Buckinghamshire New University

PROJECT

The research, design and development of the complete software lifecycle of an android mobile application with the aim of connecting users with the history of famous monuments.

by

Felix A. Monteiro Saraiva

Presented to the

School of Business, Law and Computing

BSc (Hons) Software Engineering

Supervised by

Michael Everett & Guy Walker

2020

School of Business, Law and Computing

CW3 Final Report:
Academic Year 2019 to 2020

Module Title:	Project	**Module Code:**	CO699
Assignment No/Title:	Final Report (CW3)	**Assessment Weighting:**	60%
Submission Date:	Thursday 30th April 2020 Semester 2 Week 10 by 14:00	**Feedback Date:**	+ 3 Weeks
Module Tutor:	Justin Luker	**Degree:**	Degree
Student ID:	**21713532**	**Student Name:**	**Felix Alexandre Monteiro Saraiva**
1st Supervisor:	**Mike Everett**	**2nd Supervisor:**	**Guy Walker**
Course:	BSc. (Hons) Software Engineering		
Word Count:	**10950**		

Project Title

The research, design and development of a Mobile Android Application that provides a complete interaction between users and the most famous, beautiful and historical monuments around the world.

Acknowledgments

I wish to express my sincere appreciation to my supervisor, Professor Michael Everett, who has the substance of a genius: it is whole-heartedly appreciated that your great advice for my project proved monumental towards the success of this study. Without your persistent help, the goal of this project would not have been realized. Thank you, Mike!

Also, I wish to acknowledge the support and the great love of my family, my mother, Dulce; my father Manuel; and my sister Claudia.

Last, but certainly not least, I wish to express my deepest gratitude to my godfather Nuno, and my brother in law João. People usually say that we never really become our idols, but I sure hope that one day I can inspire someone like you two inspired me!

Abstract

This project represents the research, design and development of the complete software lifecycle of an android mobile application that connects users with the history of famous monuments, providing several interactions between them.

Such a system was developed by the application of a custom hybrid development methodology, Scrumban, and designed via UML notation inspired by a mock-up prototype.

This application is developed incrementally, each increment tested using black and white box techniques and implemented by a version control software, GitHub, that demonstrates the flow of development within a Scrumban Board.

Table of Contents:

Table of Figures:

Introduction

Is it possible to evolve as a civilisation without learning today, from the lessons of yesterday?

Several historians believe that the study of human history is a way to view “the past as a window to the future” so, what is the monument’s role in a society’s evolution?

Overall, monuments reflect the great things human beings can build together, when that happens, a mark is left on the society, that together they can achieve what was once ‘impossible’ by themselves. According to the National Geographic contributor Andrew Howley- “Perhaps building big things is simply something people do when they begin to work together”. (Howley, 2013)

Nowadays, one of the world’s most magnificent monument, modern technology, is putting those big rock structures in the shadows, while new technology discoveries shine every day.

This Project’s objective is to bring both the knowledge of yesterday and the technology of today and allow the users to learn more about monuments, that have a vast historical reputation throughout nations.

Using the compelling advantages of mobile applications, this Project focuses users that have an interest in learning more about monuments and encourages visits to the respective locations. Presenting the monuments in an organised way, by their respective country and leading the users throughout the monument’s history, characteristics and curiosities and even providing the possibility of saving memories, of their travels, to such amazing places.

The entire Project will be engineered since its start, using diverse software engineering skills. An agile methodology will be followed, respecting the software lifecycle, controlling both version and time, using plans that organise iterations and help managing change and improvements. Existing applications that have a similar idea will be analysed and revised in order to learn with current market mistakes or adopt good practices for this Project. (Larson, 2019)

A challenging project, surrounded by historical knowledge, and software engineering skills that are present alongside the research and design of the application. Such research, which encapsulates consideration, originality, contribution and knowledge. (Dawson, 2015)

Project Background

Every traveller gets curious about the story behind a monument they visit, and many must 'google' why was that beautiful, artistic place, built in the first place. Several end up searching through massive 'Wikipedia' text pages, or long history detailed websites, while all they desired was a fast and interactive way to know more about the crucial facts that made past civilizations build those structures.

Thanks to the academical approach in the 'Mobile Apps' module, using a mobile method to solve the problem above seemed the right choice, the first step to the decision of taking the knowledge and experience obtained during the module further throughout this project. This way, this project can target a large number of users and allow them to take this product with them on their travels.

However, like all 'new-born' projects, with an idea comes doubt, and in order to validate this idea, to be a feasible one, more in-depth investigation is required.

This research led to a large study that started on what mobile platform could be adequate for this project, followed by the creation of mobile-catalogues, interactive mapping tools, best databases for mobile platforms (cloud vs real-time), back-end APIs, many software developing books and articles related with the best methodologies for this approach, all to ensure that a trustworthy product could be shaped from the initial vision. Also, a very critical research around current market applications, with similar objectives, took place, gathering some of the pros and cons, that help on the requirements engineering process.

With both the idea and the research around it, it is possible to validate this project's feasibility and with that validation comes the opportunity to put in practice the knowledge acquired during this course, since the quality of a mobile application depends on skills like design, planning, coding, problem-solving and many other software engineering skills.

Project Rationale

The rationale of every project takes a vital role in the aim of the respective end product, therefore influences the development of a fully operational android application that can fulfil its requirements. (Dawson, 2015)

It is present an academical wish to improve coding, design and software planning skills in a professional level, around the android environment since a “positive taste” was aroused in the “Mobile Apps” module. Due to that, the decision of taking this development to the “Android Studio” IDE was an easy choice, along with the preference of using Java instead of, for example, Kotlin.

Although ‘Google’ announced that Kotlin is now its preferred language for Android app developers, the personal experience obtained in the past with Java, undoubtedly improves the quality of this project. (Lardinois, 2019)

The rationale research exposed that applications in the present market share a similar idea to this one. The “World Explorer-Travel Guide” and” minube” applications, proved to be two good examples, in comparison to this project, taking a closer look on “minube”. A very similar idea with a very positive user interface design, but low on performance and not a practical approach to its objective. (minube, 2019) The exact opposite of “World Explorer”, which the ‘Literature Survey’ analyses in more depth.

One of this project’s goals is to capture what is positive on those two examples and create a very interactive but also practical application, therefore increasing the market quality and potential of the software.

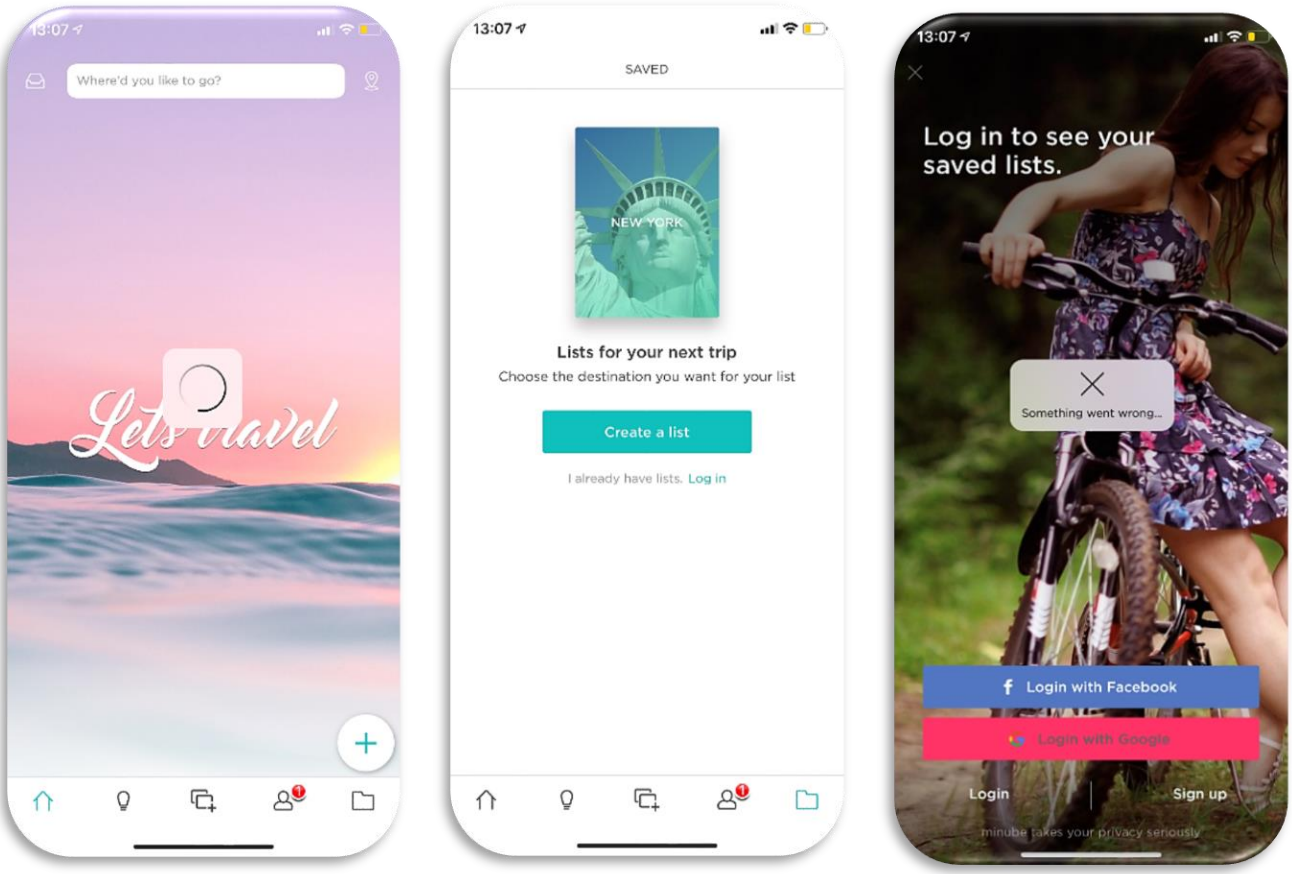


Figure 1- ‘"minube" Application’- Very Bugged Interface

Project Ethical Considerations

This research applies honesty to the presented methods, data and results.

Objectivity is present to avoid bias in any aspect of the research, including design, data analysis and interpretation. This research is reviewed carefully and critically to ensure that the results are credible. This Project works openly to assure that the same is prepared to share the data and results, along with any new tools that the research has developed. Respect for Intellectual Property will be a significant concern. It will always present correct references. Anything that has been provided in confidence will be respected.

The Project will follow guidelines on the protection of sensitive information such as the General Data Protection Regulation (GDPR). The awareness of laws and regulations that govern this work will always be respected.

The Project will ensure the quality and integrity of the research, will ensure that participants will participate in this study voluntarily and will avoid harm to participants and show that the research is independent and Impartial. (CO699 Project Ethical Considerations Notes, 2018 to 2019)

Project Aim

This project's aim focuses on the development of a mobile application, established by an interactive and user-friendly interface, that works as a bridge between the knowledge of yesterday and the technology of today, providing users with a connection between them and the monuments they visit or wish to visit, in their travels. Have a populated and prearranged database, that works as an independent back-end, and be able to visualize future market improvements to this idea.

This desirable outcome consists of fully operational software that achieves an agile lifecycle method throughout the entire project, completing as many sprints as possible, underpinned via UML notation, implementing the selected requirements.

Project Objectives

1) Feasibility Study	Determination if the idea is appropriate for further development.
a) Feasibility Research	Research around mobile development, front end and back end;
b) Feasibility Report	The collection of information needed to develop this project, '*Literature Survey*'.
2) Requirements Engineering Process	Definition of the service provided by the system.
a) Requirements Elicitation	Questionnaires and interviews focusing the product's objective;
b) Requirements Analysis	Analysing the data collection methods, to extract the system's requirements;
c) Requirements Validation	Validation of the requirements extracted;
d) Requirements Management	Prioritization of the system's requirements in the '*Product Backlog*';
e) System Models	Abstract visualization of the requirements.
3) Interface Design	Creation of designs, that users find pleasurable and easy to use.
a) Sub-system's Interfaces	Description of the sub-system's Interfaces.
4) Prototype Development	Design Verification phase of the Product Development & Interface Design.
a) Mock-up prototype	Creation of a mock-up prototype;
b) Validation of Sub-system's Interfaces	Providing a beta testing interaction, to random users (validation of Objective 3).

5) Component Design	Structuring the system by software components.
a) Components Creation	Decomposition of sub-systems into Increments;
6) Data Structure Design	The organization of data according to a database model.
a) Database Specification	Specification and visualization of Database Structure;
7) Lifecycle Method	Process of planning, creating, testing, and deploying the system.
a) Lifecycle method	Choice of a suitable lifecycle methodology;
b) Establishing Increments	i) Analyse/Planning ii) Design iii) Development iv) Implementation v) Testing vi) Deliver/Assess
8) Project Conclusion	Ensure that the approaches, techniques, methods and processes designed in the project are correct, and analyses future improvements.
a) Quality Control	i) Inspections for defect; ii) Reviews for progress;
b) Critical Evaluation	Evaluation of the final work, comparing it to the objectives and concluding whether, or not, the project has met the overall aim;
c) Future Improvements	After the 'Critical Evaluation', look for different ways of changing those negative outcomes, and share a future vision of this project.

Risks

- Risks are an uncertain event or condition that can cause a potential problem, the level of each risk is measured in terms of likelihood and impact. (Jackson, 2020)

Two Major Risks Categories:

1. **Project Risks-**Risks that surround the project's capability to deliver;
2. **Product Risks-**Risks in software or system failure.

Different Types of Risks:

1)	Safety Critical Risks
2)	Political Risks
3)	Economic Risks
4)	Technical Risks
5)	Organizational Risks
6)	Security Risks

(Jackson, 2020)

1. Project Risks

RISK ID	RISK TYPE	RISK DESCRIPTION	RISK SEVERITY	RISK LIKELIHOOD	INTERNAL IMPACT	USER IMPACT
RJK.1	Organizational	Developer lacks critical resources.	**High**	**Low**	**Extreme**	**High**
RJK.2	Technical	Poor quality in the design documentation.	**Medium**	**Low**	**High**	**High**
RJK.3	Technical	Requirements are poorly written.	**High**	**Medium**	**High**	**High**
RJK.4	Organizational	The time per increment is not well planned.	**Low**	**Medium**	**Medium**	**Medium**
RJK.5	Technical	Developer lacks technical knowledge in the required programming language.	**Extreme**	**Low**	**Extreme**	**Extreme**

2. Product Risks

REF/ID	RISK TYPE	RISK DESCRIPTION	RISK SEVERITY	RISK LIKELIHOOD	INTERNAL IMPACT	USER IMPACT
RDK.1	Technical	Delivering software that does not meet the requirements.	**Extreme**	**Medium**	**Medium**	**Extreme**
RDK.2	Safety-critical	Poor data integrity and security.	**High**	**Low**	**Low**	**Extreme**
RDK.3	Technical	Did not satisfy non-functional requirements.	**Extreme**	**Medium**	**Extreme**	**Extreme**
RDK.4	Political	Potentially harmful to an individual or company.	**Hight**	**Low**	**Low**	**High**
RDK.5	Technical	Poor functionality.	**Extreme**	**Medium**	**High**	**Extreme**

Project Literature Survey

Introduction

Several kinds of literature on this field of study took place to sustain the quality and truthfulness of this project.

With the purpose of providing a starting point, to the development of this project, the literature survey represents the understanding, critical evaluation, conceptualisation and presentation of the knowledge needed, to complete the previous objectives and respective aim. (Dawson, 2015)

In order to raise the quality of this project, the literature survey is divided by a '*Primary Research',* responsible for the study of core methods and techniques, required to the creation of this system, and a '*Secondary Research',* a collection of studies, focused on seeking the best designs and tools to increase the success of the objectives.

1. Primary Research:

1.1. The Current Market

Starting with the absence of the intended final objective, of this project, on existing applications. The best comparable mobile application found, in the current market, is called “World Explorer-Travel Guide” by *Tasmanic Editions*, a tour guide application that sorts the best places to visit, with a database of more than 350 000 rated points of interest, geolocation for instant info on what is nearby and convenient search for almost everywhere in the World. A very positive application for the intended purpose and very similar to the objective of this project. (Editions, 2019)

Although this is an efficient application, it loses in the user interaction, since it does not focus itself on the user’s opinion, as the points of interest are standardly rated, and the only interaction is reading the complete information about a monument in a “Wikipedia” article. In consequence of not keeping the user’s opinion in consideration, many negative reviews have been presented by the public, regarding the poor interface design and the massive “Wikipedia” articles.

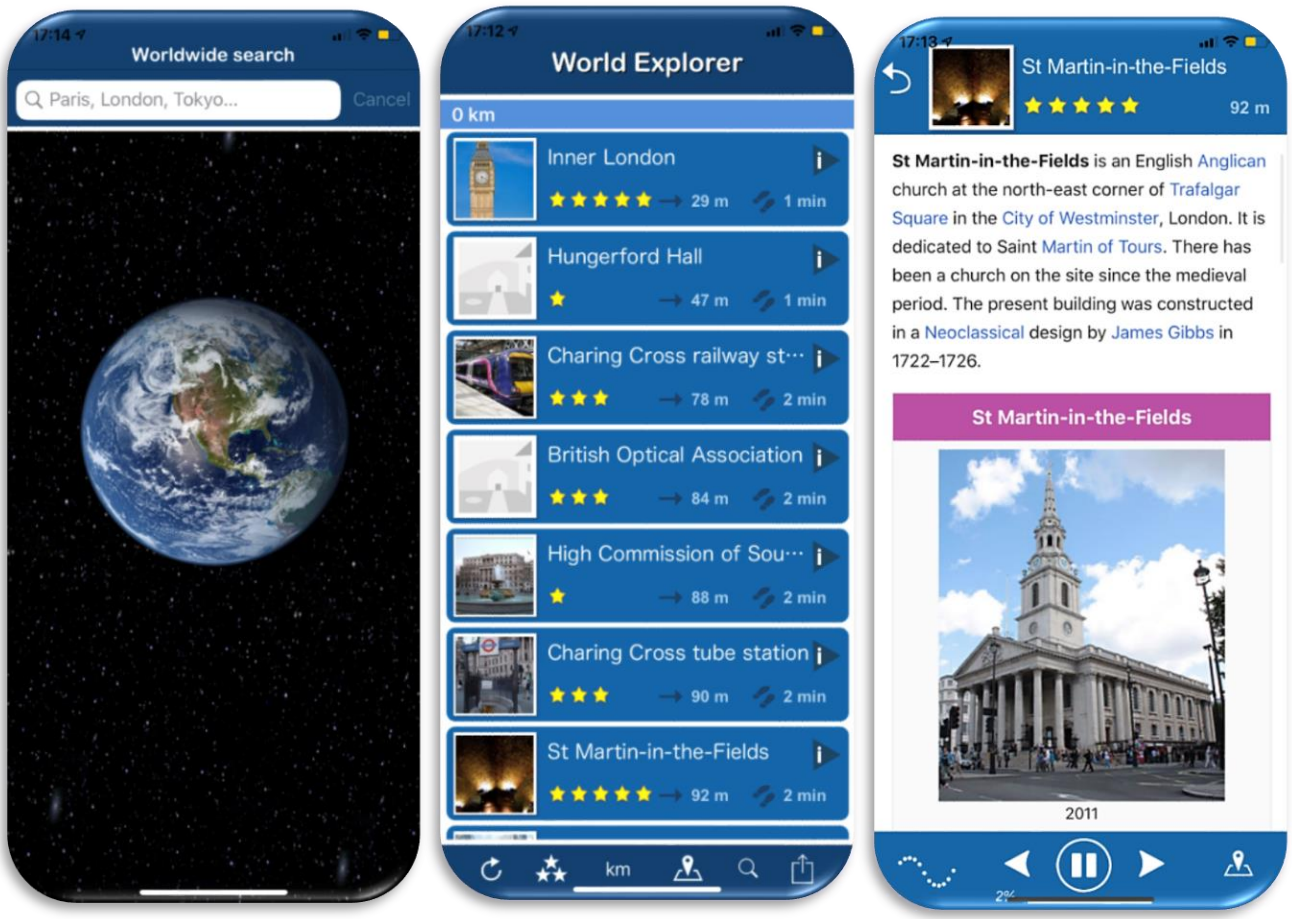


Figure 2-World Explorer Application

1.2. Research Methodologies

Research is defined as the creation of new knowledge, and the use of existing knowledge in a new and creative way to generate new concepts, methodologies and understandings. (shodhganga, Consulted at 30/10/2019)

In this Project, both qualitative and quantitative research methodologies are going to take place, as the primary data collection methods, deeply analysed in the '*Methodology'* section.

Data Collection Methods:

Data collection is a process of collecting information from all the relevant sources to find answers to the research problem, test the hypothesis and evaluate the outcomes.

Data collection methods can be divided into two categories:

1. **Primary data collection methods** can be divided into two groups: quantitative and qualitative.
   a. **Quantitative methods** of data collection and analysis include questionnaires with closed-ended questions, methods of correlation and regression and others.

   b. **Qualitative methods** aim to ensure greater level of depth of understanding and qualitative data collection methods include interviews, questionnaires with open-ended questions, focus groups, observation, game or role-playing, case studies etc. (Research Methodology, 2019)

2. **Secondary data** is a type of data that has already been published in books, newspapers, magazines, journals, online portals etc, corresponding to the research done in the '*Literature Survey'* section of this project.

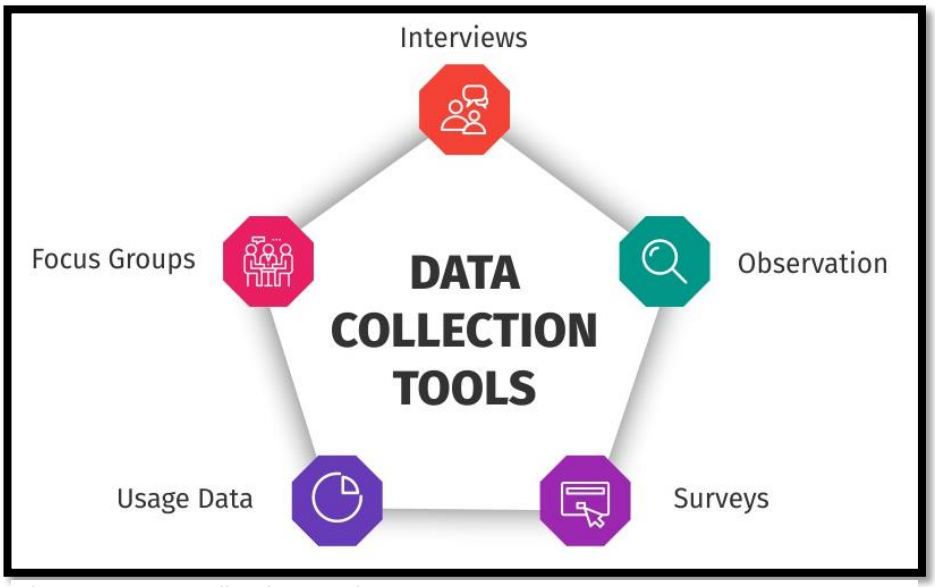


Figure 3-Data Collection Tools
(Bhat, How to Use Data Collection Tools for Market Research, 2019)

1.3. Project Management Techniques

Project management techniques are sets of guiding principles and processes for managing a project. Choosing the right management technique defines the quality of the project's development and how the communication, inside the development team, flows.

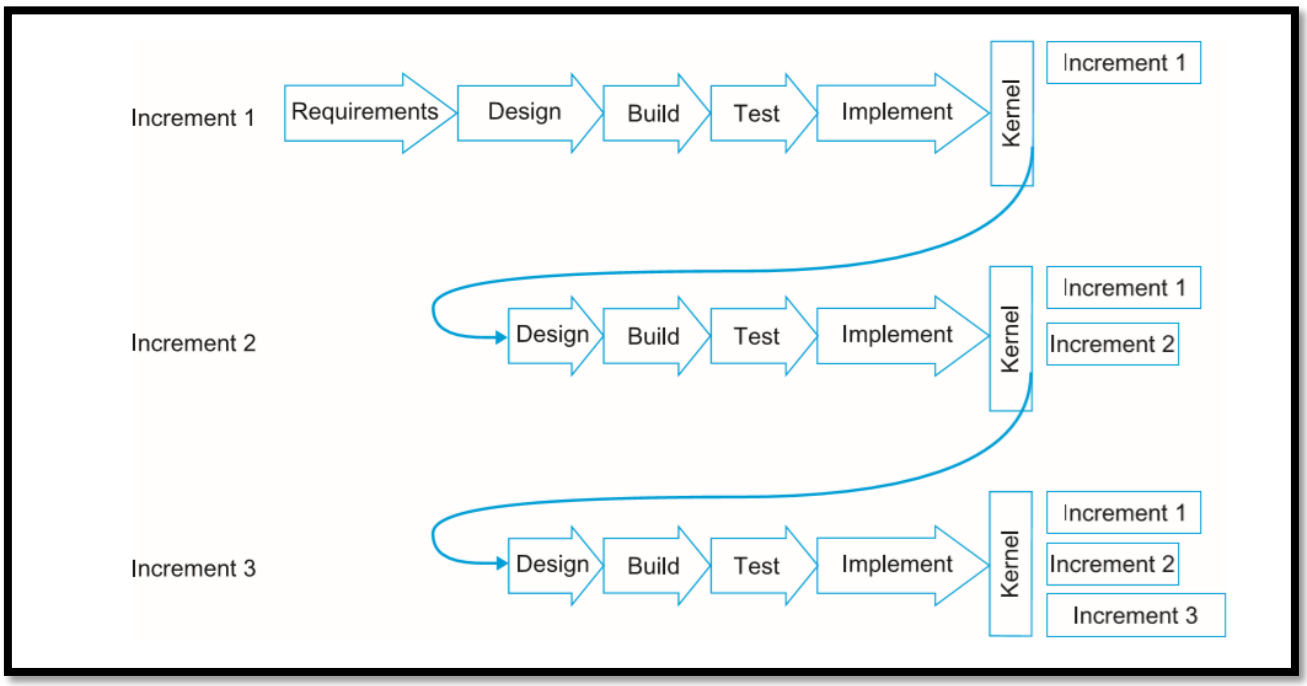
Types of Project Management Methodologies:

Waterfall	A sequential approach, heavily focused on the requirements. A clear idea of the final product is needed since there is no scope for correction once the project is underway.
Agile	Opposite of 'Waterfall', favours a fast and flexible approach, it is integrative with small incremental changes, that respond to requirement changing.
Lean	Focus itself on doing more with less, starts by identifying value and then maximizes it through continuous improvement, by optimizing the flow of value and eliminating wastage.
Extreme Programming	Focus on rules such as including user stories, test-driven development, Pair programming, and continuous integration among many others.
Prince 2	Focus on documentation and experience, in order to reduce risk. This management method, very famous in the UK, is best suited for large and complex projects, with fixed requirements.
Critical Chain	Works backwards from the end goal. This method uses past experience to map the tasks required to complete the project.
PRISM	This method extends itself beyond the end of the project. A suitable approach for projects where environmental cost and sustainability are key success criteria.

(Cohen, 2019) (Aston, 2019)

Evaluation methods to pick the right methodology:

1.	Evaluate the Project
2.	Evaluate the Team
3.	Evaluate the Organization
4.	Evaluate the Stakeholders
5.	Evaluate the Tools



Agile Incremental Methodology

Rather than delivering the software to a client at the end of the project in one 'big bang' as the conventional models do, it might be better to deliver this system as a series of intermediate working sub-systems over a period of time. Each release to the user provides added functionality to the existing system. (Dawson, 2015)

This incremental model begins by gathering the requirements of the system at the first incremental release, as well as the first component of functionality that will be useful to the client. Then other increments can take place after the first one always following the same structure of stages: Design, Build, Test and Implementation. As we can see in the example below:

Figure 4-The Incremental Model (Dawson, 2015)

Agile vs Plan-Driven Methods

Differently, standard approaches to project management are plan-driven. Managers draw up a plan for the Project showing what should be delivered and when it should be delivered.

A plan-based approach requires a manager to have a stable view of everything that must be developed and the development processes. However, it does not work well with agile methods where the requirements are developed incrementally. (Sommerville, Software Engineering, 2011)

Since this Project's increments must be delivered in a relatively short period, a more agile approach to project management is requested, which adapts to incremental development.

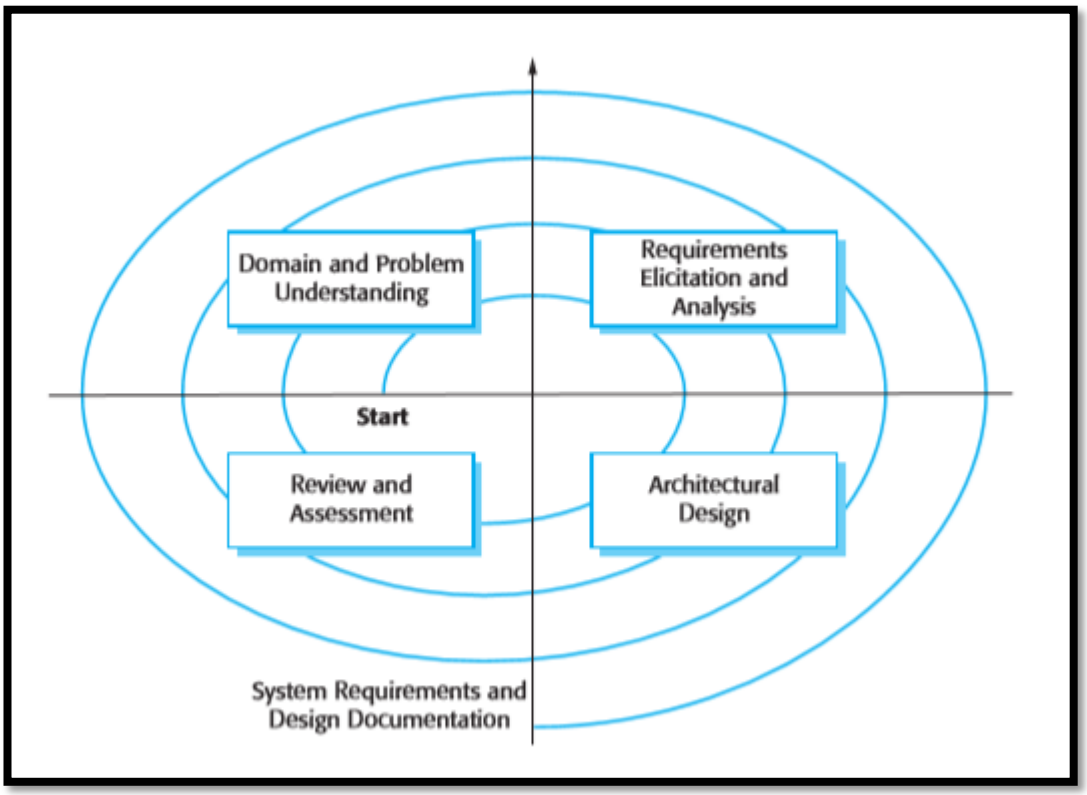


Figure 5-Requirements and Design Spiral (Sommerville, Software Engineering, 2011)

1.4. Project Development Methodologies

The Scrum Methodology

The most commonly used agile approaches such as Scrum have a two-stage approach to planning, corresponding to the start-up phase in plan-driven development and development planning:

1. **Release planning**, which looks ahead for several months and decides on the features that should be included in a release of a system.

2**. Iteration planning**, which has a shorter-term outlook, and focuses on planning the next increment of a system. This is typically 2 to 4 weeks of work for the team. (Sommerville, Software Engineering, 2011)

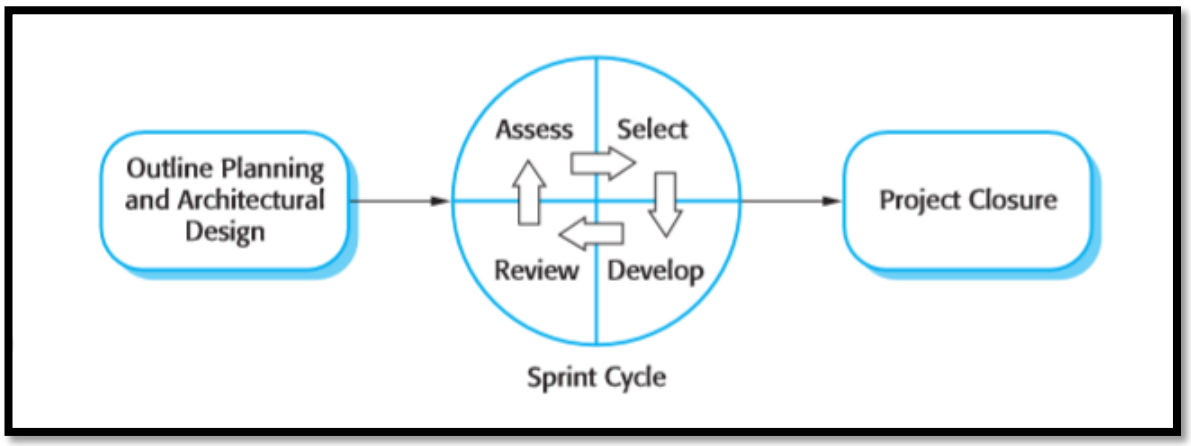


Figure 6-The Scrum Process
(Sommerville, Software Engineering, 2011)

The innovative feature of Scrum is its central phase, named the 'Sprint Cycles'. A Scrum sprint is a planning unit, in which the work to be done is assessed, features are selected for development and the software is implemented. At the end of a sprint, the completed functionality is delivered to the stakeholders.

Scrum Method Stages:

1.	Sprints are fixed length, normally 2–4 weeks;
2.	The starting point for planning is the product backlog, which is the list of work to be done on the project. During the assessment phase of the sprint, this is reviewed, and priorities and risks are assigned. The customer is closely involved in this process and can introduce new requirements or tasks at the beginning of each sprint;
3.	The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint;
4.	Once these are agreed, the team organizes themselves to develop the software. Short daily meetings involving all team members are held to review progress and if necessary, reprioritize work. During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'. The role of the Scrum master is to protect the development team from external distractions. The way in which the work is done depends on the problem and the team;
5.	At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

(Sommerville, Software Engineering, 2011)

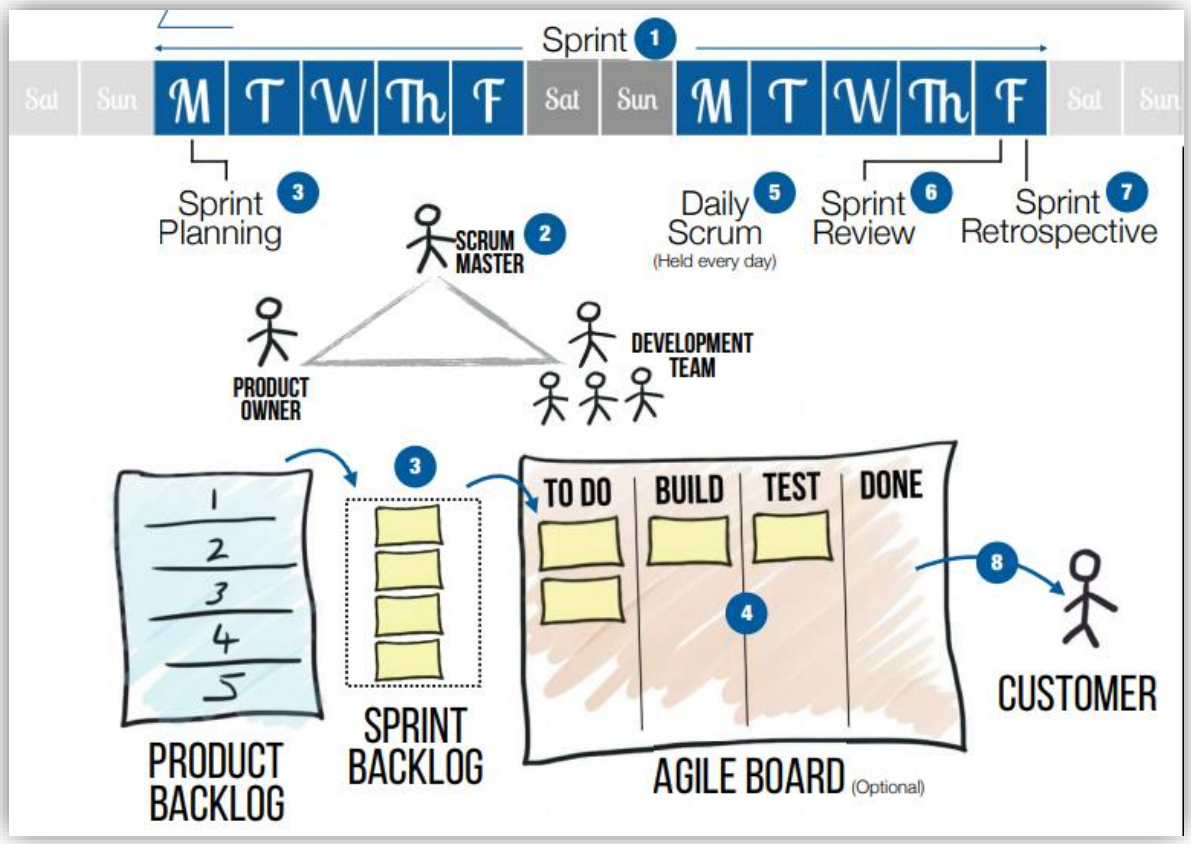


Figure 7-The Scrum Stages (Straughan, 2017)

The Kanban Methodology

Kanban, developed by Taiichi Ohno, an engineer at 'Toyota', is defined as a method for managing and improving service delivery gradually over time.

The Kanban Method is a technique to design, manage, and improve the workflow of a system. The method also allows organizations to start with their existing workflow and drive an evolutionary change. They can do this by visualizing their flow of work, limit work in progress (WIP) and their primary tool the Kanban Board, with the motivation on "stop starting and start finishing".

Kanban can be used in any knowledge work setting, and it is particularly applicable in situations where work arrives unpredictably or when work is deployed as soon as it is ready, rather than waiting for other work items (sprints).

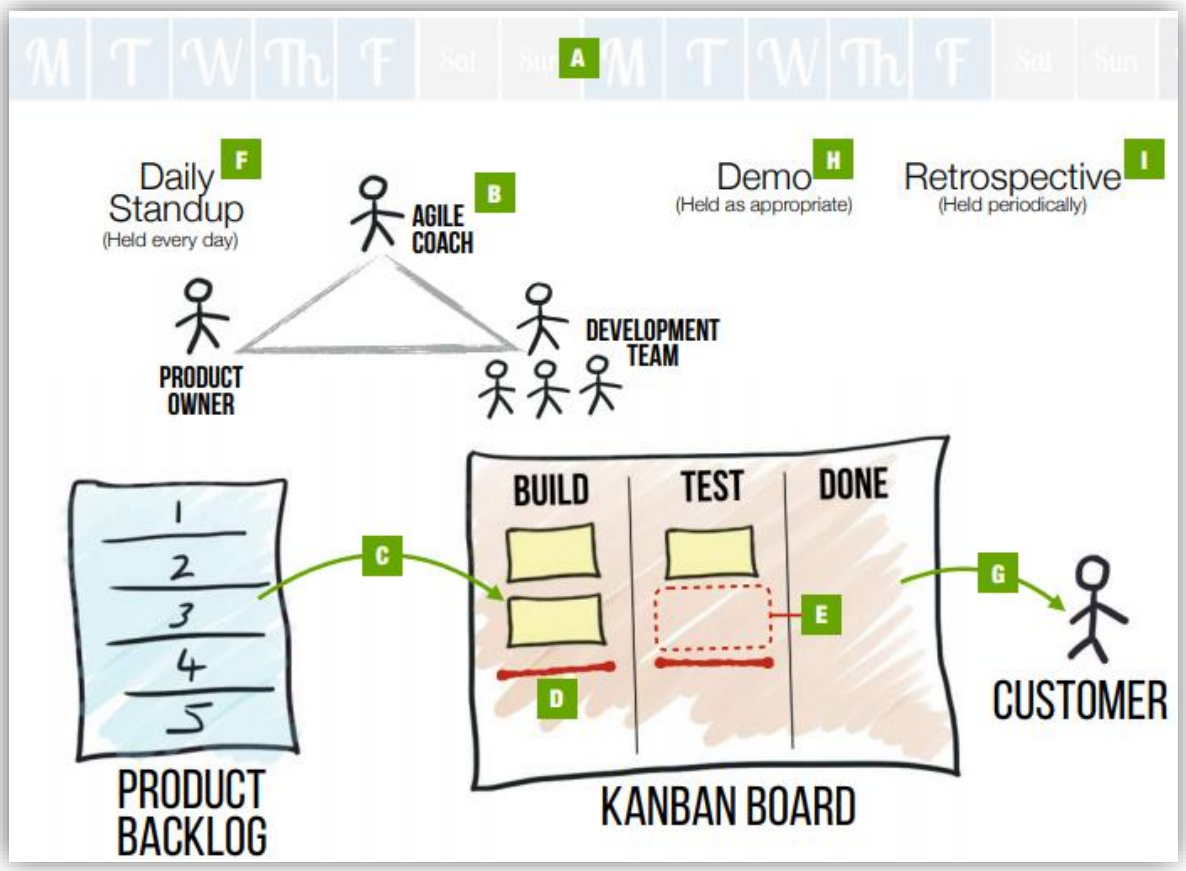


Figure 8--The Kanban Stages (Straughan, 2017)

Practices:

Kanban systems use mechanisms such as the Kanban Board, helpful to visualize work and progress.

For the visualization to be most effective, each Kanban Board, should demonstrate:

1. **Commitment Stage**: Where in the process a team working on a service agrees to do a specific work item;
2. **Delivery Stage**: Where the team delivers the work item to a customer;
3. Policies that determine what work should exist in a particular stage;
4. Management of the workflow;
5. Limit Work in Progress (WIP).

(Agile Alliance, 2019)

Kanban Method Stages:

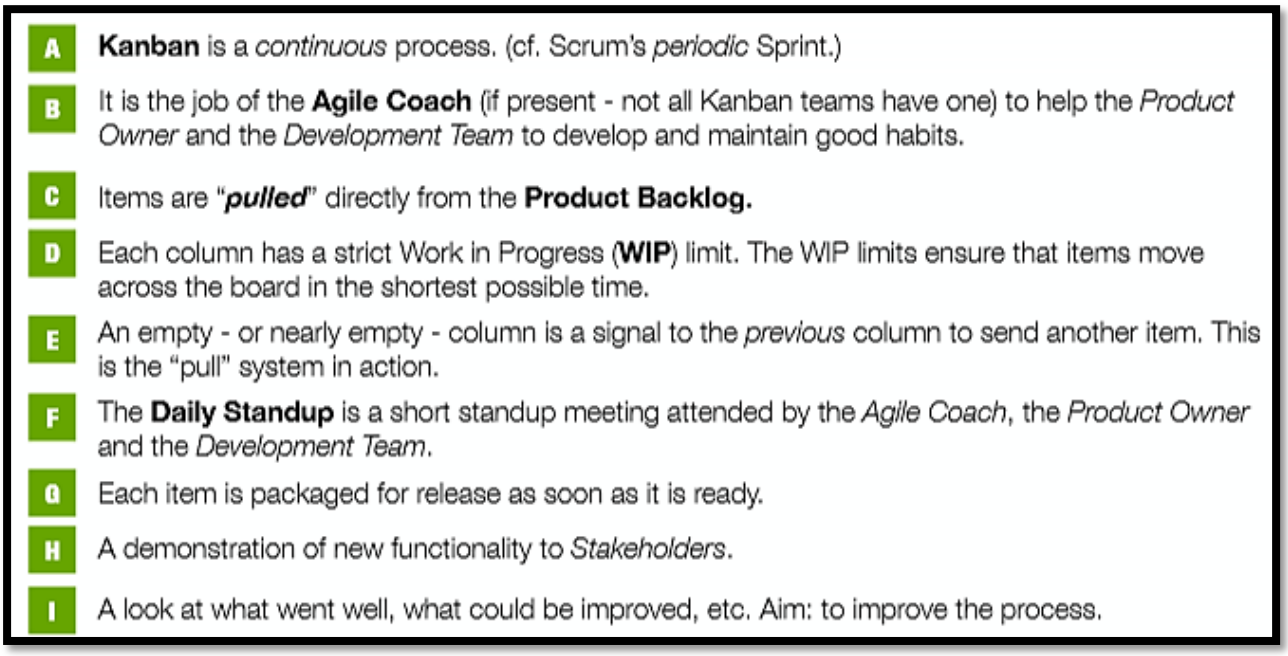


Figure 9-The Kanban Stages Descriptions
(Smartsheet Inc, 2019)

2. Secondary Research:

2.1. Mobile Operating Systems

iOS vs Android:

In the current smartphone market, there are two dominant platforms, iOS from Apple Inc. and Android from Google.

Although there are some similarities between these two platforms when building applications, developing for iOS or Android, involves using different software development kits (SDKs) and different development toolchain.

While Apple uses iOS exclusively for its own devices, Google makes Android available to other companies, provided they meet specific requirements, such as including certain Google applications on the devices they ship.

Companies can build apps for hundreds of millions of devices by targeting both platforms. (Amazon Web Services, Inc., 2020)

Major Mobile Development Approaches:

- Native Applications
- Cross-Platform Applications
- Hybrid-Web Applications
- Progressive Web Applications

Native vs Hybrid Applications:

Native	Hybrid
Platform Specific	Cross Platform
Compiled Language	Scripting / Compiled
Access to device hardware	Plugins / Native Modules
Platform Frameworks	Web Frameworks

(Amazon Web Services, Inc., 2020)

2.2. Mobile Development

Mobile application development is the process of creating software applications that run on a mobile device. This process involves creating installable software bundles, implementing back-end services such as data access with an API, and testing the application on target devices and emulators. (Amazon Web Services, Inc., 2020)

- **Front-End Development:**

The front-end is the visual and interactive part of the application. It resides on the device and can be downloaded from the chosen platform.

The responsibility of building a front-end fall on the developers that need to keep in mind the user experience, app design and good interaction calls with the back-end. These developers are skilled in the languages and technologies that are used to create this front-end application.

- **Back-End Development:**

Reliable back-end services are a key feature in a high-quality mobile application, regardless of what front-end platform is being used. There are two main options on developing a mobile back end service.

The decision relies on, should a development team run and maintain their services, or should they take advantage of a 3rd party service, like cloud-based services.

The answer is clear, to improve developer productivity and efficiency, mobile app developers should only build their services if they are highly specific to the domain of the application and embody unique intellectual property. (Amazon Web Services, Inc., 2020)

- **Front-end to Backend:**

The mobile front-end gets the data from the back-end via a variety of service calls, such as APIs.

APIs, or application program interfaces, are sets of protocols and tools that specifies how software components should interact.

In other words, an API is a messenger, that takes a request (front-end) and tells a system (back-end) what the request is, then returns a response to that request.

2.3. Object-Oriented Approach

An object-oriented system is made up of interacting objects that maintain their local state and provide operations on that state. The representation of the state is private and cannot be accessed directly from outside the object. Object-oriented design processes involve designing object classes and the relationships between these classes. When the design is realised as an executing program, the objects are created dynamically from these class definitions. (Sommerville, Software Engineering, 2011)

In OO, Cohesion is a desirable design component attribute, as when a change must be made, it is localised in a single cohesive component.

Related to cohesion, it is desirable loose coupling. Loose coupling means a change to one component is unlikely to affect other components.

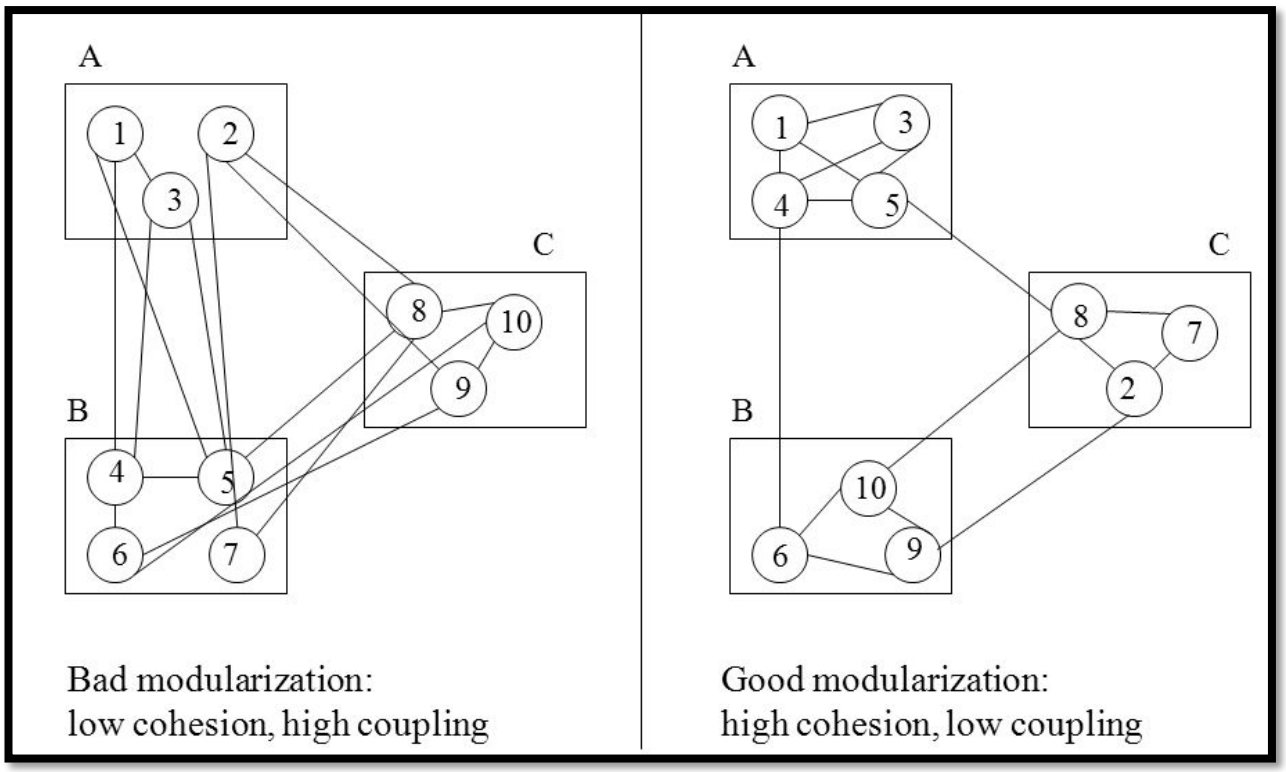


Figure 10-Coupling and Cohesion Example (Forelle, 2017)

2.4. Database Design

The research of the right database design to a project usually focuses on two different approaches that are typically called “Top-down” and “Bottom-up” methods. Both methodologies share the same objective of joining a system by describing the more significant part of the association between the procedures. (Odendaal, 2017)

Top-down design is characterized by an extensive planning and research phase, that leads to the development of the database. Commonly used when first creating a database as a high-level view of the entire database, where all requirements are often known.

Bottom-Up design takes the opposite approach. Where goals for a product are still to be outlined, the assembly of a product is done on an increment by increment basis.

In Conclusion, the Top-Down approach allows for structured control of a project, while the Bottom-Up approach lends itself to more experimentalism. (Maxey, 2012)

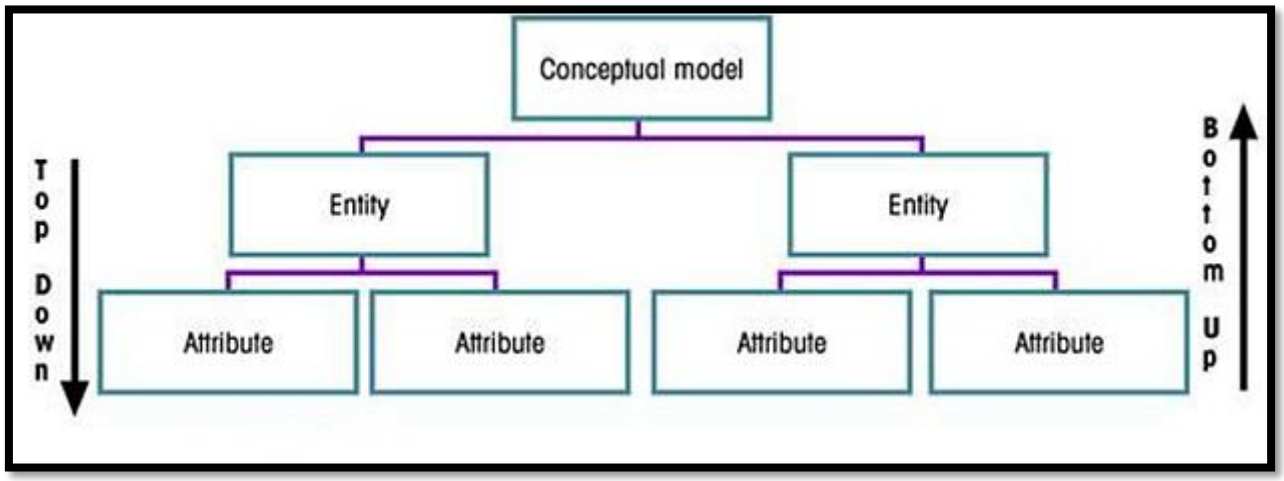


Figure 11-Top down vs Bottom-up Design Sequencing

2.5. User Interface Design

Nielsen Norman Group defines User Experience (UX) Design as “encompassing all aspects of the end-user’s interaction with the company, its services, and its products.”

A User Interface is a bridge between the developer and the user, and it must be well designed and well thought. For this project, several aspects of a “good user interface” are present on the design and development of this system.

User Interface Requirements:

1.	This system shall always keep users informed about what is going on, through appropriate feedback, within reasonable time.
2.	It shall speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.
3.	It shall follow real-world conventions, making information appear in a natural and logical order.
4.	It shall have careful design which prevents a problem from occurring, alongside with good error messages.
5.	Users shall not have to wonder whether different words, situations, or actions mean the same thing.
6.	This application shall use objects, actions, and options visible to minimize the user’s memory load.
7.	The user shall not have to remember information from one part of the dialogue to another.
8.	Instructions for the good use of the system shall be visible or easily retrievable whenever appropriate.
9.	Dialogues shall not contain information which is irrelevant or rarely needed.

(Nielsen, 1994)

2.6. Testing

Testing Process

Right from the conceptualization of the project, testers are involved in discussions with the client, project manager and various stakeholders. Only exhaustive testing can show a program is free from defects. However, exhaustive testing is impossible, and every system has its flaws, but in order to make sure that the best product is delivered test engineers and penetration testers follow a strict testing process supported by testing policies.

Component Testing:

- ⇒ Testing of individual program components;
- ⇒ Usually the responsibility of the component developer;
- ⇒ Tests are derived from the developer's experience.

System Testing:

- ⇒ Testing of groups of components integrated to create a system or sub-system;
- ⇒ The responsibility of an independent testing team;
- ⇒ Tests are based on a system specification.

Testing Policies:

Testing policies define the approach to be used in selecting system tests:

- ⇒ All functions accessed through menus should be tested;
- ⇒ Combinations of functions accessed through the same menu should be tested;
- ⇒ Where users input is required, all functions must be tested with correct and incorrect input. (Sommerville, Software Testing, 2006)

System Testing

Involves integrating components to create a system or sub-system. May involve testing an increment to be delivered to the customer and it is composed by two phases.

Phases:

- **Integration Testing:**

Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements and it occurs before validation testing, Appendix G. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- **Release Testing:**

The Process of testing a release of a system that will be distributed to customers. Primary goal is to increase the supplier's confidence that the system meets its requirements. Release testing is usually black-box testing. (Sommerville, Software Testing, 2006)

Project Methodology

1. Management Methodology

Agile Development Methodology

> ***The Agile Manifesto:***
>
> *"**Individuals and interactions** over processes and tools*
> ***Working software** over comprehensive documentation*
> ***Customer collaboration** over contract negotiation*
> ***Responding to change** over following a plan"*

Agile is a process by which a team can manage a project by breaking it up into several stages and involving constant collaboration with stakeholders and continuous improvement and iteration at every stage. The Agile methodology begins with clients describing how the end product shall work and what problem to solve. The Agile methodology is the chosen one to be applied throughout the stages of this project. (Wrike, 2019) (Manifesto for Agile Software Development, 2001)

Although this is not a team project, the team management process is applied, in order to demonstrate a more professional approach.

Advantages	Disadvantages
It allows software to be released in iterations.	Rely on real-time communication.
Iterative releases improve efficiency by allowing teams to find and fix defects.	Users often lack the documentation they need to get up to speed.
Align expectation early on.	Huge time commitment from users.
Allow users to realize software benefits earlier, with frequent incremental improvements.	Are labour intensive because developers must fully complete each feature within each iteration for user approval.
Good response to Change.	Lack of Predictability.
Less Up-Front Work.	Hard to implement Agile Methodologies in large organizations.

(Team, 2017)

2. Development Methodology

The Scrumban Methodology

Scrumban introduced by Corey Ladas, a software development methodologist enthusiast. In his book, “Scrumban: Essays on Kanban Systems for Lean Software Development”, he states that Scrumban was created as a mean of transitioning a development team, from Scrum to Lean or Kanban. However, while the intent was for teams to replace Scrum, it has become a methodology of its own, combining elements of both Scrum and Kanban.

The decision of replacing Scrum or Kanban with Scrumban should be made in response to the environment and organisational needs. Scrumban is most commonly used in development and maintenance projects. (Smartsheet Inc, 2019)

Regarding the growth of flexibility, while adopting a hybrid methodology, many different versions of Scrumban were built. After the analysis of both Scrum and Kanban, in the ‘Literature Survey’, it is possible to bring together the best characteristics of both and build a hybrid version, creating the best Scrumban methodology for this project’s development.

Project's Scrumban Method Stages:

1)	The Project Team works in a series of Increments (Sprints) of 1,2,3 or 4 weeks duration.
2)	It is the job of the **Team Master** to help the *Product Owner* and the *Development Team,* to develop and maintain good habits.
3)	Each Sprint starts with an **Increment Planning Meeting**-facilitated by the Team Master and attended by the *Product Owner* and the *Development Team* and, possibly, other *Stakeholders*. Together they select high priority requirements from the **Product Backlog** that the Development Team can commit to deliver in a single Increment.
4)	Increments are "pulled" directly from the **Product Backlog**, into the **Scrumban Board**.
5)	Each column has a strict Work in Progress (**WIP**) limit. The WIP limits ensure that requirements move across the board in the shortest possible time.
6)	An empty, or nearly empty, column is a signal to the previous column to send another requirement forward. This is the "**pull**" system in action.
7)	After an Increment is complete a **review** of the Sprint takes place.
8)	When the review is validated, occurs a demonstration of the new functionality to the *Stakeholders*.
9)	The Development team-facilitated by the Team Master- must do an examination of what went well, what went wrong and what could be improved, to make each Sprint more efficient and effective than the last.
10)	At the end of each Sprint, completed items are packaged for release. Any incomplete items return to the Product Backlog.

(Straughan, 2017) (Sommerville, Software Engineering, 2011)

3. Requirements Gathering Methodology

Qualitative research

Qualitative research is research undertaken to gain insights concerning attitudes, beliefs, motivations and behaviours of individuals to explore a social or human problem and includes methods such as focus groups, in-depth interviews, observation research and case studies.

This project's qualitative research is present in the "Background", "Rational" and "Literature Survey" sections. The combination of these sections motivates the exploration of a problem. In order to "solve" this problem, a case study is built and with it, other qualitative research methods.

Figure 12-Qualitative Research Methods
(Bhat, QUALITATIVE RESEARCH: DEFINITION, TYPES, METHODS AND EXAMPLES, 2019)

Quantitative research

Quantitative research is research concerned with the measurement of attitudes, behaviours and perceptions and includes interviewing methods such as telephone, intercept, door-to-door interviews, as well as self-completion methods such as mail-outs and online surveys.

This project undertakes in-depth interviews and surveys with the users so that the best feedback appears as a quantitative method of research.

Subsequently, the surveys will be analysed using correlational research and if a relationship between two groups or entities is established experimental research must take place in order to improve the user's interaction or to solve the identified problem.

In the conclusion of the project, a critical evaluation takes place, conducted to measure the effectiveness and performance of the system and the concept in achieving its objectives.

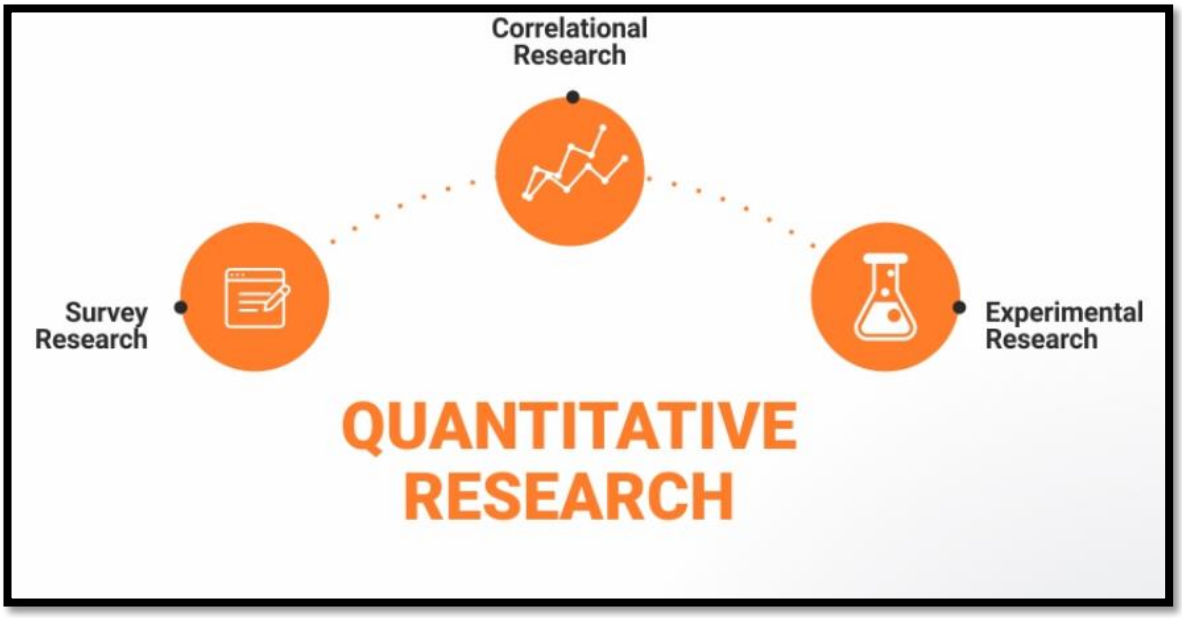


Figure 13-Quantitative Research
(Bhat, QUALITATIVE RESEARCH: DEFINITION, TYPES, METHODS AND EXAMPLES, 2019)

4. Design Methodology

The Unified Modelling Language

Unified Modelling Language is used to specify, visualize, modify, construct and document the artefacts of an object-oriented software-intensive system under development. UML supports 13 different types of models. Minimizing the number of models produced reduces the costs of the design and the time required to complete the design process. The UML design is composed of two kinds of design models:

1. **Structural models**-describe the static structure of the system using object classes and their relationships. Essential relationships that may be documented at this stage are generalization (inheritance) relationships, uses/used-by relationships, and composition relationships.

