2021

IS52018C/S: Software Projects

SPLITSY, BILL SPLIT APPLICATION

BISHAI RAI, DAVID CARDOSO, DAVID DOCHERTY, MARY SOUSA, YASMIN PAKSOY

Abstract

Online payment applications are used by many to make efficient and systematic payments. As discussed in our project proposal, in a world now focusing on privacy, we have noticed a lack of bill splitting applications that are privacy orientated. In this report we discuss and evaluate the process undergone to develop the first working prototype of our application 'Splitsy', designed to allow users to split bills without sharing bank account information while ensuring payments are not forgotten.

We built this Android application using Java, MySQL, and XML in Android Studio. Other technologies were implemented to allow for efficient integration between these aspects. During development we used Agile methodology to ensure our product could be tested as thoroughly as possible. There are many problems we experienced when developing that we will further dive into in this report.

It would require more work for this application to achieve its highest potential. For instance, there are APIs that are yet to be implemented to ensure this application is as efficient as possible and satisfying for users.

We now have a working application that can be found on our GitHub repository. Our testing results have revealed that our implementations work well and are found to be enjoyable to use by our stakeholders.

Contact Details

Bishal Rai - brai001@gold.ac.uk

David Cardoso - dcard001@gold.ac.uk

David Docherty - ddoch001@gold.ac.uk

Mary Sousa - msous001@gold.ac.uk

Yasmin Paksoy - ypaks001@gold.ac.uk

GitHub Repository

https://github.com/ysmnpksy/Splitsy

Contents

Introduction

As described in our project proposal, we are working on developing a bill splitting application targeted mainly at individuals who are often going out with friends and want an easier way to split and track group payments; our research showed that 18 to 30-year-olds go out most often compared to other age groups, so they are our primary focus.

Our system has been designed to ensure privacy between users by ensuring they do not have to share bank details, as a bank transfer would require, and security in terms of bill creators always getting paid back the money they are owed, even if bill members forget, as payments will be taken automatically after the 'pay-back' date set by the bill creator has passed.

This application will allow our users to split their bills either equally or by item ways using their Android devices. The goal of this application is to aid users in paying each other back. The scope does not include protecting the bill creator by forcing bill members to pay their share; the bill creator accepts responsibility for the whole bill when creating the bill.

There were several different aspects we needed to implement to ensure our application met the minimum viable product we outlined in our project proposal. We had listed a set of 10 tasks we believed the user should be able to complete in our minimum viable product (MVP), which you can see in figure 1.

1. Sign up and enter bank details
2. Login and change your password
3. Create bill which is split equally and continue to the share code page
4. Check past bills and pay the pending bill with cash
5. Add new bank card to your account
6. Join bill, select the first item and complete payment with card
7. Logout
8. Login again and enable fingerprint
9. Enable high contrast mode
10. Create bill, which is split by item, scan a receipt, and continue to the share code page

Figure 1: List of tasks users should be able to complete in the MVP.

Our goal was to get as close to the MVP as possible in the give time frame. We decided to prioritize the tasks from most to least affecting functionality and build our agenda that way. Our main aims were:

- Replicating our high-fidelity prototype using XML.
- Establishing a connection to our database.
- Allowing users to create bills which are split equally.
- Implementing the camera API to allow users to scan bills.
- Conducting thorough testing to ensure our application did not have any major problems.

Our current system consists of a fully functioning user interface and operational connection to our database.

Overall, we found this process to be challenging; our application consists of many intricate individual aspects, all of which took a long time to implement. We worked on multiple parts simultaneously, splitting tasks into sub-groups in order to complete as many as possible.

More work is needed to reach a point where our application could be used efficiently by our stakeholders, but we are confident that our current prototype clearly demonstrates how our application will be beneficial to our target audience and how we can implement more of the required functionalities and features to develop a more complete product.

Planning and Research

Literature

We used several different types of literature during this project. We explored various books, blog posts, and articles to gather information regarding Android development. Regarding information about specific technologies, such as the camera API, we started by looking at the documentation to ensure we understood the logic, especially since we had no experience in many of the technologies we intended to implement. We then further developed our understanding of these technologies by looking at blog posts and articles detailing their implementation, which was essential for us to understand how to go about implementing them ourselves.

We made sure to review multiple sources to ensure the accuracy and validity of the information. By doing so, we were able to see examples of different methods for various implementations, allowing us to consider which would be the smartest for our aims, instead of going with the most basic method.

We first consulted multiple blogs that we had used in other research and knew to be trusted sources due to our prior experience with them. To ensure the quality of the various sources that we had not used before, we confirmed that they were tech oriented to ensure the information was reliable. We also checked that there were multiple contributors to the blog, usually by checking the about page, to ensure we were not consulting biased material.

Additionally, we have checked the date it was published, to ensure the source was not outdated, since technology develops at a fast pace and even information published a few years ago may not be viable anymore. We found that this was not a problem for most blog posts, since many authors update their posts as technology develops, which was very useful when assessing how the technology changed over time.

Project Management

Any project long in time requires the use of well-structured project management tools. As we did in our project proposal, we used Notion and a Gantt chart in order to manage our project.

Notion is a cloud-based application that combines a wide range of features into one place. Notion allowed us to create and store documents, lists, tables, and Kanban boards in order to organise and track our progress. All members of the group were able to contribute to and edit this information at any time. This collaborative space let us easily review our tasks and ensured we remained efficient in our time allocation. Notion additionally acted as a back-up of all data generated during the course of working on this project, so no information was accidently lost.

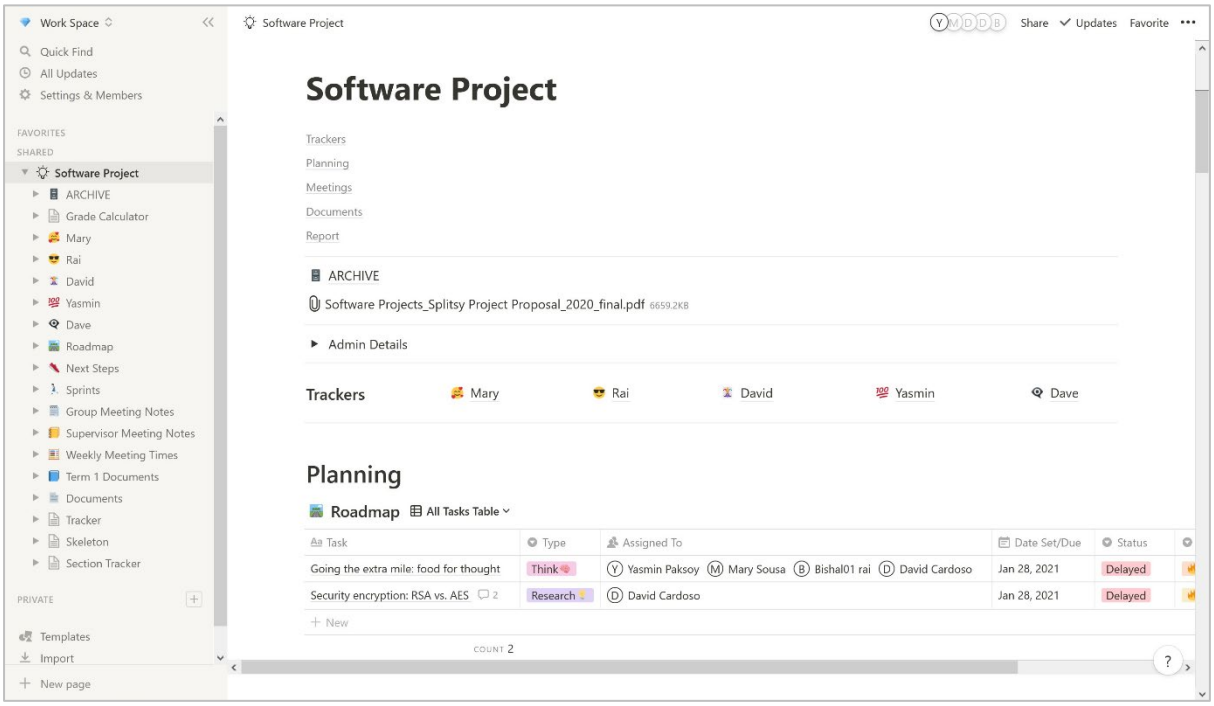


Figure 2: Project home page on Notion.

We used Notion to track our time spent on the project. Each member used their own time tracking table, allowing us to ensure we were putting it the necessary amount of time to not fall behind. We set 7 hours as our weekly targets as we felt this was the minimum amount of time we had to spend on this project per person in order to complete our initial minimum viable product in our given time frame. Below you can see an example of the time trackers we used:

Yasmin

Time Tracker

Week	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
Week 12	1.5	2.25	3.75	7	3.25	No
Week 13	1	5	6	7	1	No
Week 14	2	6.5	8.5	7	-1.5	Yes
Week 15	2	3.25	5.25	7	1.75	No
Reading Week	10	3	13	7	-6	Yes
Week 16	6	2	8	7	-1	Yes
Week 17	6	3	9	7	-2	Yes
Week 18	7.5	1.5	9	7	-2	Yes
Week 19	10.5	3.75	14.25	7	-7.25	Yes
Week 20	7	2.5	9.5	7	-2.5	Yes
Week 21	10	3.5	13.5	7	-6.5	Yes
Week 22	15	8	23	7	-16	Yes
Week 23	20	1.5	21.5	7	-14.5	Yes

+ New

Figure 3: Yasmin Paksoy time tracker.

The meeting time column is automatically filled in using another table that adds up all the time we spend in group and supervisor meeting, which you can see below:

Weekly Meeting Times

Week	Group Time	Supervisor Time	Total Time
Week 12	1	1.25	2.25
Week 13	3	2	5
Week 14	4.5	2	6.5
Week 15	2.25	1	3.25
Reading Week	3	0	3
Week 16	1.5	0.5	2
Week 17	2	1	3
Week 18	1.5	0	1.5
Week 19	2.5	1.25	3.75
Week 20	1	1.5	2.5
Week 21	3.5	0	3.5
Week 22	8	0	8
Week 23	1.5	0	1.5

+ New

COUNT 13

Figure 4: Weekly meeting times table.

These values are populated automatically using each meetings document, which details its date, the week it took place and how long it lasted. In these documents we also made notes of the topics discussed and decisions made, as well as the attendees present. We could easily refer to information from past meetings, and see the tasks and deadlines set for the week ahead. We also did this for meetings with our supervisor, allowing this critical feedback to be retained and applied going forward.

We uploaded all our documents to a 'Documents' sections on Notion, shown in figure 5, such as Entity Relationship Diagrams and technical notes. This served as a convenient storage place, as well as a useful backup of the copies we had locally. Each member could view and comment on each document, and the date each document was added is automatically recorded. We kept links to useful information here, including bibliography tables, as shown in figure 6, which we updated as we went to ensure we didn't lose track of the resources we have used. This way of sharing documents was especially useful as it allowed members to be able to catch up on tasks which they hadn't worked on themselves.

