the advantages and disadvantages of both, along with deciding which methodology would be best to use when considering the scope, project goals and time frame. A critical analysis and overview of both methodologies were made and a conclusion based on the practical analysis of each in relation to the project was undertaken.

2.3 Waterfall

The Waterfall model is composed of a series of steps, illustrated in *Figure 2.1*. These steps are used to break down a project into linear sequential phases to ease the development process. [10]. An defining feature of the Waterfall methodology is that the full nature of the project is known in advance. This allows progress to be easily measure, this option can be preferred by developers depending on the type of project and software team. The Waterfall model tends to be among the less flexible approaches, the output of each stage completed is the input for the next and the overall progress follows a fixed downwards path like a waterfall. The waterfall model is still a widely

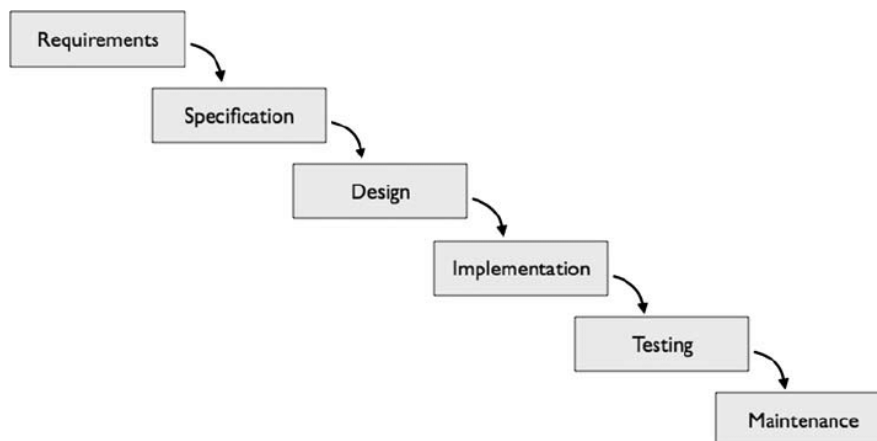


Figure 2.1: Waterfall Methodology

used method for working on software projects today. It is arguably one of the most well known development models. It is still used for very large projects that have a long development cycle. Particularly used by government agencies for the implementation of large software projects. Each phase is of a waterfall project is completed sequentially and testing is done after tasks are completed. [11]

2.4 Agile

Agile methodologies are a group of software development methods that are based on interactive and incremental development [12]. The term agile stands for 'moving quickly'. It is characterised by an approach that allows rapid and continuous delivery of small and useful software. It takes the view that production teams should start with simple and predictable approximations to the final requirement and then continue to increment the detail of these requirements throughout the life of the project. Agile methodology has an adaptive team which is able to respond to changing requirements as well as the ability to solve problems even late in the development cycle. [11].



Figure 2.2: Agile Methodology

The popularity of Agile development methodologies amongst the software industry has increased drastically, with almost 85.4% of international surveyed software developers using Agile methodologies in their work. Research has also shown that Agile projects are 28% more successful than tra-

ditional projects. The most important principle of the Agile methodology is customer satisfaction by giving rapid and continuous delivery of small and useful software.[13]

2.5 Comparison between Agile and Waterfall

During team meetings, research and analysis, both Agile and Waterfall were evaluated under the following headings:

- **Requirement delivery**
- **Flexibility**
- **Compatibility**
- **Methodology past experience**

By taking the above headings into an account comparisons were drawn for both methodologies.

Following various comparisons between Agile and Waterfall, it became clear to the team that an Agile approach would best suit the development of the outlined applied project. This approach would aid the development process in numerous ways, these include:

1. **Errors can be detected and solved quickly** - The team thought it would be beneficial to be able to detect and solve problems quickly, especially when the team did not have much experience with using GitHub collaboratively together as a team.

2. **The ability to develop small but functional software via releases** - Based on feedback from our supervisor the priorities of the project could be subject to change. Small incremental changes means there would be flexibility and versatility throughout the development stage.

3. **User Stories** A user story is a tool used in Agile software development to simply describe a software feature from an end user point of view. A user story describes the type of user, what they want and why they need this feature. It helps to create a simplified description of a requirement and focus the development team on the feature itself. We created user stories in the first few weeks of planning.

4. **Scrum Methodologies and GitHub project boards** - The idea of incremental releases in parallel with scrum sprints based on user stories

Agile	Waterfall
+ Ability to respond to the changing requirements of the project.	- Requirements are clear before development starts.
+ Errors can be detected and solved quickly. Which is a big advantage.	+ Each development phase is completed in a set time frame. Then it moves to another phase.
+ There is constant communication and inputs from the team and supervisor	+ It is a linear model so therefore easy to implement.
- Project can easily fall off track.	- Requirements, Specification and Design can take up a lot of time.
+ The end point of the project is not clearly defined. Requires continuous development and changes to design.	- Low flexibility meaning it may be difficult of even impossible to make major changes while in the implementation phase

Table 2.1: Agile V's Waterfall comparison.

using GitHub project boards to manage the workflow was something the team thought would be of major benefit to the overall project.

5. **Frequent supervisor and team meetings** - Information sharing in teams is an important aspect of successful software development. The team felt it would be crucial to hold regular team meetings to discuss what each member is working on as well as attend all weekly supervisor meetings. The team and supervisor meetings focused on what we worked on last week, what we will work on in the coming week and any issues encountered.

2.5.1 Agile Methodology

Deciding what agile methodologies was the next big step the team had to decide on. After research the team came to the conclusion that Scrum and GitHub project tasks would most beneficial for the team to use in the development of the project.

2.5.2 GitHub Tasks

Since the beginning of development for the applied project the team set up and adhered to GitHub tasks by assigning relevant work to do each week. GitHub tasks also known as GitHub Kanban is a project management tool designed for developers to coordinate, track and update their work in one place, so projects stay transparent and on schedule.

GitHub Tasks allows developers to "visualize all of your work and prioritize it right alongside your code with projects boards. See what tasks are planned or in-progress, either in a repository or across your organization". [14]

Using GitHub Tasks was easy to use, and allowed us to keep track of the project's progress. It suited the needs for the project, was very easy to see at a glance what was "to do", what was "in progress" and what was "done". It was a very useful tool which was too easy to access and did not involve another project management tool like Microsoft Project, which are often overly complicated for the task at hand.

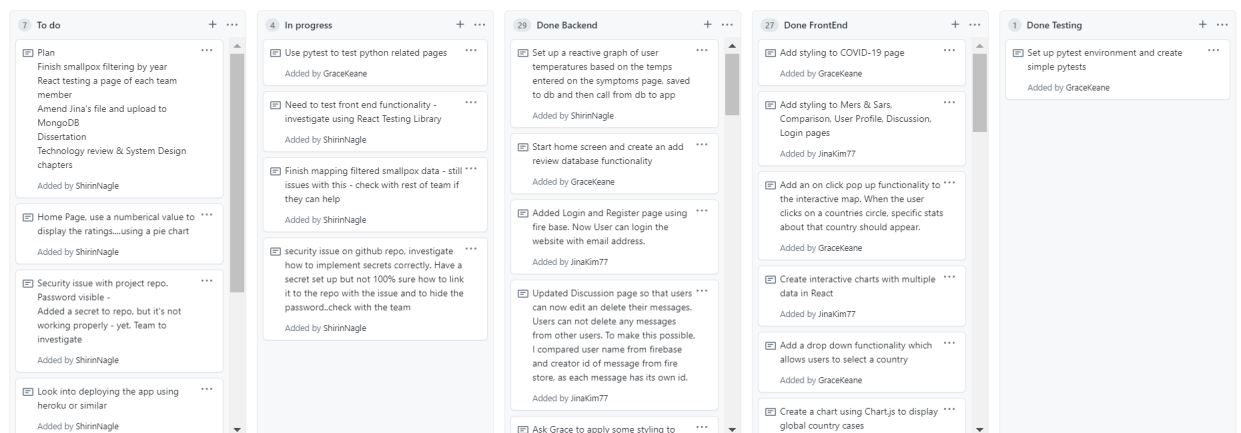


Figure 2.3: GitHub Tasks

Figure 2.3 shows the GitHub task sheet used during development of the applied project. It consists of five main columns:

- **To Do**

This section was for tasks that were identified but not actioned yet.

