Solent University

Faculty of Business, Law and Digital Technologies

Digital Design & Web Development

2020 - 2021

Krisztian Nagy

“C-19 Travel App”

Supervisor: Martin Reid

Date of submission: 06.05.2021

Acknowledgements

I would like to express my special thanks and gratitude to my course leader and project supervisor Martin Reid for his persistent work and the guidance over the years within the course and for the possibility to conduct this project that allowed me to gain in depth knowledge in the topic area and web development. I would like to thank Joe Appleton for the preparation, teaching on the fundamentals and leading with modern, industry standard technologies. Secondly I would like to thank my parents for their continuous and devoted support.

Abstract

The project is trying to identify current problems that people are experiencing due to the current pandemic and aims to help a specific target audience by producing a software that provides solution(s) to the identified problems. People who need to travel during the pandemic must provide specific documents and need to deal with limitations on how the services can be used. The project seeks the problems they are experiencing and provides solutions. The project is also facing the challenges of the implementation of modern technologies using methodologies such as Software Development Life Cycle and Waterfall methodology to provide those non-functional requirements that are necessary to the target audience. These requirements are the mobile friendly approach, shallow style library, server-side rendering, and the implementation of service-workers. The document presents an insight how the management and methodologies were used and how the requirements were satisfied with technical solutions.

Table of Contents

List of Figures

1. Introduction

The current pandemic caused the Government to practice restrictions. These restrictions affect parts of the economy by closing any shops that are not essential for everyday living. Another restriction that prohibits meetings over a certain number of group members and traveling is allowed only with the right documentation that must be submitted before taking action. These practices limit the everyday life of people across the country. The latter restriction by the Government requires travelers to provide information on their traveling along with their identity. According to the Government website, travelers must turn in a passenger locator form not earlier than 2 days before departure. The combination of restrictions and requirements causing uncomfortable situations around the world for those who want to travel.

The project aims to deliver a product that can solve or help to solve the current problems people are experiencing caused by the restrictions and requirements. The product is a downloadable web application that allows users to gain information not just by seeing an article of requirements on the internet, but to see exact, comparable data that can be used to evaluate the state of the country and how they handle the virus. Moreover, the application provides such features that allows users to rely on the application in terms of carrying their documentation and offers the possibility to extend the information presented in the product. The challenges of the project was the implementation of methodologies such as Software Development Life Cycle and Waterfall methodology to manage the project and the usage of modern technologies that allowed to develop the application with those features that fit for the target audience, such as the server-side rendering that helps accessibility by rendering the pages on the server before the mobile client would load it. Thus mobile users have to load less data. On the other hand, the product handles caching that allows the application to save parts of the application to the device and reuse on the next load that uses local memory instead of the internet.

As initiation of the project, a data collection method that was conducted via online surveys. The research showed that 23.1% of the answer givers are reading on the state of the virus and estimates the chances of getting or spreading the virus by comparison data. The survey also had sections focusing on the distribution of reliable data collection and the difficulties they experience preparing for traveling. The research on the distribution of sources showed that 50% of the answers were pointing on the Government website and 16.7% on the destination country's website, whereas 33.3% were pointing on uncertain sources. On the other hand, the section that was focusing on the difficulties that travelers experience showed that 56% of the answer givers finds it difficult to gather all the documents they might need before travel, whereas 72% of them found it difficult to gather those documents that are specific to the current situation. From the data collected for each section a problem was defined: 1. "*People are collecting information from a wide range of sources and not all of them are reliable.*", 2. "People are experiencing difficulties collecting documents before traveling.", 3. "*People are aware, but they don't take action to estimate the probability of catching or spreading the virus.*" However there are multiple problem statements, the last problem statement defined (3.) is the focus of the project and the other statements helped to describe the features within the product. The final solution to the problem and the product of the project resulted in an application that aims to address the problems by providing features to be used to solve or to help solve the problem of the users. The goals were to provide base sources where the users can start search on the virus and requirements, facilitate people by allowing them to store documents within the application that is required for traveling that is easily accessible and eliminate the time restrictions described by the Government, and the focus of the project is to provide a platform that can be easily used to gather information on the destinations and departure country.

The document aims to provide insight into the methodology and the project management followed and focuses on the technical implementation of the features by reviewing sources, listing methodologies that need to be understood for the document and explains how these methodologies were

implemented into the project. The document emphasizes the methodological approach rather than following a strict timeline. However, the timeline is not strictly followed, each step described in the document is reflected on the methods of the project following Software Development Life Cycle as an overarching methodology and Waterfall methodology as a sub-method should represent the execution of steps in a consecutive manner. However, the project focuses on technical parts, user feedback was collected where it was critical to do so to manage the project accordingly.

2. Literature Review

The product of the project itself aims to help those people who need to travel during the current pandemic by providing features within an application that helps solving the problems people are experiencing. However, the product is an application, the subject of the project emphasizes the technical implementation and analysis of the approaches, methodologies, and using current best practices within the web development industry. To select appropriate methodologies and problem solving techniques a set of sources were overviewed in relation to the project that will be discussed in the next sections. The sections below are divided in a way that represents each area of the application that needs to be studied to deliver an industry standard production.

2.1 Background context

The project background relies on the current situation caused by Covid-19 known as the Coronavirus. The information of the background context was mainly sourced from the Government website. The website was used to gain information about the current limitations and restrictions in relation to traveling. The information given clarifies the requirements and steps the travelers need to make in order to be able to move. These steps include the acceptance of a 10 day self-isolation and completing the Passenger Locator Form. However, these steps are not causing any critical problem, but they also have restrictions defined by the Government. The Government website says: "*You can submit the form any time in the 48 hours before you arrive in the UK.*"(UK Government, 2021). The announcement also mentions the possibility to provide a negative test if necessary. The advantage of referring to the Government website was its updated and current information that is written as an announcement which excludes the possibility of opinion based expressions within the subject that was a reliable source to initiate the project.

2.2 Approach and methodologies

A previous subject at the University offered the opportunity to freely choose the subject of the project that is valid as long as the given technology is used. The subject was used to deliver an application that helps schools to reopen after the lockdowns. The project included methodologies and approaches that were implemented in this project as well. These methodologies were mainly used to create the design of the application interface and the architecture design (Appendix: A).

2.1.1 Overarching methodology

As an overarching methodology two different methods were considered to be used to manage the application development: Software Development Life Cycle and Design Thinking. Software Development Life Cycle is a methodology that focuses on the quality and time constraints. An article says: "*The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software*" (ALTVATER, APRIL 8, 2020). On the other hand, Design Thinking is a solution based methodology that defines the problem by understanding the target audience. This is particularly useful when the designers themselves are on the suffering side of a problem. According to an article on Interaction Design Foundation: "Design thinking is a non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test." (Dam, R. and Siang, T., 2021, para. 1) Both methods coincided with the development. However, as the development itself was not relying on real customers to validate every step of the process the 6 phase SDLC model was used. As described by Altvater, SDLC uses the next steps: *Requirement analysis, Planning, Architectural design, Software Development, Testing, and Deployment*. Another study defines the steps of SDLC as a 5 step method: "The activities can be broken down into a very detail level but at the same time they can be grouped into five (5) core categories: Plan, Design, Develop, Test and Deploy." (Tiky, 2016, p. 7). However, the two sources describe SDLC differently; According to the study of Winston W. Royce, "*There are two essential steps common to all computer program developments, regardless of size or*

complexity. There is first an analysis step, followed second by a coding step..." (Royce, 1970, p. 1). Thus the project uses the first, 6 step model that includes the research in the first place as the project required fundamental understanding of the requirements.

2.2.2 Sub-methods

The project uses the Waterfall methodology as a sub-method for managing the implementation. In contrast with SDLC the project implements the original Waterfall methodology proposed by Dr. Winston W. Royce in 1970 in his study "*MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS*". Royce in his study explains how Waterfall is implemented in software systems and overviews the structure of the methodology. Royce also pinpoints the risks using the methodology by defining a problem that says the testing phase is introduced at the late phases of the implementation that invites risks to the development. This project implements Waterfall methodology within a test-driven environment that is recommended by the SDLC methodology, to eliminate the risk defined above.

2.2.3 Design

The project includes the User Experience Design (UX). UX was based on the principles described in "*The Design of Everyday Things*" by Donald A. Norman and the "*Gestalt theory integrated into interactive media design*" by Graham and Lisa in Humanities & Social Sciences.

Donald A. Norman emphasizes the comfort of the users by limiting their cognitive thinking. The principles described in his book promotes the design of the User Experience in a way that reduces the cognitive thinking of users resulting in the ease of use of the product. The principles described are the following: *Visibility, Feedback, Constraints, Mapping, Consistency, Affordance*. The author, Donald A. Norman is known for his labour in Human-Digital Interaction which encourages the reliability of his principles described in his book. In contrast with Norman principles that relies on human cognitive abilities, the Gestalt principles relies on the visual perception of users. Gestalt principles promote the use of a set of rules to reduce the visual perception of a

user that avoids the visually overwhelming appearance. The Gestalt principles are: *Law of Closure, Law of Common Region, Figure/Ground, and Law of Proximity*. As the scope of the study focuses on the technical implementation the User Interface Design is implemented from the previous project mentioned under the “2.2 Approach and methodologies” section.

3. Methodology

Every project uses methodologies to ensure a certain requirement or requirements to be met. The methods can be used towards the product itself to match non-functional requirements or towards project management to increase team efficiency, performance, and keep the focus on the subject. In this project a set of methodologies were collected before development reflecting on the non-functional requirements and on the resources that can be used to deliver the product. The collected methods can be separated into two sections; Those methodologies that can be followed to deliver a User Experience and User Interface Design for the application before development, and those methodologies that describe a step or steps within the development life cycle. However, the aim of the project is an application as a whole, the project focuses on the implementation of the programming side of the project rather than design. Thus the design focuses on User Experience Design, whereas User Interface Design is integrated from the previous project, which was possible due the nature of the applications that both use Task-Centred Design.