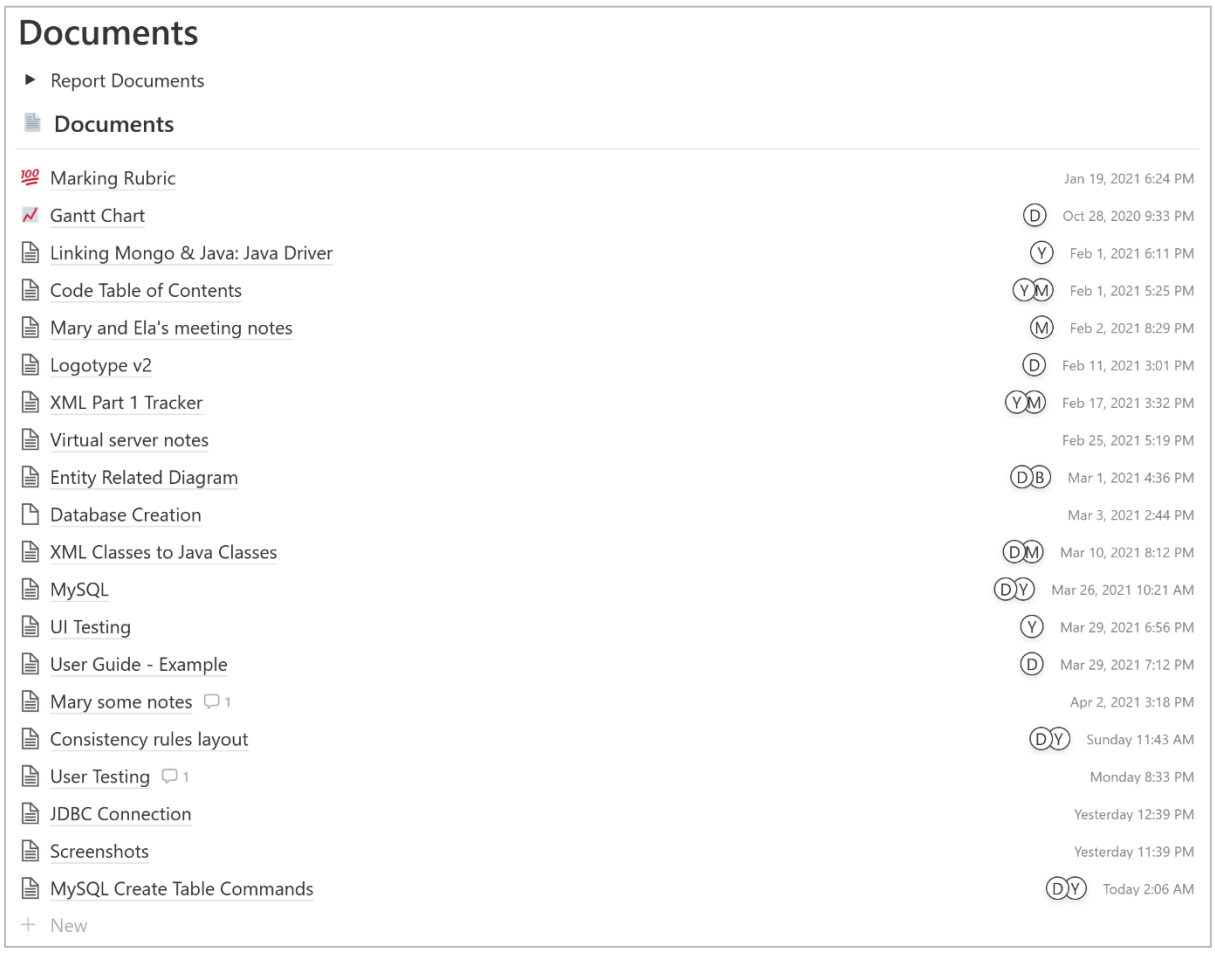
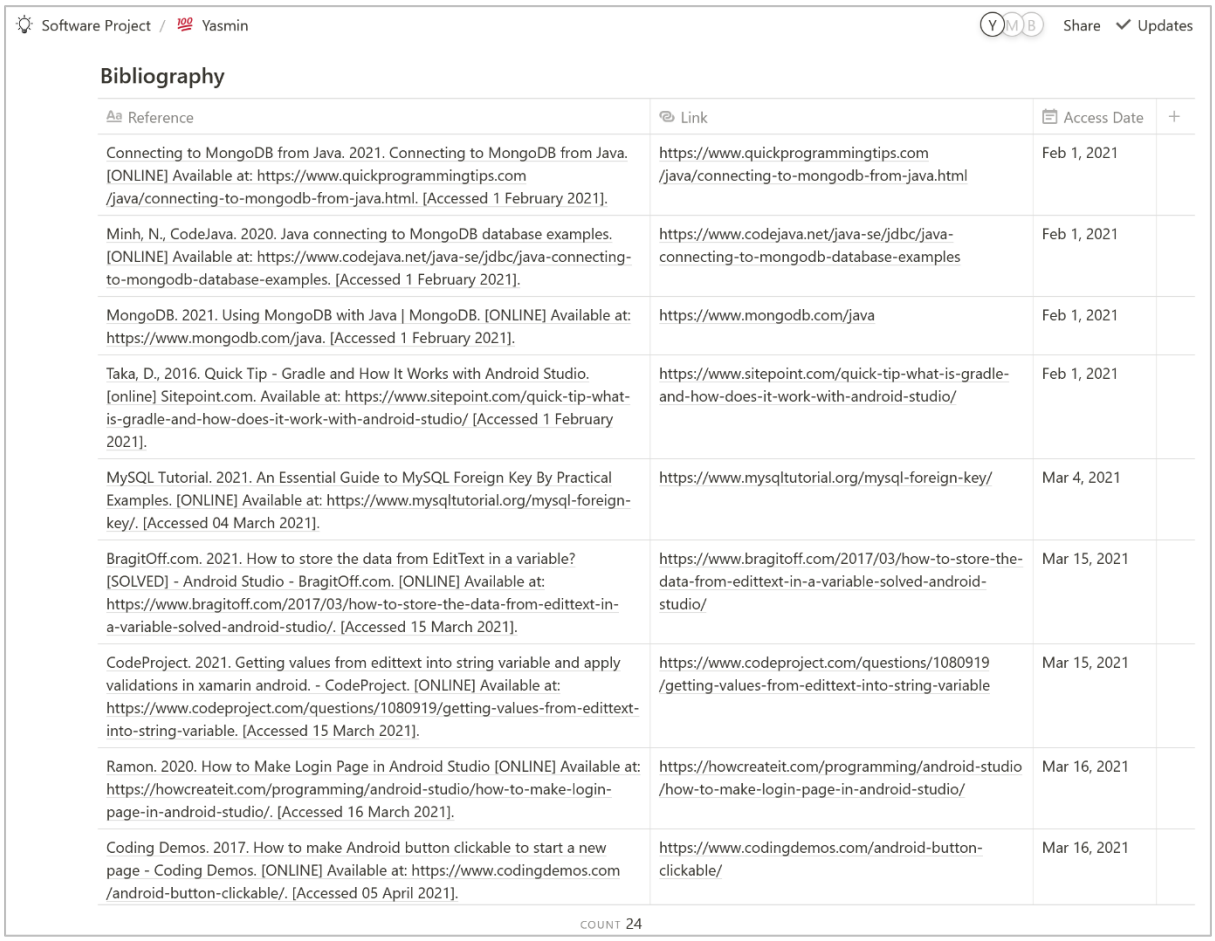


Figure 5: Documents list in Notion.



Bibliography

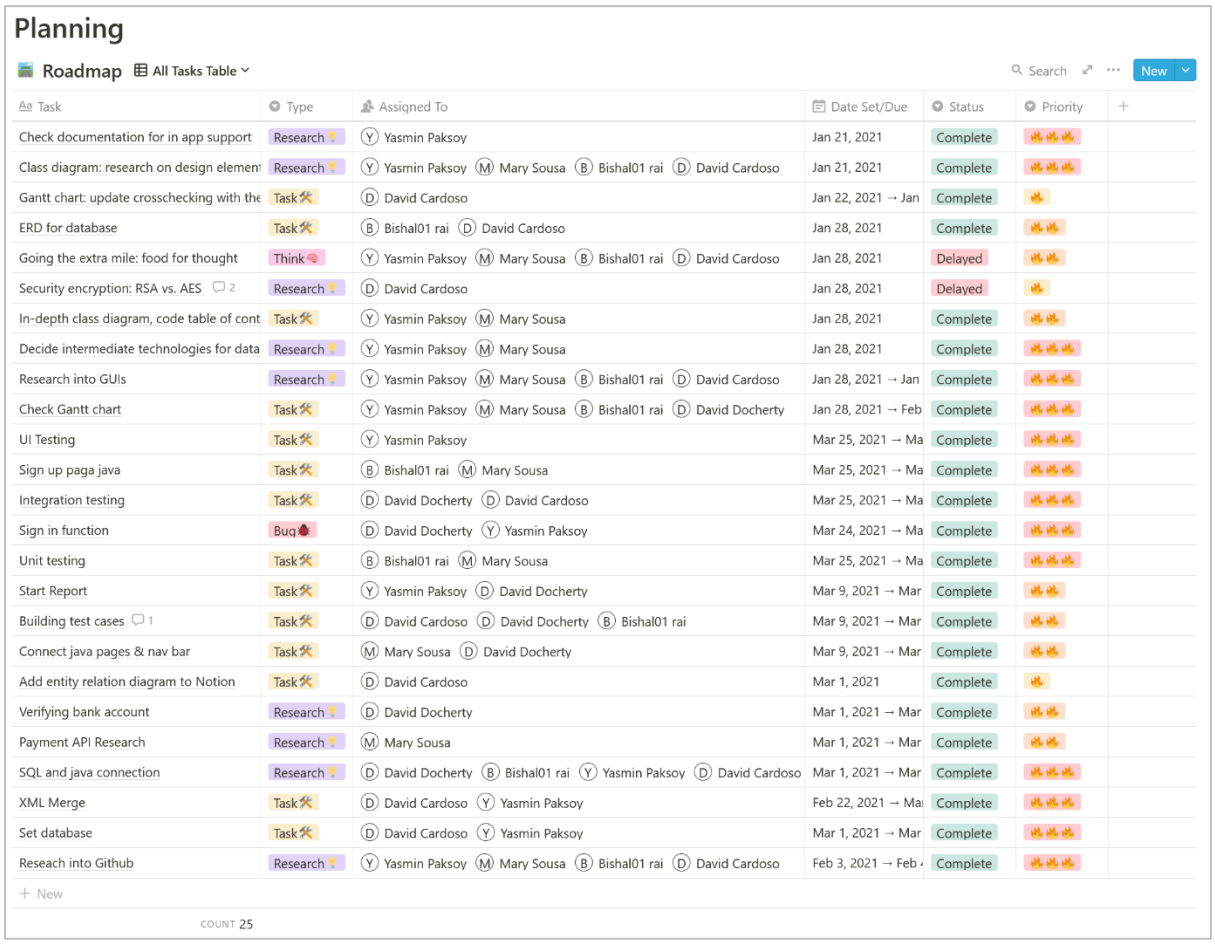
Reference	Link	Access Date
Connecting to MongoDB from Java. 2021. Connecting to MongoDB from Java. [ONLINE] Available at: https://www.quickprogrammingtips.com /java/connecting-to-mongodb-from-java.html. [Accessed 1 February 2021].	https://www.quickprogrammingtips.com /java/connecting-to-mongodb-from-java.html	Feb 1, 2021
Minh, N., CodeJava. 2020. Java connecting to MongoDB database examples. [ONLINE] Available at: https://www.codejava.net/java-se/jdbc/java-connecting-to-mongodb-database-examples. [Accessed 1 February 2021].	https://www.codejava.net/java-se/jdbc/java-connecting-to-mongodb-database-examples	Feb 1, 2021
MongoDB. 2021. Using MongoDB with Java | MongoDB. [ONLINE] Available at: https://www.mongodb.com/java. [Accessed 1 February 2021].	https://www.mongodb.com/java	Feb 1, 2021
Taka, D., 2016. Quick Tip - Gradle and How It Works with Android Studio. [online] Sitepoint.com. Available at: https://www.sitepoint.com/quick-tip-what-is-gradle-and-how-does-it-work-with-android-studio/ [Accessed 1 February 2021].	https://www.sitepoint.com/quick-tip-what-is-gradle-and-how-does-it-work-with-android-studio/	Feb 1, 2021
MySQL Tutorial. 2021. An Essential Guide to MySQL Foreign Key By Practical Examples. [ONLINE] Available at: https://www.mysqltutorial.org/mysql-foreign-key/. [Accessed 04 March 2021].	https://www.mysqltutorial.org/mysql-foreign-key/	Mar 4, 2021
BragitOff.com. 2021. How to store the data from EditText in a variable? [SOLVED] - Android Studio - BragitOff.com. [ONLINE] Available at: https://www.bragitoff.com/2017/03/how-to-store-the-data-from-edittext-in-a-variable-solved-android-studio/. [Accessed 15 March 2021].	https://www.bragitoff.com/2017/03/how-to-store-the-data-from-edittext-in-a-variable-solved-android-studio/	Mar 15, 2021
CodeProject. 2021. Getting values from edittext into string variable and apply validations in xamarin android. - CodeProject. [ONLINE] Available at: https://www.codeproject.com/questions/1080919/getting-values-from-edittext-into-string-variable. [Accessed 15 March 2021].	https://www.codeproject.com/questions/1080919 /getting-values-from-edittext-into-string-variable	Mar 15, 2021
Ramon. 2020. How to Make Login Page in Android Studio [ONLINE] Available at: https://howcreateit.com/programming/android-studio/how-to-make-login-page-in-android-studio/. [Accessed 16 March 2021].	https://howcreateit.com/programming/android-studio /how-to-make-login-page-in-android-studio/	Mar 16, 2021
Coding Demos. 2017. How to make Android button clickable to start a new page - Coding Demos. [ONLINE] Available at: https://www.codingdemos.com /android-button-clickable/. [Accessed 05 April 2021].	https://www.codingdemos.com/android-button-clickable/	Mar 16, 2021

COUNT 24

Figure 6: Bibliography table.

Notion made it easy keep track of task assignment and status. We were able to set internal deadlines for deliverables such as researching topics, fixing bugs, and implementing new functionalities. Important information such as its state, priority, due date, and the group member the task was assigned to was clearly visible in a table, which is shown in figure 7. This gave us a good overview of our project, and the ease of updating these details ensured that the information would remain accurate as the project went on. The table had multiple views which allowed each member to see a filtered version that contained only the tasks that had been assigned to them.

Members: brai001, dcard001, ddoch001, msous001, ypaks001



Planning

Roadmap | All Tasks Table | Search | New

Task	Type	Assigned To	Date Set/Due	Status	Priority
Check documentation for in app support	Research	Yasmin Paksoy	Jan 21, 2021	Complete	
Class diagram: research on design elemen	Research	Yasmin Paksoy, Mary Sousa, Bishal01 rai, David Cardoso	Jan 21, 2021	Complete	
Gantt chart: update crosschecking with the	Task	David Cardoso	Jan 22, 2021 → Jan	Complete	
ERD for database	Task	Bishal01 rai, David Cardoso	Jan 28, 2021	Complete	
Going the extra mile: food for thought	Think	Yasmin Paksoy, Mary Sousa, Bishal01 rai, David Cardoso	Jan 28, 2021	Delayed	
Security encryption: RSA vs. AES 2	Research	David Cardoso	Jan 28, 2021	Delayed	
In-depth class diagram, code table of cont	Task	Yasmin Paksoy, Mary Sousa	Jan 28, 2021	Complete	
Decide intermediate technologies for data	Research	Yasmin Paksoy, Mary Sousa	Jan 28, 2021	Complete	
Research into GUIs	Research	Yasmin Paksoy, Mary Sousa, Bishal01 rai, David Cardoso	Jan 28, 2021 → Jan	Complete	
Check Gantt chart	Task	Yasmin Paksoy, Mary Sousa, Bishal01 rai, David Docherty	Jan 28, 2021 → Feb	Complete	
UI Testing	Task	Yasmin Paksoy	Mar 25, 2021 → Ma	Complete	
Sign up paga java	Task	Bishal01 rai, Mary Sousa	Mar 25, 2021 → Ma	Complete	
Integration testing	Task	David Docherty, David Cardoso	Mar 25, 2021 → Ma	Complete	
Sign in function	Bug	David Docherty, Yasmin Paksoy	Mar 24, 2021 → Ma	Complete	
Unit testing	Task	Bishal01 rai, Mary Sousa	Mar 25, 2021 → Ma	Complete	
Start Report	Task	Yasmin Paksoy, David Docherty	Mar 9, 2021 → Mar	Complete	
Building test cases 1	Task	David Cardoso, David Docherty, Bishal01 rai	Mar 9, 2021 → Mar	Complete	
Connect java pages & nav bar	Task	Mary Sousa, David Docherty	Mar 9, 2021 → Mar	Complete	
Add entity relation diagram to Notion	Task	David Cardoso	Mar 1, 2021	Complete	
Verifying bank account	Research	David Docherty	Mar 1, 2021 → Mar	Complete	
Payment API Research	Research	Mary Sousa	Mar 1, 2021 → Mar	Complete	
SQL and java connection	Research	David Docherty, Bishal01 rai, Yasmin Paksoy, David Cardoso	Mar 1, 2021 → Mar	Complete	
XML Merge	Task	David Cardoso, Yasmin Paksoy	Feb 22, 2021 → Ma	Complete	
Set database	Task	David Cardoso, Yasmin Paksoy	Mar 1, 2021 → Mar	Complete	
Reseach into Github	Research	Yasmin Paksoy, Mary Sousa, Bishal01 rai, David Cardoso	Feb 3, 2021 → Feb	Complete	

+ New

COUNT 25

Figure 7: Task tracker in table view on Notion.

We also kept a link to our Gantt chart here, ensuring we could refer to it easily and regularly. A Gantt chart is a type of bar chart representing a project plan over a length of time. Segments are split into smaller tasks, all of which have corresponding bars which determine how long they are expected to last and when they will take place. This allowed us to visualise our workflow and made it easy to assess what stage of the project we were at to determine if tasks were progressing as planned.