- **In Progress**

This section was for tasks that were in progress and not complete.

- **Done Backend**

The amount of completed tasks was quite large it was decided to split the completed tasks into front and backend tasks. This section displays all the completed backend tasks.

- **Done Frontend**

Completed tasks relating to the front end development were moved from in progress to here.

- **Done Testing**

Completed test tasks are moved here.

2.5.3 Scrum

As well as using GitHub Tasks to manage project management, the Scrum methodology was incorporated and employed. This development methodology assumes that the project or systems progress is an unpredictable process that can be described as an overall progression [15]. The Scrum methodology is designed for teams to break down work into goals that can be completed in a specific time frame called sprints. Short sprints are generally one to two weeks in length. We decided to make our sprints weekly as we were meeting our supervisor weekly. We met up as a team at least once a week but often more to discuss the project progress. At the end of each week, a supervisor meeting would be held where the team would discuss their progress with the supervisor.

During our weekly supervisory meetings, we discussed what work we had done, areas of work to be completed or started were identified and also advised the supervisor when there were particular development issues that were causing problems for the team. The goals for the next Scrum Sprint were outlined and discussed. New goals were also assigned and recorded in the

teams GitHub Tasks board. These tasks were undertaken in a high to low priority order.

When the team encountered problems, we tried where possible to resolve them together as a team. We discussed the problems encountered and each team member tried to help with finding a possible cause to the problem and trouble shooting where possible. We researched each other's problems on line and sent relevant research and information to each other.

Chapter 3

Technology Review

This chapter will outline the technology used and the reasoning for choosing a particular technology. Below is a list of technologies used, descriptions of the technologies at a conceptual level are included for each technology.

- Git
- GitHub
- React
- Flask
- MongoDB
- FireBase
- FireStore
- D3
- JavaScript
- Python
- Visual Studio Code
- Live API Calls

3.1 Version Control

Version control is an essential element of any collaborative undertaking. An organisation in GitHub was set up to manage the version control for the project. At the beginning of the project the team had limited experience with collaborative version control as the project progressed we became more experienced. As a result in the early stages the team were investigating different aspects of data visualisation, this resulted in smaller repositories to track our work. When this investigation concluded we created a new repository where all the future work of the project would be tracked.

3.1.1 Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is a free and open source software distributed under the terms of the GNU General Public License version 2. Git allows and encourages the user to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds.[16] Git is installed and maintained on the users local system and provides a record of ongoing programming versions. It allows developers to rollback to an older version of the code if required. There are other popular version control systems for example Apache SVN or CVS, the team chose Git as it was a technology that we were familiar with but did not have a lot of experience with. We felt that there would be many large learning curves ahead of us and if we could mitigate the size of the learning curve by choosing a familiar technology it would help us focus on other areas that required more learning. Having said that there was a significant learning curve using Git with a team. Git gave us experience of using branches, merging branches and manually resolving conflicts. See Figure 3.1 for an example of git on a local machine.

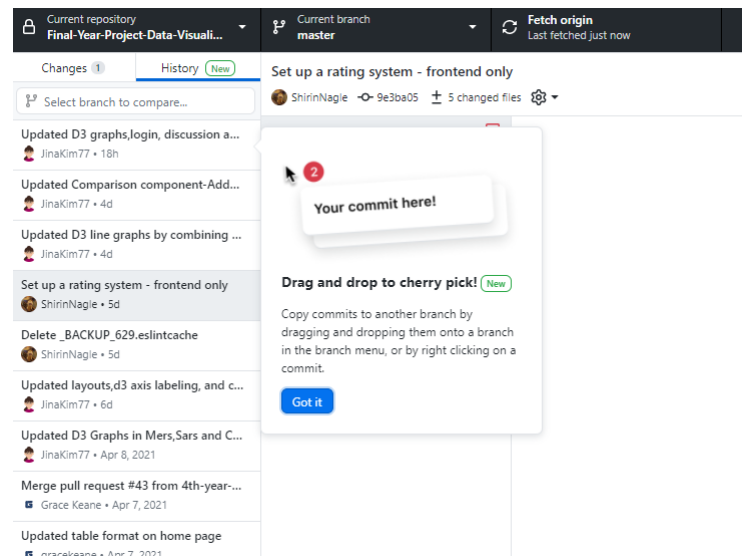


Figure 3.1: Git on local machine

3.1.2 GitHub

GitHub is a web-based code hosting platform for version control and collaboration. It lets users work together on projects from anywhere.[17] GitHub provides students of GMIT with free accounts, for this project a free account was sufficient as there is a facility to make the project public or private. Using GitHub allowed the team to work entirely remotely and resolve any issues with the project.

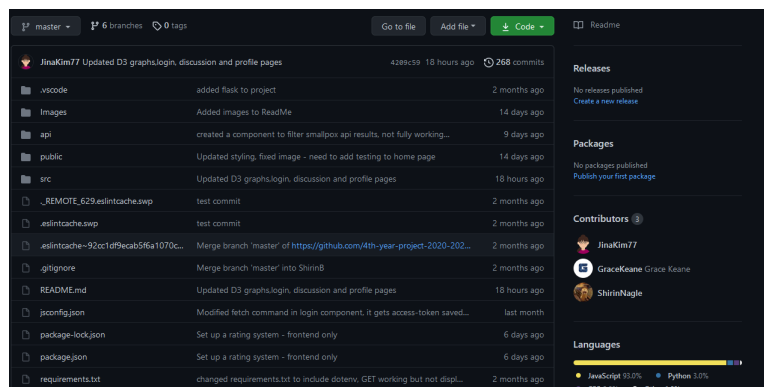


Figure 3.2: Github Organistaion

3.1.3 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. The “micro” in microframework means Flask aims to keep the core simple but extensible. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.[18] The architecture of our project required a back end, front end and a database. We had some experience with MEAN(MongoDB, Express, Angular, NodeJs) and MERN(DB, Express, React, NodeJs) stacks from previous college projects. More recently we had some exposure to Flask as a backend in Emerging Technology project. We had decided to use React for our frontend, which we had used for a previous MERN project. We had also used Angular previously for a MEAN project but decided to use React. There will be further discussion in the React section about why React was chosen. Part of the teams learning objectives was to use new technologies, to the team, and learn as much as possible about them. For this reason we chose Flask as our backend, as it offered flexible options re database and front end choices. From implementing Flask and reading documentation, the idea of virtual environments was introduced. Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating system’s packages.[19] Most importantly using Flask allowed us to write our own API calls to our MongoDB Atlas database.

```
from flask import Flask
from flask_cors import CORS
from routes import indexRoute, createRoute, itemRoute, getDescriptionRoute, smallpoxRoute
from rating import ratingRoute, indexRating
from age import ageRisk
import os
import flask

app = Flask(__name__)
CORS(app) # wrap app in CORS

# register the blueprints
app.register_blueprint(indexRoute)
app.register_blueprint(createRoute)
app.register_blueprint(itemRoute)
app.register_blueprint(smallpoxRoute)
app.register_blueprint(getDescriptionRoute)
app.register_blueprint(ratingRoute)
app.register_blueprint(indexRating)
app.register_blueprint(ageRisk)

if __name__ == "__main__":
    app.run(debug=True)
```