The next section introduces the methodologies that are critical to understand to read this document. The methodologies are explained and further discussed why they are used, how they contribute to the project and what are the strengths and weaknesses from the viewpoint of the implementation process.

3.1 Development Methodology

Development methodologies are methodologies that focus on both the progress of the development by describing a step-to-step guide, that provides a coordinated approach for teams and to match non-functional requirements. As mentioned before, the methods were selected by reflecting on the requirements of the project, which includes the following: a strict narrow time interval to deliver the project, the project includes only one developer, the understanding and familiarity of a problem to be solved, the goal of the project is to produce an application that solves or at least help users to solve the problem, the application is mobile first. Considering the requirements above the following development methods were listed to manage the project with: Design Thinking, Software Development Life Cycle, Waterfall methodology, Agile development, and Rapid Prototyping. However, there are overlaps between the requirements that support each methodology, the fact needed to be considered that there are no customers or direct users to develop the application with, thus those methodologies were in advantage that requires less human interaction with customers.

3.1.1 Overarching methodology

Software Development Life Cycle (SDLC) is used as an overarching methodology. Software Development Life Cycle is a methodology that helps teams or individuals to conduct a coordinated development with clearly defined and minimum steps required. This increases efficiency and consumes the minimum time needed to release the project. An article describes SDLC as: "*SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible*"(ALTVATER, 2021, para. 3.). These qualities that SDLC provides reflects on the requirements and resources the project has to deal with. Software Development Life Cycle describes a 6 step development method that includes the following: requirement analysis, planning, architectural design, software development, testing, and deployment.

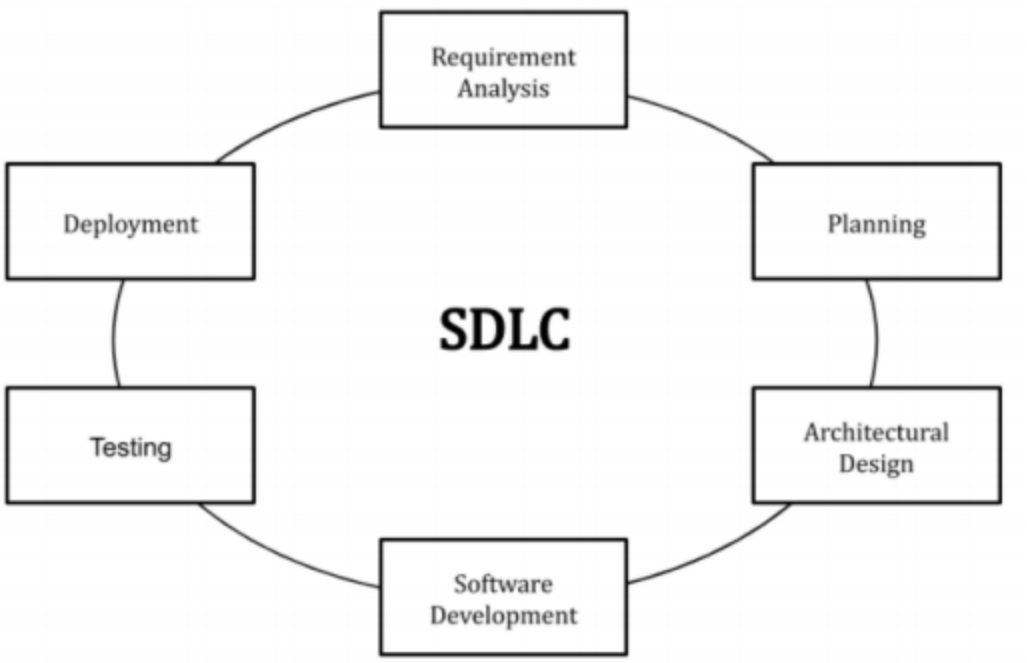


Figure 1, Software Development Life Cycle

However, this model is representative of SDLC, it focuses on the implementation of the application rather than design. To include all aspects of the software development it was modified to fit for the project. The 3rd step “Architectural Design” was used to implement User Experience and User Interface Design. Step 5, “Software Development” implements Waterfall as a sub-method that contains Architectural Design and a test-driven development approach that is kept from SDLC, which requires the developer to write tests before developing the functionality of the application that results in quality. The modification resulted in the model shown below:

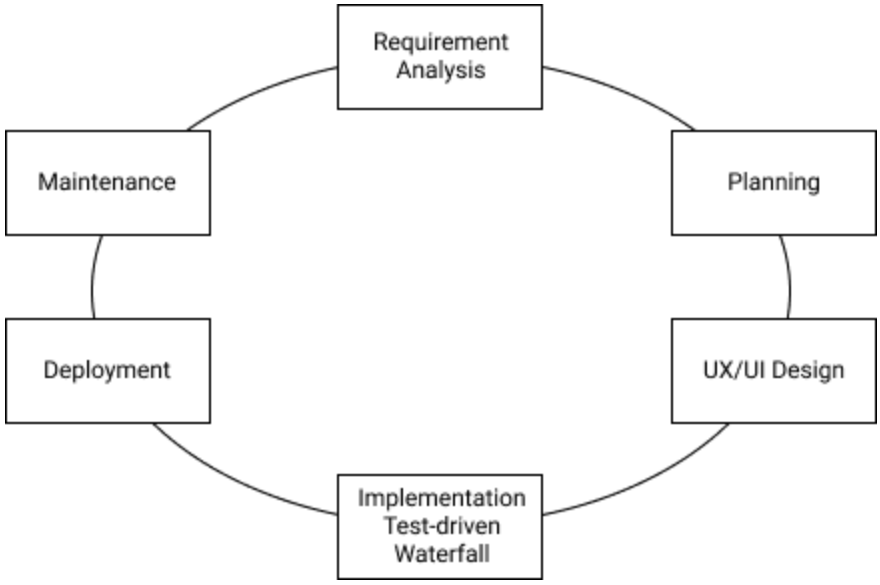


Figure 2, Modified Software Development Life Cycle Model

3.1.2 Phases in the modified SDLC model:

1. Requirement Analysis

During the first step of the development all the information is gathered that can help in the next step “Planning”, but the information is carried through the life cycle. This includes the following: conducting the initial data collection method, a survey to gather user input to define a problem statement, target audience, functional and non-functional requirements, and estimating resources.

2. Planning

Within planning the scope and boundary of concepts should be defined. A feasibility report or study in financial and technical areas should be conducted by the seniors or project owners. Planning also includes estimating how time and resources can be distributed according to the resources of the company or development team, deciding on management tools and setting up milestones for the project life time.

3. Design

Design often starts with a Software Requirements Specification Document (SRS), that should specify the functional, non-functional, and other product details that should be approved by the clients, users or customers before development. However, in a company the Designer team should compose a set of different designs, due to the resources this project includes one design that is re-defined according to user needs.

4. Implementation

Implementation starts immediately after an appropriate design is selected. The developer team should decide on development tools and stacks to be used during development, however, this is commonly determined by the company itself. Implementation also consists of choosing the right Architecture Design, design-patterns, and best practices to ensure quality and non-functional requirements such as scalability, maintainability, security, and flexibility. Functional Specification (FS), a document that describes all the functions provided within an application or project, is also often a responsibility of the development team.

5. Deployment

Deployment means the application's transition from development environment to a production environment. The first step of deployment is running a set of tests on the product software to confirm there are no unintended functionalities within the product and all the intended functions working as expected. These tests can be specified as Unit Tests, Integration Tests, End-to-End testing, and Load Testing. Placing a software into a production environment often requires developers to alter the software. These changes should be provided in a clearly defined Deployment Plan (DP) that should be approved by the management before deployment.

6. Maintenance

Maintenance starts with the evaluation of the product software, which should give a clear understanding of the product place and state. It also clarifies the recommendations of future development of the software.

3.1.3 Waterfall sub-method

Waterfall methodology is the earliest and most commonly used project management methodology that was introduced by Winston W. Royce in 1970. Waterfall describes a set of consecutive steps to deliver a product. These steps are clearly defined and minimal criteria to deliver a product. However, these steps are clear and short, Royce mentions in his study that "*I believe in this concept, but the implementation described above is risky and invites failure*"(Royce, 1970, para. 5.). He also highlights in his article that the problem originated from the fact that the model suggests testing at the end of the development cycle. In this project Waterfall was used as a sub-method to deliver the production ready software integrated into the Software Development Life Cycle. As sub-method Waterfall used from 2nd to 4th step as shown on Figure 3 below:

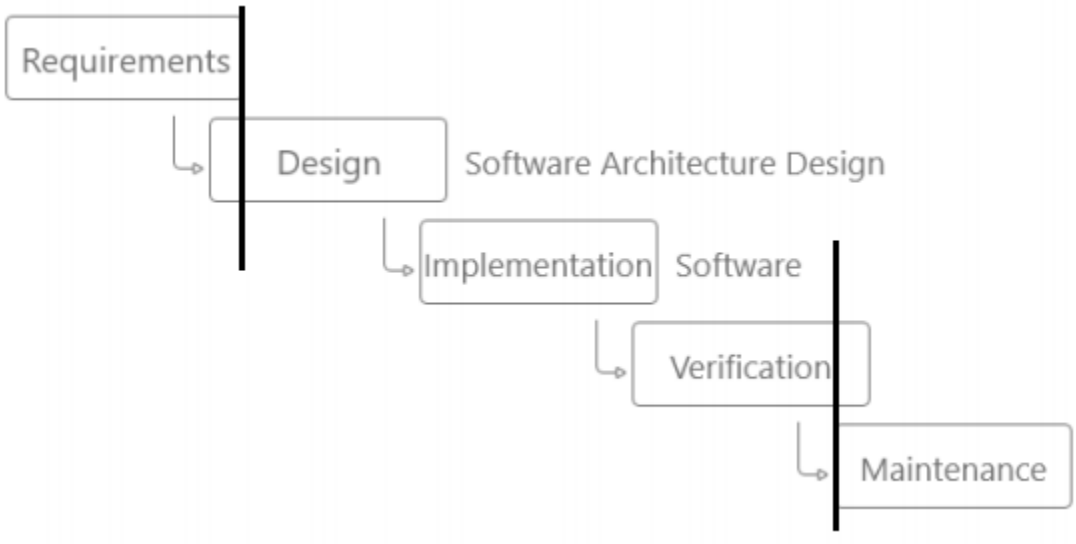


Figure 3, Waterfall Model

By using only the “Design”, “Implementation”, and “Verification” steps from Waterfall methodology integrated into the “Implementation” step of Software Development Life Cycle, which uses test-driven approach, the project meant to be organized in way that eliminates the risk factor of Waterfall methodology described by Winston W. Royce in his article, that could cause unintended behaviors within the software.