Below you can see the first version of our Gantt chart which details our initial plan for the project timeline. This is a continuation of the Gantt chart that we used to map the progress of our project proposal. As shown in figure 8, we started working on development in week 11.

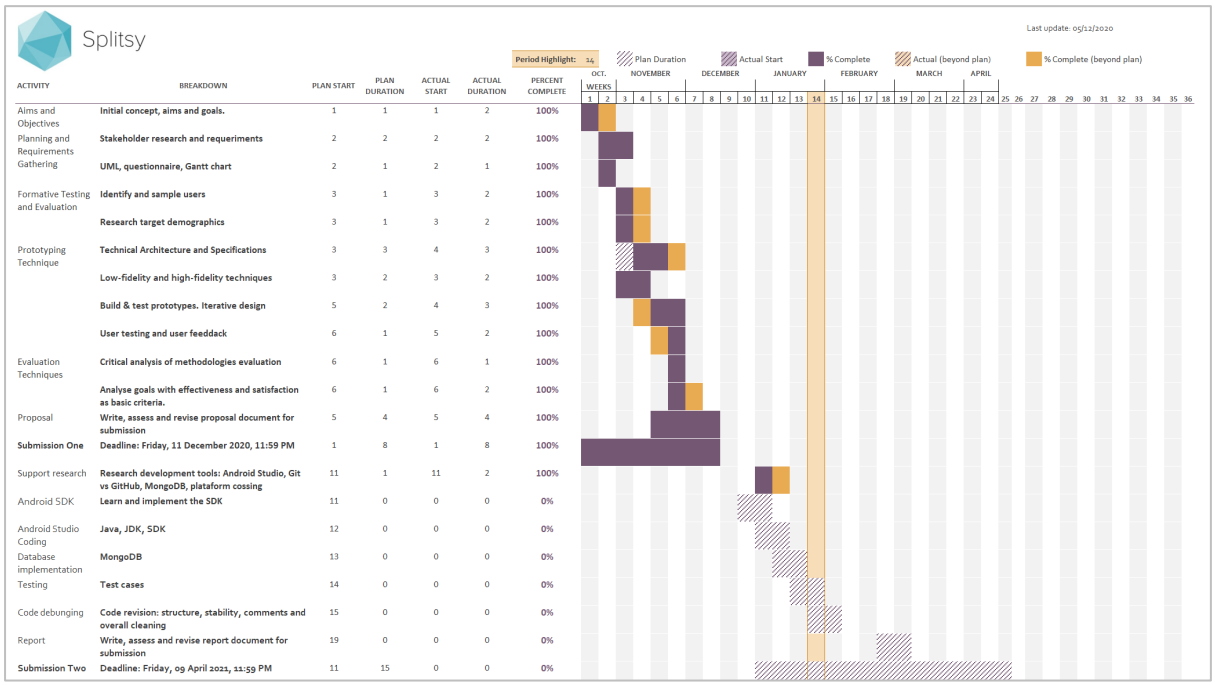


Figure 8: Initial Gantt chart draft for the second half of the project.

Initially we set the same amount of time for every aspect of the project. Each aspect was allocated 2 weeks, which we believed to be a sufficient amount of time for that section to be completed. As we had decided to use an Agile methodology, we would be working simultaneously on multiple tasks, hence the sections overlap on the Gantt chart. As we progressed through the development of the project, we formed a better understanding of which aspects would require the most time. This allowed us to better allocate time in relation to the functionalities that needed to be implemented. These changes are reflected in our updated Gantt chart, which you can see in figure 9. For example, the "Android Studio Coding" activity was allocated additional time as we came to realise that it was more time-consuming than we initially anticipated.

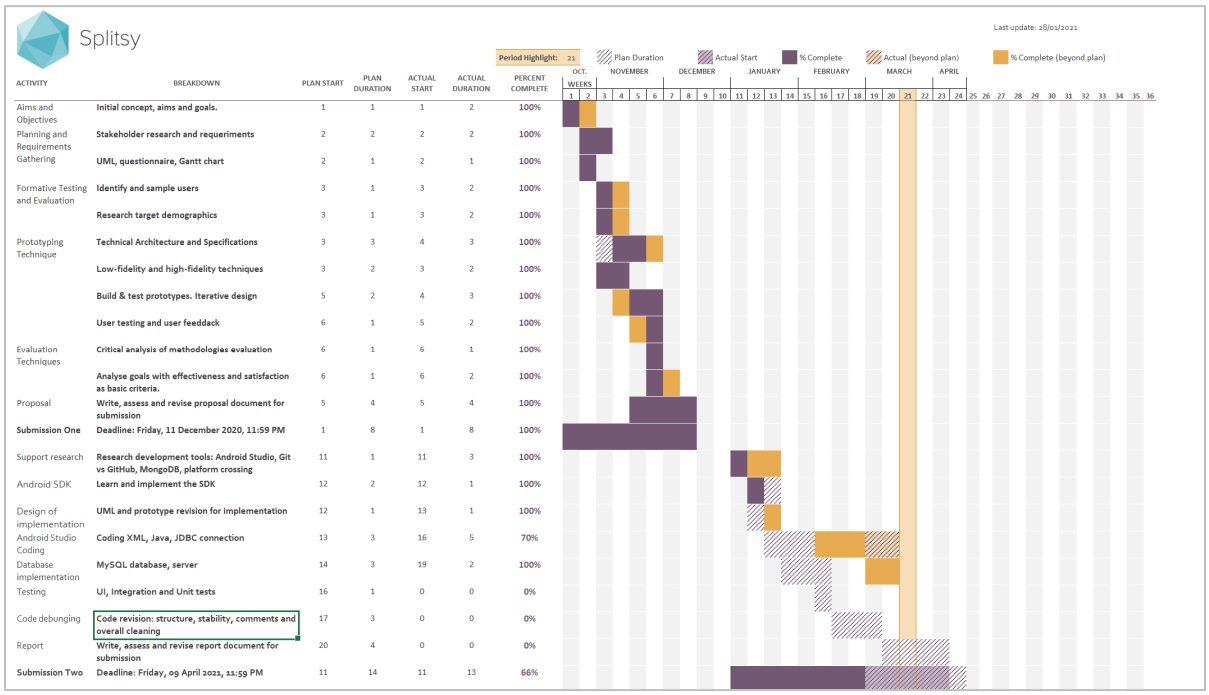


Figure 9: Updated version of the Gantt chart.

Android

According to the official Android developer's website, "API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform" [1]. The application interacts with the underlying Android system with the help of a framework API such as classes and packages, intents, user permissions.

We used manifest elements such as min SDK, target SDK and compile SDK to describe the minimum, target, and compile API Levels under which the application can run and function efficiently.

compileSdkVersion is the Android version that is used to compile and build the application by Android's build tools in order to run, debug and release the application. We set the compile SDK version to the latest SDK (30) so that we can get compilation checks on existing code, avoid new deprecated APIs, and use new available APIs [2].

Minimum SDK helps to check if users are running a compatible Android operating system on their devices. It helps in installation and running the app's functionality effectively on their devices. API level 26 and above covers more than 70% of the market share [3]. Hence, we used API level 26 as min SDK for our application so that we can implement biometric authentication, CameraX API and reach a larger audience.

Every new Android version is launched with some improved security and performance features. We declared the target SDK to the API level 30 so that users can access new APIs, improved security, performance, and compatibility features, while still allowing the application to run on older Android versions.

Android devices come in different shapes and sizes. Creating a responsive layout helps to reach a number of devices of different screen sizes. Constraint layout helps to create a responsive UI allowing to specify the position and size of each view component [4]. Hence, we decided to use the constraint as the base layout in our UI.

Our research suggested we should have a target screen size of 6 inches since most of the Android users have bigger screen size on average [5]. During development, we decided to use the Google Pixel 4 XL when implementing our UI, since it has a 6-inch screen size, is high resolution and had all additional features, such as a fingerprint. Moreover, we decided that we would test our application on other screen sizes as well because not all users will have devices of the same screen size [6].

Design

Minimum Viable Product

As stated in our project proposal, "a minimum viable product is a version of a product with just enough features to be usable by early customers who can then provide feedback for future product development" [7]. An MVP is used to provide a product to users that meets their essential needs as early as possible. They in turn can give back valuable user data that can improve future iterations of the product. This provides several benefits: core functionalities are made a priority, less resources are spent on unimportant features and user satisfaction is kept high [8]. As we aimed to create a product which is intuitive and pleasant to use, building an MVP and collecting feedback was important.

When conceptualising our MVP, we had to be careful in considering which features to incorporate. Some basic features that we wanted to include may possess dependencies which were less obvious. We found it helpful to begin by making a list of the functionalities that comprise our MVP, which can be seen below. We could then distil this into a series of components that we needed to create and select the technologies that we would use to implement these.

As our project is a bill-splitting application, our users must be able to create and view bills. Other users should be able to join these bills to pay their share. To keep track of users, we will need each user to have an account. This necessitates a way for users to create accounts, and a system for logging in and out. The data for these accounts must be stored securely, to be retrieved or updated when needed. Bill information will also need to be created, stored, and updated.

These are the tasks we wanted our MVP to be capable of completing:

1. A user can create a new account and enter their bank details. This information is saved in a database.
2. A user can sign into the application with their account information. This is compared with the information saved in the database to confirm or reject the login attempt.
3. A user can change their password. This will update the information saved in the database.

4. A user can create a bill. This bill is saved in the database.
5. A user can check a past bill. This information will be retrieved from the database.
6. A user can pay a pending bill via a payment processor. This will require a payment API. The database will be updated.
7. A user can add a new bank card to their account, which is then saved to the database.
8. A user can share a bill using the code assigned to that bill. This will need to be generated when the bill is created.
9. A user can join a bill using the code assigned to that bill. This will need to be generated when the bill is created. This information will be added to the database.
10. A user can log out from their account.
11. A user can scan a bill using their camera. This will require a camera API.

These are the components we needed to implement to allow the above:

- Back-end database – Tasks 1, 2, 3, 4, 5, 6, 7, 9
- A process for creating an account – Task 1.
- A process for logging into an account – Task 2
- A process for updating a password – Task 3.
- A process for creating a bill – Task 4.
- A process for viewing a past bill – Task 5.
- A process for paying a pending bill – Task 6.
- Payment API – Task 6
- A process to add a new bank card – Task 7.
- A process for generating a share code when a bill is created – Task 8.
- A process for joining a bill using a share code – Task 9.
- A process for logging out of an account – Task 10.
- A process for scanning a bill using the camera – Task 11.
- Camera API – Task 11

As a group we decided to use Android Studio to create our client-side application. All of the components described above as "processes" would be implemented here. These were the most essential items, as at least one exists for each MVP task. The front-end pages and activities that encompass these components would have their designs realised using XML and their functionalities coded in Java. User interface functionality that is not included in the above lists would also be implemented in this way.

The second most important element of the MVP was the back-end database, as it is required for almost every task. Together with the above Android Studio components, the database forms the core of our application. Information is passed between these two sides to store and update records and provide data to the client-side modules. We decided to use a MySQL database.

We use middleware technologies to connect our front-end application and our back-end database. We used Java Database Connectivity (JDBC) for this purpose. JDBC is a Java API for connecting programs written in the Java programming language with various types of databases. For the more specialised elements of

our MVP, we used dedicated software. For our payment processing API, we used Stripe, and for our Camera API we used CameraX.

Please read on for a detailed explanation of each of these technologies and why we chose to use them in particular.

XML

XML (Extensible Markup Language) is a metalanguage that can be used to create markup languages for specific applications [9]; markup languages are used to describe data [10]. In our case, we will use XML pages to create our client-end user interface. This is not something that we can do with a programming language such as Java; programming languages are used to describe behaviours, interactions and conditions and cannot describe the layout of our pages like XML can.

Through the use of XML, we will set the layout of elements such as buttons, images and text as well as define their colour, size, font etc, on each page. XML can also be used describe data other than the layout of pages; for example, a strings.xml file can be made to store all the text in your application, which layout files will refer to [10].

We decided to use this method for defining the layout of our application for multiple reasons. While XML does have redundancy in its syntax, leading to a higher storage and transportation cost, compared to alternatives such as JSON [11], it separates functionality from presentation; this would make it easier to refactor code without changes being made to one side affecting the other. Additionally, XML supports multiple screen sizes and configures the elements accordingly. This is fundamental for Android devices since the size of Android screens can vary [12].

MySQL

In our first report, we have chosen MongoDB as it *"speeds up the data retrieval process", "provides security features such as authentication, access control and encryption"* [13] and because we would *"use a row-based data storage type, of which MongoDB is one we have the most experience with".* During implementation of our database, we concluded that our data structure required a relational database. This is due to there being many tables that have multiple relations with others, which would be best implemented using structured tables.

SQL is a more matured relational database that allows the use of relational tables for complex inflexible data structures across a large number of tables, such as our application, while also providing the same level of security, authentication, and high performance as MongoDB would have [13].

Using SQL would ensure that the database is structured as the fields would be fixed, unlike when using MongoDB, where fields are added individually for each record in a collection. This will limit the chances of running into errors when conducting queries in the application, which may have been a bigger problem when using MongoDB. Specifically, we decided to use MySQL, which is an open-source database management system backed by Oracle, as it runs on numerous

platforms such as Windows, Linux, Unix, macOS and so on; is very fast, scalable in the order of millions, reliable due to being constantly improved by the open-source community and supports numerous data types as required for our application database [15].

Additionally, this is the type of SQL database that we are most experienced with, and thus was the one we felt most confident in using it for this project. We decided that we would host our database on a virtual server provided to us by the University. Below you can see the entity relationship diagram (ERD) showing the design of our database.