2. **Dynamic models**- describe the dynamic structure of the system and show the interactions between the system objects. Interactions that may be documented include the sequence of service requests made by objects and the state changes that are triggered by these object interactions. (Sommerville, Software Engineering, 2011)

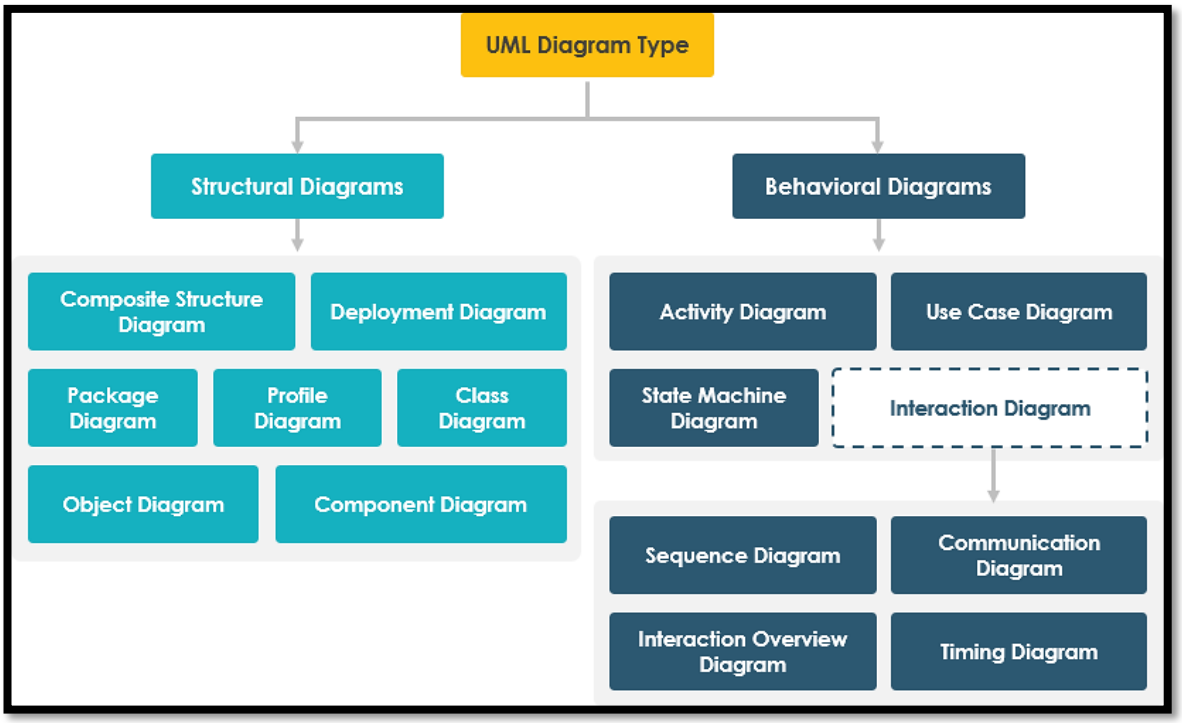


Figure 14-UML Diagram Types (Visual Paradigm, 2019)

Database Design Methodology

By investigating the two database designs presented by 'Literature Survey', it is possible to conclude, that the best choice for this project, is naturally the fastest and less duplicative way possible of storing and retrieving data.

Since both methodologies have their advantages and disadvantages, a combination of the two is more likely the best suitable option, with the beginning of the project favouring the 'Top-Down Approach', with all amendments undertaken with the 'Bottom-up Approach'. (Odendaal, 2017)

This way, the management of the database will be under control but also, with the freedom of new implementations, implementing a more agile back-end database. (Brower, 2015)

Cloud-Backend Database Methods:

In specific projects, of small teams, the mobile developers may not be experts in spinning up and running a back-end infrastructure. Therefore, this project takes advantage of a cloud services provider, specifically 'Firebase' by Google.

Firebase services handle all the drudge work and heavy lifting of managing a back-end, so the developers can focus more on the features and functionalities of the application and at the same time, reassuring that the app will have scalability, security and reliability. (Amazon Web Services, Inc., 2020)

Firebase Provides:

Data Services
Cloud Services
Real-Time Data
Application Logic

Essential
User Sign-up/Sign-in and Management
Social login (Facebook sign-in, Twitter sign-in, etc.)
Analytics and User Engagement
Push Notifications
Real Device Testing

Machine Learning
Conversional Bots
Image and Video Recognition
Speech Recognition

(Amazon Web Services, Inc., 2020)

5. Software Development Methodology

Object-Oriented Methodology

An object-oriented approach suits this project's objective to bind together the data and the functions that operate on them so that no other part of the code can access that data except that function, this has a significant increase in the readability and security of this project, because data is stored layer by layer. (Prabhu, Consulted 29/10/2019)

Project's Object-Oriented Concepts:

1) **Polymorphism:** Allows the application to be able to differentiate between entities with the same name efficiently;
2) **Inheritance:** Mechanism by which one class can inherit the features (fields and methods) of another class;
3) **Encapsulation:** Allows the wrapping up of data under a single unit;
4) **Abstraction:** The process of identifying only the required characteristics of an object ignoring the irrelevant details, furthermore, only essential details are displayed to the user.
   (Prabhu, Consulted 29/10/2019)

Object-Oriented Stages:

1)	Understand and define the context and the external interactions with the system;
2)	Design the system architecture;
3)	Identify the principal objects in the system;
4)	Develop design models;
5)	Specify interfaces.

6. Mobile Development Methodology

Native Mobile Applications Approach:

Native mobile applications are written in the programming language and frameworks provided by the platform and running directly on the device, such as iOS and Android.

Pros	Cons
Best Performance;	Higher costs when building and maintain the app;
Direct access to device APIs;	Multiple code bases for each platform;

(Amazon Web Services, Inc., 2020)

After the study in the '*Literature Survey,'* it is possible to decide the use of Android, as the chosen platform, for this project.

Furthermore, following the Native approach, Android has two main programming native languages, Java and Kotlin. Although Kotlin has recently been declared as Google's official Android primary language, Java, as an object-oriented language, was present during the study of the module 'Mobile Apps' and its use in this project is an opportunity to take that knowledge further. (Altexsoft, 2018)

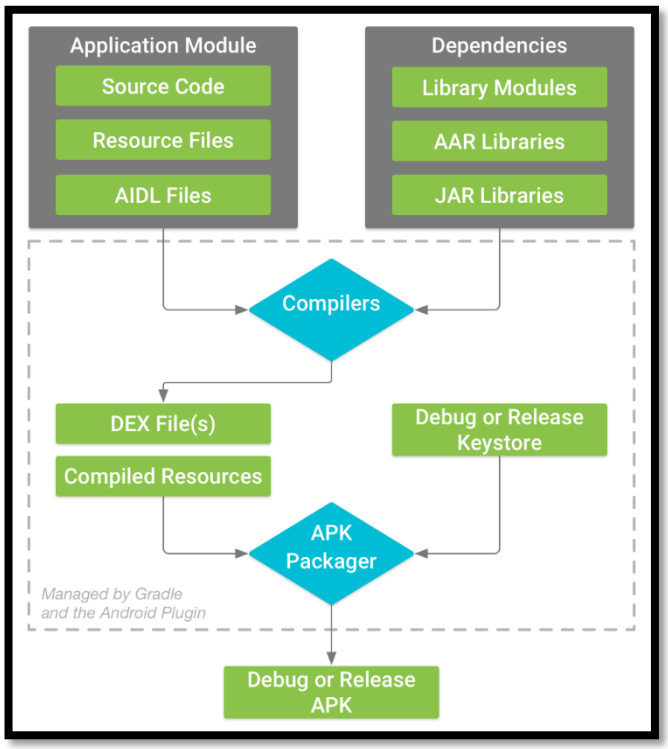


Figure 15-Android Build Process
(Patrick, 2017)

Software Design Patterns

The Model-View-Controller (MVC) Pattern:

The idea of patterns as a way of presenting, sharing, and reusing knowledge about software systems is widely used. The MVC Pattern separates presentation and interaction from the system data. An MVC system is structured into three logical components that interact with each other:

⇒ **Model-**manages the system data and associated operations on that data;
⇒ **View-**defines and manages how the data is presented to the user;
⇒ **Controller-**manages user interaction and passes these interactions to the View and the Model.

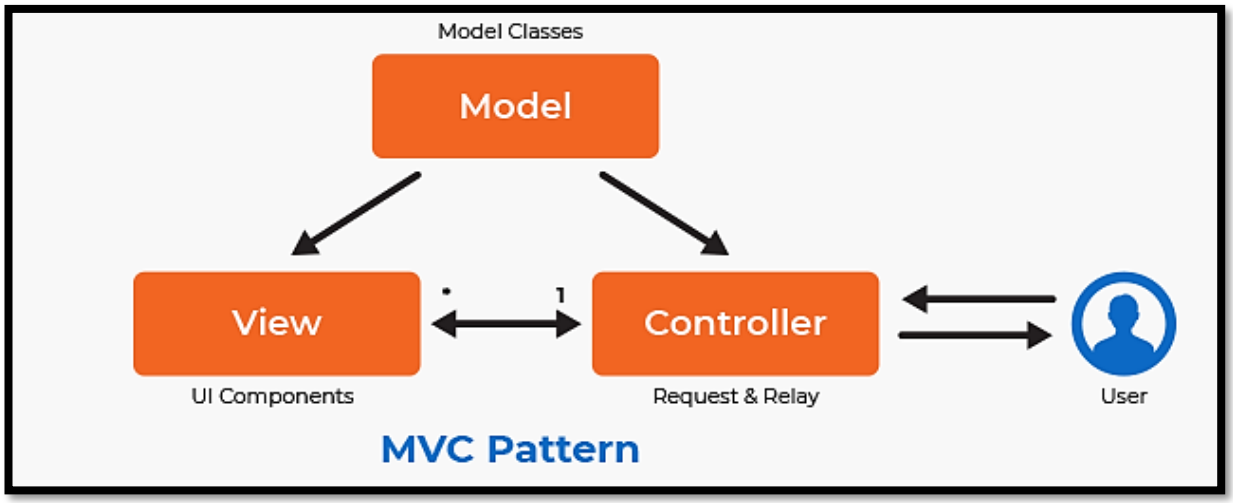


Figure 16-MVC Model (Solanki, 2019)

Problem with MVC in Android Development:

The Model-View-Controller pattern has two main disadvantages:

1. The View has a reference to both the Controller and the Model;
2. It does not limit the handling of UI logic to a single class, this responsibility is shared between the Controller and the View or the Model.

The method used to find out the best architecture design for this application can be found in '*Appendix F'*. With that technique, the solution to the MVC problem was discovered in the MVP pattern, explained in the '*Design'* section.

7. Testing Methodology

Black Box Testing

Black Box Testing, also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item tested is not known to the tester.

These tests can be functional or non-functional, though usually functional. This method is named so, because the software program, in the eyes of the tester, is like a black box inside which one cannot see.

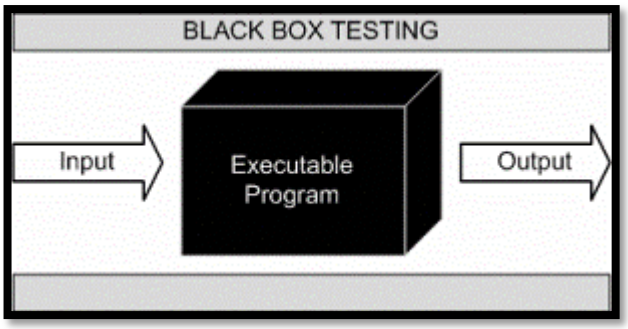


Figure 17-Black Box Testing

Black Box Testing Techniques:

Equivalence Partitioning:

⇒ It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.

Boundary Value Analysis:

⇒ It is a software test design technique that involves the determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.

Cause-Effect Graphing:

⇒ It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly. (Fundamentals, Black Box Testing, 2019)

White Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.

The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming knowledge and implementation knowledge is essential. White box testing is testing beyond the user interface and into the fundamentals of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box, inside which one clearly sees.

White Box Testing method is applicable to:

⇒ **Unit Testing**-Tests paths within a unit;
⇒ **Integration Testing**-Tests paths between units;
⇒ **System Testing**-Tests Paths between subsystems;

(Fundamentals, White Box Testing, 2019)

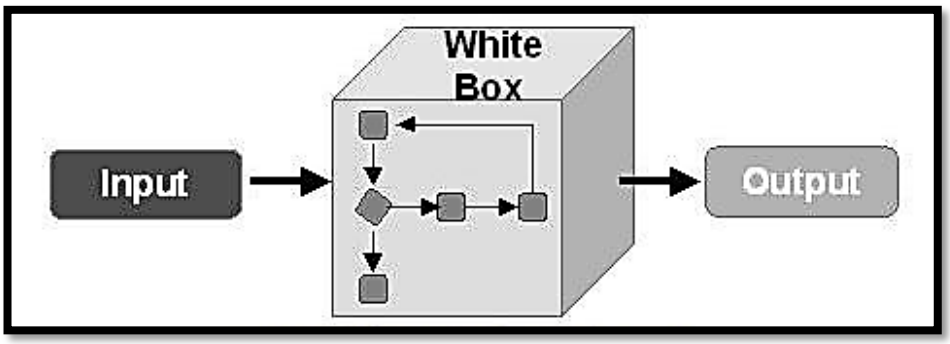


Figure 18-White Box Testing

Unit Testing:

Unit testing is a level of software testing where individual units/increments of the software get tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

Unit Testing is the first level of software testing and is performed before Integration Testing, and it is usually performed by software developers themselves or their peers. (Fundamentals, Unit Testing, 2019)

Path Testing:

The objective of path testing is to ensure that the set of test cases verifies each path through the system is executed at least once.

The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control and using the same code section as before. (Guru99, 2020)

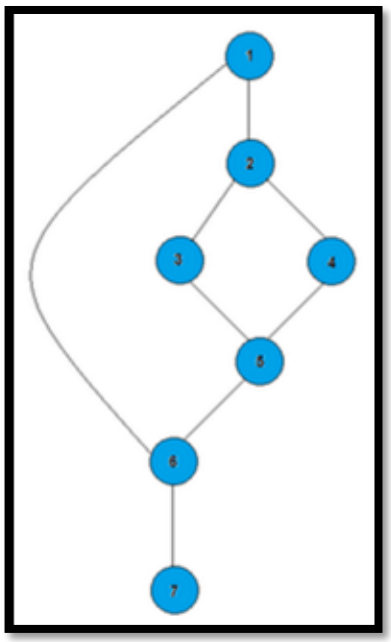


Figure 19-Flow Graph Example

Project Requirements

The requirements engineering process of this project started back at the beginning of the '*Feasibility Studies'* section. There, it was decided whether the proposed idea is achievable or not.

1. Requirements Elicitation and Analysis

It requires technical staff working with customers/end users, in order to find out more about the application's domain, the services that the system should provide and the system's operational constraints. (Sommerville, Software Engineering, 2011)

In this Project, both quantitative and qualitative methods of data gathering are implemented in the process of *requirements discovery*.

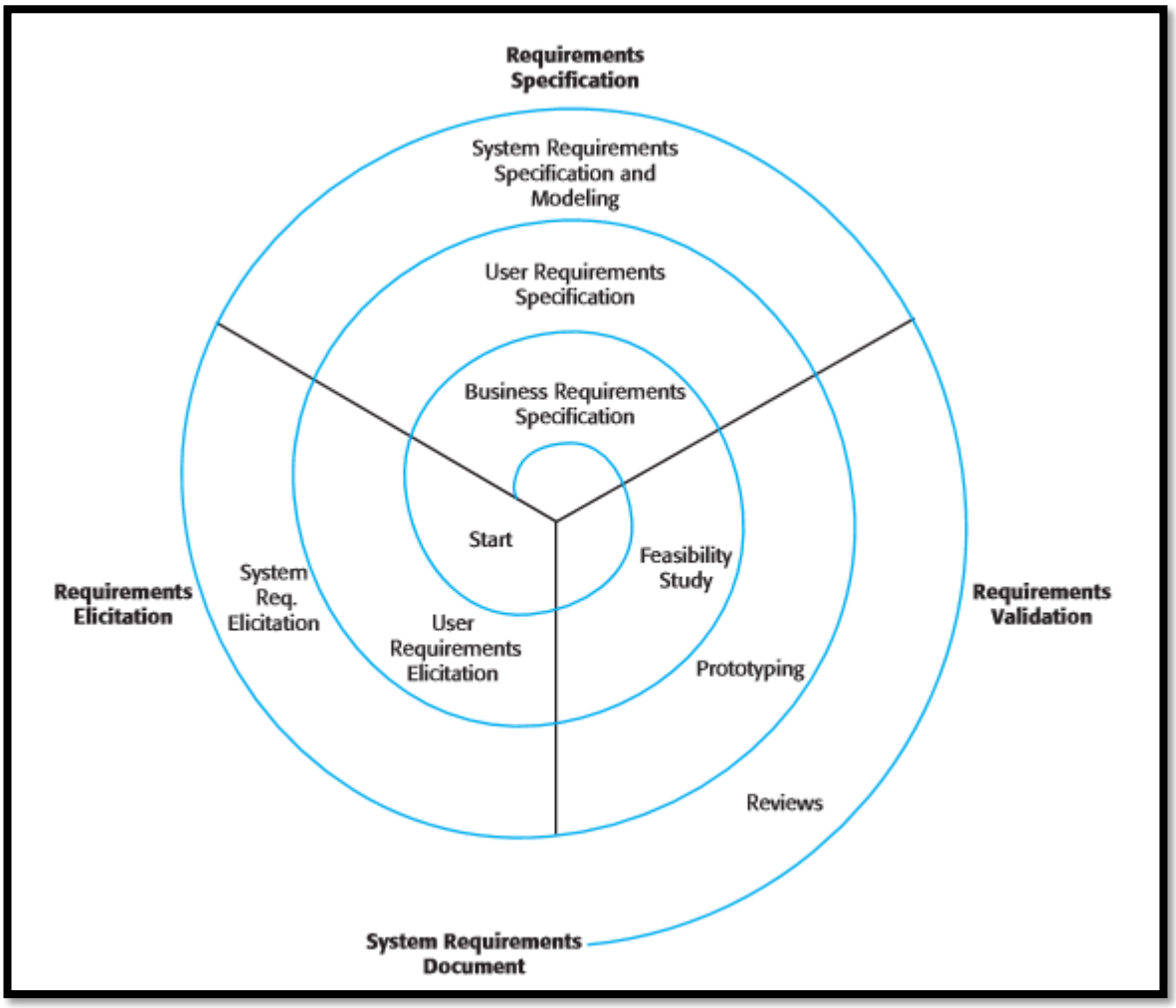


Figure 20-Requirements Engineering Process (Sommerville, Software Engineering, 2011)

2. Primary Data Collection Techniques

Quantitative Methods:

- **Questionnaires:**

Questionnaires are widely used collecting data methods, mainly when data is to be collected from a large number of people who are scattered over a wide area. They are used both as an independent and separate method of collecting data. They are also used as an additional tool to check data gathered through observation and personal interview.

Types of Questionnaires:

i) Structured Questionnaires

ii) Unstructured Questionnaires

⇒ **Structured Questionnaires:**

In this project's *Data Gathering Plan*, Structured questionnaires are the ones with more advantage in collecting quantitative data. Structured by concrete and prepared questions, it means the questions are prepared in advance and not constructed on the spot during the questioning period. Additional questions may be used only when needed or when inadequate replies by informants happen. These structured questionnaires may be of two broad types:

1) Closed-Form Type

2) Open-End Type

The Selected type for this Project's Plan is Closed-Form Questionnaire. In the closed-form questionnaire, several alternative answers are provided at the end of each question, and the informant must choose one. The advantages are that the responses are more easily tabulated, and statistical measures are easily applied because the number of possible answers to each question is fixed. (Dua, 2019)

Qualitative Methods:

- **Interviews:**

Interviews consist of collecting data by asking questions. Data can be collected by listening to individuals, recording, filming their responses, or a combination of methods. There are three types of interview:

1) Structured interview
2) Semi-structured interview
3) In-depth interview

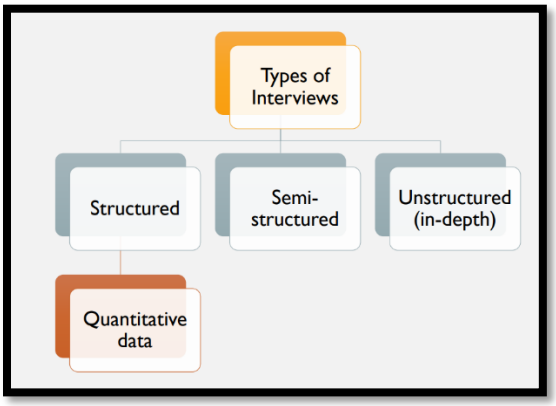


Figure 21-Types of Interviews

⇒ **Semi-structured interview:**

Semi-structured interviews include a number of planned questions, but the interviewer has more freedom to modify the wording and order of questions.

In-depth interview is less formal and the least structured, in which the wording and questions are not predetermined. This type of interview is more appropriate to collect complex information with a higher proportion of opinion-based information. The semi-structured interview is the type applied in this project's data gathering stage, and although questions are not determined in semi-structured interviews, some topic questions will be set down to help the interviewee.

Advantages:

- Collect complete information with greater understanding.
- It is more personal, as compared to questionnaires, allowing to have higher response rates.
- It allows more control over the order and flow of questions.
- Can introduce necessary changes in the interview schedule based on initial results (which is not possible in the case of a questionnaire study/survey).

Disadvantages:

- Data analysis—especially when there is a lot of qualitative data.
- Interviewing can be tiresome for large numbers of participants.
- Risk of bias is high due to fatigue and to becoming too involved with interviewees. (Abawi, 2017)

What Data to Collect and How to Use It?

In the 'Literature Survey', various techniques were described on how to use data gathering methods to obtain information about the system's features and requirements, although the details of what data to collect were left out.

These details are essential. However, the success of any analysis technique depends on the availability of the appropriate data. Such details are addressed bellow.

Important Questions:

1)	How large should the Database of Monuments be? a) How many countries for a first MVP? b) How many Monuments per country?
2)	What information should a Monument display?
3)	What information should a Country Display?
4)	Should, a monument, have a video presentation?
5)	Should, a monument, display its location via Map?
6)	Should it have a Login feature?
7)	Should it have a 'Most Famous Monuments' feature?
8)	Should it have a 'Favourites' feature?
9)	Should it have a 'User Profile' feature?
10)	What Design suits this system better? i) Colours; ii) Images; iii) Shapes; iv) Icons;

These are the type of questions and topics applied in the questionnaires and interviews. The answers to these questions are the key to build up a list of Functional and Non-Functional Requirements, approved by this application's target audience.

A template of the questionnaires is present in Appendix C, and a Script of one of the interviews is present in Appendix D, in order to have a better understanding of the process used to gather data.

3. Functional User Requirements

ID	Requirement	Specification	Source
FR.1	Continents Menu	The application shall display the Continents in a menu, with a respective shape icon and name.	Interviews
FR.2	Continent Info	After selecting a continent, the same shall display a curiosity/presentation about itself.	Surveys
FR.3	Countries Menu	The application shall display the countries in a menu, with the respective name and flag.	Interviews
FR.4	Country Info	After selecting a country, the same shall display its respective icon and a curiosity/presentation about itself.	Surveys
FR.5	Monuments Menu	Each country shall have a list of Monuments.	Interviews Surveys
FR.6	Monument's Info	Each Monument shall display basic information about itself: history, picture, location, characteristics, etc.	Interviews
FR.7	Monument's Video	Each Monument shall direct the user to a YouTube video that presents the monument.	Interviews
FR.8	Information Source	Each Monument shall have a '*Source'* bottom that directs the user to a link with more detailed information, in case the user wants to learn more about a Monument.	Surveys
FR.9	Monument's Map Location	Each Monument shall have a mapping feature that shows a small map with the exact location of that Monument.	Interviews
FR.10	Take Picture	Each Monument shall have the option of open the device's camera and let the user take a picture in case they are visiting the moment at real-time.	Interviews
FR.11	Register	The application shall have a register feature.	Surveys

ID	Requirement	Specification	Source
FR.12	Login	The application shall have a Login feature.	Surveys
FR.13	Profile	Each user shall have a Profile that lets them edit their own personal information.	Interviews
FR.14	Most Famous Monuments	In the '*Countries Menu'* the application shall display the most famous monuments among users, on that continent.	Surveys
FR.15	Create Memories	Each user shall have a feature to save memories from a monument, each memory saves a picture and a description of the visit.	Interviews Surveys
FR.16	View Memories	Each user shall have a list of memories created that display the memory picture, description and date.	Interviews Surveys
FR.17	Favourites	The user shall have the option of adding a specific monument to a list of favourite monuments.	Interviews
FR.18	Search Monument	The '*Countries Menu'* shall have a search bar, that lets the user search for a specific monument, in that continent.	Interviews Surveys

4. Non-Functional User Requirements

ID	Requirement	Specification	Source
NFR.1	Performance	The performance of the Application can be determined by it responsive time, time to complete the given task. (Ng, 2019)	Research
NFR.2	Scalability	App should able to adopt itself to increased usage or able to handle more data as time progress. (Ng, 2019)	Research
NFR.3	Responsiveness	Application should be responsive to the user Input or to any external interrupt which is of highest priority and return to same state.	Research
NFR.4	Usability	User should be able to understand the flow of App easily i.e. users should able to use App without any guideline or help from experts/manuals. If user experience needs to be explained, then it is not good UX. (Ng, 2019)	Research
NFR.5	Reliability	The application should be reliable to perform the business, i.e. when user perform some important action it should be acknowledged with confirmation. (Ng, 2019)	Research
NFR.6	Security	All the app data should be secured and be encrypted with minimum needs so that it's protected from outside environment also from internal attack. (Ng, 2019)	Research
NFR.7	Documentation	A solid Documentation should always follow the development of every project but specially in the requirements process, so change can always be welcome. (Ng, 2019)	Research
NFR.8	Availability	There should be a common plane where the user can access your application to install and look for regular updates and give feedback. (Ng, 2019)	Research

5. Requirements validation

Requirements validation is concerned with demonstrating that the requirements define the system that the users want.

In order to prevent such mistakes, some questions help to reassure that the system provides the requirements and the functionality needed.

System Requirements Checking:

System Requirements Checking	Analyse	?
Validity -> Does the system provide the functions which best support the customer's needs?	Yes. In this case the user is looking for a fully functional mobile application, which gives a presentation of the most important places in several countries.	✓
Consistency -> Are there any requirements conflicts?	No. The requirements will be the target of several validation processes that will ensure that each requirement is well identified and managed.	✓
Completeness -> Are all functions required by the customer included?	Yes. This is a complete system with all the functions required by the customer.	✓
Realism -> Can the requirements be implemented given available budget and technology without a large impact on other requirements?	Yes. After the feasibility study of this system considering the prioritisation and negotiation of the requirements, resolving any possible conflicts.	✓
Verifiability -> Can the requirements be checked?	Yes. All the requirements proposed and presented, can be checked.	✓

6. Requirements Prioritization:

In order to demonstrate the use of the adopted agile approach, Scrumban, a product backlog is vital, to prioritize the requirements in different sections. In this Scrumban Methodology, the **WIP** limit only allows the developers to pull four tasks at a time. The next task can only be pulled into the **Scrumban Board** when the previous four are complete.

P1	Major Priority Feature
P2	Significant Priority Feature
P3	Moderate Priority Feature
P4	Low Priority Feature

Product Backlog

ID	Requirement	
FR.3	Countries Menu	1st Increment
FR.4	Country Info	
FR.5	Monuments Menu	2nd Increment
FR.6	Monument's Info	
FR.9	Monument's Map Location	3rd Increment
FR.14	Most Famous Monuments	
FR.18	Search Monument	4th Increment
FR.7	Monument's Video	5th Increment
FR.8	Information Source	
FR.10	Take a Picture	6th Increment
FR.15	Create Memories	
FR.16	View Memories	7th Increment
FR.11	Register	8th Increment
FR.12	Login	
FR.13	Profile	
FR.1	Continents Menu	9th Increment
FR.2	Continent Info	
FR.17	Favourites	10th Increment

Project Design

Software design is the creative activity in which identifies the software components and their relationships, based on a customer's requirements. It is a separate design stage, modelled and documented.

Design is about how to solve problems, so there is always a design process. However, it is not always necessary or appropriate to describe the design in detail using the UML or other design description language. Still, when developing an application, the design process becomes concerned with how to use the configurated features of the system to deliver the system requirements. (Sommerville, Software Engineering, 2011)

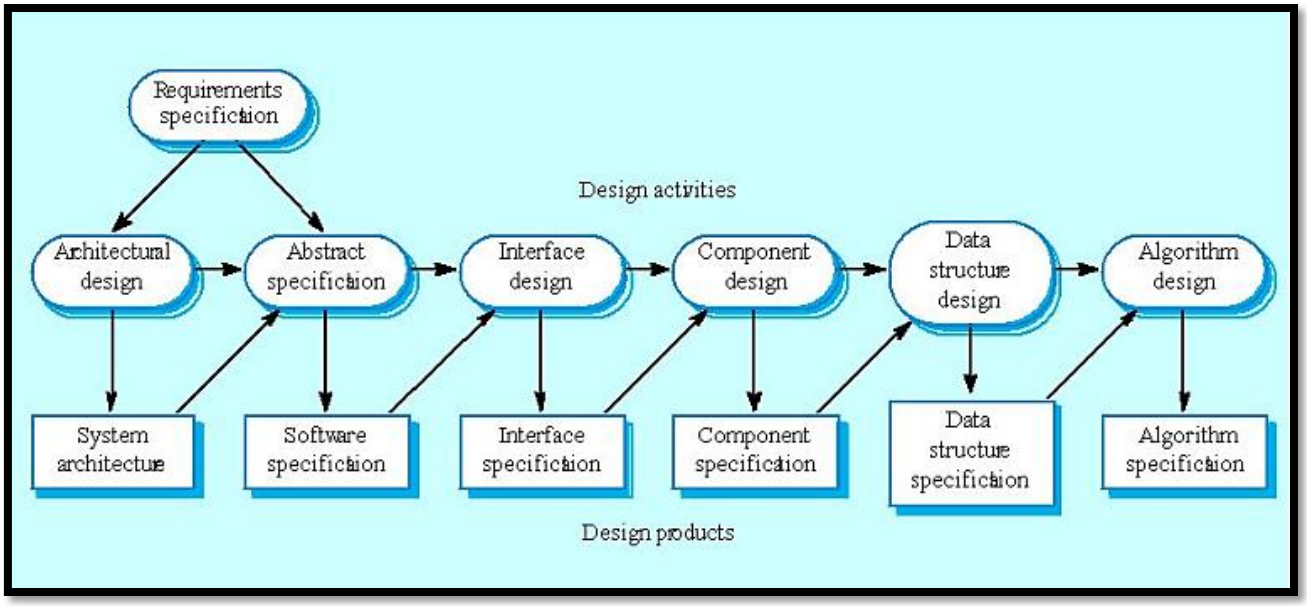


Figure 22-The Software Design Process
(Sommerville, Software Engineering, 2011)

1. System Architectural Design

In the software development process model, architectural design is the first stage. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.

Software architectures can be designed at two levels of abstraction, *architecture in the small* and *architecture in the large* in this project the most suitable one is architecture in the small:

⇒ **Architecture in the small** is concerned with the architecture of individual programs. This level, concerns with the way that an individual program is decomposed into components. (Sommerville, Software Engineering, 2011)

MVC To Model-View-Presenter

The *Model-View-Presenter* pattern solves both the issues in MVC, presented in the 'Methodology' section, by breaking the connection that the View has with the Model and creating only one class that handles everything related to the presentation of the View, the **Presenter**: a single class that is easy to unit test. (Muntenescu, 2016)

The MVP pattern has many similarities to MVC. MVP is a compound pattern to implement, but indeed it is beneficial in great benefits if applied as a well-designed solution.

Here are the roles of every component of the *Model-View-Presenter* Pattern:

- ⇒ **Model -** the data layer. Responsible for handling the business logic and communication with the network and database layers.
- ⇒ **View -** the UI layer. Displays the data and notifies the Presenter about user actions.
- ⇒ **Presenter -** retrieves the data from the Model, applies the UI logic and manages the state of the View, decides what to display and reacts to user input notifications from the View. (Muntenescu, 2016)

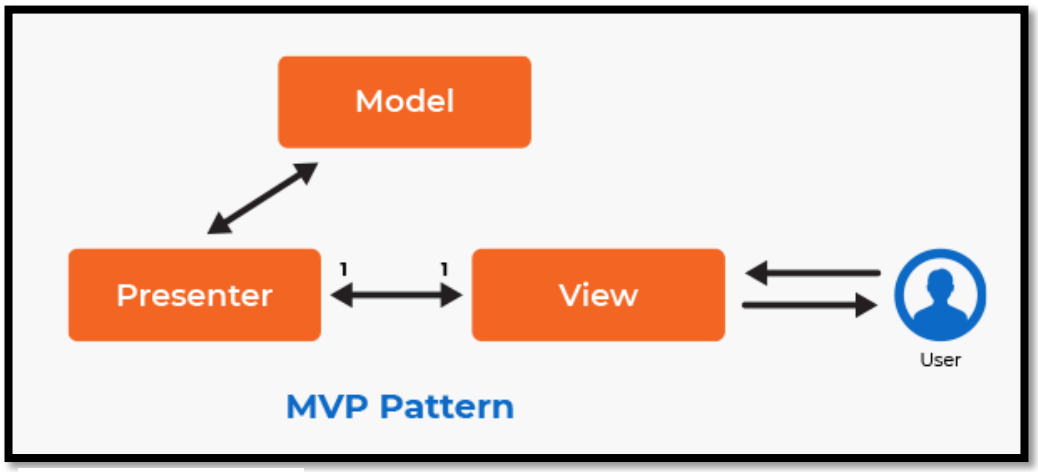


Figure 23-MVP Model (Solanki, 2019)

MVA & MVP Combined Approach

MVA, or Mode-View-Adapter, is very similar to MVP, with the main difference of having multiple adapters between a view and a model, and this difference is applied in this approach.

In this case, when a message arrives, an adapter is chosen and mediates between a Model and a View. In this approach, the Adapter is aware of the view and is the active party that observes View events. (Remer, 2013)

Both MVA and MVP design patterns are applied in the development of this project.

2. System Data Structure Design

⇒ The system's database design model is the main inspiration for the development of the back-end structure.

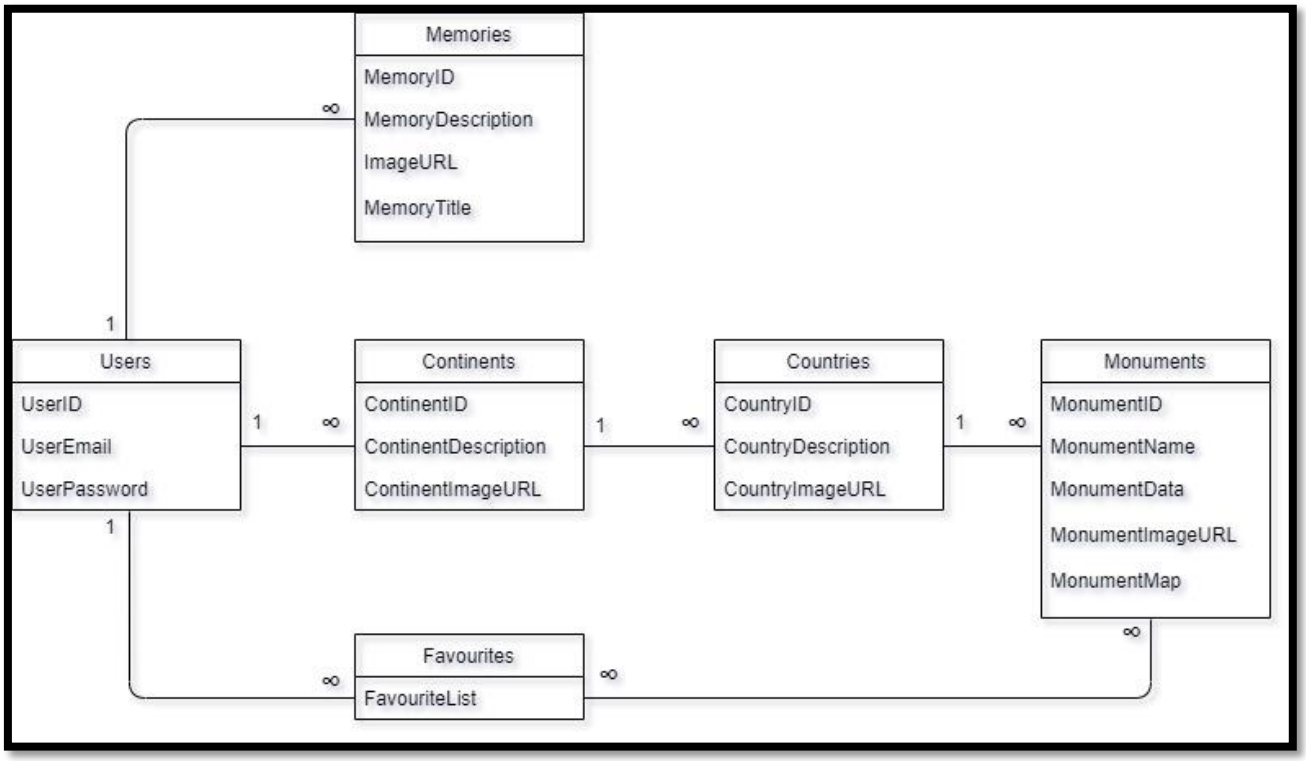


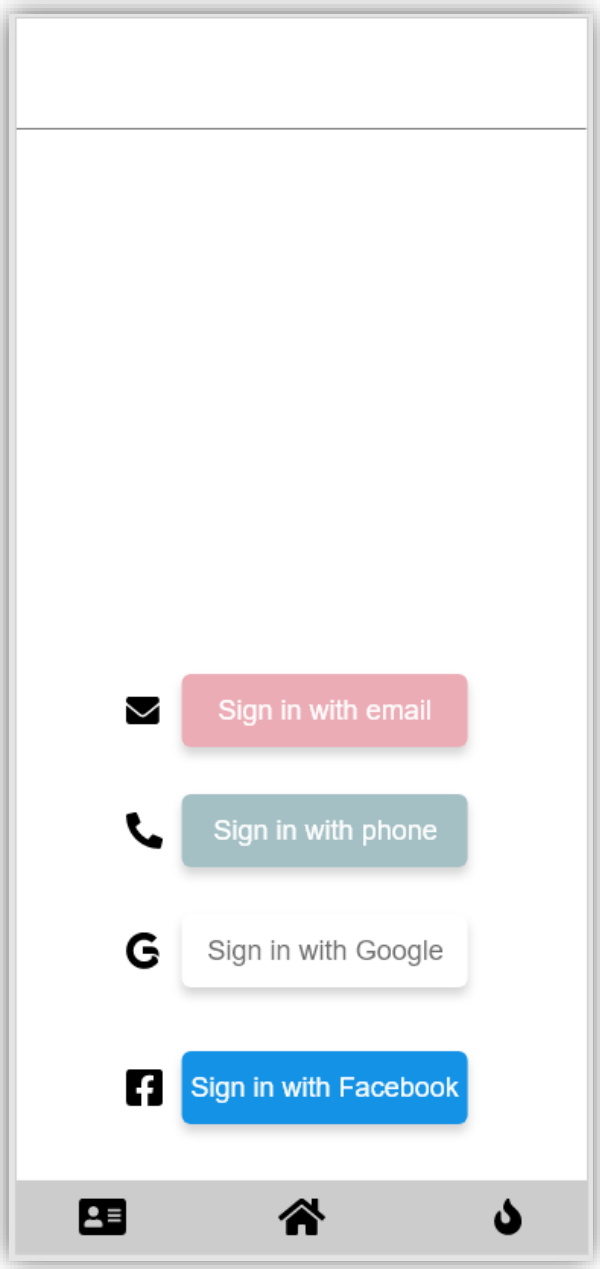
Figure 24-Database Model

3. System Interface Design

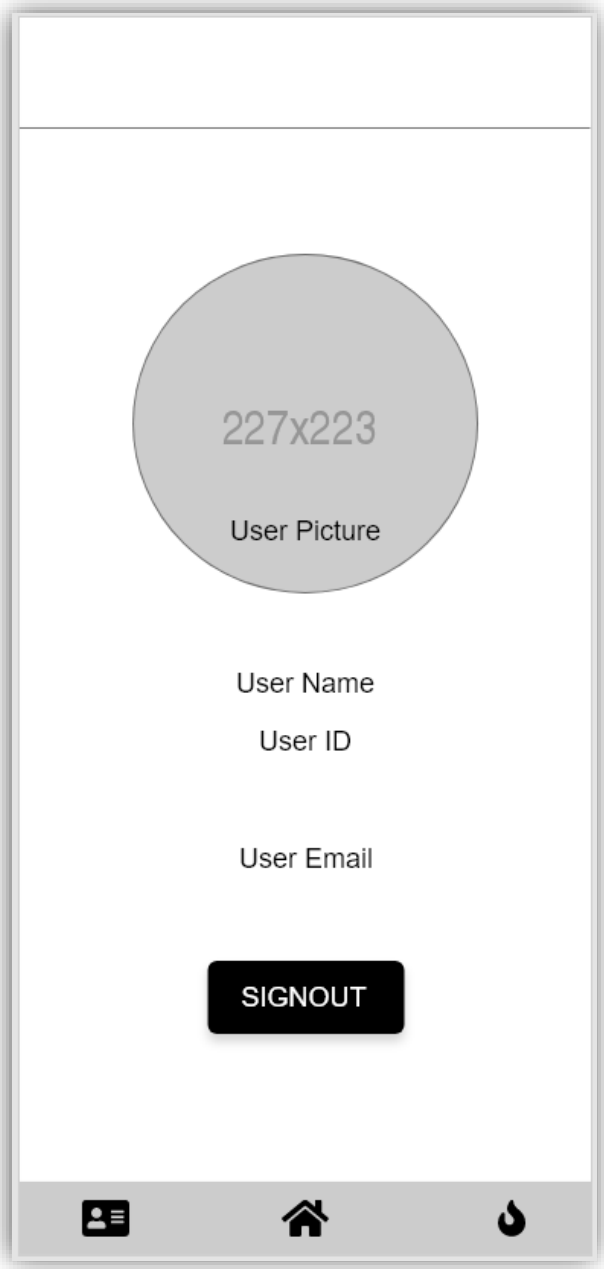
Prototype:

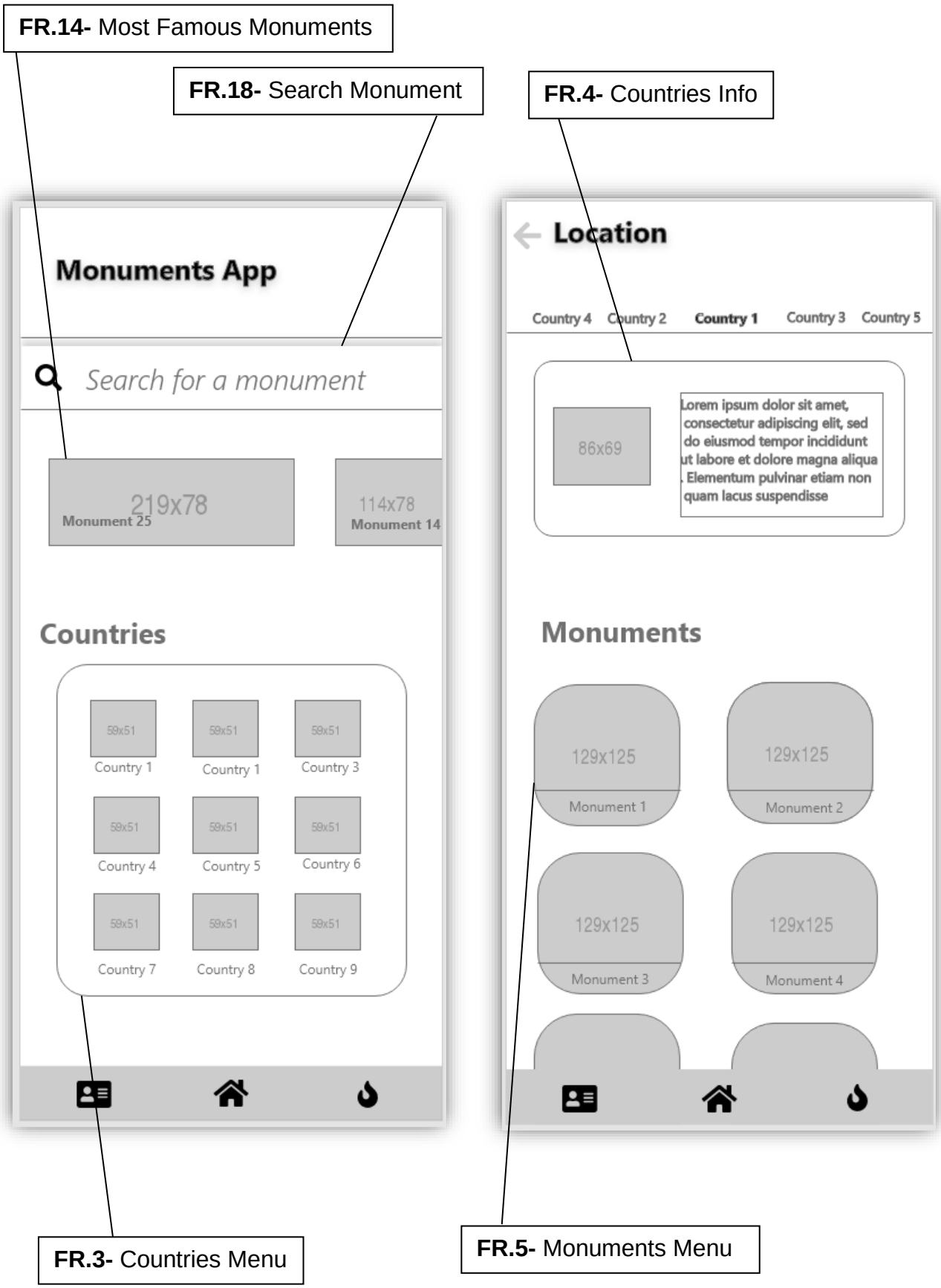
⇒ This Prototype was developed using 'Adobe XD', a graphical design tool. The following Prototype does not implement the Continent's features.

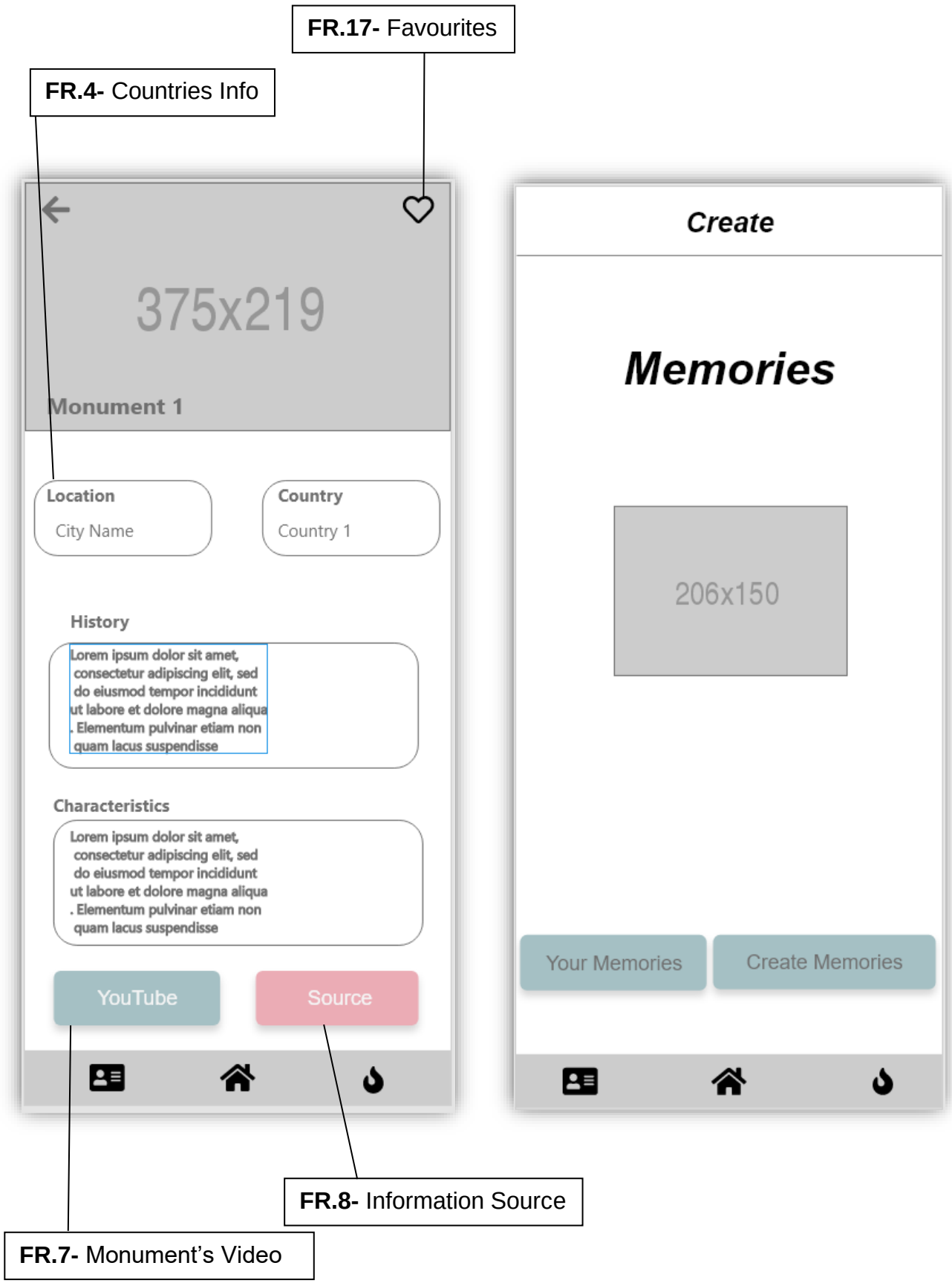
FR.11- Register **/ FR.12-**Login



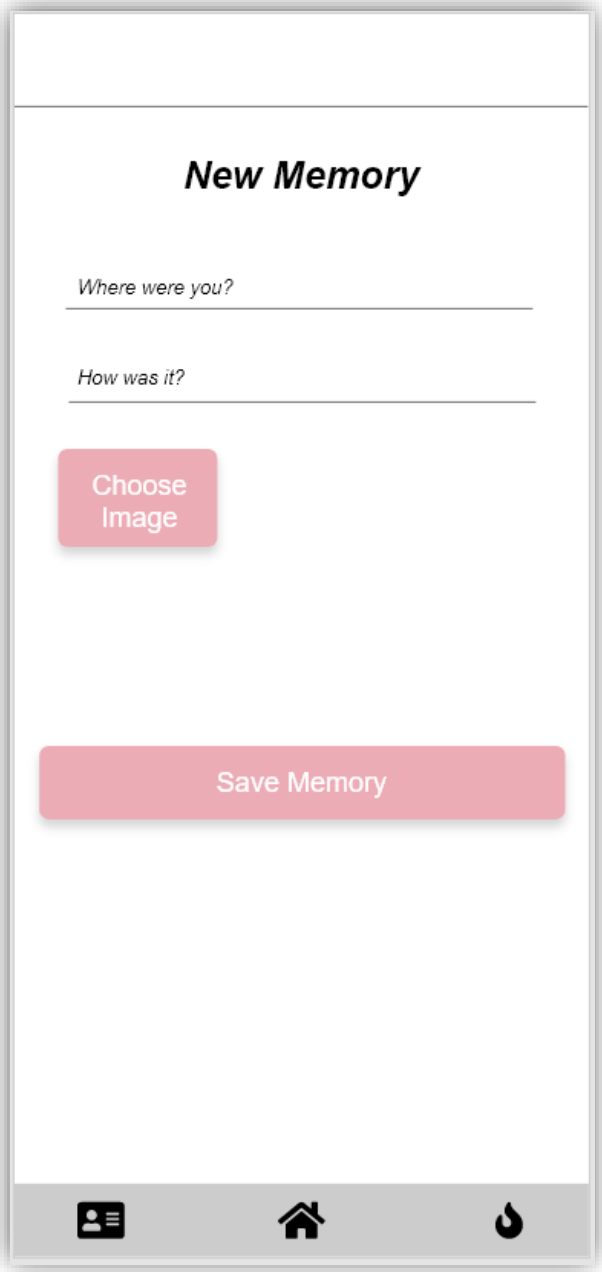
FR.13-Profile



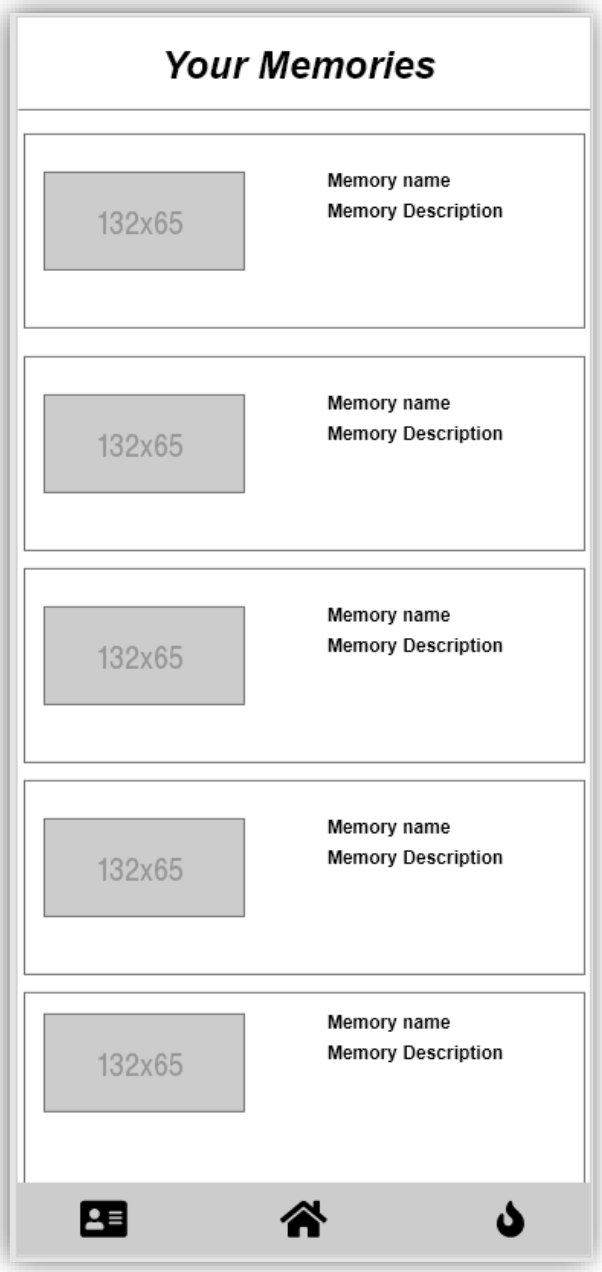

FR.14- Most Famous Monuments
FR.18- Search Monument
FR.4- Countries Info
Monuments App
Search for a monument
219x78
Monument 25
114x78
Monument 14
Countries
59x51
Country 1
Country 1
Country 3
Country 4
Country 5
Country 6
Country 7
Country 8
Country 9
Location
Country 4
Country 2
Country 1
Country 3
Country 5
86x69
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elementum pulvinar etiam non quam lacus suspendisse
Monuments
129x125
Monument 1
129x125
Monument 2
129x125
Monument 3
129x125
Monument 4
FR.3- Countries Menu
FR.5- Monuments Menu


FR.17- Favourites
FR.4- Countries Info
375x219
Monument 1
Location
City Name
Country
Country 1
History
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua . Elementum pulvinar etiam non quam lacus suspendisse
Characteristics
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua . Elementum pulvinar etiam non quam lacus suspendisse
YouTube
Source
Create
Memories
206x150
Your Memories
Create Memories
FR.8- Information Source
FR.7- Monument's Video

FR.16- View Memories



FR.15- Create Memories



4. System Abstract Specification

Context Model:

⇒ The context model of this system is represented using associations. These associations show that there are some relationships between the entities involved in the association.

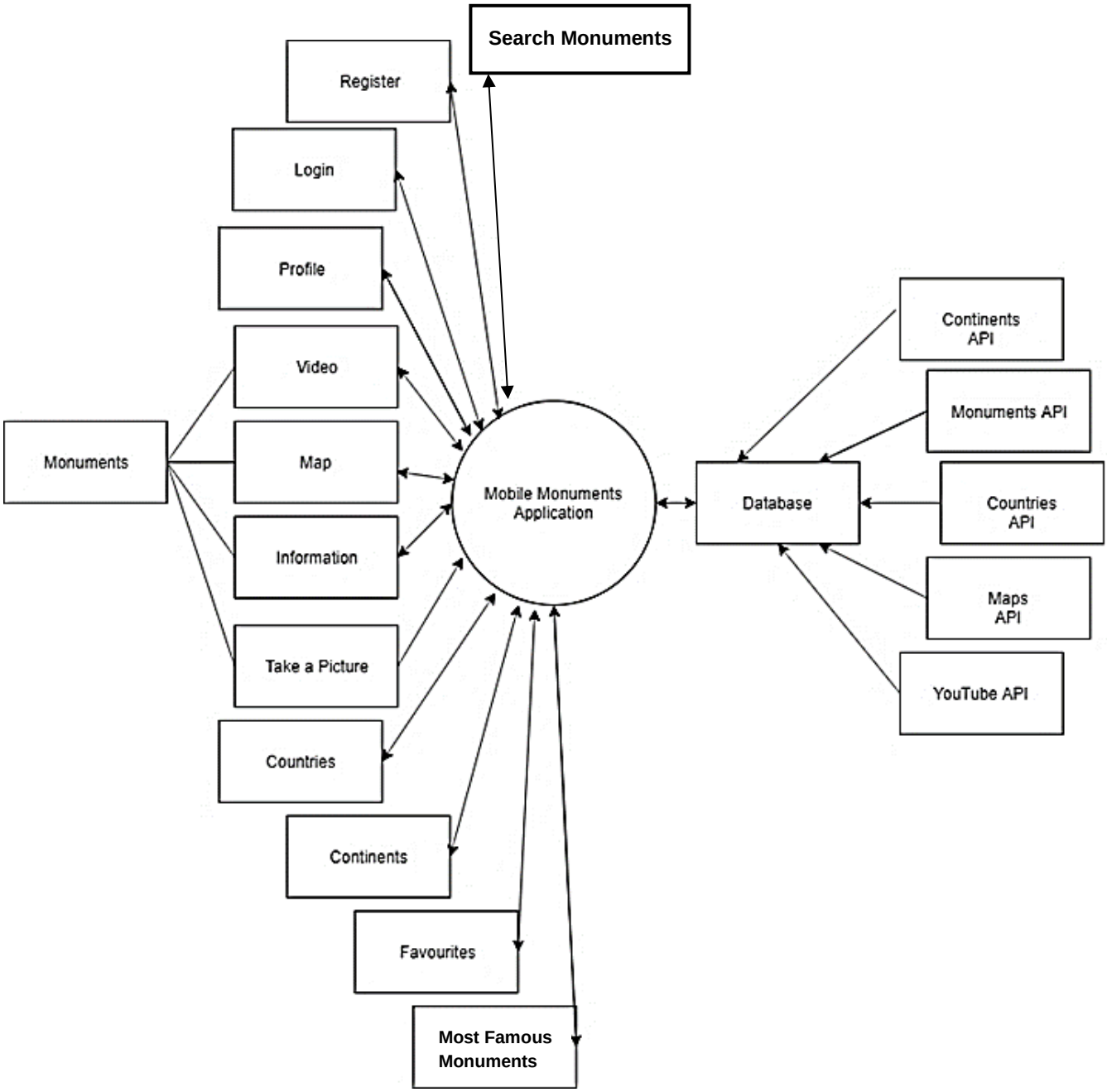
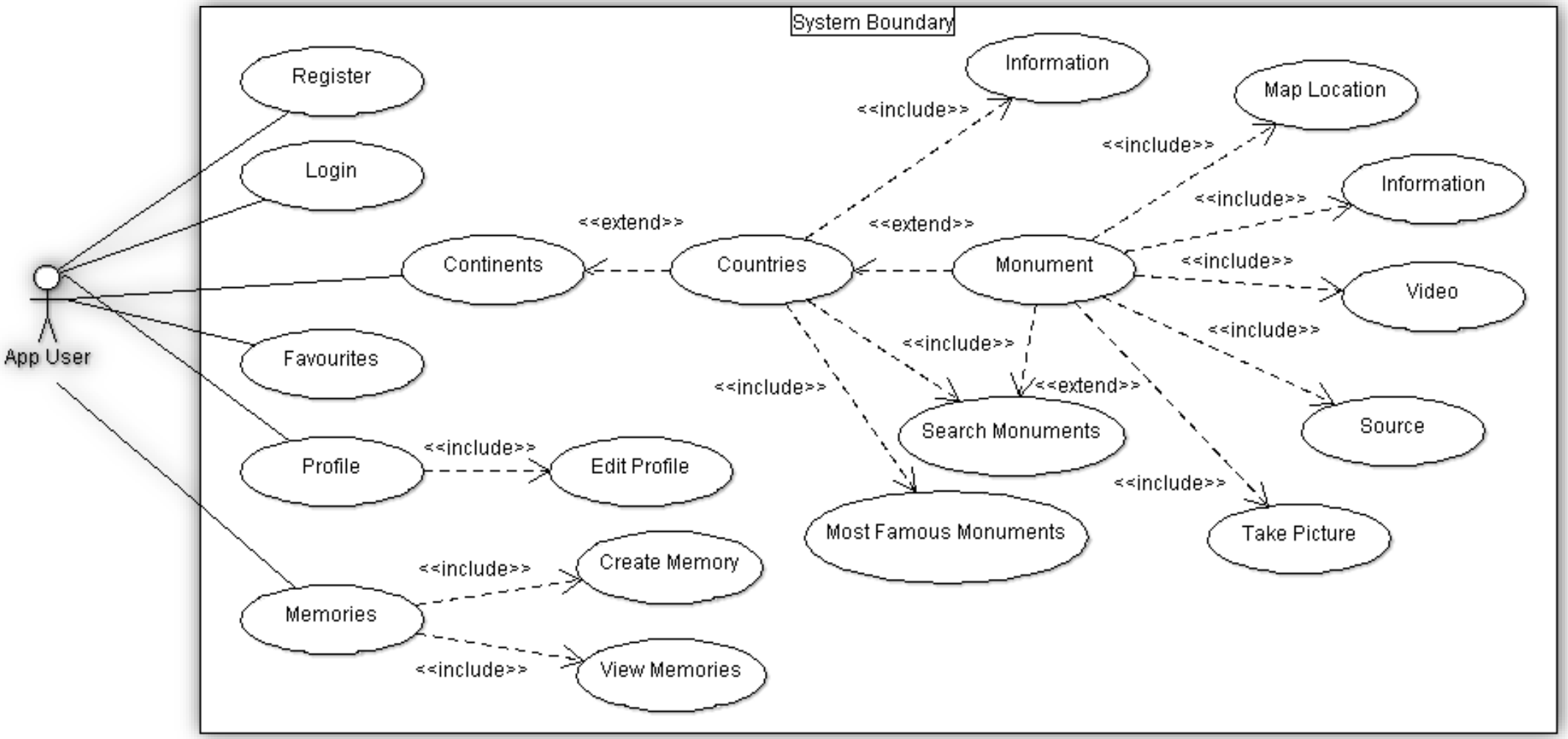


Figure 25-Context Model of the Monuments Application

Use case Diagram:

⇒ In this supplementary abstract approach, each use case, represents an interaction with the system. Each possible interaction is named in an ellipse and the external entity, involved in the interaction, is represented by a stick figure.



