Alex McCaughran H8963254

TM470 - EMA

07/09/2024

HistoryHike:
Combining Culture, Fitness and Education to Better Mental Health

Problem Description

We face a deepening mental health crisis, observable through metrics like rising suicide rates *(Hedegaard, 2018)*. Its causes are numerous, however, at its core lies an identifiable lack of exercise *(Harris, 2018)* and disconnection from one's surroundings *(Health effects of staying at Home Too Much, no date)*. A key ICT driver of this problem is addiction to social media *(Ergün et al, 2023)* and other harmful digital content, which adversely affects young people in particular.

This detachment from reality, stemming from detrimental screen habits not only increases likelihood of depression, but also diminishes their connection to local history and culture, which I personally find to be a great source of both emotional grounding and intellectual stimulation. Building on my experiences with fitness, mental health, and history, I propose a solution which replaces unhealthy screen time with healthier habits through a Software project (my chosen specialisation).

My solution is an Android app, called HistoryHike, that engages users with their surroundings and local history through exercise. Users select nearby "quests" and are guided through historical stories in the locations they unfolded. Quests have destinations as objectives, which are completed when reached. After completing a quest's set of objectives, users will be rewarded with a digital "artefact", to provide a sense of achievement.

This gamified exercise method will enrich cultural knowledge, improve fitness, and enhance mental health by allowing users to learn as they exercise. Studies have shown that exercise apps lead to mental health benefits *(Liu et al., 2023)*, and educational apps can combat depression and anxiety *(Goldberg et al, 2022)*. Using walking apps is recommended by relevant organisations with expert knowledge *(Humber, 2023)* to lower the barrier for entry to exercise and helping users discover new walking routes..

There are numerous existing fitness apps that aim to improve mental health, but HistoryHike differs in several ways. Apps like Pokémon GO and Couch to 5K encourage physical activity through gamification, but focus solely on fitness without incorporating learning. Educational apps, such as Google Arts & Culture targets only cultural learning, with no emphasis on exercise. My idea is to combine the positive effects of these apps through promoting exercise and simultaneously educating users about local history, offering a sense of appreciation towards their surroundings during physical activity and intellectual engagement. This combination aims to compound the positive effects of exercise and learning, making it a holistic solution to a social problem.

I expected that, based on prior research mentioned above, users will report positive mental health outcomes after engaging with the project, due to the exercise and educational aspects of it. These outcomes will be measured through user-survey self-reporting. Local histories of diverse communities will also be represented, to ensure their inclusion with the app's predicted mental health benefits.

Goals

Given the nature of the problem, and the solution I aim to create, certain project goals were obtained *(Table 1)*.

Table 1

Goal	Method
(1) Create an Android application	Use relevant development tools to create an easy-to-use app. Ensure User Interface is intuitive, informative and easily navigable.
(2) Model the core aspects of this software project	Obtain user stories, document requirements and draw diagrams to understand the software components needed for the project to succeed.
(3) Create hierarchical data structures to collect, store and use quests, objectives and artefacts.	Apply relevant software engineering patterns for easy development and efficient logical structures.
(4) Allow in-app location tracking and proximity detection.	Research, obtain and apply an appropriate solution to track location to quest objectives.
(5) Provide a method of viewing collected artefacts from completed quests.	Create a separate UI component within the main app's view to browse the user's collected artefacts.
(6) Creation of a backend, allowing persistent storage of user progress and application data across compatible devices. Should also allow transmission of data to users, for example downloading new quests or images.	Use relevant technologies to store relevant data, process it and provide remote read/write access to users.
(7) Collect and use feedback from real users when appropriate, then build upon it.	Provide a means of feedback collection and engage in discussions with users to understand how best to proceed, then use it to iterate the project
(8) Ensure compliance with intellectual property/copyright law and data privacy regulations, due to user data usage, such as feedback, account details and geolocation.	Research data regulations when relevant in the development process and provide security features on the backend.
(9) Collect user comments regarding mental health outcomes, addressing the original problem and the project as a solution to it.	Gather user opinions on their feelings after using the app to determine the project's success as a mental health intervention/ preventive measure.

Account of Related Literature

Throughout the life of my project, a very large number of literature was reviewed and referenced in my works. Key pieces of this, along with an assessment of its relevancy is found below *(Tables 2, 3 & 4)*.

Table 2

Category	Source	Credibility	Suitability
Problem Research	NCHS Data Brief, Number 309, June 2018	Government-backed study, therefore very reliable.	Provides a deep statistical analysis of mental health in today's youth. Very important for understanding the depth of the problem to be tackled.
	Social Media Addiction and Poor Mental Health: Examining the Mediating Roles of Internet Addiction and Phubbing	Scientific paper written by long-time psychological researchers (primary author Naif Ergün has been in the field for 12 years) across multiple universities. Likely very reliable.	Investigates the impact of unhealthy screen habits on mental health and their proclivity to become addictions. Useful to appreciate why the problem exists.
	The relationship between physical inactivity and mental wellbeing: Findings from a gamification-based community-wide physical activity intervention	Study published in a popular peer-reviewed journal, making it a highly credible source. Has been cited 26 times, indicating high reliability.	Examines the effects of gamified exercise on mental well-being over time. Provides confirmation of my approach to tackle the problem.
	The Impact of Technology on Promoting Physical Activities and Mental Health: a Gender-based Study	Research paper authored by researchers at Zhengzhou University, lending academic credibility. Also published by a reputable, peer-reviewed journal source, adding reliability.	Studies the efficacy of technological intervention in promoting physical activity and its effects on mental health outcomes. Suitable, as it examines the effectiveness of my own approach.
	The Health Benefits of Walking | BUPA	Fairly credible. Web article written by the head of Mental-Wellbeing at a leading health insurance provider.	Describes the success of my chosen method of tackling the problem. Helpful in corroborating my method's use in tackling the problem.

Table 3

Software Design Patterns & General Guidance	User Interface Design & Evaluation of Mobile Applications	Published in the highly reliable International Journal of Computer Science and Network Security, designating credibility.	Greatly suitable in providing general direction for implementation of a useful interface
	Android Architecture Patterns - MVC, MVP, MVVM, MVI, Clean Architecture	Fairly credible. Web article written by expert Android developers published on a non-scholarly platform.	Very suitable. Describes ways of writing maintainable and performant project code.
	Android™ Development Patterns: Best Practices for Professional Developers	Learning resource published by O'Reilly Media who are an authoritative source in technical publications.	Highly suitable in guiding my project's development.
Resource research	Java REST API Frameworks	Written by a Senior Lead Architect, presumably quite a reliable source.	Explores different Java frameworks for backend API development. Useful to inform choice of resource.
	What's the Difference Between MySQL and PostgreSQL?	Published by a leading cloud computing platform, therefore very credible.	Compares two of the web's top RDBMSes. High suitability in choosing a resource.
	Best Image Loading Libraries for Android Analyzed	Article written by a frontend engineer, somewhat reliable.	Compares Android image loading and caching libraries, suitable for resource choice.
	Mapbox, OpenStreetMap or Google Map	Article by a non-technical CEO, low reliability.	Compares the features of the most popular free map APIs. Useful to choose which resource.
	Google Maps vs. OpenStreetMap	Article written by an experienced CMS developer, providing fair credibility.	Compares features and ease of implementation for two popular map APIs. Also useful in resource choice.
Resource Usage	Spring Boot Best Practices for Developers	Article by an experienced enterprise microservice researcher, giving fair credibility.	Describes industry best-practices for developing projects with a chosen resource, very suitable.
	Spring REST	Trusted educational resource regarding Spring framework REST APIs cited in academic writing. High credibility.	Very relevant literature for developing a Spring REST API.

Table 4

Methodology	The Agile Coach | Atlassian's No-nonsense Guide to Agile Development	Extensive knowledgebase created by Atlassian, who are a leading project management software company, thus highly credible.	High suitability in correctly implementing my chosen software development methodology.
LSEPI	Can I sell images I create with DALL·E? | Help Center	Written by OpenAI, the creators of DALL-E. Most reliable source for understanding the usage rights of their AI-generated images.	Provides clear information on intellectual property and commercial use restrictions for these assets. Necessary to ensure legal compliance.
	Licensing – Fonts Knowledge - Google Fonts	Documentation from the creators of the fonts used in my app. Ensured reliability and accuracy.	Necessary to understand the legality of the fonts used in my app.
	UK GDPR guidance and resources | ICO	Official guidelines provided by the UK's Information Commissioner's Office (ICO), the UK data protection authority. Highly reliable.	Comprehensive guidance on the legal obligations for handling user data under the GDPR within my project. Essential to ensure no legal or ethical boundaries are crossed.

Account of Project Work and Its Outcome

Project Planning

Previous studies (TM354 & TM254) taught the importance of correctly planning a project so that its result may fulfil its stakeholders' expectations as closely as possible. My HistoryHike project required meticulous planning to balance technical challenges, user requirements, and mental health outcomes. A primary aspect of this planning was the selection of an appropriate development methodology.

Software Project Lifecycle Model Selection

Table 5

Model	Main Strengths	Main Weaknesses	Suitability for my Project
Classic Waterfall	• Clear structure between stages. • Easy to understand progression.	• Inflexible. • Risk of compounding delays. • Documentation-heavy.	Not suitable due to inflexibility and risk of overrun during development.
Iterative Waterfall	• Allows feedback and modifications after each phase. • More flexible than classic waterfall.	• Still inflexible in later stages, with costly late-stage changes. • Lengthy iterations.	Low suitability. Though more flexible than classic waterfall, this methodology lacked the full flexibility needed for a project of this magnitude while working full-time.
Prototyping	• Rapid development of working components. • Easy to refine user requirements.	• Risk of getting stuck prototyping without making overall progress towards finishing the project. • Risk of deploying a non-finished feature due to the nature of this model.	Medium suitability. Can quickly create usable components, but risks slowing overall progress.
User-Centred Lifecycle	• Prioritises user requirements. • Can reveal new use-cases through user interaction.	• Requires diverse user groups and significant time to analyse feedback.	Not suitable. Too time-consuming to gather diverse feedback and analyse user data.
Scrum (Agile)	• Highly adaptable. • Reflective, iterative approach reduces project risks. • Lowers project risks through breaking development up into small, manageable chunks ending with a deliverable.	• Requires careful time management to prevent scope creep. • Depends on some level of user feedback throughout the lifecycle.	Most suitable. Provides extreme adaptability, clear progress tracking, and manageable levels of user feedback make it ideal for my project needs.

Given the demands of my schedule, including study, full-time work, and potential unforeseen events, adaptability was the main deciding factor in selecting a lifecycle model. All models had their merits, however Scrum (with its iterative nature and focus on flexibility) was the best fit, though I chose to hybridise it with aspects of the more traditional waterfall models by emphasising documentation early in the development process. The idea here was that, through appropriate early planning, I would be able to provide an effective foundation for my project.

Adopting Scrum on my own had its drawbacks. For example, it meant adopting multiple roles: the Product Owner, who logs and prioritises tasks via a “product backlog”; the Development Team, who executes said tasks; and the Scrum Master, who eliminates distractions and ensures overall adherence to Scrum practices. Usually this separation of team member concerns is a benefit to Scrum users, providing effective and focussed project collaboration, however that was not the case for me as a solo developer. The heavy reliance on regular feedback was also a hurdle, due to scheduling availability. Finally, Scrum’s iterative nature based on said feedback can result in scope-creep, as users identify non-functional requirements and additional features which may not have been previously considered.

With these downsides in mind, Scrum has many benefits which made it suitable for this project. Firstly, it is highly flexible. This, in part, comes from the product backlog, which dynamically tracks finished work and work to be done. The backlog is revisited and amended often throughout development, providing this adaptability. Scrum implements Agile’s core idea of Sprints, whereby the entire Scrum team engages in regularly timed development sessions, ending with a Sprint Review. During this review, team members visit the product backlog and plan work upon its items, as well as reflect on previous ones with user feedback. Each Sprint also finishes by providing a deliverable– something tangible to add to the overall project– providing good tracking on the work already done and what remains. In my case, I reflected on progress made and tasks to be done at regularly timed intervals on my own. Sprints were also planned to provide Minimum Viable Products (MVPs) as deliverables upon completion as often as possible, which was then reviewed by my userbase.

Scrum also implements the idea of a Daily Scrum *(Rehkopf, Scrum Sprints: Everything you need to know)*. With this, team members discuss work for the day ahead, what may hinder them and how to overcome this. For my project, this daily reflection was done alone and involved heavy reflection into progress on backlog tasks, on a more modular level than Sprint Reviews.

Due to the above, I found that adopting a Scrum approach is clearly best. A key characteristic of Scrum, as an implementation of Agile, is a focus on “Working software over comprehensive documentation”. While I (for the most part) adopted Scrum principles, I also generated a large amount of initial structural documentation prior to creating my software, to guide me through the development process. Thus, my approach is best described as “Scrum-like,” through applying Agile principles but with necessary adjustments for my situation.

Software Design Pattern

After deciding on a lifecycle model for my project, the next step was to choose an appropriate software design pattern with which to create my application. According to Phil Dutson's work, the Android platform is particularly well-suited to a Model-View-Controller (MVC) approach *(Dutson, 2016)*, whereby an application is divided into three core components:

- **Model**: Manages the data and business logic of the application, independent of the UI.
- **View**: Handles the presentation layer, displaying data and sending user actions to the controller.
- **Controller**: Acts as the intermediary, managing interactions between the View and the Model, handling user input, and updating the View with new data.

This separation ensures that each component focuses on its specific responsibility, simplifying development and maintenance by isolating changes to one part of the codebase. There exists, however, other design patterns which also promote maintenance through separation of concerns and clearly defining system architectures. Some of these include:

- Model-View-Presenter (MVP): Similar to MVC, except the Presenter takes on a more active role in managing the UI logic. The View is passive and simply reflects what the Presenter tells it. This pattern requires more development time, due to increased complexity and adding a whole new layer of abstraction through the introduction of a presenter layer *(Dashwave, 2023)*. Because of this, MVP was not well-suited to my project.
- Model-View-ViewModel (MVVM): In this pattern, the ViewModel handles most of the logic and interacts directly with the Model, exposing properties to the View. MVVM can be used with Android's Data Binding library, making the UI automatically update in response to data changes *(MVVM (Model View Viewmodel) architecture pattern in Android 2022)*. This pattern is most well-suited for software systems which deal with large amounts of data *(Dashwave, 2023)* and requires the generation of large amounts of boilerplate code before real development can begin- both of these qualities made this pattern less ideal for my project, which aimed for rapid development and maintainability through simplicity.
- Model-View-Intent (MVI): This is a more reactive, Android-specific approach, which treats the user interface as a function of the application's state. User interactions are modelled as Intents, which trigger state changes, making this pattern suitable for apps that require a strong flow of user interaction and state management, as mine is. This, however, comes at the cost of much added complexity and verbosity to the overall codebase *(Gazzah, 2020)*.

Considering this, I chose MVC for my project due to its simplicity and clear separation of concerns, making it an ideal fit for the structure of my app, where user inputs (such as navigating through quests and collecting artefacts) need to interact with data and be reflected in the UI, without sizable data processing or introducing unnecessary complexity.

Resource Research

My studies relating to software projects (TM354 & TM254) emphasised how correctly identifying resources and risks in reference to achieving a project's goals are often the guiding success factor. With this in mind, some research to select necessary resources was required, in order to complete my project's goals. This was undertaken at the beginning of my project, with some of this research process highlighted below.

I already had extensive prior experience with many appropriate resources, so their suitability for this project was already known. Examples of such resources are:

- Git.
    - A Version Control System (VCS) essential for managing a project of this magnitude. Git offers complex project management through iteration and branching, providing crucial flexibility for my Scrum-like approach. I have extensive experience with GitHub, where my project repository is hosted.

- Canva.
    - Interface prototyping is necessary for a mobile application with such a rich feature set as this *(Samrgandi, 2021)*. I decided to use Canva, a powerful, free design tool I have used many times before.

- Spring Boot.
    - Regarding a backend web-API framework, I knew that Java would be my language of choice. It is the one which I have developed most of my University work and personal projects with, however how to go about creating the API was an important decision. To speed up development, there are many ready-to-use frameworks which are available. This powerful Java framework streamlines building a fully-featured REST API with integrated security features, essential for transferring and processing application data while considering LSEPI concerns like data privacy and legal compliance. Spring Boot excels in automating configuration tasks, enabling rapid deployment of secure and capable APIs *(García, 2023)*. Given my prior experience with this framework and through consulting some of my chosen literature, I was confident in its suitability for my project.

- Physical Android device.
    - Due to geolocation features, real devices were necessary for realistic testing. I have multiple at my disposal, as do my users.

Regarding the resources I needed guidance with, I consulted development forums and expert articles, to get advice from professionals who solve similar problems as part of their role. A primary decision to make was a choice of database system. In my professional and academic life, I have used a range of database technologies (such as MongoDB, SQLite, MSSQL & PostgreSQL). Due to this, I was unsure how best to proceed, but I knew that I at least needed a relational database, to appropriately represent the hierarchical relationship between quests and objectives. To decide, I considered *Table 6* (on the next page).