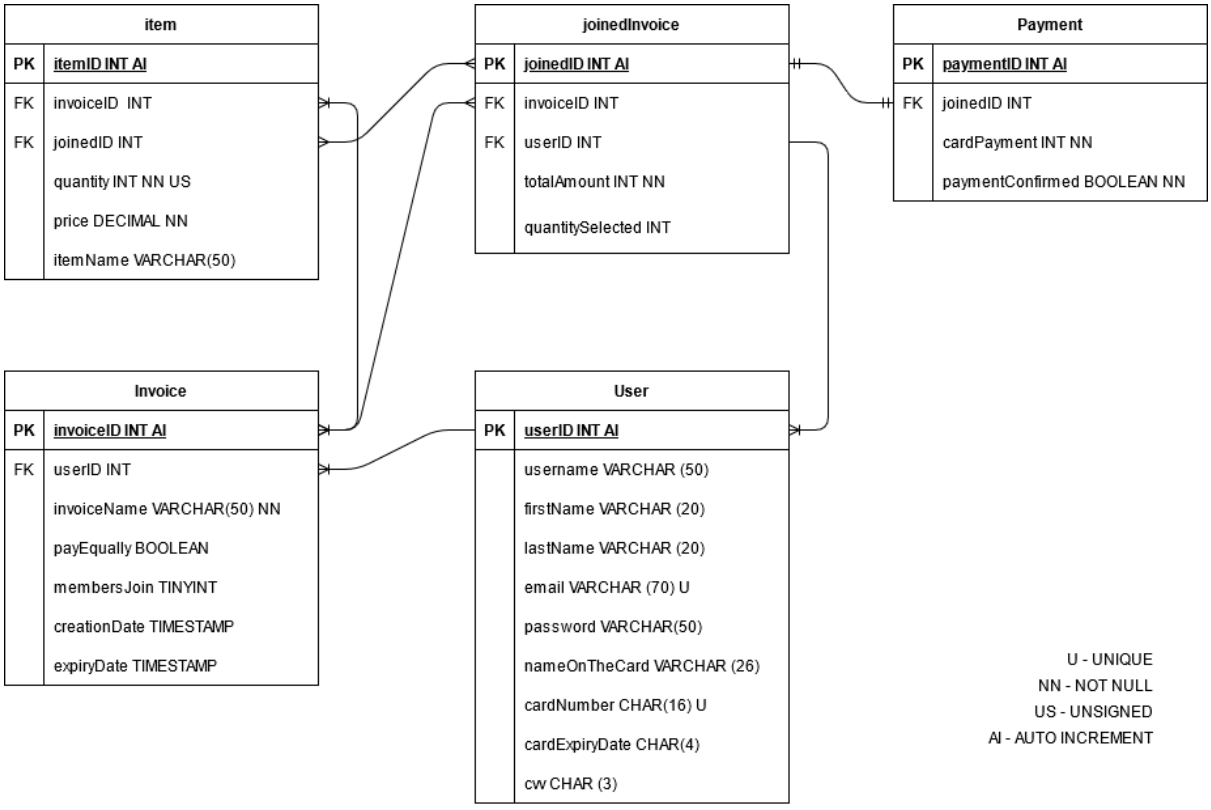


Figure 10: Final ERD.

"An entity relationship diagram (ERD) shows the relationships of data sets stored in a database" [16]. Creating an ERD gave us the opportunity to define the entities, their attributes, and show the relationships between them, allowing us to illustrate the logical structure of database, which made implementing process much easier.

All entities in our ERD consist of an auto-incrementing ID attribute as well as various other attributes that are required to complete all processes in our application. The way those entities relate to each other is called cardinality [16]. Since there are multiple many to many relations in our structure's cardinality, we have used a junction table, joinedInvoice, that will record each individual item added to all the bills created using our application. All the items that belong to a particular bill can be found by filtering this table using the bills invoiceID attribute. Using this structure allowed us to keep a record of each item added to our database, which would not be possible without the use of a junction table, since the number of items added to a bill does not have a set amount.

Through the use of an ERD, we gathered the means to visualize how the data entities our application handles are related. This ERD was a very beneficial reference when implementing many processes during development.

JDBC

In order to create a bridge between our application with our MySQL database, we decided to use JDBC.

JBDC stands for Java Database Connectivity and is "a Java API that can access any kind of tabular data, especially data stored in a Relational Database" [17]. This API works through Java and runs on most platforms such as Unix, Mac OS, and Windows. This API would allow us to connect to our database, execute SQL statements and receive results which we can then process.

Despite the existence of alternatives for this task, our research established that JDBC was the logical choice for being a low-level API and therefore suitable for our core requirements: creating, reading, updating, and deleting (CRUD) database records.

Payment API

In our project proposal, we decided to use Noodlio Pay for our payment processes in our application. Noodlio was a free and cost-efficient payment API [18] that processes transactions through another payment API, Stripe. We also considered Stripe itself, which is an "online payment processing gateway that allows businesses to send and receive payments over the internet" [19]. Stripe charges over 2.9% per successful transaction using a flat rate system [20]. Noodlio was a quick and straightforward solution for accepting payments compared to Stripe [noodlio2]. However, during our further research, we found that Noodlio Pay no longer facilitates payment services and has not been updated in 5 years.

Since Noodlio used to process transactions through Stripe, we decided to go for Stripe itself since it was the closest alternative. Stripe is a complete payment gateway platform. It allows us to accept credit card payments (in person or online) by transferring money between a merchant account and a payment processor using a physical credit card terminal or an online processor. Stripe has a cloud-based infrastructure designed for reliability, scalability, and security. It's very developer-centric, works for business websites and applications, and is rooted in both code and design [21].

Camera API

We decided to implement CameraX in our app, an addition to Jetpack. According to the official Android developer's website, "Android Jetpack is a suite of libraries created to help you write high-quality apps that follow best practices, use less boilerplate code, and work consistently across Android versions and devices" [22]. CameraX supports all devices running Android 5.0 (API level 21) or higher because it uses the Camera 2 API library that only supports devices with a minimum SDK of 21 [23]; it provides the same consistency making it compatible with Android (90% of devices) as shown in figure 11. Hence, CameraX is consistent with our Android API level 26.

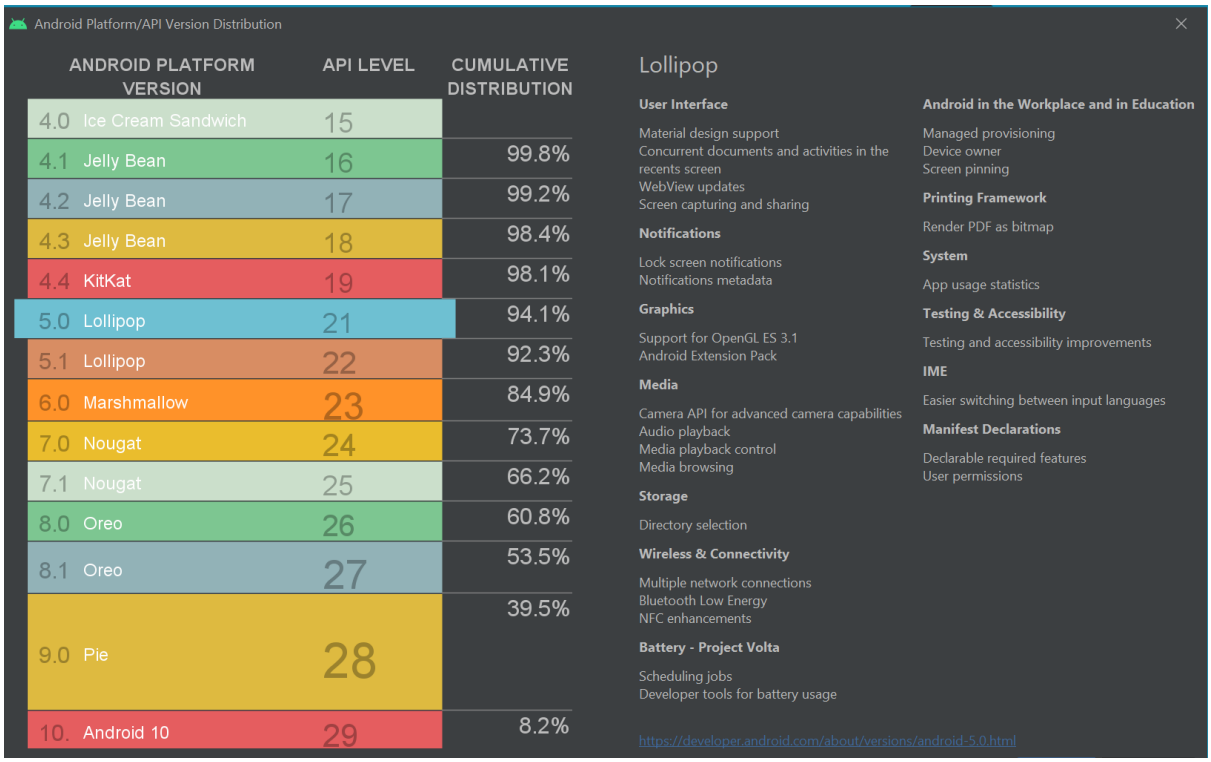


Figure 11: API Version Distribution.

The library provides several compatibility fixes and workarounds to help make the developer experience consistent across many devices. It allows access to the device's camera features and effects on supported devices with an add-on also known as Extensions [24] such as portrait: keeps objects in the foreground focused while blurring the background [25], HDR (High Dynamic Range): allows you to capture image with more detail widening the exposure range [26], and Night mode: captures different images of the same subject at varying exposure levels by taking them at different shutter speed lengths [27]. These features would allow a more comprehensive implementation of the camera in our application.

Android Studio

We compared different IDEs (Integrated Development Environments) relevant to Android application development during our research, such as Eclipse, IntelliJ, and Android Studio.

Eclipse is not a native Android IDE and makes project setup more complicated than other alternatives [28]. It does not have regular updates for Android development and is heavy and thus more resource consuming. IntelliJ IDEA was created by the company JetBrains. IntelliJ supports cross-platform development but lacks an Android Package that is used by the Android operating system for installation and distribution of the application, which makes it less appropriate for our project [29].

"Android Studio is the official IDE for Android application development, based on IntelliJ IDEA" [30], which supports various languages based on Java Virtual Machine (an engine that provides a runtime environment to control code or applications), such as Java, Groovy, and Kotlin. Android Studio has excellent tools

for Android application development. For example, it has powerful code editing tools, such as a fast and feature-rich emulator [31], which will be essential for testing our functionalities during development. Android Studio also has GitHub integration, which will be great for keeping the entire team in sync with changes to ensure that we are all working on the most up-to-date version of the code.

Unlike Eclipse, in Android Studio the Android development features are the latest, and updates are periodic. Additionally, Android Studio is lighter and supports drag-and-drop elements for a more straightforward XML layout design. Therefore, for our purposes, Android Studio is a better choice for development [28].

System Development

System development is the execution of everything settled to this point, the personification of all the prior research, decisions, and findings, now translated into a product soon to be real. Here, we have used a plethora of tools in conjunction with a rich set of processes, assets and work techniques that enhanced the development of our application, which we will further describe in the following sections.

Agile Methodology

As mentioned previously, we used Agile methodology while working. We decided against using a waterfall development methodology as it would have made it difficult to make changes during development since it consists of sequential steps. Additionally, it would have delayed testing until after development was completed. This could have caused a substantial delay had we realised during testing that large revisions had to be made [32].

Agile methodology allowed us to make the most of a small team of people by splitting our work into smaller, self-contained iterations. Since we would be working collaboratively, Agile was the more efficient choice, as team members did not have to wait for a certain task to be completed before they could start another.

Agile allowed us to be focused on user needs during development as we were able to test sections that were still in progress and make changes according to feedback. This adaptability ensured our final application would have high user satisfaction [33]. Since we could adapt to user needs during development, we eliminated having to do rework at the end of the development cycle, saving us time [34].

As we had planned in the initial project proposal, we used a mix of Kanban and Scrum as our type of Agile methodology. Scrum is a process framework used to manage product development; it is empirical in that it provides a means for teams to establish a hypothesis of how they think something works, try it out, reflect on the experience, and make the appropriate adjustments [35]. It generally consists of sprints and scrum meetings.

Sprints allowed us to focus on the most important sections that needed to be completed; each distinctive section was broken down into a sprint, where the section was planned, designed, implemented, and tested. Each sprint had at least

one meeting in its duration in order to discuss any problems and exchange ideas. This provided several benefits:

- Each sprint was planned in advance, minimising the delay between tasks.
- It allowed greater focus on, and familiarity with, the features being implemented.
- It allowed us to quickly identify any potential problems and to adapt to them sooner.
- Each sub-group could take ownership of a different development process, to be merged later upon completion.

The Kanban Method is a means to design, manage, and improve flow systems for knowledge work. The method also allows organizations to start with their existing workflow and drive evolutionary change [36].

Each sprint had its own Kanban board consisting of each feature to be added. Using a Kanban board gave us a clear overview of which tasks were most important and which needed more support [34]. We arranged our tasks by priority and status. Tasks that were in progress were completed before we moved onto the next highest priority task. Figure 12 shows the Kanban board we used during our second sprint.

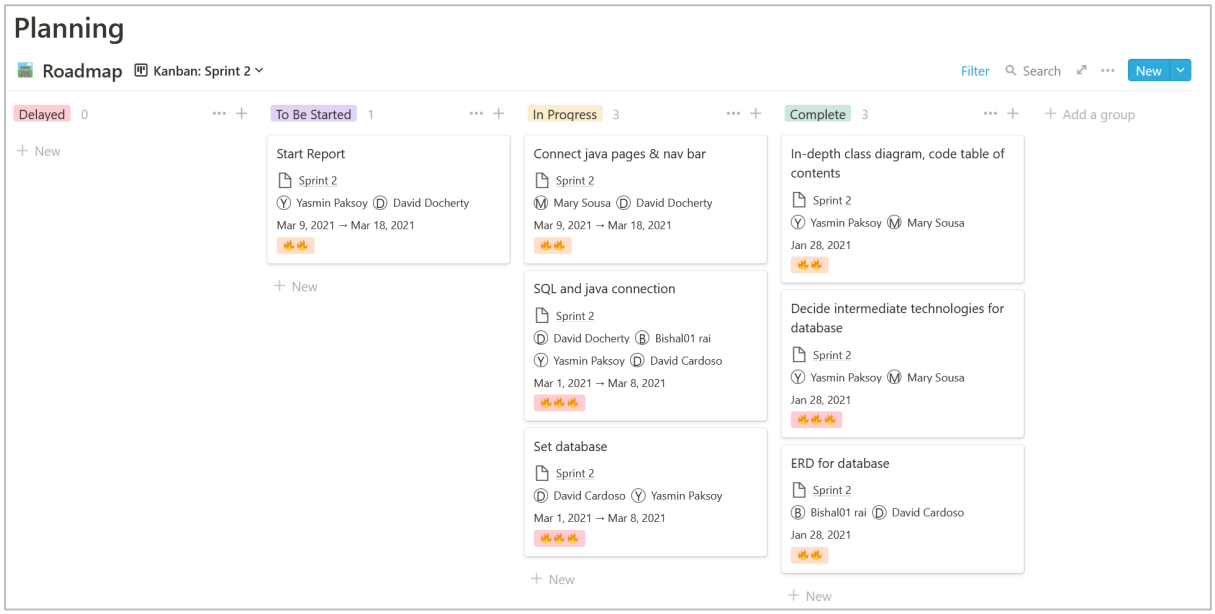


Figure 12: Sprint 2 Kanban board.