5. System Architecture

High-level architecture:

The high-level architectural design is composed of independent subsystems that communicate by broadcasting messages on a shared infrastructure. Each subsystem listens for messages on that infrastructure and picks up the messages that are intended for them.

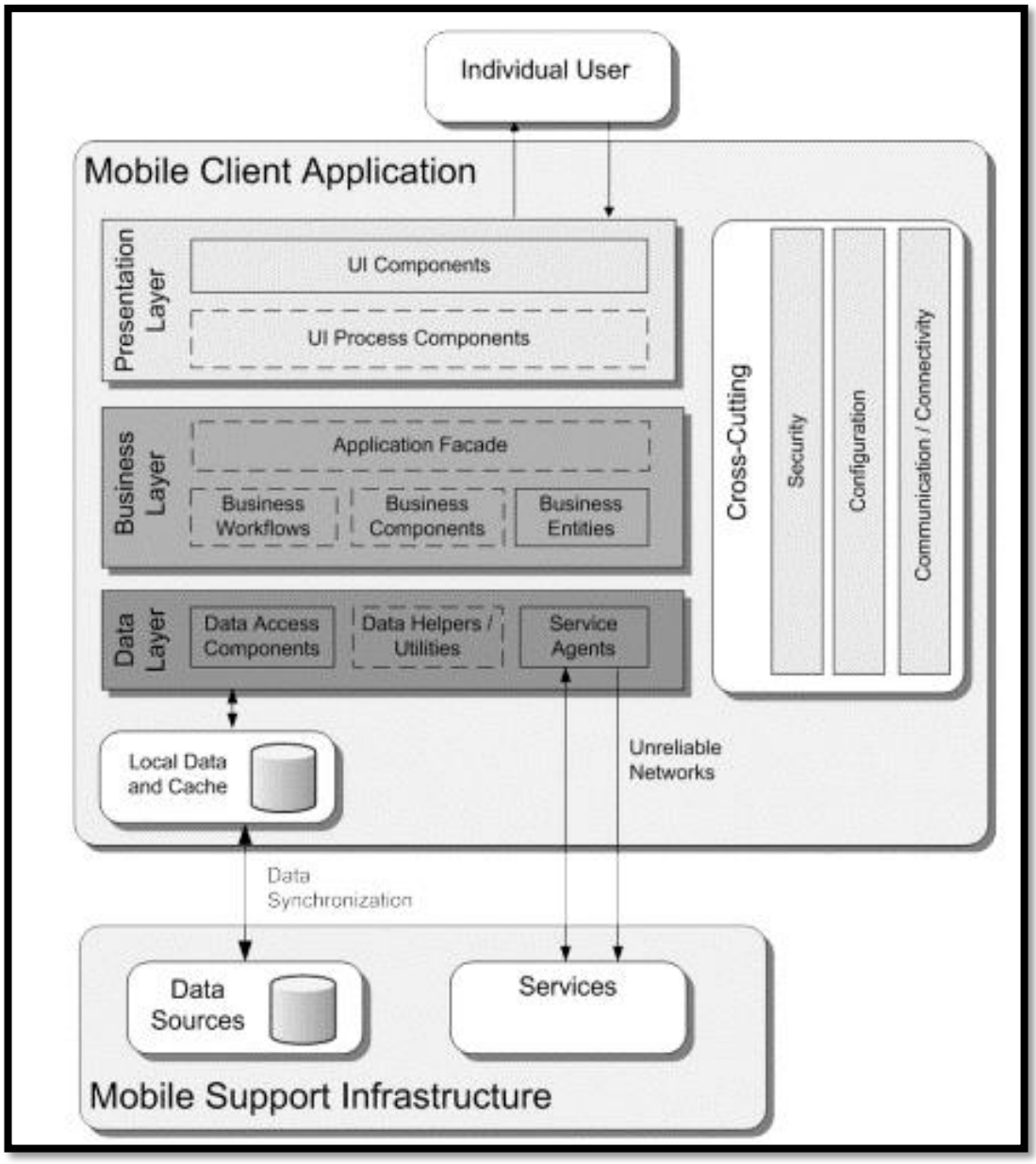
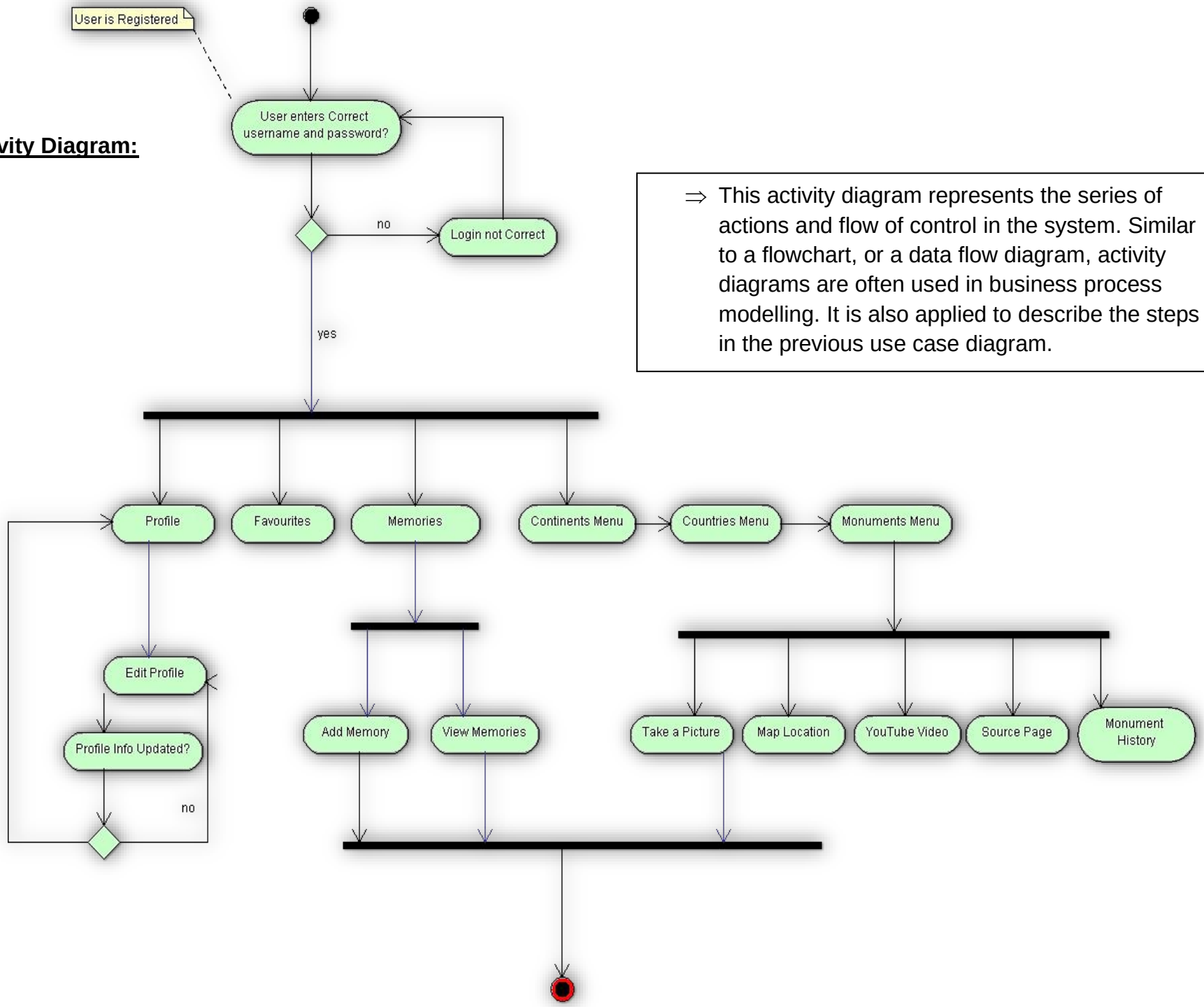


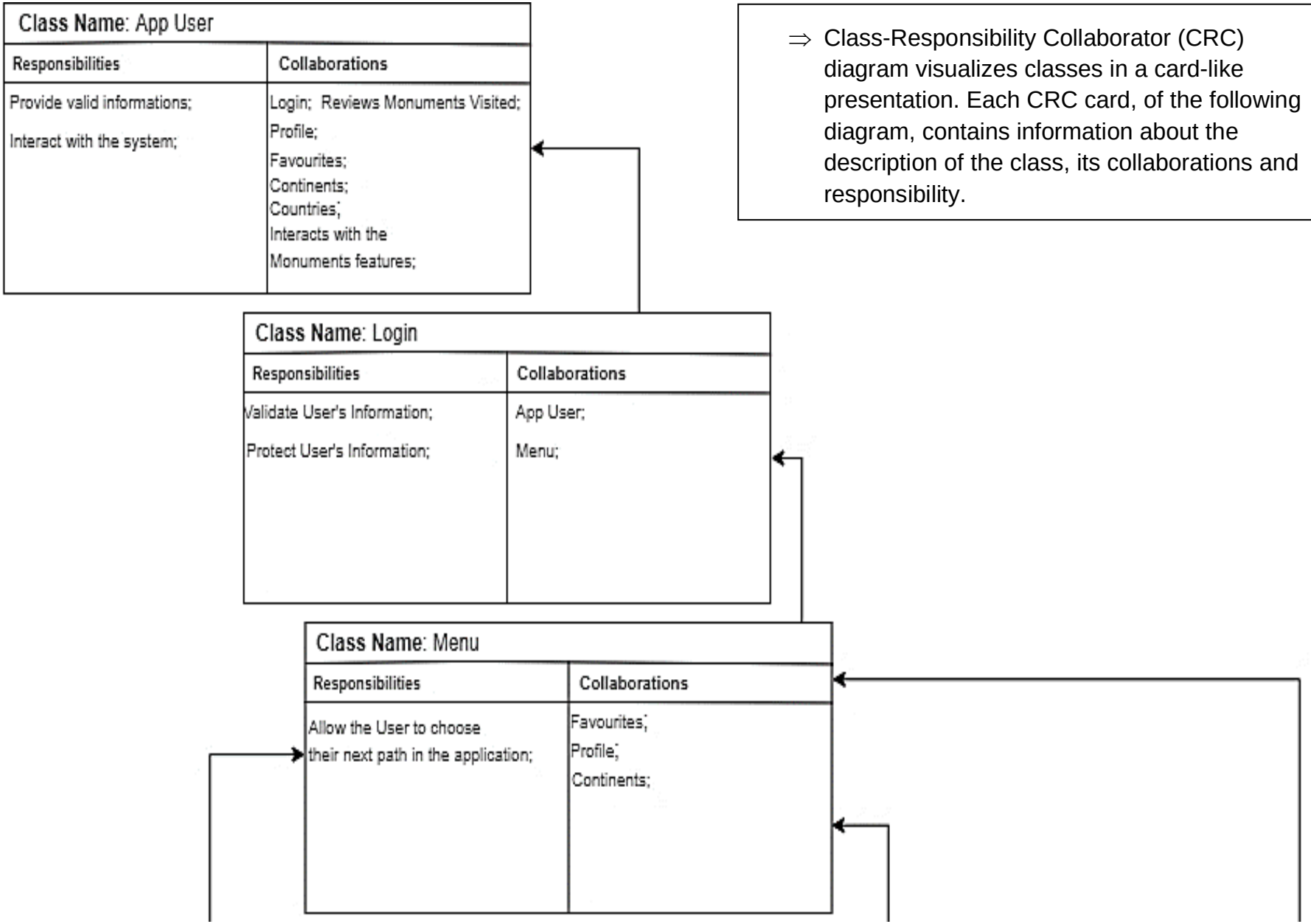
Figure 26-Mobile Application High-Level Architecture Diagram using Layers (MeierJ, et al., 2008)

System Activity Diagram:

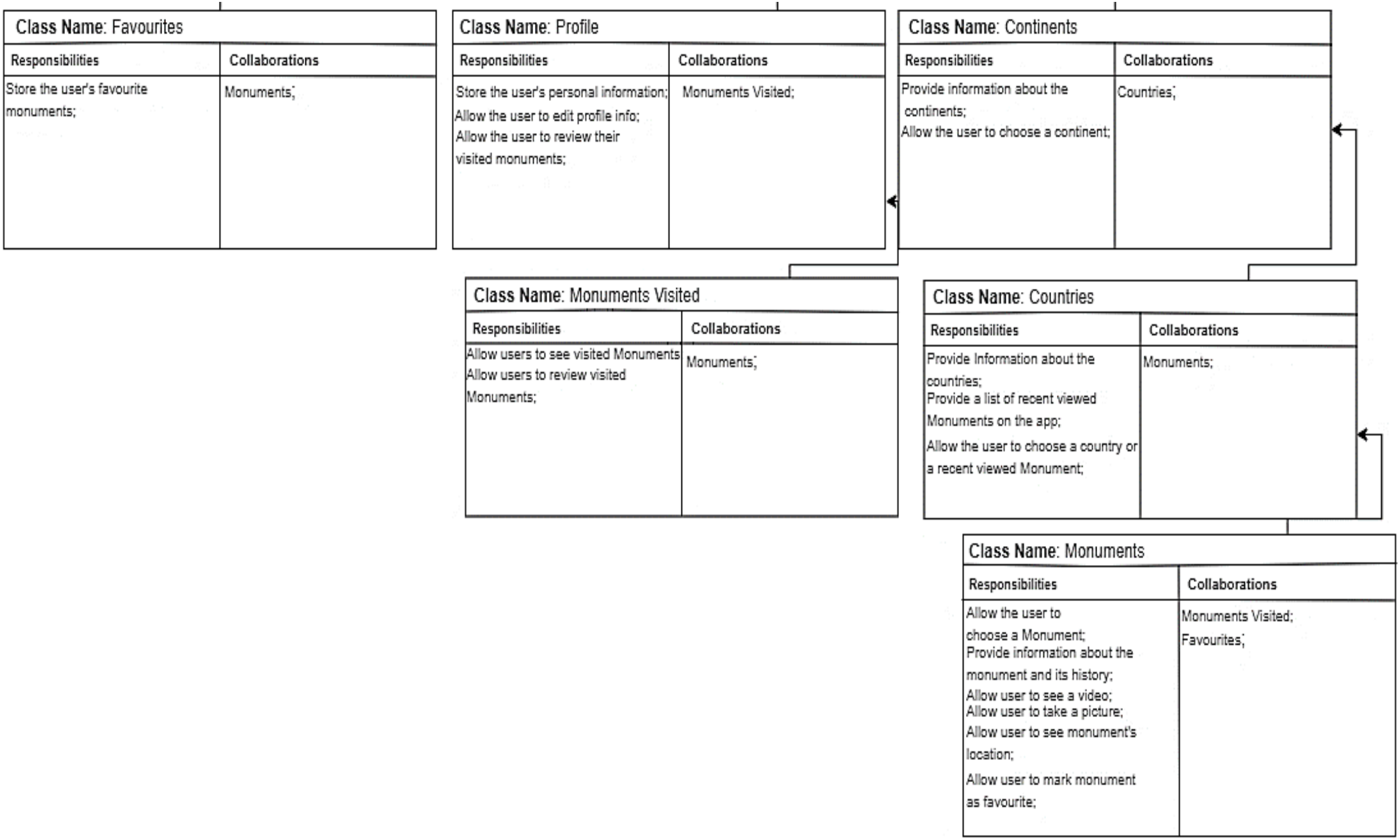


⇒ This activity diagram represents the series of actions and flow of control in the system. Similar to a flowchart, or a data flow diagram, activity diagrams are often used in business process modelling. It is also applied to describe the steps in the previous use case diagram.

CRC Diagram:



⇒ Class-Responsibility Collaborator (CRC) diagram visualizes classes in a card-like presentation. Each CRC card, of the following diagram, contains information about the description of the class, its collaborations and responsibility.


Class Name: Favourites
Responsibilities
Collaborations
Store the user's favourite monuments;
Monuments;
Class Name: Profile
Responsibilities
Collaborations
Store the user's personal information;
Allow the user to edit profile info;
Allow the user to review their visited monuments;
Monuments Visited;
Class Name: Continents
Responsibilities
Collaborations
Provide information about the continents;
Allow the user to choose a continent;
Countries;
Class Name: Monuments Visited
Responsibilities
Collaborations
Allow users to see visited Monuments
Allow users to review visited Monuments;
Monuments;
Class Name: Countries
Responsibilities
Collaborations
Provide Information about the countries;
Provide a list of recent viewed Monuments on the app;
Allow the user to choose a country or a recent viewed Monument;
Monuments;
Class Name: Monuments
Responsibilities
Collaborations
Allow the user to choose a Monument;
Provide information about the monument and its history;
Allow user to see a video;
Allow user to take a picture;
Allow user to see monument's location;
Allow user to mark monument as favourite;
Monuments Visited;
Favourites;

Use Case Descriptions:

Even more detailed than 'User Stories', use case descriptions are described in a structured natural language and help designers, identify objects in the system and give them an understanding of what the system is intended to do.

Two examples are presented in order to demonstrate the utility of use case descriptions and are the first step in the procedure that transforms these descriptions into a more low-level design, a class diagram.

FR.18-Search Monuments Use Case Description:

System	Monuments Mobile Application
Use Case	Search Monuments
Actors	App User
Description	After selecting a respective continent, the application displays the page of all the countries of that same continent, in that view there will be an interactive 'search bar', that allows the user to search for a certain monument, in that continent.
Stimulus	Click on the search bar and type the first letters of the monument's name.
Response	The Search Adapter will query those letters in the database and look for potential matches.
Comments	The database search shall be restricted to the respective continent, in order to increase performance.

FR.10-Take a Picture Use Case Description:

System	Monuments Mobile Application
Use Case	Take a Picture
Actors	App User
Description	In each monument page there will be an option of opening the device's camera, take a picture, and store it in the device's gallery. This feature gives the opportunity to interact with the application in a real-time mode in case, the user is visiting the monument at the same time he uses the application to learn more about it.
Stimulus	Click on a Take a Picture Button on the respective monument page;
Response	Opens the device's camera; After taking the Picture it stores it on the device's gallery.
Comments	The user shall be able to take multiple pictures and stay in the camera app as long as he/she wants.

The 5C Approach

The 5C is a process adapted by professional Object-Oriented developers, to map Use Cases against their proposed Classes, methodically. In other words, is the bridge that this study uses to transition from a high-level to low-level design. The following example, of this approach, takes the two previous use case descriptions further, in their design.

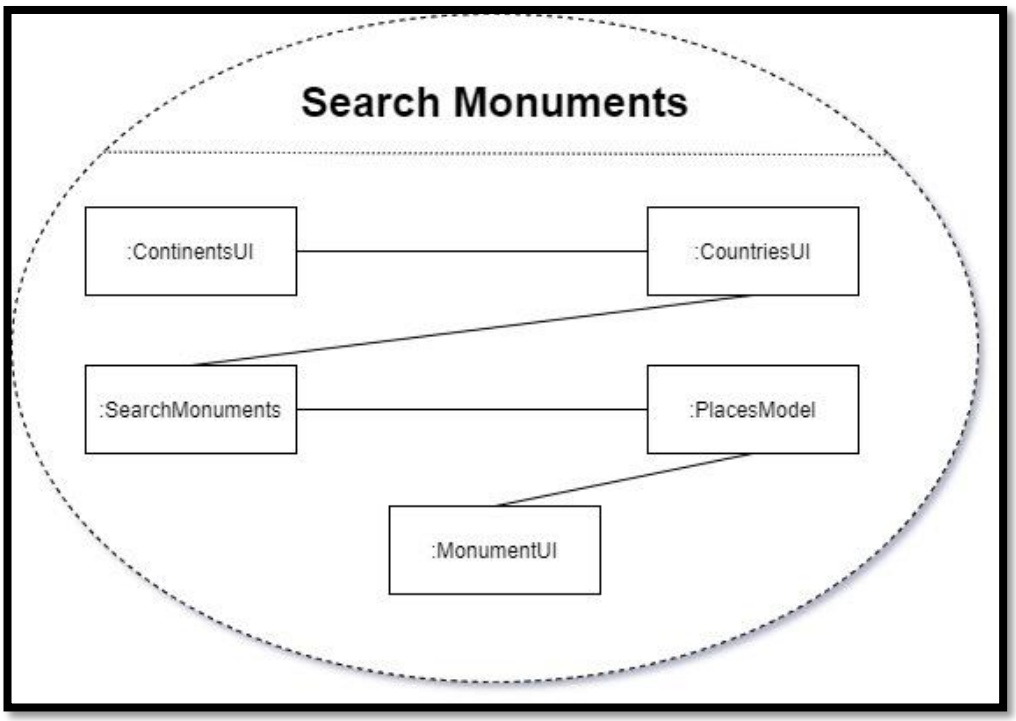
FR.18-Search Monuments 5C Approach

Use case:

⇒ A simplified use case diagram of the task, of searching a specific monument.

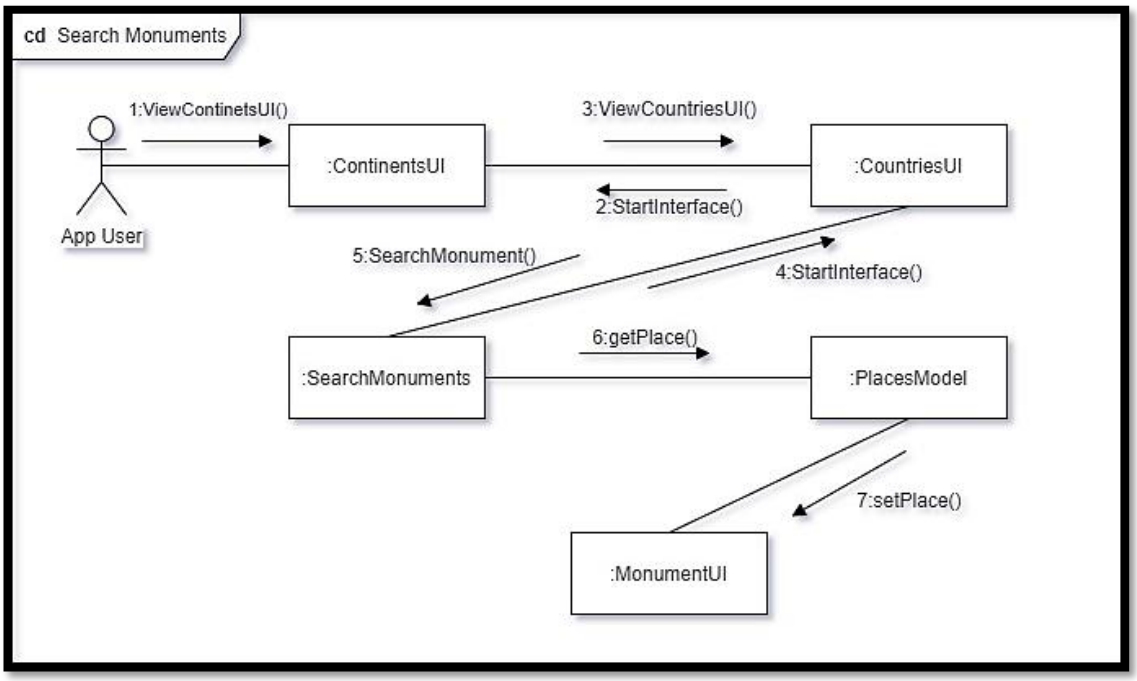
Collaboration:

⇒ Presenting the collaborations, between the objects, that complete this task, with the objective of finding a respective monument.



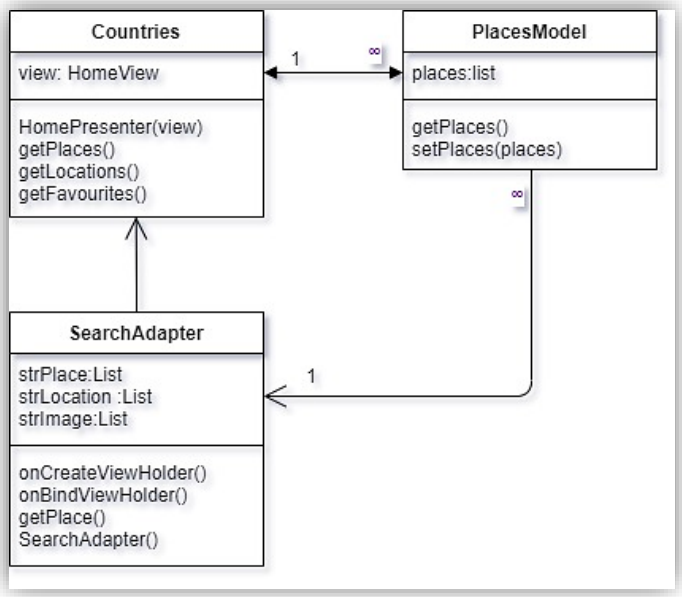
Communication Diagram:

⇒ An extension of the previous diagram, this communication diagram demonstrates not just the objects and their relations, but also the messages that travel from one object to another, in order to complete the respective task.



Semi-Class Diagram:

⇒ After describing both the relations and the messages shared between the objects, in order to achieve the respective task, it is possible to create a Semi-class diagram that follows the respective coding classes, of this feature.



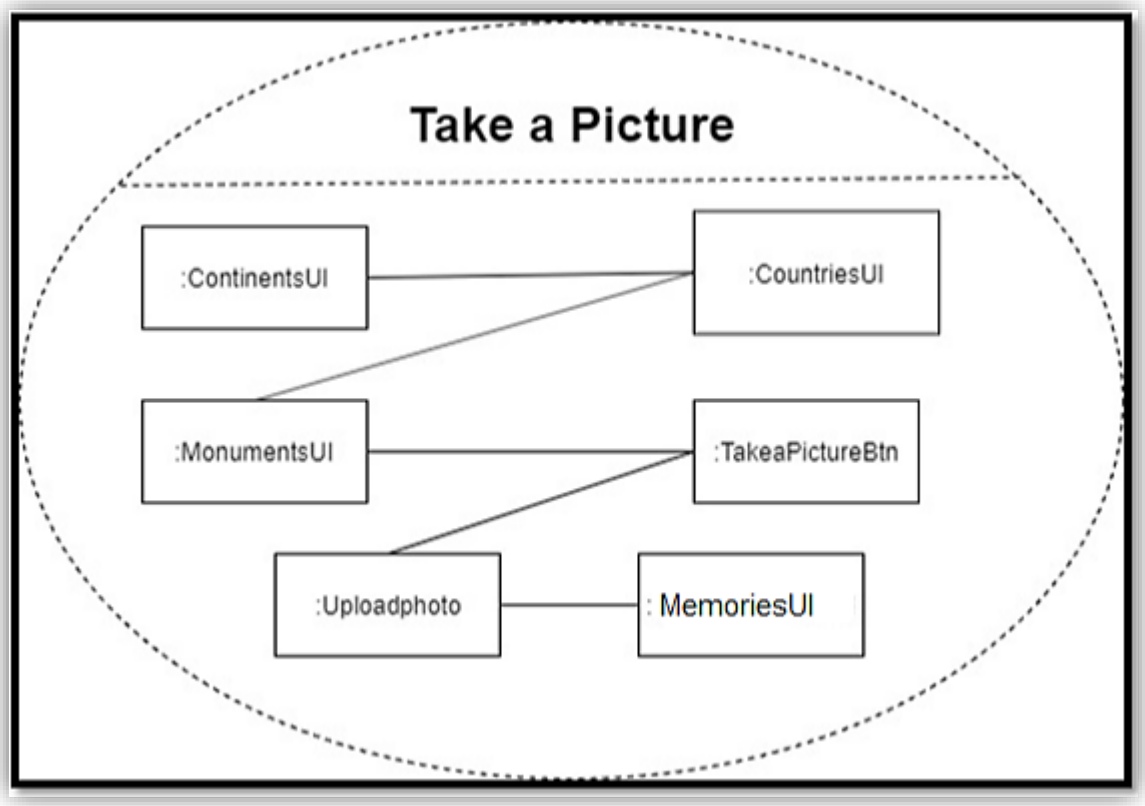
FR.10- 'Take a Picture' 5C Approach:

Use Case:

⇒ A simplified use case diagram of the specific task.

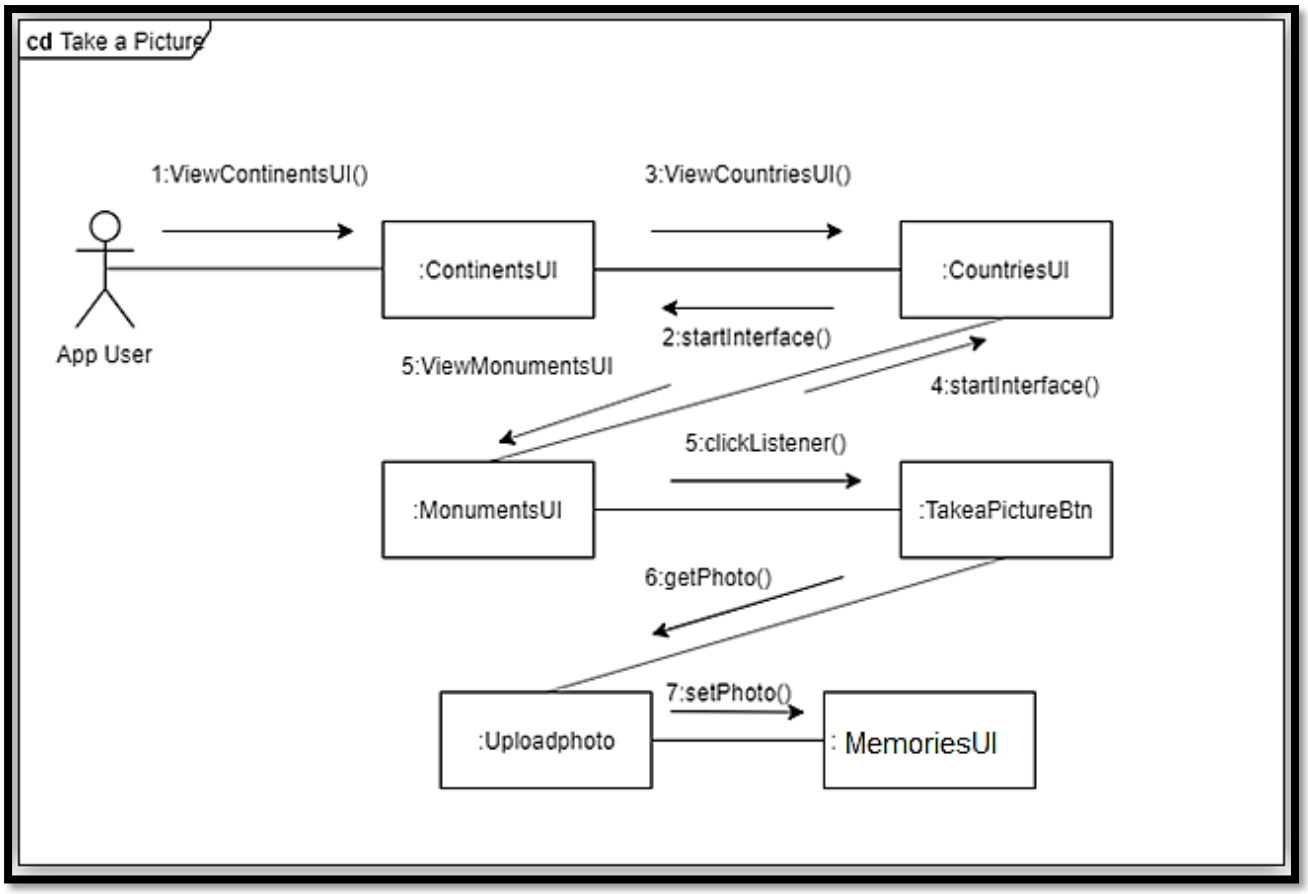
Collaboration:

⇒ Showing the collaborations, between objects, needed to perform the specific task, 'Take a Picture', and a possible following task, upload a picture to create a memory.



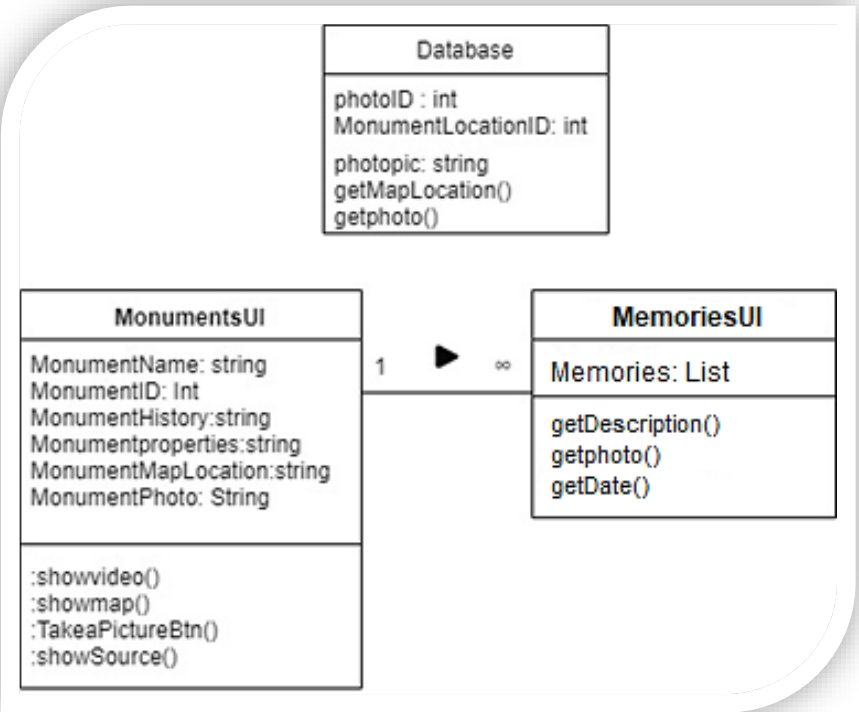
Communication Diagram:

⇒ An extension of the previous diagram, this communication diagram demonstrates not just the objects and their relations, but also the messages that travel from one object to another, in order to complete the respective task.



Semi-Class Diagram:

⇒ After describing both the relations and the messages shared between the objects, in order to achieve the respective task, it is possible to create a Semi-class diagram that follows the respective coding classes, of this feature.

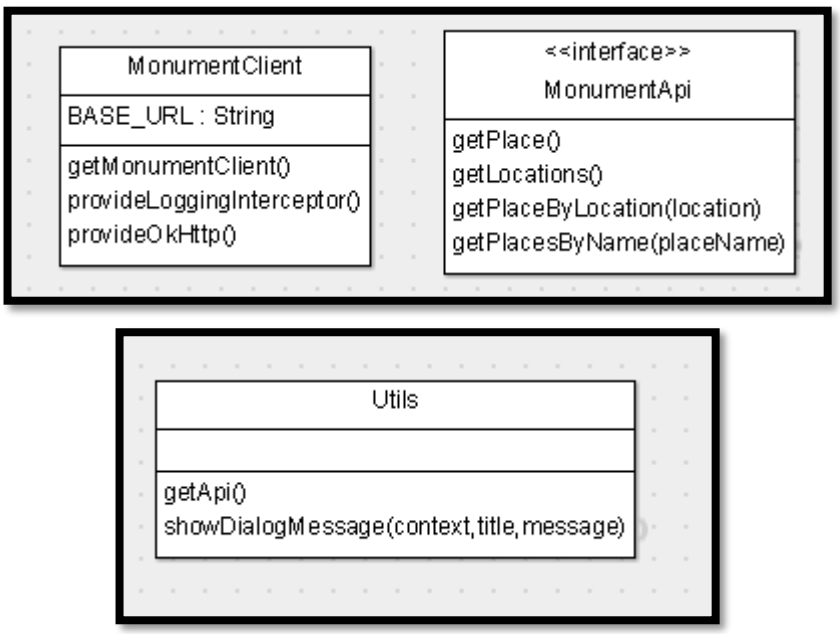


Object Identification & Class Diagram

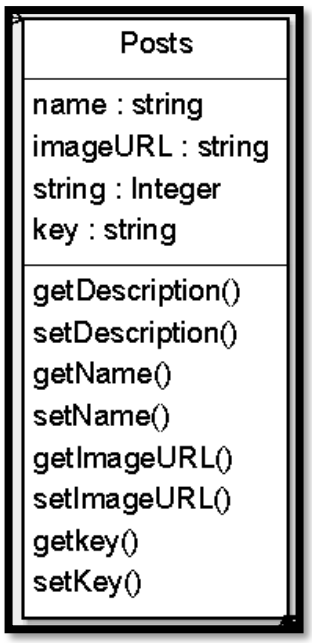
A class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

This information was collected from the requirements and the analyse of the system's use cases, throughout the 5C approach, and assembled according to the design patterns, described in the 'Methodology' section. This design of the system does not have the '*Continents Menu'* implemented.

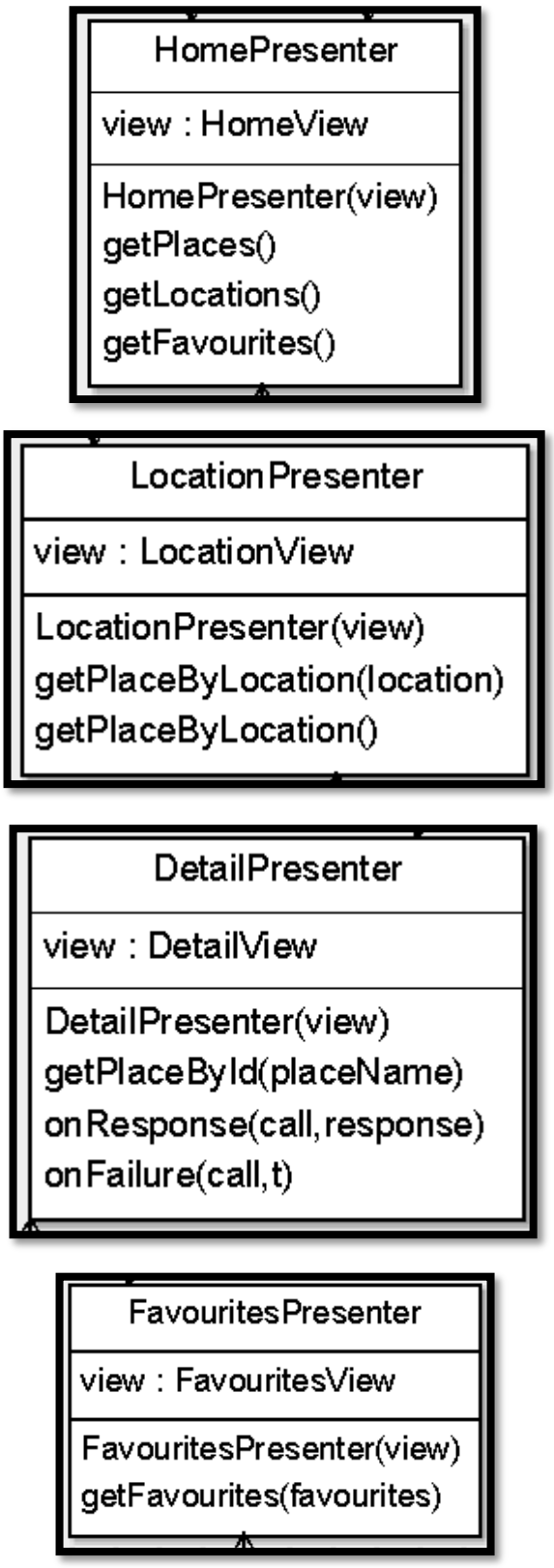
API Dedicated Classes:



Model Classes:

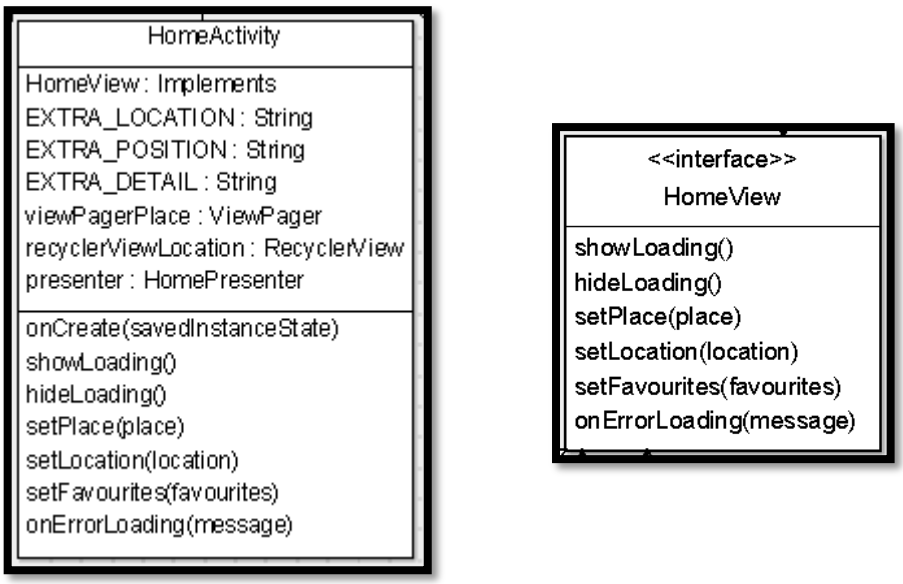


Presenter Classes:

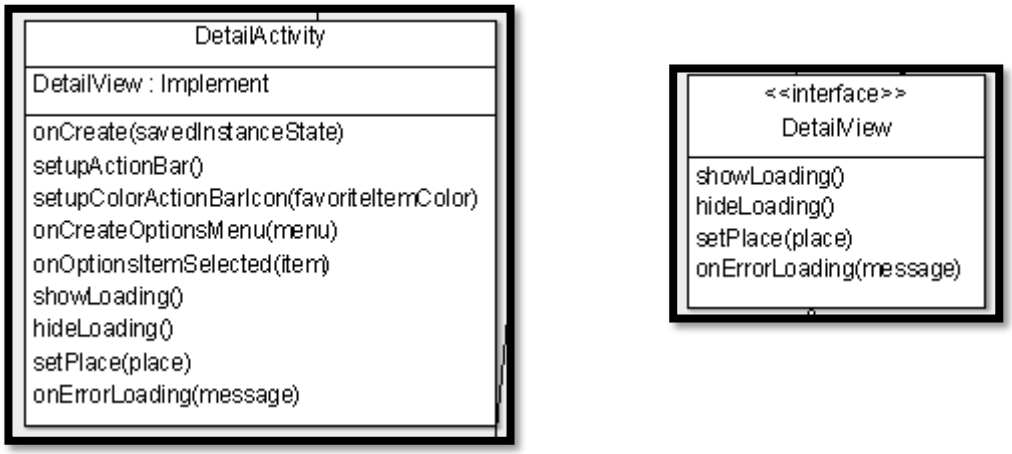


Activity & View Classes:

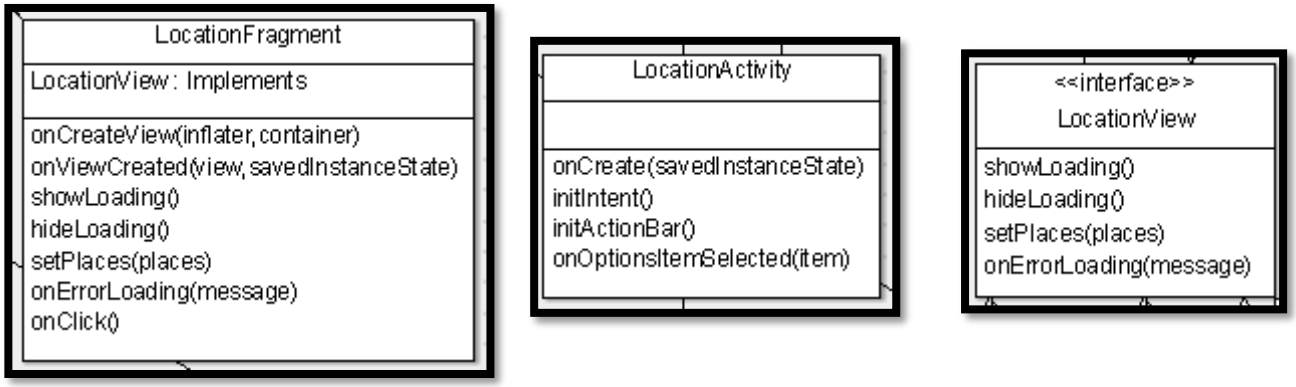
Home



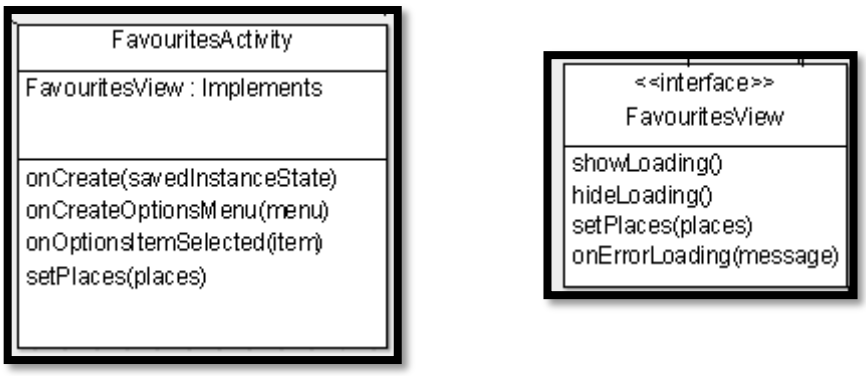
Details



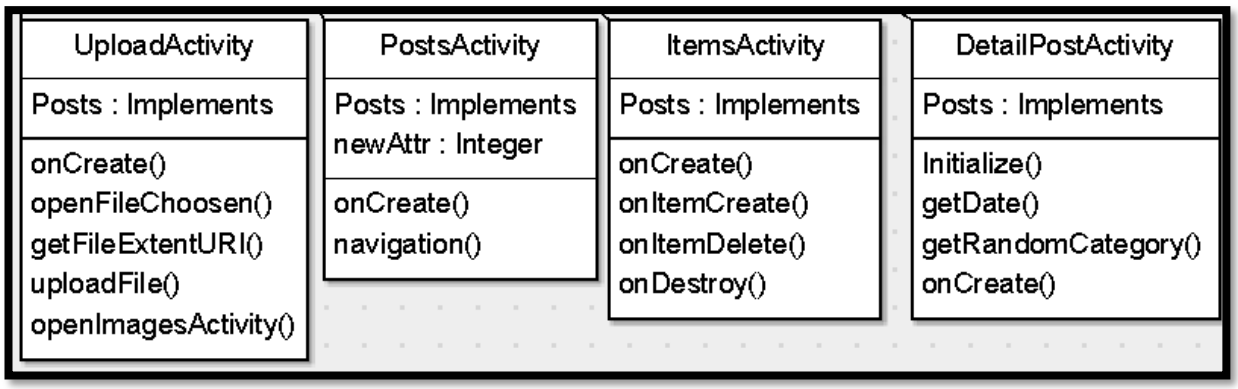
Location



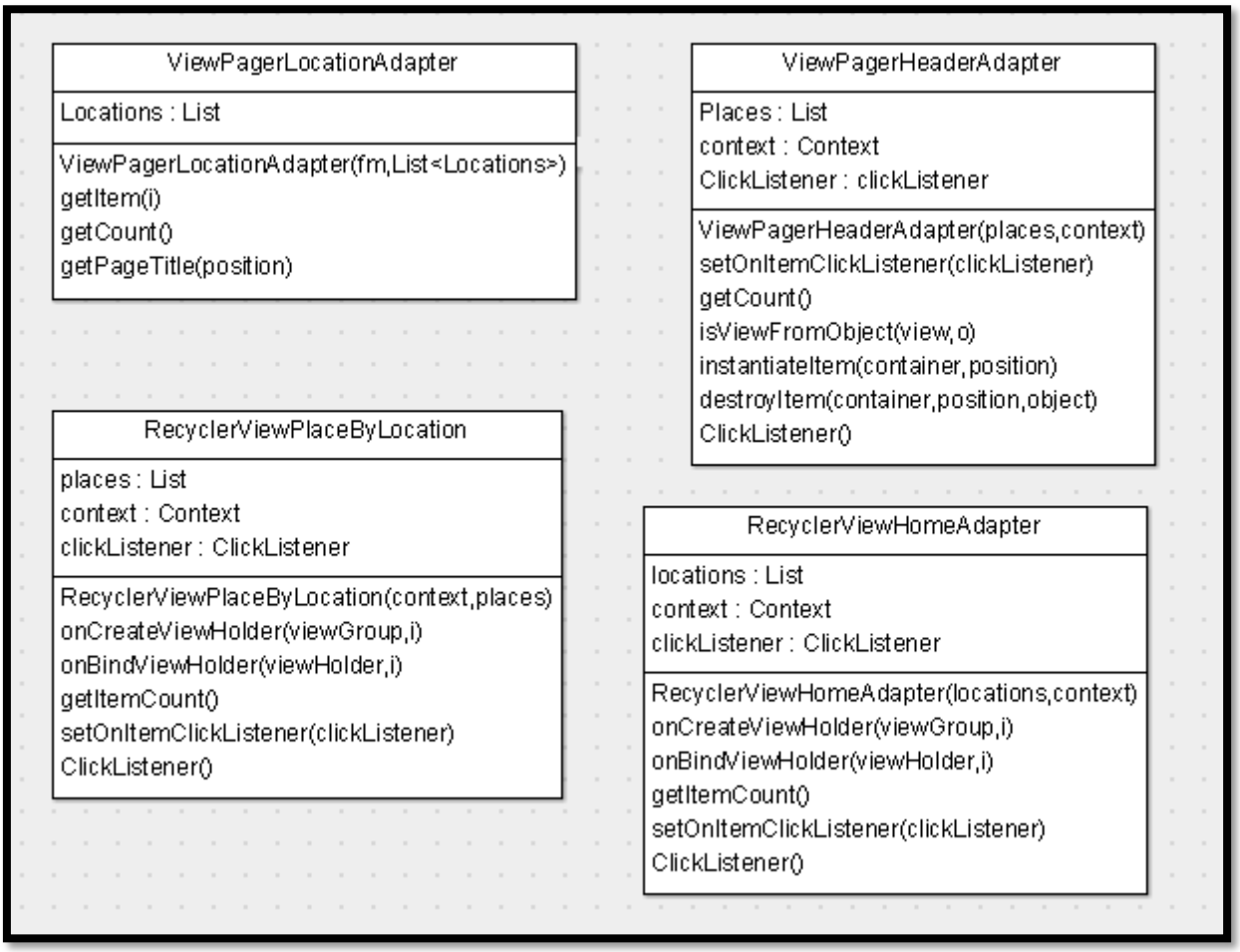
Favourites

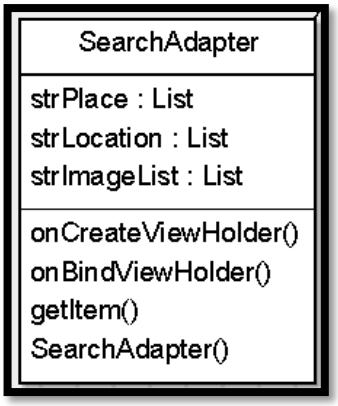


Posts



Adapter Classes:

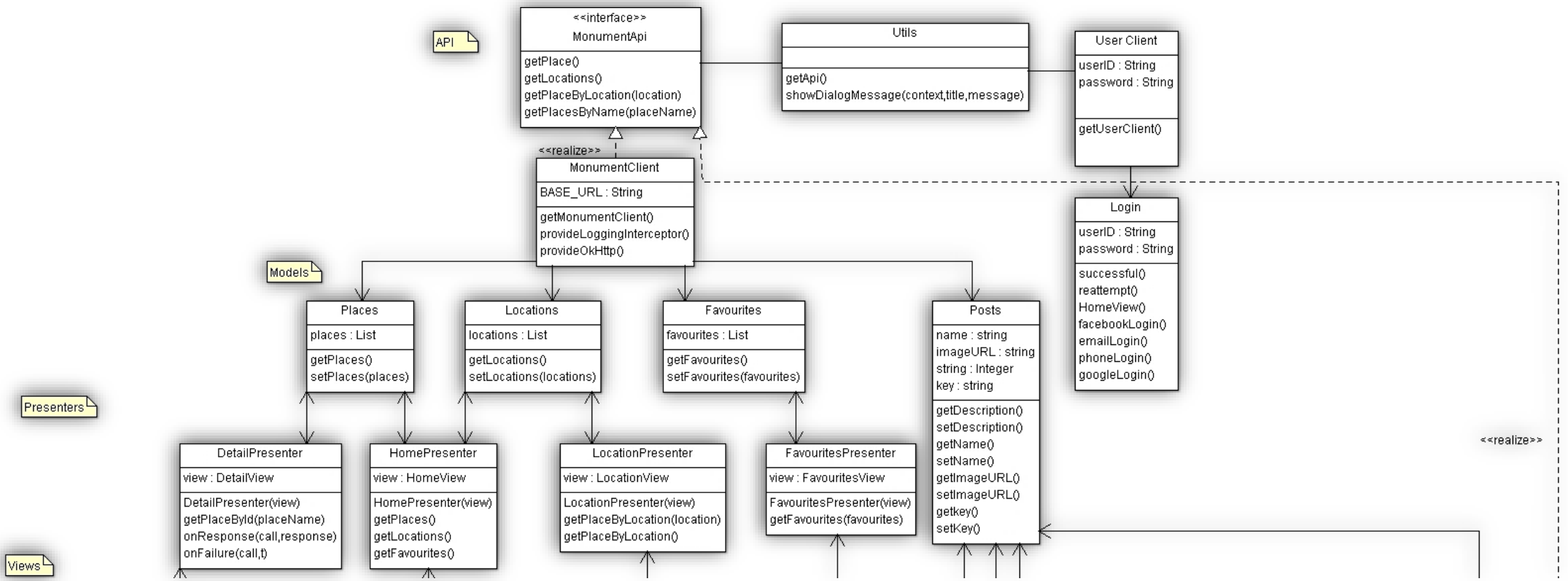


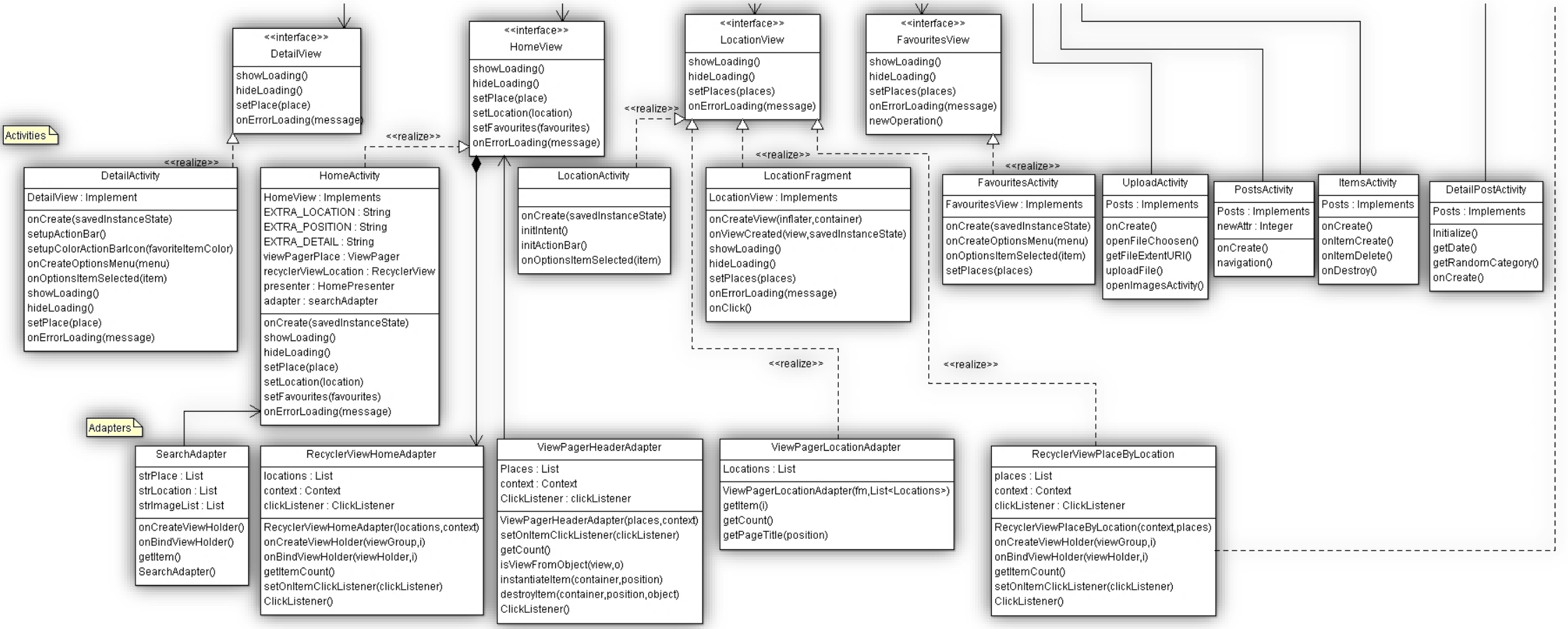


Requirements Location in Classes:

HomePresenter	Countries Menu + Most Famous Monuments + Search Monuments;
LocationPresenter	Monuments Menu + Country Info;
DetailsPresenter	Monument's Info + Monument's Video + Information Source + Monument's Map Location +Take a Picture;
DetailPostActivity	View Memories;
UploadActivity	Create Memories;
Places =	Monuments;
Location =	Countries;
Posts =	Memories;

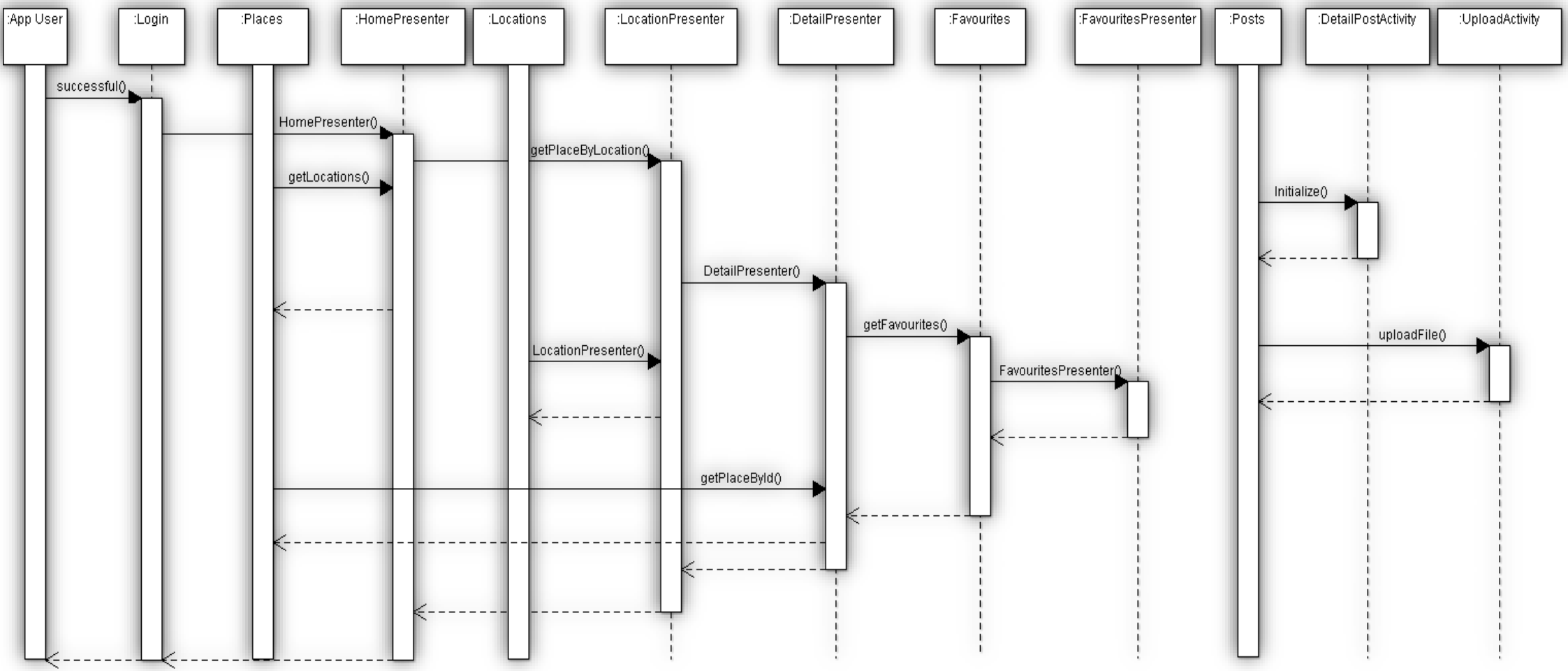
System Class Diagram




Activities
<<interface>>
DetailView
showLoading()
hideLoading()
setPlace(place)
onErrorLoading(message)
<<interface>>
HomeView
showLoading()
hideLoading()
setPlace(place)
setLocation(location)
setFavourites(favourites)
onErrorLoading(message)
<<interface>>
LocationView
showLoading()
hideLoading()
setPlaces(places)
onErrorLoading(message)
<<interface>>
FavouritesView
showLoading()
hideLoading()
setPlaces(places)
onErrorLoading(message)
newOperation()
<<realize>>
DetailActivity
DetailView : Implement
onCreate(savedInstanceState)
setupActionBar()
setupColorActionBarIcon(favoriteItemColor)
onCreateOptionsMenu(menu)
onOptionsItemSelected(item)
showLoading()
hideLoading()
setPlace(place)
onErrorLoading(message)
HomeActivity
HomeView : Implements
EXTRA_LOCATION : String
EXTRA_POSITION : String
EXTRA_DETAIL : String
viewPagerPlace : ViewPager
recyclerViewLocation : RecyclerView
presenter : HomePresenter
adapter : searchAdapter
onCreate(savedInstanceState)
showLoading()
hideLoading()
setPlace(place)
setLocation(location)
setFavourites(favourites)
onErrorLoading(message)
LocationActivity
onCreate(savedInstanceState)
initIntent()
initActionBar()
onOptionsItemSelected(item)
LocationFragment
LocationView : Implements
onCreateView(inflater,container)
onViewCreated(view,savedInstanceState)
showLoading()
hideLoading()
setPlaces(places)
onErrorLoading(message)
onClick()
FavouritesActivity
FavouritesView : Implements
onCreate(savedInstanceState)
onCreateOptionsMenu(menu)
onOptionsItemSelected(item)
setPlaces(places)
UploadActivity
Posts : Implements
onCreate()
openFileChoosen()
getFileExtentURI()
uploadFile()
openImagesActivity()
PostsActivity
Posts : Implements
newAttr : Integer
onCreate()
navigation()
ItemsActivity
Posts : Implements
onCreate()
onItemCreate()
onItemDelete()
onDestroy()
DetailPostActivity
Posts : Implements
Initialize()
getDate()
getRandomCategory()
onCreate()
Adapters
SearchAdapter
strPlace : List
strLocation : List
strImageList : List
onCreateViewHolder()
onBindViewHolder()
getItem()
SearchAdapter()
RecyclerViewHomeAdapter
locations : List
context : Context
clickListener : ClickListener
RecyclerViewHomeAdapter(locations,context)
onCreateViewHolder(viewGroup,i)
onBindViewHolder(viewHolder,i)
getItemCount()
setOnItemClickListener(clickListener)
ClickListener()
ViewPagerHeaderAdapter
Places : List
context : Context
ClickListener : clickListener
ViewPagerHeaderAdapter(places,context)
setOnItemClickListener(clickListener)
getCount()
isViewFromObject(view,o)
instantiateItem(container,position)
destroyItem(container,position,object)
ClickListener()
ViewPagerLocationAdapter
Locations : List
ViewPagerLocationAdapter(fm,List<Locations>)
getItem(i)
getCount()
getPageTitle(position)
RecyclerViewPlaceByLocation
places : List
context : Context
clickListener : ClickListener
RecyclerViewPlaceByLocation(context,places)
onCreateViewHolder(viewGroup,i)
onBindViewHolder(viewHolder,i)
getItemCount()
setOnItemClickListener(clickListener)
ClickListener()

System Sequence Diagram:

⇒ Sequence Diagrams are dynamic models that show the sequence of object interactions. The following diagram represents the interaction of the user with the application and how each '*Presenter'* receives the required information for its respective *'Model',* also showing the procedure of creating and uploading a *'Post'* (***FR.15*** & ***FR.16***).