Table 6

PostgreSQL	MySQL
Advanced features with feature-rich, but complex, syntax.	#1 choice for enterprise applications worldwide, such as Shopify and Uber *(PostgreSQL vs mysql - difference between relational database management systems (RDBMS) - AWS, no date)*.
Faster than MySQL, with 62.5% less time per query, however both are under 50ms when tested *(Musgrave, 2024)*.	Highly scalable and reliable *(PostgreSQL vs mysql - difference between relational database management systems (RDBMS) - AWS, no date)*.
Outperforms MySQL with write-heavy workloads.	Outperforms PostgreSQL with read-heavy workloads.
Complex learning curve *(PostgreSQL vs mysql - difference between relational database management systems (RDBMS) - AWS, no date)*	Simple installation and deployment, compared to PostgreSQL

Given *Table 6*, I decided MySQL was most suitable. While PostGres (pSQL) is more performant, my application's database features were relatively simple, negating any performance observations between the two. pSQL is also very powerful and versatile, however this was irrelevant in my case, due to the simple backend features mentioned previously. MySQL is noted for its ease-of-use and simplicity in installation, which I deem most important. It also outperforms pSQL in terms of reads, which is the most used transaction in my project. I also already had ***some*** experience using MySQL, giving me a headstart in using this technology. Therefore, MySQL appeared to be the best choice.

Another careful consideration to make was between the two leading map APIs, OpenStreetMaps (OSM) and Google Maps, to display user and quest/objective locations. As both of these APIs are free at my predicted usage level and provide the ability to mark locations, other factors were more important. A key element of this decision was whichever API had more available documentation, usage and discussion online, which Google Maps had more of each. Another was accuracy, which Google Maps beat OSM in, with 99% worldwide coverage *(Dovhal, 2023)*. One advantage to OSM is that their maps and location pins are highly customisable *(Google Maps vs. OpenStreetMap, 2023)*, but this is also a downside in my situation as it requires much configuration before use, impeding project progress. I found that Google Maps' default look certainly sufficed. For these reasons, I chose Google Maps.

A final technical resource decision was how best to implement Geolocation, a task I had attempted previously within an Android context. For this, I mainly considered: ease of use, accuracy and tracking update frequency. In my research, I found that Google Play Services FusedLocationprovider API was the best choice, in all three categories, as explored below.

For best accuracy, this API provides location updates using a combination of available GPS, Wi-fi and mobile network data *(Munusamy, 2023)*. Providing appropriate update frequency is achieved through simply configuring a single parameter within this API. Finally, ease of use is incredibly accessible. One must simply add Google Play Services as an Android Studio dependency, add the appropriate permission request to my app then define a LocationCallback() into an Android activity or instantiate a FusedLocationProviderClient object and call its getLastLocation() *(Fused Location Provider API | Google For Developers, no date)*. By comparison, the other major Android geolocation method is LocationManager, which requires a far more complex set-up process and only uses GPS by default, with a more difficult to use interface. The only advantage LocationManager has is that it does not require Google Play Services *(LocationManager | Android Developers, no date)* and can therefore run on all Android devices, however I intended to eventually distribute my app on the Play Store, so this advantage was irrelevant. Also, during development, my application's installation file (.apk) was directly distributed to users, also negating the need to avoid Google Play Services.

Resources Used

After performing the above research, the below resources were identified– along with how they were used and acquired.

Software Resources:

- Android Studio, with Android and Java SDKs, to develop the application front-end.
    - Free download.
- Cloud server, for hosting a Web API backend.
    - Already owned a cloud hosting package for personal projects containing a Virtual Private Server (VPS).
- Cloud MySQL database service, to store user details and app data.
    - Included in my hosting package.
- Design prototyping tool, to design UI.
    - Free online tool: Canva.
- Software board, to facilitate the product backlog.
    - Free online tool: Miro.
- JUnit, to test project features.
    - Used JUnit as it is a fully featured, industry-standard Java unit testing library, integrated with Android Studio.
- Geolocation service, to provide the core function of the project.
    - Implemented the Google Play Services built-in FusedLocationProvider class.
- Appropriate backend API framework or language.
    - Used Spring Boot, due to its capabilities, ease of configuration *(García, 2023)* and previous experience with it.
- Map provider to display user location and quest objectives.
    - Obtained a free key and integrated Google Maps API.
- Git, with GitHub, to: document/record progress; and provide a means of reverting to previous versions.
    - Used a GitHub account and valid SSH keys to access remote repositories.
- Google Play Console account, to allow distribution of my Android app.
    - Created a Play Console account and can later pay the $25 fee to publish my project once LSEPI is fully addressed *(*see *Table 16 on page 39)* as the Play Store is the easiest for subsequent users to access and install apps from.

Physical Resources:

- External stakeholders (users) for feedback and testing.
    - Obtained through personal relationships.
- Physical Android device, to test and use the app at all stages.
    - Already owned two Android devices and all of my users have Android devices on Android 13+.

Project Completion Risks

An important step to success in managing an IT project is risk identification and mitigation, as all of my related Open University studies have taught. A number of risks were initially identified at the beginning of project development, once resource research was underway. They are found below, along with how they could have been, or were, mitigated.

- Underestimation of time needed to complete tasks or learn/improve skills.
    - Medium likelihood, high impact.
        - Mitigated by revising and/or decreasing project scope. This will be reviewed during the regular daily scrums and at the end of sprints.

- Low user availability when required.
    - Low likelihood, low impact.
        - Mitigated by reaching out to others to provide feedback.

- Overestimating my availability.
    - Low likelihood, high impact.
        - Mitigated by revising the project schedule to accommodate for delays during scrums and at the end of Sprints.

- Project management issues, such as scope creep:
    - Medium likelihood, medium impact.
        - Mitigated by revising and/or decreasing project scope at the end of each Sprint.

- Legal and Intellectual Property Infringement (on artefacts used):
    - Low likelihood, high impact.
        - Mitigated by reviewing that artefacts I use are not copyrighted, or are made legally distinct/ declared as “inspired by” the real artefact.

- Loss of work undertaken (for example, disk failure on development computers):
    - Low likelihood, high impact.
        - Mitigated by regular use of version control.

- Low suitability for tools/resources chosen:
    - Medium likelihood, high impact.
        - Mitigated by reviewing progress made and problems encountered throughout the project lifecycle. Should issues be discovered, they’ll be addressed by reading other suitable tools.

Major Tasks and Subtasks, With Schedule

An important step to success in most software development projects is having a clearly defined work schedule, with tasks set at an appropriately granular level. The creation of such a schedule was crucial for me, as planning project development alongside full-time work and other life commitments is a difficult task. By defining such a schedule containing tasks and subtasks, I was able to monitor progress effectively as I moved through my project.

In Scrum, task completion is assessed at the end of sprints. My sprints ran every 14 days, with the task marking Sprint completion ***highlighted like this***. MVP creation, **highlighted like this**, usually occurred at the end of a Sprint. To stay Agile, there was at least one MVP per development phase, allowing iterative development upon user feedback *(Samrgandi, 2021)*. Product-related major tasks were colour-coded between my product backlog and schedule, to track progress at a glance.

Since my approach was Scrum-like (rather than concrete Scrum), I deemed the creation of a general project schedule necessary for success. Usually, due to the Agile tenet of “Responding to change over following a plan”, such a schedule may be amorphous and loose overall *(Drumond, no date)*. This was not totally the case for me and it was only in very extreme circumstances when I deviated too far from the schedule, such as when I identified the need for MVP creation and then swapped two of my development phases. Due to this Scrum-like model I adopted, some sprints ended early throughout development and some later, depending on product backlog jobs completion and user feedback to work on. This emphasises my approach’s flexibility and its suitability to both my study situation and specific project.

To track which goal each major task addressed, it was noted at each major task the goal which it contributes towards solving. By the end of my project’s development, all project goals had been addressed.

Table 7

First Phase			
Major Task	**Subtasks**	**Time Per Subtask (in days)**	**Subtask Start Date**
System Modelling ***(Goals 2 & 3)***	- Create user stories	2	16/02/2024
	- Formally describe system requirements	2	18/02/2024
	- Create product backlog, a key feature of Scrum project management *(Rehkopf, no date)*.	1	20/02/2024
	- Create an analysis model detailing the classes within the system	3	21/02/2024
	- Design the overall user interface	4	24/02/2024
Total	**12 Days**	**Ends On**	**28/02/2024**
Project Setup ***(Goal 1)***	- Install Android Studio.	1/3	28/02/2024
	- Create a new Java Android project.	1/3	28/02/2024
	- Set up a version control system on the project.	1/3	28/02/2024
Total	**1 Day**	**Ends On**	**29/02/2024**
Google Maps And Geolocation ***(Goal 4)***	- Obtain Google Maps API key.	***1***	***29/02/2024***
	- Add Google Maps to the project.	2	01/03/2024
	- Implement a basic initial map view displaying the user's current location.	3	03/03/2024
	- Test the map and geolocation on a real Android device, then resolve any issues identified.	**5**	**06/03/2024**
Total	**11 Days**	**Ends On**	**11/03/2024**
TM470 TMA01	- Finalise TMA01.	***5***	***11/03/2024***
Total	**5 Day**	**Ends On**	**16/03/2024**
Estimated Time: 4 Weeks, 1 day			
Small Break	- Complete other modules' remaining demands (TMAs).	14	18/03/2024

Table 8

Second Phase			
Final App Analysis ***(Goals 2 & 3)***	- Create state diagrams - Review state diagrams, analysis model and product backlog. - Make changes if required.	2 1 1	01/04/2024 03/04/2024 04/04/2024
Total	**6 Days**	**Ends On**	**05/04/2024**
Implement Main UI and Game Logic ***(Goals 2, 3, 4 & 5)***	- Implement the map interface with a quest navigation menu - Begin implementing the analysis diagram, by creating Java classes in an MVC pattern, with the fields as specified - Implement main game Model object methods (such as populating quests' objectives or completing an objective). - Implement main game View object methods (for instance, populating the map with objective markers). - Implement main game Controller methods (for example, checking user proximity from location).	5 2 ***3*** 5 5	05/04/2024 10/04/2024 ***12/04/2024*** 17/04/2024 22/04/2024
Total	**6 Days**	**Ends On**	**27/04/2024**
Test Game Logic ***(Goals 3, 4 & 5)***	- Unit test core model methods, resolving identified issues - Unit test core controller methods, resolving identified issues - Unit test view methods methods, resolving identified issues - Test Application on real device.	4 4 4 ***2***	27/04/2024 01/05/2024 05/05/2024 ***09/05/2024***
Total	**6 Days**	**Ends On**	**11/05/2024**
Small Break	- Write and finalise another module's EMA.	10	11/05/2024
TM470 TMA02	- Finalise TMA02.	7	21/05/2024
Total	**17 Days**	**Ends On**	**28/05/2024**
Estimated Time: 7 Weeks, 6 days			
Small Break	- Complete other modules' final demands (Revision and exam).	14	28/05/2024

Table 9

Third Phase			
Implement artefact collection ***(Goals 3, 4 & 5)***	- Implement MVC approach to artefact collection and storage.	4	11/06/2024
	- Create a view to browse earned artefacts stored.	3	15/06/2024
	- Test all app functions on a real device (At this point, the app stores and processes all data locally).	4	18/06/2024
	- Resolve any issues identified.	***3***	***22/06/2024***
Total	**14 Days**	**Ends On**	**25/06/2024**
Small Break	- Go on holiday to Spain	5	25/06/2024
Cloud Database Setup ***(Goal 6)***	- Design database schema for backend from the analysis model.	2	30/06/2024
	- Deploy the database in-cloud.	2	02/07/2024
	- Populate database with initial application dataset.	1	04/07/2024
	- Test the live database behaves as expected with manual queries. Amend the schema if otherwise.	***4***	***05/07/2024***
Total	**9 Days**	**Ends On**	**09/07/2024**
Small Break	- Visit and stay with family for a while.	3	09/07/2024
Database CRUD API ***(Goals 6)***	- Set up Spring boot Project and configure database access	1	12/07/2024
	- Create appropriate Object-Relational-Modelling	1	13/07/2024
	- Create CRUD operations to interact with database	2	14/07/2024
	- Create API Endpoints for front-end access to these operations	1	16/07/2024
	- Unit test the API locally and resolve any issues.	***3***	***17/07/2024***
Total	**9 Days**	**Ends On**	**20/07/2024**
TM470 TMA03	- Finalise TMA03.	7	20/07/2024
Total	**12 Days**	**Ends On**	**27/07/2024**
Estimated Time: 6 Weeks, 2 days			

Table 10

Final Phase			
Live API ***(Goals 6)***	- Deploy the API in-cloud. - Test the live API, fix any deployment related issues and redeploy, if necessary.	1 2	27/07/2024 28/07/2024
Total	**5 Days**	**Ends On**	**30/07/2024**
API Integration ***(Goals 1 & 6)***	- Create HTTP app methods to interact with the REST API - Integrate these with the app's existing functions to populate app data from the remote server	2 3	30/07/2024 01/08/2024
Total	**6 Days**	**Ends On**	**04/08/2024**
Secure the REST API ***(Goal 6)***	- Implement basic JWT authentication. - Deploy and unit test the API in-cloud. - Resolve any deployment/development issues as discovered.	***2*** 2 5	***04/08/2024*** 06/08/2024 08/08/2024
Total	**11 Days**	**Ends On**	**11/08/2024**
In-app Authentication ***(Goal 8)***	- Create a login screen layout, following initial design specification. - Implement registration/authentication methods following MVC principles.	2 ***4***	13/08/2024 ***15/08/2024***
Total	**6 Days**	**Ends On**	**02/09/2024**
Integration Testing	- Test all aspects of the app, frontend and backend, together in conjunction - Resolve any identified issues.	5 ***6***	20/08/2024 ***26/08/2024***
TM470 TMA03	- Finalise TMA03.	7	02/09/2024
Total	**12 Days**	**Ends On**	**16/09/2024**
Estimated Time: 6 Weeks, 2 days			

User Stories

To obtain initial system requirements in a Scrum manner, I discussed expectations with my users (Appendix A), contributing towards goal 7 from *Table 1*, page 2. Taking LSEPI guidelines into consideration, all users' anonymity was preserved and informed consent was gained for participation (Appendix B), after being briefed on the project's purpose. All users were treated with due respect and valid understanding of their expectations was confirmed before being translated into succinct user stories.

These discussions are summarised, as user stories:

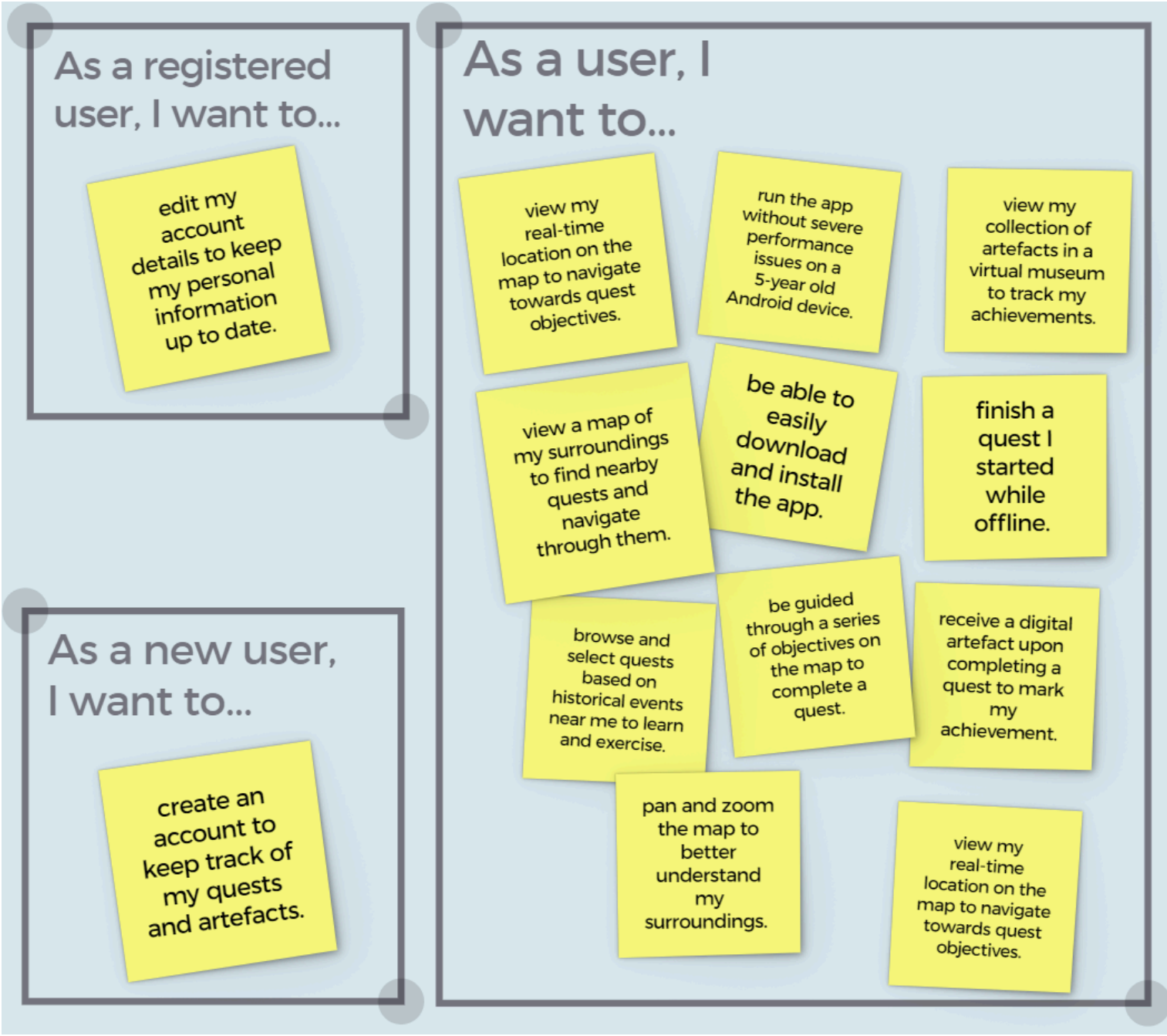


Figure 1 - User stories

Requirement Elicitation

TM352 teaches that, to successfully enact a software solution, aspects of the solution must be broken down into requirements (both functional and non-functional). Requirements can be considered the “how” to the “what” set by a project’s goals, meaning that requirements should be specific in describing by what means a project goal is met. By formally stating these, a software project can become focussed on delivering its stakeholders’ expectations accurately.