In addition to the meeting we held during sprint, we also held 2 meetings a week. In every meeting we were able to present the progress we had made and discuss what was next to be done, as well as share any challenges we had encountered. Some meetings also included code reviews, ensuring future components could be implemented in a consistent style. This gave all members of the team an opportunity to receive feedback on their progress which could be adapted into the next iteration.

Git Repository

Since Agile methodology meant we would be working on multiple different aspects of the application, version control was essential. To ensure no code was overwritten by different members of the team, we used the version control system GitHub.

Version control allows developers to work safely through branching and merging [37]. Branching allows you to duplicate part of the source code to work on without affecting the rest of the project. Once the changes made on this branch are complete, it can then be merged back into the main source code to make it official. Git is a distributed type of version control system, which means that the entire codebase and history is available on every developer's computer, allowing for easy branching and merging [37].

GitHub, a cloud-based Git repository hosting service, easily allowed us to keep track of changes made in the repository and ensured that previous versions were not lost with every iteration. Without such a system, we would have to manually send each member the code for each change/addition and ensure we were working on the same system. This would be highly time consuming and could have developed into a major problem as we were working both simultaneously and remotely, hence GitHub being one of the most essential pieces of software used.

We created one repository and used different branches for each aspect in the development process. In figure 13, for example, the creation and merging of branches is evident as is possible to see the green branch being created at the bottom to then be merged back at the top of the figure.

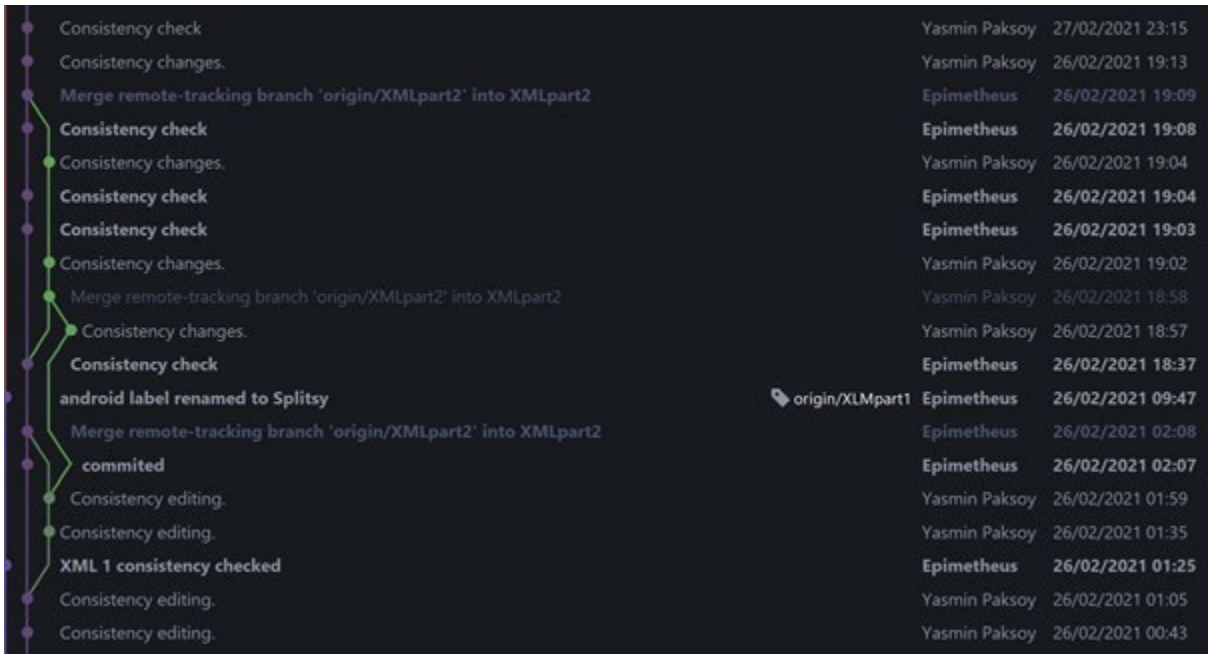


Figure 13: Our GitHub branches.

In the same way, figure 14 provides a different perspective of the same process over the course of a month. We begin the month with 3 branches, increasing that number to 5 as we implement different functionalities. These 5 branches are then merged into 3 branches by the end of the month as implementations are executed.

Figure 14: GitHub branches timeline snippet.

This work methodology allowed us to work simultaneously on different functionalities and merge our different branches once implementation was complete. Merging was only done after code reviews had taken place, to minimize the chances of exponentially producing bugs over time.

Overall, GitHub proved to be an essential tool for an efficient development process of our project.

Structure

Structure refers to the way the code is made up of a number of parts which are arranged to form a complete application.

We wanted to ensure our application had an efficient structure. This minimised the changes of new bugs affecting other functionalities and made it easier to find them, since page had its own XML and Java file. Additionally, as we were implementing an Agile methodology, it meant that we were less likely to run into merging conflicts, as most of the time members of the team were working on different files that had little to no effect on other files/functionalities.

We structured each page of our application to have two files; an XML file to define layout and a Java file to program functionality. We ensured that the file names were consistent with each other to make it clear which XML file corresponded to with Java file, for example, the XML file for the past bills page was names activity_past_bills.xml while the Java file was named Past_Bills.java.

We also ensured that all XML and Java files were names using the same convention to make the contents of the file clear: all XML files had the structure activity_page_name.xml while all Java files had the structure Page_Name.java. This way we were able to differentiate between XML and Java files easily, making coding more efficient.

To ensure our code was easy to maintain and advance in the future, we also used XML files to describe other relevant data. We used an XML file, names string.xml, to store all of our strings, as you can see in figure 15 below.

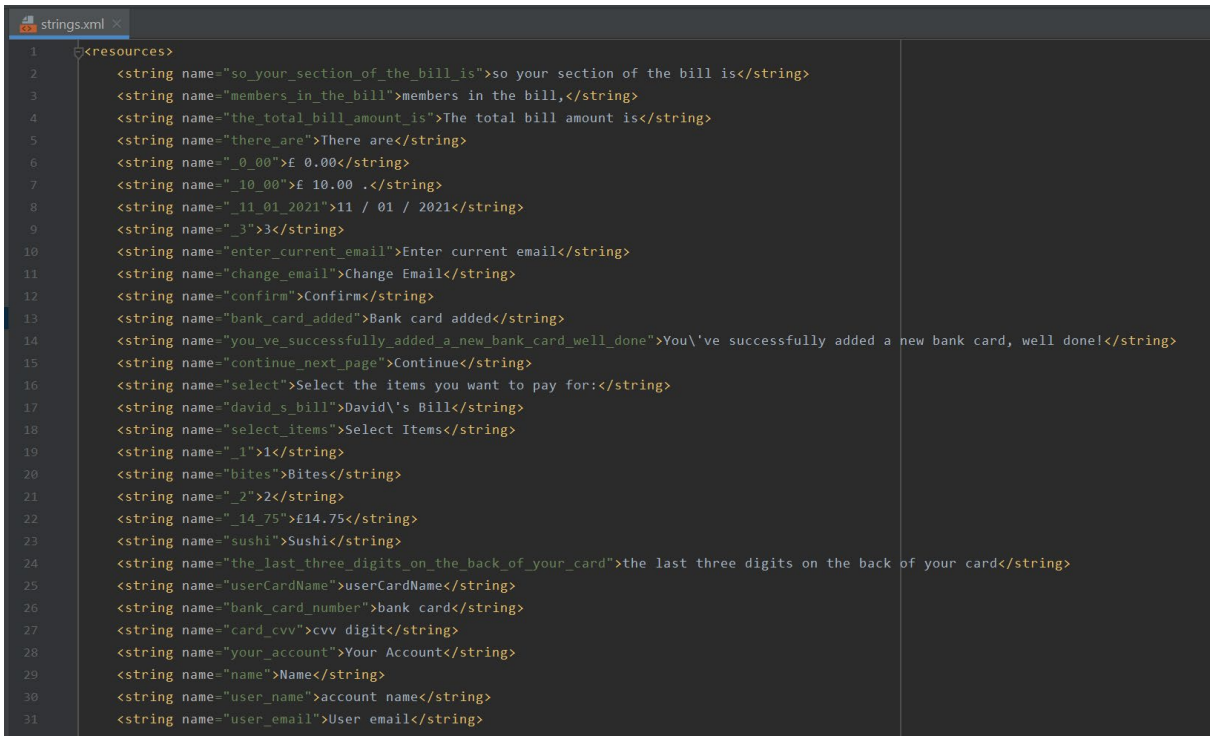


Figure 15: strings.xml file snippet.

Instead of typing out the text on each XML/Java page, we could add the reference for the relevant string. This file allowed us to keep all of our strings in one place and made for more efficient coding as we didn't need to type out the full string each time. This method also made sure our application didn't have any redundant data, since each string only needed to be added once but could be referenced from multiple XML/Java pages.

Using this same method, we also made a colors.xml files to ensure consistency through the application. This file consisted of all the colours we used in our application, which you can see below.

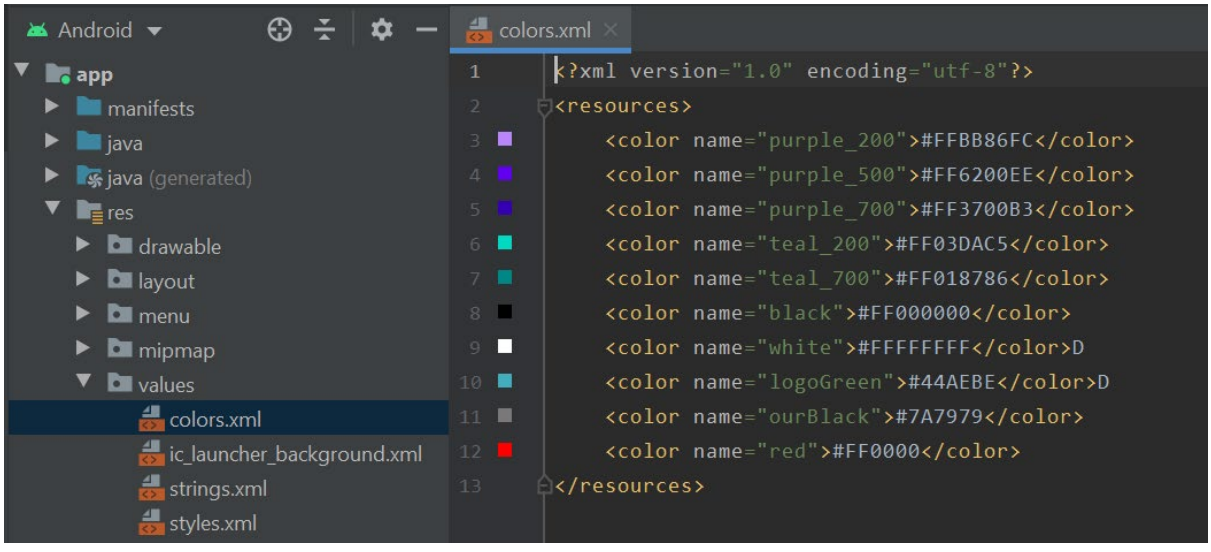


Figure 16: colors.xml file.

Using this method made sure we didn't make mistakes typing out the hex code for each element on the XML files and was much more efficient as we only had to pass the colour reference. Additionally, if we ever wanted to change the colour scheme of the application, we'd only have to change one hex code, instead of going page by page and changing the colour on each individual element.

When coding functionality on the Java files we also made sure that code was indented correctly and we added comments to each component to ensure readability, both for other group members and to make it easier for future development. You can see an example of this in the Sign_In.java page, as shown below. All the individual variables have explanatory comments and each code blocks start and end is clearly marked.

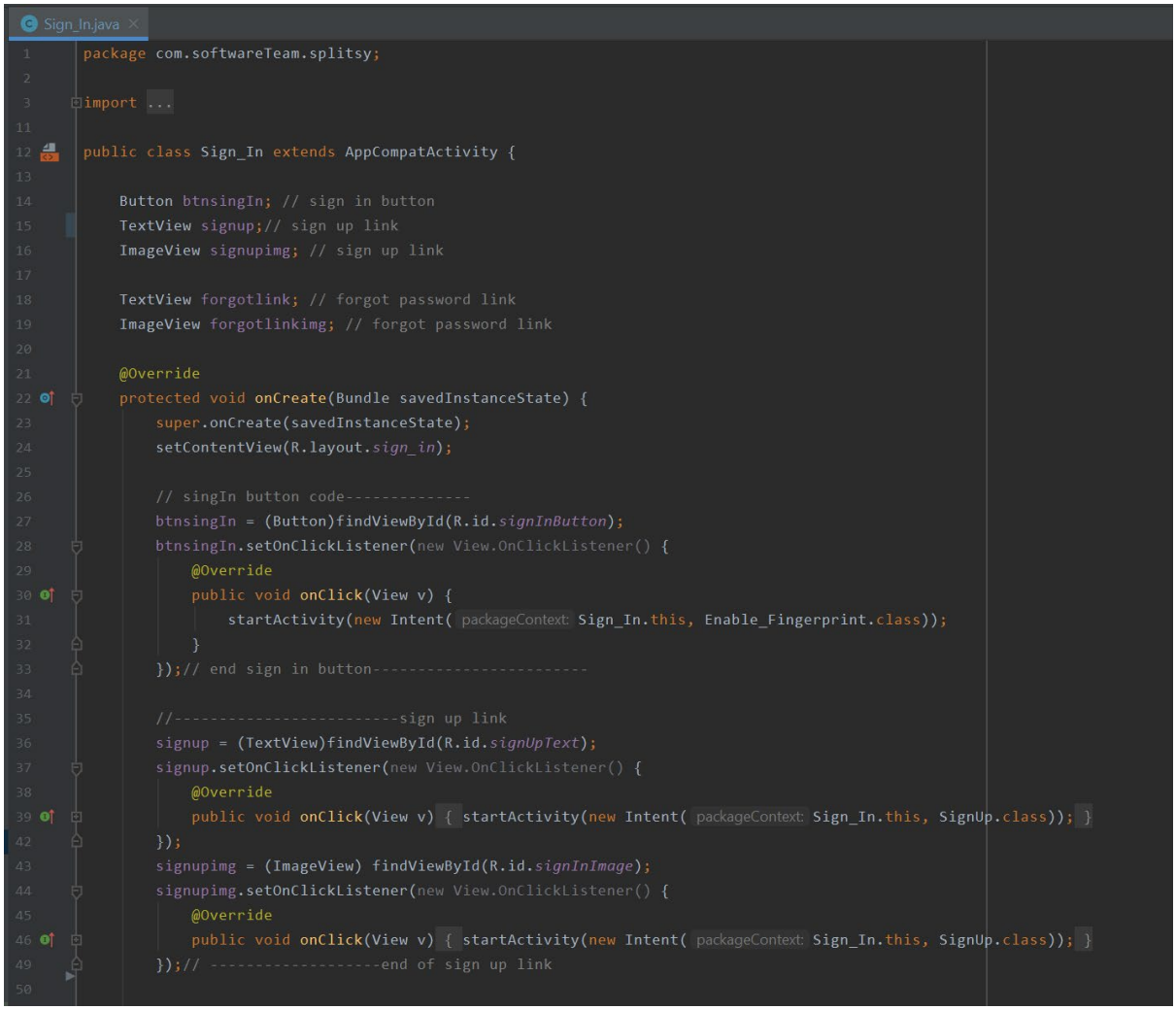


Figure 17: Sign_In.java snippet.