Project Development

Introduction

In this stage of the software lifecycle, occurs the development and coding of the application.

This section, divided into two parts, breaks down the several increments of the front-end development and explains the implementation, connection and creation of the back-end database, concluding in a fully operational system.

1. Front-End Development

The user experience, of any mobile application, starts with the development of the visual and interactive part of the system, known as 'front-end'.

The *Front-end Development* of this system, planned thru increments, demonstrates the functionalities of the application and how such functions were planned, designed, developed and implemented.

The 'front-end' reflects the research done in the '*Literature Survey'* and '*Methodology*' sections, including the use of the MVA and MVP design patterns, and how such methods influenced the flow and success, of the process.

It also clarifies, how the front-end connects itself with the back-end, and how it interacts with the back-end data.

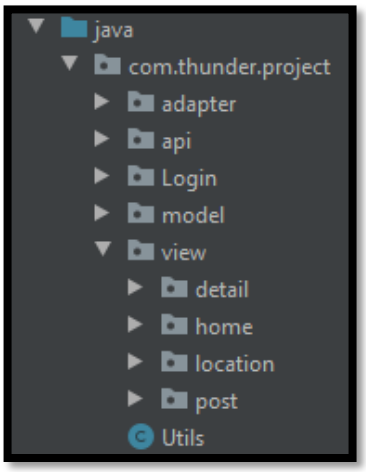


Figure 27-Model View Adapter Design Pattern

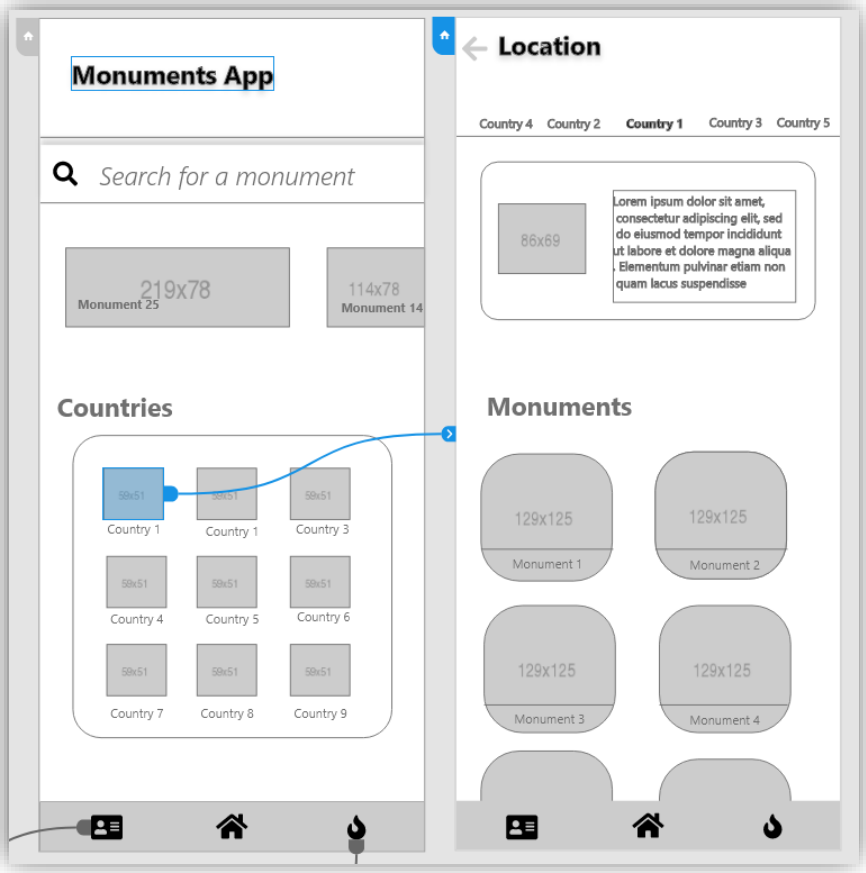
1st Increment

The first Increment represents the beginning of the development phase, intended to build the 'Home View', of a pre-chosen continent, and the 'Location View' of the chosen country, representing the features present at those views, developing the requirements FR.3 (Countries Menu) and FR.4 (Countries Info).

Present Requirements	Status
FR.3- Countries Menu	In Development
FR.4-Countries Info	In Development
FR.5-Monuments Menu	2nd Increment
FR.14-Most Famous Monuments	3rd Increment
FR.18-Search Monuments	4th Increment

Design

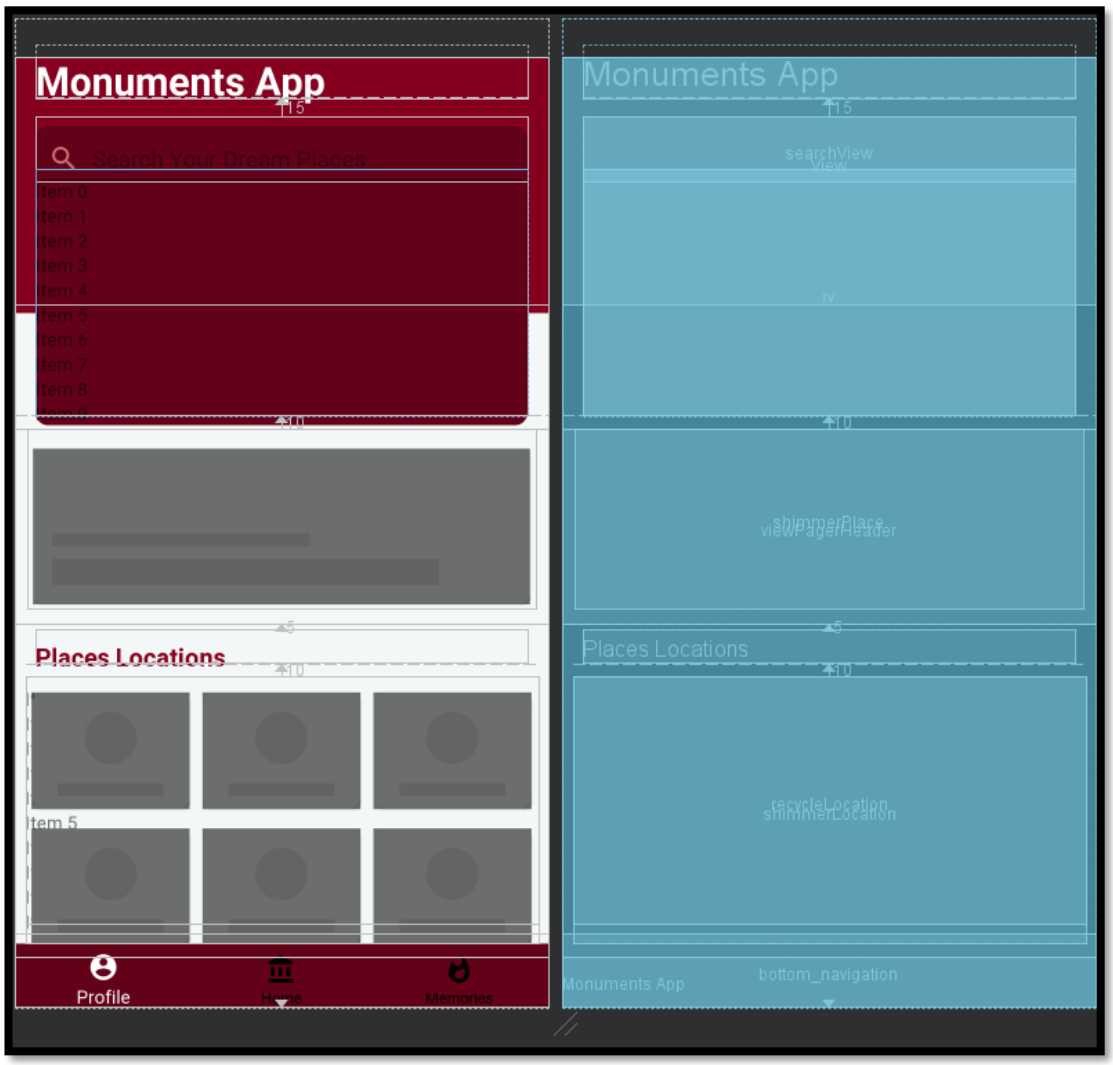
- Based on the first Prototype, the 1st Increment matches the following design:



Home View Blueprint & Layout:

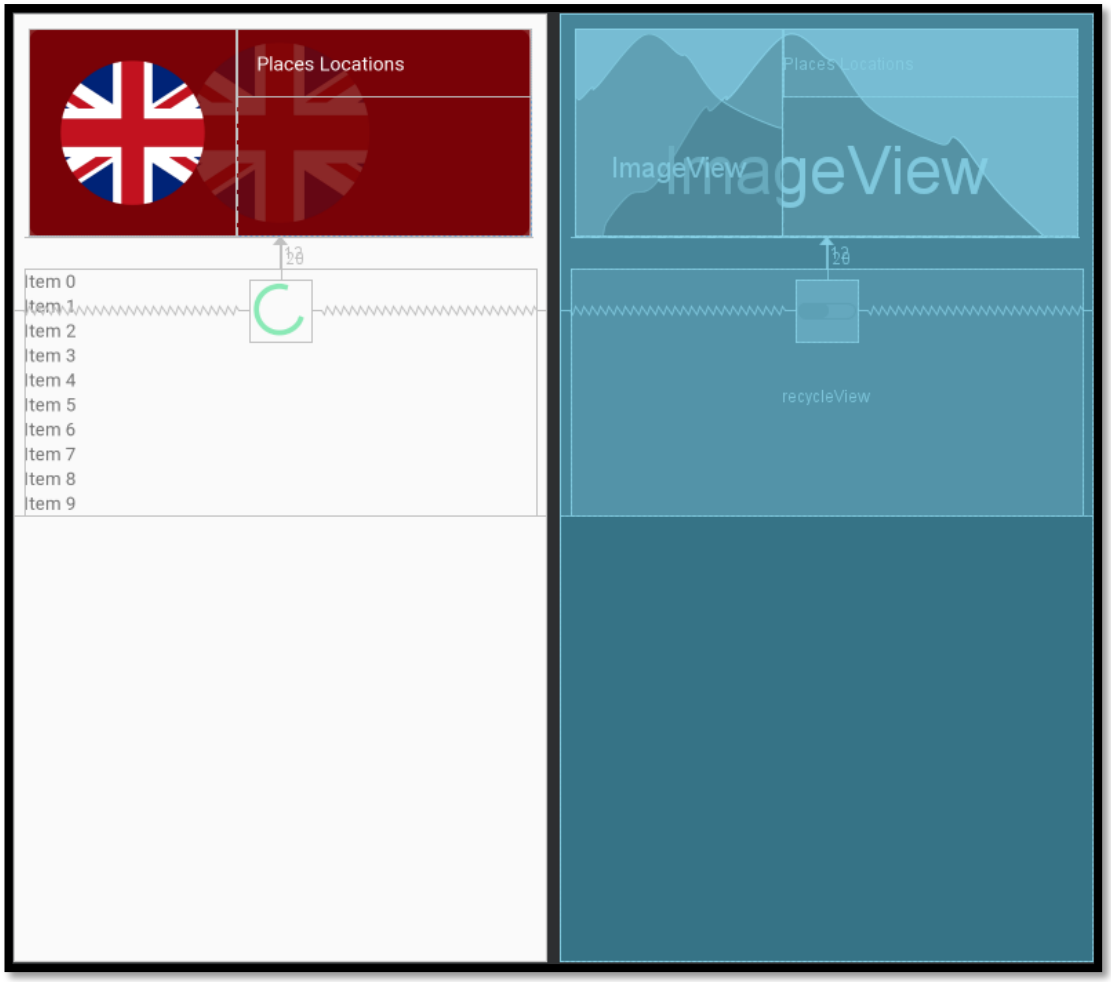
- ‘Activiy_home.xml’ is embedded within a Coordinator Layout, used to handle the present animated transitions inside, focusing the rest of the design inside a Nested Scroll View, which encapsulates several Card Views, TextViews and an EditTextView, ‘Search’, as well as a bottom navigation bar.

- For the first requirement of this Increment (FR.3), the primary layout feature, based on the development of another layout, ‘item_recycler_location_shimmer.xml’, that fills itself with views, provided by the chosen layout manager, creating a list of countries provided by the backend API.



Location View Blueprint & Layout:

- 'Fragment_location.xml' displays the second requirement of this increment (FR.4), the same embedded in a Nested Scroll View that contains the image view and text views required to display the information about the chosen country, collected from the 'Locations' Model, and then recycled into the views. (Muttaqin, 2019)



Development

This Increment started with the development of the previous layouts, inspired on the Prototype, followed by the creation of the Home View.

Each View of this design pattern is composed of an Activity, a Presenter and a View. Following the example of the Home View, composed by 'HomeActivity.java' class, that implements the 'HomeView.java', public interface, and deals with the functionalities of the Home View. The 'HomePresenter.java' class that calls the Models, required to get information from the API, and the 'HomeView.java', public interface, that calls the functions that operate at this View.

Then, 'RecyclerViewHomeAdapter.java' was developed, to recycle the data from the models and send it to the Views, in this case, the Home View.

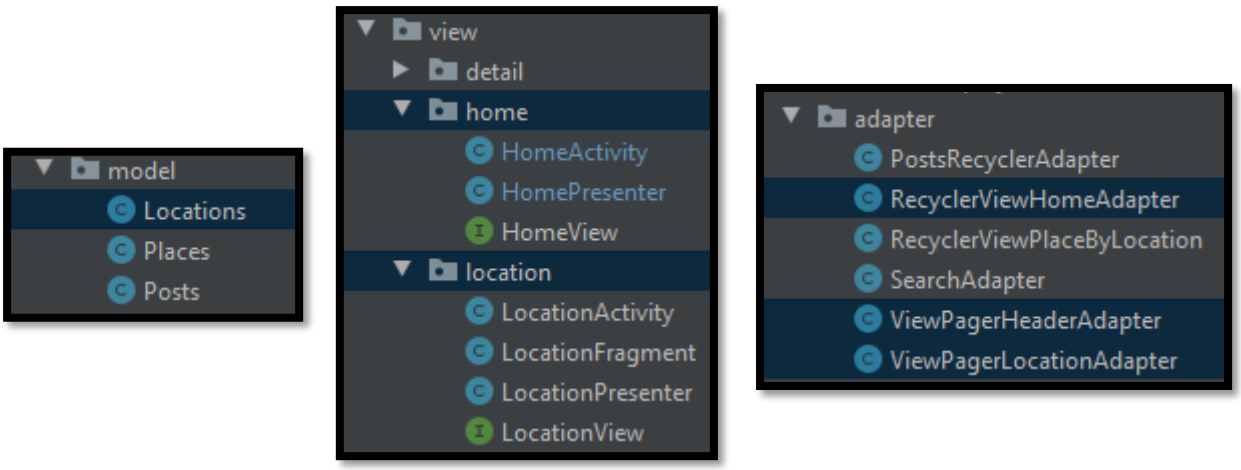
In this system, views like 'HomeView.java' class represent the OO concept of abstraction. In Java, these interfaces/abstract classes hide the internal implementation of the application. At this level of abstraction, only method signatures are defined, and each class implements them in their own way. The use of this OO concept helps to avoid repetitive code, provides flexibility and hides the underlying complexity of data. (Monus, 2018)

```
public interface HomeView {

    void showLoading();
    void hideLoading();
    void setPlace(List<Places.Place> place);
    void setLocation(List<Locations.Location> location);
    void onErrorLoading(String message);
}
```

Figure 28-HomeView Interface Class

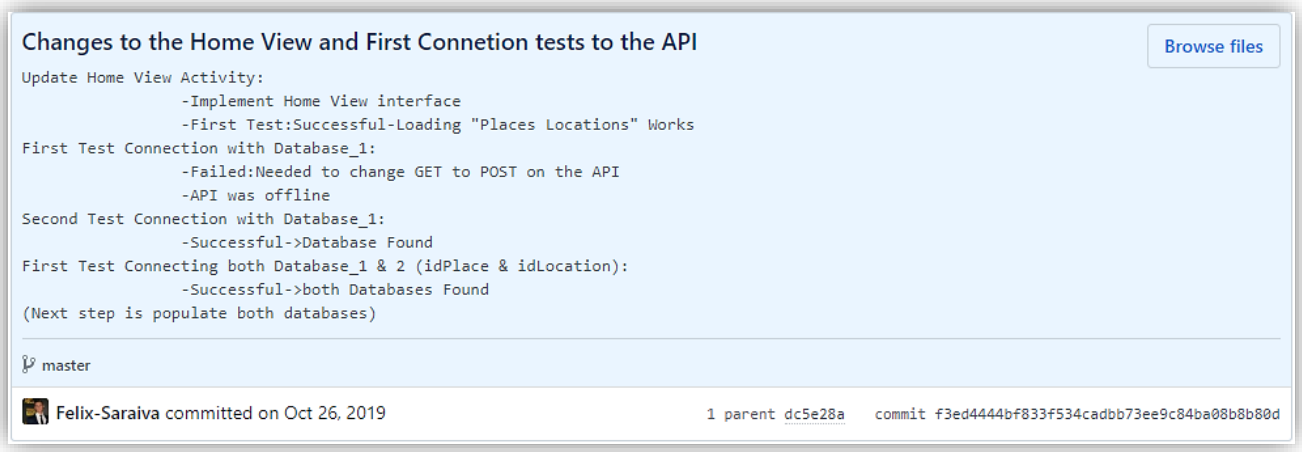
A similar process occurs in the second requirement, except for the extra 'LocationFragment.java' class that works after a country is selected on the Home View and gets the country's image and description and displays it into the respective View, after the same converts itself in the respective adapter.

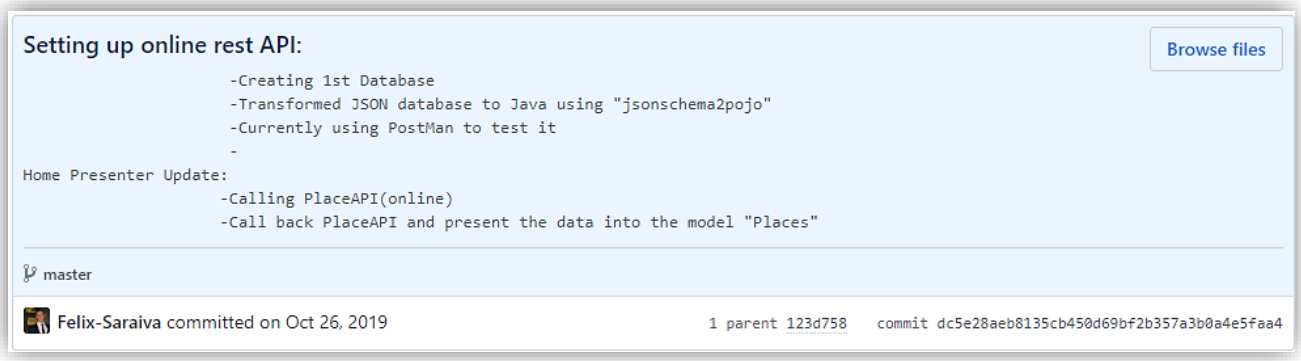
This development is present in Appendix B, pages 227 and 231, where the functions 'setLocation()', in 'HomeActivity.java', and 'onViewCreated()', on 'LocationFragment.java', particularly represent the functionality of this Increment.

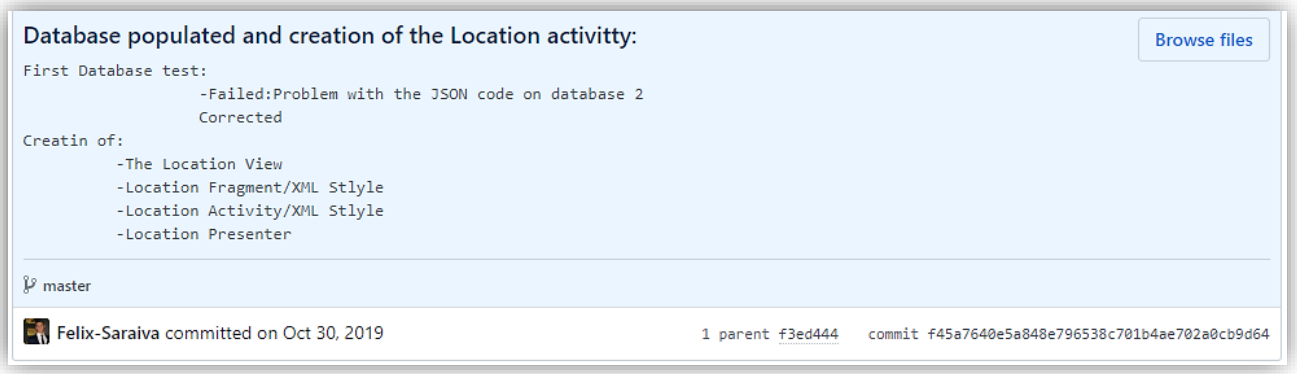


Implementation

1st Increment GitHub Commits:







Version Control:

Pre-release

V1.0

4cf39c1

Compare

1st Increment

Felix-Saraiva released this now

Development Done:

-Started projects base Code;
-Activity home layout Update;
-Setting up online rest API:

```
-Creating 1st Database;
-Transformed JSON database to Java using "jsonschema2pojo";
-Currently using PostMan to test it;
```

-Home Presenter Update:

```
-Calling PlaceAPI(online);
-Call back PlaceAPI and present the data into the model "Places";
```

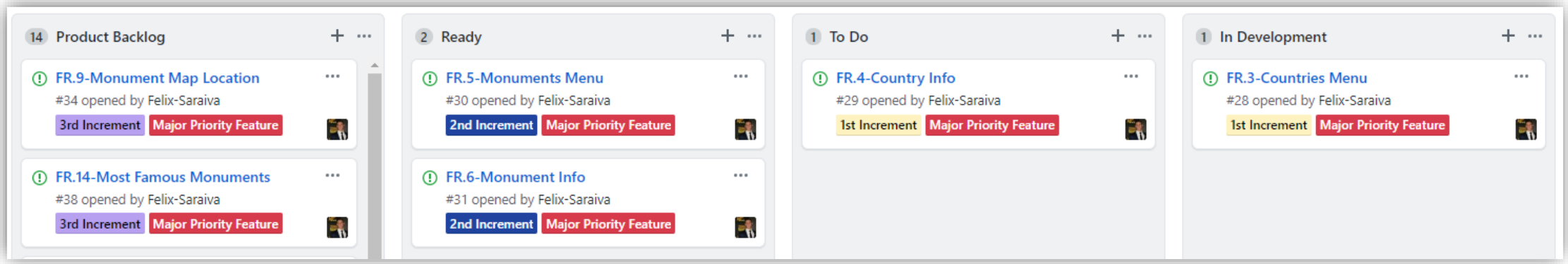
-Changes to the Home View and First Connection tests to the Mock Up API;
-Update Home View Activity:

```
-Implement Home View interface;
-First Test:Successful-Loading "Places Locations" Works;
```

-First Test Connecting both Database_1 & 2 (idPlace & idLocation);
-Database populated and creation of the Location activity;
-Creation of:

```
-The Location View;
-Location Fragment/XML Style;
-Location Activity/XML Style;
-Location Presenter.
```

Scrumban Board Progress:



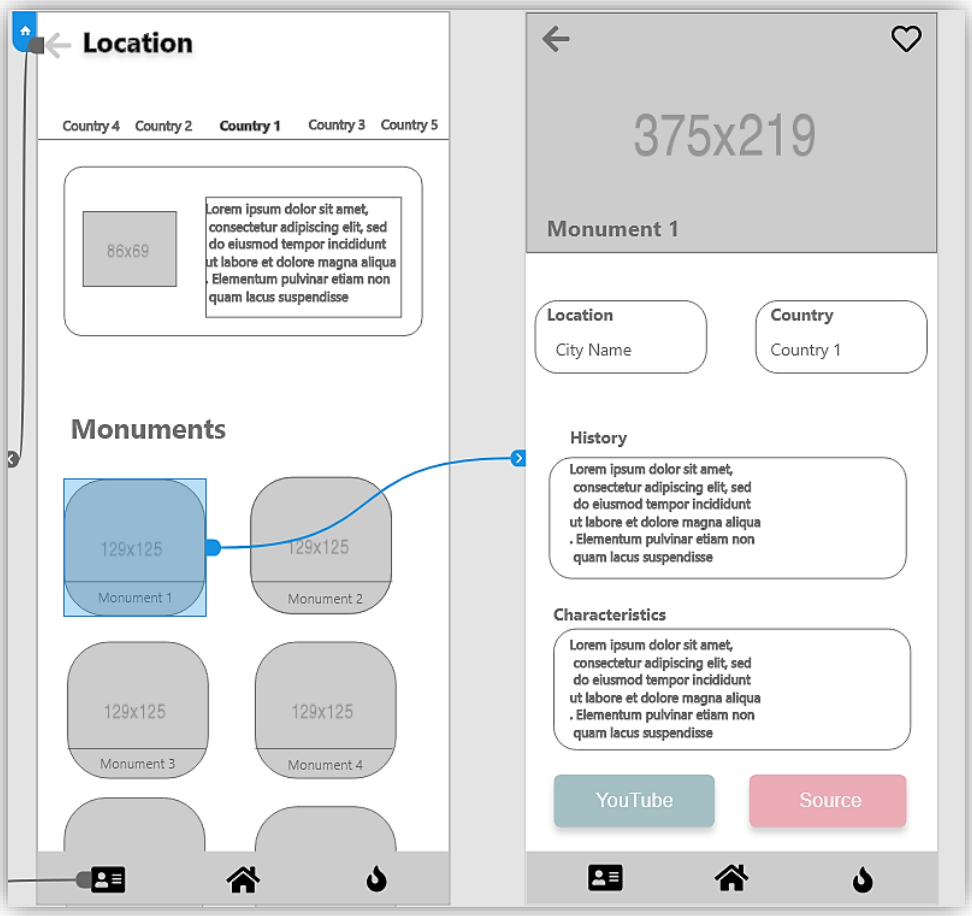
2nd Increment

The second Increment, very similar to the first one, holds the objective of developing requirement FR.5 (Monuments Menu) and FR.6 (Monument's Info), implying the creation of a monuments list on the '*Location View*' and a new '*Details View*', which opens a new monument's page chosen from the monuments list.

Present Requirements	Status
FR.4-Countries Info	Developed
FR.5-Monuments Menu	In Development
FR.6-Monument's Info	In Development
FR.7- Monument's Video	5th Increment
FR.8- Information Source	5th Increment
FR.17-Favourites	10th Increment

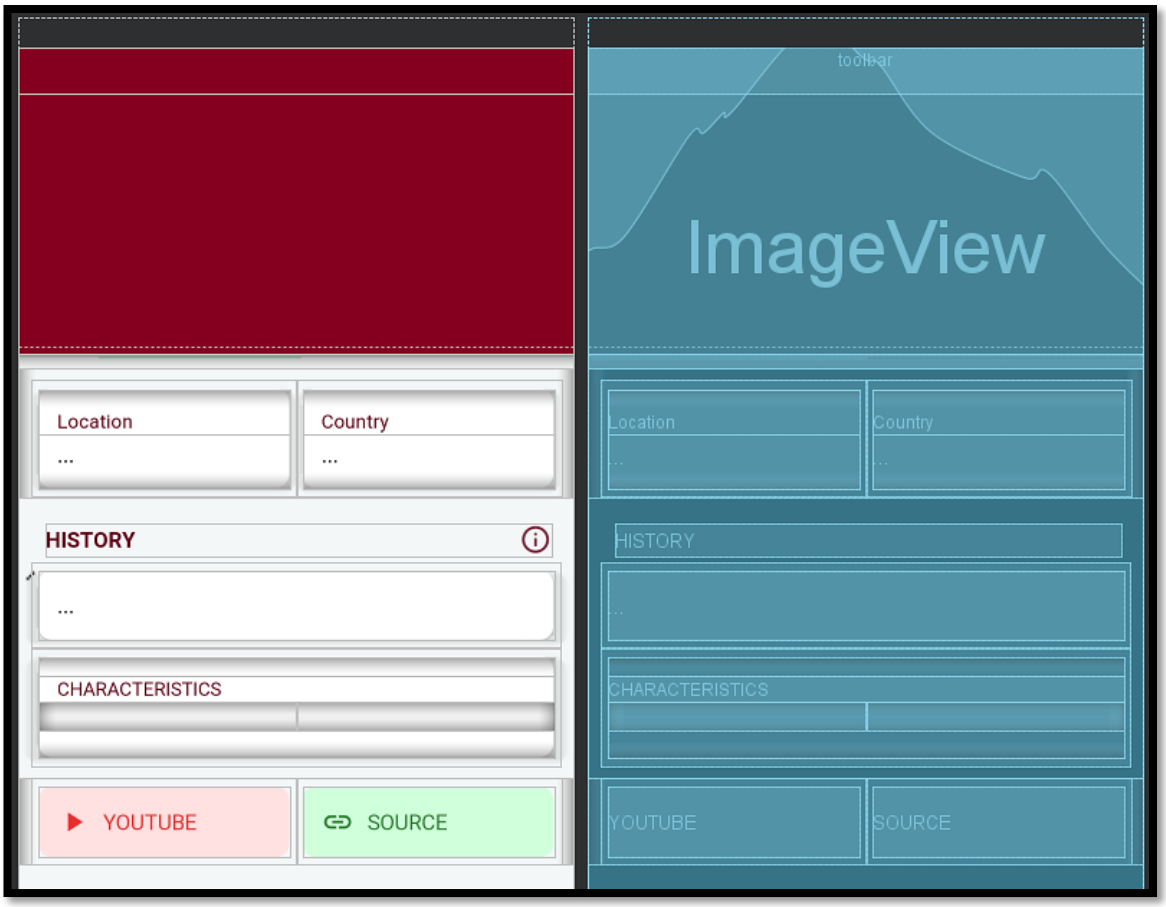
Design

Based on the first Prototype, the 2nd Increment matches the following design:



Details View Blueprint & Layout:

- The 'activity_detail.xml' layout, responsible for the display of all the features, of each monument, is embedded within two layouts, a Coordinator Layout and a Nested Scroll View, both explained in the previous increment, and it contains all the text and image views that the individual requirements need. It will also suffer upgrades in future increments, to implement new features. (Muttaqin, 2019)



Development

In order to develop this Increment, a similar approach to the previous one took place.

The creation of the Detail View also holds a Model, a View, a Presenter, an Activity and a Recycler View adapter, that organizes monuments by their countries, FR.5, represented in more depth in page 216 of appendix B, the function 'setPlace()', in 'HomeActivity.java', shows the creation of a list of monuments (FR.5) and in page 218 the function 'setPlace(Places.Place place)', displays the information of a chosen monument, in the respective Details View.

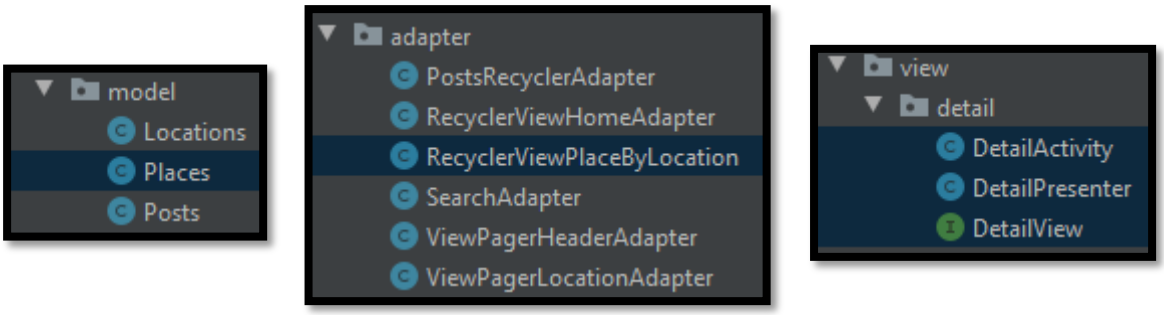
These public 'get' and 'set' methods are a representation of the OO concept of Encapsulation in this system. Taking the 'Places.java' model class as an example, Encapsulation is present by keeping the class variables private and providing these public getter and setter methods to each of the variables.

This design creates loosely coupled classes, and so, data members are protected from system-wide access by restricting direct access, fields are set private, and each field has a getter and setter method. (Monus, 2018)

```
public class Places {

    @SerializedName("places")
    @Expose
    private List<Place> places = null;

    public List<Place> getPlaces() { return places; }

    public void setPlaces(List<Place> places) { this.places = places; }

    public class Place {
        @SerializedName("idPlace")
        @Expose
        private String idPlace;
        @SerializedName("strPlace")
        @Expose
        private String strPlace;
```

Closer to the conclusion of this 2nd Increment, the developer realized that the **5th Increment** could be easily implemented on this stage, of the development process, so thanks to the flexibility of this project's development methodology, Scrumban, it was possible to **pull,** not the 3rd, but the 5th Increment out of the product backlog, respecting, of course, the WIP limit, since only four tasks were present, on the development column.

This 5th Increment based on the creation of FR.7, Monument's Video, and FR.8, Information Source, holds the creation of two buttons, one opens a YouTube video about the respective monument (FR.7) and the other a webpage with more information about the monument (FR.8).

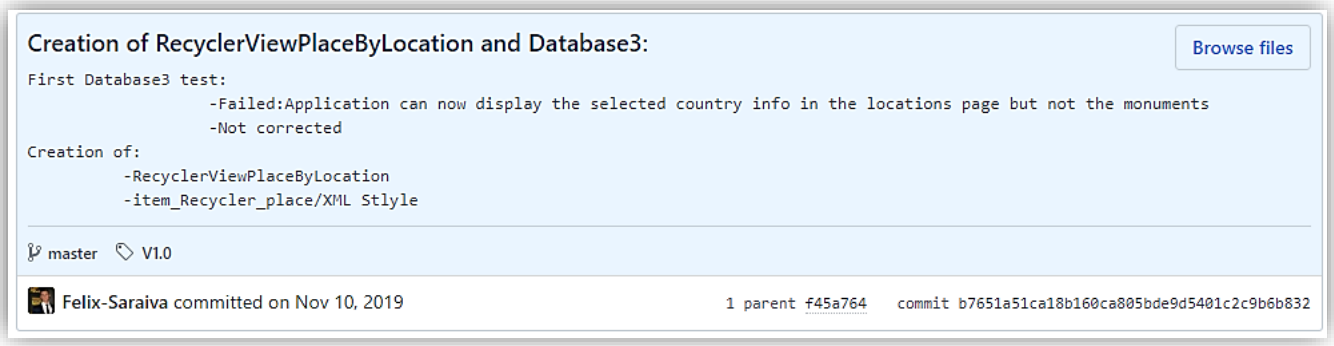



```
youtube.setOnClickListener(v -> {
    Intent intentYoutube = new Intent(Intent.ACTION_VIEW);
    intentYoutube.setData(Uri.parse(place.getStrYoutube()));
    startActivity(intentYoutube);
});

source.setOnClickListener(v -> {
    Intent intentSource = new Intent(Intent.ACTION_VIEW);
    intentSource.setData(Uri.parse(place.getStrSource()));
    startActivity(intentSource);
});
```

5th Increment Functions

Implementation

2nd & 5th Increments GitHub Commits:



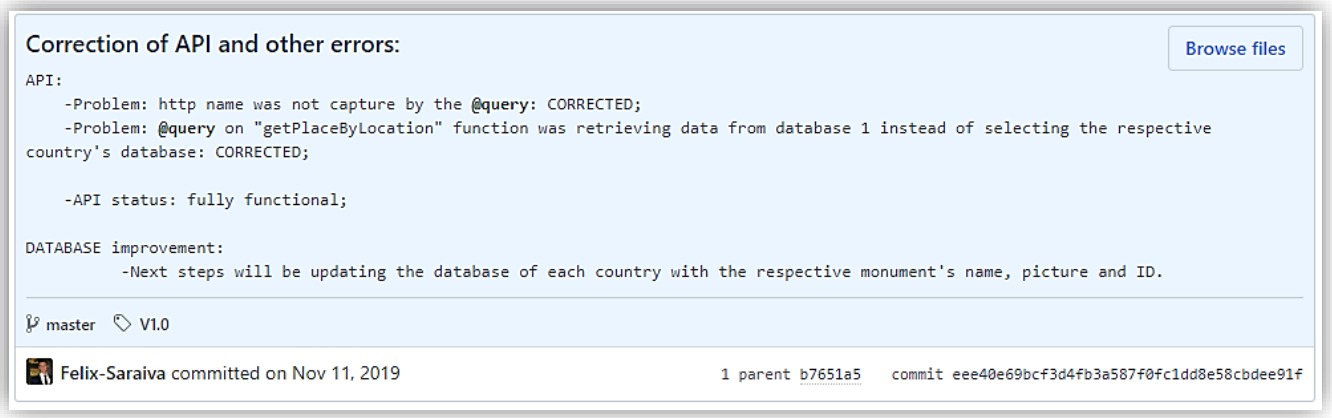
Creation of RecyclerViewPlaceByLocation and Database3:

Browse files

```
First Database3 test:
                -Failed:Application can now display the selected country info in the locations page but not the monuments
                -Not corrected
Creation of:
          -RecyclerViewPlaceByLocation
          -item_Recycler_place/XML Stlyle
```

master V1.0

Felix-Saraiva committed on Nov 10, 2019 — 1 parent f45a764 — commit b7651a51ca18b160ca805bde9d5401c2c9b6b832



Correction of API and other errors:

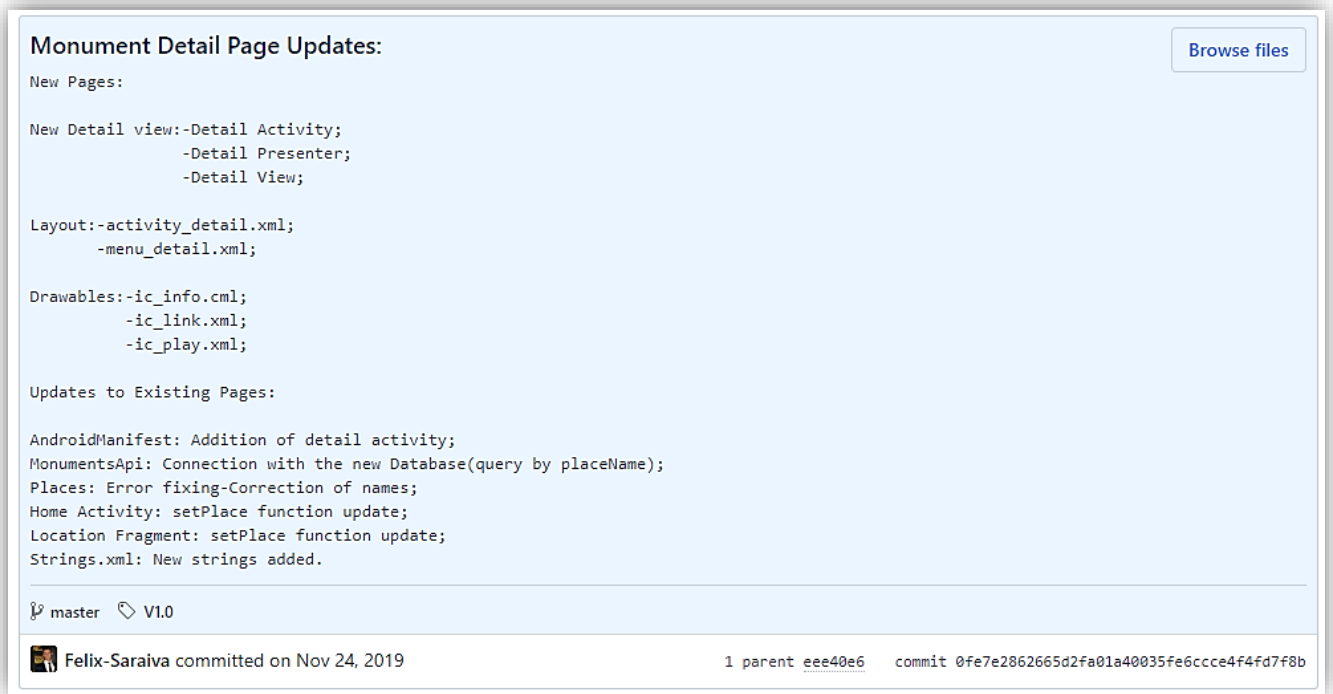
Browse files

```
API:
    -Problem: http name was not capture by the @query: CORRECTED;
    -Problem: @query on "getPlaceByLocation" function was retrieving data from database 1 instead of selecting the respective
country's database: CORRECTED;

    -API status: fully functional;

DATABASE improvement:
          -Next steps will be updating the database of each country with the respective monument's name, picture and ID.
```

master V1.0

Felix-Saraiva committed on Nov 11, 2019 — 1 parent b7651a5 — commit eee40e69bcf3d4fb3a587f0fc1dd8e58cbdee91f

---



Monument Detail Page Updates:

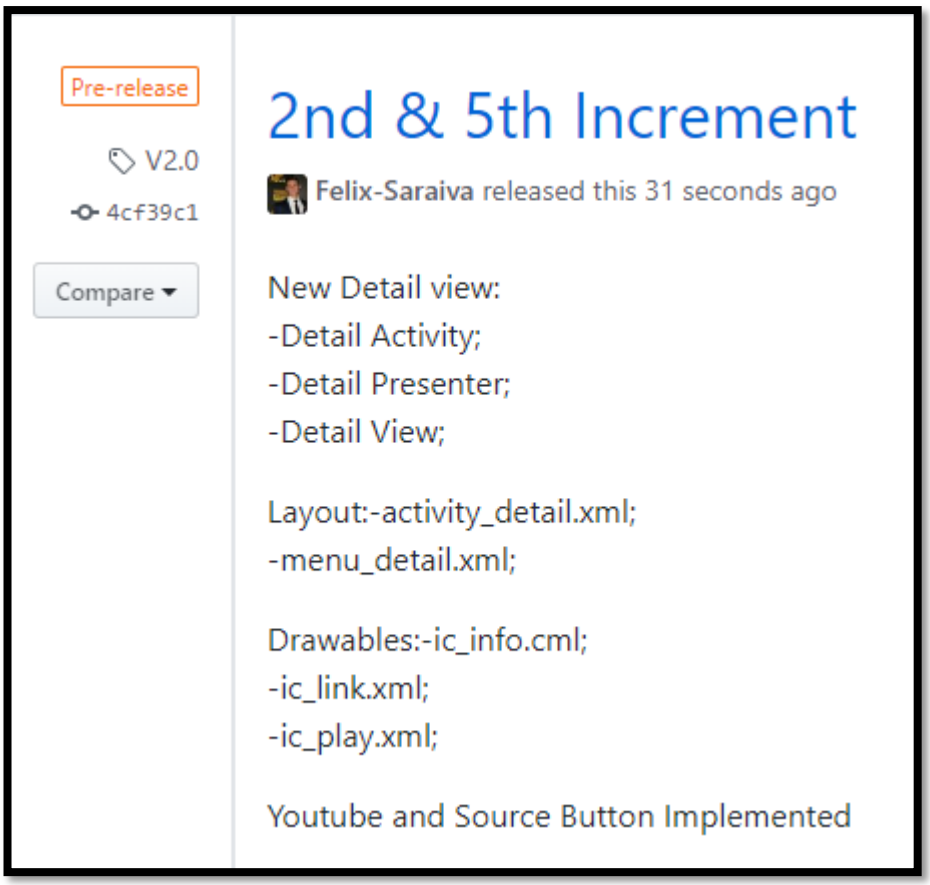
Browse files

```
New Pages:

New Detail view:-Detail Activity;
                -Detail Presenter;
                -Detail View;

Layout:-activity_detail.xml;
       -menu_detail.xml;

Drawables:-ic_info.cml;
          -ic_link.xml;
          -ic_play.xml;

Updates to Existing Pages:

AndroidManifest: Addition of detail activity;
MonumentsApi: Connection with the new Database(query by placeName);
Places: Error fixing-Correction of names;
Home Activity: setPlace function update;
Location Fragment: setPlace function update;
Strings.xml: New strings added.
```

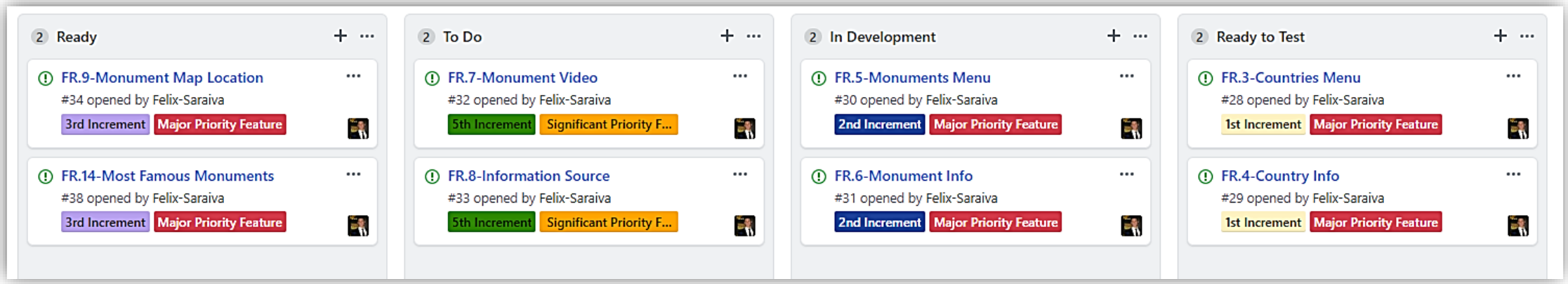
master V1.0

Felix-Saraiva committed on Nov 24, 2019 — 1 parent eee40e6 — commit 0fe7e2862665d2fa01a40035fe6ccce4f4fd7f8b

Version Control:



Scrumban Board Progress:



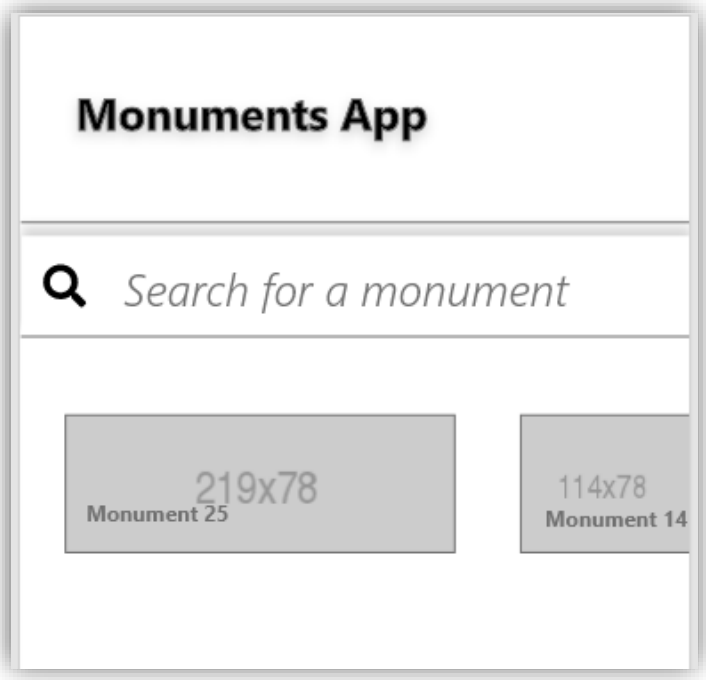
3rd Increment

The 3rd Increment of this development focuses itself on the implementation of a monument's map location (FR.9) and a list of the most famous monuments (FR.14), displayed on the Home view.

Present Requirements	Status
FR.9-Monument's Map Location	In Development
FR.14-Most Famous Monuments	In Development
FR.10-Take a Picture	6th Increment

Design

Based on the first Prototype, this requirement matches the following design:

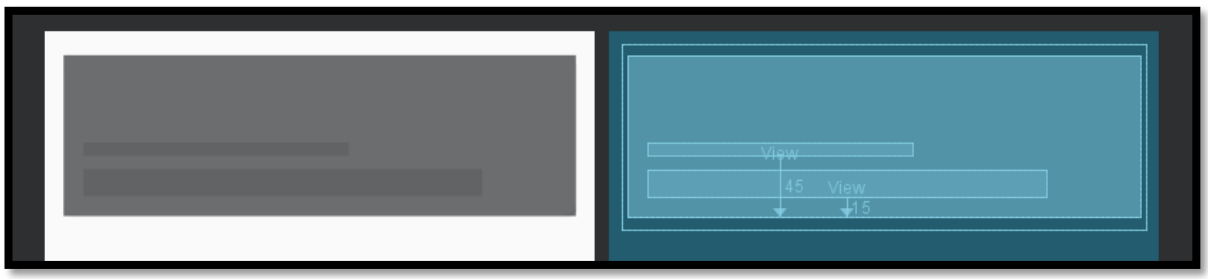


Google Map Support Fragment Blueprint & Layout:

- Within the 'activity_detail.xml' layout, implemented inside a fragment, is a gms Google Maps Support Fragment, used to display the google map that holds the monument's location. This fragment, wrapped around a view of a map, automatically handles the necessary life cycle needs.

Most Famous Monuments Pager Header Blueprint & Layout:

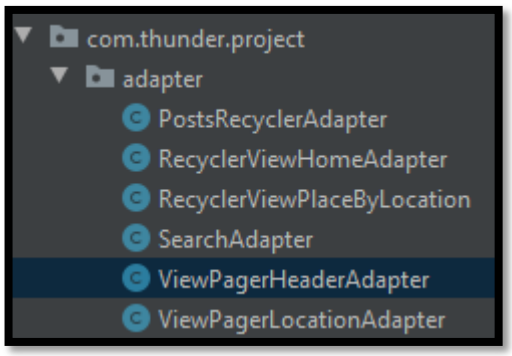
- This requirement was implemented on top of the Home View in a CardView List that presents the most famous monuments of a respective continent. In 'item_view_pager_header_shimmer.xml' using a ShimmerFrameLayout the Monuments are recycled and displayed at the HomeView.



Development

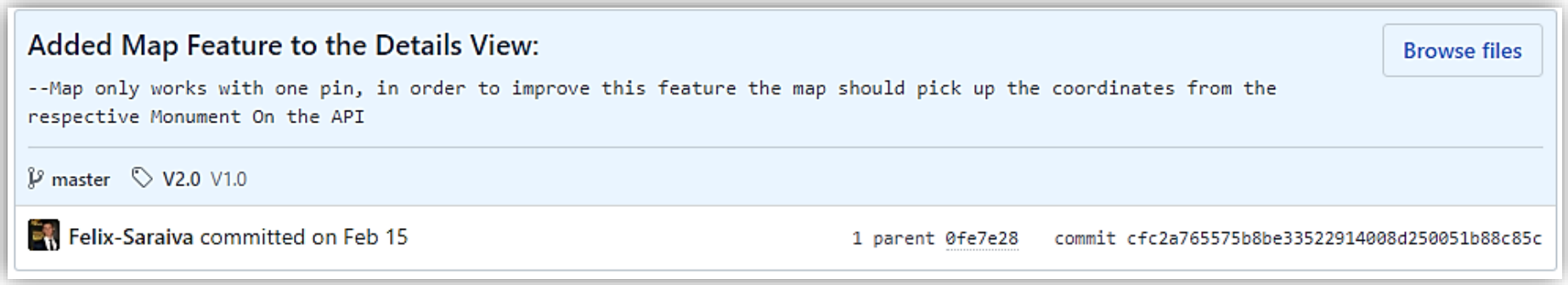
The development of this Increment started with the implementation of the design above, followed by the implementation of a google map, acquired using 'getMapAsync(OnMapReadyCallback)'. This function automatically initializes the map system and view. Then this map collects the respective monument's coordinates (latitude, longitude) in order to set up a marker that when clicked, allows the user to get directions to the respective monument. (Coding, 2018) Any objects obtained from the Google Map are associated with the view. It is essential to not hold on to objects (e.g. Marker) beyond the view's life. Otherwise, it will cause a memory leak as the view cannot be released. (Google, 2020)

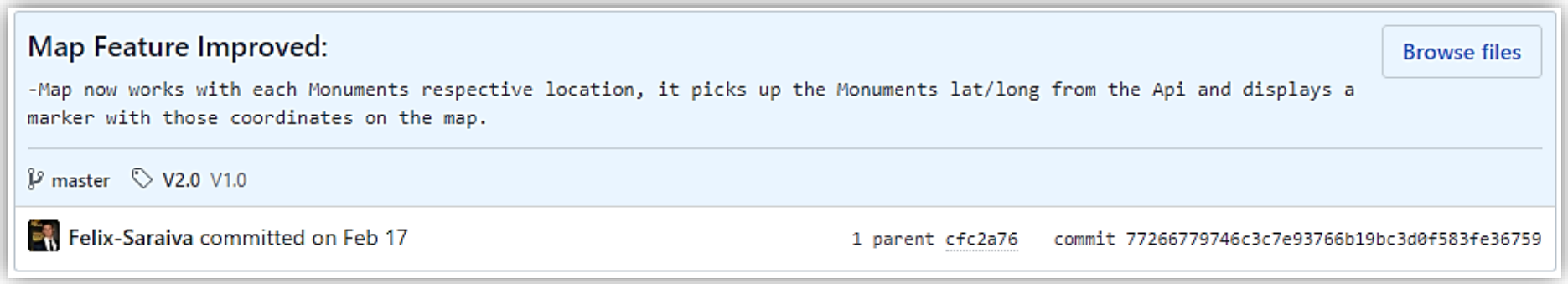
About the FR.14's development, that creates a compilation of functions, inside 'ViewPagerHeaderAdapter.java' class, that work together in order to collect, recycle and display the list of most famous monuments, present on the API, to the Home View. (CodingWithMitch, 2018)



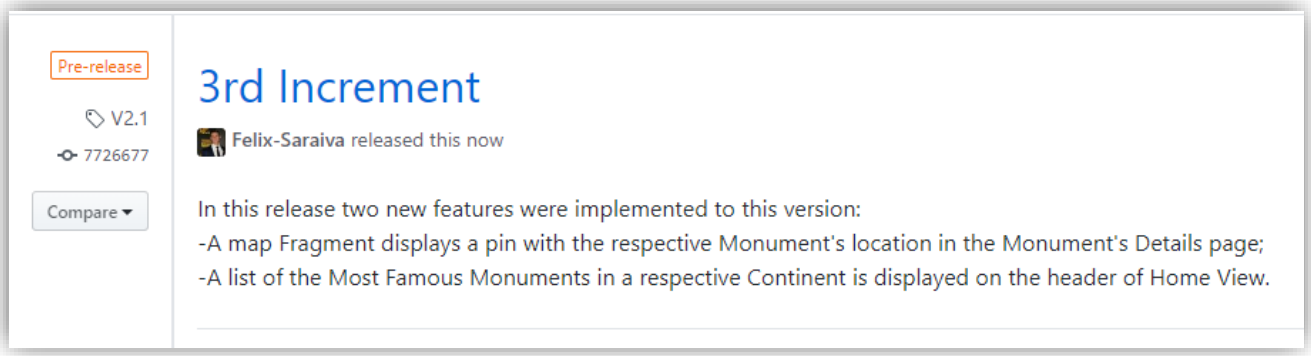
Implementation

3rd Increment GitHub Commits:

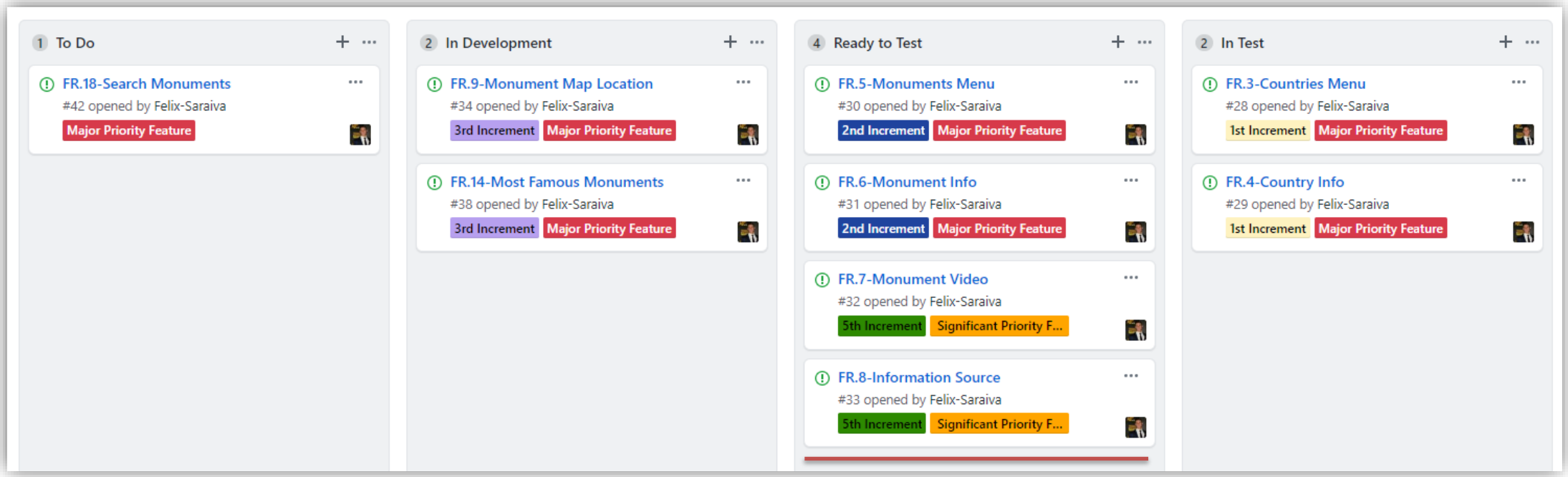




Version Control:



Scrumban Board Progress:



WIP Limit

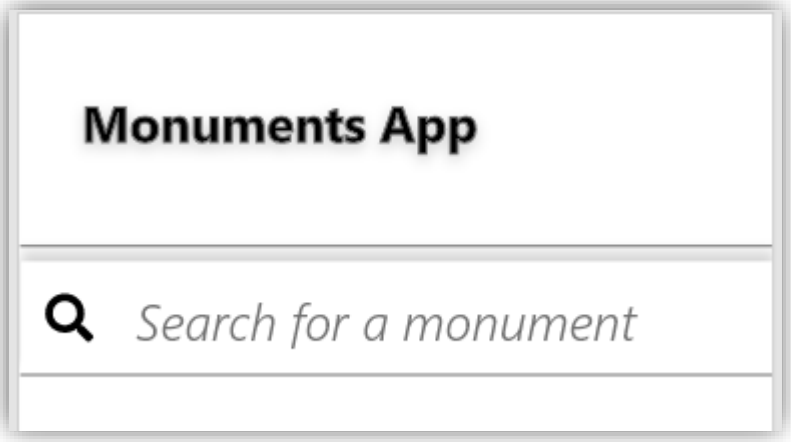
4rd Increment

In order to allow the user to search for a specific monument, in a respective continent, this increment has the objective of implementing a Search Monuments feature, FR.18.

Present Requirements	Status
FR.18-Search Monuments	In Development

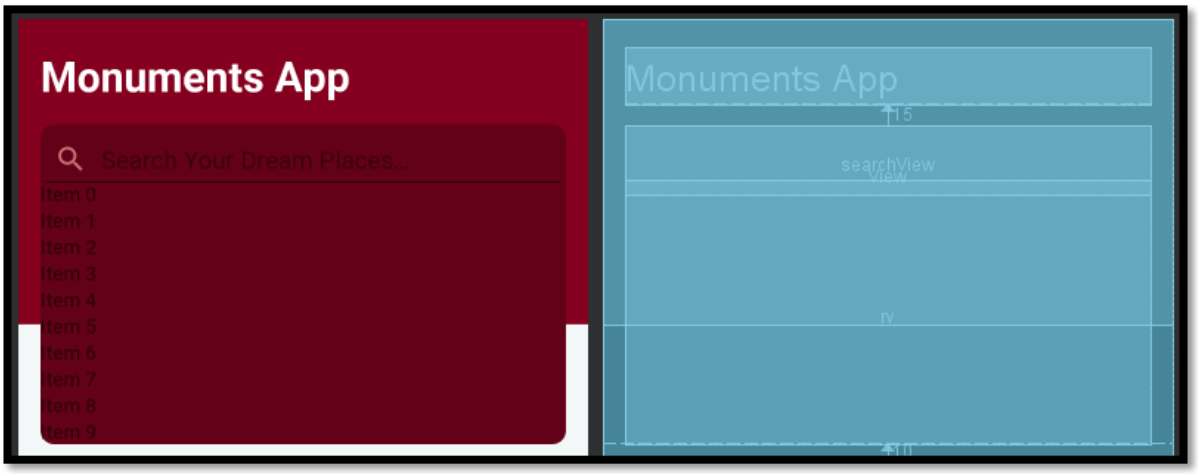
Design:

Based on the first Prototype, the 4nd Increment matches the following design:



Search Monuments Blueprint & Layout:

- In ‘activity_home.xml’, the search feature embedded in an EditView, allows the user to type the specific monument’s name, followed by a recycler view, attached to the edit view, that captures the monuments that match with the letters typed, from the ‘SearchAdapter.java’.



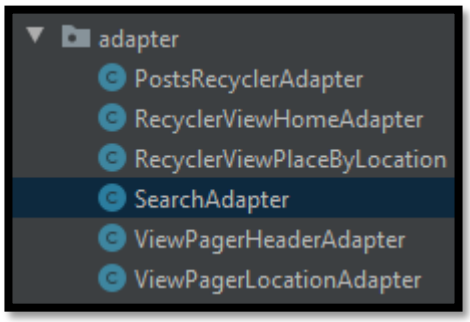
- The ‘search_list_items.xml’ is the layout used by the ‘SearchAdapter.java’ to recycle information, from the matched monument, and display it on each item represented above.

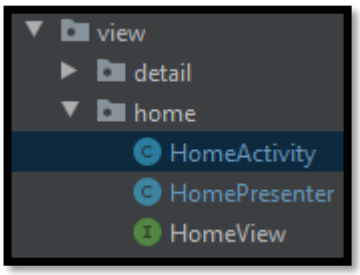
Development

Probably the most challenging Increment of this development, based on the construction of the 'SearchAdapter.java' class and search functions, that query throughout Firebase Real-time database, in search of the typed monument's name or country.

These functions, located in the ‘HomeActivity.java’ class, capture the typed letters, in real-time, sending them to the ‘SearchAdapter’, that compares them with the database ‘Places’ (Monuments), if there is a match, the adapter will send the monument back to ‘HomeActivity’ that displays it in the Views showed above. By creating these adapter classes, this system demonstrates high cohesion in its design, this way classes like the ‘SearchAdapter’ are targeting towards a more specific and specialized task, and such classes are not only easier to create but also easier to maintain and update, rather than when classes designed to do too many different tasks.

In page 225 of Appendix B, the 'setAdapter (final String searchString)' function represents most of this Increment’s logic.