Translating my abstract Agile/Scrum user stories into more formal requirements, I found the below, which assists in meeting goal 2.

Functional Requirements

Table 11

Functional Requirement Number	Functional Requirement Name	Functional Requirement Description
FR1	User Account Management	Users shall be able to register, log in to and manage their own accounts.
FR2	Mapping	Users should be able to see their surroundings as a manoeuvrable, two-dimensional map..
FR3	Geolocation	The user’s position will be superimposed onto the map ***(FR2)***, allowing them to navigate towards certain locations.
FR4	Quests	Users will be able to browse nearby historical events, select one, then navigate through a set of points on the map ***(FR2)*** to reach certain objectives or destinations, before completing their quest.
FR5	Artefact Collection	Users shall be rewarded upon completion of the quests from ***FR4*** by receiving “artefacts” representing their achievement, modelled after historical events. They can then view these in their “museum”, which is a collection of artefacts.
FR6	Compatibility	Users should be able to use this app on a range of Android mobile devices, not just the newest OS version.

Non-functional Requirements

Upon my project's initial conception, I had planned to introduce Augmented Reality capabilities to the app, to provide a further sense of achievement for my users upon obtaining a historical artefact. This, however, was discussed with users (Appendix C) upon the second development phase of my schedule and this non-functional requirement was deemed unnecessary for my project timeline.

The ability to remove such an early conceived feature is testament to my methodology's aptness for developing this project. The flexibility of my Scrum-like approach means that this could be revisited and added some time after EMA submission, following further user discussions and iterative development.

Table 12

Non-Functional Requirement Number	Non-Functional Requirement Name	Non-Functional Requirement Description
~~**NFR1**~~	~~AR (Augmented Reality) Capabilities~~	~~Users should be able to see artefacts in AR, for a heightened sense of achievement.~~
NFR2	Usability	Users should be able to browse all app components easily.
NFR3	Performance	The app should not struggle to run smoothly on supported devices.
NFR4	Offline Access	Users should be able to continue a quest without an active internet connection
NFR5	User Data Compliance	User data should be stored and used to comply with data privacy laws, especially location data.
NFR6	Historical Accuracy	Historical data should be as accurate as possible.
NFR7	Intellectual Property Compliance	Since there may be ownership involved with certain historical items, care should be taken to ensure no copyright breaches take place.
NFR8	Cultural Diversity	The histories given should reflect local communities of all forms.
NFR9	Enjoyability	Users should self-report that: they enjoyed using the app; they experienced encouragement to exercise and learn; they experienced improved mental health results from this.

Product Backlog

In Scrum project management, product backlogs outline individual tasks and subtasks that are complete, in progress, or awaiting action *(Rehkopf, no date)*. This organisation technique has allowed me to dynamically adjust and allocate tasks as needed throughout development. My approach, although Scrum-like, follows a well-defined project schedule, so my backlog has remained largely static during development phases, with only minor changes in which subtasks shall be completed before others.

However, using such a backlog enabled adaptability, allowing task engagement outside of my defined schedule, if necessary (such as when a task took longer and was not a prerequisite for others).

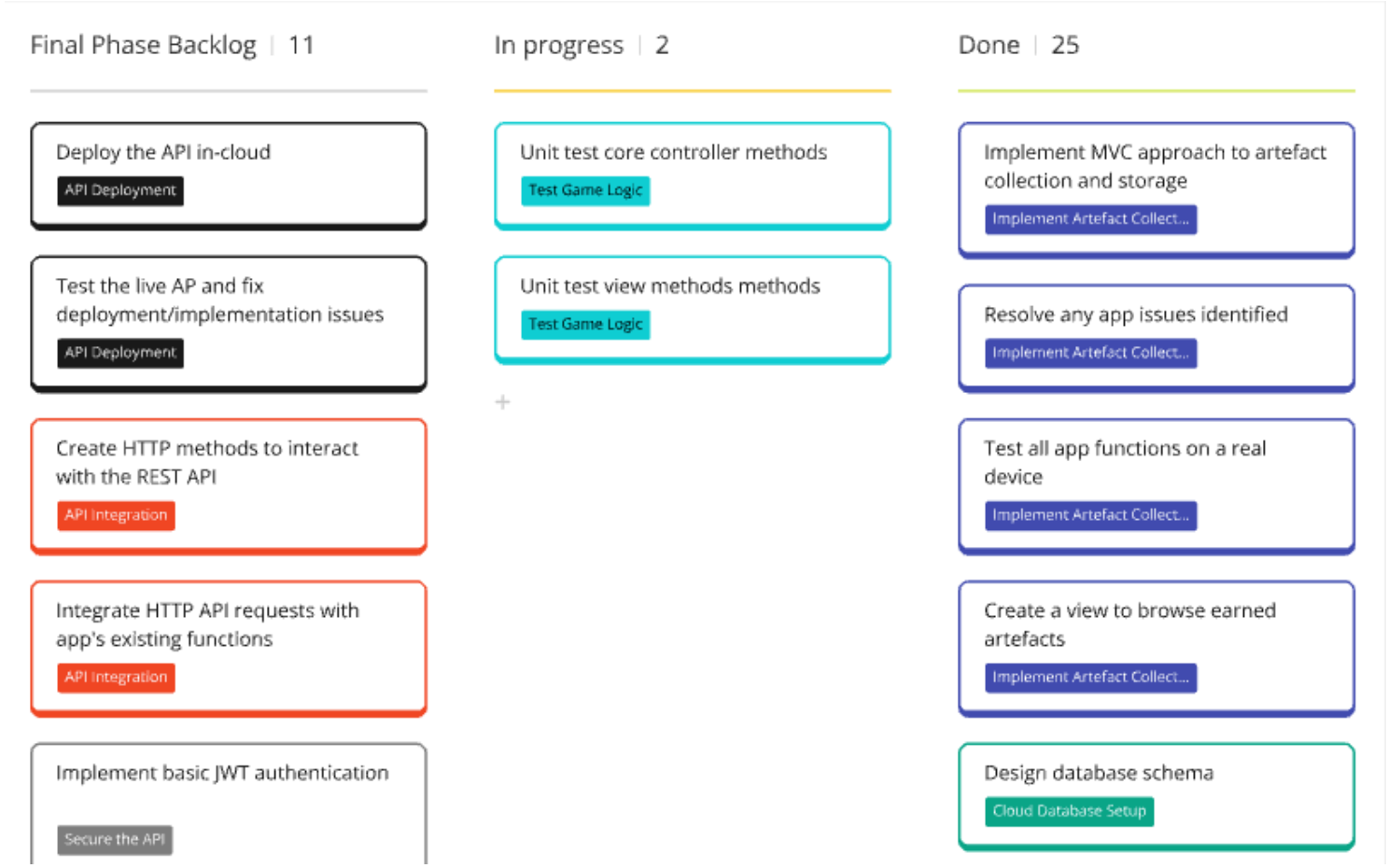


Figure 2 - Working product backlog

Figure 2 contains a snippet of my backlog during the final development phase, showing the categories denoting a backlog item's status.

All backlog cards map directly to a subtask on my schedule and were colour-coded, labelled and categorised according to their associated major task.

System Modelling

Analysis Model

TM352 analyses the utility in system modelling, an aspect of more traditional, waterfall-oriented lifecycle models. The most core of these system modelling tasks is the creation of an Analysis Model, where the classes and objects of a system are described alongside the interactions between them. Most Scrum projects neglect this stage, as it emphasises “Working software over comprehensive documentation” and “Responding to change over following a plan” *(Rehkopf, no date)*, however I decided that this was a necessary step for my project’s development to succeed. I believe that, through providing such a concrete specification, I was able to guide my project’s development successfully.

Below (Figure 3) is my final analysis model, created according to my initial schedule estimate and later amended upon future Sprint completion with time availability due to underestimation of certain tasks’ time requirements. This analysis model underwent multiple iterations from its initial conception upon Sprint reviews. For example: as previously described, augmented reality capabilities were dropped from artefact viewing; association multiplicity between User and Map were amended from 0..* - 0..1, as clearly each user can be associated with one and only one map, not 0 or one; a resourceURL field was added to Artefact to provide a way of retrieving the resource for an Artefact, that is to actually access the artefact asset image.

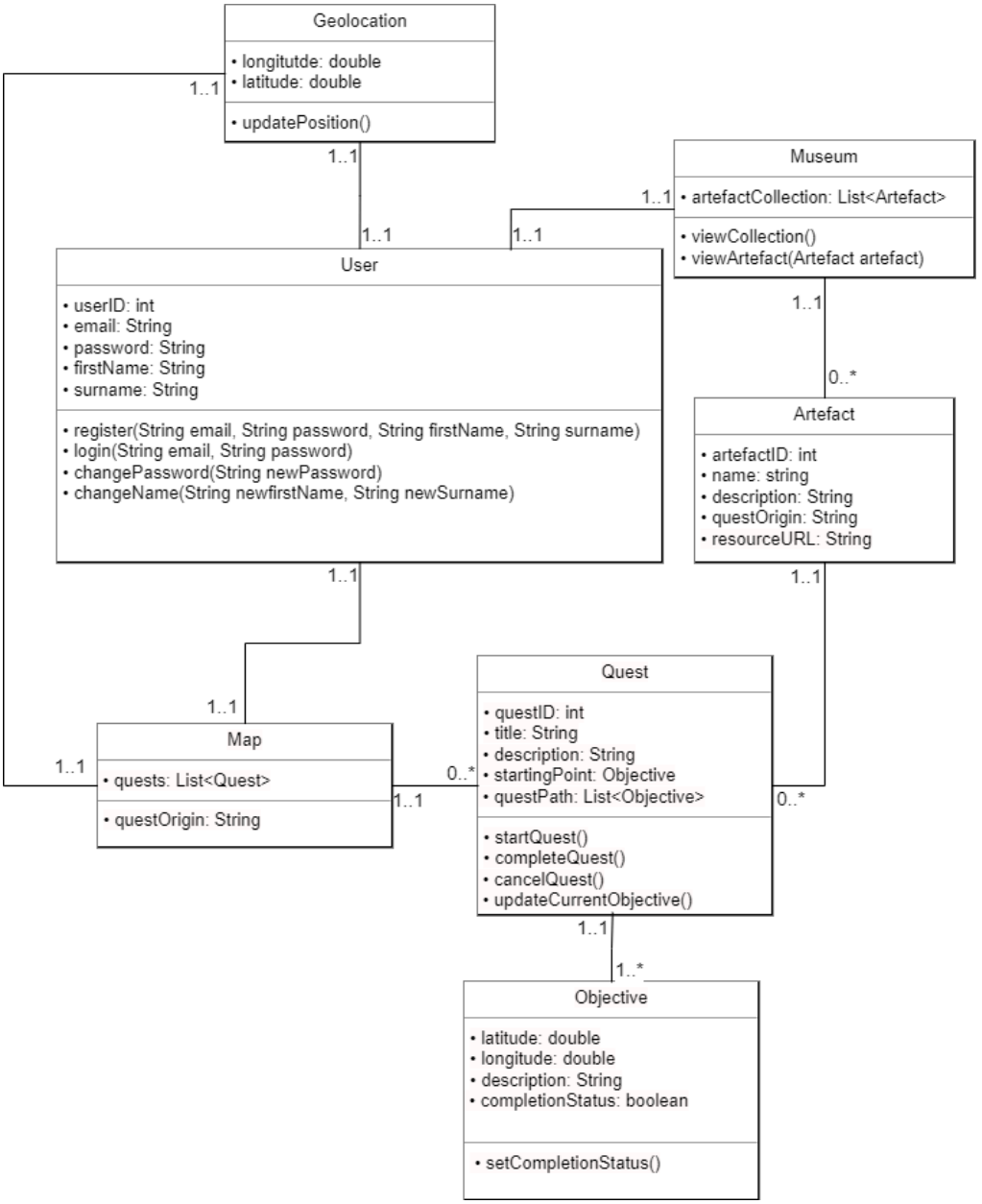


Figure 3 - Analysis Model

TM352's studies showed that loops in analysis model associations can lead to infinite loops between objects/classes during implementation, causing crashing or degraded software performance. Addressing these issues clarifies the system design, maintains state consistency, and facilitates testing. To break my loops, I defined the below constraints:

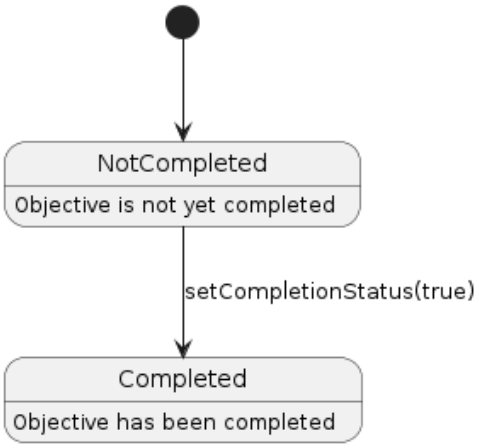
- A User may have only one active Quest at any time. When startQuest() is run, any inactive quest objects are destroyed. Inactive quests are recreated when cancelQuest() is run on the active quest.

- A User's Museum may only contain Artefacts from completed Quests. Users cannot possess an Artefact unless it has been earned through Quest completion.

State Diagrams

To conceptualise my application's control flows, I created state diagrams for two core components: Objectives and Quests. These diagrams depict the lifecycle of the application's most important features, including state descriptions and transitions managed by methods.

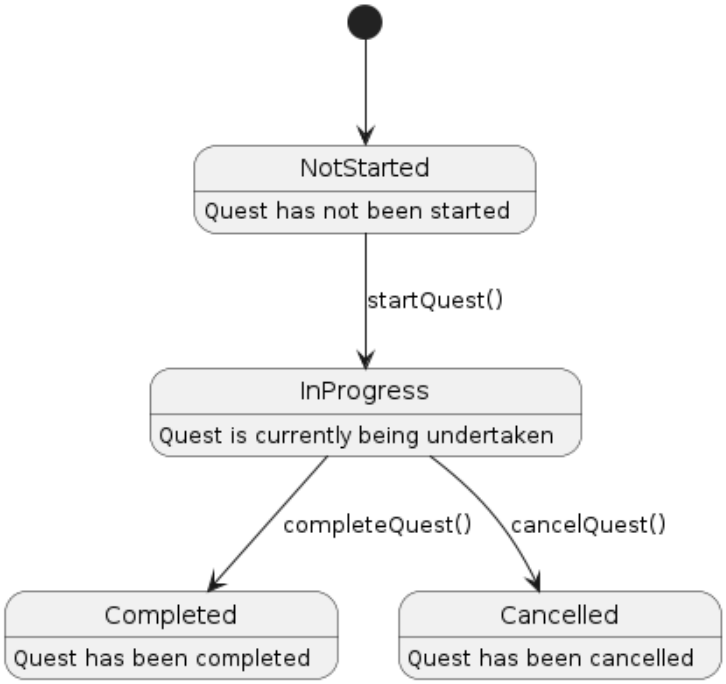
- Objectives:



Objective objects are fairly simple, with a fixed latitude and longitude location (not shown in figure) and a completion status. As objectives' location constants cannot be amended after creation, this completion status completely encapsulates an objective's state and is therefore the only attribute modelled here. It is managed through a single method in a linear way, an objective cannot be incomplete after it has been completed– that is, a user cannot un-visit a location.

Figure 4 - Objective class state diagram

- And Quests:



Quests are slightly more complex, as they have four states that follow a logical flow. A quest begins in a not yet attempted state, then moves to an in-progress state. Finally, a quest can either be completed (once all its constituent objectives are in the Completed status) or cancelled.

Figure 5 - Quest class state diagram

To ensure development of a robust, maintainable and versatile system, software engineers follow SOLID principles when designing and implementing a project (as all my programming-related University modules have taught). I respected these standards when conceptualising and developing my system in the following ways *(García, 2023)*:

- **S**: Single Responsibility Principle
    - Each class in the system has one clear responsibility. For example, the Quest class handles the logic specific to quests, while the Artefact class manages artefact-related functionality. This makes the system easier to maintain and extend.

- **O**: Open-Closed Principle
    - The system's core components, such as Quests and Objectives, are designed to be open for extension but closed for modification. This ensures new quest types or objectives can be added without breaking existing code, promoting stability while adding new features between Sprints/development phases.

- **L**: Liskov Substitution Principle
    - Subtypes, such as specialised quest types, could replace their base types (Quest) without affecting the overall system behaviour. This ensures that the system remains flexible and interchangeable. Currently, there exists only one type of quest, however the system has been designed in such a way that permits for object inheritance. This allows the possibility of subclasses of existing objects to exist in future iterations, post-EMA submission.