3.1.4 Phases in Waterfall Model

1. Design

The Waterfall model Design phase includes the Architecture Design, choosing the right design patterns, and best practices that ensure the non-functional requirements of the application, which includes: maintainability, scalability, flexibility, and security. On the other hand, the Design phase includes the design of the file system and communication between the server and the application and/or within the application that is represented with a Data Flow Diagram (DFD).

2. Implementation

Implementation results in the software product that is built with the chosen or provided tools and technology. The toolset and practices of the implementation phase should always strictly follow the defined patterns in provided documents from the Design phase to reflect and ensure non-functional requirements over the working product or software. The way of implementation always depends on how the application structure was defined and how it can be destructured to individual pieces that have a single purpose.

3. Verification

Verification is the phase in Waterfall that should present a working and tested software that needs to be approved by the management before deployment.

3.2 Design Methodology

Design methodologies are methodologies that are focusing on the part of project management that delivers the User Experience (UX) and User Interface Design (UI). However, each of these have their own methodologies, the scope of the project focuses on the implementation and the UI Design was integrated from the previous application, thus only UX methodologies will be described in the next sections.

3.2.1 User Experience Design

User Experience Design is a design process that's sole purpose is to create a system that delivers great experience for it's users. The purpose of UX Design is not limited to create a great and seamless experience. With UX Design, designers are able to customize the experience. Make customers comfortable or uncomfortable or deliver a design that leads the users and encourages them to make actions and interact with the product. In the next section a list of UX Design methods are shown that was consequently executed to customize the experience of the application.

3.2.2 User Experience Methods

1. User Requirement Analysis

User Requirements are the visual and contextual description of the product from the users perception, developed from User Need Analysis. The results are defined as visual elements and functions of the product.

2. Task-Centered Design

User-Centered Design and Task-Centered Design defines the nature of the design within the application. Task-Centered Design offers and highlights the elements and functions to the users within the application that are critical to use in order to achieve their goal.

3. Low Fidelity Application Mockup

Low Fidelity Mockup means the mockup of the application that uses the minimal resources to understand the user experience that the structure of the system designed.

4. Hierarchical Task Analysis

Hierarchical Task Analysis (HTA) is the analysis of the steps the user needs to take to achieve their goal. Referring to the nature of the application that is Task-Centered Design, the designers are looking for steps that can be merged or steps that can be removed from the user journey.

5. High Fidelity Wireframes

In contrast with Low Fidelity Wireframes, High Fidelity Wireframes are often provided within a chosen designer tool as it is used to deliver a set of prototypes of the design.

3.3 Evaluation methods

Each methodology listed above is a point for evaluation for the project as every step of the project development can be reflected on how and to what extent the methodology is implemented and how well they work together. These evaluations are passive, as they can be monitored after the development process by reflecting on the implementation. Another set of evaluation methods can be separated into 3 different groups: Those that are concerned with the technical implementation of the project, those that are concerned about the design implementation of the project, and those that are reflecting on the usability of the application to measure to what extent can the application help the users.

The implementation of the application will be reflected by a tool provided by Google Chrome developer tools, called “Lighthouse”. The Lighthouse tool measures the application based on four different aspects: Performance, Accessibility, Best Practices, and the compatibility to make the software Progressive Web Application. On the other hand, the design implementation will be reflected on the design principles described in Donald A. Norman’s “The Design of EveryDay Things'' book and by the Gestalt principles. Donald A. Norman’s principles are: Visibility, Feedback, Constraints, Mapping, Consistency, Affordance. The principles aim to limit the users cognitive thinking, this way the designers make users navigate comfortably within the application. Gestalt principles are: Figure/ground, Proximity, Closure, Similarity, and Continuation. In contrast with Norman’s principles the Gestalt principles are focusing on the limitation of visual perception, which emphasizes the importance to avoid the visually overwhelming appearance. Another and last evaluation method is gathering feedback from end users on the finished application. This should be conducted via a video call that is led by a pre-written script that is focused on the user satisfaction, the feel of use, and on the feedback on the usability and need of the application.

3.4 Professional, Legal and Ethical issues

In the initial phase of the project and Software Development Life Cycle, data was conducted to gather information on the problem that involved the opinion of human participants. The data collected was conducted via online surveys provided by Google. The survey contains a clarification before the questions about the project itself and the purpose of the survey and how the data will be used. Another data collection method was conducted via google surveys to gather feedback on the final product containing the same clarification. As the project uses these surveys to present data an ethical release was approved beforehand (Appendix: B).

3.5 Project Management

In the second phase of the Software Development Life Cycle, “Planning”, all the data, requirements and resources collected previously in the first step of Software Development Life Cycle, “Requirement Analysis” were considered to set up an estimated timeline for the development life cycle. The maximum development time was 400 hours, however the time allocated for each task was defined in weeks within the project. The reason behind the time allocation is given in weeks is the fact of lack of experience. Turning apart from hour based time allocation gives space to unexpected events and experiments that now can be solved with flexible time allocation. Each life cycle phase was allocated with 1 week time constraints except “Implementation”. As the project focuses on the implementation the allocated time was 3+1 week. The plus 1 week was held for technical or other difficulties that was expected to occur during development. Finally 2 weeks allocated for the documentation and one more week to finalize the product of the project. The allocated time makes a total of 12 weeks of development time.

Along with the time constraints a Logbook was initiated with the project, that contains the weekly meetings with the supervisor. This is an overview of the tasks on each week that are related to the project. It consists of relative sources, tasks, and recommendations by the tutor. Logbook is organized in the version control of Github. Another project management tool is Trello. Trello is an online detailed drag & drop list based management tool. As the Logbook is project timeline related, Trello was used to be only product related. It contains the breakdown of the application that was taken under development.

The project itself is managed within the Github version control system using branching strategy. Github helps developers store their work and every saved version of it by maintaining a version control logger specifically in the given folder it is initiated in. The logger helps keep track of the developer's documents and connects to an online repository where the documents can be saved separately from a local computer. The version control system allows using a branching strategy that helps separate the development of each

feature or phase of the product. In this project the branching strategy is reflecting the application development timeline shown below:

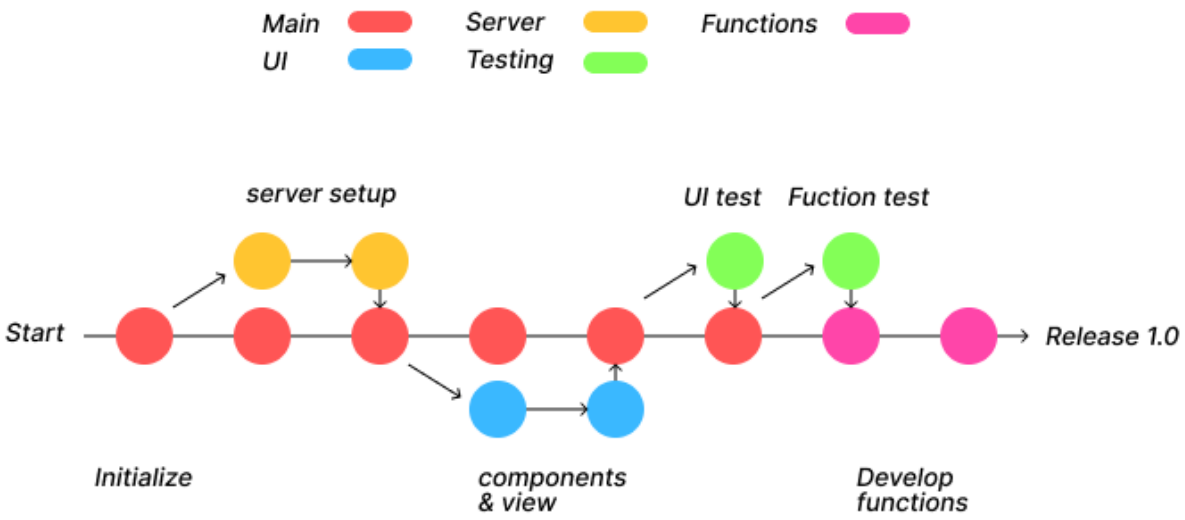


Figure 4, Github Branching Strategy

4. Design & implementation

The problems that were defined from the collected data are meant to be solved with an online web application. The purpose of the application is to save time and stress on people who need to travel by providing all the information and features they might additionally need at the country border. The purpose of the application suggests critical needs towards the application that was developed which focuses the nature of the application on mobile friendliness and optimization. To achieve mobile optimization the following criteria needed to be met; The application needs to be: perfectly fit for mobile screens, Server-side rendered to require less mobile internet and computing power, and a Progressive Web Application to cache data for faster loading that can be developed into an accessible application in offline mode. These three modern features of the application support the mobile friendliness nature that is implemented in the next sections.