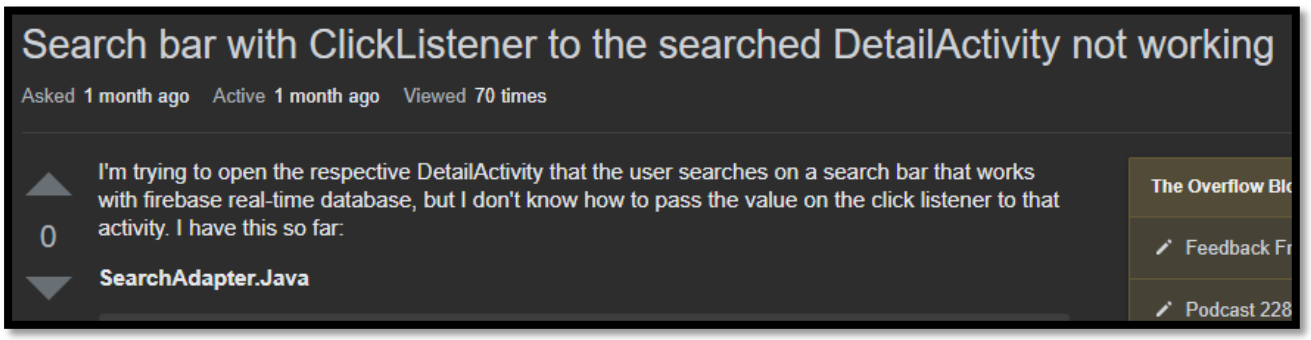




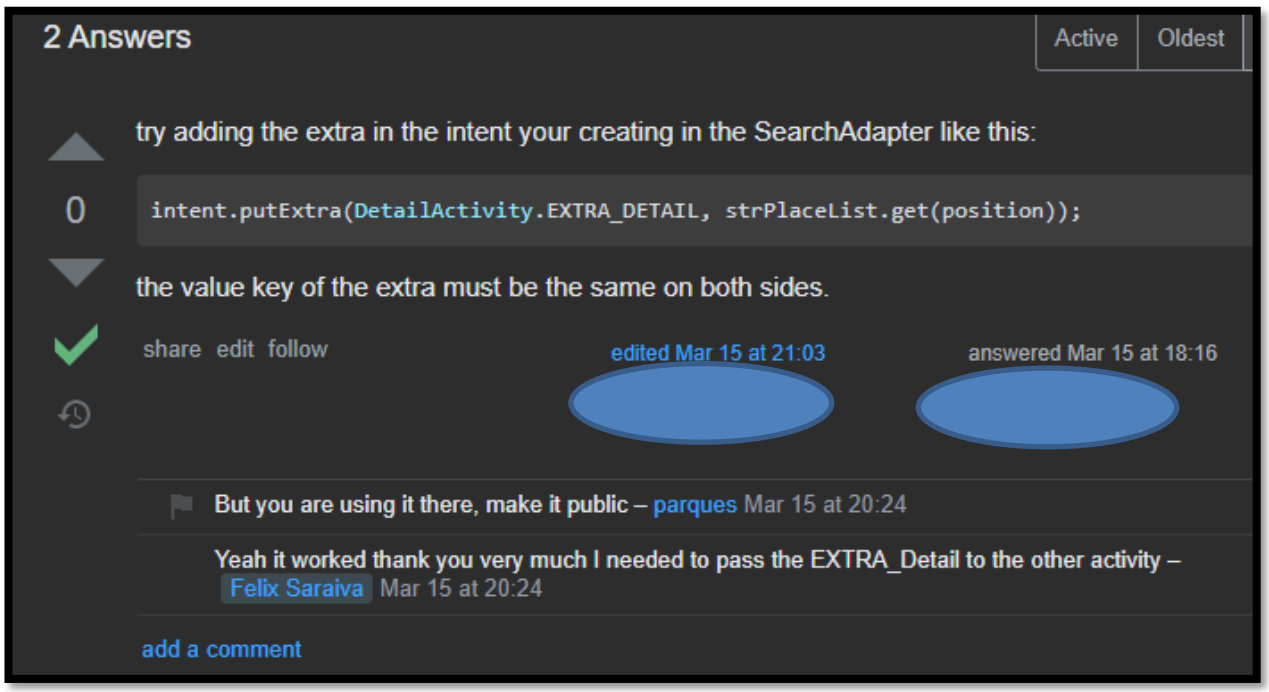
⇒ Throughout the development of this app, the developer encountered many challenges and used several tools of research to overcome them. One of the tools was the website 'Stack Overflow'. A website where coders can present their problems and receive help from more experienced developers, that have already encountered similar issues. This Increment presents an excellent example of the usage of such a tool:

Question:

Link: https://stackoverflow.com/questions/60683679/search-bar-with-clicklistener-to-the-searched-detailactivity-not-working/60697278#60697278

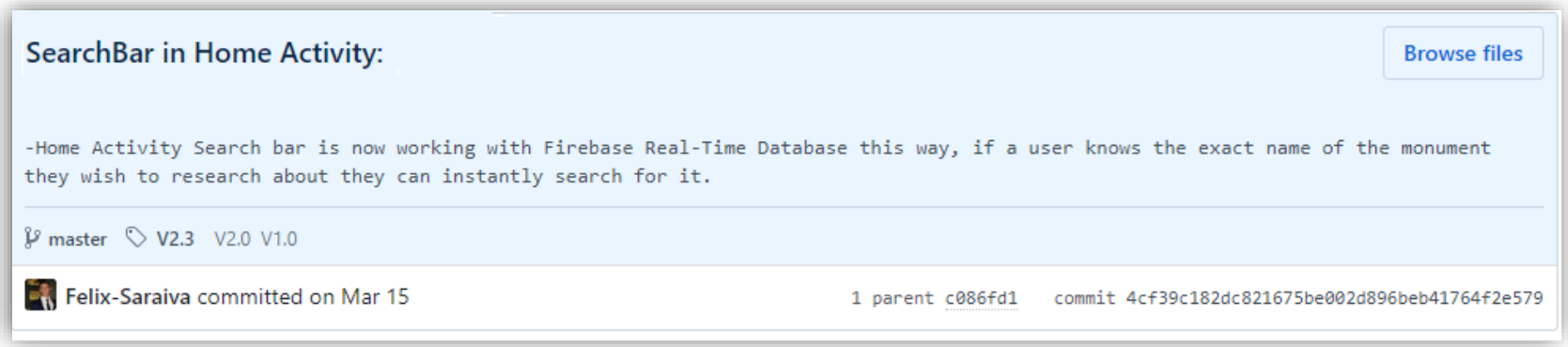


Answer:

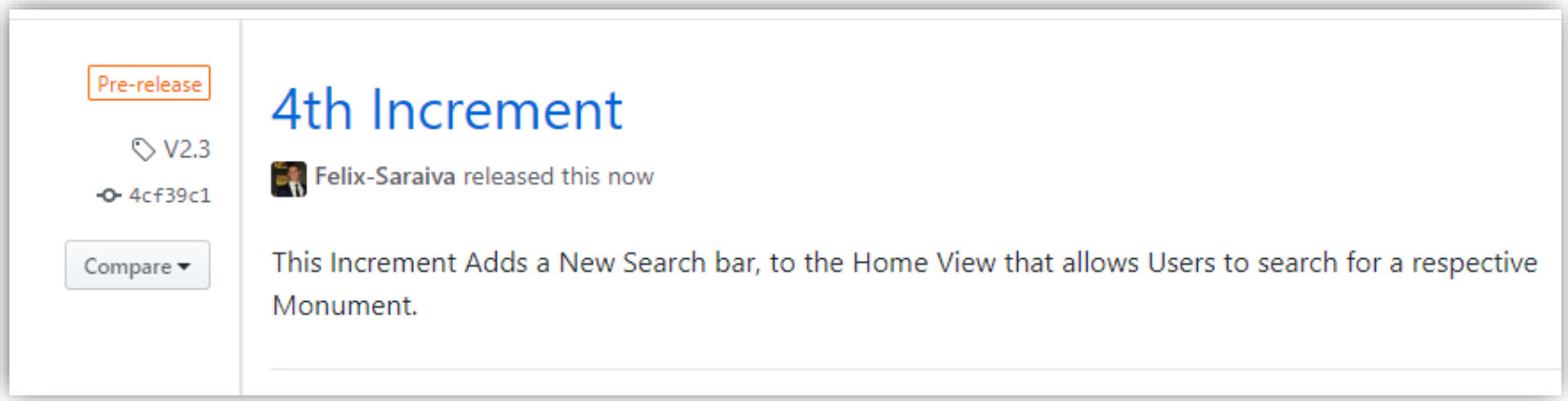


Implementation:

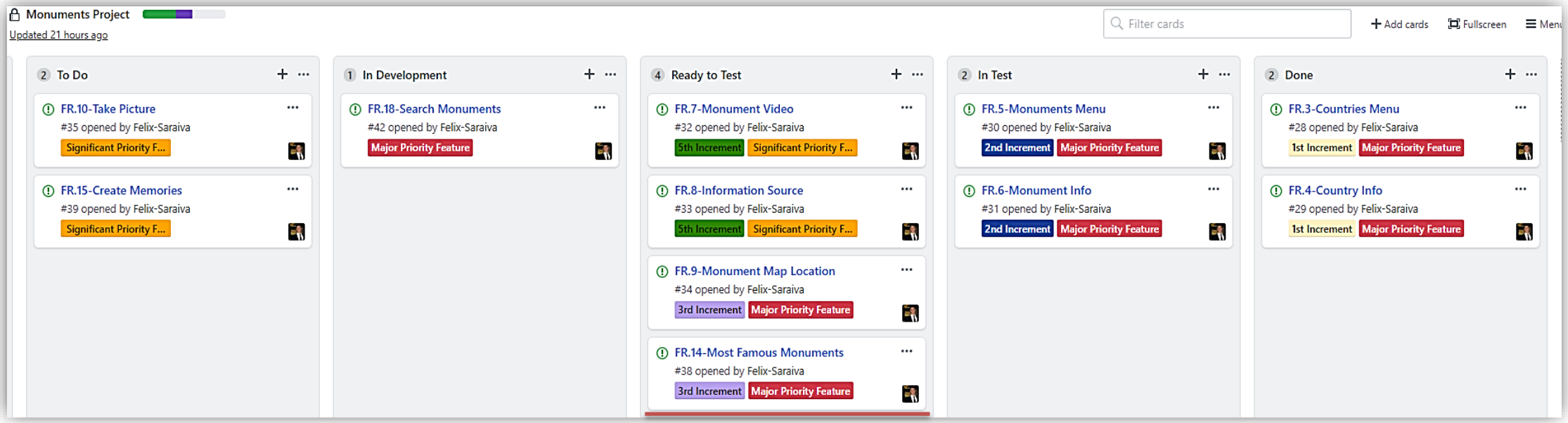
4rd Increment GitHub Commits:



Version Control:



Scrumban Board Progress:



6th Increment

This Increment focus itself on the construction of the requirements FR.10, Take a Picture and FR.15, Create Memories.

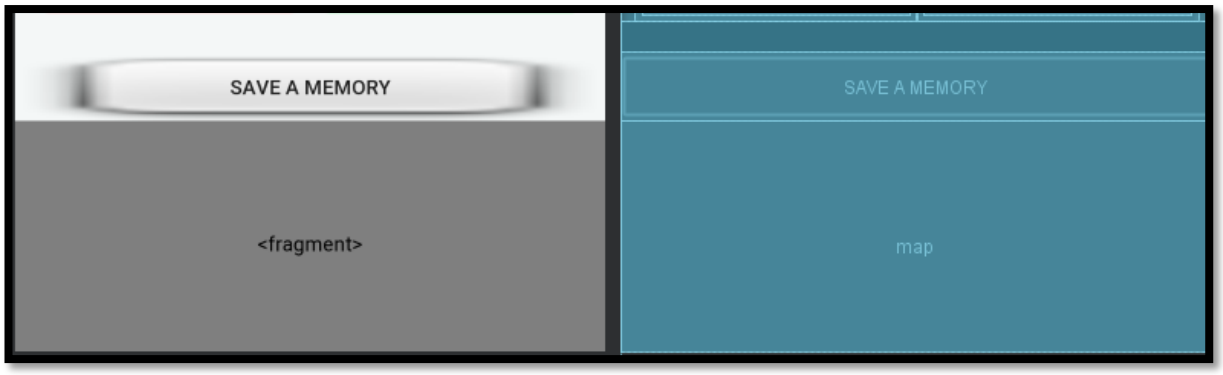
Present Requirements	Status
FR.9-Monument's Map Location	Developed
FR.10-Take a Picture	In Development
FR.15-Create Memories	In Development
FR.16-View Memories	7th Increment

Design

Based on the first Prototype, the 6th Increment matches the following design:

Take a Picture Button Blueprint & Layout:

- Within the same Linear Layout as the Map Fragment, in ‘activity_detail.xml’, the ‘Save a Memory’ feature is created using a regular button layout.

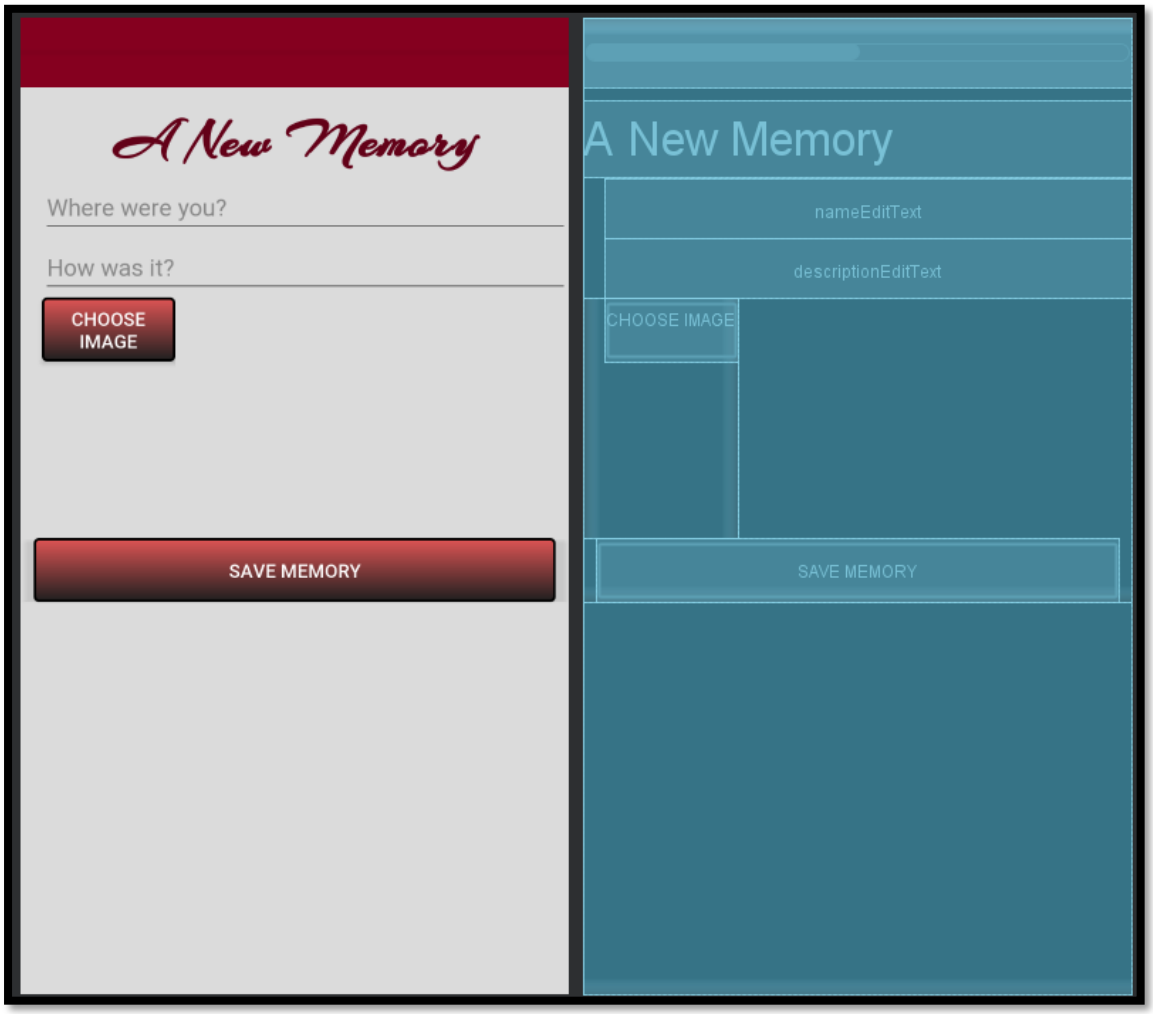


Create & View Memories Menu Blueprint and Layout:

- Inside a Relative Layout, the new ‘Posts’ menu, located in the ‘activity_posts.xml’ was created based on the creation of two buttons, one opens the ‘View Memories’ requirement, and the other opens the ‘Create Memories’ requirement, the rest of the layout is based on Image and Text Views that allow the user to understand the purpose of this page straightway.

Create Memories Blueprint and Layout:

- The ‘activity_post_up.xml’, created to be the layout of the requirement, ‘Create Memories’ (FR.15), is composed by two edit texts that allow the user to insert the name of the Monument they visited along with a description of how was the visit, plus two buttons, one that when clicked, opens the user’s device gallery/cloud storage and allows them to choose a picture, and the other uploads the created memory to user’s memories database.



Development

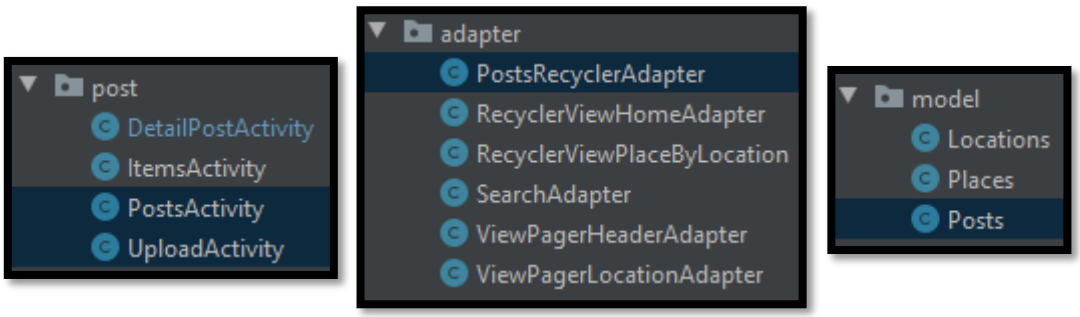
The first requirement of this Increment, 'Take a Picture', is the bridge between a monument's page and creating a memory, of the same.

Although the development is a simple 'onClick()' function that links the click of a button with the initiation of the device's camera (page 222-Appendix B), it is a smart solution to the connection of a monument and the creation of a memory.

The second requirement, 'Create a Memory', required the creation of a new menu, 'PostActivity.java' class in order to give more organization to the app, this menu gives two choices create a memory, or choose to view existing ones.

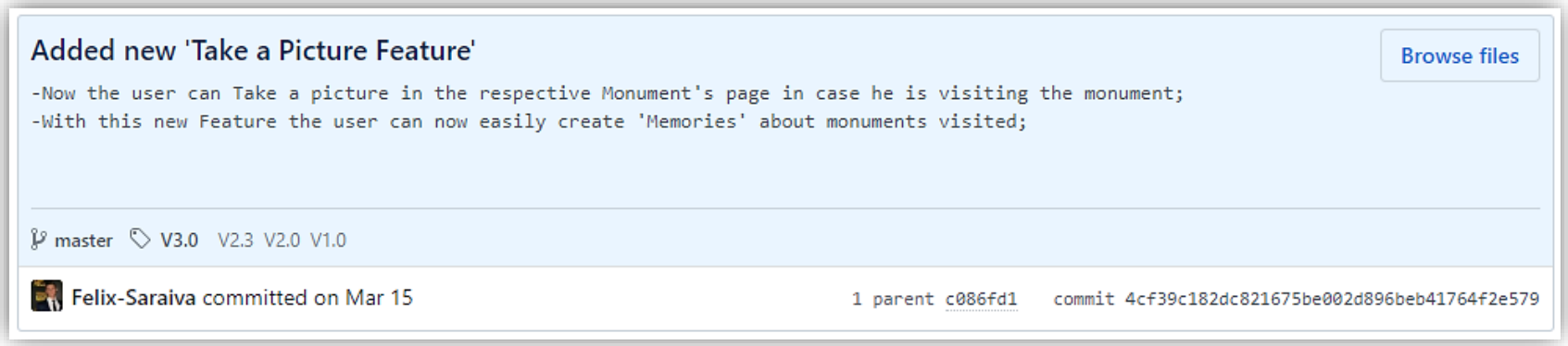
After the creation of the post's menu, follows the development of the actual functionality. This part of the Increment also follows the chosen design pattern, present in the 'Methodologies', therefore this Increment also has a Model, 'Posts', an adapter 'PostsRecyclerAdapter', and two activities in its view, 'PostsActivity' and 'UploadActivity'.

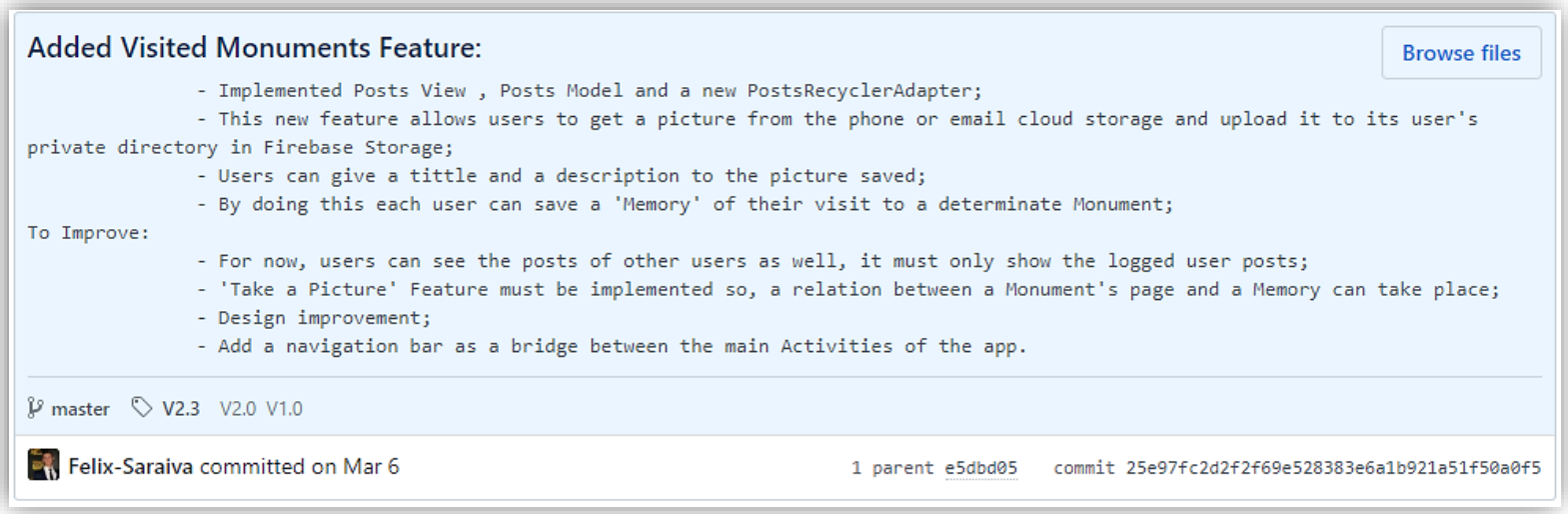
After an image is selected and a title plus a description are given to a new memory, the function' uploadFile()' (page 241) is responsible for uploading the data, to the respective user's database, and providing some error handling messages, in case of an unsuccessful update. (Flow, 2017) (ProgrammingWizards, 2018)



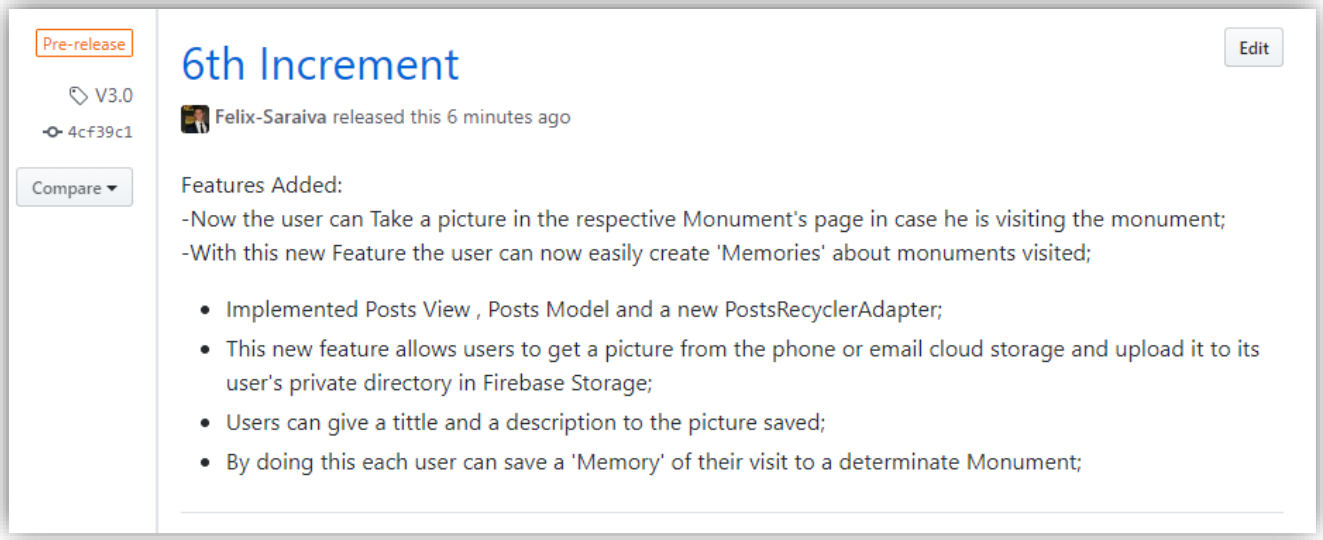
Implementation

6th Increment GitHub Commits:

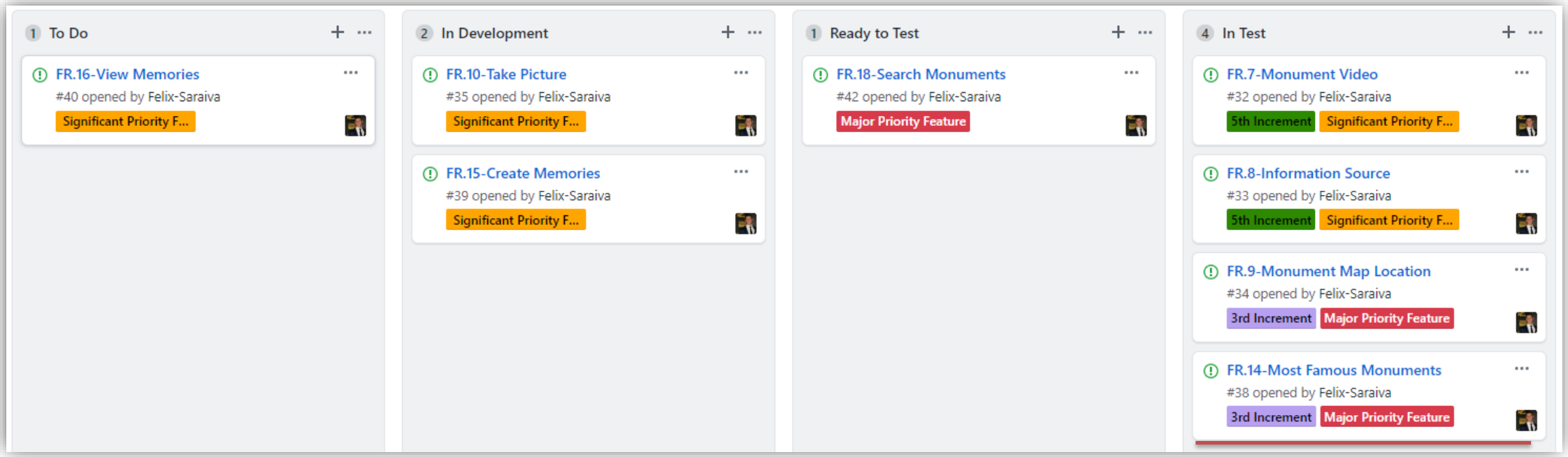




Version Control:



Scrumban Board Progress:



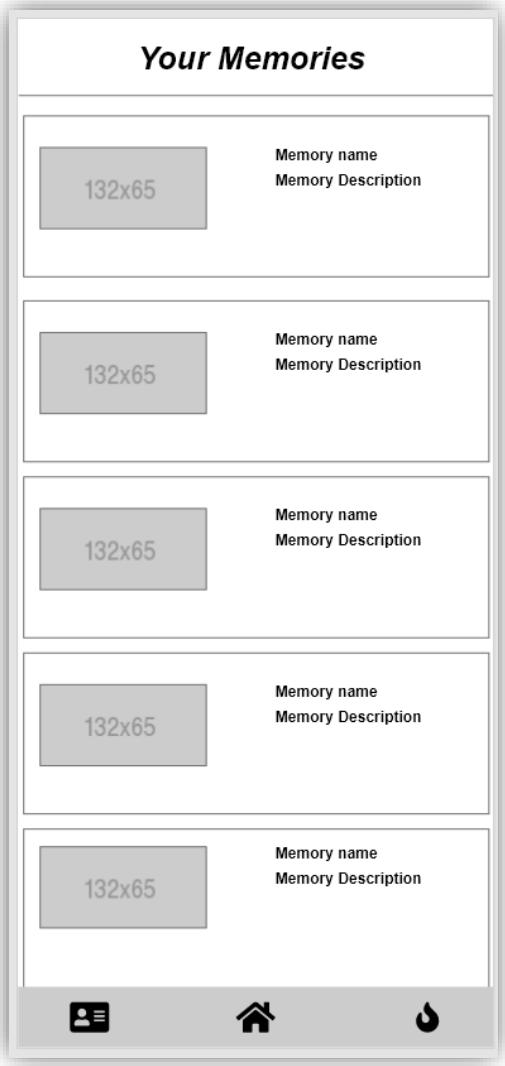
7th Increment

An extension to the previous Increment, the 7th Increment aims to complete the memories section of the application and provide the option to review created posts, with the implementation of the FR.16 requirement, 'View Memories'.

Present Requirements	Status
FR.15-Create Memories	Developed
FR.16-View Memories	In Development

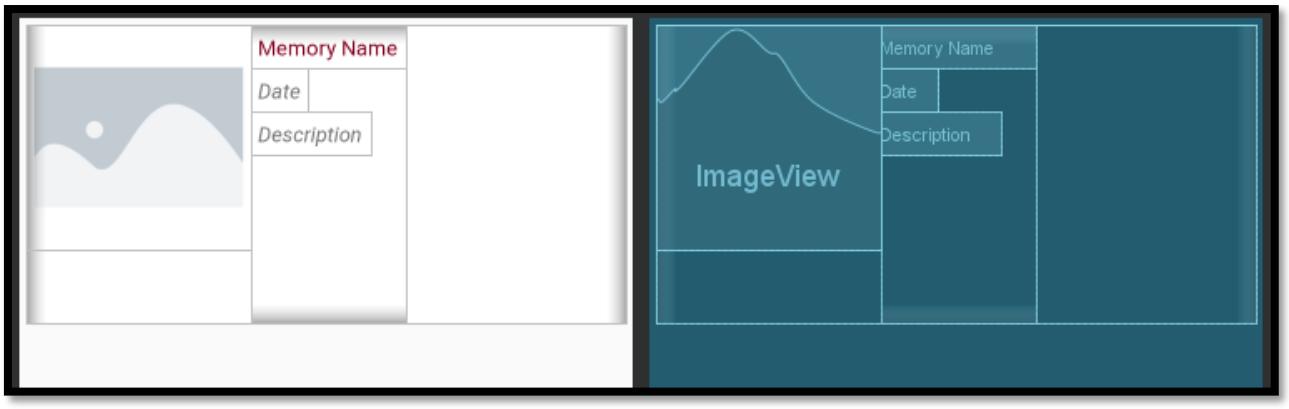
Design

Based on the first Prototype, the 7th Increment matches the following design:



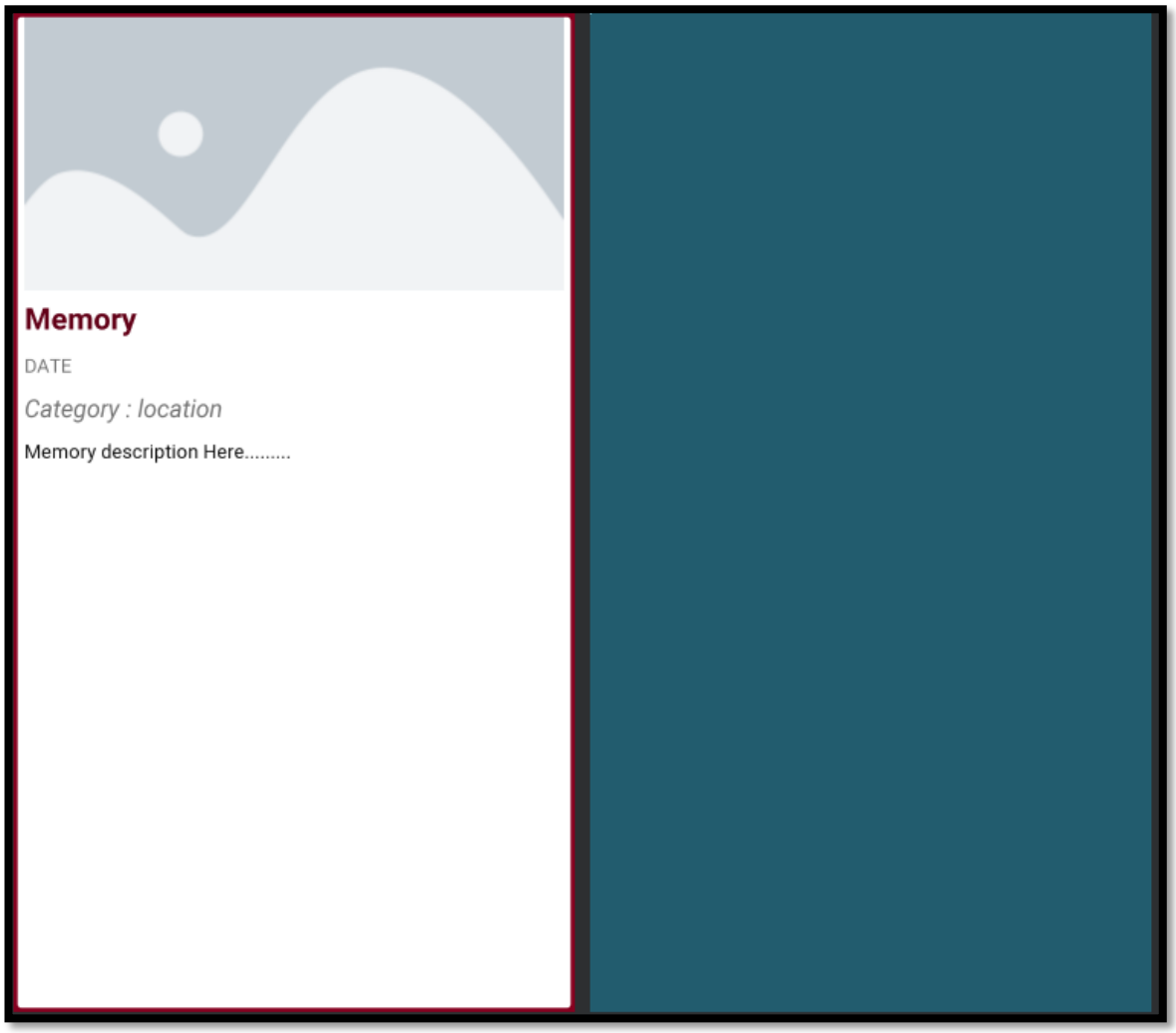
View Memories Blueprint & Layout:

- Like the Search list used on the 4th Increment, ‘activity_posts_items.xml’ uses a recycler view list that fills itself with memories, just like displayed bellow, on the ‘row_model.xml’ layout.



Memory Details Blueprint & Layout:

- The click on a memory provides the user with several options, for example, a simple click will allow the user to enlarge that memory into a new layout, 'activity_posts_details.xml', that provides a clearer vision of the memory's image and description.

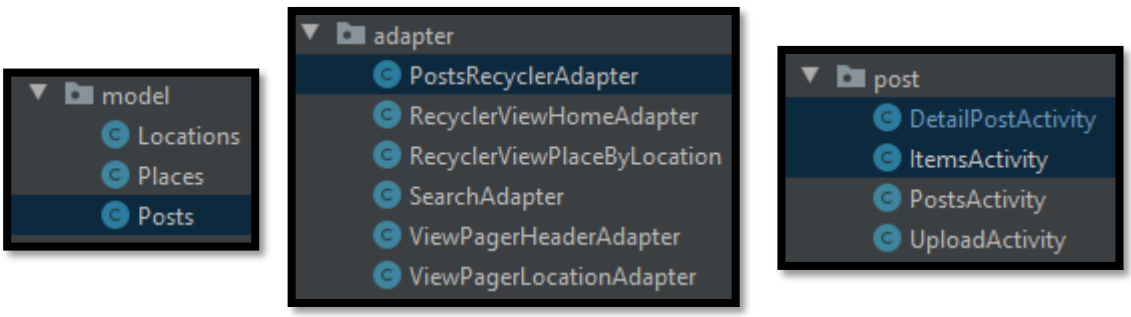


Development

Much related to the back-end, this increment provides the user with a list of their posts of monument visits.

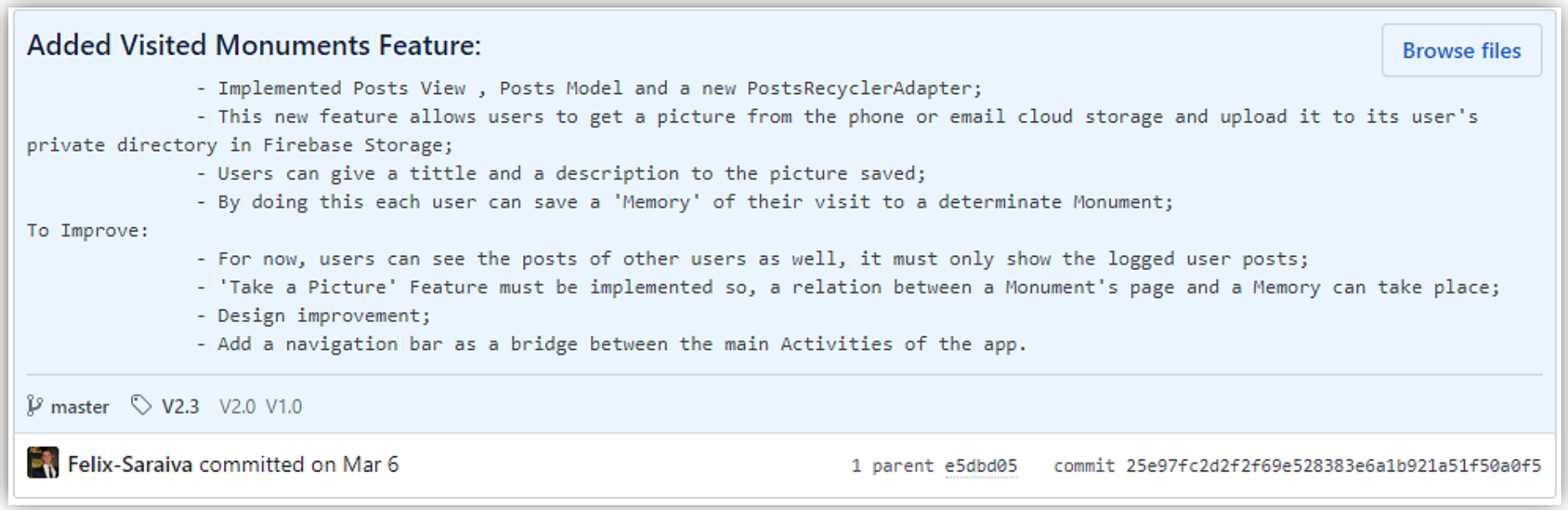
The developed Model, 'Posts', gets the information from the user's database and sends it to the 'ItemsActivity.java class', this class provides the user with the options of deleting a memory or opening the details View of a memory. (see page 236 with the respective coding functions) (Flow, 2017)

The DetailPostActivity.java class receives all the information of the clicked memory from 'ItemsActivity' class and displays it in a larger view, with the respective date and a random traveller's enthusiastic phrase.(Functions on page 235) (ProgrammingWizards, 2018)

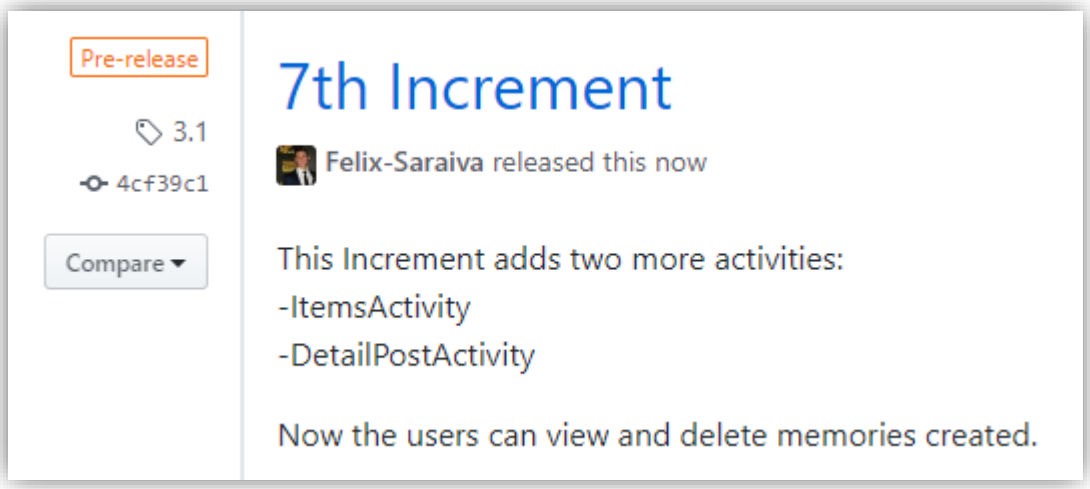


Implementation

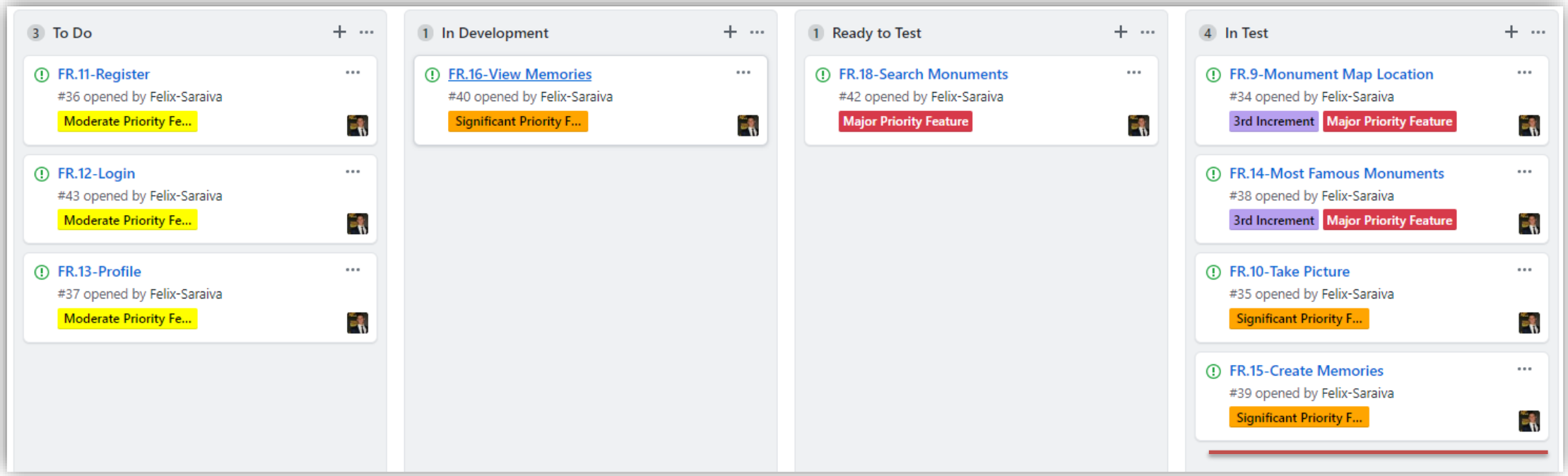
7th Increment GitHub Commits:



Version Control:



Scrumban Board Progress:



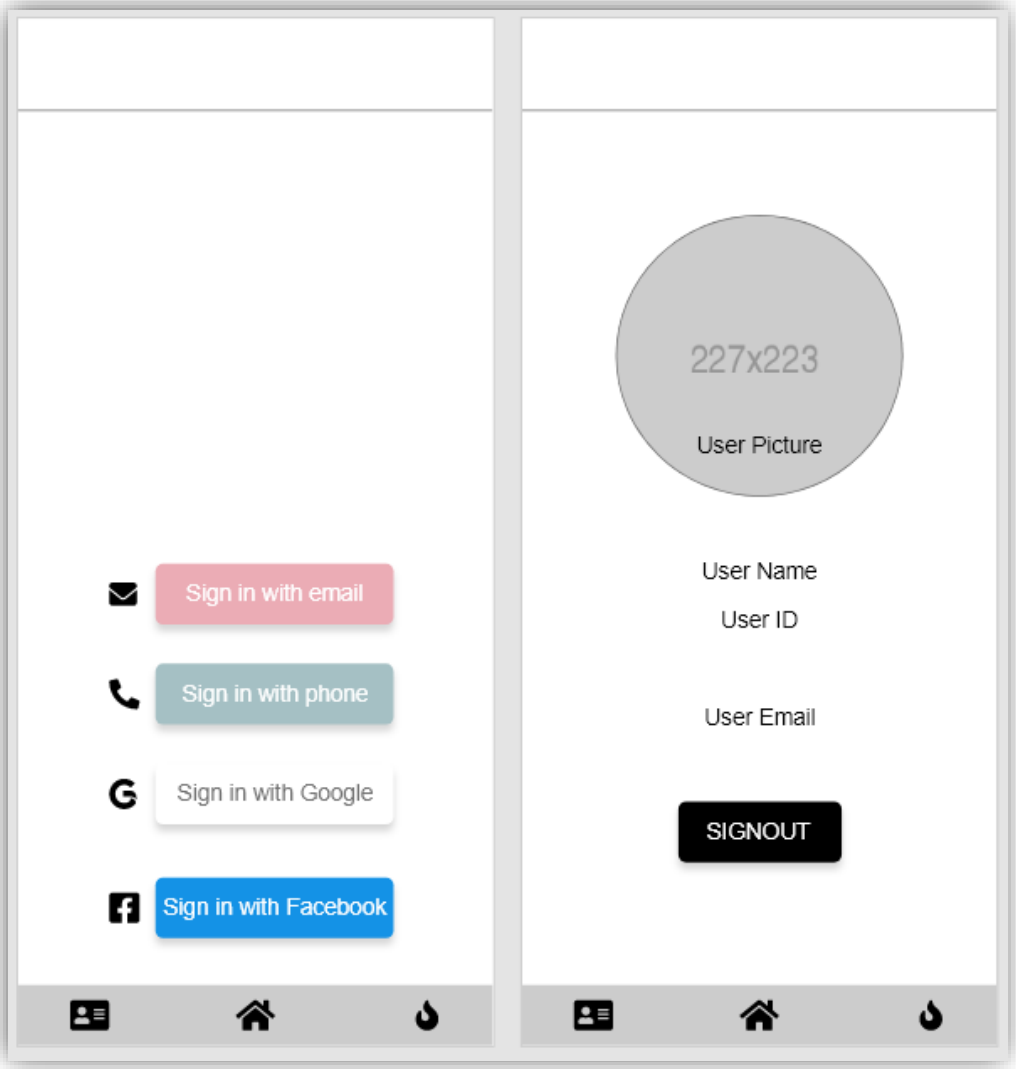
8th Increment

This increment aims to implement user security and privacy, as the present requirement's objectives are to create a user Register/Login and User Profile.

Present Requirements	Status
FR.11-Register	In Development
FR.12-Login	In Development
FR.13-Profile	In Development

Design

Based on the first Prototype, the 8th Increment matches the following design:

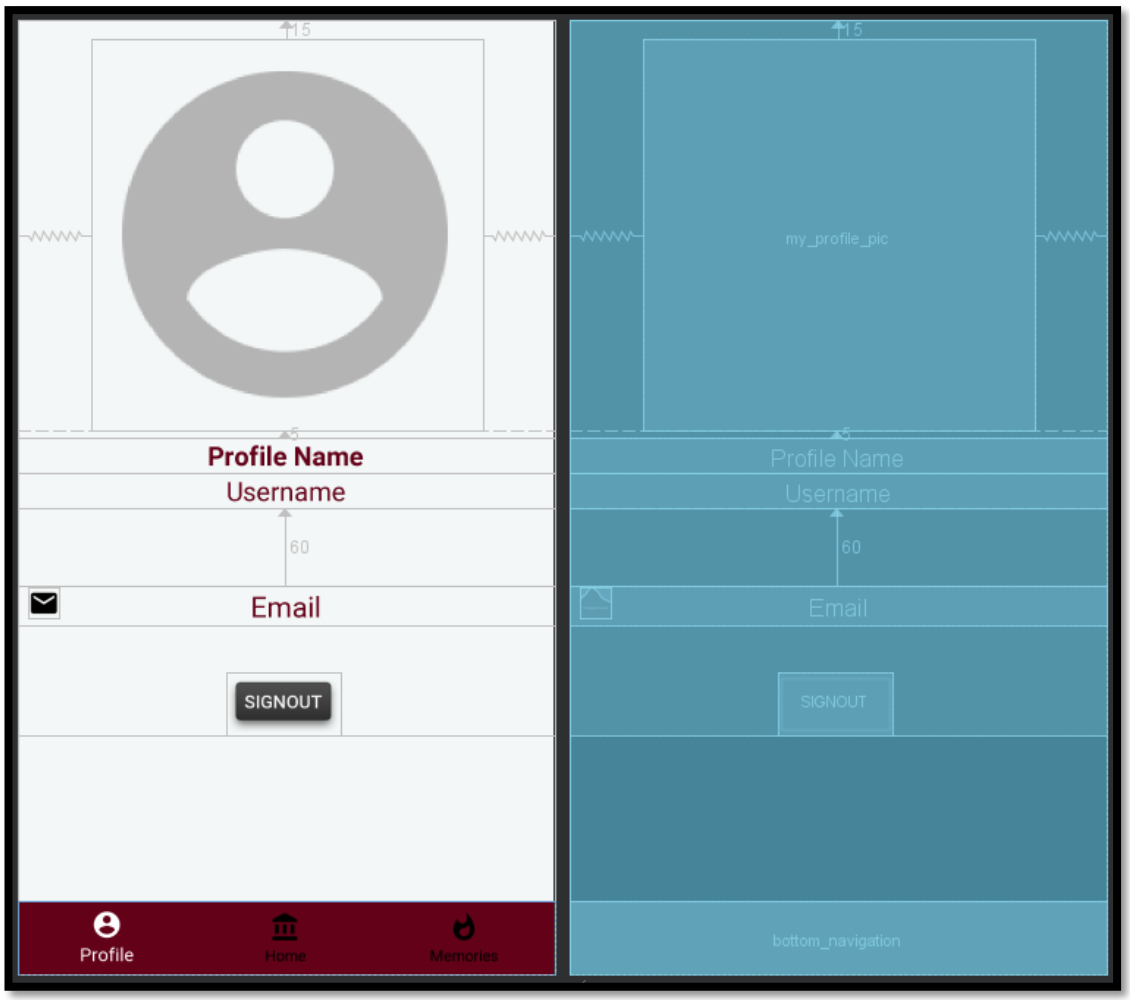


Login/Register Layout:

- This user login layout was imbedded by Firebase Authentication and sets up the four possible Register/ Login options to this application. (Google, 2020)

User Profile Blueprint & Layout:

- The layout ‘activity_profile.xml’ composed by some user information is the first prototype of the user profile feature. Inside a Relative Layout it has several text views, a circle image view and a sign out button that allows the user to log out.

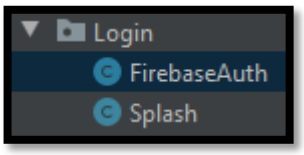


Development

The use of Firebase Authentication, on this Increment, is a vital tool to achieve security and reliability.

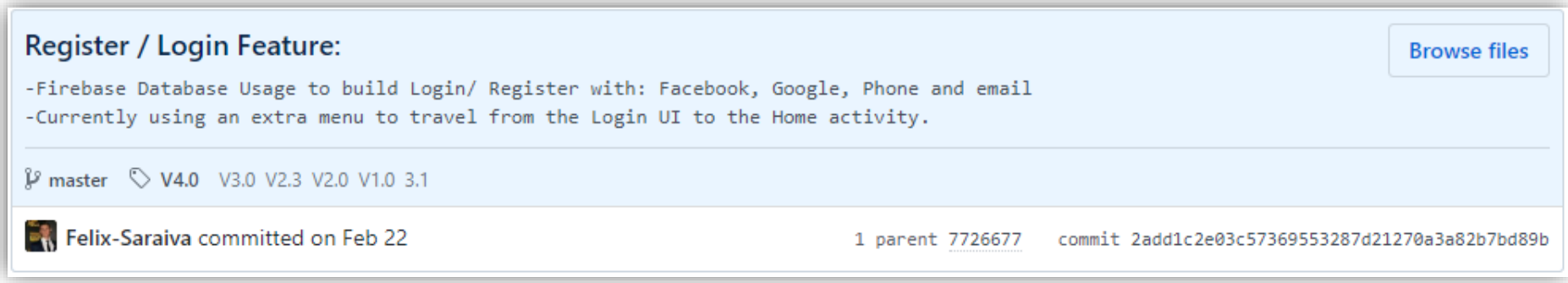
Thanks to this tool it was not required to create a complete ‘create an account’ feature, instead, inside ‘FirebaseAuth.java’ class, the system provides the user with four login options that are independent of this application, this means that this app, provides more security to the user since the developers do not possess any user personal information. Instead, the chosen option, for example, Gmail login, will provide the system with a user UID code, that this app uses to identify the user, store his data and process any other feature that requires the identification of the current user. (Google, 2020)

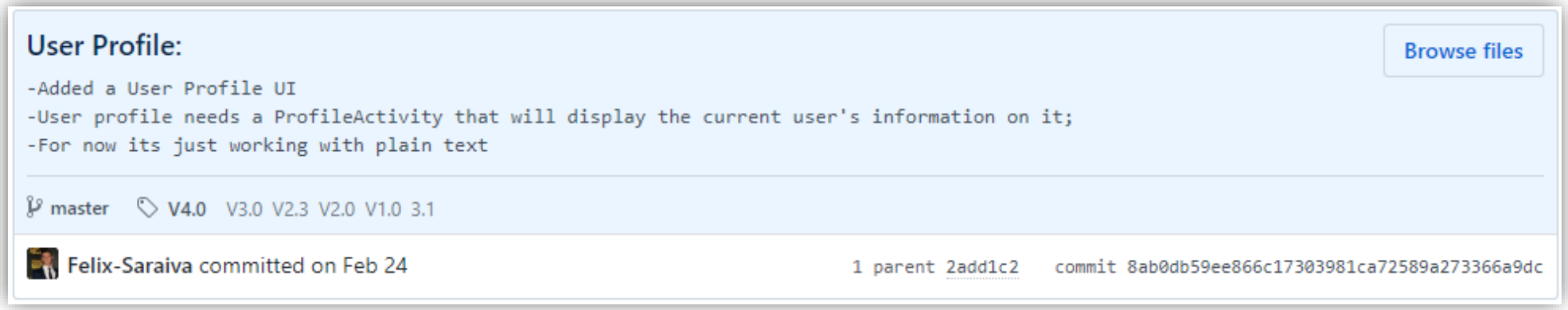
The ‘displayUser()’ function in Appendix B, page 214, is an excellent example of how the system can identify and protect the user.



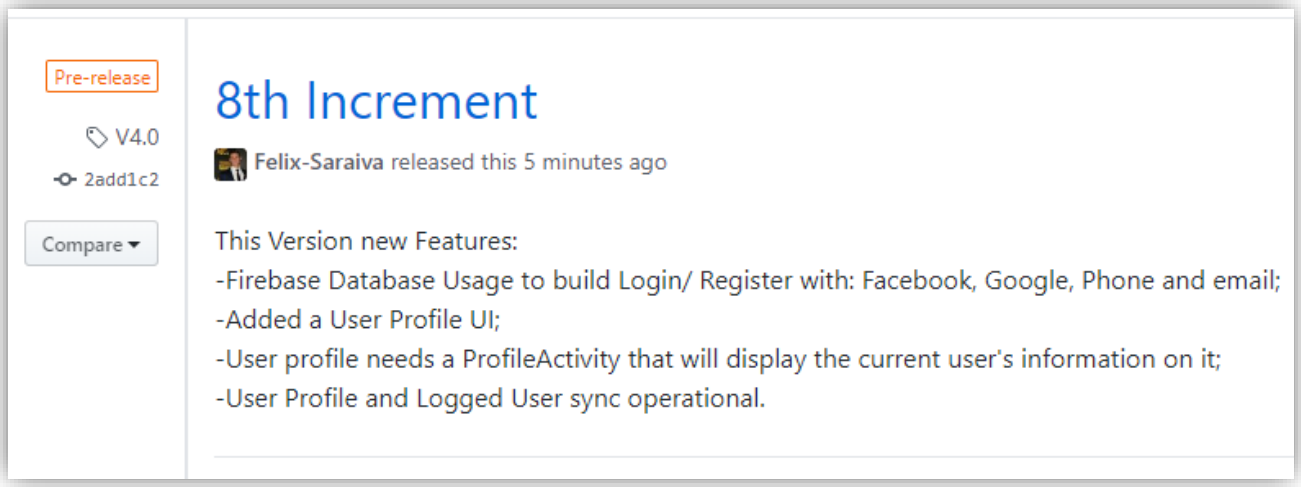
Implementation

8th Increment GitHub Commits:

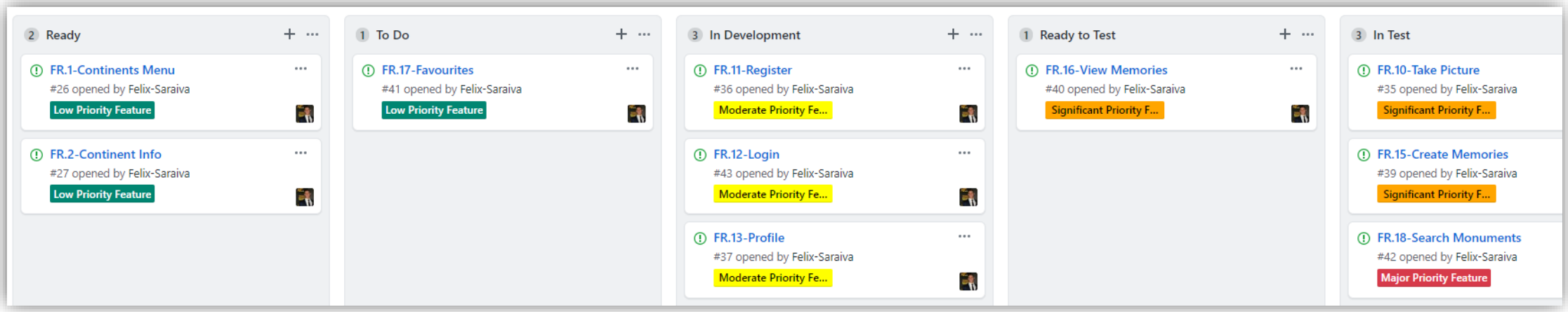




Version Control:



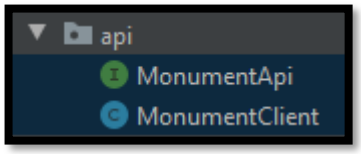
Scrumban Board Progress:



2. Back-End Development

After the specification of the front-end development, it is essential to break down how this application gets the required data.

The back-end development of this application was not built after the front-end, but side by side with it, this way the developer could test the front-end alongside with the back-end, every time a new increment got implemented. This back-end development stage is divided into two phases, the Experimental phase and the Operational phase.



2.1. Experimental Phase

Like clarified in the 'Literature Survey', mobile front-end gets its data from the back-end via a variety of service calls, named APIs.

The connection to the primary data, present in this application (monument's related information), was built for the system to be independent of the Database in use. In other words, in a team development environment, front-end and back-end developers usually work in different teams so, both 'ends' should be independent of the tool either party chooses to use, in this case, the application's back-end is fully located in 'Firebase', but it was not always like that.

The first experimental Database of this system was built in 'Postman'. Postman is a software that creates mock-up servers and APIs, without any security rules or code required, it is a tool used by many front-end developers, to test their systems, against mock-up APIs, when they are still waiting for the real Database to be built by the back-end team.

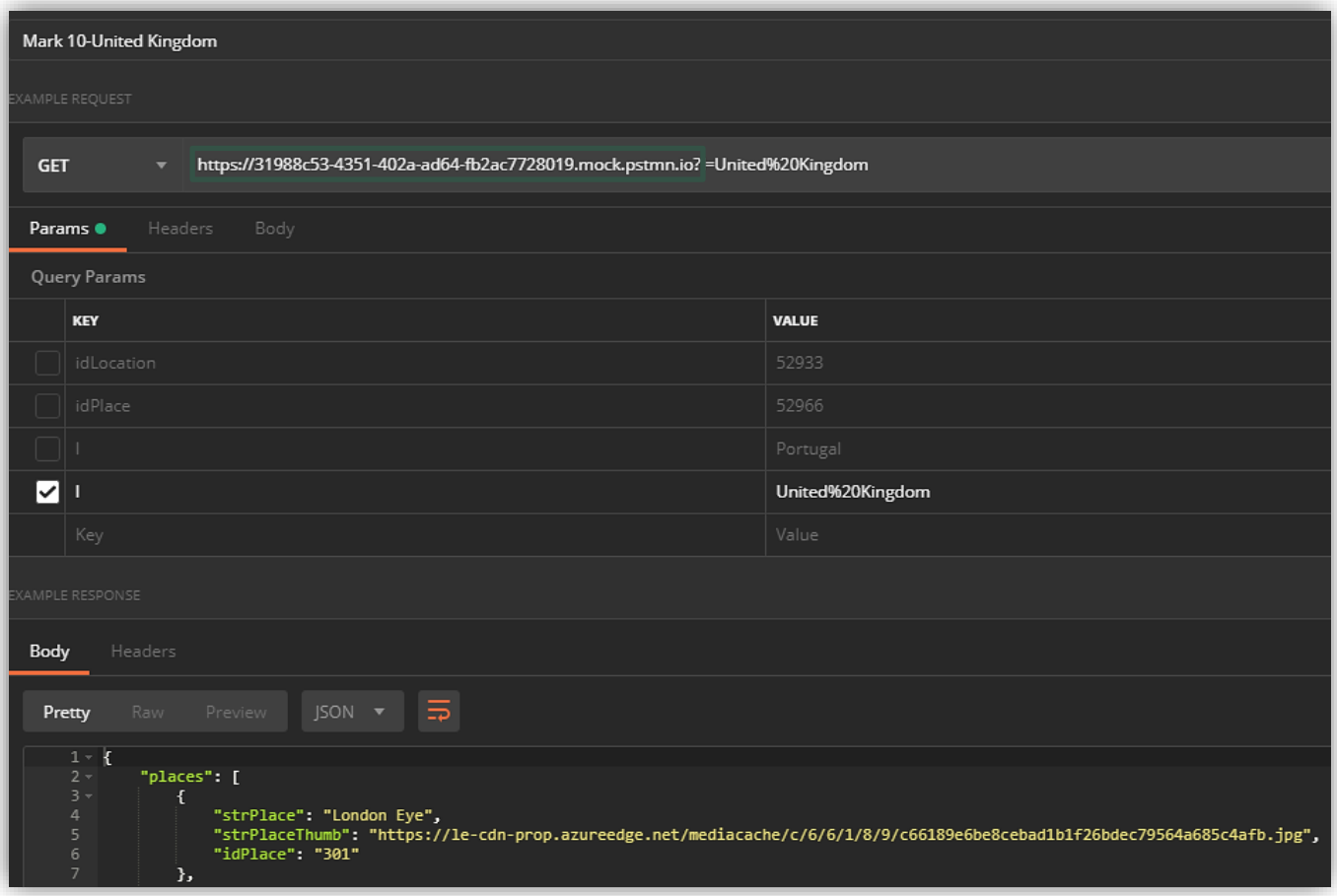
⇒ The independence of this system starts with the creation of a ‘MonumentClient’ Class that takes the Base URL Http link of an API and connects any database calls to that Base URL link. In the images below the system is already using ‘Firebase’.