Testing

Testing during coding & Error Handling

As we coded our application, we conducted systematic tests of each functionality we added. We used this type of test-driven development to ensure our implementations were working well and that we were fixing small bugs as they were created rather than allowing them to grow exponentially.

For example, during implementation of the Sign In page, we tested each combination of correct and incorrect email addresses and passwords to be certain that each case reacted as it should. As you can see in figure 18, when we input an incorrect email and a correct password, or a correct email but an incorrect password, the correct response (failure to log in) is given:

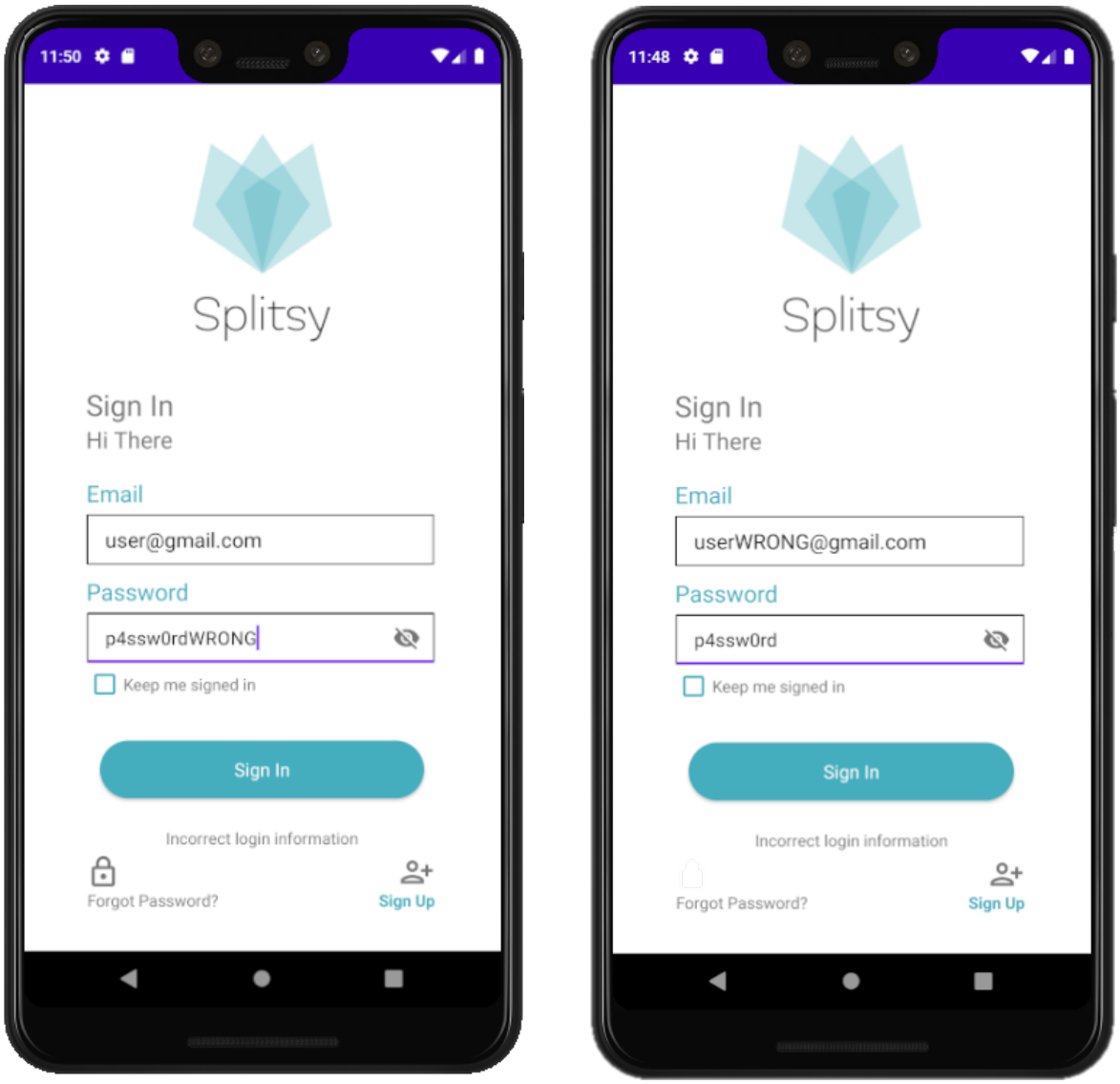


Figure 18: Sign In page error checking.

This approach ensured that our application has effective error handling. We were able to minimise the possibility that the program could reach a state that it is not prepared for, which could lead to crashes or unexpected behaviour. This could happen due to user error, defects in our program or so on. For example, an unforeseen database malfunction could result in our program not being able to retrieve necessary information. In this case, the application should not crash, but should fail gracefully by outputting an error message. This makes the application more pleasant for users, as they encounter less malfunctions and are given appropriate feedback on what went wrong. As usability was the main focus of our project, this was an essential process for us to undertake.

We added new features methodically, testing as we did so to make sure we had not broken or affected previously existing aspects. In this way we could be confident we were making progress on each new section of the development process, without hindering or reversing our progress in other areas. To this end, we used separate Git branches to implement distinct features, then merged carefully to prevent conflicts from existing functionalities.

We tried to implement similar features in a uniform way across different files. For example, when coding our Sign Up page, we referred to our previously developed Sign in page to confirm we were processing user input in the same way. As you can see in figures 19 and 20, both Java files consisted of an Async task that had a try and catch block and an onPostExecute() function:

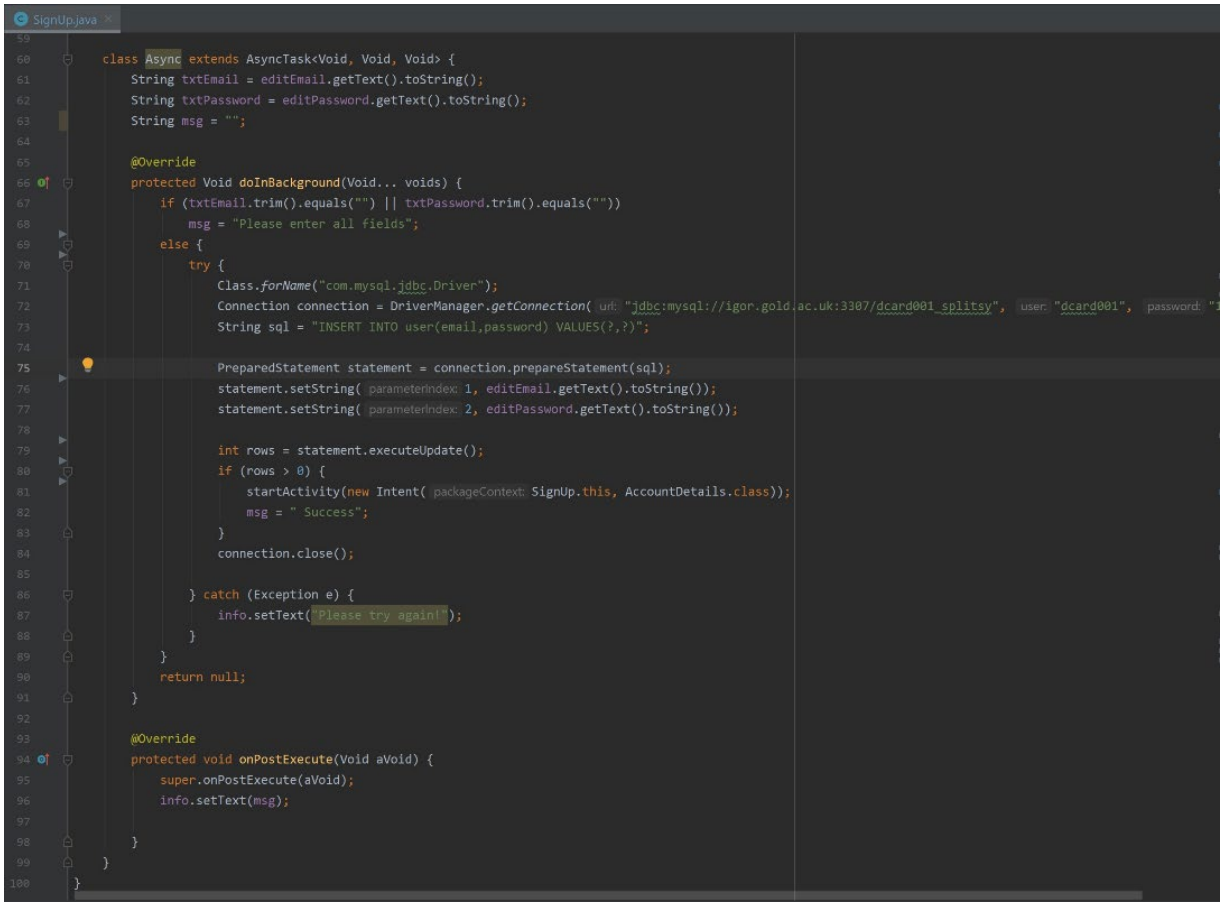


Figure 19: SignUp.java code snippet.

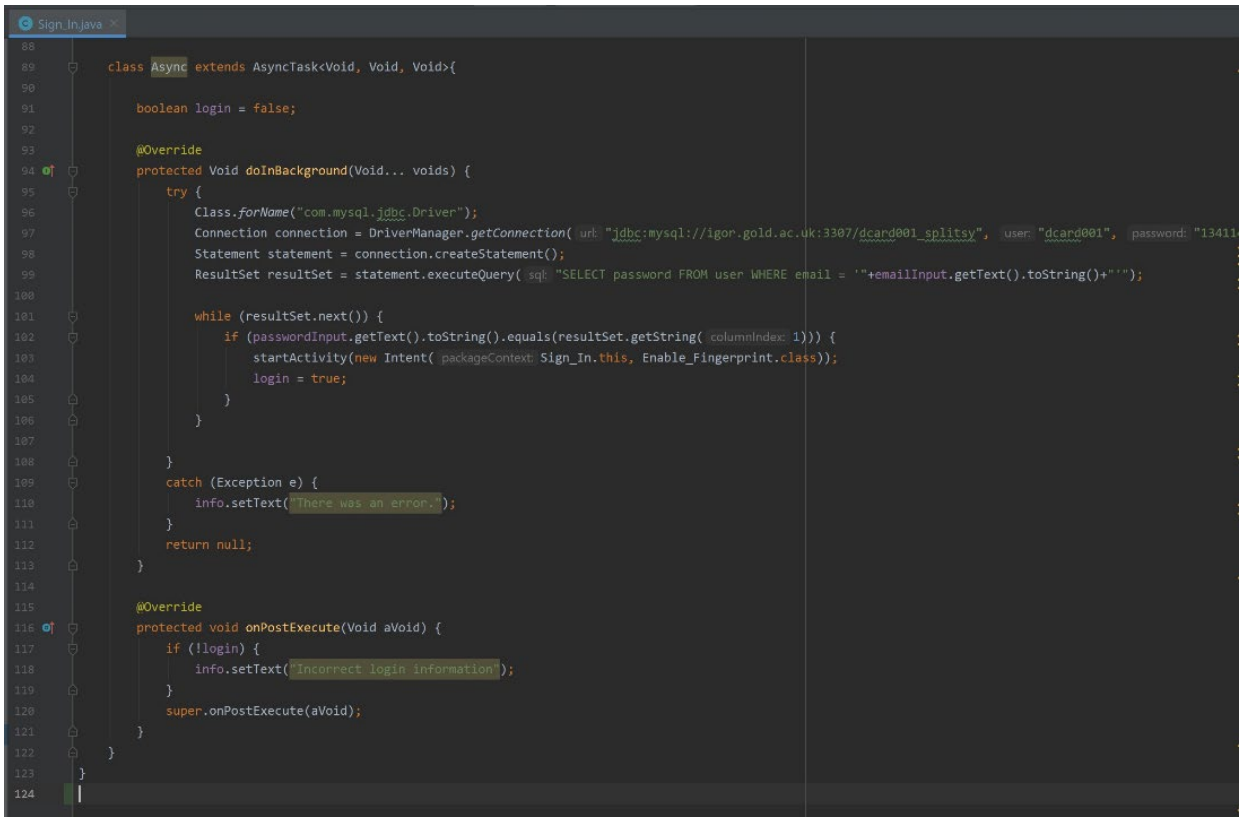


Figure 20: SignIn.java code snippet.

This provided consistency, making it straightforward for group members who weren't involved in the implementation to understand it. Following the same structure between pages meant we were less likely to make mistakes when coding. Additionally, this helped ensure the components of our application combined as expected, as the points of connection were consistent between different pages.

During implementation of the UI, we also made sure to check the XML validation we had added. Examples of this validation includes only allowing digits to be entered for certain fields such as a bank card number, adding a maximum length for text fields that reflect those set in the database, and so on. An example of this can be seen in figure 21.