The next section contains the implementation of the project. However, the focus of the project is the technical implementation, the process of Software Development Life Cycle includes “Design” that will be shortly explained. The description of the Design phase of the implementation is rather a step-to-step explanation of how the User Experience was designed. In contrast with the Design phase, the technical implementation, which is the rest of the document, is approach and explanatory based rather than a step-to-step guide. The focus of the document emphasizes how the occurring problems were solved during technical development in an explanatory manner to help understand “Why” and “How” were given techniques or tools used to overcome difficulties. However, the document remains as life-like as possible and tries to provide the best understanding of tool usage and approaches; it does not strictly follow the timeline of implementation in detail. Instead of timelines the previously outlined management methodologies will be used as a guideline for the discussion that can be reflected on time management.

4.1 SDLC Phase 1, Requirement Analysis

The first phase of the Software Development Life Cycle focused on three different aspects of the project. The first was the data collection to define the problem or problems the target audience is experiencing. Initially the data collection that was conducted via online surveys was supposed to help define a single problem statement that needs to be solved, however, the survey was designed to understand more aspects of the state. Thus it helped to understand the context of the occurring problem that led to other problems that are not critical yet contributes to the original statement. The problem statement is that “people do not estimate the chances of catching or spreading the virus by traveling”. Other problem statements that are treated as sub-problems that contribute to the original problem are “people having hard times collecting documents before travel” and “people collecting data from unreliable sources on the virus”. The solve or at least help to solve these problems the product built on the following statements: present visible and diagram on the virus, additional documents should be available within the product, and provide reliable news on the virus and travel possibilities.l

The second focus point of the phase was to search for examples of existing applications that are related to the virus. The application NHSCOVID-19 provided by the National Health Service was taken under investigation on what features they offer within the application. The main feature of the application is to track users based on location and state and offer guidance for those with symptoms. Another feature is the “Read latest advice” that takes to the government website providing reliable information that is similar to the initial news feed feature proposal in this project. However, this feature always takes the user on the same website, this project implements the feature by providing reliable sources, but it can be extended by the user when they want to save sources they find useful. As a whole this feature provides a list of initial sources that leads to reliable information, but new list elements can be added.

The last point of analysis was to generate requirements towards the product from the collected information that help the next steps of the development. The requirements are reflecting on the nature of the application, target audience, non-functional requirements, and functional requirements. The first source is the founding of data collection which describes three features that can be listed as functional requirements: *a diagram on the virus*, *a form to provide document data*, *a display surface for the documents*, *a list of customizable links to sources on the virus*. The second source is the reflection of mobile friendliness. Mobiles frequently have weak internet connection, very limited computing powers and each device provides different screen sizes. These requirements are implying non-functional requirements such as: *mobile friendliness*, *implementation of server-side rendering*, *and the development into a Progressive Web Application*. The outcome of the phase is a list of requirements that helped to describe the application and its features and a list of sources that can be useful during the next step of Software Development Life Cycle: Planning.

4.2 SDLC Phase 2, Planning

Planning started with the measurements on the project considering the resources and sorting appropriate sub-methodologies and design patterns. First the scope of the project was determined by overviewing the purpose of the application which is to help people solve a certain problem that was defined in the first phase of Software Development Life Cycle, along with the provided non-functional and functional requirements. It is important to mention this document is not equal to the Software Requirement Specification Document (SRS) that will be provided after the third phase of SDLC, however, the purpose is similar; provide a clear understanding of the requirements. The most important requirement was the mobile friendliness, Server-Side Rendering (SSR), and developing the application into a Progressive Web Application that needed to be considered. On the other hand, the outline of the application is: a form, a display for the form or user data, a diagram on the virus data, and an expandable information feed. To measure the application by the means of scope, the list of requirements were translated into a Data Flow Diagram (DFD). Each requirement listed above points to an aspect of the application. Server-Side Rendering requires runtime processing, that means the application needs to be run on a platform that provides a runtime environment. Another point is the display of the diagram on the virus, that suggests the application requires an external source for Covid-19 data. The last point was that the form provided to provide travel information and expand the information feed, requires a database in the background to save the information. The produced Data Flow Diagram can be seen below:

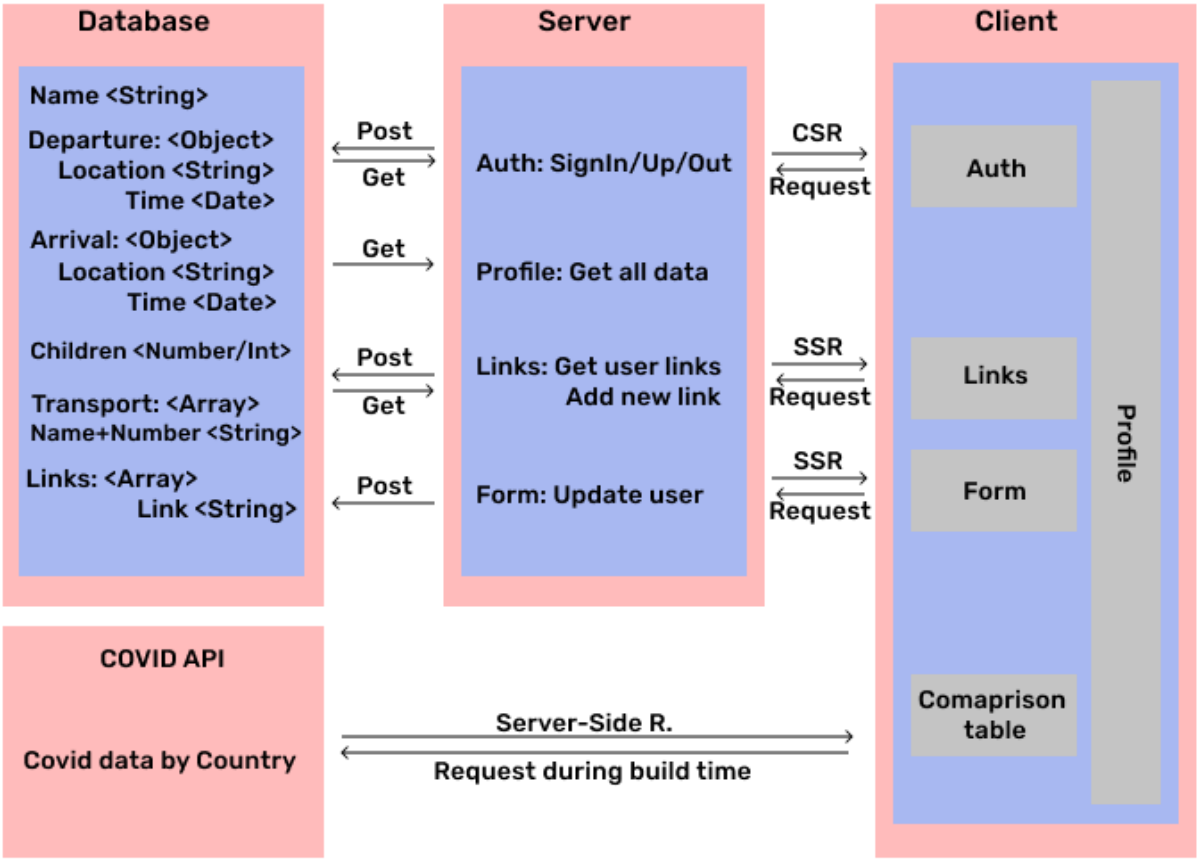


Figure 5, Data Flow Diagram

Another segment of the planning phase was to set up time constraints. Now that the project had a Data Flow Diagram, it could be destructured into individual parts. By breaking the application into its individual parts, each part represents a milestone for the initial application architecture in the outer scope. The outer scope milestones were the following consecutive steps: Database Design, Server initialization, and Client initialization. However, this was only the outer scope of the application, the inner scope of the application contained other goals on the client-side, such as: creating libraries, building User Interface, writing tests on UI and providing pre-written tests for functionality, the last goal is writing the functionality.

The clarification on the scope of the project gave a clear understanding on the areas of the application in terms of front and back-end. DFD was outlined on the base concepts that the application needed. It is important to mention that

this phase contains the production of a feasibility report. This study is based on a previously written feasibility report that measures and proposes a plan for the project, as well as seeks and reviews the sources to be used throughout the implementation. The short document also compares and discusses methodologies to be used.

4.2.1 Methodologies

The planning also involved the consideration of further methodologies that were used in the “Implementation” phase of SDLC. On the other hand, as the project does not rely on real customers, rather data collection is conducted on the critical milestones of the development, Waterfall sub-methodology was selected to manage the implementation. During planning Agile development and Design Sprints were also considered along with Design Thinking as overarching methodology, however, these methodologies require a closer and open contact with real customers or users due to their customer based and iterative approach.

4.2.2 Tools

For the technical implementation of the project a set of development tools were proposed based on the requirements listed in the documentation of the first phase of SDLC. Reflecting on those requirements Next.js was chosen to be used as a front-end framework as it offers a built-in solution for server-side rendering. The application also takes advantage of dynamic rendering as not all pages are server-side rendered. Another reason for Next.js is that it focuses on interface building by defining components, implementing a simple “View Model” in contrast with Angular that implements a “Model-View-ViewModel”.

The first point was to keep the application as lightweight as possible due to the mobile friendly approach. For this purpose a Cascading Style Sheet (CSS) library, Sassy Cascading Style Sheet (SCSS). SCSS has the flexibility to include variables anywhere in the file and due to its nested nature component attributes cannot affect each other. The library based approach is due to the fact that once the library is written, it does not matter how the application scale, the components from the libraries will not be extended, just reused. As a result the amount of SCSS the phones and personal computers need to download remains the same size.