- **I**: Interface Segregation Principle
    - All system interfaces were designed to be specific to the needs of classes which interact with them. For example, if a View only needs to display artefacts, it doesn't depend on methods related to quests or objectives. This keeps my system efficient and maintainable through loose coupling.

- **D**: Dependency Inversion Principle
    - High-level modules, like the Controller in MVC, depend on abstractions (interfaces) rather than concrete classes. This makes the system more flexible and adaptable to changes, such as switching how artefact data is retrieved without altering the controller logic.

User Interface Design

When designing a complex application such as this, the views and interactions a user has with it are very important, and so I have used the paper "User Interface Design & Evaluation of Mobile Applications" by Najwa Samrgandi to guide my presentation development.

A formative idea from Samrgandi's paper is the emphasis on User-Centred Design (UCD), which suggests that every element in the application should be tailored to enhance the user's experience, ensuring simplicity and clarity in the interface design *(Samrgandi, 2021)*. By adopting this approach, I have ensured that HistoryHike's UI maintains intuitive and logical, including labels and icons where appropriate. Samrgandi also advocates for iterative design, where user feedback plays a critical role in refining the UI. This is a core idea of my lifecycle model and was particularly useful during my design and development of the project's UI, as I integrated real user feedback to improve design clarity and deliver value to users. I also used this feedback to incorporate another of Samrgandi's techniques, cognitive walkthroughs (Appendix D), discussed later.

In meeting **FR1** from *Table 11*, I designed the account-related UI screens (*Figure 6*).

Figure 6 - Account UI screen designs

The third screen, showing account editing, is only accessible once logged in to a valid, registered account on the app.

In *Figure 7*, you can see some interfaces for my applications main functions, working towards the functional requirements from *Table 11* as annotated.

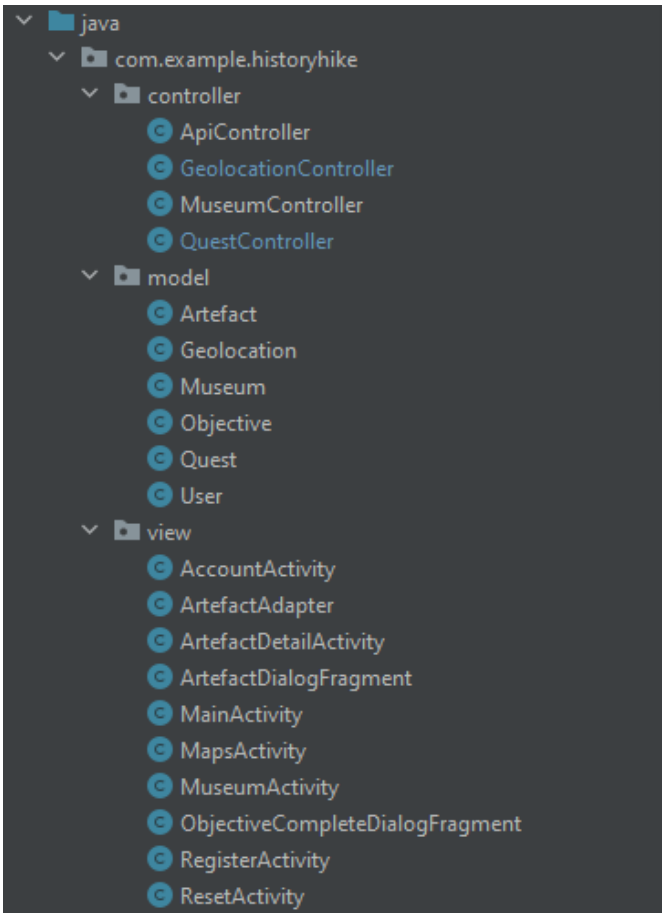
Figure 7 - Quest/artefact-related UI screen designs

A core principle of UCD, and Scrum, is user interaction throughout development, to obtain and take on board their feedback with regards to the interactions with the application, which is fundamental to an application's success *(Samrgandi, 2021)*. Taking an approach as Samrgandi advises, I took feedback from 4 of my users on the above design to iterate upon, contributing to project goal 7 from *Table 1* (page 2). These were then incorporated into the final, finished product. A summary of this below, in *Table 13*, alongside the agreed solution to issues raised:

Table 13

Feedback	Solution
Seeing the distance from a quest in the quest list is unnecessary and not the best way to browse which is nearest. Lacks context from surroundings.	Implement a clickable interface on quest items in this menu to take the map to the quest's starting location (and other objectives when currently doing a quest).
It's unclear, when just looking at the design, exactly how everything works. I might need to explore how the app works when first using it.	Perform cognitive walkthroughs *(Samrgandi, 2021)*, whereby users and I run the app with the implemented design and discuss it together (Appendix D).
Artefact icons take up a large amount of screen space.	Lower the size of artefact icons when viewed as a collection and provide a means of viewing it as a larger image.

Project Development
MVC Implementation



In its current stage, my application classes looked as shown in *Figure 8*. Concerns between application and display logic have been separated, as recommended by industry best-practices *(Dutson, 2016)*.

This class structure demonstrates adherence to SOLID principles through ensuring classes have a single, clearly-defined responsibility *(García, 2023)*.

Within my approach:
Model classes hold and transform application data, **View** classes contain and display presentation logic and **Controller** classes provide user interaction with the underlying model and update **View** classes.

Figure 8 - Android app project structure

Phil Dutson's work *(Dutson, 2016)*, advises adhering to the pattern in *Figure 9* with Android development. I ensured my object interactions follow this pattern. This ensures that the Interface Segregation Principle of the SOLID principles are followed, clearly separating concerns, increasing modularity, and improving maintainability/readability.

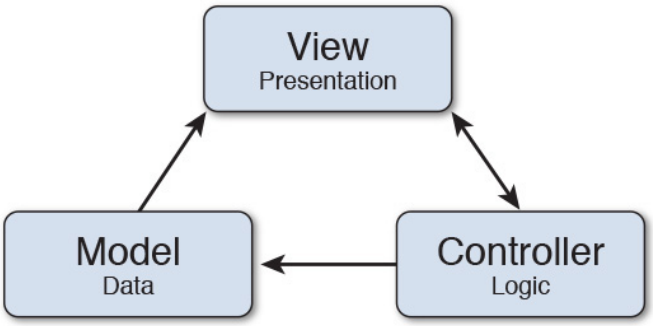


Figure 9 - MVC Interaction diagram (Dutson, 2016)

Unit Testing

In all of my previous studies, I have researched the importance of testing software solutions. Doing so ensures: proper functionality (in line with expected requirements); quality solutions; robustness; and facilitates maintenance. With this in mind, I tested my application's various functions to verify its utility.

Using JUnit, Android Studio's industry-standard built-in unit testing library, I ran unit tests against my MVC app classes. I first created test classes populated with dummy data, such as the setup for Quest unit tests in *Figure 10*.

```
@Before
public void setUp() {
    objective1 = new Objective( id: 1,  latitude: 50.123,  longitude: -4.032,
            name: "Test1",  description: "desc1");
    objective2 = new Objective( id: 1,  latitude: 50.124,  longitude: -4.030,
            name: "Test2",  description: "desc2");

    ArrayList<Objective> objectives = new ArrayList<>();
    objectives.add(objective1);
    objectives.add(objective2);

    quest = new Quest(objectives);
    quest.setId(1);
    quest.setTitle("Test Quest");
    quest.setDescription("Test Quest Desc.");
    quest.setLongDescription("This is a long description of the quest.");
    quest.setFinishDescription("Congratulations!");
}
```

Figure 10 - The steps of setting up a quest object in-app (with test data).

```
@Test
public void testGetCurrentObjective() {
    assertEquals(objective1, quest.getCurrentObjective());
    objective1.setComplete(true);
    assertEquals(objective2, quest.getCurrentObjective());
}
```

I then ran JUnit assertions (*Figure 11*) against all methods to manually check that they output the expected results.

These same tests were performed against live data from the API during later development phases, ensuring

Figure 11 - Simple JUnit test, checking that quests are incrementing their objectives as expected.

After testing all relevant classes, I reached *Figure 12*, confirming that my classes are implemented correctly, following my system modelling work.

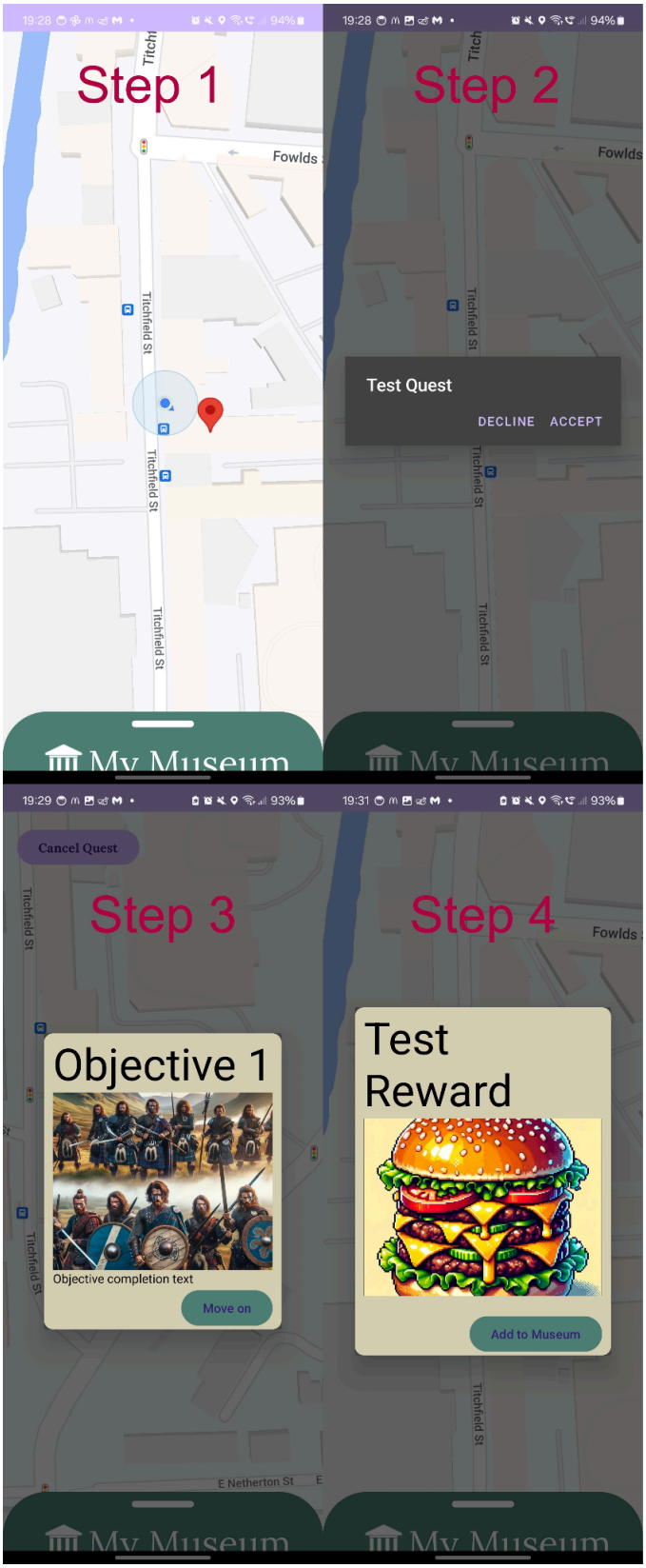
```
BUILD SUCCESSFUL in 1s
22 actionable tasks: 2 executed, 20 up-to-date
```

Figure 12 - Successful build, with no failed tests flagged.

Initial Testing On Real Device (Game Logic)

After successful unit testing, my app was then tested on a real device (*Figure 13*) by my users.

Firstly, I populated the application's QuestController with a test quest, consisting of only a single objective and an artefact reward.



Step 1 shows the user near the quest's starting point (also its only objective).

At Step 2, the user selects the quest. When "accept" is tapped, the quest begins, starting a state change as described in my state diagram.

After the next location update, the View changes to Step 3, showing a DALL·E generated image (royalty-free) of Scottish warriors. The user can cancel the current quest, changing its state again (via the top-left cancel quest button).

After completing the final objective, the artefact is gained, in Step 4. This artefact is another DALL-E generated image (a burger).

A frequent piece of user feedback at this stage was to display the artefact's description when it is earned, to give context and an explanation as to what they are. Another was to provide a means of navigating back to a user's location when the map is moved off of it. A final was to increase the proximity to an objective required to complete it- which was set (arbitrarily) at 2 metres until this point.

These features were promptly added to my product backlog and later implemented into the final product.

Figure 13 - Application running on real device

Backend

Database

In designing a suitable database schema, it's important to ensure all functional requirements can be addressed. My design implemented below allows for all relevant application details to persist on the backend, allowing for game data to be updated without requiring users to install application updates.

Through providing a user table, user's details are stored allowing them to log in and access their personal game data, addressing goal 6. By only storing hashed passwords in the database, and processing the hashing solely at runtime within my REST API, user confidentiality/privacy is maintained. Passwords are combined with a random salt when hashing, to further increase security. User location details are also never stored, just whether or not they have completed a quest. This addresses one of my LSEPI concerns *(Table 16, page 40)*.

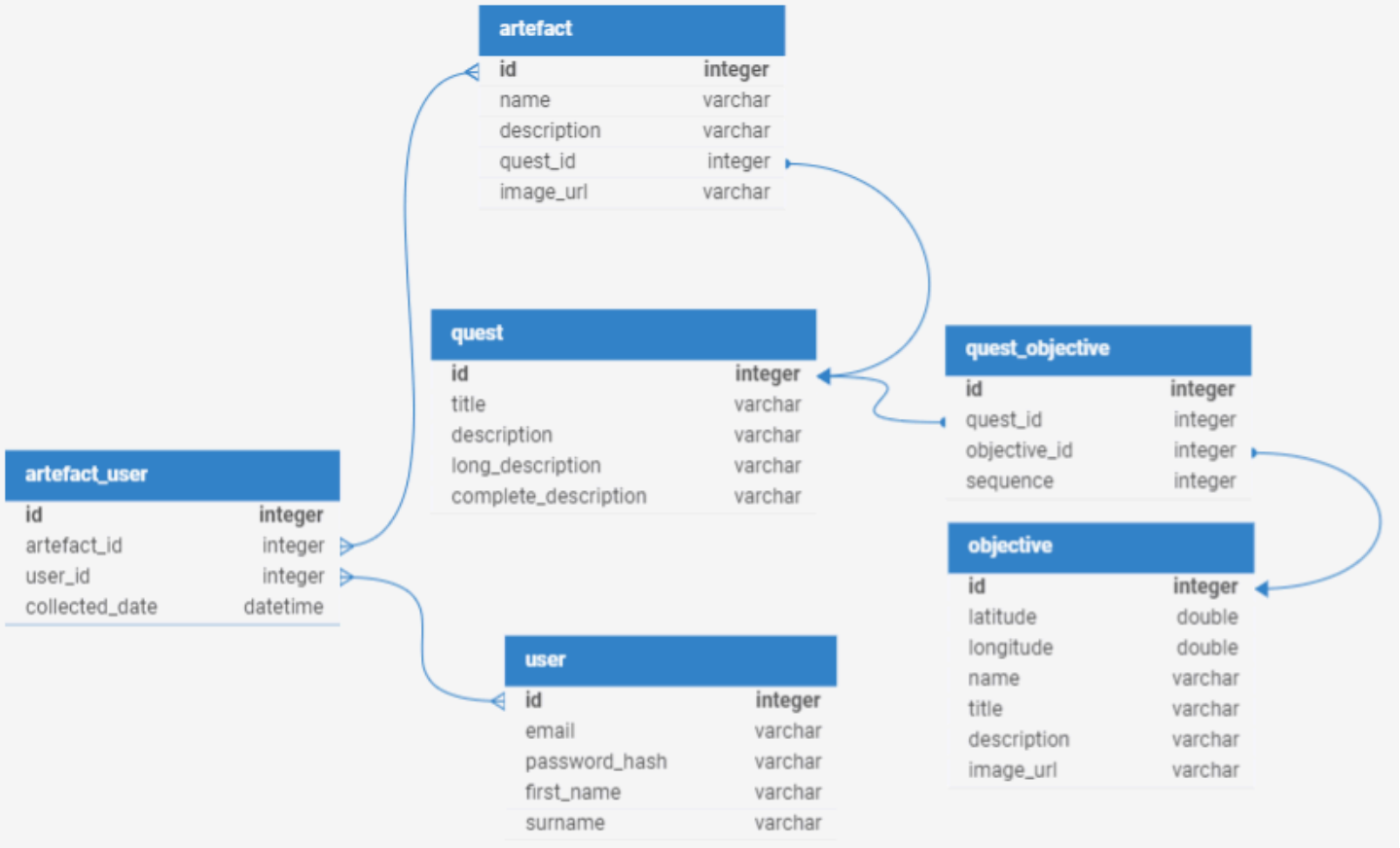


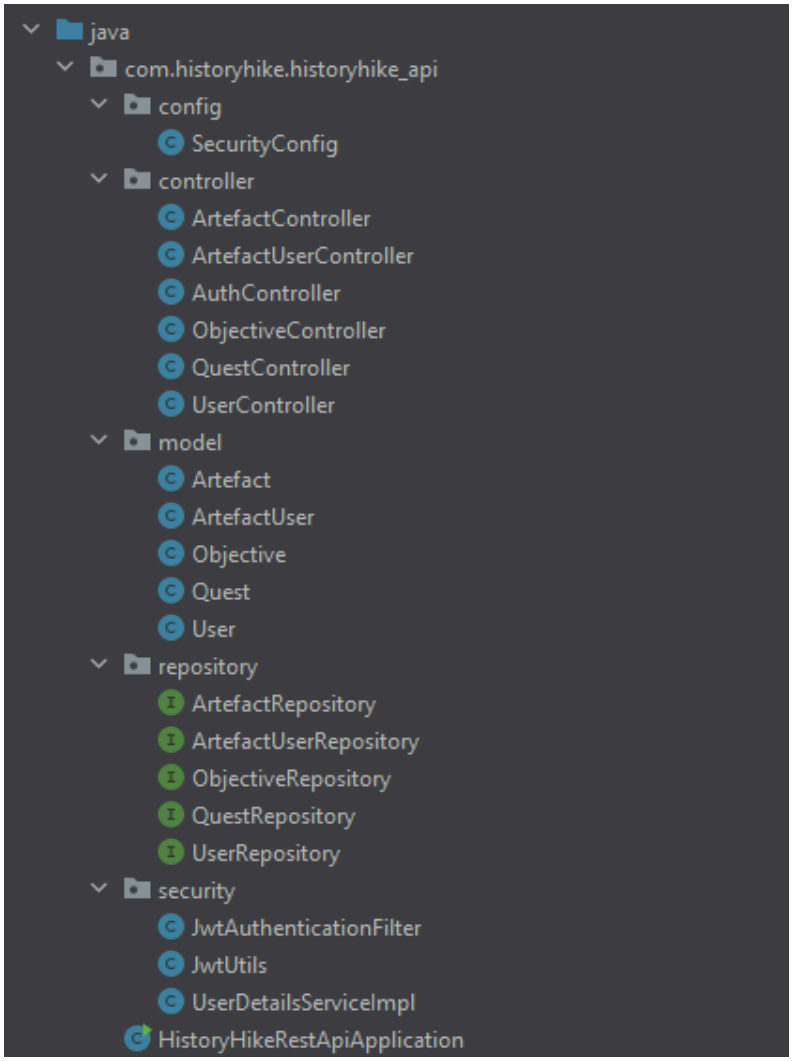
Figure 14 - HistoryHike database schema

After populating this database with dummy data, it was deployed to a cloud-hosting service (as mentioned in my resources section), and could be queried remotely with the database admin credentials. I had issues in doing this, which will be addressed later in the Schedule Adherence section (within Project Management Review).