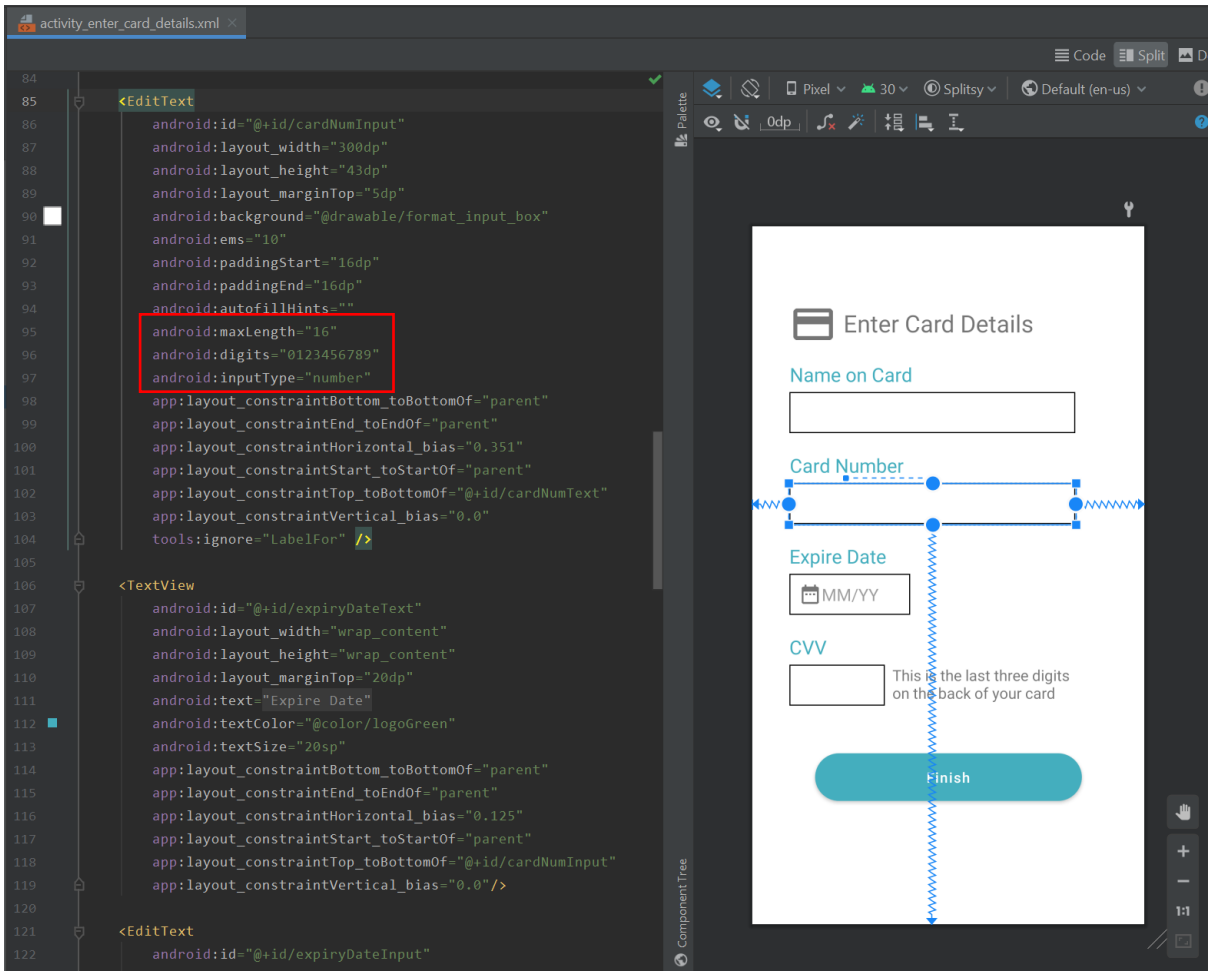


Figure 21: activity_enter_card_details.xml code snippet.

This snippet shows the validation added for the 'Card Number' field on the 'Enter Card Details' XML file. The property 'maxLength' sets the character limit to 16, as this is the international standard for the length of a card number [38]. The property 'digits' ensures that only the digits contained in the following string will be accepted, which is every digit in this example. This ensures that the provided card number will be in the correct format, as users will not be able to enter other characters or extra digits. We additionally set the property input type to number, as this will ensure the correct soft input method is shown, i.e., the keyboard will only consist of numbers [39], so users are directed to use the appropriate input method.

We tested the validation using the emulator. We first began by entering 16 valid digits, which the application accepted, as seen below. We moved on to entering invalid characters, which was not possible, showing us that the validation worked as intended. We then tested the character limit by attempting to enter more than 16 characters. The application did not allow this, showing us that both validation criteria were working correctly.

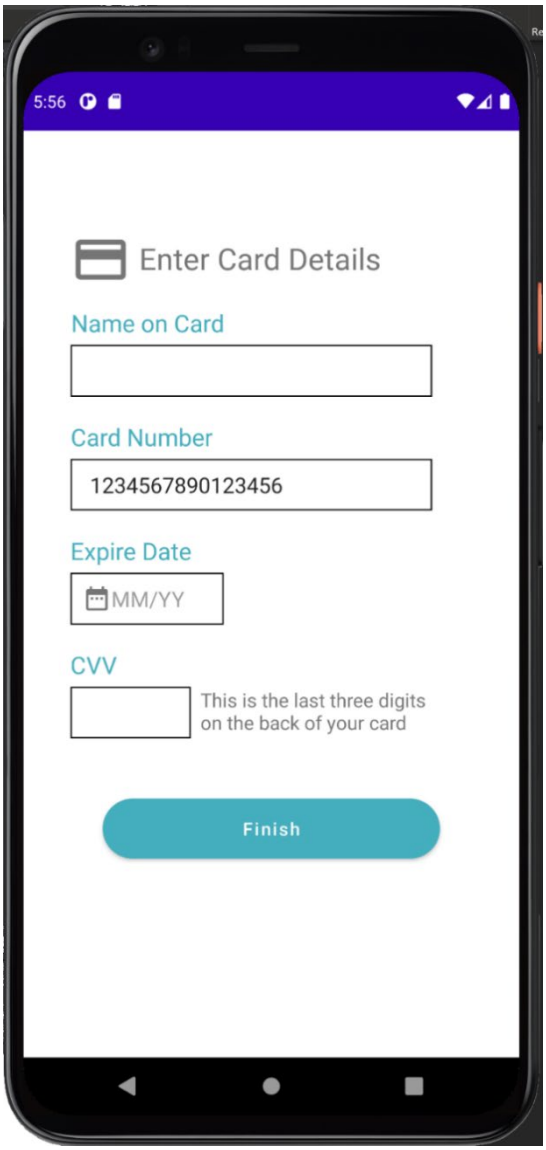


Figure 22: Enter Card Details page validation testing.

We completed this process for every page with user input, which allowed us to be confident that users could not cause the application to enter an error state by entering data what would be rejected by the database. This testing procedure was essential since this was the first time our team had worked with XML, so we wanted to be sure the validation we added would work as intended.

Overall, these processes were very beneficial. They made future test processes more effective as most minor bugs produced were eliminated, allowing us to focus on higher-level issues.

Unit Testing

"Unit testing is a type of software testing where individual units or components of a software are tested to verify its correctness. The purpose is to validate that each unit of the software code performs as expected" [40].

JUnit is the most popular and widely used unit testing framework for Java. JUnit 4 allows us to write tests in a cleaner and more flexible way [41] that allows us to

programmatically simulate user actions. We used JUnit to test our local components such as email and password input data with the Assert method. Assert is a method useful in determining the Pass or Fail status of a test case with the help of org.JUnit.Assert Class [42].

We also used the Mockito framework as it can be used in conjunction with JUnit. Mockito allows us to create and configure mock objects. It helps to run tests more quickly and efficiently [43].

We carried out unit tests using JUnit and Mockito to test our sign in and signup page. For login validation, we tested the components such as email and password. We tested the email address to check that it meets the standard format. We tested password input to check the password field must not be empty. We also tested validation using dummy data with the help of the Mockito framework. It helped us to validate user input with the test data and verify the output.

Case Id		Test component	Expected result	Result	Pass / fail
Junit	1.a	Email input with a valid email	The test method compares the email. The expected result is true	True	Pass
	1.b	Email input with a valid email	The test method will detect the valid email input. The expected result is false	False	Pass
	2.a	Email input with whitespace	The test method is set to false which checks the email input with whitespace. The expected result is false	False	Pass
	2.b	Email input with whitespace	The test method is set to true which checks the email input with whitespace. The expected result is false.	False	Pass
	2.c	Email input with whitespace	The test method checks the email input with whitespace assigning the assert method to true implementing the trim method to remove the whitespace.The expected result is true.	True	Pass
	3.a	Password with a string input	The test method tests null password input.The result is false because the password input is a string.	Unsuccessful	Pass
	3.b	Password with a null input	The test method tests null password input. The result is true because the password input is equal to null.	Successful	Pass
Mockito	4.a	Email and password with a valid data	We compared the user's email and password to validate login using a mock object. The expected result is true because the test data(email and password) along with the mock string matches with the user input data(email and password) and its output result.	True	Pass
	4.b	Email and password with a invalid data	We compared the user's email and password to validate login using a mock object. The expected result is false because the test email along with the mock string does not match with the input email string.	False	Pass
	4.c	Email and password with mock string	We compared the user's email and password to validate login using a mock object. The expected result is false because although the email and the password input matches with the test email and password data, the output of the test does not match with the mock string.	False	Pass

Figure 23: Unit Test table.

JUnit test 1

This type of test verifies that the target application behaves as expected when a user input an email address.

Test 1a: valid email input

We tested the email component so that users can sign in only with a valid email address. The test was successful (see figure 24).

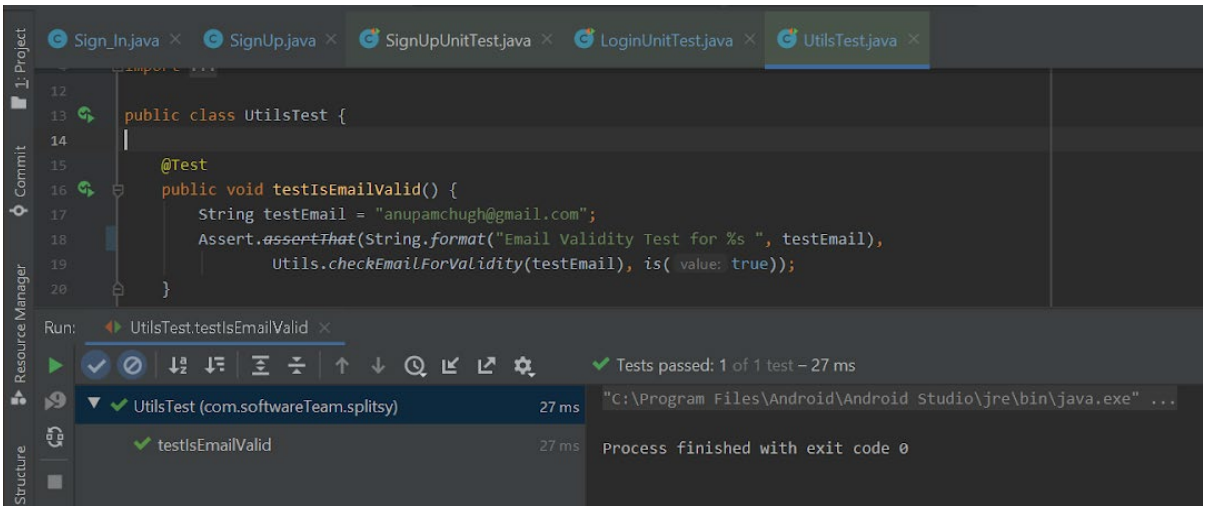


Figure 24: Unit test for valid email input.

Test 1b: invalid email input

We tested the email component with valid input. As a result, the test result was false because we use the **assert false** method. Users need to enter a valid email address to access the application (see figure 25).

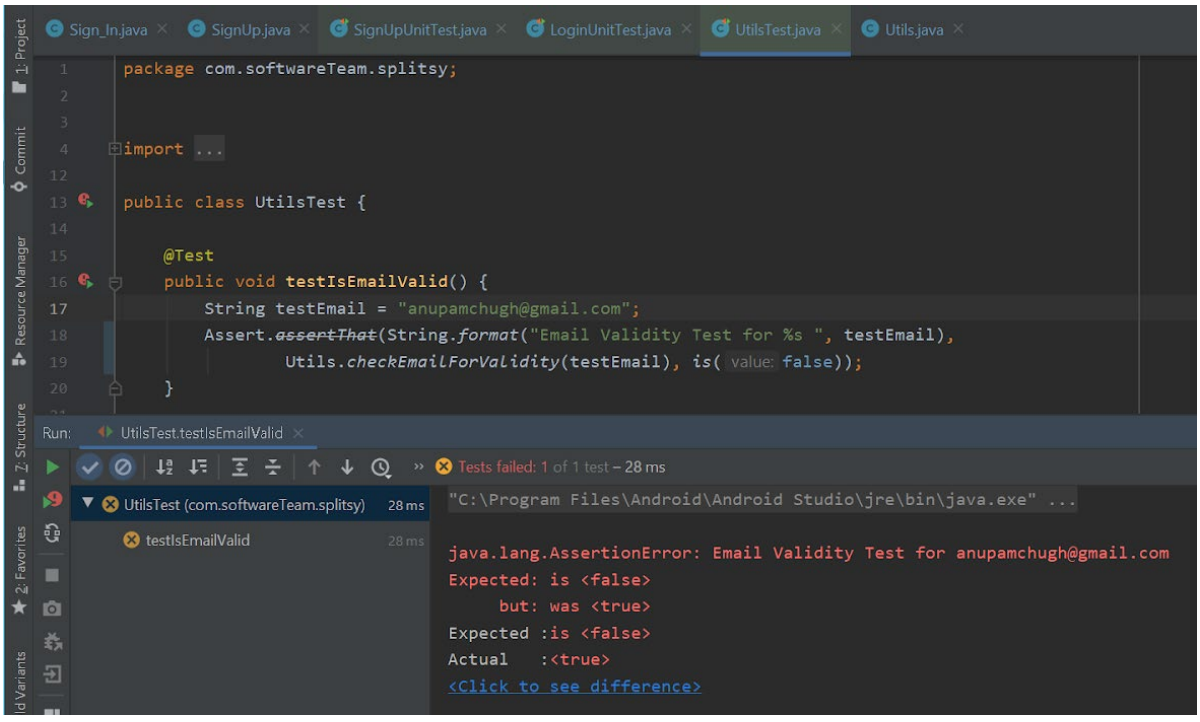


Figure 25: Unit test for invalid email input.

JUnit test 2

The test method *testEmailValidityWhitespace* verifies email input with whitespace.

Test 2a: Email input with whitespace and assertFalse() method

We tested the email input with whitespace assigning the assert method to false. The result for this test was successful because the email input was invalid which matches the assertFalse() method (see figure 26).