For the server-side of the application Node.js and Express.js were selected. Node.js can produce reusable code that leads to faster development, easy to understand as it is a runtime environment for JavaScript that is being used on the front-end as well and offers compatibility with multi-threading due to asynchronous functions over PHP. Express.js is a library for Node.js that is focused on creating servers for web applications. The last part is MongoDB. MongoDB is used for its document based database. As the application is meant to be lightweight, a document based database was selected for the project due to its real time updates and data requests.

4.3 SDLC Phase 3, Design

The Design phase started with User Requirements Analysis (URA). URA provides the visual description of the application from the perception of the users. URA was used to describe visual elements in the application, and based on the purpose of what they want to achieve the functionality of the elements are given. The functionality was used to produce the Software Requirement Specification Document (FRS), which was provided as a list of elements and their purpose at the end of the Design phase.

At this stage a Low Fidelity Wireframe was constructed based on a design method that was concluded from the analysis of existing applications related to the virus. The application NHSCOVID-19. The application is providing everything that the user might need to use in order to achieve their goals. On the other hand, it does not provide anything that is related to entertainment and does not encourage the users to spend time within the application. The application uses Task-Centred Design (TCD), which was implemented in this project.

The last step was the execution of Hierarchical Task Analysis (HTA) that is the overview of the steps a user needs to make in order to achieve their goals. The initial flow fidelity wireframe separated the application into 5 different views: home, form, diagram on the virus, information list, and profile. Conducting HTA following the focus of TCD the design resulted in 3 views as the “diagram on the virus” is merged into the Home screen and profile is merged into the Header that made it an openable field on any view. Lastly the feedback on the High Fidelity Wireframe design resulted in the replacement of “diagram on the virus” with a “comparison table” and the prototype was delivered using a

neomorphism UI kit provided by Figma community (Appendix: E).

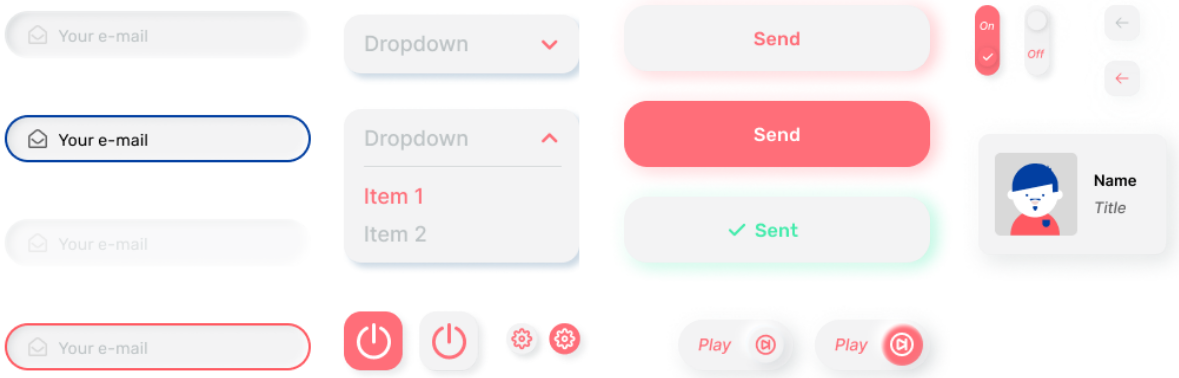


Figure 6, UI kit

4.4 SDLC Phase 4, Implementation

The technical implementation of the application followed Waterfall methodology, however, the project included only 3 steps of Waterfall as: Design, Implementation, and Verification was merged with the feedback gathering method. It was possible to follow only these 3 steps as the context of the project is provided by the overarching methodology, SDLC.

The next sections will present the project in an approach based manner. However, the approach based presentation follows the timeline loosely, it still reflects on the real project timeline and offers a better view of how the occurring problems were solved.

4.4.1 Design

The first phase of Waterfall methodology was the Design of the application. The Design refers to the Architecture Design , file system design, and design patterns that were used during development. To create the Architectural Design for the application a collection of devices and software components needed to be considered. This step was reflected on the Data Flow Diagram (DFD) proposed in the second step of SDLC, Planning. The difference is that the diagram does not include the mobile devices, just the client of the project. The DFD can be presented from the Architectural Design's view as the following:

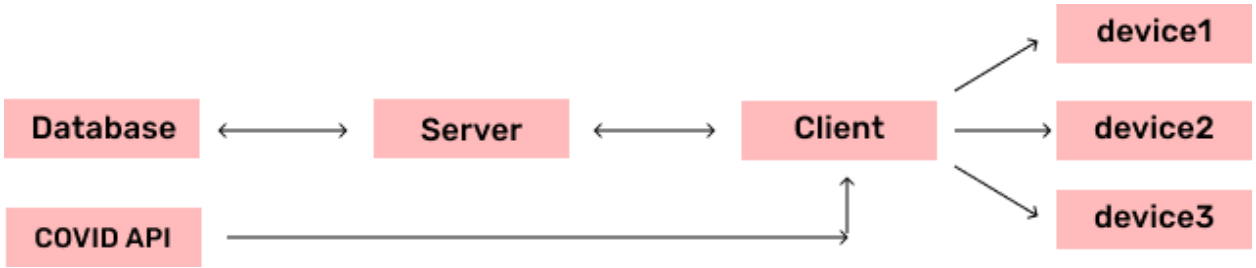


Figure 7, Monolith Architecture Design

The Model above meets the criteria of a Monolith Architectural Design, which was implemented in the project. The application is single threaded, however, JavaScript is capable of mimicking multithreading by providing asynchronous functionality, the parts, features and aspects of the application depend on each other, even if they are loosely coupled. An article says: "*Despite having different components/modules/services, the application is built and deployed as one Application for app platforms...*" (Haq, May 2, 2018). The understanding of the previous citation is critical as it is the base of a Monolith Architectural Design that reflects on the delivery method of this project. The application is developed as a whole, deployed into a production environment.

Another step of design was to decide on the design pattern the Monolith Architecture will be built with. Monolith Architecture commonly represented with a layered model. The "Model-View-Controller" (MVC) model is the most common, however, modern front-end frameworks are provides the implementation of a model such as Angular with "Model-View-ViewModel" (MVVM) and React with "ViewModel" (VM). Thus the chosen development tools and techniques containing the front-end framework already provided the "View Model". React.js implements "View Model" for building interfaces that is developed further into Next.js that was used for the project for extended functionality (Appendix: C).

The last step of preparing for the technical implementation was to design the file system the application was written on. The project uses the default file system provided by Next.js, that is reflected on the routes of the pages. Next.js generates the URL of the application by reading the access path to each page under the "pages" folder. The design has a sign in, signup, home, form, and links page, that should be accessed on the following routes respectively: "/signin", "/signup", "/" ,"/form", and "/links". Under the pages directory the components folder is also provided as a route that is ignored by the router. This route is used to store files that describe components of the website but not entire pages of the application. The project also uses the "lib" folder in the root directory to store functional components that are loosely coupled to the display. The root directory also contains; the public folder to serve static

assets such as pictures, styles folder that contain style libraries and component styles, custom server folder to serve the client from the database, and all the configuration files. The initial file system can be seen below on Figure 8 and Figure 9 that is the completion of Figure8:

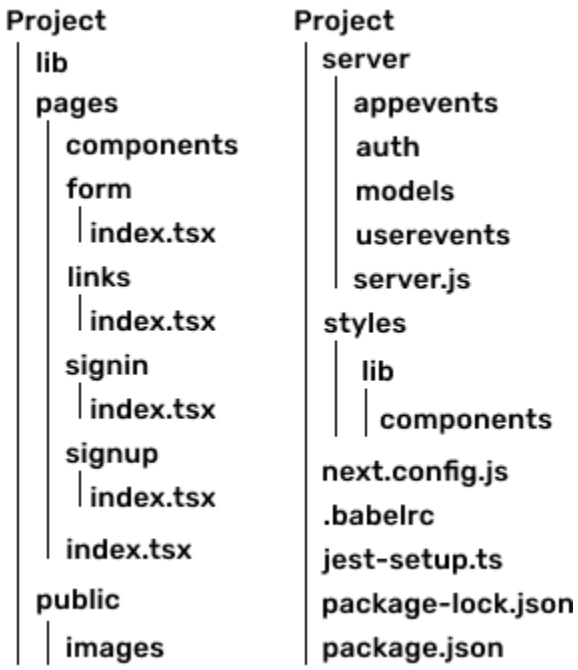


Figure 8, File system (left) and Figure 9, File system (right)

4.4.2 Implementation

The implementation of the application can be separated into 4 different sections that were executed consecutively to build the application. This included the database design and initial server setup, creation of style libraries based on the provided assets from the design with the skeleton of UI components based on frequently used components in the prototype along with the construction of the views or pages, the connection of the server and client, testing, and writing the functionality of the web application. However, the implementation required the handling of configurations and the consideration of other environmental aspects of the application; it will not be

discussed in detail as it might contain sensitive data such as API routes, keys and identifiers.

4.4.2.1 Database Design & Initial Server Setup

The first step was the database design. For storing data the project uses a non structured database MongoDB. The user object contains the data for the user that is the name, email,and an encrypted password. Normally the travel data would be separated into another object but as the application does not provide any social interactions and the client requires a single source for usage, all the data that is required is stored under the user object so the user object also contains information about the travel as: departure time and location, arrival time and location, transport number and name, if they are traveling with children, and the collected sources they found useful. Another document in the database is the session. Session objects contain the userID, the date the user logged in as creation date, the validation that can be true or false, and a termination date that represents the date when the user logged out (Appendix: D).