REST API

To provide data-related functionality to my application, development of a REST API was identified as necessary early in my project's planning. I developed this with Spring Boot, a Java framework specialising in applications of this type.

In creating and maintaining this API, I created modular classes to handle data retrieval/writing (repositories), functionality (controllers) and data modelling (models), as recommended by industry professionals *(Gaur, 2024)*.



All aspects of my database schema, and therefore my analysis model, have been created to provide full functionality.

Through my studies in web development (TT284 & TM352), various forms of authentication were explored, including OAuth, simple HTTP authentication and JSON Web Tokens (JWTs). In these studies, JWT was frequently discussed as a robust method of providing security through simple implementation.

JWTs encode: a user's identity (in my case, their email address); a predefined security key, to prove the JWt was issued authentically; and an expiration time, which I set to 24 hours after the last API request is made.

My REST API handles authentication through JWT (JSON Web Tokens), providing good user security. User's passwords are hashed and salted prior to being written to the database, as mentioned previously, preserving the core security concepts of confidentiality, integrity and availability *(What is Information Security: Policy, Principles & Threats: Imperva, no date)*.

Figure 15 - Spring Boot REST API project structure

These API functions (along with all others) were tested using JUnit, as *Figure 16* shows.

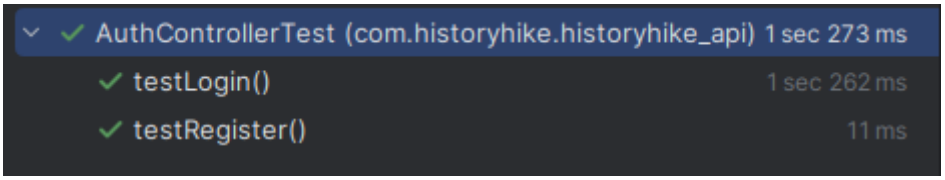


Figure 16 - Spring Boot REST API JUnit tests

This API was then deployed to the same cloud-hosting as my database and could then also be tested through simple HTTP requests to the API's endpoints, both through a browser and Postman (an API testing suite). SSL is included in my hosting package so I configured my API to use HTTPS encryption, obfuscating sensitive user data in transit, acknowledging LSEPI privacy and legal concerns.

Controllers should be kept lean with Spring Boot though focussing only on handling HTTP requests *(Yasas, 2022)*, not business logic, while repositories should focus on database operations. I have ensured that this principle is in place throughout my API (as *Figure 17* shows), maintaining readability and separation of concerns. This is also in keeping with the Single Responsibility and Interface Segregation principles of SOLID.

```
@PostMapping("/register")
public String register(@RequestBody User user) {
    user.setPasswordHash(passwordEncoder.encode(user.getPasswordHash()));
    userRepository.save(user);
    return "User registered successfully";
}

@PostMapping("/login")
public String login(@RequestBody User user) {
    Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(user.getEmail(), user.getPasswordHash()));

    SecurityContextHolder.getContext().setAuthentication(authentication);
    return jwtUtils.generateToken(user.getEmail());
}
```

Figure 17 - UserController class methods from Spring Boot REST API

Catching errors and handling them appropriately is crucial in any software application, particularly when dealing with user input or authentication. With JWT-based authentication, handling exceptions such as missing or invalid tokens and database-related errors is essential to ensure that the system remains reliable and secure. For Spring REST APIs, robust error handling should be implemented at the controller level and managed gracefully, providing meaningful errors, to not disrupt the application *(Varanasi & Belida, 2015)*. We can see this applied in *Figure 18*, where a user's identity cannot be extracted and verified from their supplied JWT, the application exits the method fittingly by providing a meaningful error message with an HTTP 400 BAD_REQUEST response, to be handled by the application (by simply returning displaying the error message "User not found" returned then returning the user to the login screen).

```
@GetMapping("/uncompleted")
public ResponseEntity<?> getUncompletedQuests(@RequestHeader("Authorization") String token) {
    // Extract the JWT token from the Authorization header
    String jwt = token.substring(7);
    String email = jwtUtils.extractUsername(jwt);

    // Find the user by email
    User user = userRepository.findByEmail(email).orElse( other: null);
    if (user == null) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("Error: User not found for the provided email.");
    }

    // Fetch uncompleted quests for the user
    List<Quest> uncompletedQuests = questRepository.findUncompletedQuests(user.getId());

    return ResponseEntity.ok(uncompletedQuests);
}
```

Figure 18 - QuestController method to retrieve available quests from Spring Boot REST API

Current Stage of Work

Following all development stages, I have successfully addressed and met all of my project's stated goals (from *Table 1*, page 2). TM352 and other project-related studies have taught that the functional requirements derived from project goals must be testable *(Ten Attributes of a Testable Requirement 2020)*– meaning they must be clear and unambiguous– to ensure that they are met. These requirements are outlined in *Table 11* on page 19.

At the end of project development, my application is largely as planned, adhering closely to the schedule as outlined previously in this report. I maintained a flexibly structured approach, using my Scrum-like methodology to ensure iterative progress was made across throughout all development phases.

Table 14

Goal	Achieved By
(1) Create an Android application	The app was built using Java in Android Studio, incorporating libraries like JUnit for testing and Glide for efficient image loading and caching. Users engage with historical content through physical, location-based quests.
(2) Model the core aspects of this software project	Application objects (such as quests, objectives, and artefacts) were successfully modelled using detailed descriptions and associations, ensuring productive project progression over time.
(3) Create hierarchical data structures to collect, store and use quests, objectives and artefacts.	The final system modelling work had been faithfully implemented upon non-functional requirement changes successfully.
(4) Allow in-app location tracking and proximity detection.	In-app tracking was implemented using Google Maps API and FusedLocationProvider for real-time location updates based on user proximity to quest objectives.
(5) Provide a method of viewing collected artefacts from completed quests	Implemented a UI component allowing users to view their collected artefacts. Display of artefacts is accessible through the main dashboard upon quest completion.
(6) Creation of a backend, allowing persistent storage of user progress and application data across compatible devices. Should also allow transmission of data to users, for example downloading new quests or images.	A Spring Boot backend was developed to handle user data, progress, and quests, with JWT authentication for secure login and data transfer. The backend interacts with a MySQL database hosted in the cloud.
(7) Collect and use feedback from real users when appropriate, then build upon it	User feedback from testing phases informed design changes and user experience, leading to improvements in navigation and user interface simplification.
(8) Ensure compliance with intellectual property/copyright law and data privacy regulations, due to user data usage, such as feedback, account details and geolocation.	Only open-source fonts were used and all image assets were generated by Dall-E, which denotes the user as the legal image owner. GDPR compliance was ensured through proper handling of geolocation and user data. Data was encrypted where needed, and sensitive information was stored securely.
(9) Collect user comments regarding mental health outcomes, addressing the original problem and the project as a solution to it.	Final feedback confirmed achievement of the project's ultimate aim (Appendix E)– a reduced desire to use social media and positively impact users' mental health. Users reported feeling encouraged to do physical activity and feeling intellectually stimulated.

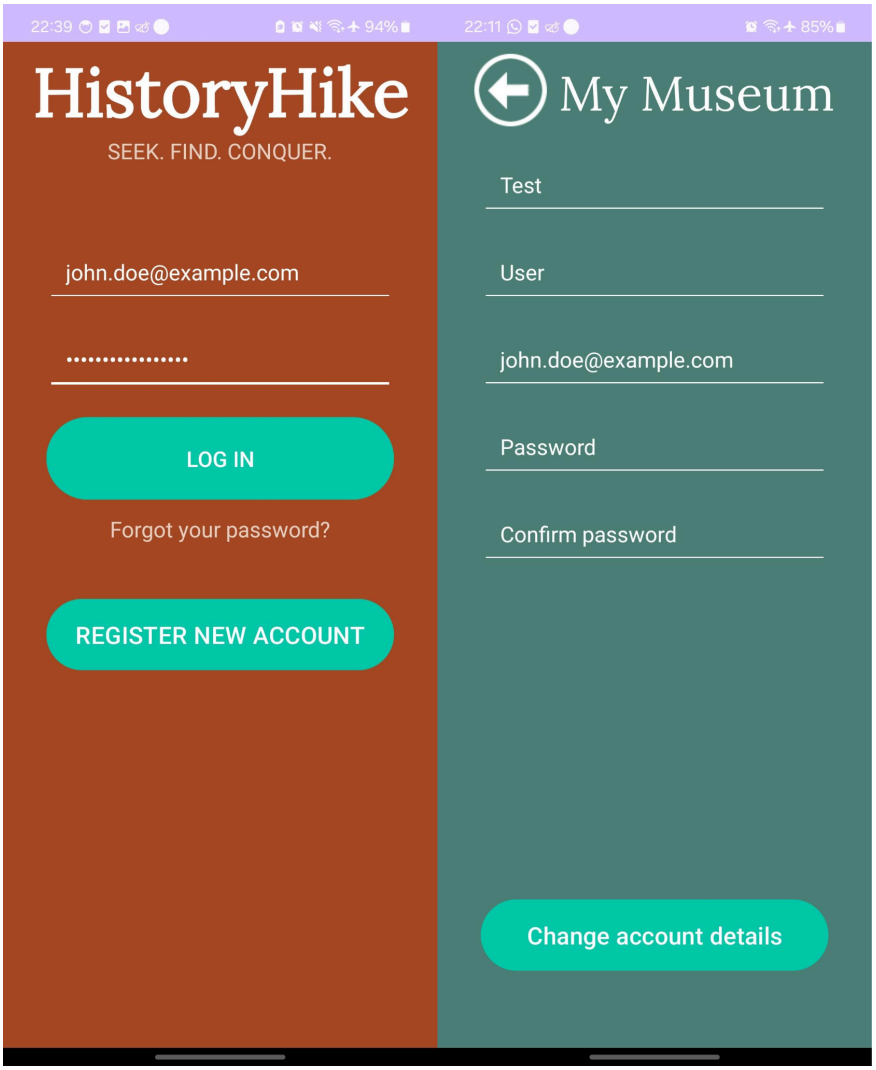


Figure 19 - Current app screenshots depicting login and account editing activities

As figures 19 and 20 show, functional requirement **FR1** was met through providing an initial login and registration interface, and an account amendment activity once logged into a valid account (auto-filled with the current user's account details name and email address).

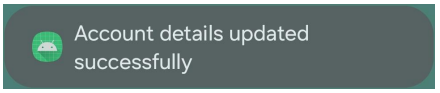


Figure 20 - On-screen confirmation of account detail changes successfully processed by the API

FR2, **FR3** and **FR4** were addressed by the main game function within *Figure 21*, which outlines how a user can see a map of their surroundings and their position in it (reflected accurately). Step 1 in the image shows what a user sees upon logging into the app. Step 2 shows the user accepting the offer to start the nearby quest, while step 3 displays an objective's completion dialog. Users can also navigate back to their current map position if they scroll away from it (via the compass button at the top-right of the map).

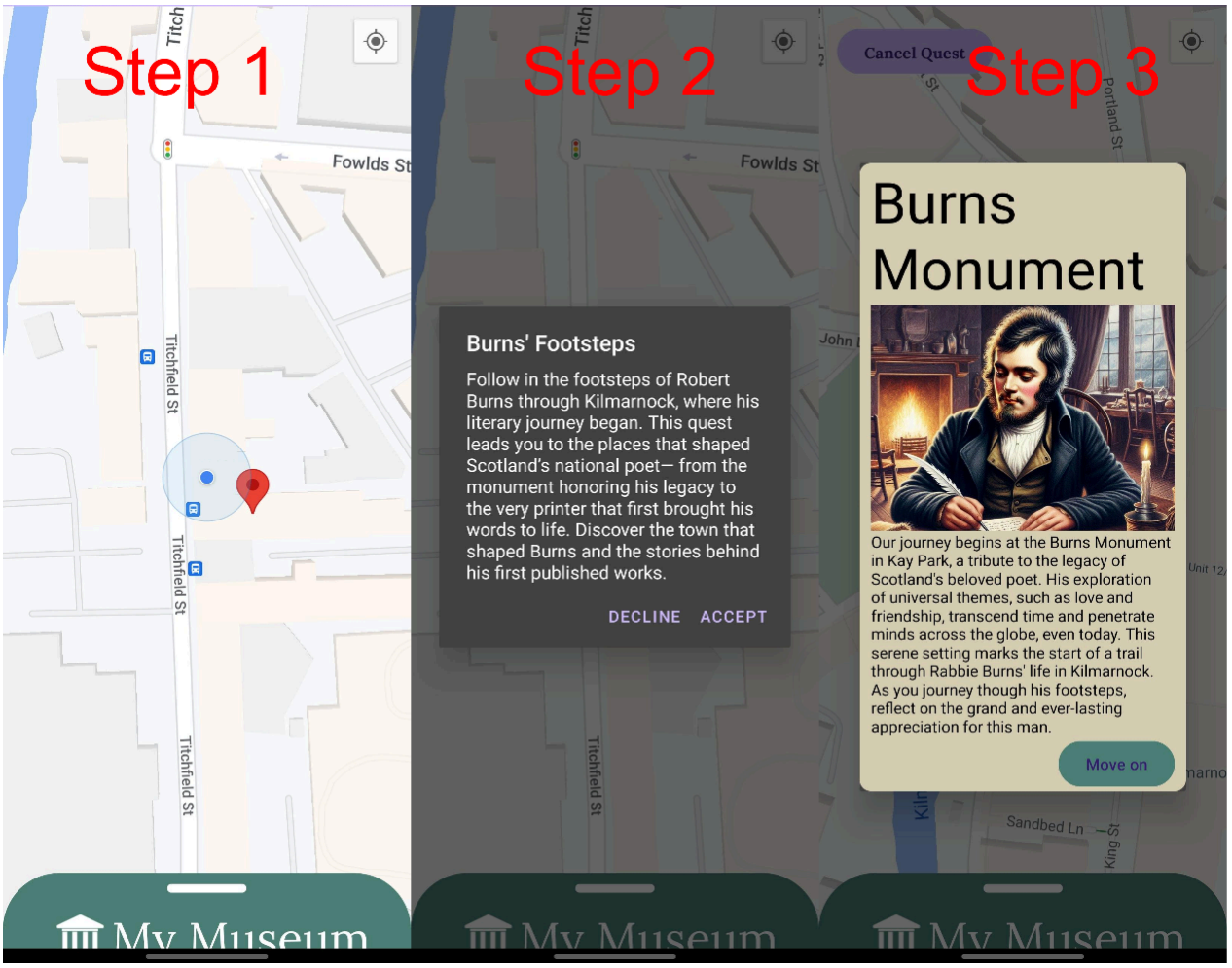


Figure 21 - Screenshots showing the app's main functions, accepting a quest and completing its objective when nearby.

Figure 22 shows my efforts to tackle **FR5**. Step 4's screenshot was taken when the final objective in a quest was reached. Upon approaching the location, another objective completion dialog is shown (like in 20, page 35).