Figure 3.3: Flask

3.1.4 React

React is a JavaScript library, used for building reusable components. React allows users to design simple views for each state in their application, and React will efficiently update and render just the right components when the data changes. Declarative views make the code more predictable and easier to debug. Build encapsulated components that manage their own state, then compose them to make complex User Interfaces. React does not make assumptions about the users technology stack, allowing the development of new features in React without rewriting existing code.[20] React recommends using JSX, an syntax extension of Javascript. We have used JSX and JavaScript in our project. See example code from the project.

```
class MersAndSars extends React.Component{
  state={
    virus:"mers"
  }

  virusSelected=(virus)->{
    this.setState({
      //virus:virus
      virus
    })
  }

  render(){
    return <Wrapper className="App">
      <Navbar bg="light">
        <Navbar.Brand>MERS and SARS</Navbar.Brand>
      </Navbar>
      <Container>
        <Row>
          <Col xs={12}><VirusDropdown virusSelected={this.virusSelected} /></Col>
        </Row>
        <Row>
          <Col xs={12}><ChartWrapper virus={this.state.virus} /></Col>
        </Row>
      </Container>
    </Wrapper>
  }
}
export default MersAndSars;
```

Figure 3.4: Example of JSX

When researching what front end software to use we considered Angular and React. From our research we compared the two technologies. Included below is a table outlining the advantages and disadvantages of both.

Information about Job popularity taken from a google search on the 15th of April 2021 Information about professional popularity taken from Stack-Overflow survey(2020) of most loved Web framework.[21]

There is no definitive way to decide in advance, which technology would be the best to use, but comparing relevant aspects between technologies can help narrow the choices and help with making the final decision.

FrontEnd Technology Comparison Angular v's React			
Heading	React	Angular	Best Option
Structure	JS Library	Full framework	React
Application	Suits SPA	Suits large App	React
Updates	Often	Twice a year	React
Components and Size	Uses virtual DOM	Uses Real DOM	React
Industry Demand	897	366	React
Professional Popularity	68.9%	54%	React
Dubug	Easy	Not so Easy	React

Table 3.1: React V's Angular comparison.

Having considered all of the information from the table above and taking into account our previous experiences with both Angular and React we chose React as our front end development technology. Choice of FrontEnd Technology - can be looked at from 2 perspectives 1. Learning outcomes. 2. Good fit for the task we were trying to achieve within the project.

Having completed the project, we feel it had been a good choice from both of those perspectives.

3.1.5 MongoDB Atlas

MongoDB Atlas is a database as a service, a cloud based service. When designing the application we were unsure what format our data would take, if the data was purely relational it would only need a SQL type database, if the data was non relational it would need a noSQL database. NoSQL databases (aka "not only SQL") are non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads. NoSQL databases can store relationship data—they just store it differently than relational databases do. When compared with SQL databases, many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn't have to be split between tables. The simplest type to describe is the document database, in which it would be natural to combine both the basic information and the customer information in one JSON document. In this case, each of the

SQL column attributes would be fields and the details of a customer's record would be the data values associated with each field.[22]

For example: Last_name: "Jones", First_name: "Mary", Middle_initial: "S", etc .

For our data storage we decided to use a noSQL Database as this stored data in the way we required. We could have created a cloud based sql database using something like Postgres, but the amount of data we were handling did not require separation. In the end most of our data was relational data but we felt that MongoDB Atlas was still a good fit as it allowed us to add other noSQL type data if the need arose. Each document has a different number of fields, size, content, and is stored in a JSON like format called BSON. The documents in MongoDB do not need to have a schema defined beforehand. The fields can be created on the fly. The data model available within the MongoDB allows developers to represent the hierarchical relationships, store arrays, and other more complex structures easily. The mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations.[22] We used the Mongo shell to upload files to the database from a local system. This tool was used to upload large csv files with over 4000 lines to a collection in our database. We used MongoDB Atlas Database, a product from MongoDB which offers MongoDB as a cloud based database(DBaaS). We accessed data stored on MongoDB, via api calls from our flask api. From this data graphs can be generated. We also stored data inputted by the application users, for example ratings on the home page or a symptoms tracker. This data could be retrieved from MongoDB and displayed on the application. this data was posted to the database using an api call. This data could also be retrieved.

An aspect of using MongoDB Atlas is how the database is structured in terms of being a distributed system. It also offered peace of mind about the integrity of our data and we knew that the data would not be lost, or that the service would always be continuous, which is a fantastic option for a free version of the product. This helped solidify some of the central properties of distributed systems which we learned about in the Distributed Systems Module.

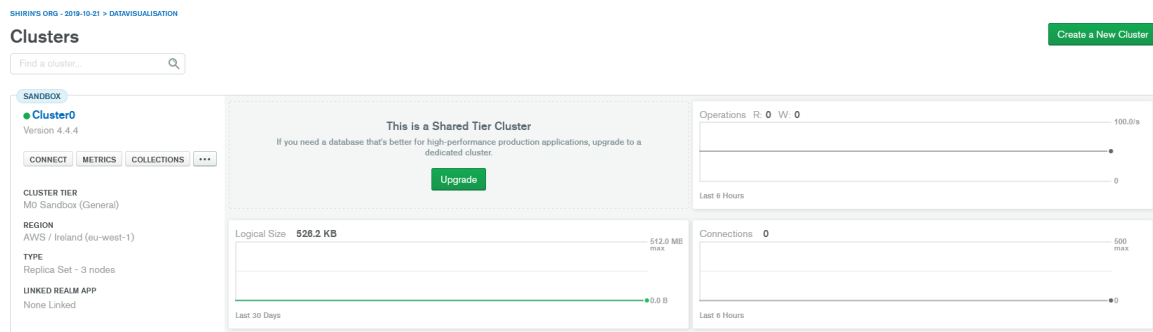


Figure 3.5: MongoDB Atlas Cluster

3.2 Languages Used

3.2.1 JavaScript

JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. Over 97% of websites use it client-side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but they are now core components of other software systems, most notably servers and a variety of applications.[23] Parts of the project were written in JavaScript, these include Live Covid-19 data visualisation, home page, SARs and MERs.

3.2.2 JSX

JSX is an extension of Javascript and stands for JavaScript XML. JSX allows us to write HTML elements in JavaScript and place them in the DOM without any `createElement()` and/or `appendChild()` methods. JSX converts HTML tags into react elements.[24] React encourages the use of JSX for development, though a developer could use JavaScript if preferred. JSX was

used in the stats page for the creation of the country drop down, overlay graph and interactive bar chart feature.

3.2.3 Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.[25] Flask is written in python, the api backend of the project is written on python code. All api routes use Flask Blueprint(). Flask Blueprints encapsulate functionality, such as views, templates, and other resources.

3.3 Development Environment

3.3.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).[26] According to Visual Studio Code,"Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas."[27]

The team agrees, VS code is extremely easy to use, is much faster than other IDE's, the inbuilt support for JS and Node.js is something we considered, it also allowed us to develop the backend which was written in python, without having to use another IDE. It allowed seamless integration between the different coding languages.

3.4 Firebase

Firebase is a Backend-as-a-Service that was founded in 2011 that provided developers an API that enables integration of online chat functionality into their websites.

In 2014, Google acquired Firebase and has since built it into a full fledged mobile and web development platform with over 19 different products and services with more than 1.5 million developers worldwide.



Figure 3.6: hosted in the cloud

Firebase uses what is known as a NoSQL database for storing data in a Realtime Database. As the name suggest this means that data is not stored in the tables and rows found in relational database management systems (RDBMS) such as Oracle Database or Microsoft SQL Server. Nor is the data accessed using Structured Query Language (SQL) statements. Instead, the data is stored in the form of a JSON object.