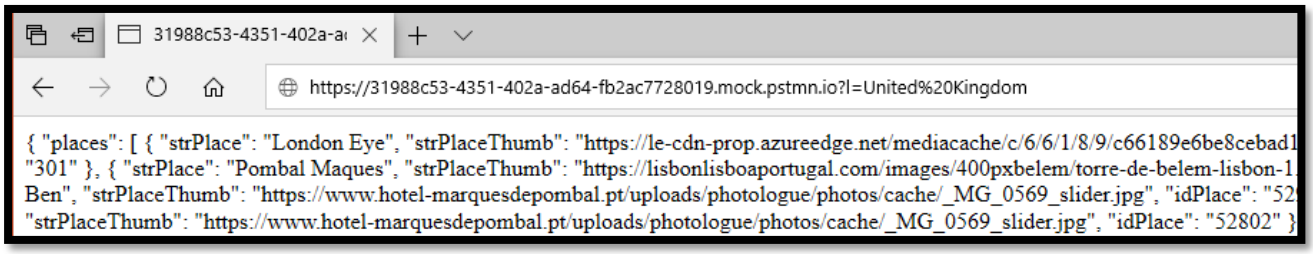
```
public class MonumentClient {

    //Link to Firebase Real-Time Database
    private static final String BASE_URL = "https://project-4a153.firebaseio.com";

    public static Retrofit getMonumentClient() {
        return new Retrofit.Builder().baseUrl(BASE_URL).client(provideOkHttp()).addConverterFactory(GsonConverterFactory.create()).build();
    }

    private static Interceptor provideLoggingInterceptor() {
        return new HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY);
    }
```

⇒ Then, the development of an abstract interface class, ‘MonumentApi’, makes the API calls and sends the data to the respective Models.

```
public interface MonumentApi {

    //GET Monuments Request
    @GET("/%3fPlace.json?print=pretty")
    Call<Places> getPlace();
    //GET Countries List Request
    @GET("/%3fLocation.json?print=pretty")
    Call<Locations> getLocations();
    //Get a Country Request
    @GET("/Countries/{l}")
    Call<Places> getPlaceByLocation(@Path("l")String location);
    //Get a Monument Request
    @GET("/Monuments/{m}")
    Call<Places> getPlacesByName(@Path("m") String placeName);

}
```

⇒ In the experimental phase of the development, the Http link on the base URL was not the one presented before, but the one on the image below.
This image represents the first 'Postman' mock-up API for the United Kingdom, using a JSON format, the data is displayed online to the respective link directory responsible for the United Kingdom data, just as shown on the second image.
This method helped the flow of the project and mostly, helped to test the system before implementing the final operational database.



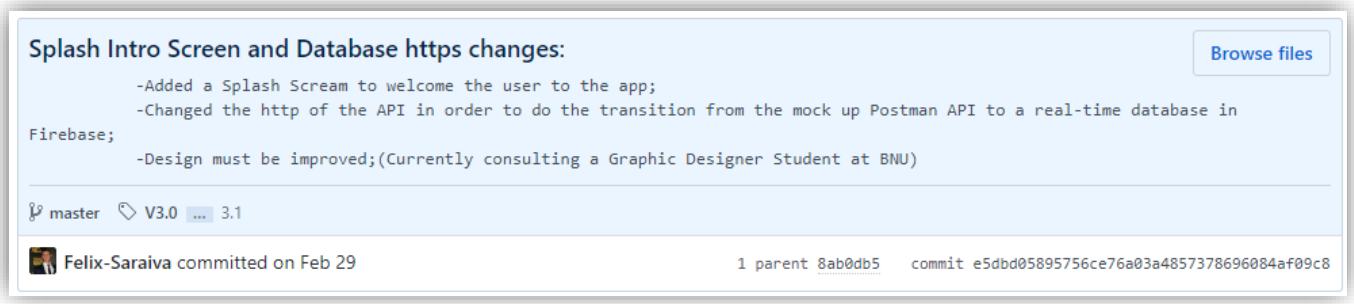


2.2. Operational Phase

Following all the experiments and tests done in the previous phase, it was time to find a secure and reliable back-end to implement in this project and so, with the same logic explained before, started the transaction from ‘Postman’ to ‘Firebase’.

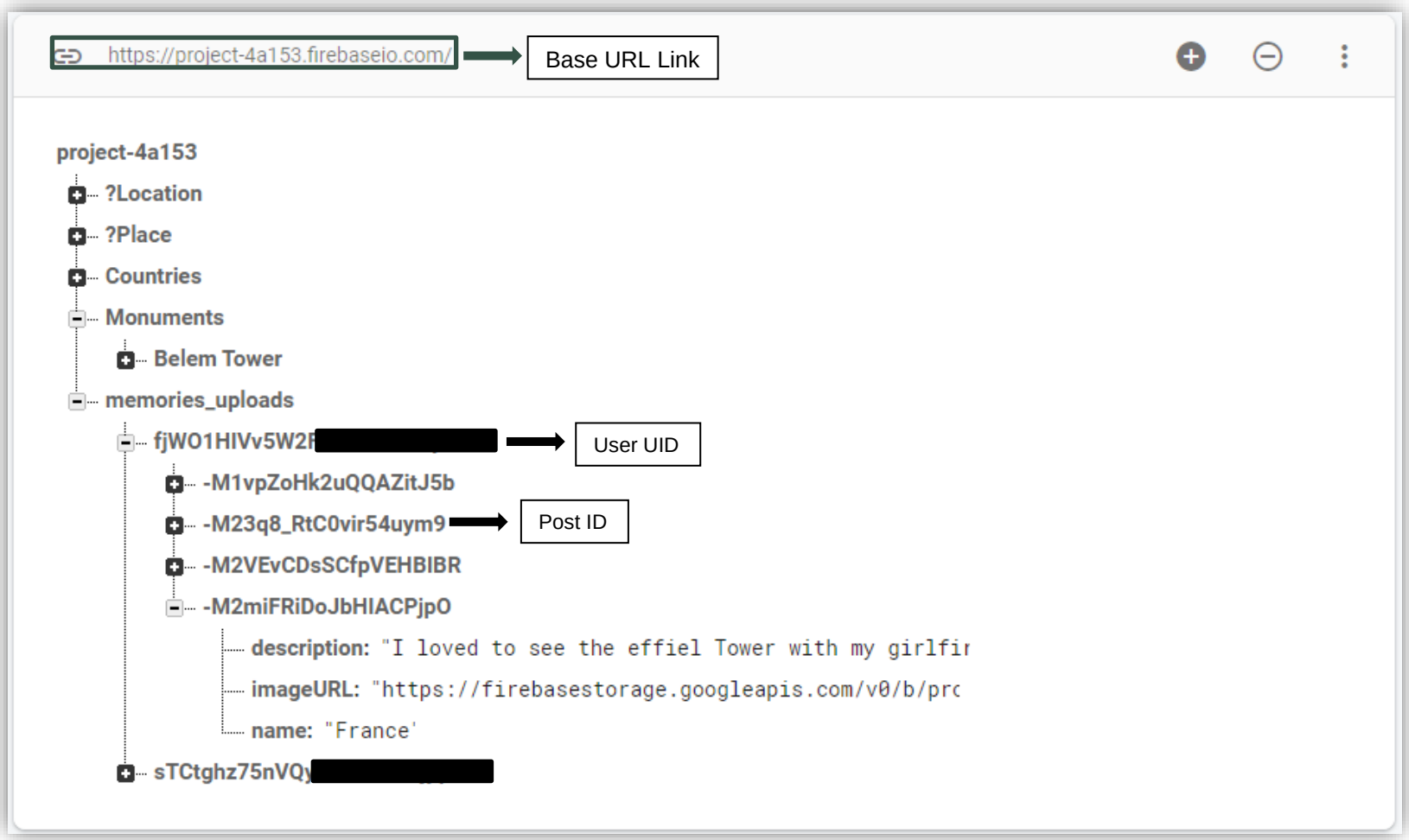
With all the JSON schemas ready and the structure of the database designed, it was possible to implement that knowledge into Firebase.

Transition Commit:



Firebase Real-Time Database:

⇒ The replacement for ‘Postman’ was found in Firebase Real-Time Database, a cloud-hosted database where data is stored as JSON and synchronized in real-time to every connected client, all the users share one Realtime Database instance and automatically receive updates with the newest data. (Google, 2020)



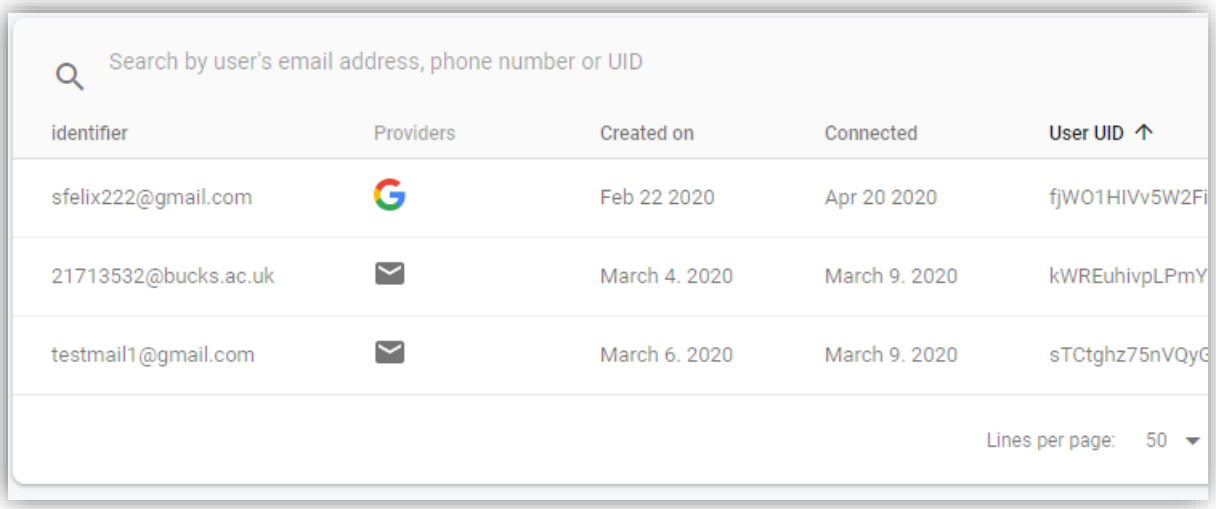
Real-Time Database Security Rules:

⇒ One of the best advantages of using a cloud-based database is the security level that it provides. Firebase Realtime Database Rules determine who has read and write access to the database, how the data is structured, and what indexes exist. These rules, located on the Firebase servers, are enforced automatically at all times. Every read and write request will only be completed if the rules allow it. (Google, 2020) The following rules show that the user can always read the monuments related information, but never write on it, with a different rule set for the memories section, where only a logged user can have access to read and write his memories.

Firebase Authentication System:

⇒ This application, just like many others, needs to know the identity of a user, this allows it to securely save user data in the cloud and provide the same personalized experience across all the user's devices.
This system uses Firebase Authentication tools that provide backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users.
The login system is the face of this process, using passwords, phone numbers and popular federated identity providers like Google and Facebook. (Google, 2020)

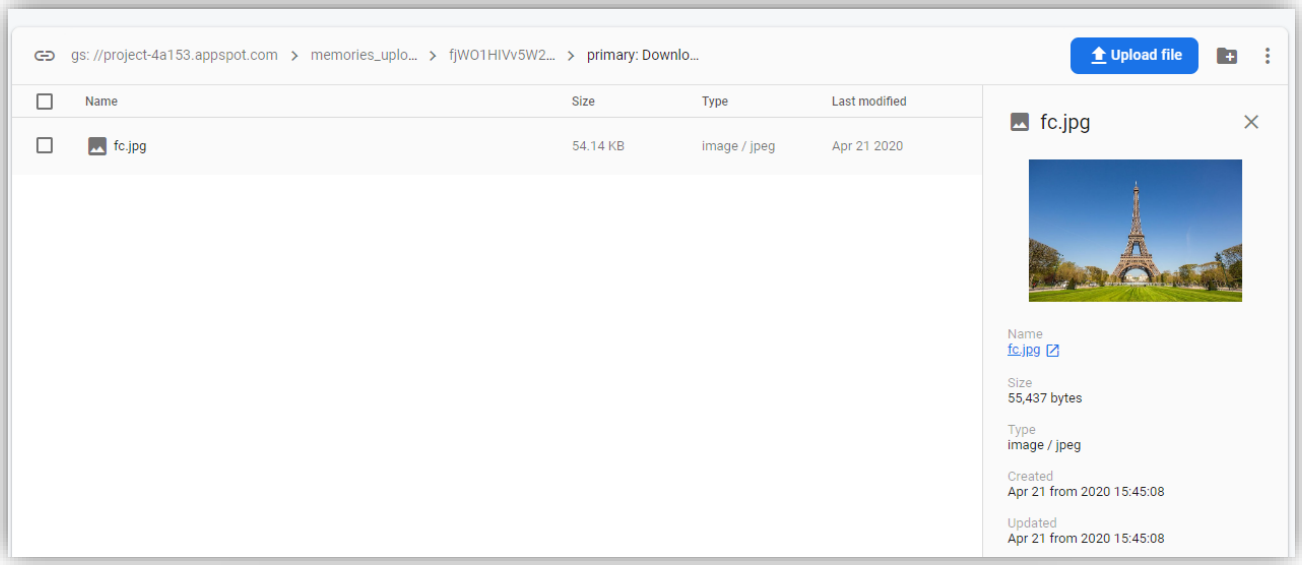
User Authentication Management in Firebase:



Firebase Cloud Storage System:

⇒ At this stage, the back-end is ready to support every monument related information, but the system also needs online storage that stores the user's memories. Since the description and title of each memory can be stored in the Firebase Real-Time Database, a solution to store the images of each memory is required.
Cloud Storage for Firebase adds security to each file uploaded, regardless of the network quality. Subsequently, this app uses firebase's SDKs to store images. The same files can be accessed via server-side.
So, each user has its own special memories storage, each memory has its memory ID, and it is composed by a description, a title and an image URL that contains the image's location on Firebase Cloud Storage.

Cloud Storage Memory Upload:



Firebase Cloud Storage Security Rules:

⇒ Similar to how Firebase Authentication makes it easy to authenticate the users, Firebase Security Rules for Cloud Storage makes it easy to authorize users and validate requests. Storage Security Rules manage the complexity by allowing specific path-based permissions. The following 'code' specifies that each user has a private 'path', which means only a logged user can write or read memories, and he is restricted to do so in his private storage.

```
service  firebase . storage  {
  match  / b / { bucket } / o  {
      match  / memories_uploads / { userId } / { allPaths = ** } {
        allow  read ,  write :  if  request . auth . uid  ==  userId ;
      }
  }
}
```

Project Testing

To make sure that the final product has a high level of quality, the testing process of this system will follow a strict testing process, supported by testing policies clarified in the 'Literature Survey' and 'Methodology.'

1. Monuments Application Incremental Testing

1.1. Black Box Testing

This stage takes the techniques present in the 'Methodology' section and tests them, against the final product, starting by performing Black Box testing against the system requirements.

1st Increment Testing

ID	Requirement
FR.3	Countries Menu
FR.4	Country Info

Test Cases:

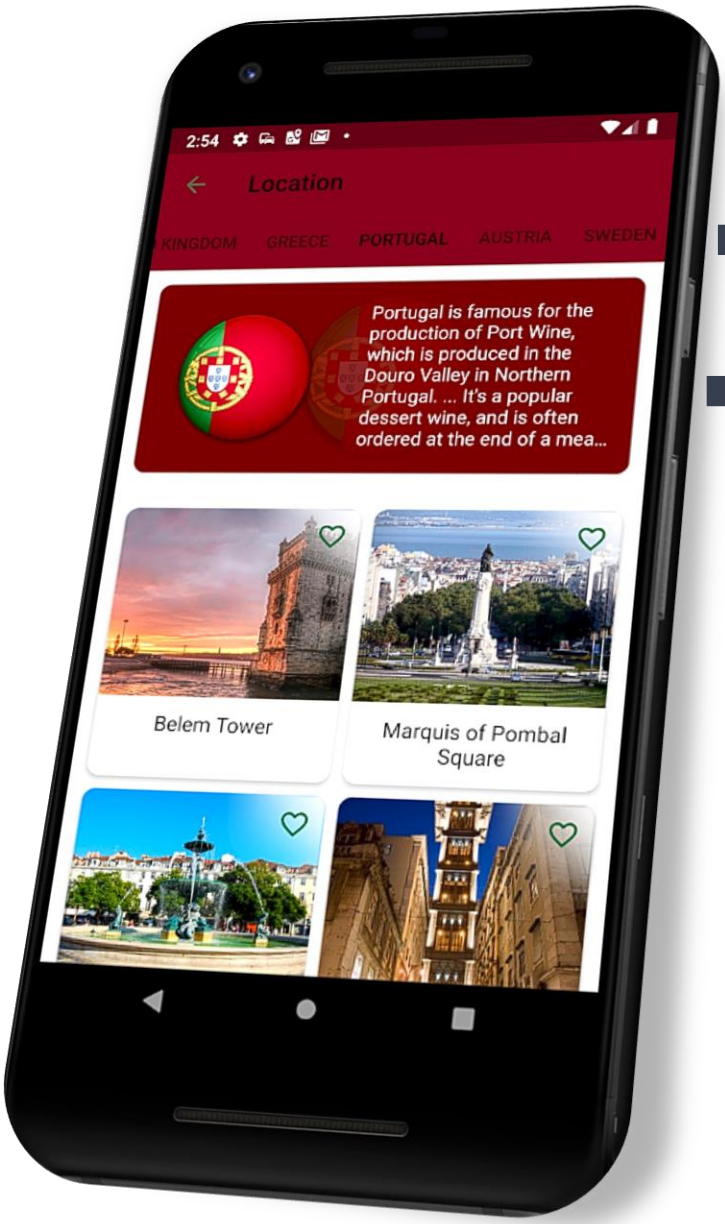
Test Scenario ID	First Inc-1	**Test Case ID**	First Inc-1A
Test Scenario Description	First Inc- Check Increment Features against Test Cases	**Test Priority**	High
Pre-Requisite	Login validated	**Post-Requisite**	Home Page Open

Test No.	Actions	Input(s)	Output		Successful ?
			Expected Output	Actual Output	
1	Display all the Countries in a Menu.	*Open Home Page.*	The different Countries are displayed in a menu with the respective names and flags.	*The different Countries are displayed in a menu with the respective names and flags.*	✓
2	Display a list of all the Countries on the top bar, of the selected country page, that allows to switch between countries.	*Select a different Country on the Top Bar list.*	The country page changes to the selected country, on the top bar list.	*The country page changes to the selected country, on the top bar list.*	✓
3	Display the Country's Information and flag.	*Input not required.*	The chosen country displays its information and flag image.	*The chosen country displays its information and flag image.*	✓
4	Display a pop up with the complete country's information.	*Click on the country's information card view.*	A pop-up view displays the country complete information.	*A pop-up view displays the country complete information.*	✓

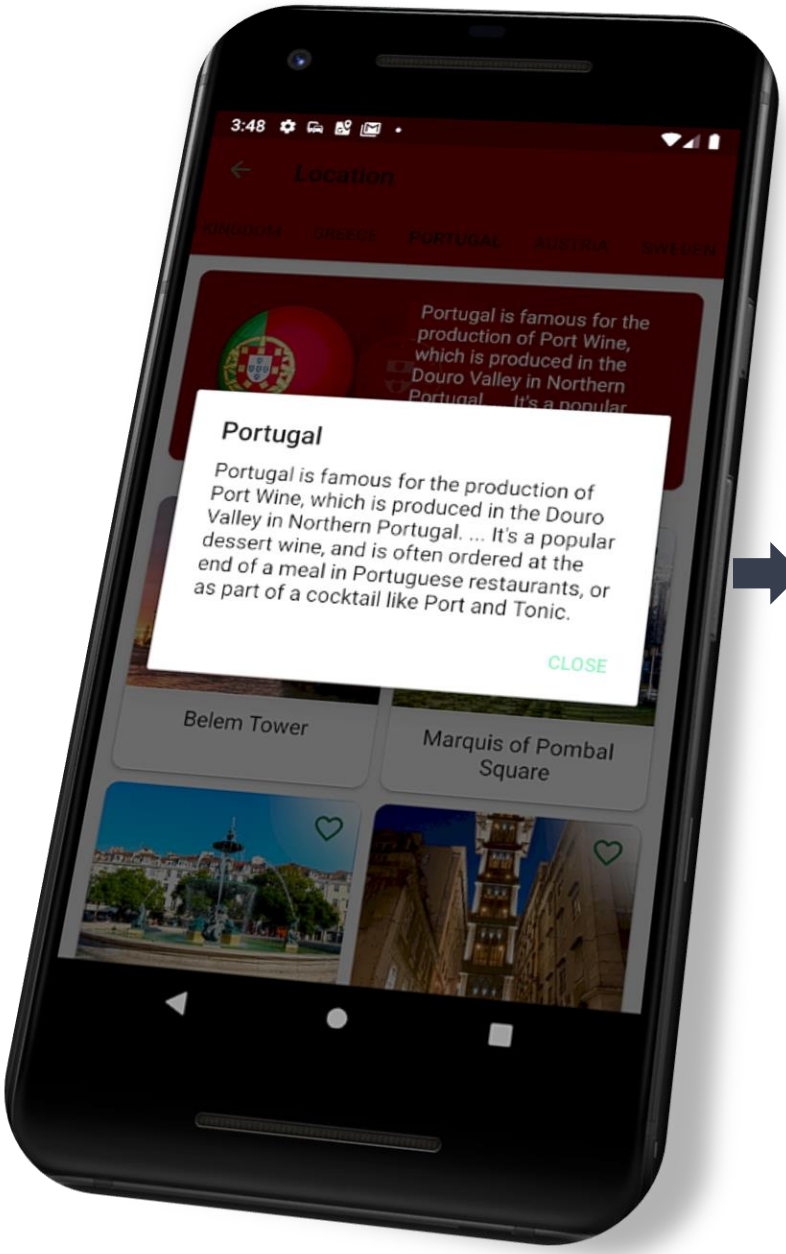
First Inc-1A-Test Screenshots:

Test No.1:

Tests No.2 & No.3:



Test No.4:


3:48
Location
PORTUGAL
Portugal is famous for the production of Port Wine, which is produced in the Douro Valley in Northern
Portugal
Portugal is famous for the production of Port Wine, which is produced in the Douro Valley in Northern Portugal. ... It's a popular dessert wine, and is often ordered at the end of a meal in Portuguese restaurants, or as part of a cocktail like Port and Tonic.
CLOSE
Belem Tower
Marquis of Pombal Square
Complete Country Information

2nd Increment Testing

ID	Requirement
FR.5	Monuments Menu
FR.6	Monument's Info

Test Cases:

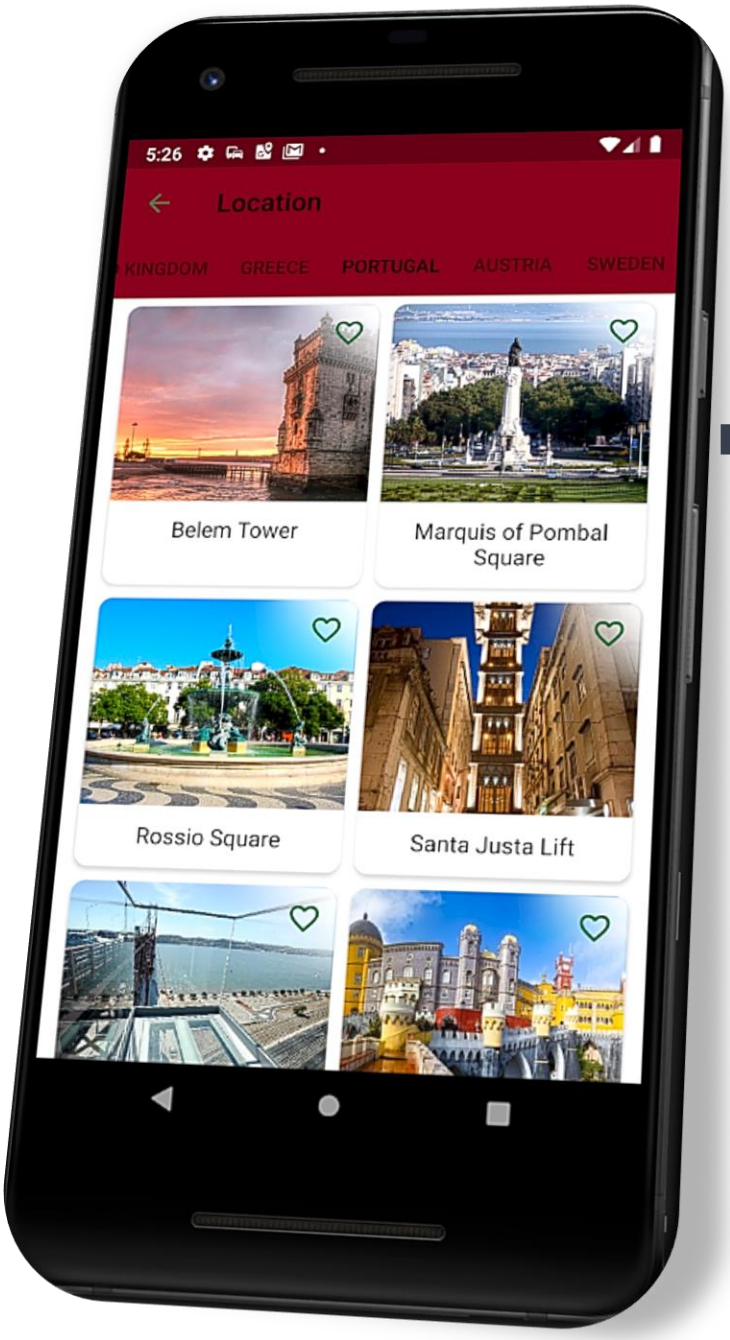
Test Scenario ID	Second Inc-2	**Test Case ID**	Second Inc-2B
Test Scenario Description	Second Inc-Check Increment Features against Test Cases	**Test Priority**	High
Pre-Requisite	Choice of a Country	**Post-Requisite**	NA

Test No.	***Actions***	***Input(s)***	***Output***		***Successful ?***
			Expected Output	***Actual Output***	
1	Display all the Monuments from the respective country, in a menu.	*Input not required.*	The Monuments from the respective country are displayed, in a menu, with a picture and the name of the Monument.	*The Monuments from the respective country are displayed, in a menu, with a picture and the name of the Monument.*	✓
2	After a Monument is selected, the respective Monument's page opens.	*Click on a Monument*	The right Monument page opens.	*The right Monument page opens.*	✓

Test No.	*Actions*	*Input(s)*	*Output*		*Successful ?*
			Expected Output	*Actual Output*	
3	The Monument's Page displays all the Monument's Information from the API.	*Input not required.*	All the information about the chosen monument is displayed, in the right order.	*All the information about the chosen monument is displayed, in the right order.*	✓

Second Inc-2B-Test Screenshots:

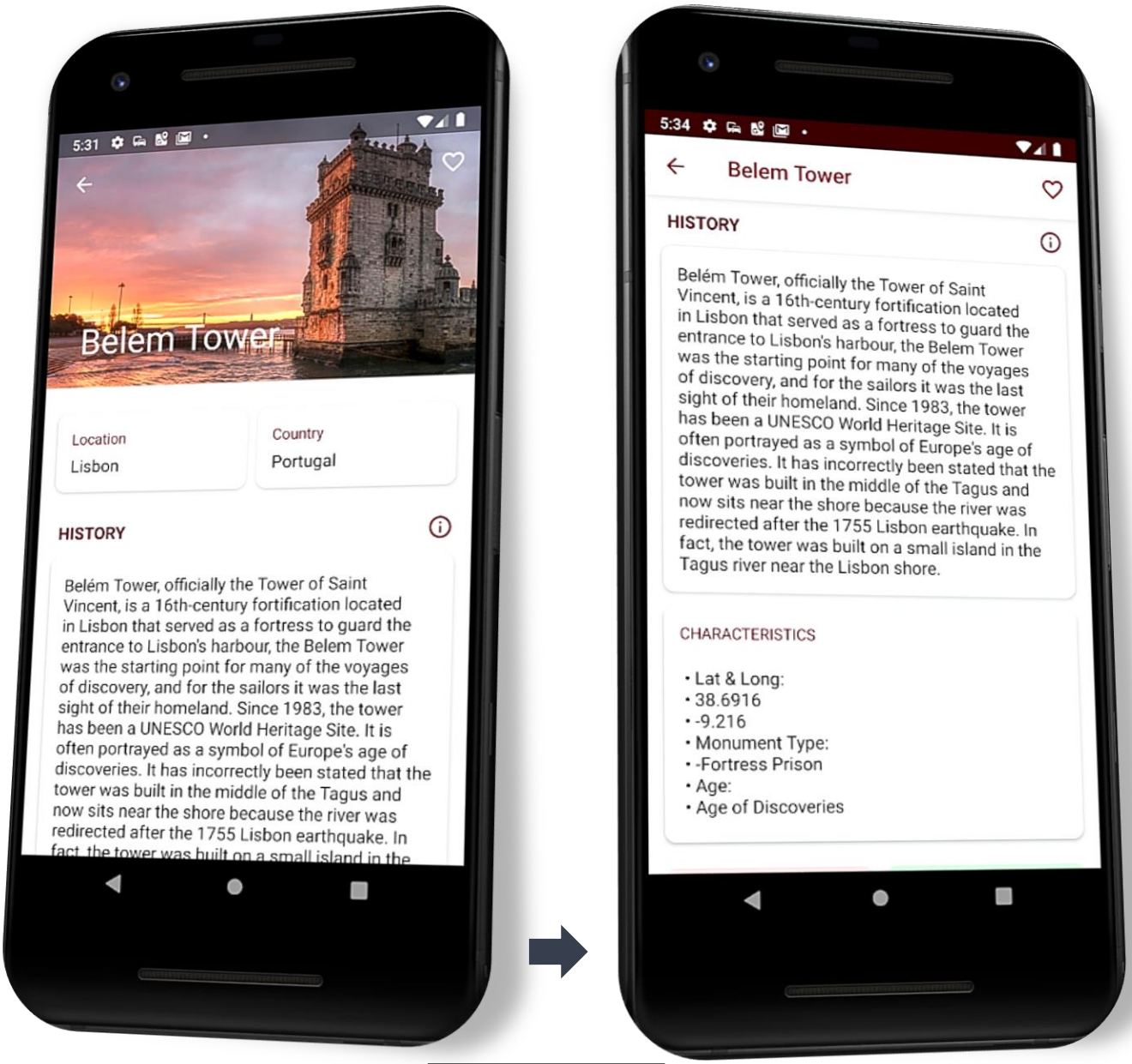
Test No.1:



Test No.2 & No.3:

Monument's Page

Monument's Information



Scrolling Down

3rd Increment Testing

ID	Requirement
FR.9	Monuments Map Location
FR.14	Most Famous Monuments

Test Cases:

Test Scenario ID	Third Inc-3	**Test Case ID**	Third Inc-3C
Test Scenario Description	Third Inc-Check Increment Features against Test Cases	**Test Priority**	High
Pre-Requisite	NA	**Post-Requisite**	NA

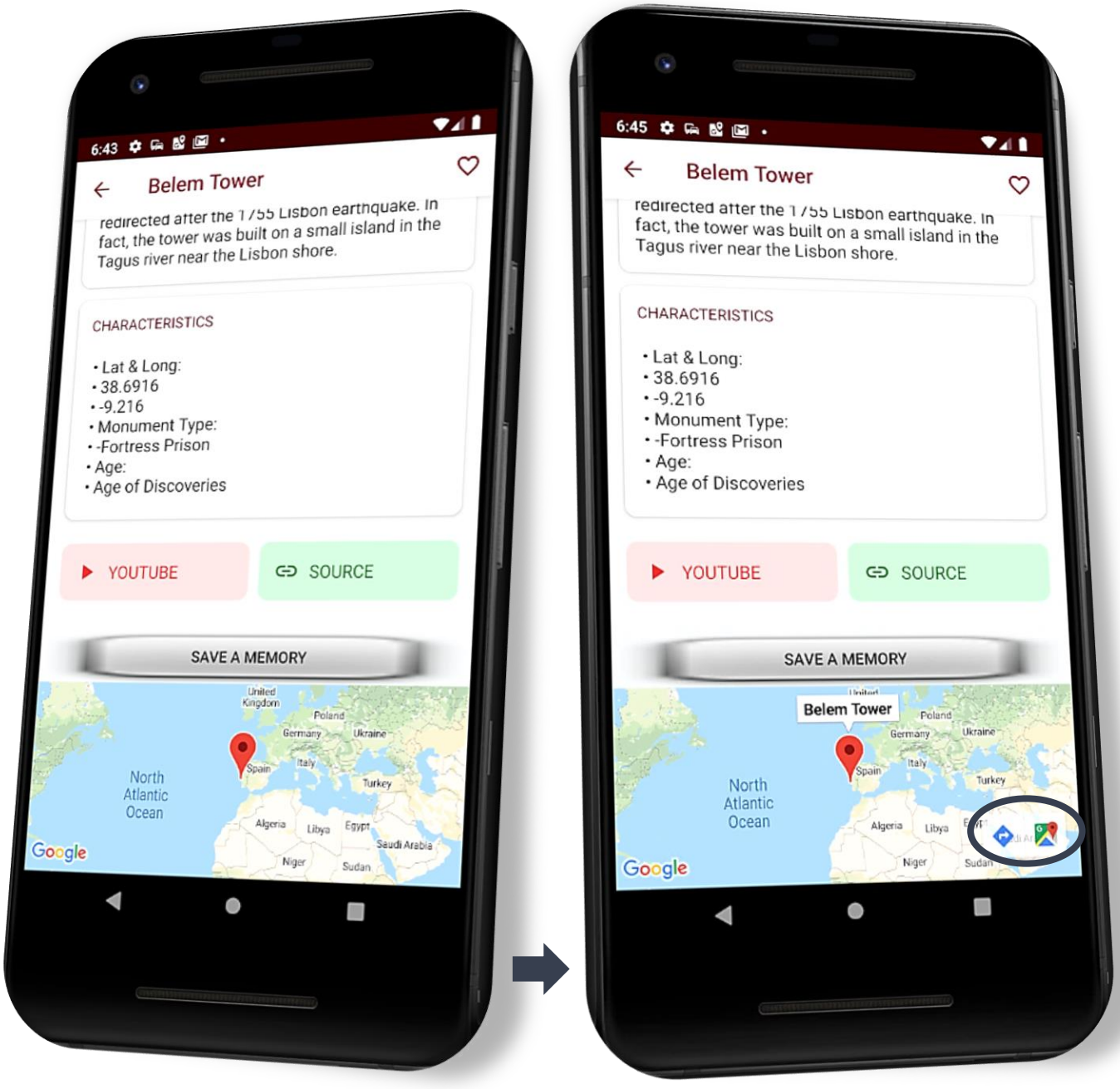
Test No.	***Actions***	***Input(s)***	***Output***		***Successful ?***
			Expected Output	***Actual Output***	
1	Map Fragment appears on the bottom of each Monument's page.	*Scroll down on a Monument's page.*	A map appears on the bottom of the Monument's page.	*A map appears on the bottom of the Monument's page.*	✓
2	Map displays marker with the monument's location.	*Input not required.*	A red marker is located on the map in the monument's location.	*A red marker is located on the map in the monument's location.*	✓

Test No.	*Actions*	*Input(s)*	*Output*		*Successful ?*
			Expected Output	*Actual Output*	
3	Marker provides directions to the Monument via 'Google Maps'.	*Click on the Marker, then click on directions icon.*	Google Maps app, or website, opens with the directions to the respective Monument.	*Google Maps app, or website, opens with the directions to the respective Monument.*	✓
4	A list of the Most Famous Monuments appears on the Home page.	*Input not required.*	List with the Most Famous Monuments, present on the API, are displayed in the Home page.	*List with the Most Famous Monuments, present on the API, are displayed in the Home page.*	✓
5	Most famous monuments list, takes the user directly to the Monument's page.	*Click on a Monument from the list.*	Opens the respective monument's page.	*Opens the respective monument's page.*	✓

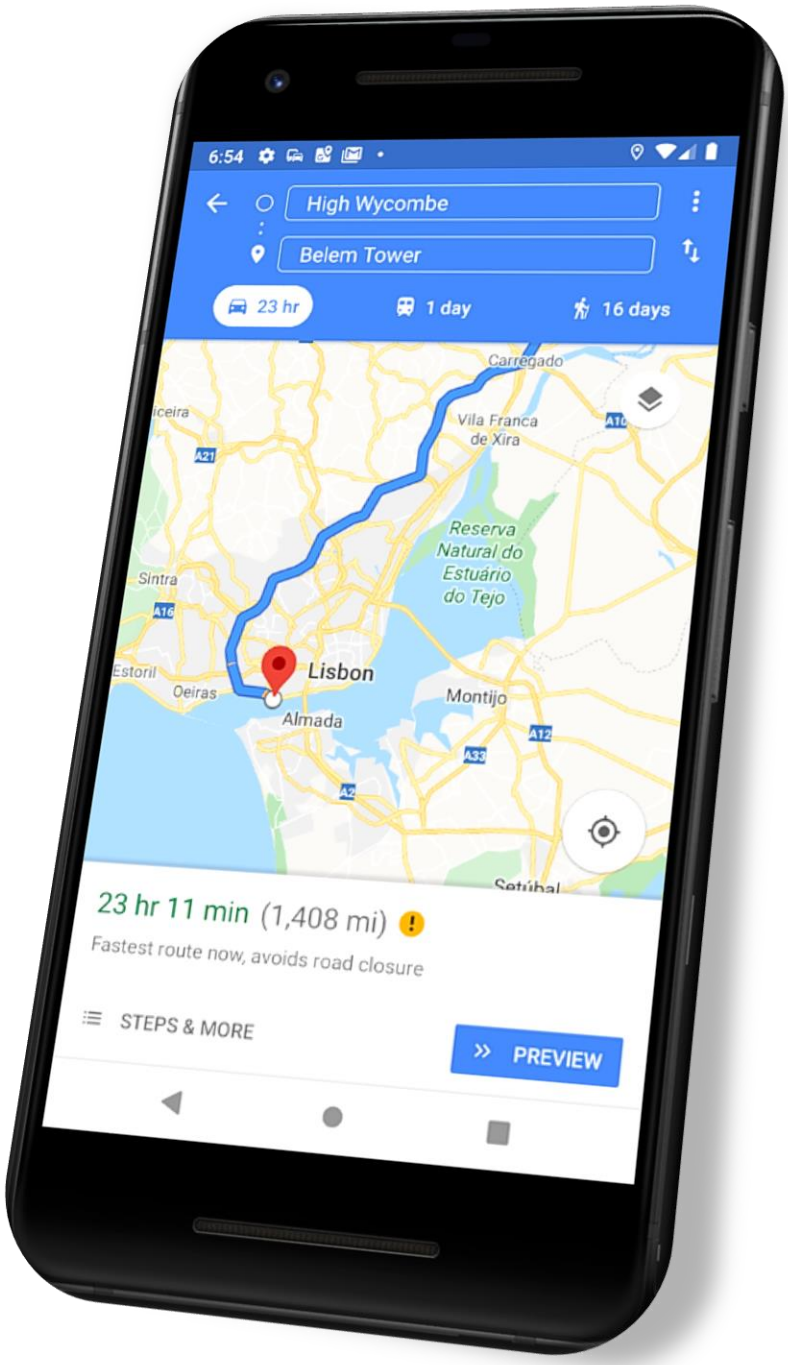
Third Inc-3C-Test Screenshots:

Test No.1 & No.2:

Marker Click

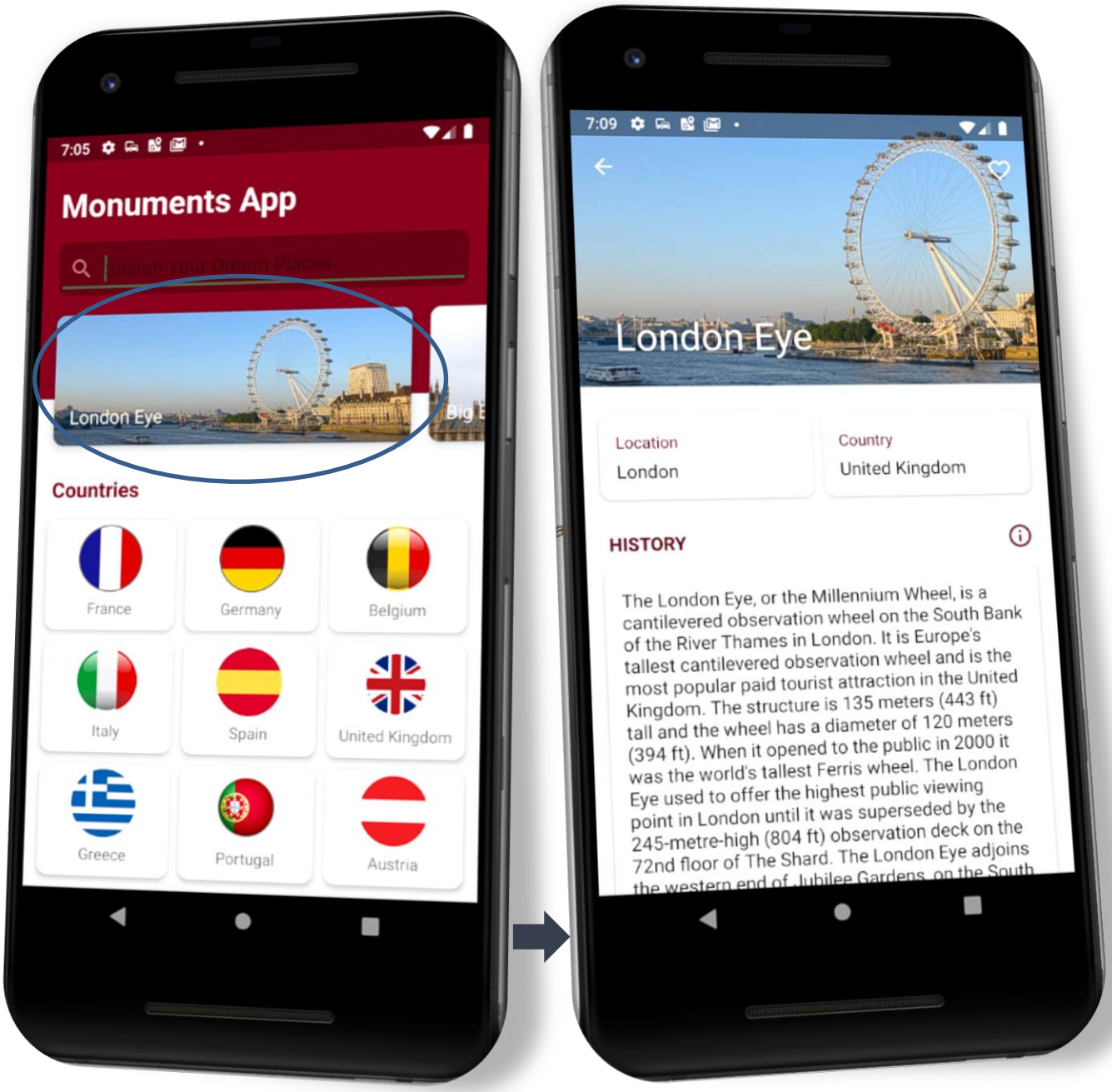


Test No.3:


6:54
High Wycombe
Belem Tower
23 hr
1 day
16 days
Carregado
Vila Franca de Xira
Reserva Natural do Estuário do Tejo
Sintra
Estoril
Oeiras
Lisbon
Almada
Montijo
Setúbal
23 hr 11 min (1,408 mi)
Fastest route now, avoids road closure
STEPS & MORE
PREVIEW

Test No.4 & No.5:

After Click



4rd Increment Testing

ID	Requirement
FR.18	Search Monument

Test Cases:

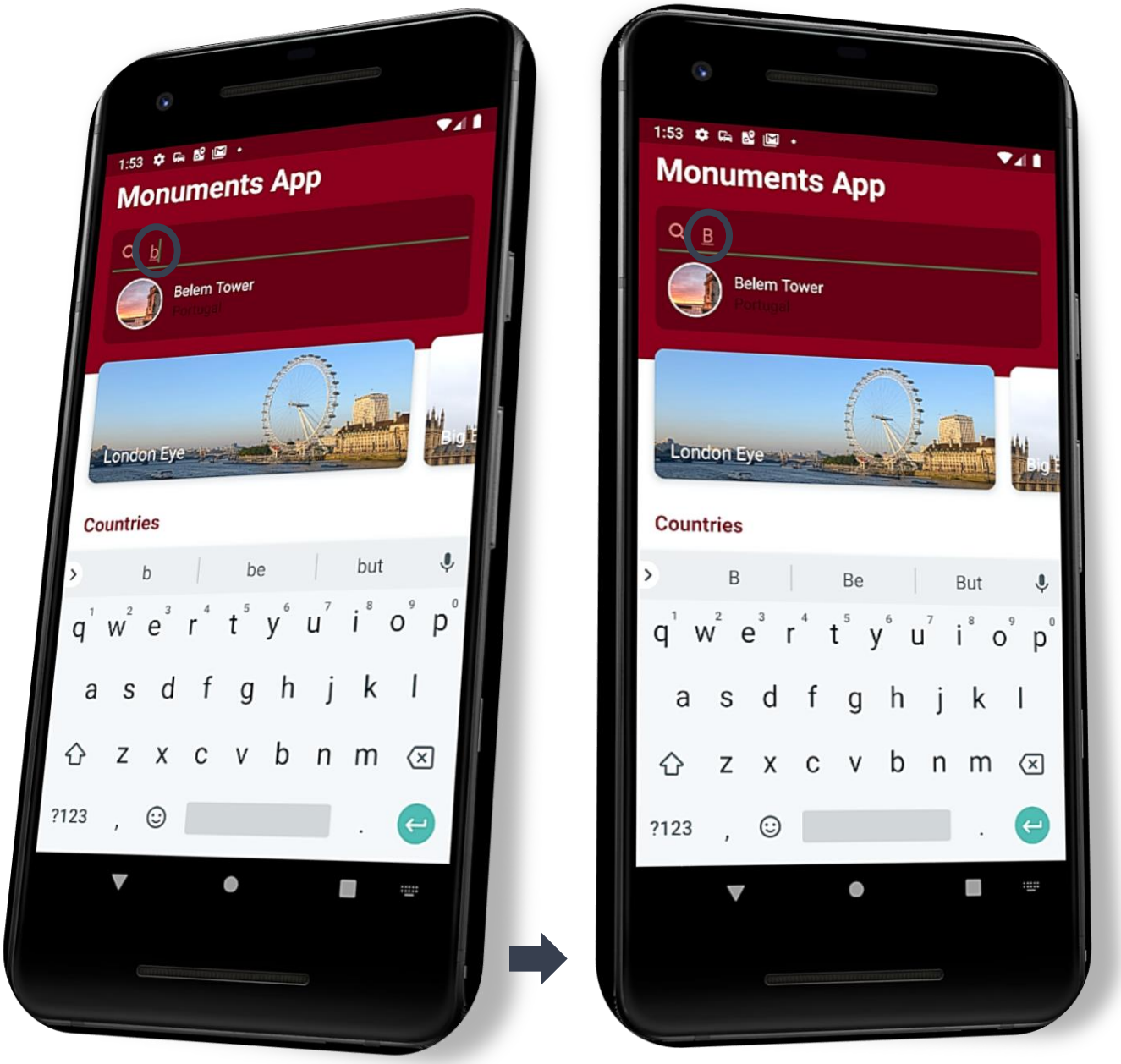
Test Scenario ID	Fourth Inc-4	Test Case ID	Fourth Inc-4D
Test Scenario Description	Third Inc-Check Increment Features against Test Cases	Test Priority	High
Pre-Requisite	Populated Database	Post-Requisite	NA

Test No.	*Actions*	*Input(s)*	*Output*		*Successful ?*
			Expected Output	*Actual Output*	
1	Type a number in the search engine.	*Insert number '1'.*	No monument is displayed.	*No monument is displayed.*	✓
2	Type a lower-case and an uppercase letter.	*Insert 'b' then insert 'B'.*	Search is not case sensitive, both inputs should show the same result.	*Both inputs should show the same result.*	✓
3	Type a letter that is common to both monuments.	*Type the letter 'e'.*	Both elements displayed.	*Both elements displayed.*	✓
4	Type the name of a country.	*Type 'Portugal' .*	Portugal monuments displayed.	*Portugal monuments displayed.*	✓

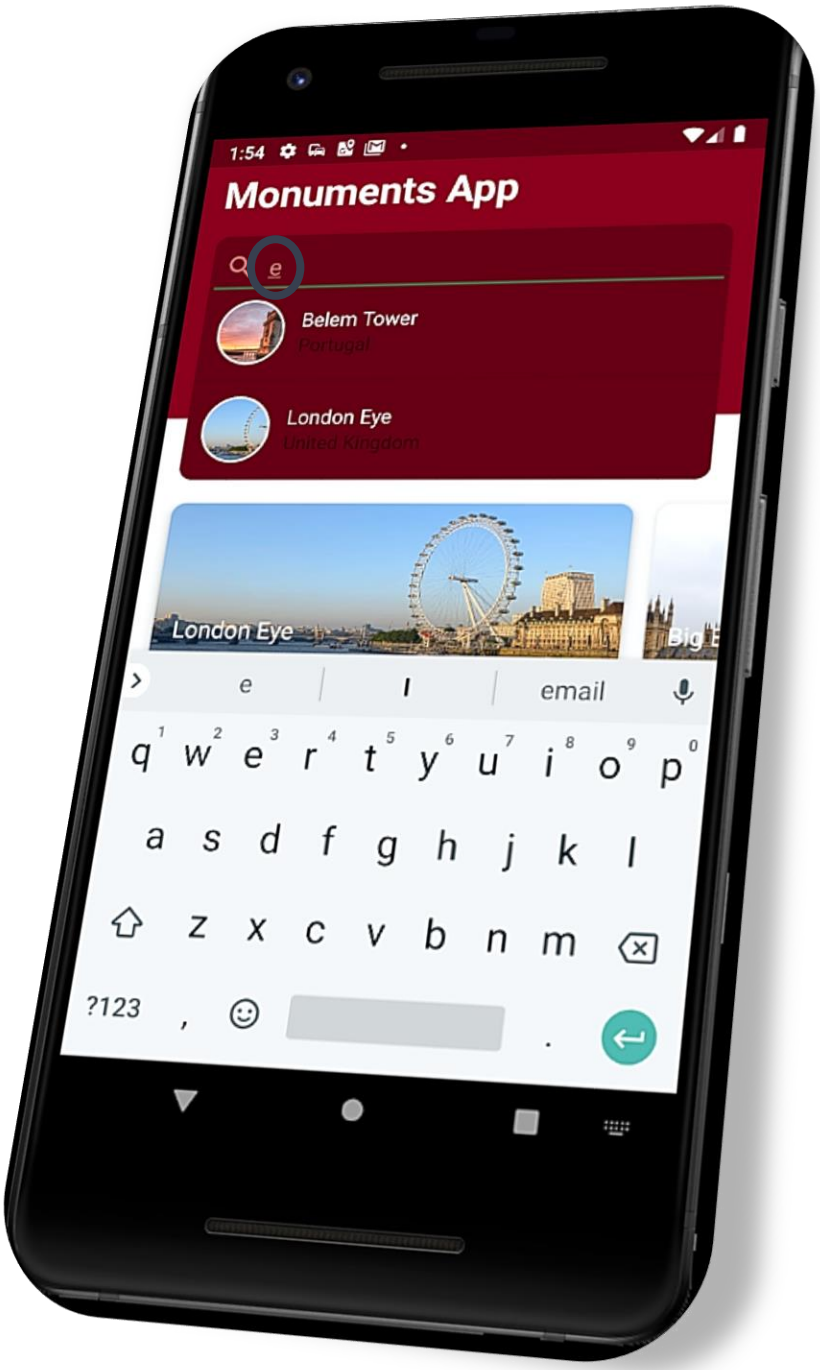
Fourth Inc-4D-Test Screenshots:

Test No.1:

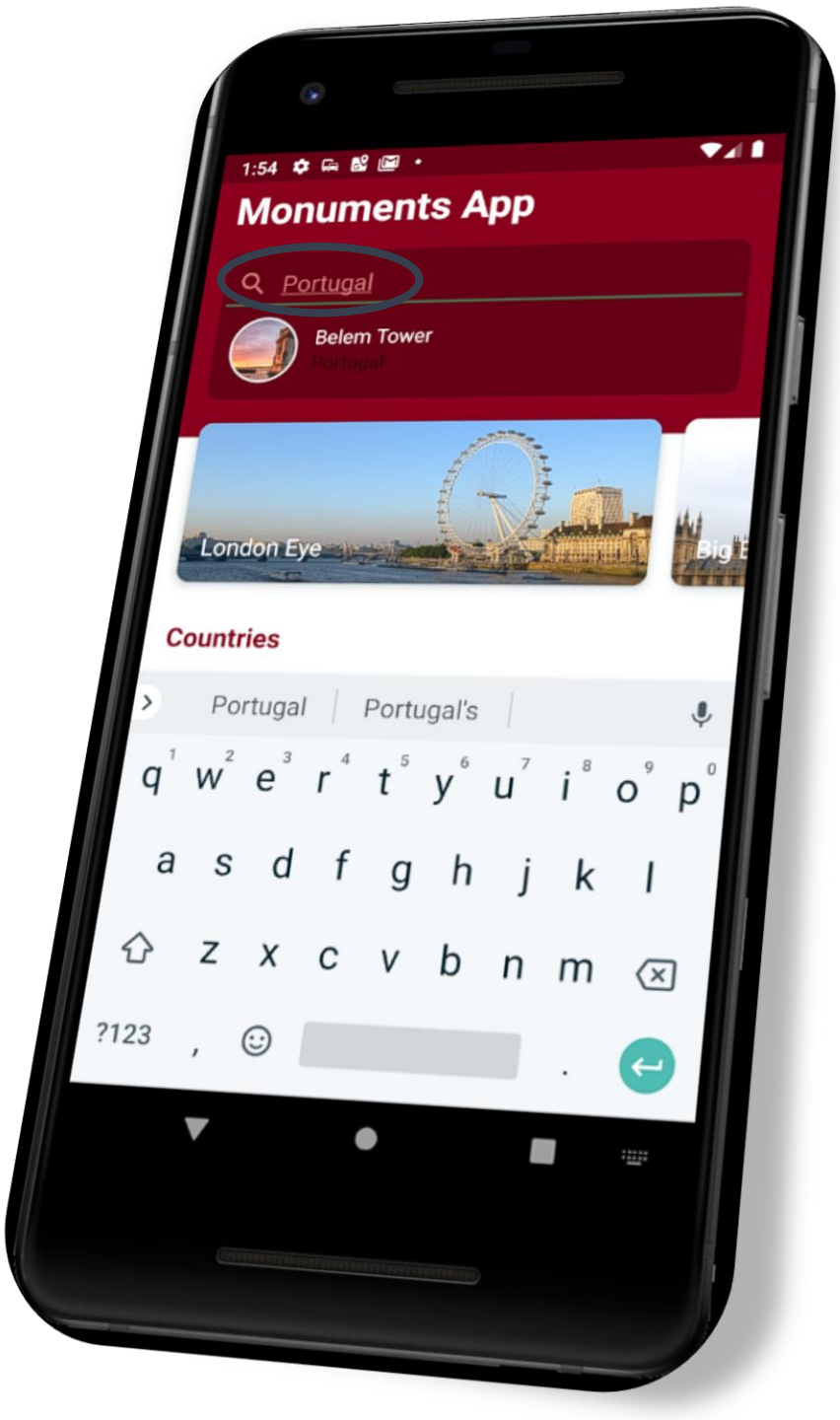
Test No.2:


1:53
Monuments App
b
Belem Tower
Portugal
London Eye
Countries
b
be
but
1:53
Monuments App
B
Belem Tower
Portugal
London Eye
Countries
B
Be
But

Test No.3:


1:54
Monuments App
e
Belem Tower
Portugal
London Eye
United Kingdom
London Eye
e
I
email
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

Test No.4:


1:54
Monuments App
Portugal
Belem Tower
Portugal
London Eye
Countries
Portugal
Portugal's
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

5th Increment Testing

ID	Requirement
FR.7	Monument's Video
FR.8	Information Source

Test Cases:

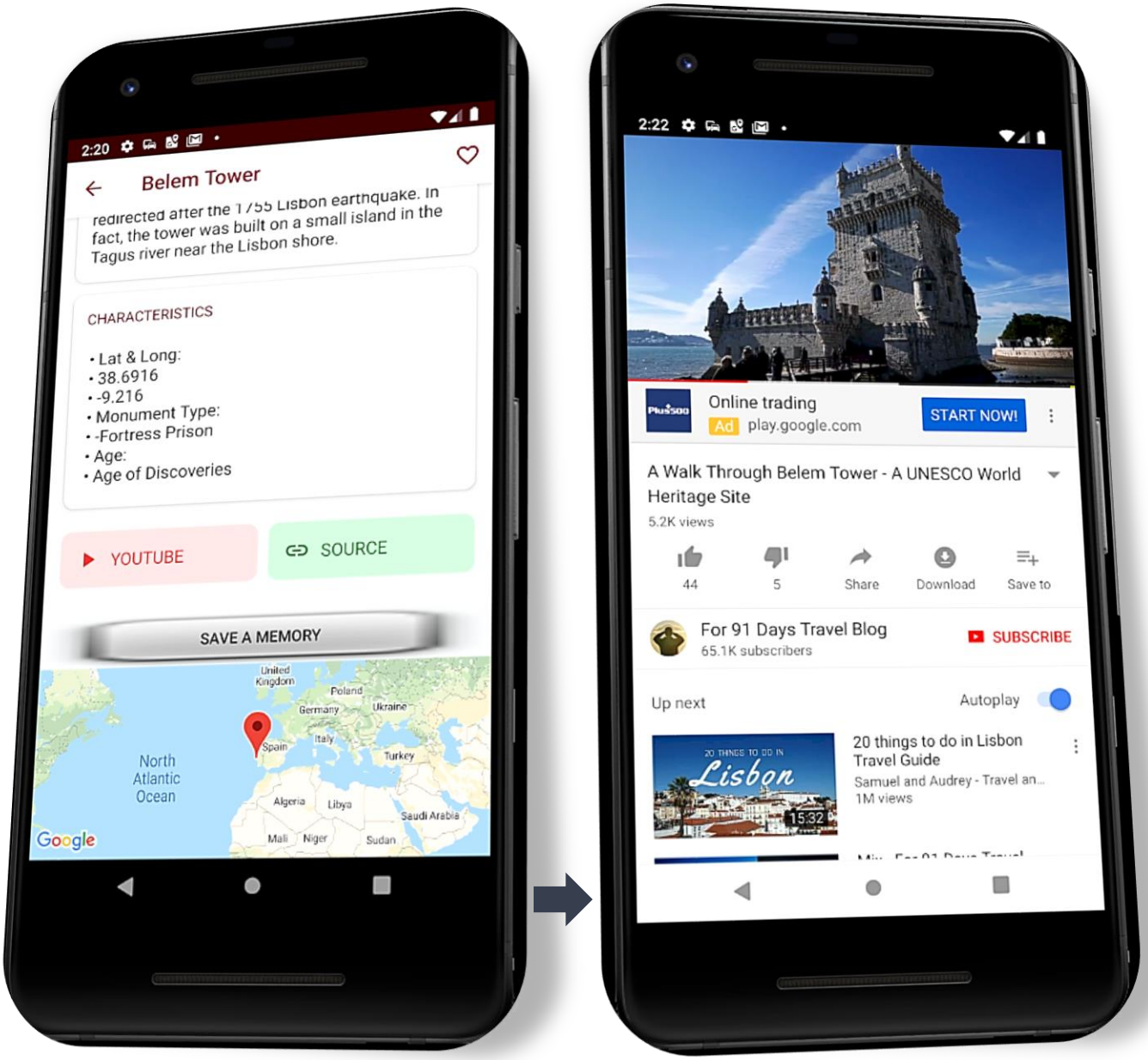
Test Scenario ID	Fifth Inc-5	**Test Case ID**	Fifth Inc-5E
Test Scenario Description	Fifth Inc-Check Increment Features against Test Cases	**Test Priority**	Medium
Pre-Requisite	NA	**Post-Requisite**	NA

Test No.	***Actions***	***Input(s)***	***Output***		***Successful ?***
			Expected Output	***Actual Output***	
1	Click on 'YOUTUBE' button.	*Button Click*	YouTube app or website opens with a video of the respective monument.	*YouTube app or website opens with a video of the respective monument.*	✓
2	Click on 'SOURCE' button	*Button Click*	The page where the information about the monument was consulted, opens in the device's browser.	*The page where the information about the monument was consulted, opens in the device's browser.*	✓

Fifth Inc-5E-Test Screenshots:

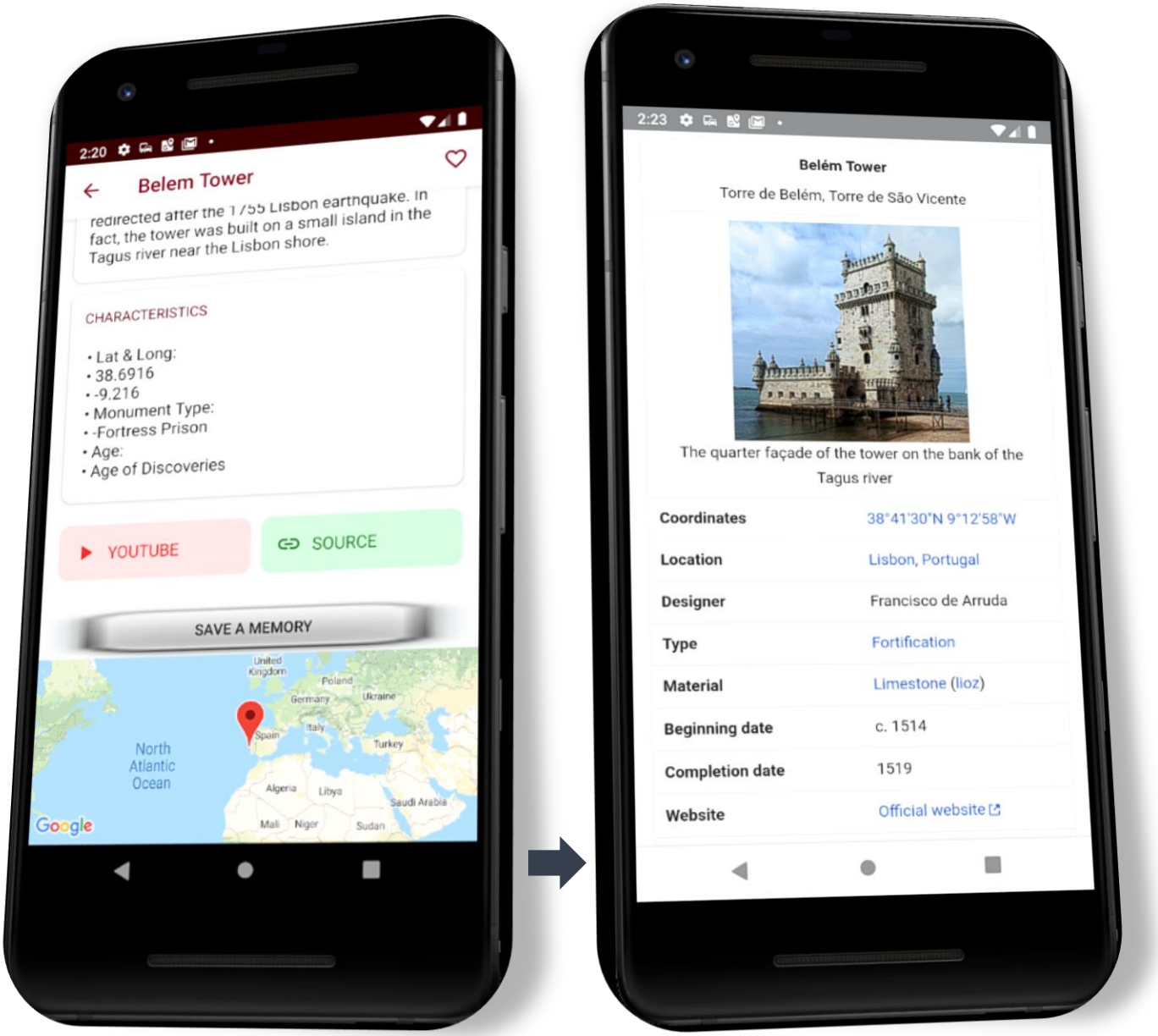
Test No.1:

YouTube Button Click



Test No.2:

Information Source Button Click



6th Increment Testing

ID	Requirement
FR.10	Take a Picture
FR.15	Create Memories

Test Cases:

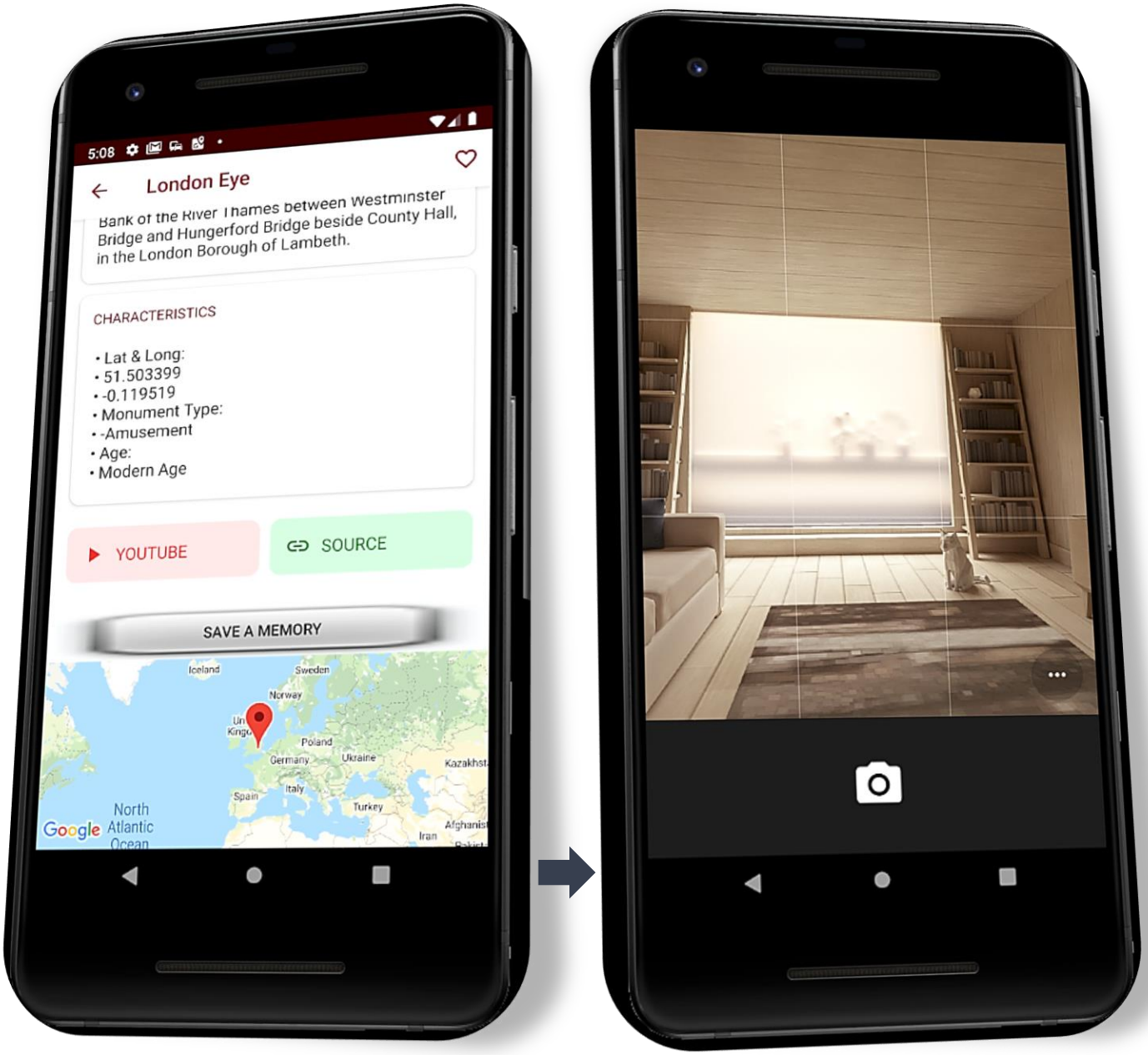
Test Scenario ID	Sixth Inc-6	**Test Case ID**	Sixth Inc-6F
Test Scenario Description	Sixth Inc-Check Increment Features against Test Cases	**Test Priority**	Medium
Pre-Requisite	NA	**Post-Requisite**	NA

Test No.	***Actions***	***Input(s)***	***Output***		***Successful ?***
			Expected Output	***Actual Output***	
1	Click on the button 'Save a Memory'.	*Button Click*	Opens device's camera.	*Opens device's camera.*	✓
2	Open 'Memories Menu' and click on 'Create New Memory'.	*Button Click*	Opens 'Create a New Memory' page.	*Opens 'Create a New Memory' page.*	✓
3	Create memory without title.	*Type a description with an image.*	Memory uploaded successful but with a default tittle *'No name'.*	*Memory still saved but with a default tittle 'No name'.*	✓

Test No.	*Actions*	*Input(s)*	*Output*		*Successful ?*
			Expected Output	*Actual Output*	
4	Create memory with no image.	*Type a title and a description without an image*	(A memory must have an image to be uploaded)- Error message "You haven't Selected any file'.	*Error message "You haven't Selected any file'.*	✓

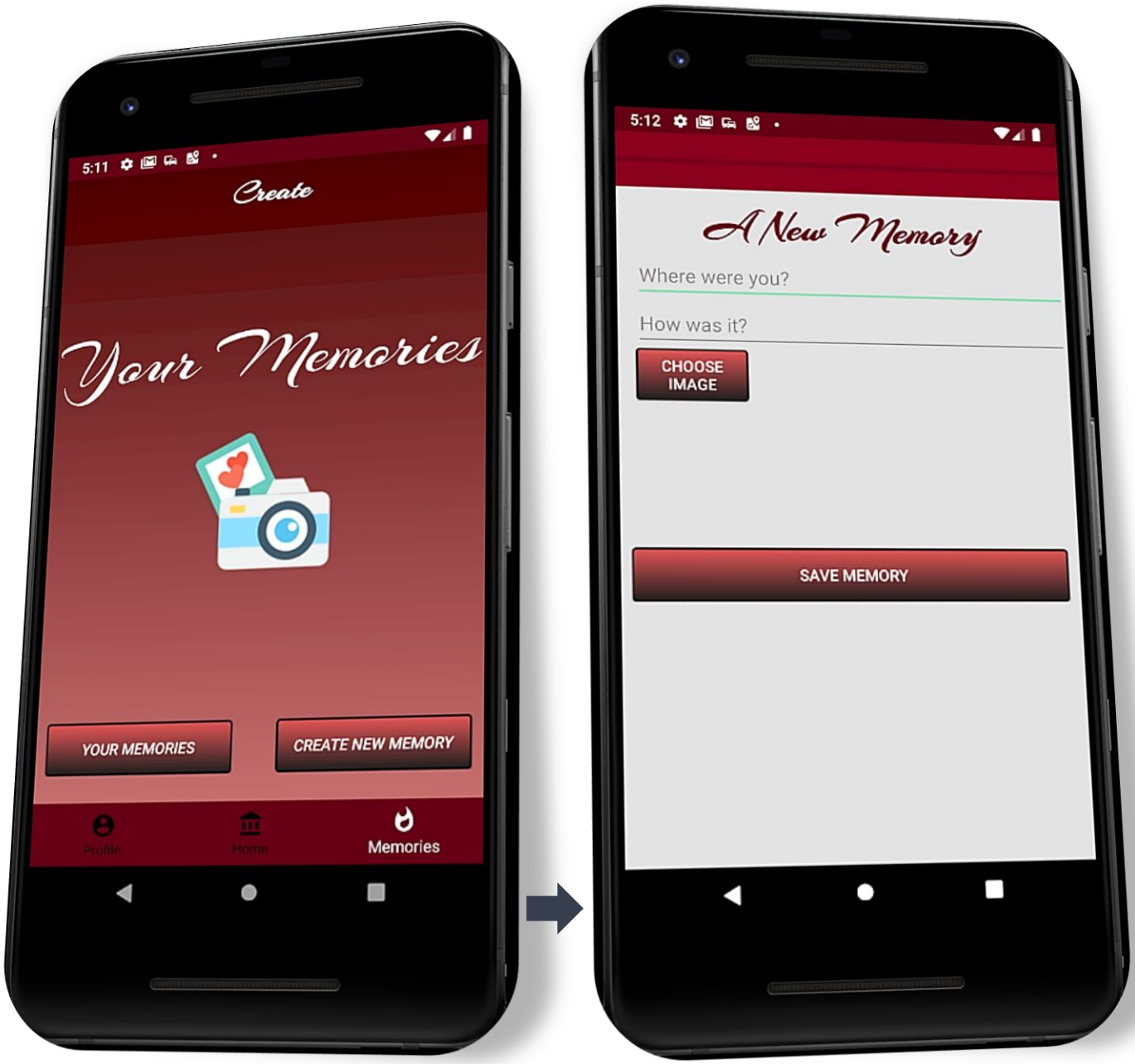
Test No.1:

Save a Memory Button Click

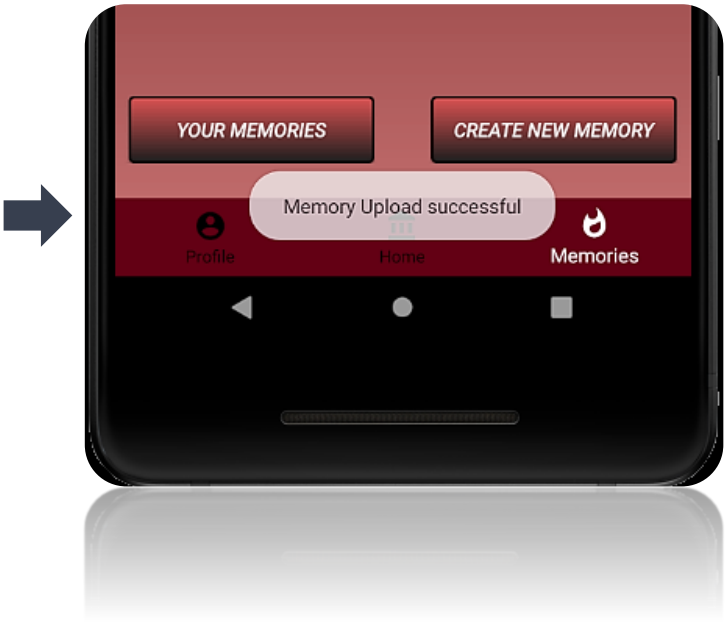


Test No.2:

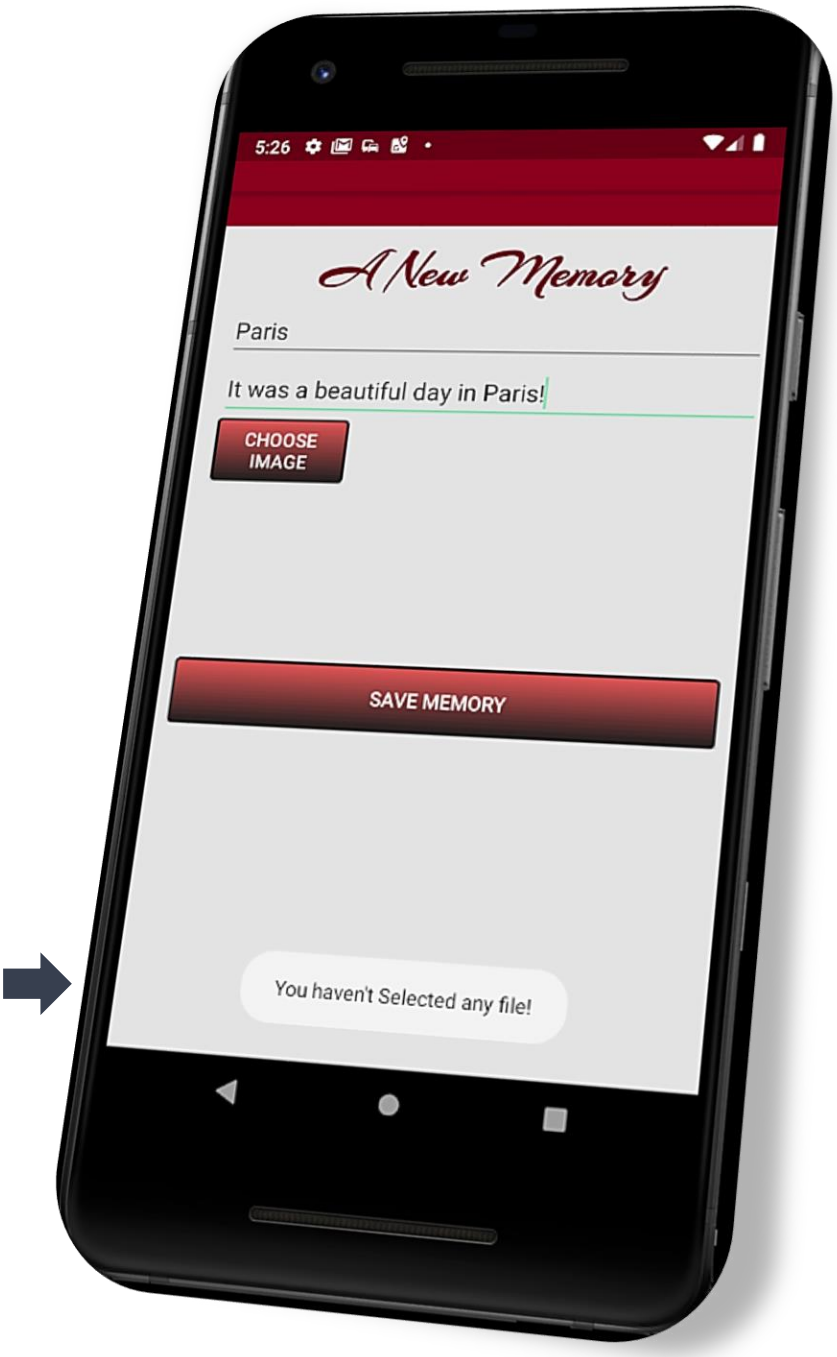
Create New Memory Button



Test No.3:


YOUR MEMORIES
CREATE NEW MEMORY
Memory Upload successful
Profile
Home
Memories

Test No.4:


5:26
A New Memory
Paris
It was a beautiful day in Paris!
CHOOSE IMAGE
SAVE MEMORY
You haven't Selected any file!

7th Increment Testing

ID	Requirement
FR.16	View Memories

Test Cases:

Test Scenario ID	Seventh Inc-7	Test Case ID	Seventh Inc-7G
Test Scenario Description	Seventh Inc-Check Increment Features against Test Cases	Test Priority	Medium
Pre-Requisite	Memories uploaded to database.	Post-Requisite	NA

Test No.	*Actions*	*Input(s)*	*Output*		*Successful ?*
			Expected Output	*Actual Output*	
1	Open 'Memories Menu' and click on 'Your Memories'.	*Button Click*	Opens 'Your Memories' page.	*Opens 'Your Memories' page.*	✓
2	Create memories and review them in 'Your memories'.	*Input not required*	Only personal Memories are displayed.	*All the database memories are displayed.*	✗
3	Click on a memory to open detail page of the same.	*Click on a Memory.*	Detail page of the memory opens.	*Detail page of the memory opens.*	✓

Test No.	*Actions*	*Input(s)*	*Output*		*Successful?*
			Expected Output	*Actual Output*	
4	Delete a Memory.	*Hold a click on a memory, choose the 'Delete' action.*	Deletes a Memory.	*Deletes a Memory.*	✓
5	Show a memory.	*Hold a click on a memory, choose the 'Show' action.*	Shows the detail page of the memory.	*Shows the detail page of the memory.*	✓

Test No.1:

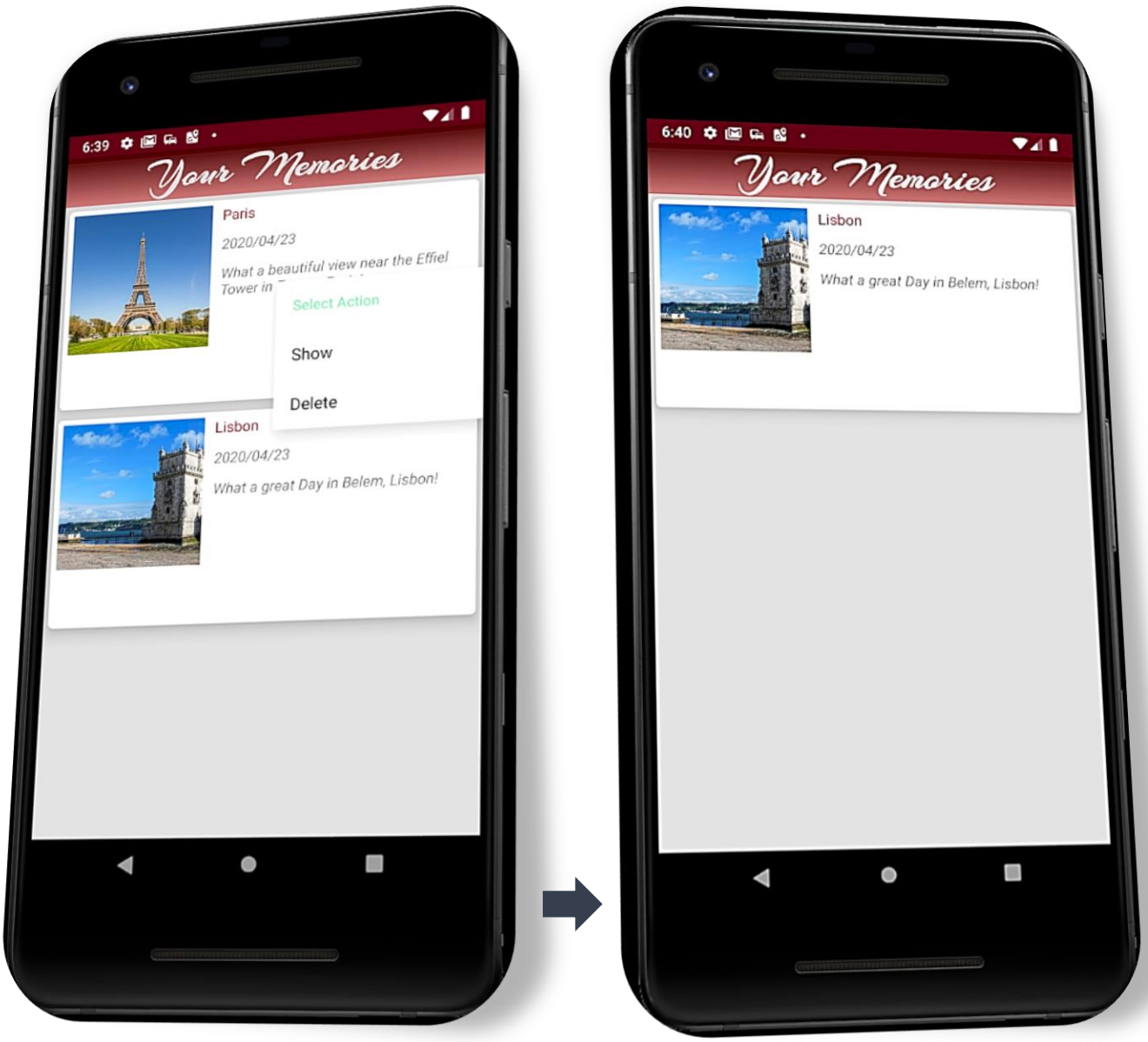
View Your Memories

Tests: No.3 & No.5:

Memory Details View

Test No.4:

Delete Memory



Test No. 2-White Box Testing:

⇒ An error was detected during the Black Box testing implementation, of the 7th Increment. In order to solve this error, the white box method will look deep within the code to locate, analyse and correct the mistakes made.

Error Identification:

Increment:	7th;
Requirement:	FR.16-View Memories;
Black Box ID:	Test No.2;
Functionality:	The logged user has his memories displayed in a list;
Error Description:	The logged user can see all the other user's memories;
Classes Responsible for *FR.16*:	'ItemsActivity.java' + 'PostsRecyclerAdapter.java'

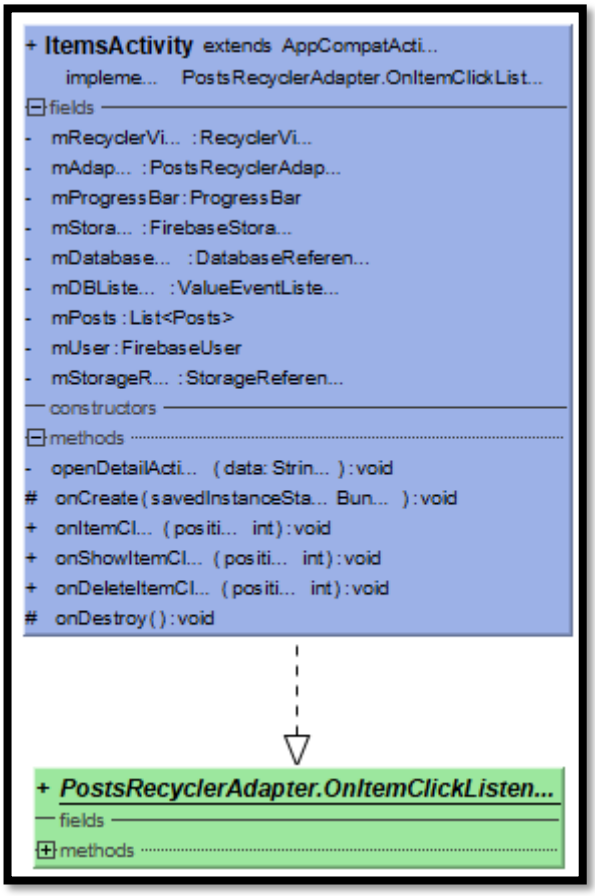


Figure 29-Class diagram of FR.16 Responsible classes

Code Analysis:

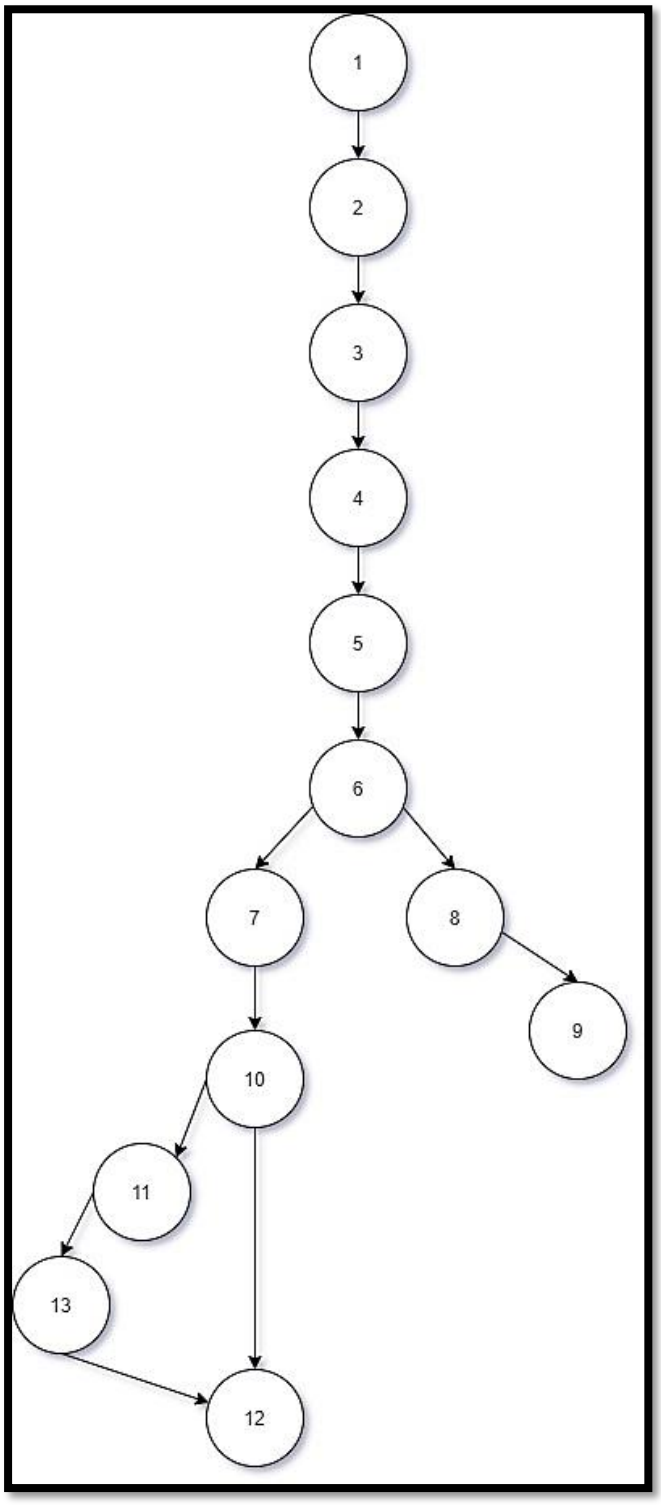
⇒ After analysing each function of the ItemsActivity.java class it is possible to understand that the functionality of this requirement is present inside the 'onCreate()' function.

```
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_posts_items);

    mRecyclerView = findViewById(R.id.mRecyclerView);
    mRecyclerView.setHasFixedSize(true);
    mRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));

    mProgressBar = findViewById(R.id.DataLoaderProgressBar);
    mProgressBar.setVisibility(View.VISIBLE);

    mPosts = new ArrayList<>();
    mAdapter = new PostsRecyclerAdapter( context: ItemsActivity.this,mPosts);
    mRecyclerView.setAdapter(mAdapter);
    mAdapter.setOnItemClickListener(ItemsActivity.this);

    mUser = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
    mStorage = FirebaseStorage.getInstance();
    mDatabaseRef = FirebaseDatabase.getInstance().getReference( path: "memories_uploads");

    mDBListener = mDatabaseRef.addValueEventListener(new ValueEventListener() {
        //Gets all the posts of the logged user
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mPosts.clear();

            for (DataSnapshot postSnapshot : dataSnapshot.getChildren()){
                Posts upload = postSnapshot.getValue(Posts.class);
                upload.setKey(postSnapshot.getKey());
                mPosts.add(upload);
            }
            mAdapter.notifyDataSetChanged();
            mProgressBar.setVisibility(View.GONE);
        }

        //Error Handling
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText( context: ItemsActivity.this,databaseError.getMessage(),Toast.LENGTH_SHORT).show();
            mProgressBar.setVisibility(View.INVISIBLE);
        }
    });
}
```

Path Testing:

In order to ensure that the set of test cases will cover each path through the onCreate() function, the following path graph analyses the independent paths of the functionality in question, its complexity and confirms that each path is executed at least once.

Path Graph:



Nodes:

1.	onCreate()
2.	setContentView()
3.	getCurrentUser()
4.	getStorage()
5.	getReference
6.	mDatabase.addValueEventListener()
7.	onDataChange()
8.	onCancelled()
9.	DatabaseError
10.	mPosts.clear()
11.	for(postSnapshot)
12.	mAdapter.notifyDataSetChanged()
13.	mPosts.add(upload)

Independent paths:

- 1,2,3,4,5,6,7,8,9;
- 1,2,3,4,5,6,7,10,12;
- 1,2,3,4,5,6,7,10,11,13,12

Cyclomatic Complexity:

⇒ In cyclomatic complexity the number of tests to test all control statements equals the cyclomatic complexity.

- *Complexity = Number of edges – Number of nodes + 1*

Complexity of this Functionality = 13-13+1 = 1

⇒ According to cyclomatic complexity levels, presented in Appendix J this result represents that the code is structured, well written, has high testability and a low cost and effort.

White Box Test Cases:

Test No.	*Test*	*Output*		*Successful ?*
		Expected Output	*Actual Output*	
1	Is the function onCreate() taking the current user Firebase Authentication ID?	Yes	*Yes*	✓
2	Is the function onCreate() taking a database storage reference?	Yes	*Yes*	✓
3	Is the function taking a snapshot of the database?	Yes	*Yes*	✓
4	Is the Data Snapshot function getting the right Database reference to take the Snapshot?	Yes	*Yes*	✓
5	Is the Database Reference searching in the logged user UID directory?	Yes	*No*	✗

Test No.1:

```
mUser = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
```

Test No.2:

```
mStorage = FirebaseStorage.getInstance();
```

Test No.3:

```
mDBListener = mDatabaseRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        mPosts.clear();

        for (DataSnapshot postSnapshot : dataSnapshot.getChildren()){
            Posts upload = postSnapshot.getValue(Posts.class);
            upload.setKey(postSnapshot.getKey());
            mPosts.add(upload);
        }
        mAdapter.notifyDataSetChanged();
        mProgressBar.setVisibility(View.GONE);
    }
```

Test No.4:

```
mDatabaseRef = FirebaseDatabase.getInstance().getReference("memories_uploads");
```

Test No.5:

⇒ After the error was located in the test case number 5, the problem was fixed with the following solution:

```
mDatabaseRef = FirebaseDatabase.getInstance().getReference("memories_uploads");
//Needs improvement so it only calls individual users storage

mDatabaseRef = FirebaseDatabase.getInstance().getReference("memories_uploads/"+mUser.getUid().toString());
```

8th Increment Testing

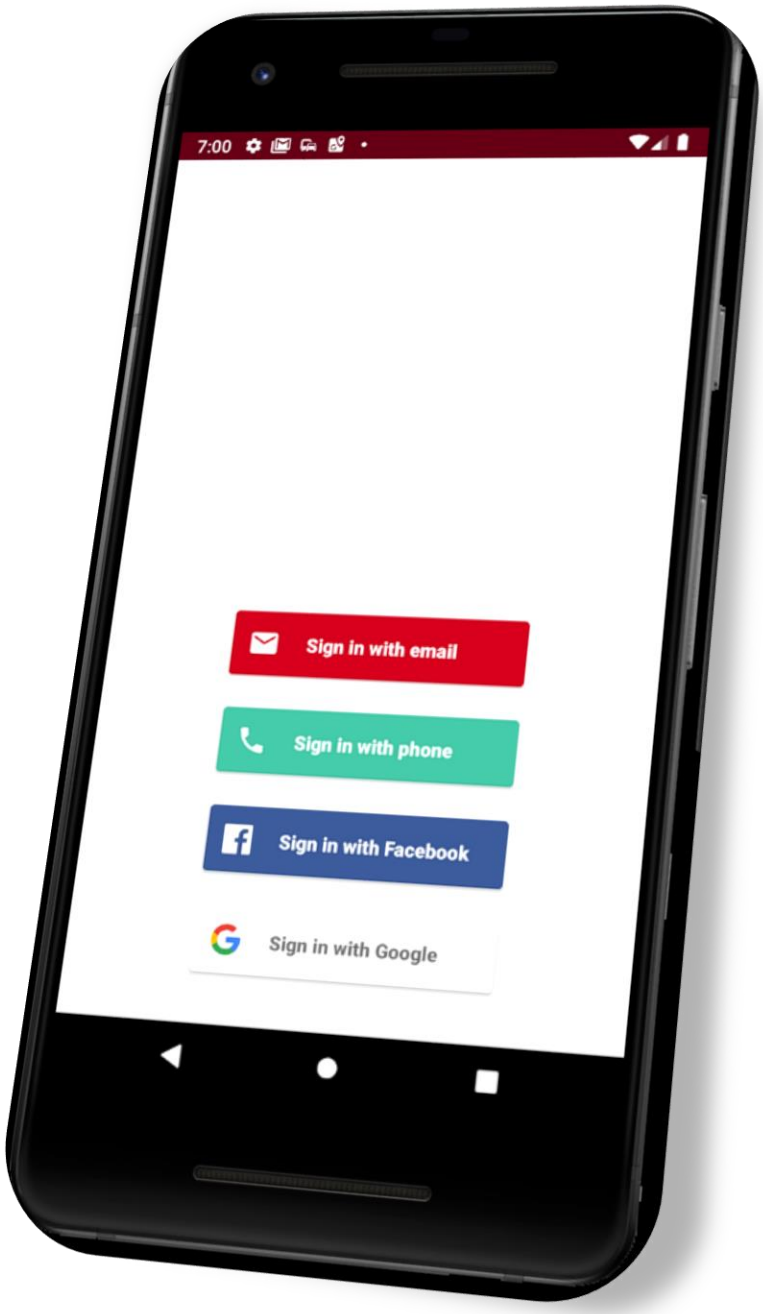
ID	Requirement
FR.11	Register
FR.12	Login

Test Cases:

Test Scenario ID	Eight Inc-8	**Test Case ID**	Eight Inc-8H
Test Scenario Description	Eight Inc-Check Increment Features against Test Cases	**Test Priority**	Low
Pre-Requisite	NA	**Post-Requisite**	NA

Test No.	***Actions***	***Input(s)***	***Output***		***Successful ?***
			Expected Output	***Actual Output***	
1	Sign in with Email	*Email and Password.*	Successful login.	Successful login.	✓
2	Sign in with Phone	*Phone number and phone number verification.*	Successful login.	Successful login.	✓
3	Sign in with Facebook.	*Facebook account login.*	Successful login.	Successful login.	✓
4	Sign in with Google.	*Google account login*	Successful login.	Successful login.	✓

Login Page:


7:00
Sign in with email
Sign in with phone
Sign in with Facebook
Sign in with Google

Implementation

This section demonstrates the process of implementing this system in a real-world environment, using the knowledge obtained in the 'Development' and 'Testing' sections to explain how users would access the app and what improvements are needed to make this system ready for release.

1. System Performance in the Real-world

As illustrated in the previous sections, the last three requirements, of the product backlog (page 58), were not implemented in this stage of the system, although that does not make this product unreliable or not operational. Thanks to the planning methods applied on the requirements prioritisation, it was possible to reassure that a minimal viable product would be ready for release after all the 'Major Priority Features' were implemented. (More information regarding future implementations at page 196)

Bellow follows the analyse of the positive and negative aspects, regarding the performance of the system, in a current product release:

Positive aspects:

1)	All the implemented Increments are operational according to the current tests, although more testing is required;
2)	The system's front-end is well responsive and built with a positive and interactive design;
3)	The system's back-end suits a large number of users;
4)	Firebase Security Rules provide a secure and private environment to each user, where their data is safe;
5)	The several login options provide flexible accessibility;
6)	A future use of Google Analytics, integrated in Firebase, would help understand user behaviour, which enables informed decisions regarding app marketing and performance optimizations.

(Google, 2020)

Negative aspects:

1)	The application is only currently working with 'Europe', other continents are still to be implemented (FR.1 & FR.2);
2)	User Profile is not yet editable and does not show complete user information (still in progress Increment);
3)	Several Monuments and Countries need to have their information stored in the database;
4)	'Favourites' feature yet to be implemented where users can store favourite monuments.

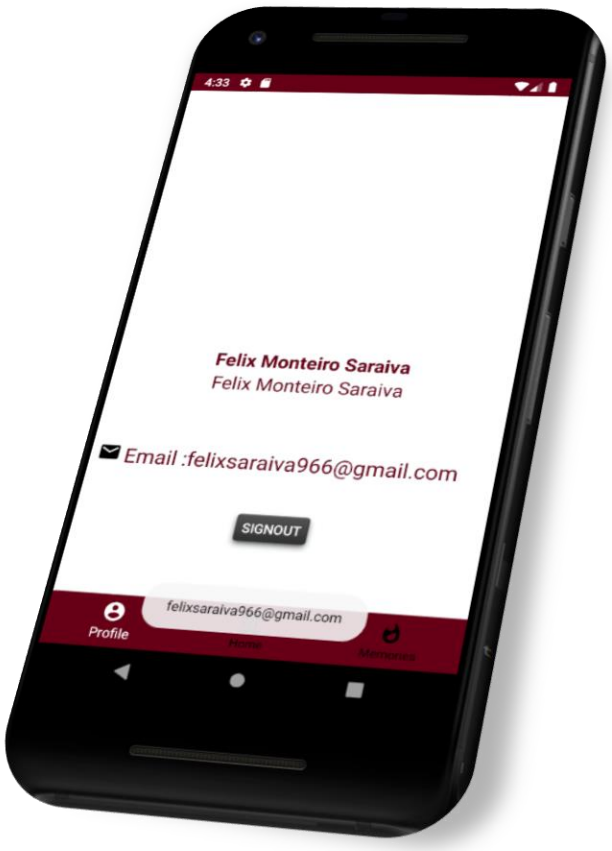


Figure 30-Profile Feature current state

- Most of the current negative aspects are based on future implementations. That is because an incremental model is implemented in this development. Therefore, those features can be implemented in future versions of the app, not influencing the current functionality of the present, "released", features.

2. Real-World User Access Process

Publishing an Android Application:

Publishing is the process of making an application available to users. At this stage the publish of this system divides itself in two main tasks:

1. **Preparation Stage**: Tasks to build the release version of the application, which users can download and install on their android-powered devices;
2. **Release Stage**: During this stage, appends the publicity, and distribution of the release version to the users.

Preparing the application for Release:

1	Configuration of the application for release;
2	Build and signing of a release version of the application;
3	Testing of the release version of the application;
4	Update of application resources for release;
5	Preparation of the back-end services that the application depends on.

(Google Developers, 2019)

Extra Steps:

1	Get a private key for signing the application;
2	Create an Icon for the application;
3	Prepare End User License Agreement (EULA);

Releasing the application to users:

1	Preparation of promotional materials;
2	Configuration of the options and uploading assets;
3	Publication of the release version of the application.

Releasing through an App Marketplace:

The best way of distributing an app to the broadest possible audience is through an app marketplace, such as Google Play. Google Play is the premier marketplace for Android apps and is particularly useful in case of distributing applications to a vast global audience. However, it is possible to distribute apps through any app marketplace or use multiple marketplaces. (Google Developers, 2019)

Conclusion

At its start, this study questions whether there can be any connections between the history of large monumental structures and the technology that changes the world every day, in this modern society.

In this conclusion and by analysing the final product of the research and development done in this project, it is possible to state, that the answer to that question, is yes.

By completing this project's objectives, it was possible to achieve its aim, to develop an interactive mobile application with a user-friendly interface that provides users with the connection between them and the monuments of the world, in a way that allows them to learn more about their history and save precious memories from visits to such places.

The research taken in the literature survey allowed the gathering of enough knowledge that helped to create the methodology of this project. This methodology represents the tools and methods that guided the development of the system, from the data gathering methods to the chosen testing methodology, the skills acquired by applying such methods, are the reason to the success of this project.

Thanks to the interviews and questionnaires, it was possible to collect enough data to build a list of the essential requirements that shaped this system. By guiding these requirements throughout the chosen agile software lifecycle, setting up and completing each increment, respecting the previously designed system, via UML notation and prototyping, it was conceivable to create a working mobile application that achieves the aim and objectives of the initial idea.

After the detailed report of the development of each increment, it was possible to test the functionality and non-functionality of the requirements. In this final stage, the system is evaluated, and the required measures to deploy such a system into the real world marked are assessed.

It is hard to describe all the skills acquired within the development of this project, but unquestionably the most valuable aspect of this study was the mistakes made.

For each mistake represents a new skill, a new lesson, a new knowledge.

Recommendations for Further Work

The research undertaken for the design and development of this project highlighted several topics on which further research would be beneficial.

The literature review and methodology cover a vast amount of techniques, all useful and required for the development of this project. The techniques were all applied with success, and there is no need to change the methods chosen for this development.

However, just has highlighted in the testing section, the development stopped before the User Profile, and the Continents features could be developed and implemented in its full, although the requirements were collected in order to fulfil them.

While planning, some requirements were declared more important than others, and because of that, some requirements had more development time than others, in order to ensure the high quality and usability of the system.

Therefore, the application is only currently working within Europe, and only a few monuments have data seeded in them. A further recommendation would be to create a Continents Menu and increase the database range of monuments and countries. Also, the user profile should be complete in order to display a user's picture and allow the user to edit their information.

After the product backlog is finished, the product would be ready for release and so, an investment in publishing the app in the Android 'Play Store' would be recommended.

In case of positive feedback from the 'Google Analytics', explained in the Implementation, then the app should be ready for a new version with new increments, that would provide the users with new interactive features. Maybe a way of connecting users, allowing them to choose which memories they want to keep private and which ones they wish to share with others, or even a feature where users are allowed to discuss historical topics of a particular monument.

The future enhancement of the social interaction provided by the application seems like the best way to attract new users to this project.

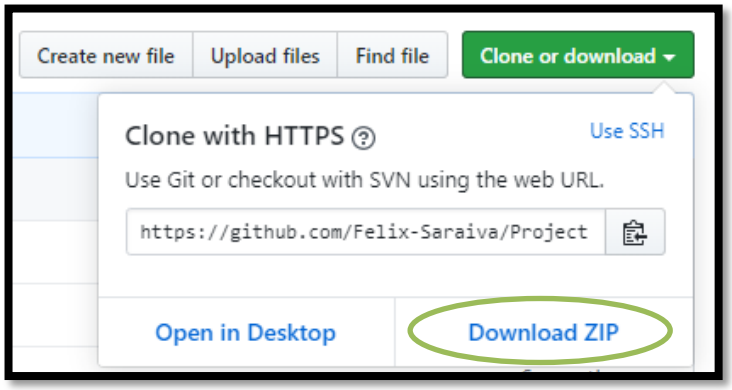
Software Artefact Download

Download Link: https://github.com/Felix-Saraiva/Project

1. Technical Instructions:

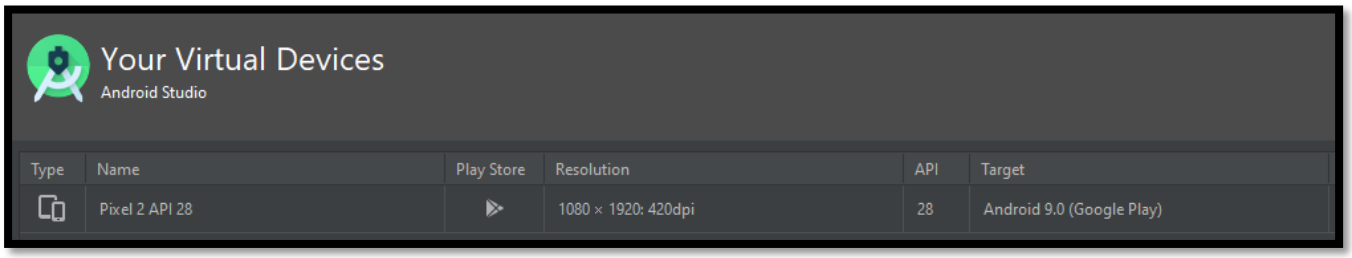
Step 1: Download Android Studio SDK;

Step 2: Download ZIP Project File from the link provided above:

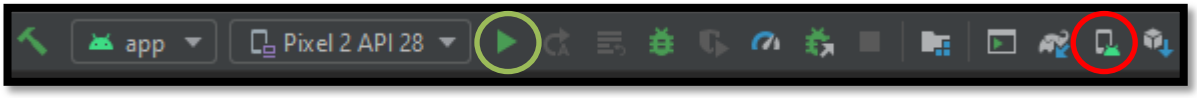


Step 3: Unzip File and open it in Android Studio;

Step 4: Set up the right Android Emulator and API version, when creating a new Virtual device select '*Pixel 2, API 28, Android 9.0':*



Step 5: Create a new virtual device by clicking on the android phone symbol and run the app by clicking on the green play button symbol.



Tip: Google Login might not work, because, like stated in the implementation, the developer would need to request for a special product release google key. – In such case use a different Login system; Also, different monuments/countries from the ones in the testing section might not be seeded with data.

Glossary

AAR Libraries: It can include everything needed to build an app, including source code, resource files, and an Android manifest.

AIDL Files: An AIDL file is used by Android app developers to enable communication between different apps.

Android: Android is a mobile operating system developed by Google.

API: An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact.

Firebase: Firebase is a mobile and web application development platform developed by Firebase Inc.

JAR libraries: A library is a more abstract concept, in java, a library is usually packaged as a JAR (Java Archive), or a collection of JARs. A jar file is zip archive containing among other files, the java class files.

Kernel: A Kernel is the central part of an operating system. It manages the operations of the computer and the hardware, most notably memory and CPU time.

Sprints: Sprint is one timeboxed iteration of a continuous development cycle. Within a Sprint, planned amount of work has to be completed and made ready for review. The term is mainly used in Scrum Agile.

UML: Unified Modelling Language

Lean: Software Development Methodology that relies on a build, measure, learn flow for continuous improvement and focuses only on what brings value to customers

MVP: Minimal Viable Product

IDE: An integrated development environment (**IDE**) is a software application that provides comprehensive facilities to computer programmers for software development.

References:

Smartsheet Inc. (2019). *Scrumban: Choosing the Middle Ground Between Scrum and Kanban*. Retrieved from Smart Sheet: https://www.smartsheet.com/scrumban-choosing-middle-ground-between-scrum-and-kanban

Abawi, D. K. (2017). *Data Collection methods*. (T. i. Health, Editor) Retrieved from gfmer: https://www.gfmer.ch/SRH-Course-2017/Geneva-Workshop/pdf/Data-collection-methods-Abawi-2017.pdf

Agile Alliance. (2019). *Kanban*. Retrieved from Agile Alliance: https://www.agilealliance.org/glossary/kanban/#q=~(infinite~false~filters~(postType~('page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~('kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~

Altexsoft. (2018, Oct 24). *The Good and the Bad of Android App Development*. Retrieved from Altexsoft: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-android-app-development/

Amazon Web Services, Inc. (2020). *What is Mobile Application Development?* Retrieved from AWS Amazon: https://aws.amazon.com/mobile/mobile-application-development/

Aston, B. (2019, September 9). *9 Of The Most Popular Project Management Methodologies Made Simple*. Retrieved from The Digital Project Manager: https://thedigitalprojectmanager.com/project-management-methodologies-made-simple/

Bhat, A. (2019). *How to Use Data Collection Tools for Market Research*. Retrieved from QuestionPro: https://www.questionpro.com/blog/data-collection-tools/

Bhat, A. (2019). *QUALITATIVE RESEARCH: DEFINITION, TYPES, METHODS AND EXAMPLES*. Retrieved from QuestionPro: https://www.questionpro.com/blog/qualitative-research-methods/

Brower, D. (2015, Aug 21). *What are the advantages and disadvantages of both top-down and bottom-up approaches to database design?* Retrieved from Quora: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-both-top-down-and-bottom-up-approaches-to-database-design

Burton, M. (n.d.). *Why Develop for Android?* Retrieved from Dummies: https://www.dummies.com/web-design-development/mobile-apps/why-develop-for-android/

CO699 Project Ethical Considerations Notes . (2018 to 2019). Bucks New University.

Coding, A. (2018, 12 27). *How to Implement Google Map in Android Studio | GoogleMap | Android Coding*. Retrieved from YouTube: https://www.youtube.com/watch?v=eiexkzCI8m8&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=15&t=0s

CodingWithMitch. (2018, 09 20). *Displaying a Google Map using a MapView*. Retrieved from YouTube: https://www.youtube.com/watch?v=118wylgD_ig&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=14&t=0s

Cohen, E. (2019, July 14). *The Definitive Guide to Project Management Methodologies*. Retrieved from Workamajig: https://www.workamajig.com/blog/project-management-methodologies

Dawson, C. W. (2015). *Projects in Computing and Information Systems* (3 ed.). Prentice Hall.

Developers, G. (n.d.). *Documentation*. Retrieved from Firebase: https://firebase.google.com/docs

Dua, P. (2019). *Essay on Questionnaire Method of Data Collection*. Retrieved from Share Your Essays: http://www.shareyouressays.com/essays/essay-on-questionnaire-method-of-data-collection/87509

Editions, T. (2019, 01 28). *World Explorer - Travel Guide*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.audioguidia.worldexplorer&hl=en_US

Flow, C. i. (2017, 12 31). *Firebase Storage - Upload and Retrieve Images - Part 1 - PREPARATIONS AND LAYOUT - Android Studio*. Retrieved from Youtube: https://www.youtube.com/watch?v=MfCiiTEwt3g&t=4s

Forelle, K. (2017, December 8). *Cohesion+and+Coupling+Bad+modularization_+low+cohesion,+high+coupling*. Retrieved from Everything Voluntary: https://everything-voluntary.com/entanglements/cohesionandcouplingbadmodularization_lowcohesionhighcoupling

Fundamentals, S. T. (2019). *Black Box Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/black-box-testing/

Fundamentals, S. T. (2019). *Unit Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/unit-testing/

Fundamentals, S. T. (2019). *White Box Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/white-box-testing/

GANTPRO. (2019). *Development*. Retrieved from GANTPRO: https://ganttpro.com/software-development-plan-template/

Google. (2019). *Woovly - The Bucket List App For Lifetime Goals*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.woovly.bucketlist&hl=pt

Google. (2020). *Easily add sign-in to your Android app with FirebaseUI*. Retrieved from Firebase Documents: https://firebase.google.com/docs/auth/android/firebaseui

Google. (2020). *Firebase Authentication*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/auth

Google. (2020). *Firebase Realtime Database*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/database

Google. (2020). *Google Analytics*. Retrieved from Mobile App Reporting in Google Analytics - Android: https://developers.google.com/analytics/devguides/collection/firebase/android

Google. (2020). *MapFragment*. Retrieved from Google APIs for Android: https://developers.google.com/android/reference/com/google/android/gms/maps/MapFragment

Google. (2020). *Understand Firebase Realtime Database Rules*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/database/security

Google Developers. (2019, 12 27). *Publish your app*. Retrieved from Android Studio Developers: https://developer.android.com/studio/publish

Guru99. (2020). *Path Testing & Basis Path Testing with EXAMPLES*. Retrieved from Guru99: https://www.guru99.com/basis-path-testing.html

HELP, S. T. (2019, April 23). *What is Incremental Testing: Detailed Explanation With Examples*. Retrieved from Software Testing Help: https://www.softwaretestinghelp.com/incremental-testing/

Howley, A. (2013, May 1). *Why Did Ancient Civilizations Build Such Huge Monuments?* Retrieved from National Geographic: https://blog.nationalgeographic.org/2013/05/01/why-did-ancient-civilizations-build-such-huge-monuments/

Jackson, J. (2020). *Quality Assurance & Testing- Test Management.* High Wycombe: Bucks New University.

Kakkar, M. a. (2013). *Risk Analysis in Mobile Application Development* (Vol. 2013). doi:10.1049/cp.2013.2351

Lardinois, F. (2019, May 07). *Kotlin is now Google's preferred language for Android app development*. Retrieved from Tech Crunch: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

Larson, E. a. (2019). *10 Steps To Creating A Project Plan*. Retrieved from ProjectTimes: https://www.projecttimes.com/articles/10-steps-to-creating-a-project-plan.html

Manifesto for Agile Software Development. (2001). Retrieved from Manifesto for Agile Software Development: https://agilemanifesto.org/

Maxey, K. (2012, November 29). *Top-Down and Bottom-Up Design*. Retrieved from Engineering.com: https://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/5023/Top-Down-and-Bottom-Up-Design.aspx

MeierJ, J., Homer, A., Hill, D., Taylor, J., Bansode, P., Wall, L., . . . Bogawat, A. (2008). *Mobile Application Architecture Guide.* Retrieved from Microsoft patterns & practices: http://robtiffany.com/wp-content/uploads/2012/08/Mobile_Architecture_Guide_v1.1.pdf

minube. (2019, October 27). *minube- travel planner & guide*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.minube.app&hl=eng&showAllReviews=true

Monus, A. (2018, Jul 20). *6 OOP Concepts in Java with examples*. Retrieved from Raygun: https://raygun.com/blog/oop-concepts-java/

Muntenescu, F. (2016, October 8). *Android Architecture Patterns Part 2:Model-View-Presenter*. Retrieved from UpDay: https://upday.github.io/blog/model-view-presenter/

Muttaqin, H. (2019, 03 22). *ANDROID FOOD APP (Meal Recipe) - MPV Design Pattern, Retrofit2, API*. Retrieved from Youtube: https://www.youtube.com/watch?v=njTHtyzaBug&list=PLT3-dzFEBix1jUBD2ygQDi0x6XKoOY8_y

Ng, V. (2019, March 10). *Non-Functional Requirement of the Mobile Development system*. Retrieved from Medium: https://medium.com/@vishwasng/non-functional-requirement-of-the-mobile-development-system-e0ed98f2a872

Nielsen, J. (1994, April 24). *10 Usability Heuristics for User Interface Design*. Retrieved from Nielsen Norman Group: https://www.nngroup.com/articles/ten-usability-heuristics/

Odendaal, A. (2017, November 27). *AO*. Retrieved from Top-down vs Bottom-up Database Design: https://ao.gl/top-down-vs-bottom-up-database-design/

Patrick, T. (2017, Feb 16). *The Relationship Between Android and Java*. Retrieved from The Iconic: https://theiconic.tech/android-java-fdbd55aadc51

Prabhu, R. (Consulted 29/10/2019). *Object Oriented Programming (OOPs) Concept in Java*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

ProgrammingWizards. (2018, 12 11). *Android S10E9 : Firebase RecyclerView Images Text - Upload, CRUD*. Retrieved from YouTube: https://www.youtube.com/watch?v=4v8Y8mPLmQc&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=20&t=1744s

Radigan, D. (2019). *What is kanban?* Retrieved from ATLASSIAN: https://www.atlassian.com/agile/kanban

Remer, R. (2013, Jul 21). *What's the difference between Model-View-Presenter and Model-View-Adapter?* Retrieved from Stack Overflow: https://stackoverflow.com/questions/15588562/whats-the-difference-between-model-view-presenter-and-model-view-adapter

Research Methodology. (2019). *Data Collection Methods*. Retrieved from Research Methodology: https://research-methodology.net/research-methods/data-collection/

shodhganga. (Consulted at 30/10/2019). *Research Methods in Software*. Retrieved from shodhganga: https://shodhganga.inflibnet.ac.in/bitstream/10603/172044/13/13_chapter%203.pdf

Solanki, R. (2019). *A Comparison of Architecture Presentation Patterns: MVC vs. MVP vs. MVVM*. Retrieved from Bacancy Technology: https://www.bacancytechnology.com/blog/mvc-vs-mvp-vs-mvvm

Sommerville, I. (2006). *Software Engineering* (8th edition ed.).

Sommerville, I. (2006). *Software Engineering* (8th edition ed.).

Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson Education.

Straughan, G. (2017). *Development That Pays*. Retrieved from Scrum vs Kanban: https://www.developmentthatpays.com/

Team, S. E. (2017, March 28th). *Software Integrity Blog*. Retrieved from Synopsys: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

Visual Paradigm. (2019). *UML - Behavioral Diagram vs Structural Diagram.* Retrieved from Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/

Wikipedia. (2019, April 22). *Integration testing*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Integration_testing

Wrike. (2019). *What is Agile Methodology in Project Management?* Retrieved from Wrike: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

Bibliography:

Smartsheet Inc. (2019). *Scrumban: Choosing the Middle Ground Between Scrum and Kanban*. Retrieved from Smart Sheet: https://www.smartsheet.com/scrumban-choosing-middle-ground-between-scrum-and-kanban

Abawi, D. K. (2017). *Data Collection methods*. (T. i. Health, Editor) Retrieved from gfmer: https://www.gfmer.ch/SRH-Course-2017/Geneva-Workshop/pdf/Data-collection-methods-Abawi-2017.pdf

Agile Alliance. (2019). *Kanban*. Retrieved from Agile Alliance: https://www.agilealliance.org/glossary/kanban/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~

Altexsoft. (2018, Oct 24). *The Good and the Bad of Android App Development*. Retrieved from Altexsoft: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-android-app-development/

Amazon Web Services, Inc. (2020). *What is Mobile Application Development?* Retrieved from AWS Amazon: https://aws.amazon.com/mobile/mobile-application-development/

Aston, B. (2019, September 9). *9 Of The Most Popular Project Management Methodologies Made Simple*. Retrieved from The Digital Project Manager: https://thedigitalprojectmanager.com/project-management-methodologies-made-simple/

Bhat, A. (2019). *How to Use Data Collection Tools for Market Research*. Retrieved from QuestionPro: https://www.questionpro.com/blog/data-collection-tools/

Bhat, A. (2019). *QUALITATIVE RESEARCH: DEFINITION, TYPES, METHODS AND EXAMPLES*. Retrieved from QuestionPro: https://www.questionpro.com/blog/qualitative-research-methods/

Brower, D. (2015, Aug 21). *What are the advantages and disadvantages of both top-down and bottom-up approaches to database design?* Retrieved from Quora: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-both-top-down-and-bottom-up-approaches-to-database-design

Burton, M. (n.d.). *Why Develop for Android?* Retrieved from Dummies: https://www.dummies.com/web-design-development/mobile-apps/why-develop-for-android/

CO699 Project Ethical Considerations Notes . (2018 to 2019). Bucks New University.

Coding, A. (2018, 12 27). *How to Implement Google Map in Android Studio | GoogleMap | Android Coding*. Retrieved from YouTube: https://www.youtube.com/watch?v=eiexkzCI8m8&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=15&t=0s

CodingWithMitch. (2018, 09 20). *Displaying a Google Map using a MapView*. Retrieved from YouTube: https://www.youtube.com/watch?v=118wylgD_ig&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=14&t=0s

Cohen, E. (2019, July 14). *The Definitive Guide to Project Management Methodologies*. Retrieved from Workamajig: https://www.workamajig.com/blog/project-management-methodologies

Dawson, C. W. (2015). *Projects in Computing and Information Systems* (3 ed.). Prentice Hall.

Developers, G. (n.d.). *Documentation*. Retrieved from Firebase: https://firebase.google.com/docs

Dua, P. (2019). *Essay on Questionnaire Method of Data Collection*. Retrieved from Share Your Essays: http://www.shareyouressays.com/essays/essay-on-questionnaire-method-of-data-collection/87509

Editions, T. (2019, 01 28). *World Explorer - Travel Guide*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.audioguidia.worldexplorer&hl=en_US

Flow, C. i. (2017, 12 31). *Firebase Storage - Upload and Retrieve Images - Part 1 - PREPARATIONS AND LAYOUT - Android Studio*. Retrieved from Youtube: https://www.youtube.com/watch?v=MfCiiTEwt3g&t=4s

Forelle, K. (2017, December 8). *Cohesion+and+Coupling+Bad+modularization_+low+cohesion,+high+coupling*. Retrieved from Everything Voluntary: https://everything-voluntary.com/entanglements/cohesionandcouplingbadmodularization_lowcohesionhighcoupling

Fundamentals, S. T. (2019). *Black Box Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/black-box-testing/

Fundamentals, S. T. (2019). *Unit Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/unit-testing/

Fundamentals, S. T. (2019). *White Box Testing*. Retrieved from Software Testing Fundamentals: http://softwaretestingfundamentals.com/white-box-testing/

GANTPRO. (2019). *Development*. Retrieved from GANTPRO: https://ganttpro.com/software-development-plan-template/

Google. (2019). *Woovly - The Bucket List App For Lifetime Goals*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.woovly.bucketlist&hl=pt

Google. (2020). *Easily add sign-in to your Android app with FirebaseUI*. Retrieved from Firebase Documents: https://firebase.google.com/docs/auth/android/firebaseui

Google. (2020). *Firebase Authentication*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/auth

Google. (2020). *Firebase Realtime Database*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/database

Google. (2020). *Google Analytics*. Retrieved from Mobile App Reporting in Google Analytics - Android: https://developers.google.com/analytics/devguides/collection/firebase/android

Google. (2020). *MapFragment*. Retrieved from Google APIs for Android: https://developers.google.com/android/reference/com/google/android/gms/maps/MapFragment

Google. (2020). *Understand Firebase Realtime Database Rules*. Retrieved from Firebase Documentation: https://firebase.google.com/docs/database/security

Google Developers. (2019, 12 27). *Publish your app*. Retrieved from Android Studio Developers: https://developer.android.com/studio/publish

Guru99. (2020). *Path Testing & Basis Path Testing with EXAMPLES*. Retrieved from Guru99: https://www.guru99.com/basis-path-testing.html

HELP, S. T. (2019, April 23). *What is Incremental Testing: Detailed Explanation With Examples*. Retrieved from Software Testing Help: https://www.softwaretestinghelp.com/incremental-testing/

Howley, A. (2013, May 1). *Why Did Ancient Civilizations Build Such Huge Monuments?* Retrieved from National Geographic: https://blog.nationalgeographic.org/2013/05/01/why-did-ancient-civilizations-build-such-huge-monuments/

Jackson, J. (2020). *Quality Assurance & Testing- Test Management.* High Wycombe: Bucks New University.