Finally, a user's obtained artefacts can be viewed from within a collection screen. As we can see, following user feedback from *Table 13 on page 27*, the artefact icons are much smaller than in the initial design. This takes up less screen-space and is more viewable. They can then be viewed in full-screen by tapping on them.

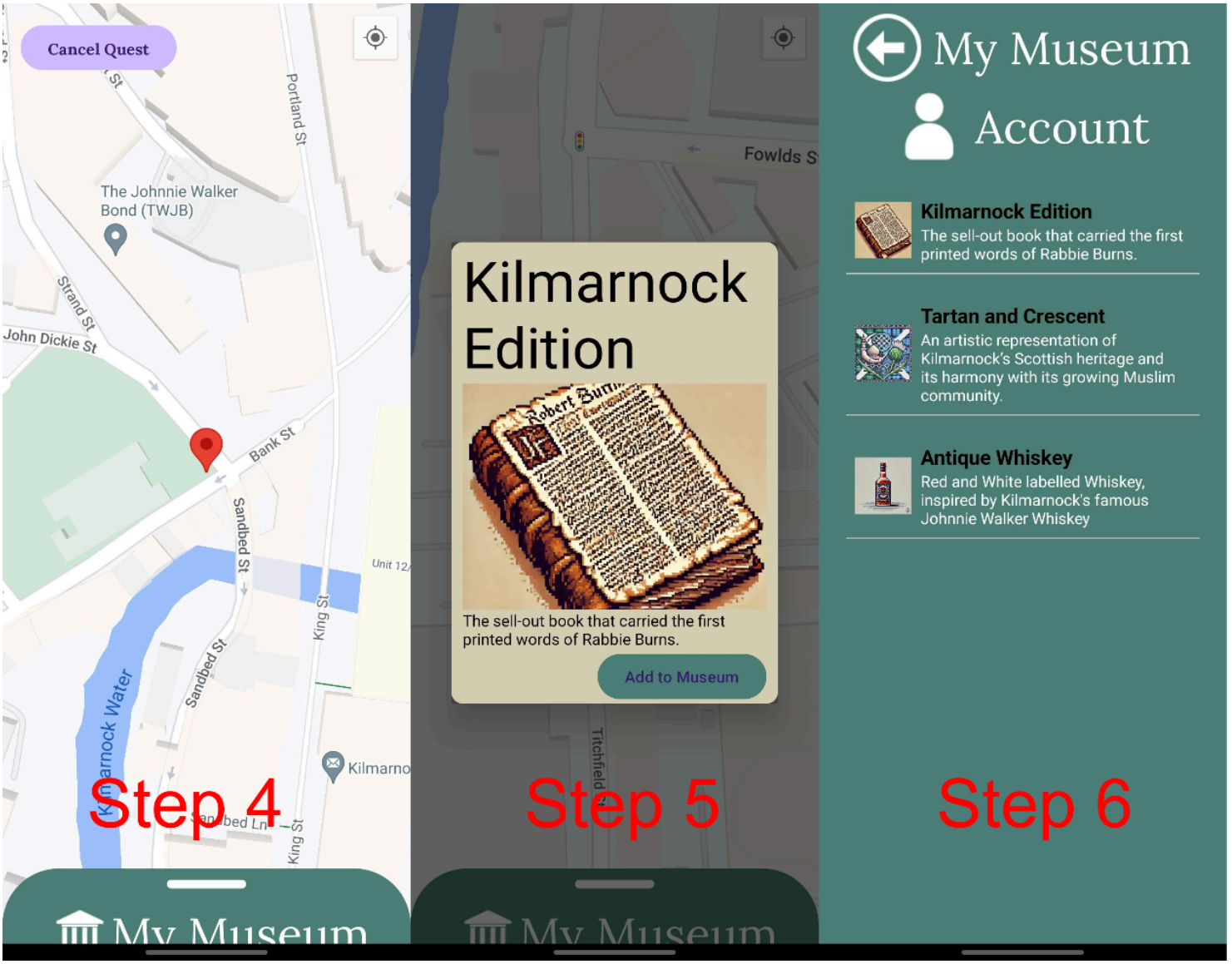
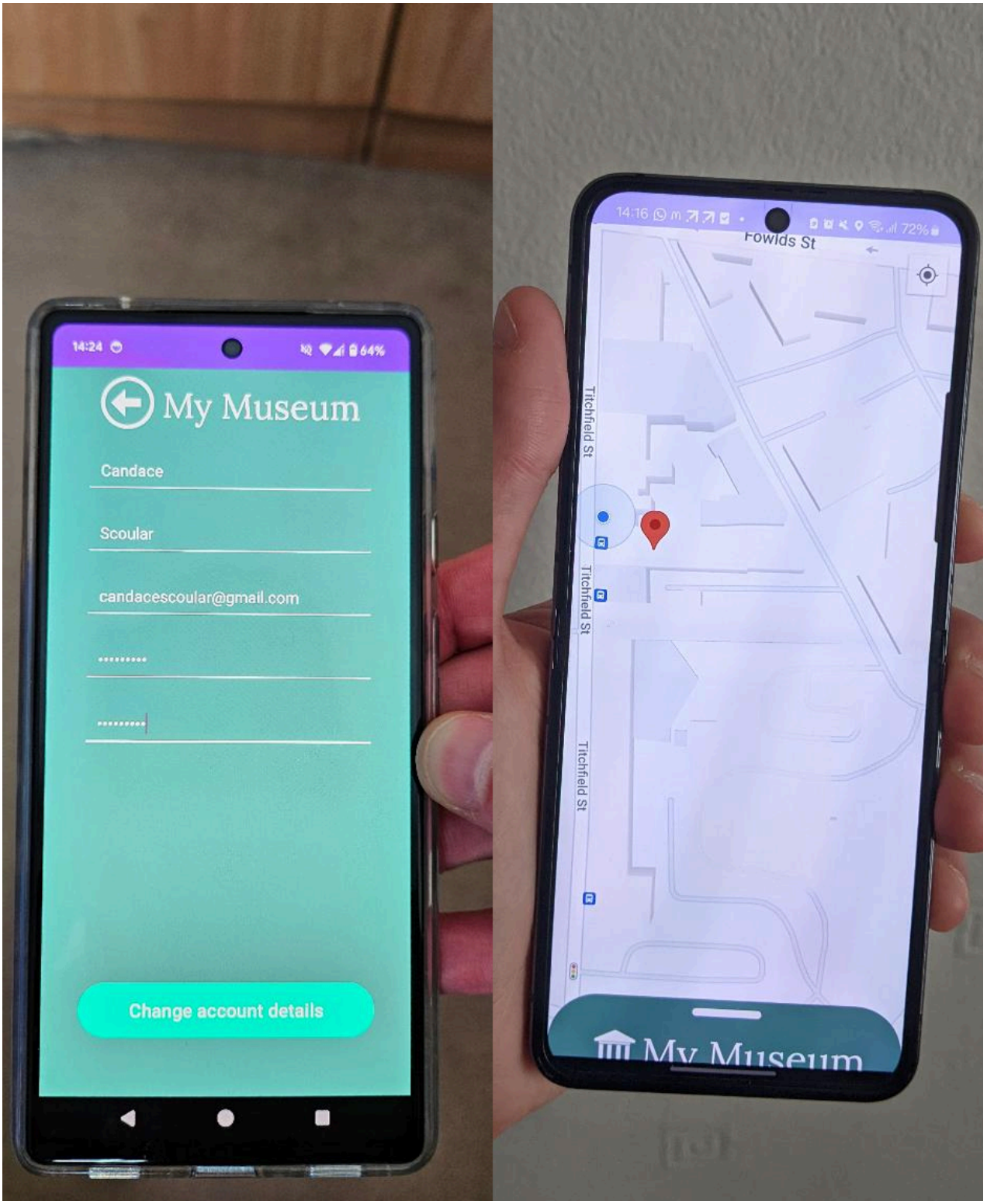


Figure 22 - Screenshots showing the app's other main functions, obtaining an artefact upon quest completion and viewing them later

Finally, below is evidence of **FR6**'s completion. Here we see HistoryHike running on two of my users' devices: a Google Pixel 3a XL (from 2019) and on a Samsung Z Flip 5 (from 2023). Both users reported that there were no performance issues and therefore we can consider **FR6** also addressed entirely.



Throughout the development of this project, I adhered to User-Centred Design (UCD) principles by actively seeking feedback upon Sprint completion and integrating it into the next development cycle. This approach ensured that user needs and preferences were considered throughout the project. By prioritising my users' needs at every stage, I was able to understand their needs more closely and ensure that the final product aligned as closely as possible with their expectations *(Samrgandi, 2021)* and provided a more satisfying user experience overall. The ways that feedback was addressed has been discussed already, however *Table 15* provides a quick reference to this evidence and more

Table 15

Feedback Conclusion	Addressed?	Evidence
Implement a clickable interface on quest items in this menu to take the map to the quest's starting location.	✓	Appendix F
Walk users through application functions, through performing cognitive walkthroughs.	✓	Appendix D
Small icons for collected artefacts, with a full screen viewing option.	✓	Figure 22, page 37
Display an artefact's description when initially collected	✓	Figure 22, page 37
Allow easy navigation of the map back to users' positions	✓	Figure 21, page 36
Objective proximity detection was increased to 10 metres- which was originally set to 2 metres.	✓	Appendix G

With the above noted, however, not all non-functional requirements have been fully realised due to time constraints. While I initially considered implementing AR-features for added engagement, feedback from users suggested that it might not be worth the development time due to its minimal contribution to the core experience of the app (Appendix C). Given time constraints and the need to focus on higher priority features, I decided to postpone AR integration and focus on the more central aspects of my project. The ability to do this lends itself to the flexibility of my Scrum-like approach to this project.

Also, while I researched image caching using Glide, I was unable to implement this feature within the available time frame. Glide would have significantly improved the app's performance by reducing image loading times and conserving data usage; however, without it, images are still fetched directly during runtime, meaning that no functional requirement is impacted. Implementing caching remains a future goal to further optimise the app's performance.

LSEPI

While developing this project, careful attention was given to the Legal, Social, Ethical, and Professional Issues (LSEPI) to ensure compliance, professionalism, and inclusivity. Addressing these concerns not only protects the app from legal consequences, it also ensures that it meets the needs of my user base while maintaining ethical and professional standards. It is important to acknowledge both how these emergent LSEPI were addressed throughout development, but also to explore how these must be considered post-EMA submission.

Table 16

Area	Concern	Mitigation Efforts Made	Future Mitigation
Legal	User data protection (DPA & GDPR Compliance)	User data, particularly account details, was stored and processed securely using industry best-practices (such as hashing, salting and HTTPS encryption in-transit). User locations are purposely never stored or transmitted. Feedback participants' details were also anonymised in this study project.	Regularly perform audits of data storage and privacy practices. Allow users to further manage their own data, such as providing the ability to delete their account and associated data, a legal requirement in the UK *(Right to Erasure, no date)*.
	Intellectual Property Infringement	Artefacts were legally distinct or based on items which are in the public domain, occurring 70 years after the creator's death *(Copyright Notice: Duration of Copyright Term, no date)*, such as Robert Burns' Kilmarnock Edition. Only free-use fonts and royalty-free image assets were used in the app.	Continue ensuring all third-party app assets and stories do not violate copyright or intellectual property laws.
Ethical	Historical accuracy	Historical content was verified using multiple sources, including directly from their sources.	Collaborate with historians or experts to maintain accuracy. Add in-app feedback loops.
Professional	Usability	Thoroughly tested all app components in my development environment and on a range of devices, by a number of users.	Implement in-app feedback loops *(Samrgandi, 2021)* within the app to highlight usability issues to address them in future updates.
	Accessibility	Made the app usable for disabled users, through creating short-distance quests with no extreme-elevation changes.	Implement screen reader compatibility and adjustable font sizes for visually impaired users. Include audio-based quests to support users with reading disabilities.
Equality, Diversity, and Inclusion	Diverse Representation	Quests represented various communities, e.g., Islamic history in Kilmarnock	Expand representation by including more underrepresented communities.
	Inclusive Rewards	Artefact rewards were sensitively selected and do not exclude or persecute diverse communities.	Consult with individuals from different backgrounds to discuss new rewards and content possibilities without perpetuating stereotypes or biases.

While mitigation efforts were taken to address immediate concerns, future steps outlined above must be considered as the project evolves further. I acknowledge that, in my project's current state, addressing Equality, Diversity and Inclusion remain areas for much improvement. Looking forward, development must maintain a focus on preserving data privacy, maintaining historical accuracy, enhancing accessibility, and promoting inclusivity. Ongoing evaluation and user collaboration would remain essential in adapting to challenges (new and existing), ensuring a compliant, ethical, and inclusive project as development continues.

Project Management Review

Developing an application of this magnitude, with a complex frontend and robust backend, presented significant challenges throughout the project lifecycle. Performing the appropriate project management to facilitate development was an equally demanding task. It was not just the technical complexity of the project but also the difficulty of balancing development with personal commitments.

On a more personal note, the challenges of TM470 were exacerbated by life events. Earlier in the year, I was involved in a car accident that impacted my ability to study for several months. Additionally, I was managing two other level-three modules, each with their own rigorous demands. Alongside this, I started a graduate software engineering role with a large multinational corporation, went on a (much-needed) holiday and became engaged during this period. These major life events at such a pivotal time added another layer of complexity to managing my time and focus, making the need for effective project management even more necessary.

In light of these challenges, a structured approach to planning and tracking progress became all the more important to ensure I successfully completed this project. The tools and techniques employed through my Scrum-like approach, such as task management systems and systematic progress tracking, played a crucial role in accomplishing my study goals with TM470.

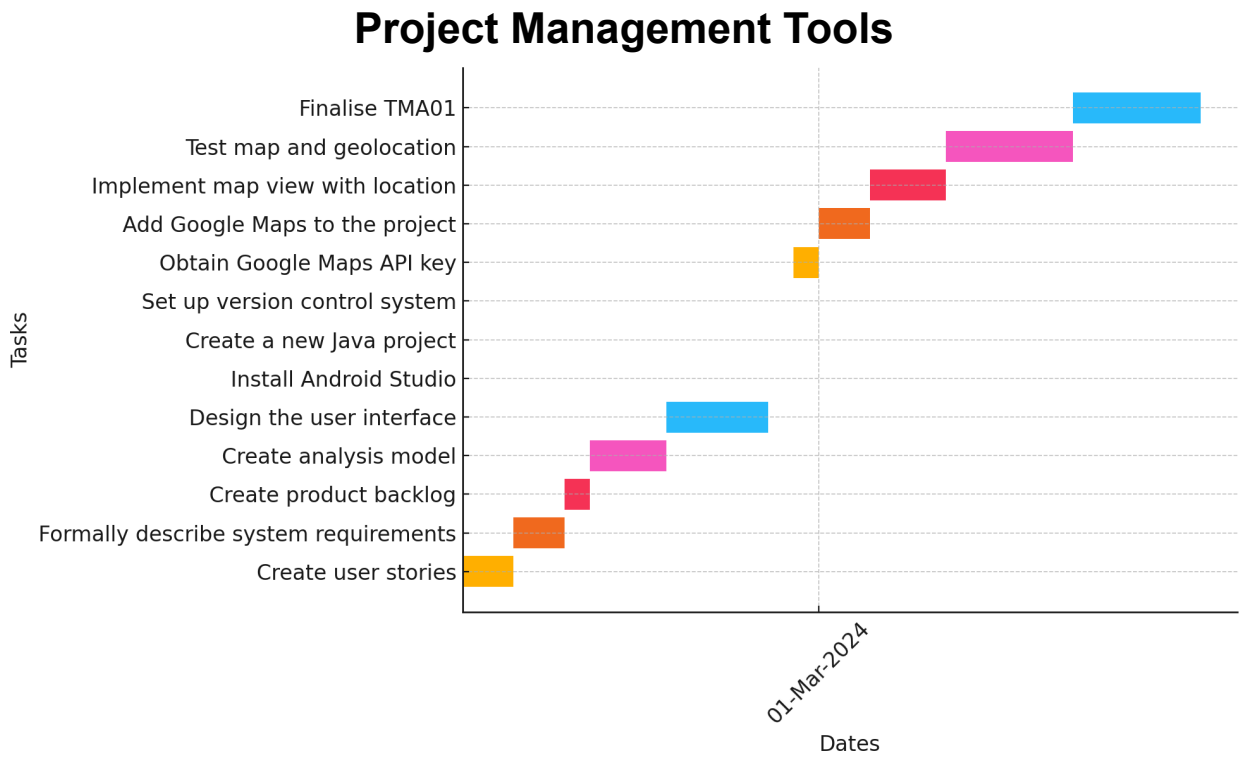


Figure 23 - Initial Gantt chart attempt

The first major challenge was scheduling my very limited time. Before any of my technical work began, I experimented with Gantt charts as a visual way of representing my progression through project tasks- however I quickly found that a simple tabular approach with development broken down into phases was a more suited option to me. I found that the time taken to create and amend these charts, even just at the earliest of stages, led to me wasting time. I also believe that, through manually updating a table with plain dates, I remained more mindful of my tasks and deadlines than I would have otherwise. Overall, the time-cost outweighed any benefits from the results of such work and therefore I deemed this tool un-necessary and distracting– opting for a more simple approach as shown in *Tables 7, 8, 9* & *10*. I found this easier to plan development using my chosen lifecycle model.