JSON is an acronym for JavaScript Object Notation and it defines a syntax used to transmit data in a format that is both lightweight and easy for both humans and software to read, write and understand.

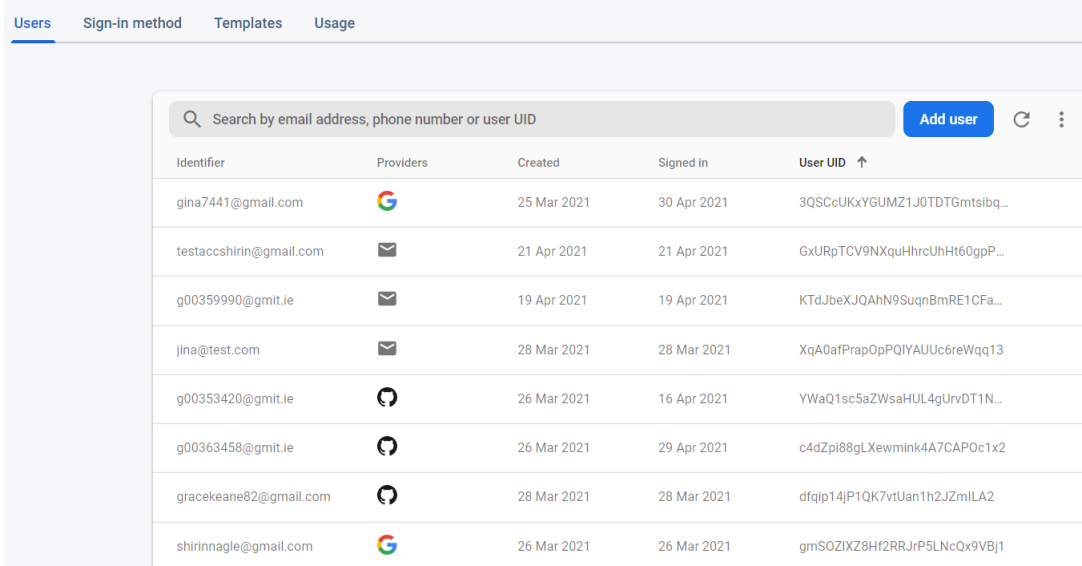
JSON objects typically consist of a key/value pair, where the key uniquely identifies the object within the database and the value represents the data that is being stored. Multiple JSON objects are structured in the form of a JSON tree.

The Firebase SDK supports programming in C++, Java, JavaScript, JavaScript/Node.js, Objective-C, and Swift. Angular, Backbone, Ember and React are supported through bindings to the database.[28]

3.4.1 User Authentication

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made User Interface libraries as shown in Figure 3.7 to authenticate

users to our app.



The screenshot shows the 'Users' tab in the Firebase console. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. Below these is a search bar with the placeholder text 'Search by email address, phone number or user UID' and an 'Add user' button. The main content is a table with the following columns: Identifier, Providers, Created, Signed in, and User UID (with an upward arrow icon). The table contains 9 rows of user data.

Identifier	Providers	Created	Signed in	User UID ↑
gina7441@gmail.com		25 Mar 2021	30 Apr 2021	3QSCcUKxYGUMZ1J0TDTGmtsibq...
testaccshirin@gmail.com		21 Apr 2021	21 Apr 2021	GxURpTCV9NXquHrcUhHt60gpP...
g00359990@gmit.ie		19 Apr 2021	19 Apr 2021	KTdJbeXJQAhN9SuqnBmRE1CFa...
jina@test.com		28 Mar 2021	28 Mar 2021	XqA0afPrapOpPQIYAUUc6reWq13
g00353420@gmit.ie		26 Mar 2021	16 Apr 2021	YWaQ1sc5aZWsaHUL4gUrvDT1N...
g00363458@gmit.ie		26 Mar 2021	29 Apr 2021	c4dZpi88gLXewmink4A7CAPOc1x2
gracekeane82@gmail.com		28 Mar 2021	28 Mar 2021	dfqip14jP1QK7vtUan1h2JZmlLA2
shirinnagle@gmail.com		26 Mar 2021	26 Mar 2021	gmSOZIXZ8Hf2RRJrP5LNcQx9VBj1

Figure 3.7: User data to the Database

It supports authentication using email and password accounts, phone auth, Google, Facebook, Twitter, and GitHub login, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.[29]

To sign a user into our app, we first get authentication credentials from the user. These credentials can be the user's email address and password, or an OAuth token from a federated identity provider. Then, we pass these credentials to the Firebase Authentication SDK. See Figure 3.8 for login and identity code.[29]

Our backend services will then verify those credentials and return a response to the client. After a successful sign in, it will allow the navigation of sensitive routes. Authenticated user will access routes that previously locked.

3.4.2 Realtime database

The Firebase Realtime Database is a cloud-hosted NoSQL database. Data is synced across all clients in real-time and remains available even when an app goes offline. Whenever you update data in the real-time database, it stores the data in the cloud and simultaneously notifies all interested devices in milliseconds. The real-time database is also optimized for offline use.


```
const onSubmit = async (event) => {
  event.preventDefault();
  try {
    let data;
    if (newAccount) {
      //create account
      data = await authService.createUserWithEmailAndPassword(email, password);
    } else { //log in
      data = await authService.signInWithEmailAndPassword(email, password);
    }
    console.log(data);
  } catch (error) {
    setError(error.message);
  }
};
```

Figure 3.8: user login and identity

Whenever a user loses thier connection, the database STK uses a local cache on the device to serve and store changes. This means that when the user comes back online, their local data is automatically synchronized. To keep our data secure, we used database security rules. The security rules are securely stored with the real-time database on our server.[30]

We wanted to use Mers and Sars data to display D3 graphs for our group project. We could not find an API that provided the data we required, but we were able to find all the data we needed in CSV format from various sources. We parsed all that CSV data into JSON and used Firebases' Command-line interface to upload all the data into the server, as shown in Figure 3.9.

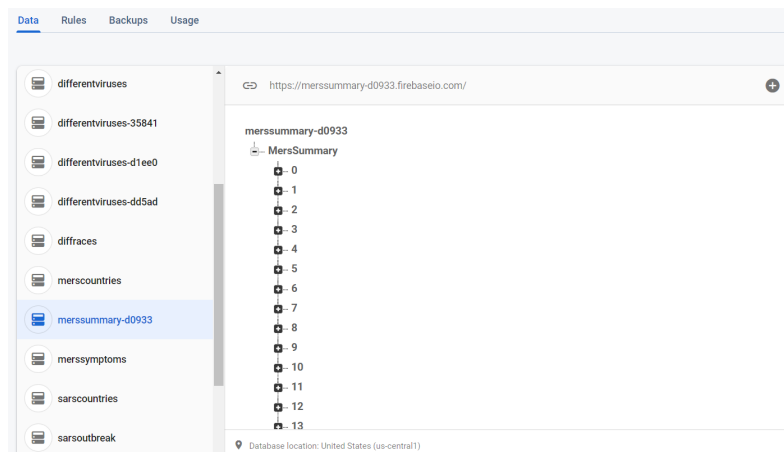


Figure 3.9: Storing and managing our data

3.5 Cloud Firestore

Cloud Firestore is Firebase’s fully managed cloud-native NoSQL document database that is fast and serverless. It’s a new and improved version of the Real-Time Database, and its capabilities include real-time updates, offline synchronization, scalability, and multi-region deployment. Unlike a SQL database, there are no tables or rows. Instead, you store data in documents, which are organized into collections. Each document contains a set of key-value pairs. Cloud Firestore is optimized for storing large collections of small documents. All documents must be stored in collections. Documents can contain subcollections and nested objects, both of which can include primitive fields like strings or complex objects like lists.[31]

We used Cloud Firestore for our Message Board. User identity is an important security concept. Different users have different data, and sometimes they have different capabilities.