Kakkar, M. a. (2013). *Risk Analysis in Mobile Application Development* (Vol. 2013). doi:10.1049/cp.2013.2351

Lardinois, F. (2019, May 07). *Kotlin is now Google's preferred language for Android app development*. Retrieved from Tech Crunch: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

Larson, E. a. (2019). *10 Steps To Creating A Project Plan*. Retrieved from ProjectTimes: https://www.projecttimes.com/articles/10-steps-to-creating-a-project-plan.html

Manifesto for Agile Software Development. (2001). Retrieved from Manifesto for Agile Software Development: https://agilemanifesto.org/

Maxey, K. (2012, November 29). *Top-Down and Bottom-Up Design*. Retrieved from Engineering.com: https://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/5023/Top-Down-and-Bottom-Up-Design.aspx

MeierJ, J., Homer, A., Hill, D., Taylor, J., Bansode, P., Wall, L., . . . Bogawat, A. (2008). *Mobile Application Architecture Guide.* Retrieved from Microsoft patterns & practices: http://robtiffany.com/wp-content/uploads/2012/08/Mobile_Architecture_Guide_v1.1.pdf

minube. (2019, October 27). *minube- travel planner & guide*. Retrieved from Google Play: https://play.google.com/store/apps/details?id=com.minube.app&hl=eng&showAllReviews=true

Monus, A. (2018, Jul 20). *6 OOP Concepts in Java with examples*. Retrieved from Raygun: https://raygun.com/blog/oop-concepts-java/

Muntenescu, F. (2016, October 8). *Android Architecture Patterns Part 2:Model-View-Presenter*. Retrieved from UpDay: https://upday.github.io/blog/model-view-presenter/

Muttaqin, H. (2019, 03 22). *ANDROID FOOD APP (Meal Recipe) - MPV Design Pattern, Retrofit2, API*. Retrieved from Youtube: https://www.youtube.com/watch?v=njTHtyzaBug&list=PLT3-dzFEBix1jUBD2ygQDi0x6XKoOY8_y

Ng, V. (2019, March 10). *Non-Functional Requirement of the Mobile Development system*. Retrieved from Medium: https://medium.com/@vishwasng/non-functional-requirement-of-the-mobile-development-system-e0ed98f2a872

Nielsen, J. (1994, April 24). *10 Usability Heuristics for User Interface Design*. Retrieved from Nielsen Norman Group: https://www.nngroup.com/articles/ten-usability-heuristics/

Odendaal, A. (2017, November 27). *AO*. Retrieved from Top-down vs Bottom-up Database Design: https://ao.gl/top-down-vs-bottom-up-database-design/

Patrick, T. (2017, Feb 16). *The Relationship Between Android and Java*. Retrieved from The Iconic: https://theiconic.tech/android-java-fdbd55aadc51

Prabhu, R. (Consulted 29/10/2019). *Object Oriented Programming (OOPs) Concept in Java*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

ProgrammingWizards. (2018, 12 11). *Android S10E9 : Firebase RecyclerView Images Text - Upload, CRUD*. Retrieved from YouTube: https://www.youtube.com/watch?v=4v8Y8mPLmQc&list=PL2uX89Q43ixTM3uSvjXbBJE22b6zRbkmf&index=20&t=1744s

Radigan, D. (2019). *What is kanban?* Retrieved from ATLASSIAN: https://www.atlassian.com/agile/kanban

Remer, R. (2013, Jul 21). *What's the difference between Model-View-Presenter and Model-View-Adapter?* Retrieved from Stack Overflow: https://stackoverflow.com/questions/15588562/whats-the-difference-between-model-view-presenter-and-model-view-adapter

Research Methodology. (2019). *Data Collection Methods*. Retrieved from Research Methodology: https://research-methodology.net/research-methods/data-collection/

shodhganga. (Consulted at 30/10/2019). *Research Methods in Software*. Retrieved from shodhganga: https://shodhganga.inflibnet.ac.in/bitstream/10603/172044/13/13_chapter%203.pdf

Solanki, R. (2019). *A Comparison of Architecture Presentation Patterns: MVC vs. MVP vs. MVVM*. Retrieved from Bacancy Technology: https://www.bacancytechnology.com/blog/mvc-vs-mvp-vs-mvvm

Sommerville, I. (2006). *Software Engineering* (8th edition ed.).

Sommerville, I. (2006). *Software Engineering* (8th edition ed.).

Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson Education.

Straughan, G. (2017). *Development That Pays*. Retrieved from Scrum vs Kanban: https://www.developmentthatpays.com/

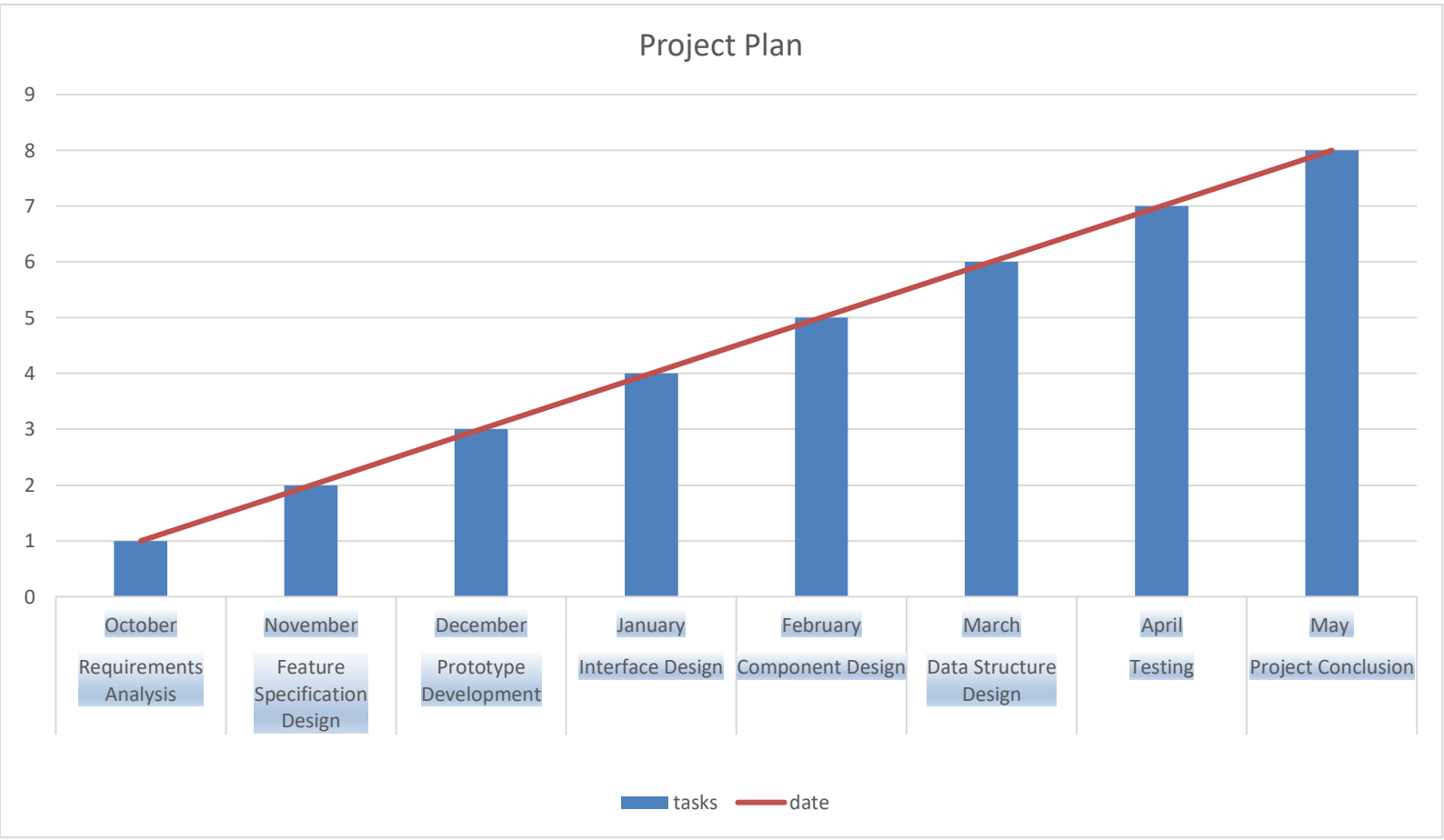
Team, S. E. (2017, March 28th). *Software Integrity Blog*. Retrieved from Synopsys: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

Visual Paradigm. (2019). *UML - Behavioral Diagram vs Structural Diagram.* Retrieved from Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/

Wikipedia. (2019, April 22). *Integration testing*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Integration_testing

Wrike. (2019). *What is Agile Methodology in Project Management?* Retrieved from Wrike: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

Appendix A: Project Plan:



Appendix B: Main Code Classes:

- **FirebaseAuth.java**

```
public class FirebaseAuth extends AppCompatActivity {

    FirebaseUser user;
    private TextView userName, userProfName , userEmail ;
    private CircleImageView userProfileImage;

    private String currentUserId;

    private static final int MY_REQUEST_CODE = 1424; //My code
    List<AuthUI.IdpConfig> providers;
    Button btn_sign_out;
    Button open_activity_button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        btn_sign_out = (Button)findViewById(R.id.btn_sign_out);
        btn_sign_out.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Logout Button Click
                AuthUI.getInstance().signOut(FirebaseAuth.this).addOnCompleteListener(new
OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        btn_sign_out.setEnabled(false);
                        showSignInOptions();

                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {

Toast.makeText(FirebaseAuth.this,""+e.getMessage(),Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
```

```
//Init provider (all the possible login options)
        providers= Arrays.asList(
                new AuthUI.IdpConfig.EmailBuilder().build(),//Email Builder
                new AuthUI.IdpConfig.PhoneBuilder().build(),//Phone Builder
                new AuthUI.IdpConfig.FacebookBuilder().build(),//Facebook Builder
                new AuthUI.IdpConfig.GoogleBuilder().build() //Google Builder
        );


        showSignInOptions();
        navigationBar();


    }

    public void navigationBar(){

        //Navigation barView
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
        //Set Home Selected
        bottomNavigationView.setSelectedItemId(R.id.profile);
        //Perform Item SelectListener
        bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()){
                    case R.id.profile:
                        startActivity(new Intent(getApplicationContext(),
FirebaseAuth.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.home:
                        startActivity(new Intent(getApplicationContext(),
HomeActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.memories:
                        startActivity(new Intent(getApplicationContext(),
PostsActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                }
                return false;
            }
        });
    }
```

```
private void displayUser(){
        //Set up user info display
        userName = (TextView) findViewById(R.id.my_username);
        userProfName = (TextView) findViewById(R.id.my_profile_name);
        userEmail = (TextView) findViewById(R.id.my_email);
        userProfileImage = (CircleImageView) findViewById(R.id.my_profile_pic);


        user = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            for (UserInfo profile : user.getProviderData()) {
                // Id of the provider (ex: google.com)
                String providerId = profile.getProviderId();

                // UID specific to the provider
                String uid = profile.getUid();

                // Name, email address, and profile photo Url
                String name = profile.getDisplayName();
                String email = profile.getEmail();
                Uri photoUrl = profile.getPhotoUrl();

                //Display Information
                userName.setText(name);
                userProfName.setText(name);
                userEmail.setText("Email :"+email);
                userProfileImage.setImageURI(photoUrl);
            }
        }
    }


    private void showSignInOptions() {

        user = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();

        if (user!=null) {
            Toast.makeText(this,""+user.getEmail(),Toast.LENGTH_SHORT).show();
            displayUser();
            btn_sign_out.setEnabled(true);
        }else  {
            startActivityForResult(

AuthUI.getInstance().createSignInIntentBuilder().setAvailableProviders(providers).setTheme
(R.style.MyTheme).build(), MY_REQUEST_CODE
            );
        }
    }
```

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data){
        super.onActivityResult(requestCode,resultCode,data);
        if (requestCode==MY_REQUEST_CODE){
            IdpResponse response = IdpResponse.fromResultIntent(data);
            if (resultCode==RESULT_OK)
            {
                //get user
                FirebaseUser user =
com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
                //show email on Toast
                Toast.makeText(this,"Logged as
:"+user.getEmail(),Toast.LENGTH_SHORT).show();
                //Set Button Signout
                btn_sign_out.setEnabled(true);
                //finish();
                startActivity(getIntent());
            }
            else {
                Toast.makeText(this, ""+response.getError().getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

- **DetailActivity.java**

```
public class DetailActivity extends AppCompatActivity implements DetailView ,
OnMapReadyCallback {

    public static String EXTRA_DETAIL_SEARCHED;
    GoogleMap map;
    Button camera;

    @BindView(R.id.toolbar)
    Toolbar toolbar;

    @BindView(R.id.appbar)
    AppBarLayout appBarLayout;

    @BindView(R.id.collapsing_toolbar)
    CollapsingToolbarLayout collapsingToolbarLayout;

    @BindView(R.id.placeThumb)
    ImageView placeThumb;

    @BindView(R.id.location)
    TextView location;

    @BindView(R.id.country)
    TextView country;

    @BindView(R.id.instructions)
    TextView instructions;

    @BindView(R.id.characteristics)
    TextView characteristics;

    @BindView(R.id.measure)
    TextView measures;

    @BindView(R.id.progressBar)
    ProgressBar progressBar;

    @BindView(R.id.youtube)
    TextView youtube;

    @BindView(R.id.source)
    TextView source;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_detail);
        ButterKnife.bind(this);
        setupActionBar();


        Intent intent = getIntent();
        String placeName = intent.getStringExtra(EXTRA_DETAIL);//Get the Monument's
```

```
        Details from the users choice, on each monuments menu.
        String placeNameSearched= intent.getStringExtra(EXTRA_DETAIL_SEARCHED);//Get
the Monument's Details from the users search, on the search bar.

        DetailPresenter presenter = new DetailPresenter(this);
        presenter.getPlaceById(placeName);//Get Id from monuments menu choice.
        presenter.getPlaceById(placeNameSearched);//Get Id from user's search.

        //Monument's Map Location
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        //Take a Picture
        memoryCamera();
    }

    private void setupActionBar() {
        setSupportActionBar(toolbar);

collapsingToolbarLayout.setContentScrimColor(getResources().getColor(R.color.colorWhit
e));

collapsingToolbarLayout.setCollapsedTitleTextColor(getResources().getColor(R.color.col
orPrimary));

collapsingToolbarLayout.setExpandedTitleColor(getResources().getColor(R.color.colorWhi
te));
        if (getSupportActionBar() != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        }
    }

    void setupColorActionBarIcon(Drawable favoriteItemColor) {
        appBarLayout.addOnOffsetChangedListener((appBarLayout, verticalOffset) -> {
            if ((collapsingToolbarLayout.getHeight() + verticalOffset) < (2 *
ViewCompat.getMinimumHeight(collapsingToolbarLayout))) {
                if (toolbar.getNavigationIcon() != null)

toolbar.getNavigationIcon().setColorFilter(getResources().getColor(R.color.colorPrimar
y), PorterDuff.Mode.SRC_ATOP);

favoriteItemColor.mutate().setColorFilter(getResources().getColor(R.color.colorPrimary
),
                        PorterDuff.Mode.SRC_ATOP);

            } else {
                if (toolbar.getNavigationIcon() != null)

toolbar.getNavigationIcon().setColorFilter(getResources().getColor(R.color.colorWhite)
, PorterDuff.Mode.SRC_ATOP);

favoriteItemColor.mutate().setColorFilter(getResources().getColor(R.color.colorWhite),
                        PorterDuff.Mode.SRC_ATOP);
            }
        });
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_detail, menu);
    MenuItem favoriteItem = menu.findItem(R.id.favorite);
    Drawable favoriteItemColor = favoriteItem.getIcon();
    setupColorActionBarIcon(favoriteItemColor);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home :
            onBackPressed();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

@Override
public void showLoading() {
    progressBar.setVisibility(View.VISIBLE);
}

@Override
public void hideLoading() {
    progressBar.setVisibility(View.INVISIBLE);
}

@Override
public void setPlace(Places.Place place) {
    //get Information from Model (Places)
    Picasso.get().load(place.getStrPlaceThumb()).into(placeThumb);
    collapsingToolbarLayout.setTitle(place.getStrPlace());
    location.setText(place.getStrArea());
    country.setText(place.getStrLocation());
    instructions.setText(place.getStrHistory());
    setupActionBar();

    //===
    if (!place.getStrProperty1().isEmpty()) {
        characteristics.append("\n \u2022 " + place.getStrProperty1());
    }
    if (!place.getStrProperty2().isEmpty()) {
        characteristics.append("\n \u2022 " + place.getStrProperty2());
    }
    if (!place.getStrProperty3().isEmpty()) {
        characteristics.append("\n \u2022 " + place.getStrProperty3());
    }
    if (!place.getStrProperty4().isEmpty()) {
        characteristics.append("\n \u2022 " + place.getStrProperty4());
    }
    if (!place.getStrProperty5().isEmpty()) {
```

```
                characteristics.append("\n \u2022 " + place.getStrProperty5());
            }
            if (!place.getStrProperty6().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty6());
            }
            if (!place.getStrProperty7().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty7());
            }
            if (!place.getStrProperty8().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty8());
            }
            if (!place.getStrProperty9().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty9());
            }
            if (!place.getStrProperty10().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty10());
            }
            if (!place.getStrProperty11().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty11());
            }
            if (!place.getStrProperty12().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty12());
            }
            if (!place.getStrProperty13().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty13());
            }
            if (!place.getStrProperty14().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty14());
            }
            if (!place.getStrProperty15().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty15());
            }
            if (!place.getStrProperty16().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty16());
            }
            if (!place.getStrProperty17().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty17());
            }
            if (!place.getStrProperty18().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty18());
            }
            if (!place.getStrProperty19().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty19());
            }
            if (!place.getStrProperty20().isEmpty()) {
                characteristics.append("\n \u2022 " + place.getStrProperty20());
            }
            if (!place.getStrMeasure1().isEmpty() &&
    !Character.isWhitespace(place.getStrMeasure1().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure1());
            }
            if (!place.getStrMeasure2().isEmpty() &&
    !Character.isWhitespace(place.getStrMeasure2().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure2());
            }
            if (!place.getStrMeasure3().isEmpty() &&
```

```
!Character.isWhitespace(place.getStrMeasure3().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure3());
            }
            if (!place.getStrMeasure4().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure4().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure4());
            }
            if (!place.getStrMeasure5().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure5().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure5());
            }
            if (!place.getStrMeasure6().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure6().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure6());
            }
            if (!place.getStrMeasure7().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure7().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure7());
            }
            if (!place.getStrMeasure8().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure8().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure8());
            }
            if (!place.getStrMeasure9().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure9().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure9());
            }
            if (!place.getStrMeasure10().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure10().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure10());
            }
            if (!place.getStrMeasure11().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure11().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure11());
            }
            if (!place.getStrMeasure12().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure12().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure12());
            }
            if (!place.getStrMeasure13().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure13().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure13());
            }
            if (!place.getStrMeasure14().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure14().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure14());
            }
            if (!place.getStrMeasure15().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure15().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure15());
            }
            if (!place.getStrMeasure16().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure16().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure16());
            }
            if (!place.getStrMeasure17().isEmpty() &&
```

```
!Character.isWhitespace(place.getStrMeasure17().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure17());
            }
            if (!place.getStrMeasure18().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure18().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure18());
            }
            if (!place.getStrMeasure19().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure19().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure19());
            }
            if (!place.getStrMeasure20().isEmpty() &&
!Character.isWhitespace(place.getStrMeasure20().charAt(0))) {
                measures.append("\n : " + place.getStrMeasure20());
            }
            //===

            //Youtube Video Button Click.
            youtube.setOnClickListener(v -> {
                Intent intentYoutube = new Intent(Intent.ACTION_VIEW);
                intentYoutube.setData(Uri.parse(place.getStrYoutube()));
                startActivity(intentYoutube);
            });

            //Information Source Button Click.
            source.setOnClickListener(v -> {
                Intent intentSource = new Intent(Intent.ACTION_VIEW);
                intentSource.setData(Uri.parse(place.getStrSource()));
                startActivity(intentSource);
            });

            //Gets Monument Coordinates
            double lat= new Double(place.getStrProperty2());
            double longi = new Double(place.getStrProperty3());

            //Insert them into a map marker (pin).
            LatLng ThisMonument = new LatLng(lat,longi);
            map.addMarker(new
MarkerOptions().position(ThisMonument).title(place.getStrPlace()));
            map.moveCamera(CameraUpdateFactory.newLatLng(ThisMonument));

        }

        @Override
        public void onErrorLoading(String message) {
            Utils.showDialogMessage(this,"Error",message);
        }

        //Calling googleMap
        @Override
        public void onMapReady(GoogleMap googleMap){
            map=googleMap;
        }

        //Take a Picture Button
        public void memoryCamera(){
```

```
        camera = (Button)findViewById(R.id.camera);
        camera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try
                {
                    Intent intent = new Intent();
                    intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivity(intent);
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

- **DetailPresenter.java**

```
public class DetailPresenter {

    private DetailView view;

    public DetailPresenter(DetailView view){ this.view = view; }

    void getPlaceById(String placeName){

        view.showLoading();

        //Gets the Monument json data from the API
        Utils.getApi().getPlacesByName(placeName+".json").enqueue(new
Callback<Places>() {
            @Override
            public void onResponse(@NonNull Call<Places> call,@NonNull
Response<Places> response) {
                view.hideLoading();
                if (response.isSuccessful() && response.body() !=null){
                    view.setPlace(response.body().getPlaces().get(0));
                }else{
                    view.onErrorLoading(response.message());
                }
            }

            //Error handling
            @Override
            public void onFailure(@NonNull Call<Places> call,@NonNull Throwable t) {
                view.hideLoading();
                view.onErrorLoading(t.getLocalizedMessage());
            }
        });
    }
}
```

- **HomeActivity.java**

```
public class HomeActivity extends AppCompatActivity implements HomeView {

    public static final String EXTRA_LOCATION="location";
    public static final String EXTRA_POSITION="position";
    public static final String EXTRA_DETAIL = "detail";

    @BindView(R.id.viewPagerHeader) ViewPager viewPagerPlace;
    @BindView(R.id.recycleLocation) RecyclerView recyclerViewLocation;

    HomePresenter presenter;

    //Monument Search---
    EditText search_edit_text;
    RecyclerView recyclerView;
    DatabaseReference databaseReference;
    FirebaseUser firebaseUser;
    ArrayList<String> strPlaceList;
    ArrayList<String> strLocationList;
    ArrayList<String> strImageList;
    SearchAdapter searchAdapter;
    Context context;
    //---

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        ButterKnife.bind(this);

        presenter=new HomePresenter(this);

        //Monument Search---
        search_edit_text= (EditText) findViewById(R.id.searchView);
        recyclerView= (RecyclerView) findViewById(R.id.rv);

        databaseReference = FirebaseDatabase.getInstance().getReference();
        firebaseUser =
com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();

        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.addItemDecoration(new
DividerItemDecoration(this,LinearLayoutManager.VERTICAL));

        strPlaceList = new ArrayList<>();
        strLocationList = new ArrayList<>();
        strImageList = new ArrayList<>();
```

```
        search_edit_text.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after)
{
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }

            @Override
            public void afterTextChanged(Editable s) {
                if (!s.toString().isEmpty()){
                    setAdapter(s.toString());
                }else {
                    strLocationList.clear();
                    strPlaceList.clear();
                    strImageList.clear();
                    recyclerView.removeAllViews();
                }
            }
        });


        //---

        presenter.getPlaces();
        presenter.getLocations();

        navigationBar();


    }

    //Monument Search---
    public void setAdapter(final String searchString){

        databaseReference.child("Monuments").addListenerForSingleValueEvent(new
ValueEventListener() {

            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                strLocationList.clear();
                strPlaceList.clear();
                strImageList.clear();
                recyclerView.removeAllViews();
```

```
                int counter = 0;
                //Takes a snapshot of the database each time the user types a letter
                for (DataSnapshot snapshot:dataSnapshot.getChildren()){
                    String monumentid=snapshot.getKey();
                    String
strPlace=snapshot.child("/places/0/strPlace").getValue(String.class);
                    String
strLocation=snapshot.child("/places/0/strLocation").getValue(String.class);
                    String
strImage=snapshot.child("/places/0/strPlaceThumb").getValue(String.class);

                    if (strPlace.toLowerCase().contains(searchString.toLowerCase())){
                        strPlaceList.add(strPlace);
                        strLocationList.add(strLocation);
                        strImageList.add(strImage);
                        counter++;
                    }
                    else if
(strLocation.toLowerCase().contains(searchString.toLowerCase())){
                            strPlaceList.add(strPlace);
                            strLocationList.add(strLocation);
                            strImageList.add(strImage);
                            counter++;
                        }
                    if (counter == 15){
                        break;
                    }
                }
                searchAdapter = new
SearchAdapter(HomeActivity.this,strPlaceList,strLocationList,strImageList);
                recyclerView.setAdapter(searchAdapter);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });
    }
    //---

    public void navigationBar(){

        //Navigation barView
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);

        //Set Home Selected
        bottomNavigationView.setSelectedItemId(R.id.home);

        //Perform Item SelectListener
```

```
        bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()){
                    case R.id.profile:
                        startActivity(new Intent(getApplicationContext(),
FirebaseAuth.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.home:
                        startActivity(new Intent(getApplicationContext(),
HomeActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.memories:
                        startActivity(new Intent(getApplicationContext(),
PostsActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                }
                return false;
            }
        });
    }

    @Override
    public void showLoading() {
    findViewById(R.id.shimmerPlace).setVisibility(View.VISIBLE);
    findViewById(R.id.shimmerLocation).setVisibility(View.VISIBLE);
    }

    @Override
    public void hideLoading() {
        findViewById(R.id.shimmerPlace).setVisibility(View.GONE);
        findViewById(R.id.shimmerLocation).setVisibility(View.GONE);
    }

    @Override
    public void setPlace(List<Places.Place> place) {
        //Most Famous Monuments
        ViewPagerHeaderAdapter headerAdapter = new ViewPagerHeaderAdapter(place,this);
        viewPagerPlace.setAdapter(headerAdapter);
        viewPagerPlace.setPadding(20,0,150,0);
        headerAdapter.notifyDataSetChanged();

        headerAdapter.setOnItemClickListener((view, position) -> {

            TextView placeName = view.findViewById(R.id.placeName);
```

```
            Intent intent = new Intent(getApplicationContext(), DetailActivity.class);
            intent.putExtra(EXTRA_DETAIL, placeName.getText().toString());
            startActivity(intent);
        });
    }

    @Override
    public void setLocation(List<Locations.Location> location) {
        //Countries Menu
        RecyclerViewHomeAdapter homeAdapter = new RecyclerViewHomeAdapter(location,this);
        recyclerViewLocation.setAdapter(homeAdapter);
        GridLayoutManager layoutManager = new GridLayoutManager(this,3,
GridLayoutManager.VERTICAL, false);
        recyclerViewLocation.setLayoutManager(layoutManager);
        recyclerViewLocation.setNestedScrollingEnabled(true);
        homeAdapter.notifyDataSetChanged();

        homeAdapter.setOnItemClickListener((view, position) -> {
            Intent intent = new Intent(this, LocationActivity.class);
            intent.putExtra(EXTRA_LOCATION,(Serializable) location);
            intent.putExtra(EXTRA_POSITION,position);
            startActivity(intent);
        });
    }

    @Override
    public void onErrorLoading(String message) {
        Utils.showDialogMessage(this,"Title",message);
    }
}
```

- **HomePresenter.java**

```
class HomePresenter {

    private HomeView view;

    public HomePresenter(HomeView view) {
        this.view = view;
    }

    void getPlaces() {
        view.showLoading();
        Call<Places> placesCall = Utils.getApi().getPlace();
        //Gets the Monuments form the API to the Most Famous Monuments List
        placesCall.enqueue(new Callback<Places>() {
            @Override
            public void onResponse( @NonNull Call<Places> call, @NonNull Response<Places>
response) {
                view.hideLoading();

                if (response.isSuccessful() && response.body() != null) {
                    view.setPlace(response.body().getPlaces());
                }
                else {
                    view.onErrorLoading(response.message());
                }
            }
            //Error Handling
            @Override
            public void onFailure(Call<Places> call, Throwable t) {
                view.hideLoading();
                view.onErrorLoading(t.getLocalizedMessage());
            }
        });
    }

    void getLocations() {
        view.showLoading();
        //Gets the Countries from the API to the Countries Menu
        Call<Locations> locationsCall = Utils.getApi().getLocations();
        locationsCall.enqueue(new Callback<Locations>() {
            @Override
            public void onResponse(@NonNull Call<Locations> call,@NonNull
Response<Locations> response) {

                view.hideLoading();
                if (response.isSuccessful() && response.body() != null) {
                    view.setLocation(response.body().getLocations());
                }
                else {
                    view.onErrorLoading(response.message());
                }
            }
            //Error Handling
            @Override
            public void onFailure(@NonNull Call<Locations> call,@NonNull Throwable t) {
```

```
                view.hideLoading();
                view.onErrorLoading(t.getLocalizedMessage());
            }
        });
    }
}
```

- **LocationActivity.java**

```
public class LocationActivity extends AppCompatActivity {

    @BindView(R.id.toolbar)
    Toolbar toolbar;
    @BindView(R.id.tabLayout)
    TabLayout tabLayout;
    @BindView(R.id.viewPager)
    ViewPager viewPager;

    @Override
    protected void  onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_location);
        ButterKnife.bind(this);

        initActionBar();
        initIntent();
    }

    private void initIntent() {
        Intent intent = getIntent();
        List<Locations.Location> locations= (List<Locations.Location>)
intent.getSerializableExtra(HomeActivity.EXTRA_LOCATION);

        int position = intent.getIntExtra(HomeActivity.EXTRA_POSITION,0);

        ViewPagerLocationAdapter adapter= new
ViewPagerLocationAdapter(getSupportFragmentManager(), locations);
        viewPager.setAdapter(adapter);
        tabLayout.setupWithViewPager(viewPager);
        viewPager.setCurrentItem(position,true);
        adapter.notifyDataSetChanged();
    }

    private void initActionBar() {
        setSupportActionBar(toolbar);
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        }
    }


    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case android.R.id.home:
                onBackPressed();
                break;
        }
        return true;
    }
}
```

- **LocationFragment.java**

```
public class LocationFragment extends Fragment implements LocationView {

    @BindView(R.id.recycleView)
    RecyclerView recyclerView;
    @BindView(R.id.progressBar)
    ProgressBar progressBar;
    @BindView(R.id.imageLocation)
    ImageView imageLocation;
    @BindView(R.id.imageLocationBg)
    ImageView imageLocationBg;
    @BindView(R.id.textLocation)
    TextView textLocation;

    AlertDialog.Builder descDialog;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_location, container, false);
        ButterKnife.bind(this, view);
        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        //Creates Countries Info
        if(getArguments() !=null){
            textLocation.setText(getArguments().getString("EXTRA_DATA_DESC"));

Picasso.get().load(getArguments().getString("EXTRA_DATA_IMAGE")).into(imageLocation);

Picasso.get().load(getArguments().getString("EXTRA_DATA_IMAGE")).into(imageLocationBg);

            descDialog = new
AlertDialog.Builder(getActivity()).setTitle(getArguments().getString("EXTRA_DATA_NAME")).s
etMessage(getArguments().getString("EXTRA_DATA_DESC"));

            LocationPresenter presenter = new LocationPresenter(this);
            presenter.getPlaceByLocation(getArguments().getString("EXTRA_DATA_NAME"));
        }
    }

    @Override
    public void showLoading() {
        progressBar.setVisibility(View.VISIBLE);
    }

    @Override
    public void hideLoading() {
        progressBar.setVisibility(View.GONE);
    }

    @Override
```

```
    public void setPlaces(List<Places.Place> places) {
        RecyclerViewPlaceByLocation adapter = new
RecyclerViewPlaceByLocation(getActivity(),places);
        recyclerView.setLayoutManager(new GridLayoutManager(getActivity(),2));
        recyclerView.setClipToPadding(false);
        recyclerView.setAdapter(adapter);
        adapter.notifyDataSetChanged();

        adapter.setOnItemClickListener((view, position) -> {
            TextView placeName = view.findViewById(R.id.placeName);
            Intent intent = new Intent(getActivity(), DetailActivity.class);
            intent.putExtra(EXTRA_DETAIL, placeName.getText().toString());
            startActivity(intent);
        });
    }

    @Override
    public void onErrorLoading(String message) {
        Utils.showDialogMessage(getActivity(), "Error ", message);
    }

    @OnClick(R.id.cardLocation)
    public void onClick(){
        descDialog.setPositiveButton("CLOSE", ((dialog, which) -> dialog.dismiss()));
        descDialog.show();
    }
}
```

- **LocationPresenter.java**

```
public class LocationPresenter {

    private LocationView view;
    public LocationPresenter(LocationView view) {
        this.view = view;
    }

    void getPlaceByLocation(String location) {
        //Gets the Monuments List based on the selected Country
        view.showLoading();
        Call<Places> placesCall = Utils.getApi().getPlaceByLocation(location+".json");
        placesCall.enqueue(new Callback<Places>() {
            @Override
            public void onResponse(@NonNull Call<Places> call,@NonNull Response<Places>
response) {
                view.hideLoading();
                if (response.isSuccessful() && response.body() != null) {
                    view.setPlaces(response.body().getPlaces());
                } else {
                    view.onErrorLoading(response.message());
                }
            }

            //Error Handling
            @Override
            public void onFailure(@NonNull Call<Places> call,@NonNull Throwable t) {
                view.hideLoading();
                view.onErrorLoading(t.getLocalizedMessage());
            }
        });
    }
}
```

- **DetailPostActivity.java**

```
public class DetailPostActivity extends AppCompatActivity {

    TextView
nameDetailTextView,descriptionDetailTextView,dateDetailTextView,categoryDetailTextView;
    ImageView postDetailImageView;

    private void initializeWidgets(){
        nameDetailTextView = findViewById(R.id.nameDetailTextView);
        descriptionDetailTextView = findViewById(R.id.descriptionDetailTextView);
        categoryDetailTextView = findViewById(R.id.categoryDetailTextView);
        postDetailImageView = findViewById(R.id.memoryDetailImageView);
        dateDetailTextView = findViewById(R.id.dateDetailTextView);
    }

    private String getDateToday(){
        DateFormat dateFormat=new SimpleDateFormat("yyyy/mm/dd");
        Date date = new Date();
        String today = dateFormat.format(date);
        return today;
    }

    private String getRandomCategory(){
        String[] categories={"What a View!","5 star Visit!","Now that's a monument!!"};
        Random random = new Random();
        int index = random.nextInt(categories.length-1);
        return categories[index];
    }

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_posts_details);

        initializeWidgets();

        //Receive data from itemsactivity from API
        Intent i = this.getIntent();
        String name = i.getExtras().getString("NAME_KEY");
        String description =i.getExtras().getString("DESCRIPTION_KEY");
        String imageURL = i.getExtras().getString("IMAGE_KEY");

        //set received data to textviews and image views
        nameDetailTextView.setText(name);
        descriptionDetailTextView.setText(description);
        dateDetailTextView.setText("DATE:"+ getDateToday());
        categoryDetailTextView.setText("CATEGORY: "+ getRandomCategory());

Picasso.get().load(imageURL).placeholder(R.drawable.placeholder).fit().centerCrop().into(p
ostDetailImageView);
    }
}
```

- **ItemsActivity.java**

```
public class ItemsActivity extends AppCompatActivity implements
PostsRecyclerAdapter.OnItemClickListener {

    private RecyclerView mRecyclerView;
    private PostsRecyclerAdapter mAdapter;
    private ProgressBar mProgressBar;
    private FirebaseStorage mStorage;
    private DatabaseReference mDatabaseRef;
    private ValueEventListener mDBListener;
    private List<Posts> mPosts;
    private FirebaseUser mUser;
    private StorageReference mStorageReff;


    private void openDetailActivity(String[] data){
        Intent intent = new Intent(this,DetailPostActivity.class);
        intent.putExtra("NAME_KEY",data[0]);
        intent.putExtra("DESCRIPTION_KEY",data[1]);
        intent.putExtra("IMAGE_KEY",data[2]);
        startActivity(intent);

    }
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_posts_items);

        mRecyclerView = findViewById(R.id.mRecyclerView);
        mRecyclerView.setHasFixedSize(true);
        mRecyclerView.setLayoutManager(new LinearLayoutManager(this));

        mProgressBar = findViewById(R.id.DataLoaderProgressBar);
        mProgressBar.setVisibility(View.VISIBLE);

        mPosts = new ArrayList<>();
        mAdapter = new PostsRecyclerAdapter(ItemsActivity.this,mPosts);
        mRecyclerView.setAdapter(mAdapter);
        mAdapter.setOnItemClickListener(ItemsActivity.this);

        mUser = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
        mStorage = FirebaseStorage.getInstance();
        mDatabaseRef =
FirebaseDatabase.getInstance().getReference("memories_uploads/"+mUser.getUid().toString())
;

        mDBListener = mDatabaseRef.addValueEventListener(new ValueEventListener() {
            //Gets all the posts of the logged user
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                mPosts.clear();

                for (DataSnapshot postSnapshot : dataSnapshot.getChildren()){
                    Posts upload = postSnapshot.getValue(Posts.class);
                    upload.setKey(postSnapshot.getKey());
```

```
                    mPosts.add(upload);
                }
                mAdapter.notifyDataSetChanged();
                mProgressBar.setVisibility(View.GONE);
            }

            //Error Handling
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(ItemsActivity.this,databaseError.getMessage(),Toast.LENGTH_SHORT).show();
                mProgressBar.setVisibility(View.INVISIBLE);
            }
        });
    }

    //opens DetailActivity on click
    public void onItemClick (int position){
        Posts clickedPost = mPosts.get(position);
        String[] postData =
{clickedPost.getName(),clickedPost.getDescription(),clickedPost.getImageURL()};
        openDetailActivity(postData);
    }

    @Override
    public void onShowItemClick (int position){
        Posts clickedPost = mPosts.get(position);
        String[] postData =
{clickedPost.getName(),clickedPost.getDescription(),clickedPost.getImageURL()};
        openDetailActivity(postData);
    }

    //Deletes a Post
    @Override
    public void onDeleteItemClick(int position){
        Posts selectedItem = mPosts.get(position);
        final String selectedKey = selectedItem.getKey();

        StorageReference imageRef =
mStorage.getReferenceFromUrl(mUser.getUid()+selectedItem.getImageURL());
        imageRef.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                mDatabaseRef.child(selectedKey).removeValue();
                Toast.makeText(ItemsActivity.this,"Item
deleted",Toast.LENGTH_SHORT).show();
            }
        });
    }

    protected void onDestroy(){
        super.onDestroy();
        mDatabaseRef.removeEventListener(mDBListener);
    }
}
```

- **PostsActivity.java**

```
public class PostsActivity extends AppCompatActivity {
    private Button openPostsActivityBtn,openUploadActivityBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_posts);

        openPostsActivityBtn = findViewById(R.id.openMemoriesActivityBtn);
        openUploadActivityBtn = findViewById(R.id.openUploadActivityBtn);

        //View Memories Button
        openPostsActivityBtn.setOnClickListener(new View.OnClickListener(){
            @Override
            public  void onClick(View view){
                Intent i= new Intent(PostsActivity.this, ItemsActivity.class);
                startActivity(i);
            }
        });

        //Create Memories Button
        openUploadActivityBtn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                Intent i=new Intent(PostsActivity.this,UploadActivity.class);
                startActivity(i);
            }
        });

        navigationBar();
    }

    public void navigationBar(){
        //Navigation barView
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
        //Set Home Selected
        bottomNavigationView.setSelectedItemId(R.id.memories);
        //Perform Item SelectListener
        bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()){
                    case R.id.profile:
                        startActivity(new Intent(getApplicationContext(),
FirebaseAuth.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.home:
                        startActivity(new Intent(getApplicationContext(),
HomeActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                    case R.id.memories:
```

```
                        startActivity(new Intent(getApplicationContext(),
PostsActivity.class));
                        overridePendingTransition(0,0);
                        return true;
                    }
                    return false;
                }
            });
        }
}
```

- **UploadActivity.java**

```
public class UploadActivity extends AppCompatActivity {

    private static final int PICK_IMAGE_REQUEST = 1;
    private Button choosenImageBtn;
    private Button uploadBtn;
    private EditText nameEditText;
    private EditText descriptionEditText;
    private ImageView chosenImageView;
    private ProgressBar uploadProgressBar;

    private FirebaseUser mUser;
    private Uri mImageUri;
    private StorageReference mStorageRef;
    private DatabaseReference mDatabaseRef;
    private StorageTask mUploadTask;

    @Override
    protected void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_posts_up);

        choosenImageBtn = findViewById(R.id.button_choose_image);
        uploadBtn = findViewById(R.id.uploadBtn);
        nameEditText = findViewById(R.id.nameEditText);
        descriptionEditText = findViewById(R.id.descriptionEditText);
        chosenImageView = findViewById(R.id.chosenImageView);
        uploadProgressBar = findViewById(R.id.progress_bar);

        mUser = com.google.firebase.auth.FirebaseAuth.getInstance().getCurrentUser();
        mStorageRef = FirebaseStorage.getInstance().getReference("memories_uploads");
        mDatabaseRef =
FirebaseDatabase.getInstance().getReference("memories_uploads/"+mUser.getUid().toString())
;

        //Image Button
        choosenImageBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openFileChooser();
            }
        });

        //Upload Button
        uploadBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mUploadTask != null && mUploadTask.isInProgress()){
                    Toast.makeText(UploadActivity.this,"An Upload is Still in
Progress",Toast.LENGTH_SHORT).show();
                }else {
                    uploadFile();
                }
            }
```

```
        });
    }

    //Image choice function
    private void openFileChooser(){
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent,PICK_IMAGE_REQUEST);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data){
        super.onActivityResult(requestCode,resultCode,data);

        if (requestCode== PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data!=null &&
data.getData() != null){
            mImageUri = data.getData();

            Picasso.get().load(mImageUri).into(chosenImageView);
        }
    }

    private String getFileExtention(Uri uri){
        ContentResolver cR= getContentResolver();
        MimeTypeMap mime = MimeTypeMap.getSingleton();
        return mime.getExtensionFromMimeType(cR.getType(uri));
    }

    //Uploads a Memory to Firebase
    private void uploadFile(){
        if (mImageUri != null){

            //It creates separate directories for each user in the database
            final StorageReference imageRef =
mStorageRef.child(mUser.getUid()+"/"+mImageUri.getLastPathSegment());
            uploadProgressBar.setVisibility(View.VISIBLE);
            uploadProgressBar.setIndeterminate(true);

            mUploadTask = imageRef.putFile(mImageUri).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                    Handler handler = new Handler();
                    handler.postDelayed(new Runnable() {
                        @Override
                        public void run() {
                            uploadProgressBar.setVisibility(View.VISIBLE);
                            uploadProgressBar.setIndeterminate(false);
                            uploadProgressBar.setProgress(0);

                        }
                    },500);

                    Toast.makeText(UploadActivity.this,"Memory Upload
successful",Toast.LENGTH_LONG).show();
```

```
                    imageRef.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
                        @Override
                        public void onSuccess(Uri downloadurl) {
                            Posts upload = new
Posts(nameEditText.getText().toString().trim(),downloadurl.toString(),descriptionEditText.
getText().toString());
                            String uploadId = mDatabaseRef.push().getKey();
                            mDatabaseRef.child(uploadId).setValue(upload);
                        }
                    });

                    uploadProgressBar.setVisibility(View.VISIBLE);
                    openImagesActivity();
                }
            })
             .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                   uploadProgressBar.setVisibility(View.VISIBLE);

Toast.makeText(UploadActivity.this,e.getMessage(),Toast.LENGTH_SHORT).show();

                }
             })
               .addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
                   @Override
                   public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
                       double progress = (100.0 * taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                       uploadProgressBar.setProgress((int) progress);
                   }
               });
        }else{
            Toast.makeText(this,"You haven't Selected any
file!",Toast.LENGTH_SHORT).show();
        }
    }
    private void openImagesActivity(){
        Intent intent = new Intent(this,PostsActivity.class);
        startActivity(intent);
    }
}
```

Appendix C: Requirements Gathering Survey:

REQUIREMENTS GATHERING SURVEY

This is a survey about a mobile Android application, that shows the most famous and beautiful monuments and historical landmarks in several countries around the world, to help travellers decide which places they wish to know more about, or even travel to. To help us provide benefits that meet your needs in such app, please complete this survey and return it to Felix Saraiva.

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
App Focus					
The app should focus on historians.					
The app should focus on travellers.					
The app should focus on amaze everyone.					
The app should help people share with others the places they physical visited.					
The help should have young monuments/places of interest.					
Design					
Should the app be divided in sections?					
Should the app use flags to identify countries?					
Are shapes the best option to identify continents?					
Colours must be a concern in this app layout.					
Information					
The app should have a ‘monuments per country’ limit number.					
Should a Monument display its own history?					
Should a country display its own history?					
Should, a monument, have a video presentation?					
Should, a monument, display its location via Map?					
Should a landmark display its own characteristics?					
Interactions					
Should the user be able to rate and review a Monument?					
Should the user be able to see the last viewed monuments in the app?					
Should the user be able to store their favourite monuments in a favourites section?					
Should the user be able to display in their profile the monuments they visited physically.					

Should the user be able to take a picture of the monument they are visiting in real-time?					
Should other users be able to see each other's profile and comment/ like the pictures of other users?					
Overall					
The app should be available for both android and IOS.					
The app should have a web version for computer users.					
Should it have a Login feature?					
Should it have a User Profile feature?					
Additional Comments: (Please use this section to justify some of your answers)					

Appendix D: Interview:

INTERVIEW WITH A CYBER-SECURITY CONSULTANT FROM ERNST & YOUNG (EY)

The Interview started with an explanation of the system in study and the main idea around it.

Interview Script:

Q: How should the application be organized?

A: In the mobile applications market, the developing and design teams always try to work together to make the user's life more comfortable regarding the interaction with the respective system. Therefore, comparing this idea with the current market, I think it should be better if this system worked in layers of different menus, this way the user could go back and forward in the app without losing track of what he wants to do, and the developers can have better control over the user's decision.

Q: What information should the App provide?

A: The way I see it, I would like to use such app when I am, for example, visiting the respective monument, because sometimes, when we go on our travels, and we are doing tourism, we do not really know the history behind that monument, why it was built, when and how. I think it would be nice to have some history displayed and some interesting characteristics. However, if you are for example planning a trip and you wish to know more about a particular monument then I think it would be nice to see a video where you can look at the place and ear a bit about it, plus a lot of people do not like to read.

Q: What extra features and future improvements would you like to see in this system?

A: We cannot lose the track that this application is directly related with travelling and the most crucial thing about travels are memories, because of that I think people should be able to save and store their experiences with those landmarks. Maybe in the future, you will build a place to store those visits has trophies may be in a user profile where people can share the places they have visited and have a profile of photos of those places. Also, help people planning their future travels with a favourites list of what monuments they would like to see in the future.

Q: What interaction should a monument provide to the user?

A: It would be nice to review a monument if you have been there, like 1-5 stars and comment your opinion about it. Also, a map feature would be perfect if you can find a way to put a Google Maps link to the location of the monument, we talked about taking photos and store them and the YouTube video I think all of these features would make a robust application, one that I would download.

Q: How big should be the first Prototype's database?

A: You do not need to have a fully functional system with all the features that we discussed on your first release, if you can show the main idea with some interactions and functional design, I think any client would be happy, well…most of them if we can be positive. My suggestion is: pick up a continent, maybe Europe because it has many Monuments, then choose maybe…12 countries and 12 monuments per country and start with that. This way, you are being ambitious and showing that it is possible to build this system. Also pick up the essential features that you think this system needs and then organize the other features by importance, like in a sprint backlog and that is how you build your first MVP.

Appendix E: Risk Management Process:

Risk Management:

RISK TYPES	STRATEGY
Requirements	Increments should be used to welcome requirements changes and to plan these changes in advance, in order to preserve the projects agility.
Algorithm	The algorithm used for the application should be cost optimal as well as space optimal. The application should not occupy a lot of memory in the system. (Kakkar, 2013)
Time	A project plan should be assembled and consulted along the process of development. Each Sprint should not have more than a 3 weeks development window.
Platform	The platform on which the application is going to run should be feasible in nature and should be one that is relatively stable and secure. (Kakkar, 2013)
User Interface	The application GUI should be user friendly along with certain enhanced features which make it stand apart from the rest of the applications already available in the market. (Kakkar, 2013)

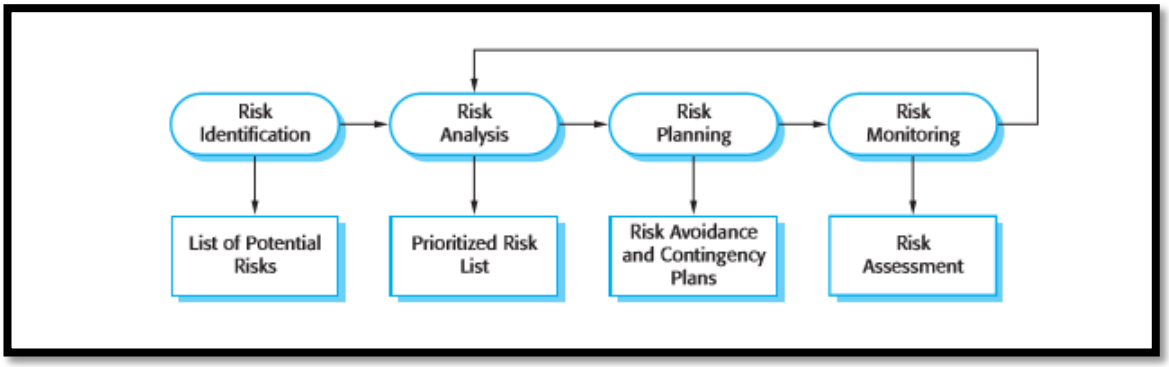


Figure 32 The Risk Management Process (Sommerville, Software Engineering, 2011)

Appendix F: Architectural Design:

Architectural Design Decisions

During the architectural design process, system architects have to make a number of structural decisions that profoundly affect the system and its development process. Based on their knowledge and experience, they have to consider the following fundamental questions about the system:

1. Is there a generic application architecture that can act as a template for the system that is being designed?
2. How will the system be distributed across a number of cores or processors?
3. What architectural patterns or styles might be used?
4. What will be the fundamental approach used to structure the system?
5. How will the structural components in the system be decomposed into subcomponents?
6. What strategy will be used to control the operation of the components in the system?
7. What architectural organization is best for delivering the non-functional requirements of the system?
8. How will the architectural design be evaluated?
9. How should the architecture of the system be documented?

Although each software system is unique, systems in the same application domain often have similar architectures that reflect the fundamental concepts of the domain. (Sommerville, Software Engineering, 2011)

Architectural views

It is impossible to represent all relevant information about a system's architecture in a single architectural model, as each model only shows one view or perspective of the system. It might show how a system is decomposed into modules, how the run-time processes interact, or the different ways in which system components are distributed across a network. All of these are useful at different times so, for both design and documentation, you usually need to present multiple views of the software architecture.

- ⇒ What views or perspectives are useful when designing and documenting a system's architecture?
- ⇒ What notations should be used for describing architectural models? (Sommerville, Software Engineering, 2011)

Appendix G: Testing Techniques:

Testing Process Goals

Validation Testing:

- To demonstrate to the developer and the system customer that the software meets its requirements;
- A successful test shows that the system operates as intended.

Defect Testing:

- To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification;

- A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

The Verification & Validation Process

Verification:

- In the software lifecycle, verification verifies if the product conforms itself to its specifications. It is an answer to the question “Are we building the product in the right way?”.

Validation:

- The validation process looks at the user’s point of view, it is used to be certain that the product is what the user really requires. It answers to the question “Are we building the right product?”.

The verification and validation processes must be applied at each stage in the software life-cycle process. Both processes work in harmony with the objective of discover defects in a system and in the assessment of whether or not the system is useful and useable in an operational situation. The confidence of this processes depends on:

⇒ **Software Function:** The level of confidence depends on how critical the software is to an organisation.

⇒ **User expectations:** Users may have low expectations of certain kinds of software.

⇒ **Marketing environment:** Getting a product to market early may be more important than finding defects in the program. (Sommerville, Verification and Validation, 2006)

System Testing

Involves integrating components to create a system or sub-system. May involve testing an increment to be delivered to the customer and it is composed by two phases.

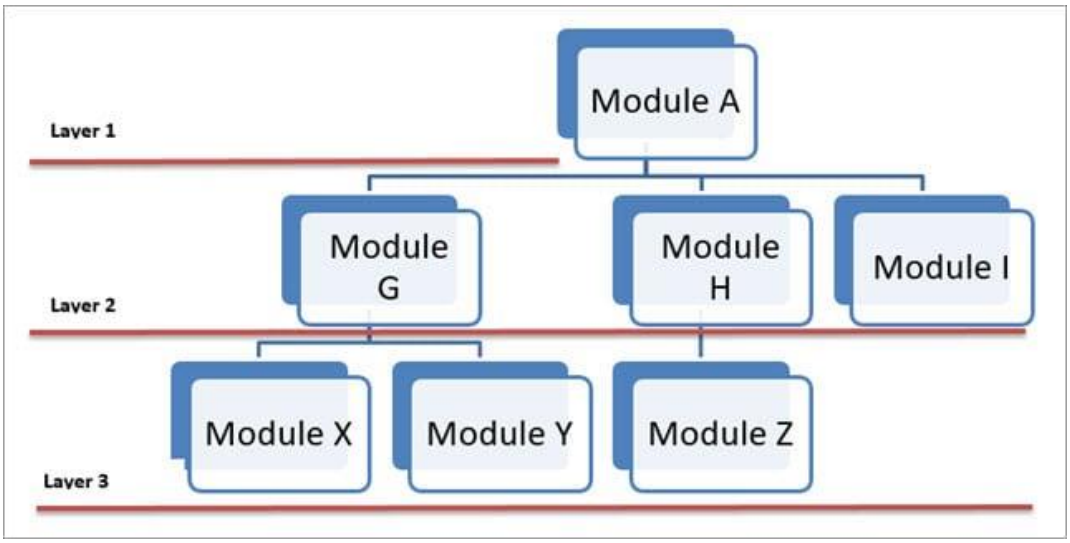
Phases:

- **Integration testing**- the test team have access to the system source code. The system is tested as components are integrates.
- **Release Testing**- the test team test the complete system to be delivered as a black box. (Sommerville, Software Testing, 2006)

Incremental Integration Testing

Incremental Testing comes under the blanket of Integration testing. In this approach of testing, integration testing is done on the individual module as a part of unit testing and in the next phase, modules or components of the application are integrated incrementally and testing is performed on combined modules as a group.

Let's see an example using Sandwich Testing:



Following test cases can be derived with Sandwich Testing Strategy:

- ⇒ **Test Case1:** Test A, X, Y, and Z individually – where Test A comes under Top layer test and Test X, Y and Z comes under Bottom layer tests
- ⇒ **Test Case2:** Test A, G, H and I
- ⇒ **Test Case3:** Test G, X, and Y
- ⇒ **Test Case4:** Test Hand Z
- ⇒ **Test Case5:** Test A, G, H, I, X, Y, and Z

(HELP, 2019)

Integration Testing Approaches

Big-bang Approach:

⇒ Most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

Bottom-up Integration:

⇒ Is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top-down Integration:

➢ Top-down testing is an approach to integrated testing where the top integrated modules are tested, and the branch of the module is tested step by step until the end of the related module. Basically, it develops the skeleton of the system and populates it with components.

Sandwich Integration:

➢ Sandwich testing is an approach to combine top down testing with bottom up testing. (Wikipedia, 2019)

Black Box Testing

Advantages & Disadvantages

Advantages:

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.
- Tester need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
- Test cases can be designed as soon as the specifications are complete.

Disadvantages:

- Only a small number of possible inputs can be tested, and many program paths will be left untested.
- Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- Tests can be redundant if the software designer/developer has already run a test case.
- Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing. (Fundamentals, Black Box Testing, 2019)

White Box Testing

Advantages & Disadvantages

Advantages:

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

Disadvantages:

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available. (Fundamentals, White Box Testing, 2019)

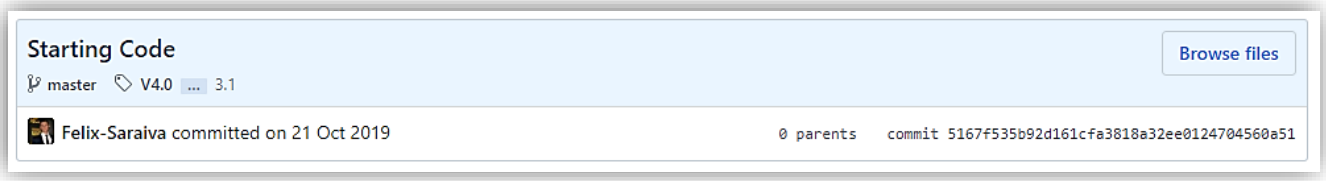
Path Testing

Advantages of Basic Path Testing:

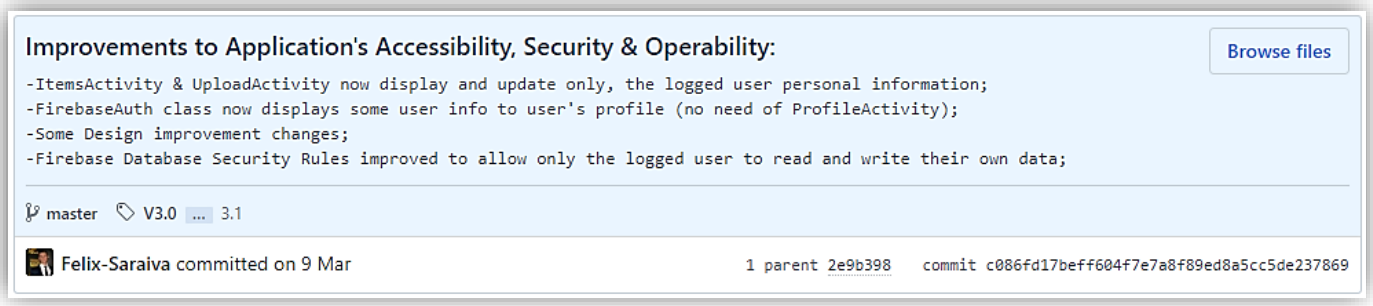
- It helps to reduce the redundant tests
- It focuses attention on program logic
- It helps facilitates analytical versus arbitrary case design
- Test cases which exercise basis set will execute every statement in a program at least once

Appendix H: Additional GitHub Commits:

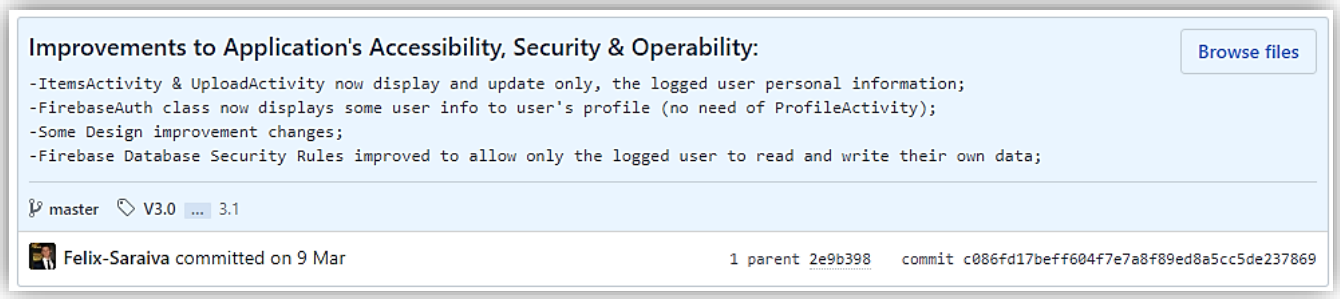
⇒ This was the first Commit of this project's development, containing just the creation of some important files and some core basic functionality;



⇒ During the development of the application the developer decided to update the xml layout files to work with a more recent version called AndroidX. And implement a bottom navigation bar to facilitate switching between Views;



⇒ This commit demonstrates a report of when the developer introduced restrict back end security rules.



Appendix J: Cyclomatic Complexity:

The number of tests to test all control statements equals the cyclomatic complexity.

- Complexity = Number of edges – Number of nodes + 1

Complexity Number	Meaning
1-10	⇒ Structured and well written code; ⇒ High Testability; ⇒ Cost and Effort is less.
10-20	⇒ Complex Code; ⇒ Medium Testability; ⇒ Cost and effort are Medium.
20-40	⇒ Very complex Code; ⇒ Low Testability; ⇒ Cost and Effort are high.
>40	⇒ Not at all testable; ⇒ Very high Cost and Effort.

Appendix K: Ethics Checklist:

Ethics Checklist

A checklist should be completed for every research project. This is used to identify whether a full application for ethics approval needs to be submitted to the University Ethics Panel or one of its sub-committees. Further guidance can be found on the Ethics Blackboard shell.

1 Applicant details	
Name of Lead Researcher (applicant):	Felix Alexandre Monteiro Saraiva

2 Project details
Project title: Monument Android Application
Please provide a brief description of the project: The research, design and development of a mobile Android application, that shows the famous and most beautiful monuments and places in many main cities around the world, to help travellers decide which places they want to visit.

3 Research checklist Please answer each question by checking the appropriate box:			
Research that may need to be reviewed by an NHS Research Ethics Committee or another external Ethics Committee		YES	NO
1	Will the study involve recruitment of patients or staff through the NHS or Social Care, or the use of NHS data or premises and/or equipment?	☐	☒
2	Does the study involve participants age 16 or over who are unable to give informed consent (e.g. people with learning disabilities: see Mental Capacity Act 2005)? NHS	☐	☒
3	Will tissue samples (including blood) be obtained from participants?	☐	☒
If you have answered 'Yes' to questions 1, 2 or 3 please refer to http://www.hra.nhs.uk/ for guidance. If external ethical approval is not needed, University ethical approval will still be required.			
Research participants		YES	NO
4	Does the study involve students within the University?	☐	☒
5	Does the study involve employees of the University?	☐	☒

6	Does the research involve potentially vulnerable groups: children, those with cognitive impairment, or those in unequal relationships? (eg your own students)	☐	☒
7	Does the research involve members of the public or people worked with in a professional capacity?	☐	☒
8	Will the study require the co-operation of a 'gatekeeper' for initial access to the groups or individuals to be recruited and/or to give permission for initial contact? (e.g. children, students, members of self-help group, residents of nursing home, employees).	☐	☒
Research methods		YES	NO
9	Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g. covert observation of people in non-public places)	☐	☒
10	Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	☐	☒
11	Will the study involve discussion of sensitive topics or illegal activity (e.g. sexual activity, drug use)?	☐	☒
12	Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	☐	☒
13	Is physical pain or more than mild discomfort likely to result from the study?	☐	☒
14	Could the study induce psychological stress or anxiety or cause harm or negative consequences beyond the risks encountered in normal life?	☐	☒
15	Will the study involve prolonged or repetitive testing?	☒	☐
16	Is there a possibility that the safety of the researcher may be in question?	☐	☒
17	Will any of the research take place outside the UK (excluding on-line surveys)?	☒	☐
Data and confidentiality			
18	Will the research involve administrative or secure data that requires permission from the appropriate authorities before use?	☐	☒
19	Will the research involve visual/vocal methods where respondents may be identified?	☐	☒
20	Will research involve the sharing of data or confidential information beyond the initial consent given?	☐	☒

21	Will the research involve security-sensitive data? (eg commissioned by the military or under an EU security call; involve the acquisition of security clearances; concerns terrorist or extremist groups).	☐	☒

If any item is checked "YES" you will need to seek advice from your supervisor / course leader regarding the appropriate sub-committee for ethical approval.

4. Declarations

I have read and will abide by the University's *Ethics Policy*.

I have read and will abide by the University's *Code Research Practice*.

I am aware of, and will abide by the ethical guidelines published by the relevant subject and/or professional associations most appropriate to my topic.

The responses given above are an accurate and true reflection of the nature of my research project.

Applicant:

Name (please print): Felix Alexandre Monteiro Saraiva
Signed: Felix Saraiva
Date: 05/11/2019

Project supervisor / Line manager

I confirm that the above details are accurate, the proposed methods are appropriate, ethical concerns have been considered and that time and resources are available for the research to take place.

Name (please print):
Signed:
Date:

Note: Electronic approval by above signatories is acceptable