At the beginning of my project, I broke my development tasks progression into two-week chunks, called Sprints. I used these as a means to review progress made in each two-week period and discuss the project with my userbase. My Sprints were planned in such a manner that allowed for the creation of a Minimum Viable Product, a meaningful deliverable asset that I could give to my users to comment on, as advised by Agile *(Rehkopf, no date)* and UCD advocates alike. At the end of these Sprints, I made brief notes regarding task completion and anticipation (Appendix H). This approach helped me maintain motivation and concentration through such a long, complex project.

A key project management tool for me was Miro, an online Agile product backlog creation tool. I was able to use this effectively to manage my project's major tasks and subtasks from conception to completion. I found that this Agile-focussed tool allowed me to implement my Scrum-like methodology exactly as intended. By mapping a colour-coded, categorised list of individual tasks to an overall project schedule (as described on pages 13 and 21), I was able to monitor progress to ease progression through my project.

Another critical tool used throughout the development of this project was Git. Without a form of version control, the technical aspects of this project would have been exponentially harder to complete. Git allowed me to maintain a clear and organised history of changes, enabling the safe implementation of new features without risking the integrity of the existing codebase. This meant that I was able to work on individual features while ensuring that any mistakes could be easily rolled back when necessary. This ensured the stability and continuity of the application as it evolved.

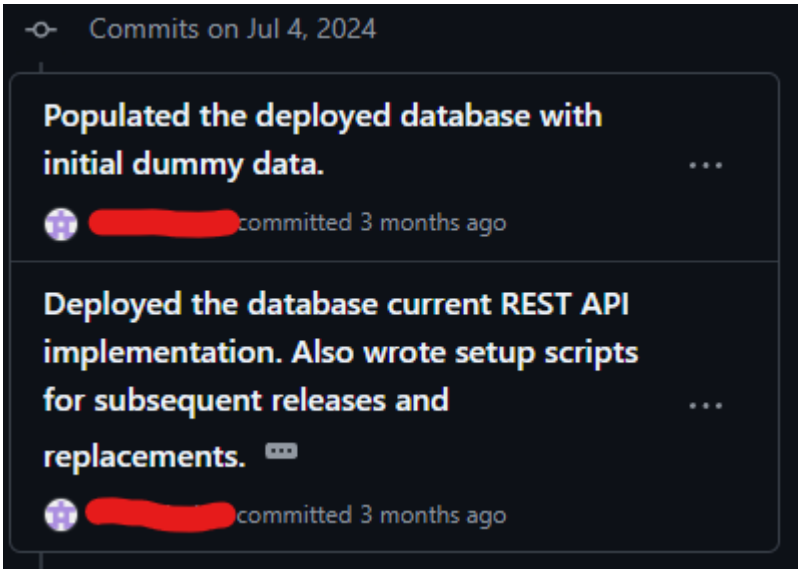


Figure 24 - Backend-related Git commits

Project Resource Review

As part of developing this complex application, I researched and used a wide range of resources for managing and undertaking development. Throughout the project, I relied on a combination of software tools and physical devices to ensure both the technical feasibility and practical testing of the app's features.

In *Table 17*, page 43, I have reviewed the key resources employed in this project. Each resource was evaluated based on its suitability for the project and its necessity. Any comments regarding these aspects are also highlighted.

Table 17

	Resource	Appropriate?	Necessary?	Comments
Software	Miro (Product Backlog)	✓	✗	Useful for visualising tasks, but simpler tools (paper or sticky notes) could have sufficed.
	Canva (UI prototyping tool)	✓	✗	Helpful for creating a visual design, though basic prototyping tools would have sufficed.
	Git/GitHub	✓	✓	Essential for version control and tracking changes efficiently.
	Cloud server	✓	✗	Useful for reliable hosting, though a more traditional or local server would have sufficed. Had I not already owned such a product, I would have went a cheaper route.
	Spring Boot	✓	✓	Absolutely critical for building the backend.While other frameworks I researched would have met requirements, I was able to hit the ground running with a tool I was familiar with and develop with less learning required.
	MySQL database	✓	✓	Necessary to store and retrieve data across multiple devices.
	Android Studio	✓	✓	The primary development environment for the app. Crucial for building and testing the app, especially with JUnit integration.
	JUnit	✓	✓	Critical for testing app components throughout development.
	Google Maps API	✓	✓	Required for displaying location and providing necessary app features.
	FusedLocationProvider	✓	✓	Needed for real-time, accurate geolocation and proximity tracking with fast development.
Physical	External stakeholders	✓	✗	Not completely essential for development, but very important for gathering feedback and improving the app.
	Physical Android devices	✓	✓	Required to test geolocation and proximity features in a real-world environment.

Schedule Adherence

During early Daily Scrums, I realised that my original schedule was unsuitable. Agile approaches, like my Scrum-like methodology, emphasise outputting a Minimum Viable Product frequently as often as possible *(Rehkopf, no date)*. I hadn't fully considered this at first. After reviewing my chosen literature, I decided to swap two development phases entirely, focussing on delivering the project's core app functions before developing a backend. The fact that I was able to perform such a drastic change with minimal disruption to the overall project outcome is testament to my chosen methodology's suitability.

Throughout development, I found the granularity of my task-subtask schedule to provide appropriate flexibility and meaningful milestones throughout my project's progress. As I progressed through these, I have refined the goals/aims of my project and the schedule to achieve them. I found that many tasks, such as "Implement Main UI and Game Logic" could be completed faster (60% the estimated time), and some, such as "Deploy the database in-cloud", took longer (almost 3x as long, in fact). One which was a frequent, repeated obstacle was my implementation of FusedLocationProvider, which ended up causing recurring issues throughout my project, due to a lack of understanding of its edge cases . For example, when a user entered an area with poor GPS signal and no mobile network, the FusedLocationProvider caused my app to crash (an issue that was resolved by telling it to remember the last known location until a signal is regained).

These facts support my risk identification, specifically that of time underestimation and overestimating my ability. Both of these risks compounded when I encountered difficulties in deploying my database in-cloud, a problem which required discussions with my cloud provider's customer service to resolve. My risk-mitigation plans, however, sufficed in dealing with this problem, through addressing next steps throughout my daily scrums. In these daily reflections, I was able to move onto different tasks based on my ongoing project backlog.

I effectively researched required resources beforehand, managing the risk of "Low suitability for tools/resources chosen" by selecting appropriate technologies with which to implement necessary features. This also managed the "Underestimation of time needed to complete tasks or learn/improve skills" risk by avoiding overly complicated tools early on. This risk was also avoided thanks to user feedback regarding the lack of value in introducing AR-features, something which I'd planned from the start. This meant that my Scrum-like methodology had merit, as it allowed me to refocus my efforts based on user comments, embodying the Agile principle of "Responding to change over following a plan" and its aim to "...satisfy the customer through early and continuous delivery of valuable software" *(Lamborn, 2022)*. Through effective use of my identified version-control resource, Git, I've successfully avoided any impact from my "Loss of work undertaken" risk.

Regarding my development approach, I think that my Scrum-like methodology has served me well. Through adopting this, I've reflected regularly as tasks have progressed (during daily scrums and at the end of sprints) which has helped me keep on track and also retain motivation throughout the project's development. This has allowed me to re-assess my project's aims and remove the proposed AR functionality, as discussed earlier, something which may not have been possible with other methodologies (such as a typical waterfall approach).

Personal Development

My experience with TM470 was my first attempt at completing such an ambitious, multi-faceted project, and I have found it tremendously valuable in developing my skills. Throughout this journey, I have handled the various roles required to bring forth a highly complex software solution, including project management, software development, and UX designer. This project has deepened my understanding of independent learning, challenged my problem-solving abilities, and provided me to critically evaluate my work. Each phase presented opportunities for growth, through adapting to challenges (such as life events) and new technologies (like geolocation APIs and Android-platform development). while also learning to balance competing priorities and unforeseen challenges.

A key area of growth was independent learning, especially as I encountered new tools and technologies that were essential to the project. For example, learning true best-practices for a Spring Boot backend and handling real-time geolocation through APIs were both steep learning curves. I was initially unfamiliar with some of these technologies but, through research, online resources, iterative development and trial and error, I developed a strong understanding of these concepts. I can now comfortably say that I am confident in exploring a problem domain to discover appropriate tools. This independent learning experience has been invaluable and I feel that it will certainly serve me well for future projects, both personally and professionally.

Undertaking this project has also significantly deepened my skills in both frontend and backend technologies– with those I was already familiar with and otherwise. I gained hands-on, real-world experience with RESTful APIs, JWT authentication, and MySQL database management. Implementing real-time geolocation using FusedLocationProvider and integrating Google Maps API into the app were areas of particular difficulty at first, but rewarding to implement. Additionally, I enhanced my skills in UI/UX design, where I iterated on feedback to create a more user-friendly interface. These technical skills, particularly in mobile app development and backend architecture, will surely also benefit my future software engineering projects.

Managing the project as a whole alongside other commitments was a significant challenge. I tracked my progress through tools like Miro for backlog management and GitHub for version control. Despite these tools, I occasionally had to adjust task prioritisation due to their complexity or external factors, which was testament to my methodology's suitability. I applied Scrum-like Sprints to break down the project into manageable tasks and incorporated user feedback regularly at Sprint Reviews, which kept me Agile and kept my project relevant to expectations. This was also my first time working under this lifecycle model, exposure to which has already helped me in my Graduate Software Engineer role through providing comfort working under it. Through this, I learned the dire importance of flexibility and prioritisation in project management

Finally, throughout the project, I continuously reviewed and improved upon my previous work. After each Sprint, I conducted retrospective assessments to assess what went well and where improvements were needed. For example, early feedback from users highlighted the need for a more intuitive and useful UI, which led me to revise aspects of the interface. Additionally, I recognised the importance of better scope management after deciding to drop certain features, such as Augmented Reality integration. This iterative process of evaluation and refinement greatly enhanced the project outcome and reinforced the importance of being adaptable.

Epilogue

Completing this project has been both an extremely challenging and rewarding journey. The overall objective was to create an application that combines physical activity with local historical education to improve users' mental well-being. I am very pleased to reflect that the project has successfully met all of its primary goals and functional requirements, such as real-time geolocation, quest tracking, and an intuitive user interface. Additionally, most non-functional requirements, like security and usability, were achieved, though some (such as providing offline quest completion) remain areas to improve.

This project helped me grow in both technical and personal capacities. Despite the challenges I have faced, I have successfully delivered a functional and meaningful solution that I can honestly say I am proud of. Moving forward, I feel confident in my ability to further learn independently, manage complex projects, and critically evaluate my work in an honest manner to ensure continuous improvement.

References

Copyright Notice: Duration of Copyright Term (no date) GOV.UK. Available at: https://www.gov.uk/government/publications/copyright-notice-duration-of-copyright-term/copyright-notice-duration-of-copyright-term (Accessed: 11 September 2024).

Dashwave (2023) *Android architecture patterns - MVC, MVP, MVVM, MVI, Clean Architecture*, *Medium*. Available at: https://medium.com/droidblogs/android-architecture-patterns-mvc-mvp-mvvm-mvi-clean-architecture-cde8029b8f37 (Accessed: 11 September 2024).

Dovhal, M. (2023) Mapbox, OpenStreetMap or Google Map - supply your business with Top Mapping API, SolidBrain. Available at: https://solidbrain.net/blog/openstreetmap-api-vs-mapbox-vs-google-map-chose-your-map (Accessed: 25 September 2024).

Drumond, C. (no date) What is Scrum?, Atlassian. Available at: https://www.atlassian.com/agile/scrum (Accessed: 11 September 2024).

Dutson, P. (2016) *Android Development Patterns: Best practices for developers*. O'Reilly.

Ergün, N., Özkan, Z. and Griffiths, M.D. (2023) 'Social Media Addiction and Poor Mental Health: Examining the mediating roles of internet addiction and phubbing', *Psychological Reports* [Preprint].

Fused Location Provider API | Google For Developers (no date) Google. Available at: https://developers.google.com/location-context/fused-location-provider (Accessed: 11 September 2024).

García, F.J.H. (2023) *Java rest api frameworks - DZone*. Available at: https://dzone.com/articles/java-rest-api-frameworks-1 (Accessed: 11 September 2024).

Gazzah, R. (2020) *MVI architecture with Android*, *Medium*. Available at: https://medium.com/swlh/mvi-architecture-with-android-fcde123e3c4a (Accessed: 11 September 2024).

Goldberg, S.B. *et al.* (2022) 'Social Media Addiction and Poor Mental Health: Examining the mediating roles of internet addiction and phubbing', *Psychological Reports* [Preprint].

Harris, M.A. (2018) *The relationship between physical inactivity and mental wellbeing: Findings from a gamification-based community-wide physical activity intervention*, *PubMed Central*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5774736/ (Accessed: 11 September 2024).

Health effects of staying at Home Too Much (no date) *Vinmec International Hospital*. Available at: https://www.vinmec.com/eng/article/health-effects-of-staying-at-home-too-much-en (Accessed: 11 September 2024).

Hedegaard, H. (2018) *U.S. Centres for Disease Control and Prevention*. Available at: https://www.cdc.gov/nchs/data/databriefs/db309.pdf (Accessed: 11 September 2024).

Humber , N. (2023) *The health benefits of walking*. Available at: https://www.bupa.co.uk/newsroom/ourviews/walking-health (Accessed: 11 September 2024).

LocationManager | Android Developers (no date) Android Developers. Available at: https://developer.android.com/reference/android/location/LocationManager (Accessed: 11 September 2024).

Lamborn, J. (2022) Agile explained: The 4 Agile Manifesto Values and 12 principles, LogRocket Blog. Available at: https://blog.logrocket.com/product-management/agile-manifesto-4-values-12-principles-explained (Accessed: 11 September 2024).

Munusamy, B. (2023) *Accessing User's Location Guide Android 2023*, *Medium*. Available at: https://medium.com/@boobalaninfo/accessing-users-location-guide-android-2023-60a6f018a718 (Accessed: 11 September 2024).

Musgrave, Z. (2024) *Quoting Differences Between Mysql and PostgreSQL, and Converting Between Them*, *DoltHub Blog*. Available at: https://www.dolthub.com/blog/2024-07-09-mysql-postgres-quoting/ (Accessed: 11 September 2024).

MVVM (Model View Viewmodel) architecture pattern in Android (2022) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/ (Accessed: 11 September 2024).

PostgreSQL vs mysql - difference between relational database management systems (RDBMS) - AWS (no date) AWS Documentation. Available at: https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/ (Accessed: 23 September 2024).

Rehkopf, M. (no date) Scrum sprints: Everything you need to know, Atlassian. Available at: https://www.atlassian.com/agile/scrum/sprints (Accessed: 11 September 2024).

Samrgandi, N. (2021) *User Interface Design & Evaluation of Mobile Applications*, *ResearchGate*. Available at: https://www.researchgate.net/publication/349087972_User_Interface_Design_Evaluation_of_Mobile_Applications (Accessed: 11 September 2024).

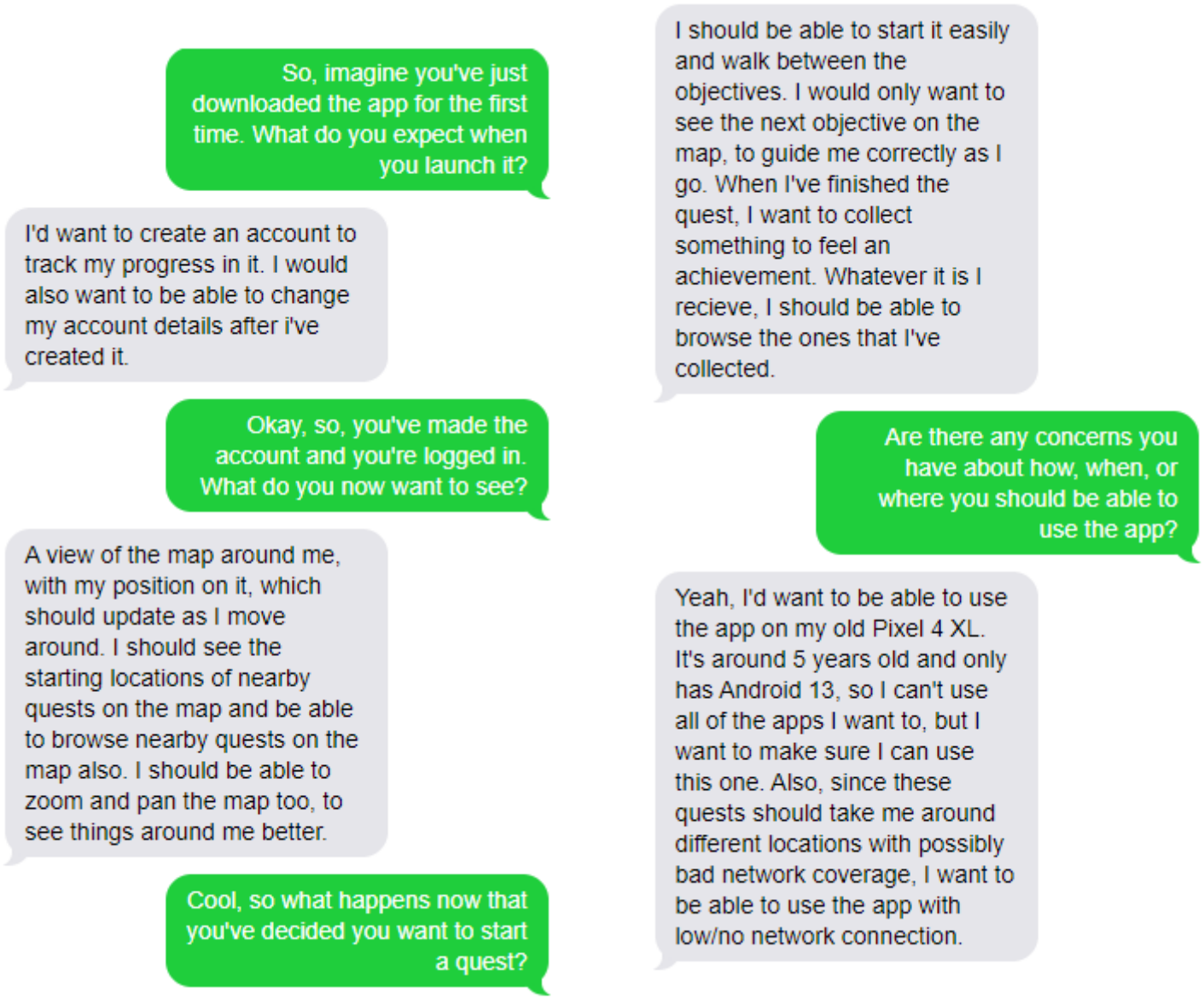
Varanasi, B. and Belida, S. (2015) Spring Rest. Apress.