For example each message is associated with the user that created it. Users may also be able to delete their own messages, but not messages posted by other users.

The SDK handles all the state management and data syncing in between the client and server. A collection of messages was set up and for each of these messages a sub collection of messages was created.

If we look at the actual database we can see that each message has an owner and then each individual message is an object. See Figure 3.10 for an example of storing and managing messages.

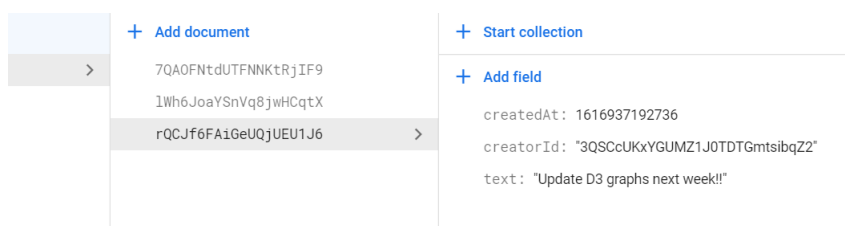


Figure 3.10: Storing and managing messages

The database contains the user ID, a created timestamp and the message content. When a user is logged in and they created a new message it will create this document with their user ID as the owner ID. See Figure 3.11 as an example of the code.

```
const onSubmit = async (event) => {
  event.preventDefault();

  await dbService.collection("messages").add({
    creatorId: creatorIds,
    text: message,
    createdAt: Date.now(),
  });
  setMessage("");
};
```

Figure 3.11: How to add message to the message collection

3.6 D3

D3.js (Data-Driven-Documents) is an open-source JavaScript library that lets you create dynamic data visualizations in web browsers using SVG, HTML 5, and CSS. Most data visualization tools require Python, D3.js visualizations are created entirely using JavaScript.

While most charting libraries (such as Chart.js and Highcharts) provide ready made charts D3 consists of a large set of building blocks from which custom charts or maps can be constructed.

D3's approach is much lower level than other charting libraries. Creating a bar chart with Chart.js is just a few lines of code.[32]

Creating the same chart with D3 you need to:

- **create SVG rect elements and join them to the data**
- **position the rect elements**
- **size the rect elements according to the data**
- **add axes**

In order to **use DOM in React directly**, we needed to use refs. See Figure 3.12 for an example of refs.

It takes two simple steps:

- **creating a ref in constructor()**
- **connecting [div] to the ref**

See Figure 3.12 for ChartWrapper.js code.

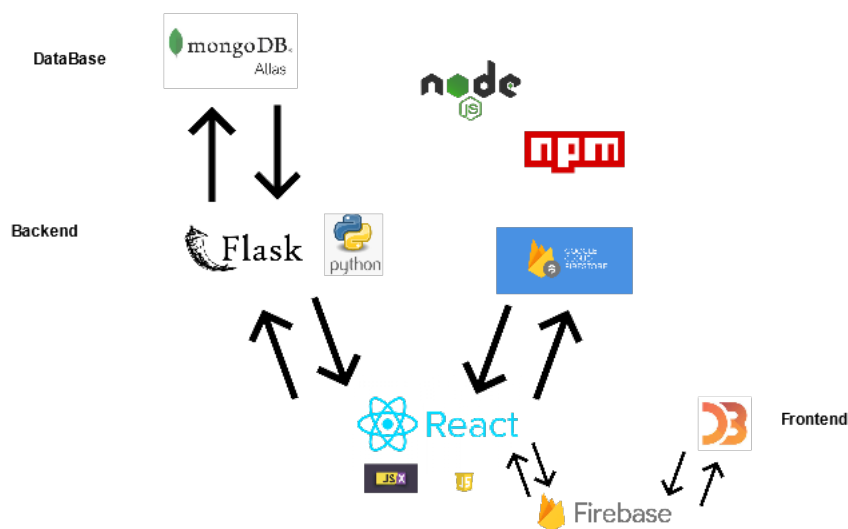
```
1 import React, {Component} from 'react';
2 import D3MersSars from './D3Components/D3MersSars';
3 import styled from 'styled-components';
4
5 const Wrapper = styled.div`
6   background: #f5f6fa;
7 `;
8
9 export default class ChartWrapper extends Component{
10
11   componentDidMount(){
12     this.setState({
13       //To make an instance of D3Chart class
14       chart : new D3MersSars(this.refs.chart)
15     })
16   }
17
18   //This allows me specify whether or not I want to re-render a react componenet when something changes
19   shouldComponentUpdate(){ //render() will not be invoked if shouldComponentUpdate() returns false.
20     return false
21   }
22
23   //componentWillReceiveProps(nextProps){
24   UNSAFE_componentWillReceiveProps(nextProps){
25     this.state.chart.update(nextProps.virus)
26   }
27
28   render(){
29     //lifecycle method uses the ref system to keep track of an element
30     return <Wrapper>
31       <div ref="chart"></div>
32     </Wrapper>
33   }
34 }
35 }
```

Figure 3.12: How to access D3

Chapter 4

System Design

System Architecture



4.1 Architecture Overview

The architecture of the project is a three tiered architecture, consisting of a front-end, a back-end and a data tier. Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user

interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed. The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers.[33] Our architecture also includes a login facility using Firestore and some of the static data for the components rendering D3 data is stored in Firebase. The project also uses Firebase to access static data for D3 visualisation and manages user login authentication. Firestore is used to manage a message board which is only accessible for an authenticated user.

The front end was worked on initially, once the front-end was functioning a decision was made about the architecture of the project. Flask was decided on as the back-end, which would interact with the database tier and the front-end. Once the architecture was finalised, we wrote our own api calls to the database.

We had decided on Flask because React does not tie a developer to use a particular back-end, and we wanted to learn more about Flask and python, which we had some familiarity with.

4.2 Live API Calls

Data visualization gives us a clear idea of what the information means by giving it visual context through graphs. This makes the data more natural for the human mind to comprehend and therefore makes it easier to identify trends and patterns within large data sets. Having attractive and clear visualizations are important but it is just important to work with reliable and correct data so as not to mislead users.

The most important aspect of developing a data visualization application is the data itself. It is essential for data visualization applications to fetch data from a source that is up to date and reliable. When developing the live API component and after much research the team felt it would be best to use Disease.sh API and MathDroid API to gather live Covid-19 data from around the world.

Disease.sh is a popular open source API for disease related statistics that has handled over 45 billion API requests to date. Both site source their data from John Hopkins University, the New York Times, Worldometers and Apple reports to give comprehensive, informative and most importantly reliable data. Responses from these APIs are served in JSON format which allows for easy and reliable integration.[34]

See Figure 4.1 for a code example of how data was fetched from Disease.sh

API and assigned features to be used with the API data through the use of JavaScript.

```
// Fetching api data
useEffect(() => {
  fetch("https://disease.sh/v3/covid-19/all")
    .then((response) => response.json())
    .then((data) => {
      setCountryInfo(data);
    });
}, []);

// useEffect runs a piece of code based
// on a given condition
useEffect(() => {
  // Code run once when the component loads
  const getDropDownCountries = async () => {
    // Send a req to server, wait, do something
    await fetch("https://disease.sh/v3/covid-19/countries")
      .then((response) => response.json())
      .then((data) => {
        // Restructure countries
        const dropcountries = data.map((country) => ({
          name: country.country, // e.g. Ireland
          value: country.countryInfo.iso2, // e.g. IRE
        }));
        const sortedData = sortData(data);
        setDropDownCountries(dropcountries);
        setTableData(sortedData);
        setMapCountries(data);
      });
  };
}, []);
```

Figure 4.1: Live API Calls

4.3 Three tier architecture

4.3.1 Data Tier

The project relies on daily updated apis for Covid19 data. As part of the initial project we investigated other pandemics and epidemics. Fortunately these occurred years ago, unfortunately it meant that the data was not easy to find. We found some limited data on SARs MERs and Smallpox. We chose a cloud based database MongoDB Atlas to manage some of the data required for the project. The smallpox data was stored in MongoDB and the SARs and MERs data was stored in Firebase. Firebase was used for data that rendered D3 images. External api's were used for the Covid-19 data. The app allows a user to record temperature, symptoms and leave a rating or review, this information is stored in MongoDB and can be retrieved. Everytime the a temperature is entered a temperature graph dynamically changes.