On the other hand, the initial server setup started with the installation of node and express.js. The server was created under the server folder in the root directory and uses a Node Manager Package called concurrently, to run the client and server in the same terminal. The server was created initially to use express-session and the database objects as a logger to keep track of traffic, but express-session was not working with the express server served by Next. By serving the application from the back-end via Next package made express-session to be non persistent. It lost data on every page rerender that was made to be server-side rendered. The session was moved back to a custom session using the database session objects.

Once the core of the server was working Next.js was configured to redirect every request to port 8080 while the client was running on port 3000 in next.config.js. As the client could communicate to the server, the server routes were developed based on the client actions. The first step was the

implementation of the authentication system that included the sign in, signup, session, and sign out. To create these functions first the MongoDB models needed to be created, as UserSession and User, which are the representative objects of the real database objects. Then to create the authentication these objects were used to create database objects with the given credentials or to request data from the database based on the representative objects to compare data to gain access to the application. Another function is the “verify” under the “auth” route that was implemented to check the user session before accessing or triggering trusted events.

The actions within the application can be trusted or untrusted events. Trusted events are those events that are triggered by the user and are implemented into the server/userevents accordingly. These events are the update of locator form and the saving of sources or links. On the other hand, untrusted events that are triggered by the environment are implemented in the server/appevents folder. These events are the load of user data triggered by page load. At this stage the initial server setup was tested using Advanced REST API to mimic requests from the client.

4.4.2.2 Style libraries, style components and UI construction

Building the User Interface started with the creation of libraries based on the assets provided by the design. Sassy Cascading Style Sheets (SCSS) provided the flexibility of global scope variable usage along with mixins. In the font library mixins are representing the h1, h2, and p tags within HTML separated into two sections based on the screen size; mobile or desktop. Another library Palette is a global scope library containing only named variables and values as color codes. The last library that was created is a media library that determines how the objects should be displayed on the defined screen sizes.

As the libraries were built the component creation started. The component creation was the creation of a mix of TSX (HTML in JavaScript/TypeScript) and SCSS components that are frequently used across the application. These components are designed to be loosely coupled. A style component uses

default values in mixins when a property is not provided during usage that allows partial flexibility on the included properties. A set of components can be seen below:

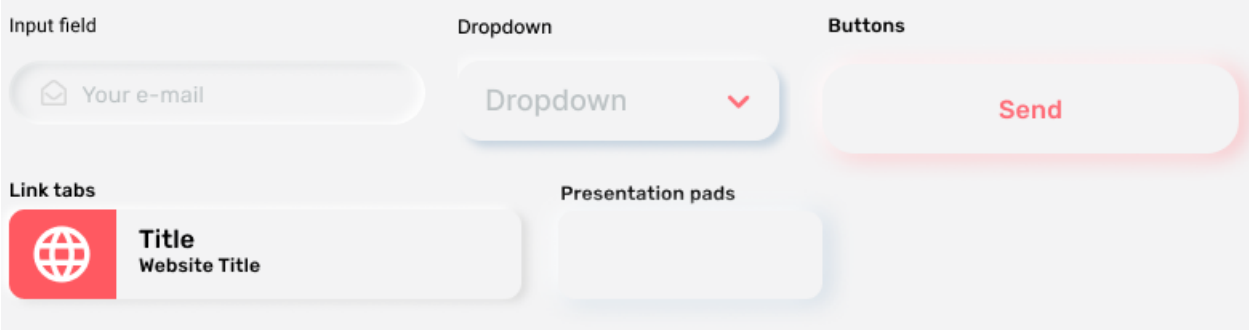


Figure 10, Frequently used components

Once the components were created the construction of UI had begun. The construction of UI meant to build up views within the application using components. The components were imported to the views and customized using the libraries during development..

4.4.2.3 Test driven development

The project followed the test driven development approach of Software Development Life Cycle to eliminate the risk factor of Waterfall methodology that was the late invitation of testing in the development process. Before connecting the Server and the Client with the existing User Interface, Jest testing library was installed to the project. The first step was the testing of the UI. The components on the page should appear, be visible with the allocated styling, and appear the same way on each load or render. Using the react-test-renderer NPM package, a sample of the UI render was printed into a file that was used as comparison on each test iteration. The tests were written to all pages within the application. However, the focus of the testing should be written to mimic the functionality of the application, due to the strict and narrow assessment deadline the project does not provide a full scale unit, integration, and end-to-end testing for the application.

4.4.2.4 Functionality and connection to the server

The functionality of the application added to the index.tsx files in the application. As each index.tsx file represents a page of the application, these files are used as the data layer for the application and all the data is represented by the components created earlier. This approach preserved the separation of the data layer from the presentation layer approach of the development structure.

The first step was the implementation of the authentication system. This required a data collection method through a form. Each page, sign in and signup uses the same approach, a dynamic data collection function that detects which field the user is typing in and fills out a formData object accordingly. Once the form is submitted the client makes a request to the server to create a new user and a session upon signup, or compare the data with those in the database. If the response is successful, the server responds with the newly created session object's id from the database. Then the session ID is then saved into the cookies. As the application uses server-side rendering, cookie storage is the only storage that could be accessed from the server before rendering through the request method using the built-in server-side rendering method of Next.js, in "getServersideProps" function. This approach allows the home page, form page, and links page to access the session ID to verify the user on each page transition, and retrieve the user's data to populate the page. If the user data is not present, the client is redirected to the signin page, or if the redirect fails, a link is shown to the signin page.

The features within the application are the comparison table, the locator form to be submitted, the useful resources list that can be expanded and the profile page to display user data. Comparison table uses a simple API request in the server-side render function to populate the table. The response, covid data per country is available in the destructured variable that the page components accept. Then a map function and a sort function is used to list all countries in the dropdown lists. If a country is selected, it filters out the country from the

object and saves it into a local variable by a function that listens to the changes on the dropdown selector.

The locator form is a form that contains input fields and dropdown menus. The form uses the same dynamic data collection method that was introduced in the signin and signup pages. The dropdowns contain the date input for the form. These dropdowns implement a time based feature that limit users to be able to choose only those dates that are not expired for departure. On the other hand, arrival dates are limited only to those days that come after the departure date. Unfortunately these functions are still breakable from the front-end, however, the server sorts out every input. Every input is checked if they are not null, empty, or undefined and sends feedback accordingly, that is displayed on the send button. The server also prevents the user from sending an arrival date that is earlier than their departure date.

The links page implements a simple data request to the server that retrieves all the information that can be found under the links array in the user object in the database. These links are then displayed on the screen by a map function that returns each link data into a link style component. Another feature of this page is another form that can be used to expand the source list. The form is initially hidden and can be opened by an animation that uses a farmer-motion animation library. The animation library was implemented into the profile tab as well to create the transition on the X axis to appear when opened. The profile page is fed with user data by the main pages and is included in the navigation bar.

The navigation bar implements two buttons. A button for multipurpose and the profile button that triggers the animation to open the profile tab. The multipurpose button's main functionality is a "back" page transition from the front page or links page. However, if the user has its profile open, it closes the profile instead. If the user is already on the home page, the sign on the button changes from a back arrow to an "X" sign and is going to signout from the application by adding information to the stored session object. The session object will contain the termination date, and overwrite the "isValid" attribute

to false and the session ID is not valid anymore. Then the application needed to be built by the script provided by Next.js. See the output below:

```
> [PWA] Compile client (static)
> [PWA] Auto register service worker with: D:\University\Year3\Dissertation_Project\App\trav
> [PWA] Service worker: D:\University\Year3\Dissertation_Project\App\travelapp\public\sw.js
> [PWA]   url: /sw.js
> [PWA]   scope: /
> [PWA] Compile server
info  - Using external babel configuration from D:\University\Year3\Dissertation_Project\App
info  - Creating an optimized production build
info  - Compiled successfully
info  - Collecting page data
info  - Generating static pages (11/11)
info  - Finalizing page optimization

Page                                                      Size     First Load JS
┌ λ /                                                     2.59 kB         104 kB
├   └ css/10f51d15ae9e84adf308.css                        1.42 kB
├   /_app                                                 0 B            68.9 kB
├ o /404                                                 3.46 kB         72.4 kB
├ o /components/button                                   299 B           69.2 kB
├ o /components/dropdown                                 370 B           69.3 kB
├ o /components/inputfield                               487 B           69.4 kB
├ o /components/link                                     1.75 kB         70.7 kB
├ o /components/navbar                                   169 B            101 kB
├   └ css/1ee41f6e8028acb96cb2.css                        814 B
├ o /components/prespad                                  372 B           69.3 kB
├ o /fallback                                            393 B           69.3 kB
├ λ /form                                                3.2 kB           104 kB
├   └ css/aa022734b68764077ad8.css                        1.37 kB
├ λ /links                                               4.09 kB          105 kB
├   └ css/1014565212f73b0eae28.css                        1.45 kB
├ o /signin                                              1.32 kB         75.1 kB
└ o /signup                                              1.34 kB         75.1 kB
+ First Load JS shared by all                            68.9 kB
  ├ chunks/be342905a48d05806873b228b0053776b12e6c36.1c301c.js  17 kB
  ├ chunks/framework.e3de07.js                           41.8 kB
  ├ chunks/main.48dc4d.js                                8.79 kB
  ├ chunks/pages/_app.78a6a8.js                          661 B
  ├ chunks/webpack.50bee0.js                             751 B
  └ css/15e19ea0cac43cd1cba7.css                         344 B

λ  (Server)  server-side renders at runtime (uses getInitialProps or getServerSideProps)
o  (Static)  automatically rendered as static HTML (uses no initial props)
●  (SSG)     automatically generated as static HTML + JSON (uses getStaticProps)
   (ISR)     incremental static regeneration (uses revalidate in getStaticProps)
```

Figure 11, Output of application building

The figure X above shows the output of the application build. The report shows that service-worker is successfully registered and the application is turned into an initial Progressive Web Application. On the other hand, the built application uses dynamic rendering that determines the form of rendering from page-to-page within the application. As the figure X shows the access panels such as signin and signup pages are client-side rendered, whereas the other views within the application are server-side rendered as they are pre-populated with user and API data.