What is Information Security: Policy, Principles & Threats: Imperva (no date) Imperva. Available at: https://www.imperva.com/learn/data-security/information-security-infosec/ (Accessed: 11 September 2024).

Yasas, R. (2022) *Spring Boot Best Practices for developers*, *Medium*. Available at: https://medium.com/@raviyasas/spring-boot-best-practices-for-developers-3f3bdffa0090 (Accessed: 11 September 2024).

Appendices

Appendix A:
A visual, approximate representation of my discussion with users, to craft user stories and elicit requirements.



Appendix B:
The form which my users have signed and dated to provide consent for participation in this project.

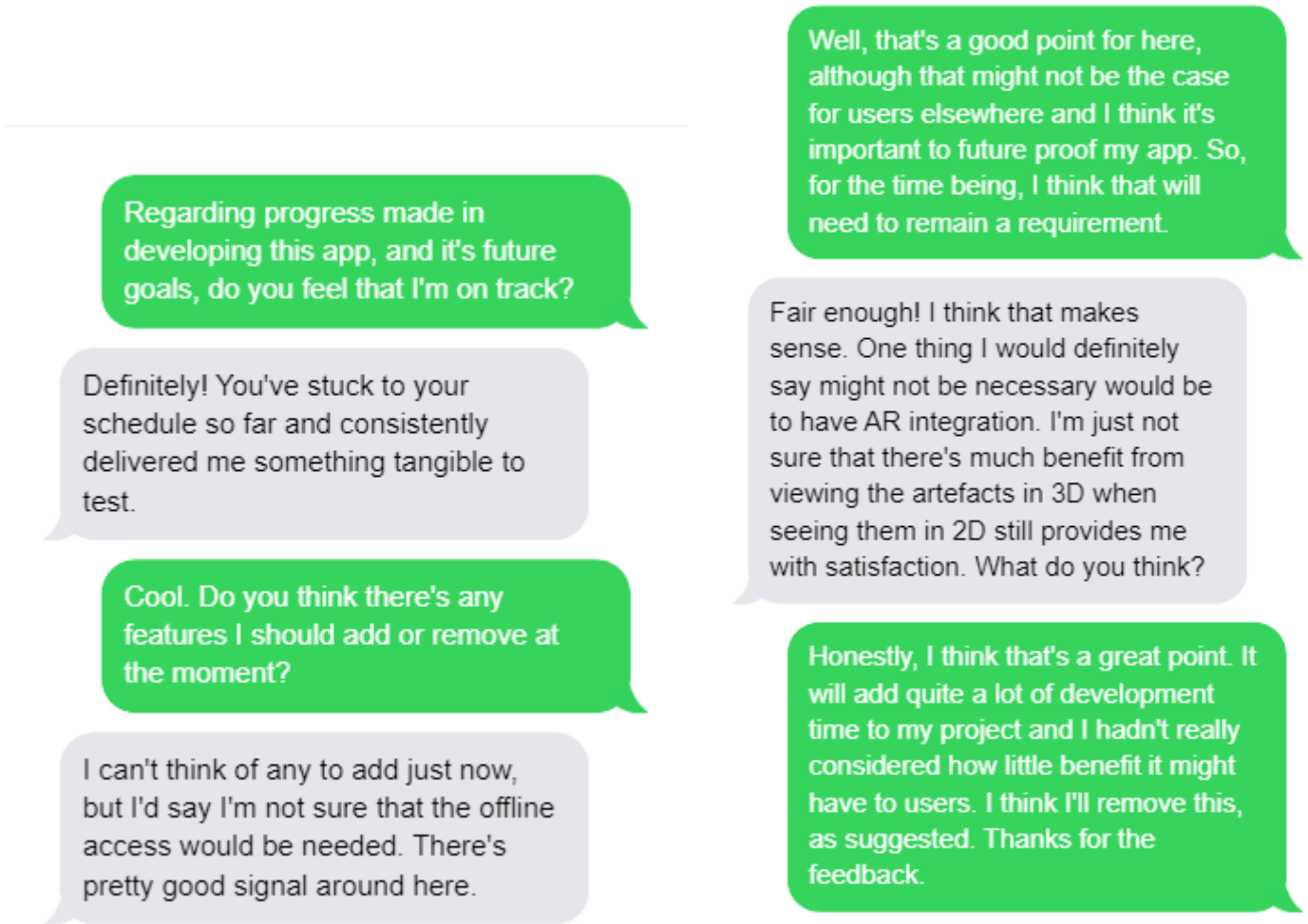
HistoryHike - LSEPI Informed Consent User Statement

I, _____, voluntarily agree to provide feedback for the development of the HistoryHike application. I understand the feedback will:

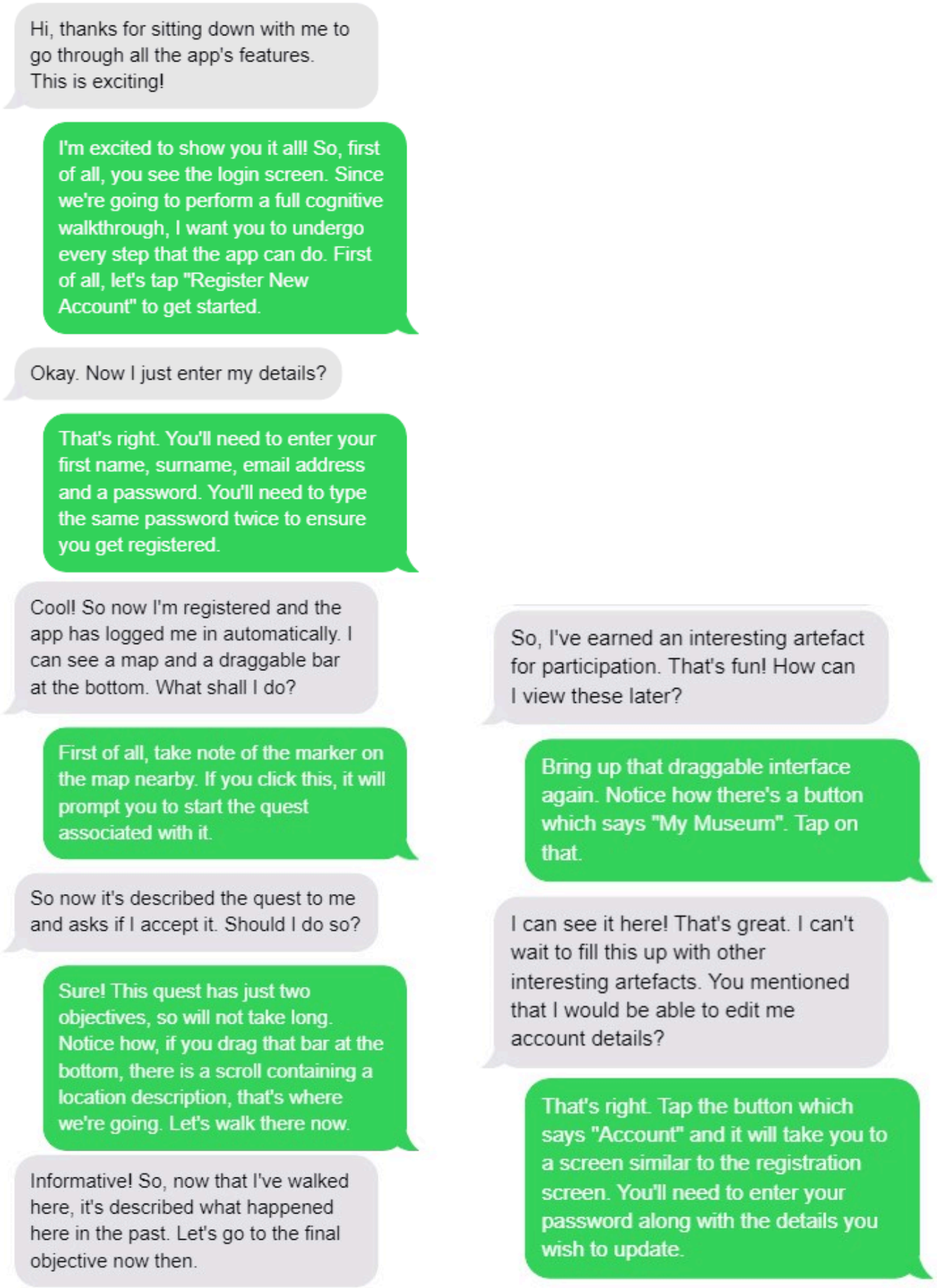
Improve the app's functionality, usability, and overall user experience.
Contribute to Alex's University project assessment.
I understand my personal details will remain confidential, and only my comments regarding the app will be used. I can withdraw my consent at any time, and any data collected from me will be permanently deleted upon request.

Signature: _____
Date: _____

Appendix C:
A visual, approximate representation of a discussion with users early in the development process, whereby the project's scope was narrowed following constructive feedback.



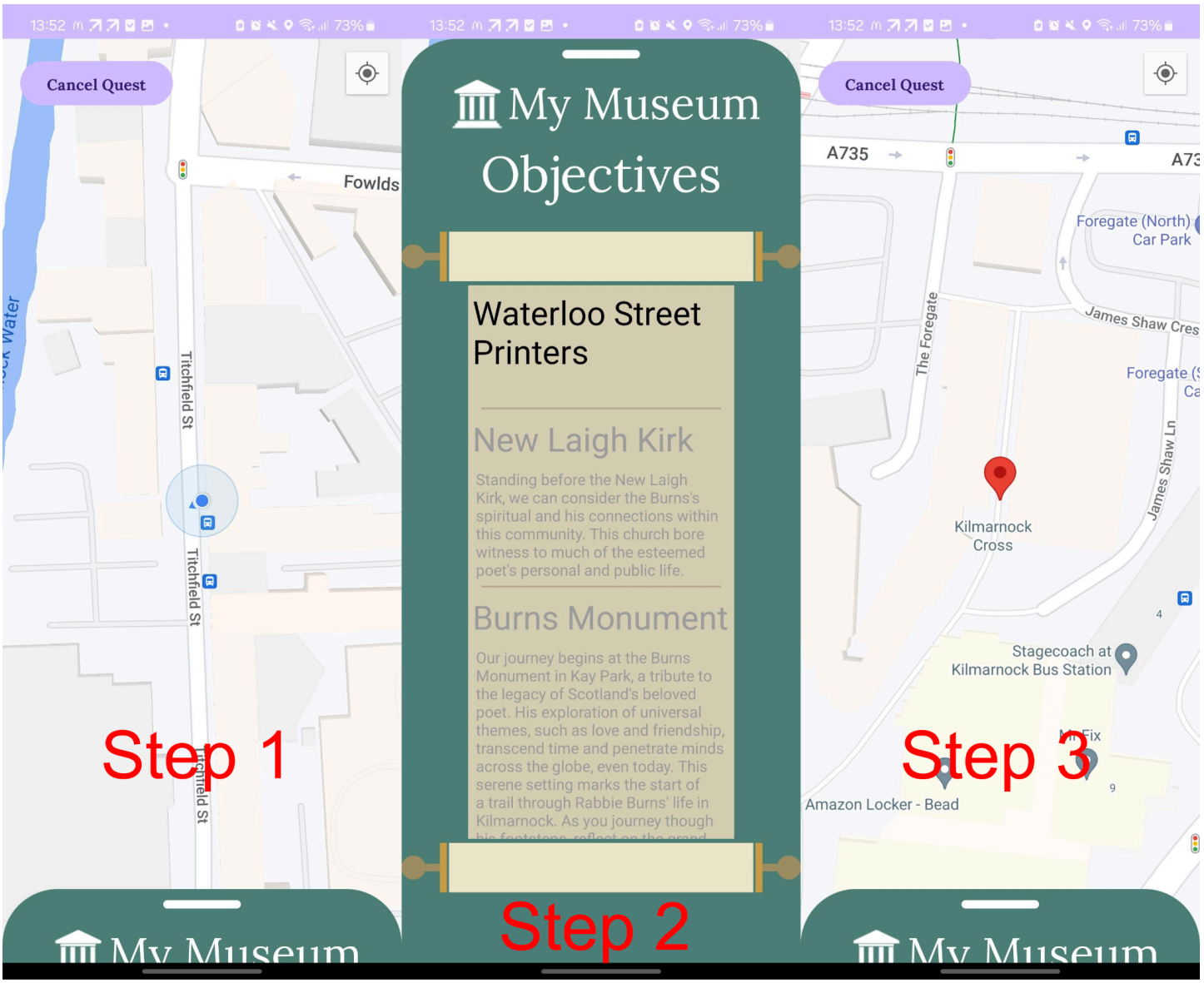
Appendix D:
A visual representation of the cognitive walkthrough performed with my userbase to acquaint them to my app's features.



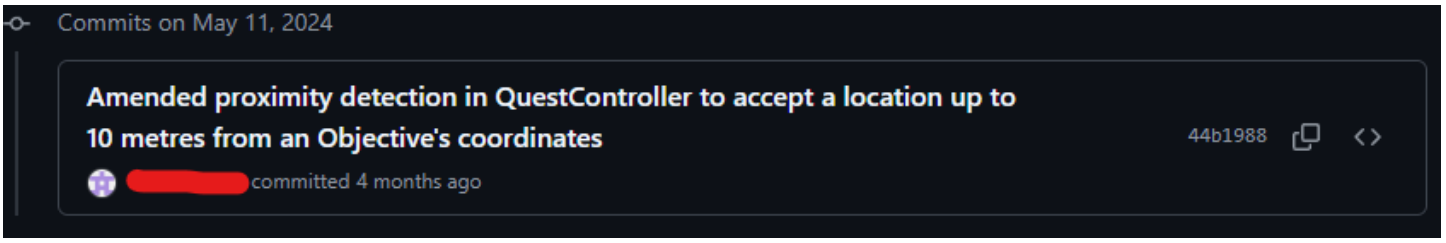
Appendix E:
The final comments regarding the outcome of HistoryHike usage.

Feedback Area	Positive Comments	Constructive Feedback
Encouragement to Exercise	Helped me stay active regularly.	Notifications/reminders to take part in quests would be useful, to encourage further.
	Learning while exercising was a unique approach which I enjoyed	The quests could be more varied in length to fit different time slots.
Knowledge of Local History	I learned interesting facts about the history of my surroundings.	Some historical quests were too brief and I wanted to continue them.
Mental Health Impact	I felt grounded through learning about my own surroundings.	More quests with more variance would enhance this feeling. Currently there are not enough quests to keep me coming back.
	I felt less of a desire to partake in unhealthy screen habits (social media, for example) during and after HistoryHike use.	I still use social media too much outside of HistoryHike usage. Again, notifications would be helpful to encourage habit replacement further.
	I felt a sense of achievement after completing a quest.	While it did help with my exercise routine, I'm unsure of its long-term mental health benefits.
	I felt satisfied that I had made progress when viewing my collected artefacts.	Integration of other mental health features would be useful.
User Interface	Easy to navigate all aspects of the app.	It could be more visually engaging, especially the artefact display.

Appendix F:
A screenshot evidencing the implementation of the requested functionality for a user's map to centre on the next objective within a quest when it is tapped from the current objective list. Step 1 shows the user's location, Step 2 shows the objectives list and Step 3 shows what the user then sees after clicking the upcoming (bold) objective.



Appendix G:
The Git commit where Geolocation distance was increased five-fold.



Appendix H:
An example of my Sprint notes, which were maintained throughout the project.

Sprint 1		Progress Notes	User Feedback
Start Date	16/02/2024	All tasks completed within schedule	Design seems cluttered
End Date	1/3/2024	Possible issues with next sprint's integration of Google Maps API. Needs to be top priority to ensure no wasted time.	Icons and text on museum collection screen are too large. Take up too much space. Congitive walkthroughs might be necessary, or at least beneficial.

Sprint 2		Progress Notes	User Feedback
Start Date	1/4/2024	Creating controller methods took 1 day longer	Implement a method of centering the map back onto the user, if scrolled away
End Date	12/4/2024	Implementing and passing comprehensive unit testing took only 8 days, instead of the planned 12 Using spare time to work on other module's schedules, will contine default Sprint length this time	Review AR capabilities. Drop it??