4.4 Logic Tier - Back end

Using RESTful API's which follow an architecture that use predefined and stateless operations to access web resources. The API is written with Flask, uses Pymongo, a python library specifically for interacting with a Mongo database. Requests from the app are sent by a request from the code in the

front end, using JavaScript or JSX to the RESTful API, this API sends a connection request to MongoDB Atlas which is on the web. Flask retrieves the information and sends it to be displayed at the application end.

4.5 Application Tier - Front end

React was used to develop the client side of the application. React is an open source JavaScript library for rendering views. React apps are an example of single application applications (SPA), the application code, like HTML, CSS and Javascript is loaded once. The SPA only sends what the user needs, for example by clicking on a button only the information associated with that button is reloaded not the whole page. This piece by piece, client side method makes load time much faster for users and makes the amount of information a server has to send a lot less. When a user interacts with the app JavaScript intercepts the browser events and depending on where the data is coming from different API calls are made, for example if the Data is from the MERs or SARs dataset an API call is made to Firebase. If data is required from the Smallpox data or a users inputted information an API call is made using Flask and connected to the relevant MongoDB Atlas collection.

- Flask
- React.js
- MongoDB Atlas
- Firebase

These technologies were chosen as they give developers the ability to handle large volumes of data and, at the same time, render a nice modern interface on the client side.

4.6 Heroku - Deployment



Heroku is a platform as a service(PaaS) based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps.[35] We deployed the project to the web using Heroku. We investigated different options for deployment, there are many companies that offer a free option or trial PaaS product, companies like Plesk, Cloudify and Heroku, as well as Open Source PaaS tools. The project deployment was implemented towards the end of project development, because we had prior experience with Heroku and limited time we decided to try Heroku first. If we felt that Heroku was not suitable we would try another platform like Plesk or an open source option like Dokku, but Heroku worked well and with the time frame left we decided there was no need to investigate another option. Heroku's free option was sufficient for our app deployment.

Chapter 5

System Evaluation

5.1 Chapter Overview

In the System Evaluation chapter we will touch on the overall design of the system. We will evaluate whether or not the objectives were met in the completion of the web application. The testing that went into the project will also be documented. We will also be reviewing any limitations and areas of improvement within this project.

5.2 Project Objectives

Initially the goal of this project was to develop a data visualization website that analyses various viruses such as Covid-19, Sars, Mers and Smallpox using different technologies.

5.3 Testing

This section will discuss how the application was tested. The entire system was fully tested to ensure that there were no faults in our web application. We felt that it was essential to fully test our web application in the development as well as in the completion of the project. Some of the testing that we conducted included Unit Testing and System Testing.

5.3.1 Unit Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of

the software code performs as expected.

A unit may be an individual function, method, procedure, module, or object.[36] The reason that it is done this way is to verify that each unit of our application is working correctly.

Jest

Jest is a JavaScript unit testing framework, used by Facebook to test services and React applications. Jest also provides Snapshot testing, the ability to create a rendered ‘snapshot’ of a component and compare it to a previously saved ‘snapshot’. The test will fail if the two do not match.[37]

Jest was used to test both Firebase and Firestore functions. We researched different options for testing Firebase and Firestore, from our research we concluded that Jest was a common framework used with good documentation available. See Figure 5.1 for testing with Jest.

```

1  const firebase = require('@firebase/testing') //<---this should be on top!
2  const admin = require('firebase-admin')
3
4  //const projectId = "testfirebaseemulators"
5  const projectId = "projectauth-466ef"
6  process.env.GCLOUD_PROJECT = projectId
7  process.env.FIRESTORE_EMULATOR_HOST = "localhost:8082";
8  let app = admin.initializeApp({projectId}) //so that you wouldn't need any permission to write,delete or update
9  let db = firebase.firestore(app) //firestore instance which is going to be calling be firebase
10
11
12  beforeAll(async ()=>{
13    await firebase.clearFirestoreData({projectId});
14  })
15
16  // When Document written to '/Testcollection/{DocumentId}' , trigger function to copy it to '/copies/{DocumentId}'
17  test("Expect to find a copy in 'Copies' collection", async ()=>{
18    const testDoc = {
19      name: 'Sam',
20      age: 21,
21      city: 'Galway'
22    }
23
24    const ref = db.collection('Testcollection').doc()
25    await ref.set(testDoc)
26    //jest.setTimeout(30000);
27
28    const copyId = ref.id
29
30    const copyRef = db.collection('Copies').doc(copyId)
31
32    await new Promise((r)=>setTimeout(r,30000))
33
34    const copyDoc = await copyRef.get()
35
36    expect(copyDoc.data()).toEqual(testDoc)
37  })

```

Figure 5.1: Testing with Jest

Firestore Emulators

The Firebase Local Emulator Suite consists of individual service emulators built to accurately mimic the behavior of Firebase services. This means we can connect our app directly to these emulators to perform integration testing or QA without touching production data.[38]

The Firebase Emulators make it easier to fully validate the app's behavior and verify our Firebase Security Rules configurations. We used the Firebase Emulators to run and automate unit tests in a local environment.

```

i ui: Emulator UI logging to ui-debug.log
i functions: Watching "C:\Users\gina7\Desktop\FinalProject\Final-Year-Project-Data-Visualization\function\functions..."
+ functions[testCollectionTriggers-onCreate]: firestore function initialized.

✓ All emulators ready! It is now safe to connect your app.
i View Emulator UI at http://localhost:4000


```

Emulator	Host:Port	View in Emulator UI
Authentication	localhost:9095	http://localhost:4000/auth
Functions	localhost:5002	http://localhost:4000/functions
Firestore	localhost:8083	http://localhost:4000/firestore
Database	localhost:9001	http://localhost:4000/database
Hosting	localhost:5000	n/a

```

Emulator Hub running at localhost:4400
Other reserved ports: 4500