4.5 SDLC Phase 5, Deployment

Deployment started with the overview of the application paying attention to endpoints and routes. As not all the routes were configured to detect development and production environments. The deployment plan consisted of the extension of the routes to be able to detect development and production environments and the server source of the client is rebased to the server instead of running the client and server separately by webpack and node and to move all the sensitive data that grant access or details about data sources into into Heroku's environment variables. Then the project was saved into the Github repository that was connected to Heroku to install dependencies, build the application, remove the unused files and compress the new build, then launch the application (Appendix: F).

4.6 SDLC Phase 6, Maintenance

The maintenance started with the testing of the application in production environment which failed for the first time as the application was first published with express-session which resulted in open sessions. Once a user signed in, the next user was automatically logged in to the previous user's session even without a token. The problems were solved by moving back to the earlier custom session system.

5. Results

The results were concluded from the survey about the final product that provided data on multiple aspects of the application such as the usability, experience design, and the user's opinion on how useful each feature is. The next was the usage of Google Chrome’s Lighthouse to generate feedback on the application. On the other hand, there were sources introduced within the document that are now used to evaluate the design from both cognitive and visual perception using Gestalt and Norman principles.

The participants in the survey had the opportunity to use the application as much as they wished before reflecting on the experience in the survey as they already received the published application. The surveys focused on 3 different aspects of the application to gather data on. The first section was the technical implementation from the end of usage. According to the feedback the users accessed the application from mobile and desktop devices and 100% stated that the application is clearly displayed and works as expected. However, 1 out of 5 had trouble using any feature within the application (Appendix: G). The second section was focusing on User Experience Design. Most users had no trouble understanding the interface and the actions required to use the application, however, the feedback within the application represented the same data to those who had no trouble using the application (Appendix: G). The last section focused on the usability of the application and tried to measure the success of the application. According to the user feedback the application found to be useful, however, with different success rates between each feature. The main feature, the comparison table, was found to be useful by all users, however, the listing side feature was found to be useful only by 40% of the users, whereas the integrated passenger locator form is 80% success rate again (Appendix: G).

Google Chrome’s Lighthouse was used to generate feedback on the application to reflect on the technical implementation of the application. The feedback generated can be separated into device usage. As the application targets

mobile applications first, but also desktop friendly, both results are presented below:

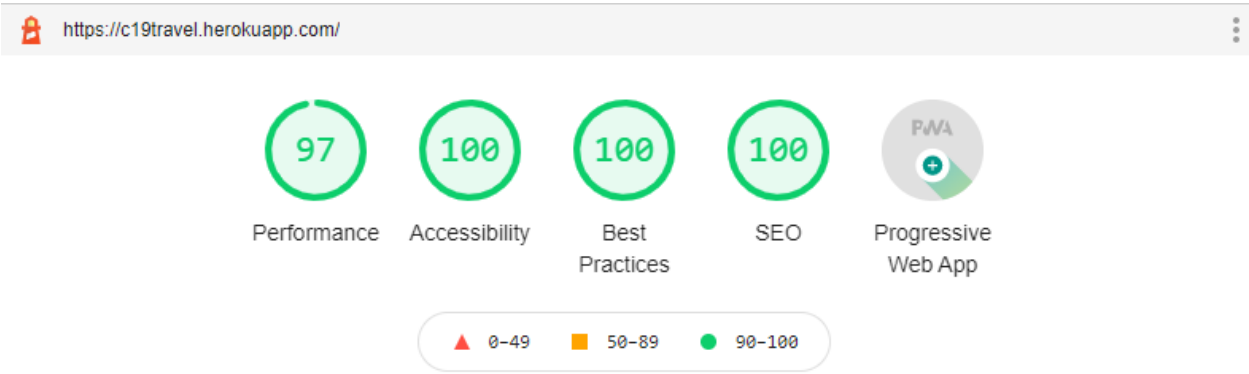


Figure 12, Lighthouse results for desktop

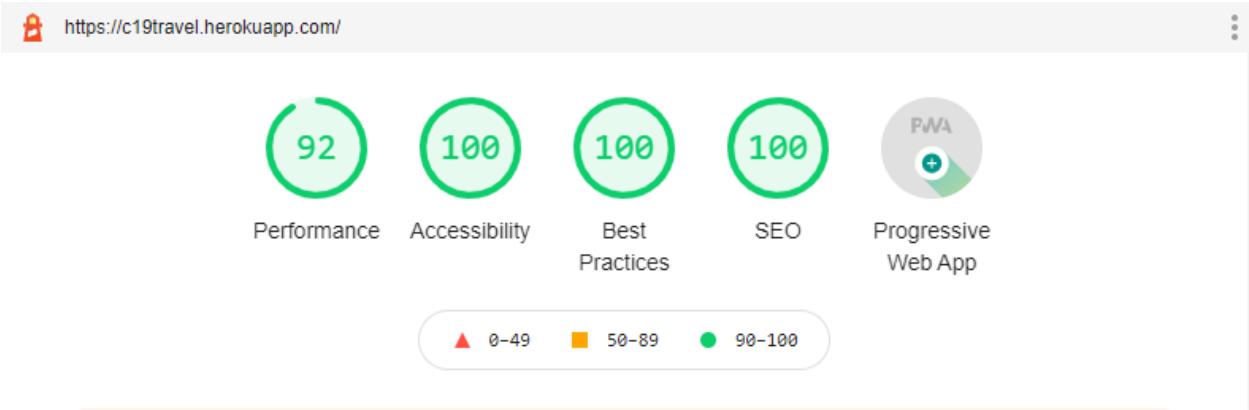


Figure 13, Lighthouse results for mobile

The results presented are the best out of 3 tests on each device. The worst scenario was 3 points lower on each devices' performance. The computing power of desktop versus a mobile device is a big difference, however, thanks to server-side rendering and caching, the application operates almost with the same performance on mobiles as on desktop computers. One problem that is presented by Lighthouse is the opportunity to use HTTP/2 connection for better performance.

The last point was the overview of the released application reflecting on the Gestalt and Norman principles. This was conducted via a checklist where all

the principles were listed and ticked, if every part of the application satisfies a certain rule. See the result below:

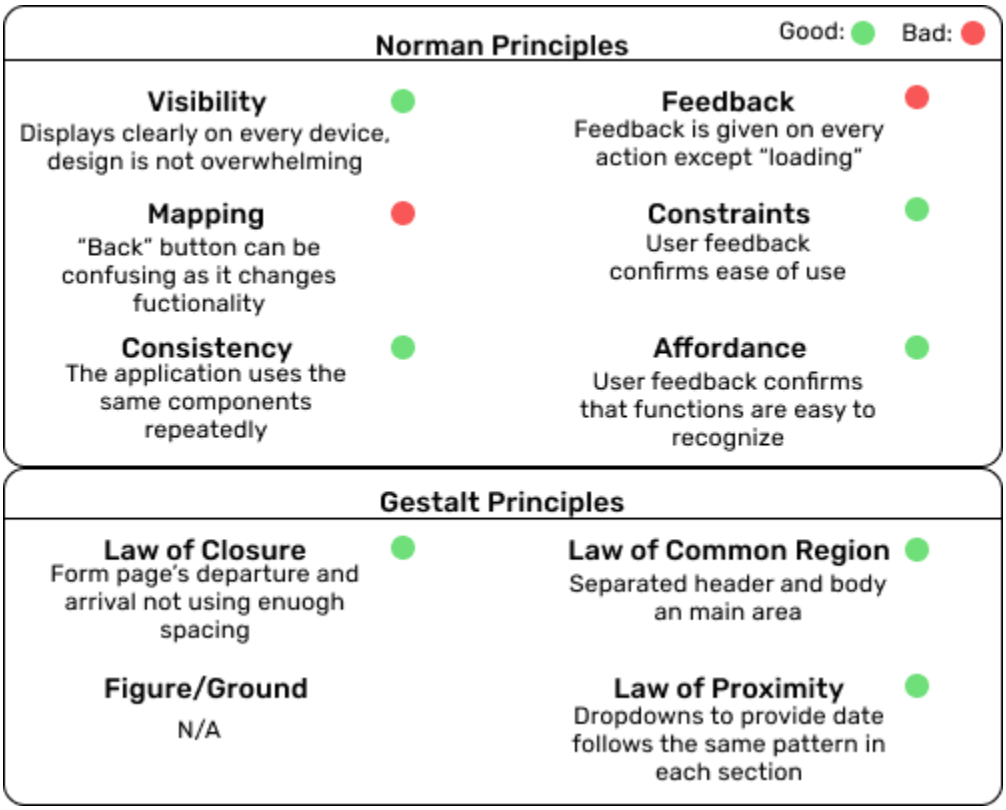


Figure 14, Principle checklist

.

6. Conclusions

The aim of the project was the production of an application that offers features that help users solve the defined problems that were created based on personal experience and with the help of an online survey to confirm the context of the problem. The project defined a critical problem as: *People are aware, but they don't take action to estimate the probability of catching or spreading the virus.* The product of the project addresses this problem by simply showing an integrated comparison table right on the landing page, that allows users to compare the state of the virus between the departure country and the arrival country. This has shown great results in the user feedback on the final product, which implies the feature of the application addresses the problem well. However, the product tackles the critical problem, there were other two (2) other problems defined at the start of the project. The two problem definitions were: *People are collecting information from a wide range of sources and not all of them are reliable*, *People are experiencing difficulties collecting documents before traveling.* The project addresses these problems with the implementation of side features, however, the success of the features measured by the user feedback on the product shows different results. The results show that the idea of the features could be useful, however, the implemented module has a lower rating of success.