```

Figure 5.2: Testing with Firebase Emulator

Postman

Postman is an application for testing APIs, by sending request to the web server and getting the response back. It allows users to set up all the headers and cookies the API expects, and checks the response.[39]

```
1 from flask import Flask, request, jsonify
2
3 app = Flask(__name__)
4
5 @app.route('/test', methods=['GET', 'POST'])
6 def test():
7     if request.method == "GET":
8         return jsonify({"response": "Get Request Called"})
9     elif request.method == "POST":
10         req_json = request.json
11         name = req_json['name']
12         return jsonify({"response": "Hi there " + name})
13 if __name__ == "__main__":
14     app.run(debug=True)
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Figure 5.3: Testing API

Figures 5.4 and 5.5 show a GET and POST request using Flask.

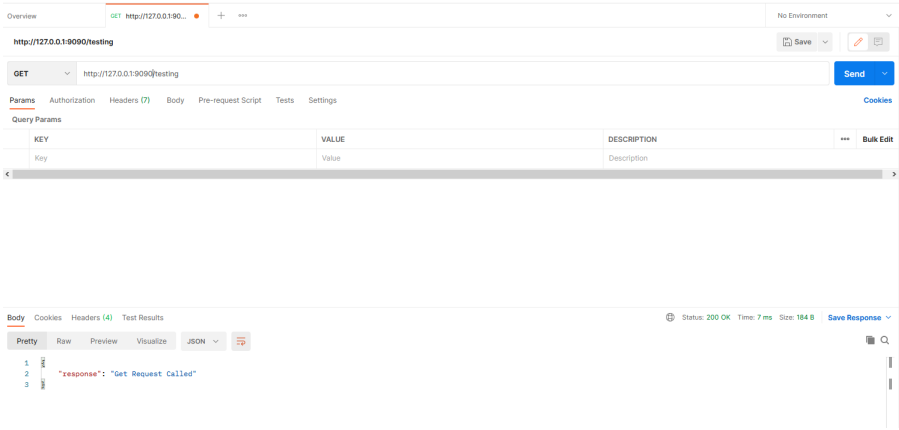


Figure 5.4: API Get Request

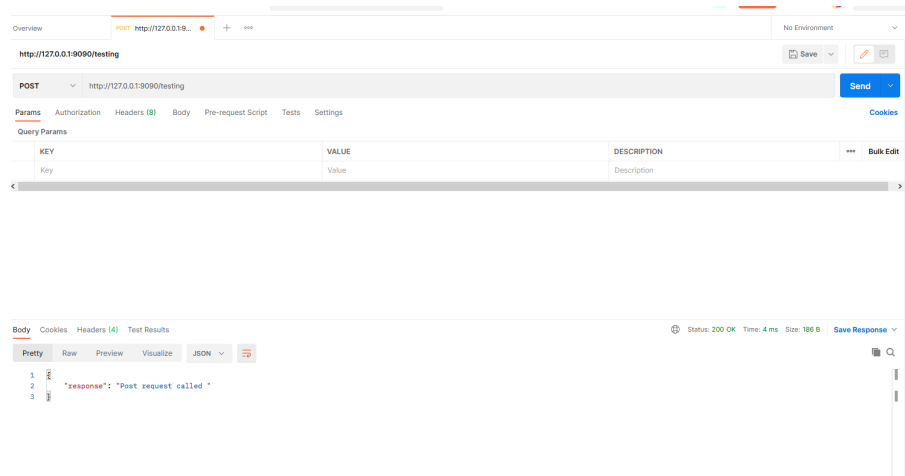


Figure 5.5: API Post Request

5.3.2 System Testing

Selenium

Selenium is a portable framework for testing web applications.[40] This allows the user to create tests which will monitor the users actions performed and log them so that these tests can be run and re-run with the click of a button. Selenium was used to test the system functionality to determine if everything worked as expected. These tests were run on Firefox to test essential functionality, including drop down lists in stats, Covid-19, Mers/Sars, button clicks and for the page navigation links.

Selenium uses automated clicks to check the functionality of application. In Figure 5.6 the projects Heroku URL is specified for Selenium to run pre-written tests against. By defining specific actions on features by adding their Xpath, Selenium can automate an isolate tests against the specified path to determine if the feature is working smoothly and as expected.

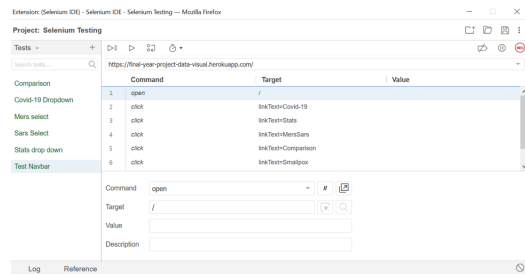


Figure 5.6: API Post Request

5.4 Limitations and areas of Improvement in this project

5.4.1 Limitations

At the time of submission there are a few limitations. Our application is at a prototype stage of development. Currently our application works fully locally. It is desirable for us to have the application up and running on the cloud. The project is deployed on the cloud using Heroku as a Platform as a service. The deployment is not complete as we encountered issues with deploying flask, to Heroku. As a result the deployed project has some functionality missing, such as our rating and checking temperature functionalities, but most elements are working as expected.

5.4.2 Improvements

Improvements that can be made to the current system could include adding more features. Initially, we had an idea of having an online symptom checker by using a chatbot powered by artificial intelligence. The user could diagnose symptoms for signs of Covid-19. More interactive elements. For example the project could have Covid-19 news tracker or worldwide headlines for Covid-19.

5.4.3 System Evaluation Conclusion

Overall we are happy with how the project turned out. We feel that we have covered all of the objectives stated in the introduction chapter. We investigated the field of data visualisation, we researched other like applications like epiviewer.[6]

We evaluated each potential framework against each other and from this evaluation chose what we felt was most appropriate for our development. We researched modern technologies how applicable they would be in our software development careers. All application developments were met. We adhered to the outlined metrics for success and failure, the scope of the project was sufficient for a three person project and we learned new technologies while also improving our existing skills in familiar technologies.

Chapter 6

Conclusion

6.0.1 Context Objectives

Selecting the topic of data visualisation was partly inspired by an interest in how data visualisation can impart a lot of information without the need for reading extensive data and inspired by the Covid-19 pandemic and lock-down. The overall goal of the project was to investigate different methods of data visualisation, and to present our findings to an end user via a web based application. We investigated the field of data visualisation and the scope for the project. We evaluated and researched the appropriate frameworks and tools available for development. We incorporated state of the art technologies, frameworks, databases and tools that allow users to view, add, update, delete, post and retrieve data. The web application, allows users to sign-up, login, view data visualizations as well as incorporate user interaction features, create unique user visuals and post user data to databases.

6.0.2 Methodology Objectives

The team found working entirely remotely challenging, however we overcame that. Working remotely is its own discipline and one that is definitely worth learning. Our choice of implementing the Agile methodology was ideally suited to remote collaborative teamwork. Weekly sprints and using user stories helped us stay focused on the task. The team had weekly meetings with just the team and separate weekly meetings with our supervisor. The fact that we had to work remotely made us more aware of the need to be in frequent communication with each other, if we were on campus there would be much more opportunistic meetings about the project, which would have made resolving complex issues easier.

6.0.3 System Evaluation Findings

- **Importance of testing**

Towards the end of development the team became aware of the importance of testing. It is possible that we could have implemented testing earlier in the product development. Initially we used trial and error as our method of testing but realised that if we could have implemented Test Driven Development or Behaviour Driven Development it may have helped with the project.

- **Working with different technologies**

The team feels that we have benefited from working with the technologies we chose. Often we faced problems with certain aspects of different technologies but usually found a way of working through the problem. These problems helped hone our problem solving skills. By reading the relevant documentation to try and figure out how and why everything would work together. However, we realised how important research is when creating a software application.

6.0.4 Opportunities for future Investigation

As the project is at a prototype stage there is plenty of scope to research further features.

- **Fetch Covid-19 headlines from external news outlets**

The team felt it would have been useful to fetch live news headlines from external news websites, if we had more time this unique feature would allow users to access real time Covid-19 updates while still interacting with the application.

- **More data on different epidemics**

It would be interesting to find other epidemic data sets, for example Ebola or Polio and generate interactive charts and graphs from these.

- **Overlay data**

With more data sets it may be possible to track older epidemics and overlay generated graphs to see if there was any new information that could be learned from this comparison.

- **More querying of the existing databases**

We could write more API calls to the database and generate many different types of charts, or map information to other types of graphs.

- **Upload User Data** We could set up an option for a user to upload their own data sets and generate different visualisation graphs based on this data.

6.0.5 Group Reflections

The benefit of hindsight is a wonderful thing, if we had to do it again there are many things we would probably do differently. We would try to manage our time more effectively. In hindsight we would make better plans for the project and would know which areas to prioritise and focus on more. We would be more decisive about what did and did not work and cut our losses earlier. Try to improve our research of topics that posed difficulties. We most definitely would deploy the project at an earlier stage of development. This would allow us more time to resolve any deployment issues. The project is currently deployed on Heroku but we ran out of time to fix an issue we were having with Flask connection. Now that we know more about the area of testing we would incorporate some elements of testing methodologies in the project, particularly Behaviour Driven Development as this is an agile software development process. Having completed the project we feel that if we were to do it again we would be in a very good position to know what worked, what did not work and how to tackle these problems earlier. We definitely would be able to manage our time better a second time around.

We found the project itself difficult at times but we had an excellent working relationship. It is rewarding to think that we completed this project in an unusual environment, normally we would be on Campus and this would have given us more opportunities to meet up between labs and lectures, which we feel would have made the project a bit easier in terms of coordination and especially writing the dissertation.

There are many achievements from working on a group project. The biggest achievements we gained from this module would be team working and problem-solving skills. As we had weekly meetings on Microsoft Teams and discussed everything we needed to complete this project we feel that this has helped us improve in areas like communication and decision making. An invaluable outcome is the knowledge that what we have learned from doing this project is applicable in many other fields that rely on understanding fast moving data. Fields like logistics, urban planning and traffic flow management. The same underlying principles apply in terms of collating separate streams of information and bringing them together in clear and coherent visualisations.

In times of crisis the technological tools that are used to face the crisis may be of lasting benefit after the crisis has passed. Remote working and learning have certainly become the norm for many, and data visualisation may become a key tool in understanding an increasingly complex world.

Appendix A

Appendix

A.1 GitHub Link

Final Year Project Link

A.2 Heroku

Heroku Link

A.3 Screencast Link

Screencast Link

Bibliography

- [1] Britannica, “How long did the flu pandemic last.” ”<https://www.britannica.com/story/how-long-did-the-flu-pandemic-of-1918-last>”, 2020.
- [2] J. Lammers, J. Crusius, and A. Gast, “Correcting misperceptions of exponential coronavirus growth increases support for social distancing,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 28, pp. 16264–16266, 2020.
- [3] B. Preim and K. Lawonn, “A survey of visual analytics for public health,” in *Computer Graphics Forum*, vol. 39, pp. 543–580, Wiley Online Library, 2020.
- [4] C. Tebé, J. Valls, P. Satorra, and A. Tobías, “Covid19-world: a shiny application to perform comprehensive country-specific data visualization for sars-cov-2 epidemic,” *BMC Medical Research Methodology*, vol. 20, no. 1, pp. 1–7, 2020.
- [5] K. Fontichiaro, “Using data visualizations to spark inquiry conversations about health,” *Teacher Librarian*, vol. 48, no. 3, pp. 55–63, 2021.
- [6] S. Thorve, M. L. Wilson, B. L. Lewis, S. Swarup, A. K. S. Vullikanti, and M. V. Marathe, “Epiviewer: an epidemiological application for exploring time series data,” *BMC bioinformatics*, vol. 19, no. 1, pp. 1–10, 2018.
- [7] Gapminder, “About gapminder.” ”<https://www.gapminder.org/about/about-gapminder/history/>”, 2018.
- [8] T. Lancet, “The lessons of smallpox eradication for covid-19.” ”[https://www.thelancet.com/journals/lancet/article/PIIS0140-6736\(20\)32713-6/fulltext](https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)32713-6/fulltext)”, 2020.
- [9] C. F. Tovissodé, B. E. Lokonon, and R. Glèlè Kakaï, “On the use of growth models to understand epidemic outbreaks with application to covid-19 data,” *Plos one*, vol. 15, no. 10, p. e0240578, 2020.

- [10] K. Petersen, C. Wohlin, and D. Baca, “The waterfall model in large-scale development,” in *International Conference on Product-Focused Software Process Improvement*, pp. 386–400, Springer, 2009.
- [11] S. Balaji and M. S. Murugaiyan, “Waterfall vs. v-model vs. agile: A comparative study on sdlc,” *International Journal of Information Technology and Business Management*, vol. 2, no. 1, pp. 26–30, 2012.
- [12] G. Kumar and P. K. Bhatia, “Impact of agile methodology on software development process,” *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, vol. 2, no. 4, pp. 46–50, 2012.
- [13] hygger.io, “Popularity of agile;” ”<https://hygger.io/blog/why-agile-is-so-popular-in-project-management/#:~:text=Agile%20projects%20are%2028%25%20more,use%20Agile%20in%20their%20work.&text=27.4%25%20of%20manufacturing%20companies%20rely,on%20the%20combination%20of%20methodologies.> ”, 2018.
- [14] GitHub, “Usage of github tasks.” ”<https://github.com/features/project-management/>”, 2017.
- [15] K. Schwaber, “Scrum development process,” in *Business object design and implementation*, pp. 117–134, Springer, 1997.
- [16] Git, “About git.” ”<https://git-scm.com/>”.
- [17] GitHub, “About github.” ”<https://guides.github.com/activities/hello-world/>”.
- [18] Wikipedia, “Flask framework.” ”[https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))”.
- [19] Flask, “Virtual environments,” <https://flask.palletsprojects.com/en/1.1.x/installation/>.
- [20] React, “About react.” ””.
- [21] Stackoverflow, “Most loved dreaded and wanted.” ”<https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted>”.
- [22] MongoDB, “About mongodb.” ”<https://www.mongodb.com/nosql-explained>”.

- [23] Wikipedia, “Javascript.” ”<https://en.wikipedia.org/wiki/JavaScript>”.
- [24] W3Schools, “Jsx.” ”https://www.w3schools.com/react/react_jsx.asp”.
- [25] Wikipedia, “Python.” ”[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))”.
- [26] V. Code, “Visual studio code.” ”<https://code.visualstudio.com/docs>”.
- [27] V. Code, “Why visual studio code.” ”<https://code.visualstudio.com/docs/editor/whyvscode>”.
- [28] Codeburst, “Firebase.” ”<https://codeburst.io/firebase-spend-more-time-developing-the-features-that-actually-matter-814f57>”.
- [29] Google, “Firebase products.” ”<https://firebase.google.com/products/auth>”.
- [30] Google, “Firebase database.” ”<https://firebase.google.com/products/realtime-database>”.
- [31] Google, “Firestore.” ”<https://firebase.google.com/products/firestore>”.
- [32] Educative.io, “D3 tutorial.” ”<https://www.educative.io/blog/d3js-tutorial>”.
- [33] IBM, “Three tier architecture.” ”<https://www.ibm.com/cloud/learn/three-tier-architecture>”.
- [34] disease.sh, “Api.” ”<https://disease.sh/>”, 2021.
- [35] Heroku, “Heroku platform.” ”<https://www.heroku.com/platform>”.
- [36] Wikipedia, “Unit testing.” ”https://en.wikipedia.org/wiki/Unit_testing”.
- [37] Wikipedia, “Js framework.” ”[https://en.wikipedia.org/wiki/Jest_\(JavaScript_framework\)](https://en.wikipedia.org/wiki/Jest_(JavaScript_framework))”.
- [38] Google, “Firebase emulator.” ”<https://firebase.google.com/docs/rules/unit-tests>”.

- [39] medium.com, “Testing api’s.” ”<https://medium.com/aubergine-solutions/api-testing-using-postman-323670c89f6d>”, 2018.
- [40] Wikipedia, “About selenium.” ”[https://en.wikipedia.org/wiki/Selenium_\(software\)](https://en.wikipedia.org/wiki/Selenium_(software))”.