The challenge of the project was the implementation of a set of methodologies that were used to deliver the application. The project uses Software Development Life Cycle (SDLC) as an overarching methodology, which clearly defines a set of steps that is necessary and minimal to deliver the application. The initial SDLC model needed to be modified to fit for the project as the model presented was not focusing on Design rather than implementation. The modification clearly resulted in the advantage of a more structured approach that invited every step of the development process focusing on the implementation. The modification represented disadvantages at the early stages due the fact that it was time-consuming to re-model and advantage at the full scale of the project as it represented a followable model without shortcomings. The advantage of SDLC over Design Thinking, was the linear

approach which deals easier with time constraints that allowed a fluent planning phase with straight ideas and the consideration of resources. Another advantage was the combination of SDLC with Waterfall methodology that would result in the elimination of the risks of Waterfall method, which was presented by the late invitation of testing in the development lifetime. Due to the narrow deadline, the test-driven approach suggested by the SDLC model was conducted only partially that tested only the User Interface of the application, which leaves the opportunity of remaining bugs to cause unintended behavior of the application. However, the application was tested before and after the deployment into the production environment, a test-driven approach would represent a critical part of the development that was left out.

Another part of the project was the implementation of new technologies and the assurance of non-functional requirements. This was ensured by the lightweight style and component library, the server-side rendered pages, and the caching provided by the fundamentals of a progressive web application. The style library made the application scalable and flexible without the need of mobile devices to load heavy style bundles that resulted in fast loading times. Another point was the server-side rendered pages that reduced the required computing power to load the web application, which resulted in the narrow difference between the performance of the application on desktop and on mobile devices. The progressive web application and the implementation of service-workers allows users to add the application to their home screen and the caching of UI components and application routes that also contributes to the performance.

The project itself presented a success in terms of the critical tasks and requirements, however, non-critical features should be developed into more usable functions. Please note that the application requires further development.

7. Recommendations

Based on the evaluation that is reflected on the implementation and proper usage of methodologies some of the aspects of the application were highlighted to be made wrong or not with scalability in mind. Tus the first recommendation is before further development the application should pass on a preparing phase that would revise those points of the project and would prepare the application for scalability, such as the reconstruction of the database objects. Although the main functionality and purpose of the application are proven to be useful based on user feedback, not every aspect of the application tends to show the same results. These features like the expandable source list could be developed further into a news feed to allocate a more effective purpose to the feature. However, the application is downloadable, and currently cannot be used fully offline. A further development of the application should implement the offline usage of Progressive Web Application and push notifications that could be used with the news feed. The last functionality, the passenger locator firm, is currently very limited in terms of usability. The users should be able to scan existing passenger locator forms to fill the application if they already have one. Another way to extend the functionality of the application is the implementation of QR code generation that would allow the security at an airport or country border to scan your passenger locator form instead of reading the application.

Another recommendation is the overview of the User Experience and User Interface Design along with the consideration of new features. Based on the feedback on the final application a certain percentage of users answered that the application's interface is not clearly understandable and the feedback for the users is not providing clearly what went wrong. On the other hand, as this report is written, the official Government website has already announced new rules to be followed such as the requirements of booking tests or providing negative tests.

8. Reference list

ALTVATER, A., 2021. *“What Is SDLC? Understand the Software Development Life Cycle*.” [online] Stackify. Available at: <https://stackify.com/what-is-sdlc/>, [ACCESSED: April 2021].

Donald A. Norman *"The Design of Everyday Things"* (1988).

Dr. Winston W. Royce, 1970. “MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS”. Available at: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf, [ACCESSED: 2021]

Government Website. *“Entering the UK”*. Available at: <https://www.gov.uk/uk-border-control>, [ACCESSED: 2021].

Government Website. *“Fill your passenger locator form”*. Available at: <https://www.gov.uk/provide-journey-contact-details-before-travel-uk>, [ACCESSED: 2021].

Graham, Lisa. *"Gestalt theory in interactive media design."* Journal of Humanities & Social Sciences 2.1 (2008).

Interaction Design Foundation, "*Gestalt Principles*". Available at: <https://www.interaction-design.org/literature/topics/gestalt-principles>, [ACCESSED: 2021].

Sachin Rekhi, *"Don Norman's Principles of Interaction Design"*. [online] Medium. Available at: <https://medium.com/@sachinrekhi/don-normans-principles-of-interaction-design-51025a2c0f33>, [ACCESSED: 2021].

Dam, R. and Siang, T., 2021. *What is Design Thinking and Why Is It So Popular?*. [online] The Interaction Design Foundation. Available at: <https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular> [Accessed 22 April 2021].

Tiky, Y. T. *"Software Development Life Cycle."* Hongkong: The Hongkong University of Science and Technology (2016).

9. Bibliography

ALTVATER, A., 2021. *“What Is SDLC? Understand the Software Development Life Cycle*.” [online] Stackify. Available at: <https://stackify.com/what-is-sdlc/>, [ACCESSED: April 2021].

Donald A. Norman *"The Design of Everyday Things"* (1988).

Dr. Winston W. Royce, 1970. “MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS”. Available at: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf, [ACCESSED: 2021]

Government Website. *“Entering the UK”*. Available at: <https://www.gov.uk/uk-border-control>, [ACCESSED: 2021].

Government Website. *“Fill your passenger locator form”*. Available at: <https://www.gov.uk/provide-journey-contact-details-before-travel-uk>, [ACCESSED: 2021].

Graham, Lisa. *"Gestalt theory in interactive media design."* Journal of Humanities & Social Sciences 2.1 (2008).

Interaction Design Foundation, "*Gestalt Principles*". Available at: <https://www.interaction-design.org/literature/topics/gestalt-principles>, [ACCESSED: 2021].

Interaction Design Foundation, "Heuristic Evaluation" Available at: < https://www.interaction-design.org/literature/topics/heuristic-evaluation>, [Accessed 26 April 2021].

Interaction Design Foundation, "User Experience (UX)Design" Available at: < https://www.interaction-design.org/literature/topics/ux-design>, [Accessed 27 April 2021].

Interaction Design Foundation, "What is Usability?" Available at:< https://www.interaction-design.org/literature/topics/usability>, [Accessed 26 April 2021].

Sachin Rekhi, *"Don Norman's Principles of Interaction Design"*. [online] Medium. Available at: <https://medium.com/@sachinrekhi/don-normans-principles-of-interaction-design-51025a2c0f33>, [ACCESSED: 2021].

Dam, R. and Siang, T., 2021. *What is Design Thinking and Why Is It So Popular?*. [online] The Interaction Design Foundation. Available at: <https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular> [Accessed 22 April 2021].

Tiky, Y. T. *"Software Development Life Cycle."* Hongkong: The Hongkong University of Science and Technology (2016).

10. Appendices

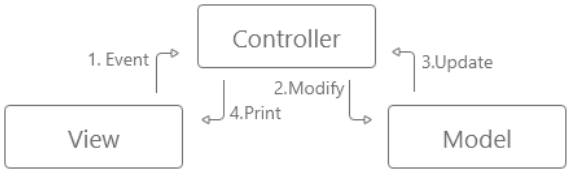
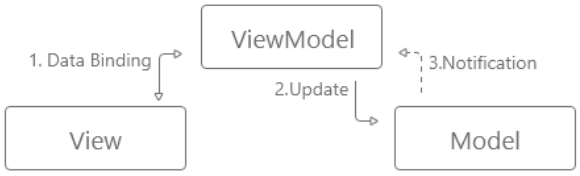
Appendix A: Title

The document is referencing the previous study for its design and other methodologies that was implemented during the development. The application can be accessed at: <https://github.com/Chrys-code/UniMeetingsApp>

Appendix B: Ethical consideration

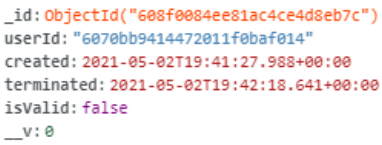
Appendix C: Design Pattern

The table below shows the difference in how the “Model-View-Controller” Model and the “Model-View-ViewModel” Model works.

Model	**Diagram**
“Model-View-Controller” Model	
“Model-View-ViewModel” Model	

Appendix D: Database Objects

The table below represents the database objects created during the phase of database design.

Collection name	Object
UserSession	
Users	

Appendix E: Figma prototype

Access link to the Figma prototype available at:
<https://www.figma.com/file/elYQlSbZU37hGadCjPM4gG/Travel-app-UX?node-id=0%3A1>

Appendix F: Links to the application

Please note the production environment is provided by Heroku. Heroku is the only free platform that provides a runtime environment that allows the application to process in the background. This is required for the application due to its server-side rendered pages and the server itself.

The problem is that Heroku shuts down the server if the application is not in usage and the service-worker is generated on each build, which becomes unavailable if the application is once

shut down then started again. The solution is that the application needs to be re-deployed to the environment for the full usage that lasts until the server shuts down again otherwise the application will not do caching and will not be downloadable and just runs as a normal web application.

Please get in touch if the feature needs to be confirmed or check the Github repository.

Github repository available at: <https://github.com/Chrys-code/Dissertation>
Application available at: <http://c19travel.herokuapp.com/>

Appendix G: Survey results

Access link to the survey data collected on the final product. Survey data available at:<https://docs.google.com/forms/d/19Hb55tRtC36-AGuG1QemGsnHN5aWo_3snBoB2ppWemE/edit?usp=sharing>