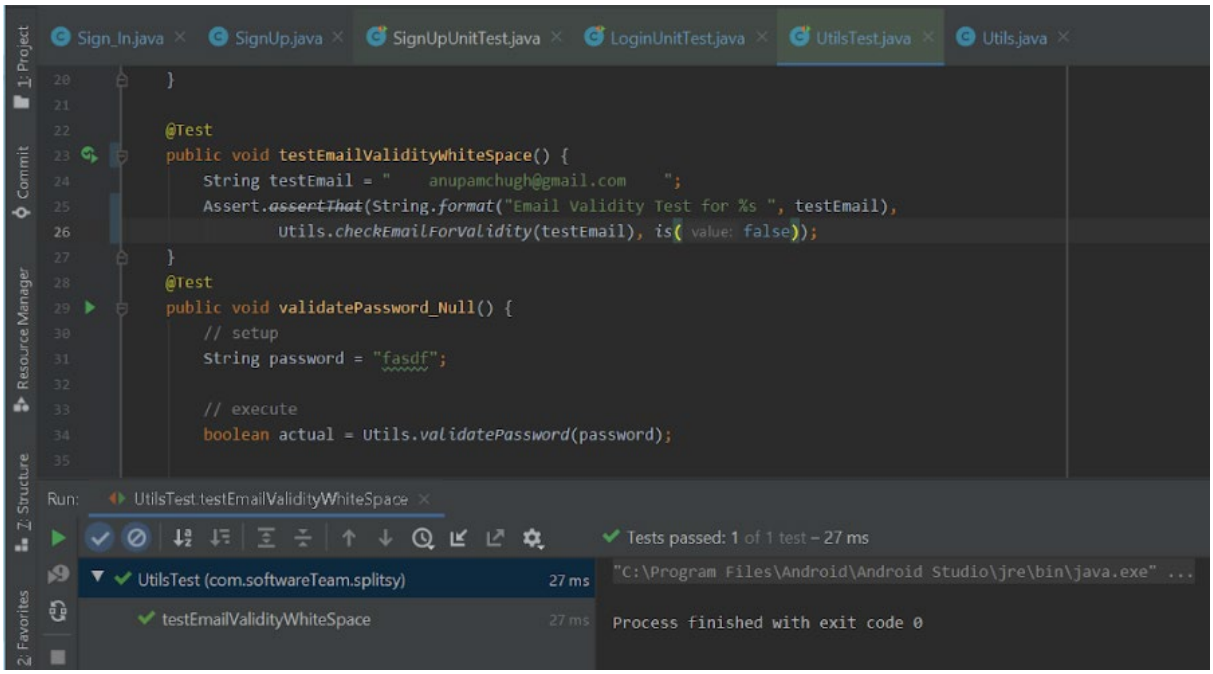


Figure 26: Unit test for email input with whitespace using assert False method.

Test 2b: Email input with whitespace and assertTrue() method

We tested the email input with whitespace assigning the assert method to true. The result for this test was unsuccessful because the email input was invalid which does not match to assertTrue() method (see figure 27).

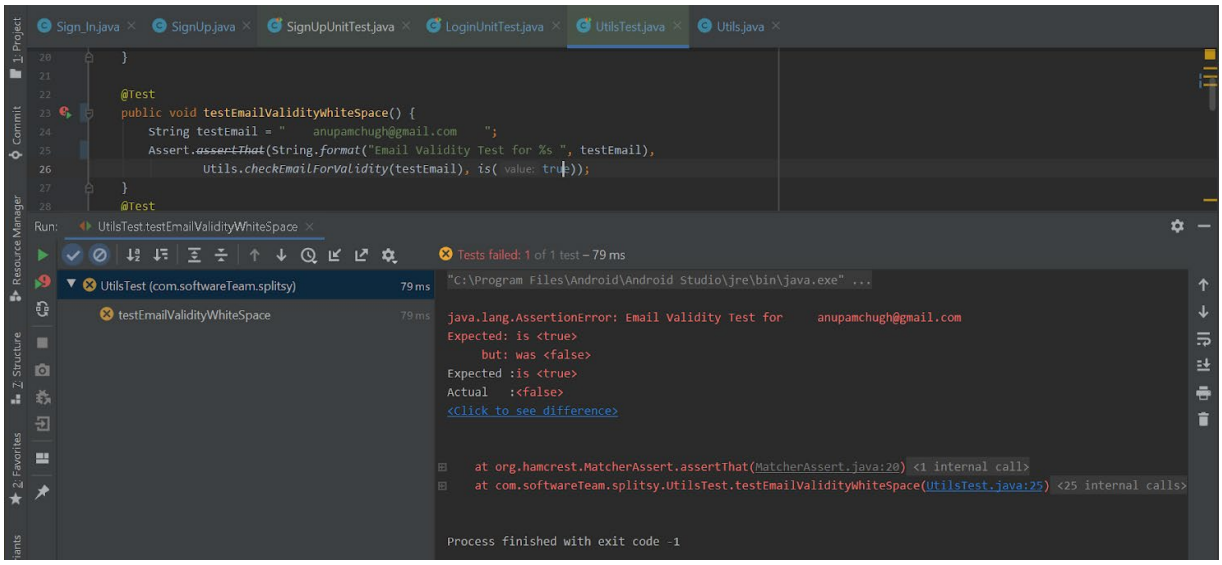


Figure 27: Unit test for email input with whitespace using assert True method.

Test 2c: Email input with whitespace and trim method

We tested the email input with whitespace assigning the assert method to true implementing the trim method to remove the whitespace. The result for this test was successful as all the whitespace was removed by the trim method (see figure 29).

```
public static boolean checkEmailForValidity(String email) {
    email = email.trim();

    Matcher matcher = VALID_EMAIL_ADDRESS_REGEX.matcher(email);
    return matcher.find();

}
```

Figure 28: Unit test for email input with whitespace using trim method.

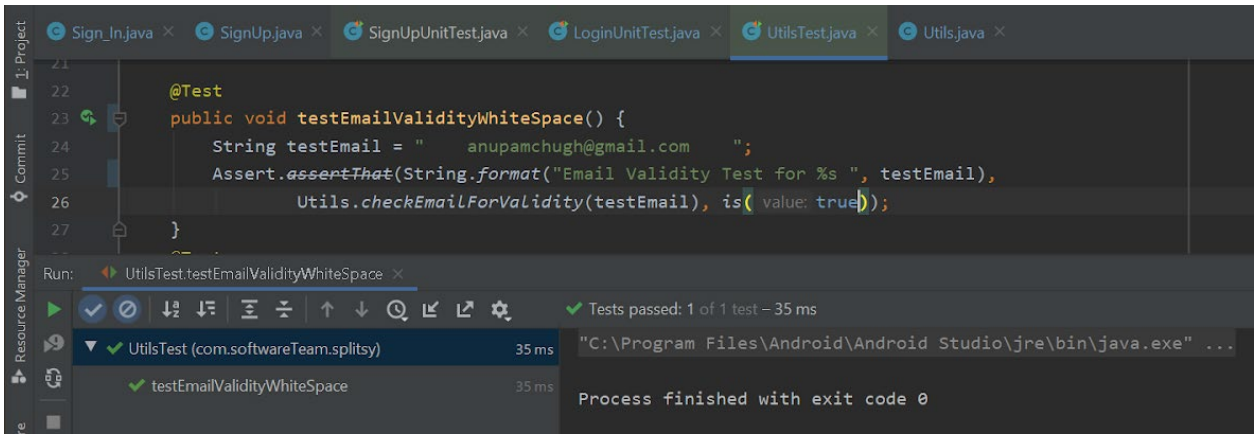


Figure 29: Unit test for email input with whitespace using trim method.

JUnit test 3

The test method validate Password Null verifies the password input.

Test 3a: Password input if it is Null

We tested the password input using a string. The result for this test was unsuccessful because the input is not null (see figure 30).

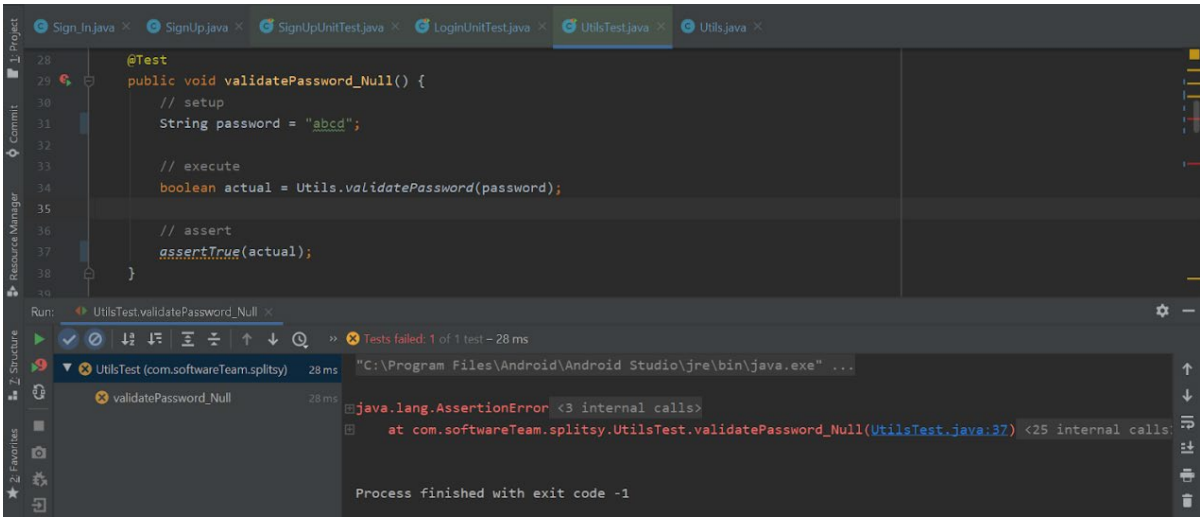


Figure 30: Unit test for null password input using assert True method.

Test 3b Password input if it is Null

We tested the password input assigning it to null. The result for this test was successful because the result matches with the test method (see figure 31).

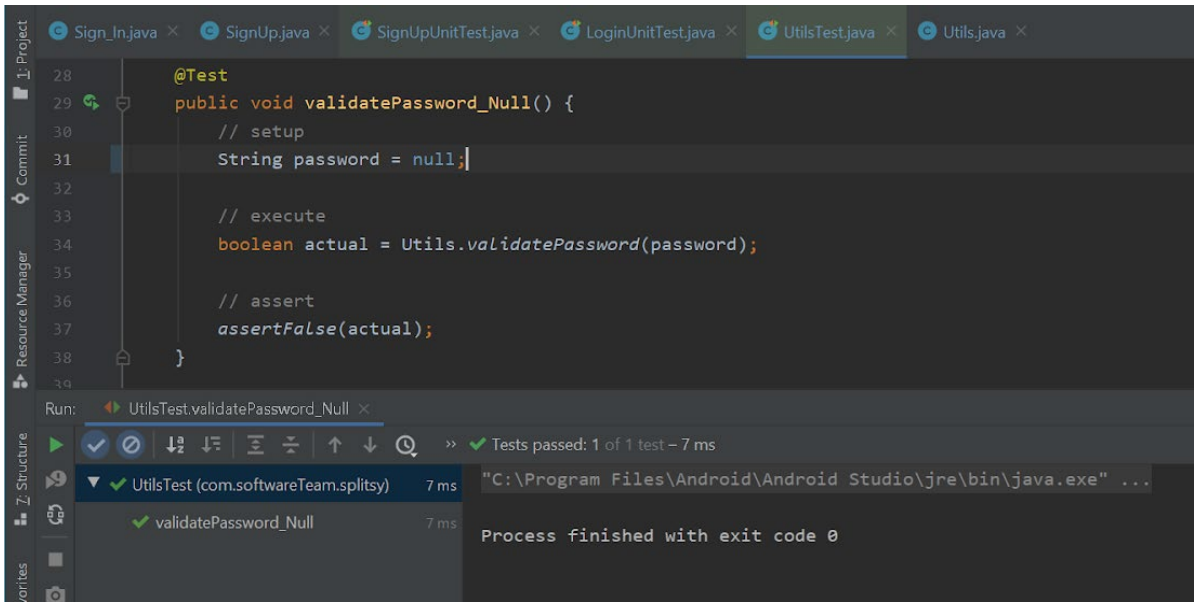


Figure 31: Unit test for null password input using assert False method.

Mockito unit test 4

The test method *readStringFromContext_LocalizedString()* verifies the *validate* method.

Test 4.a Email and password verification with valid test data

We compared the user's email and password to validate login using a mock object. It was successful because the test data (email and password) along with the mock string matches with the user input data (email and password) and its output result (see figure 33).

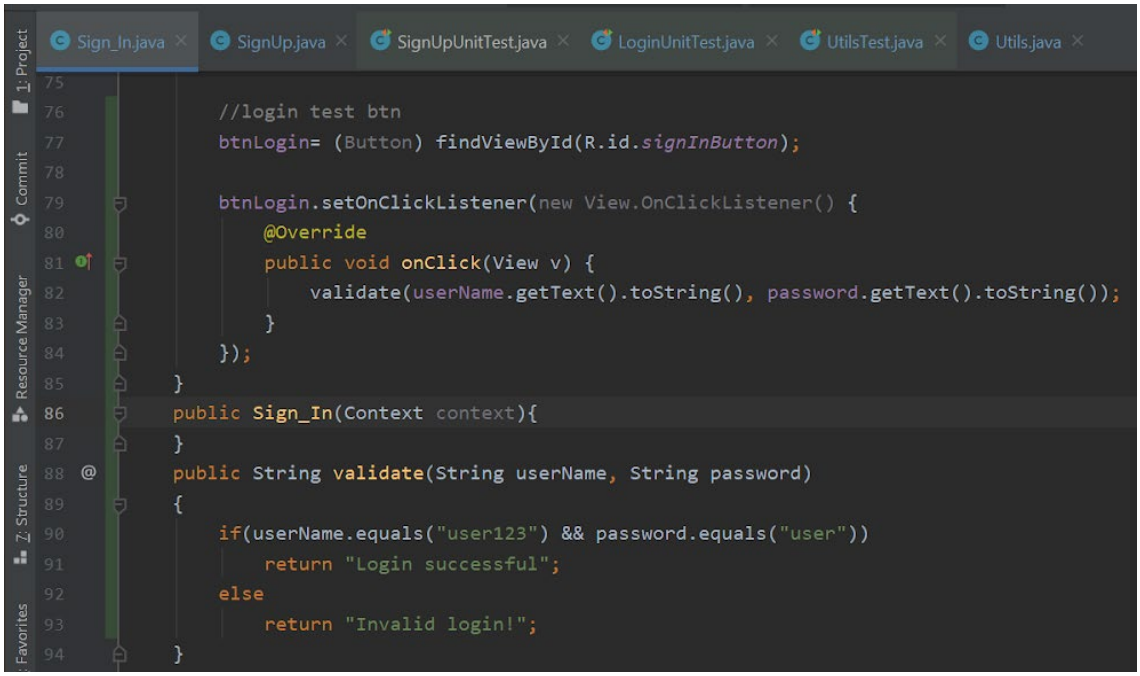


Figure 32: Unit test for email and password verification using Mockito with valid data.

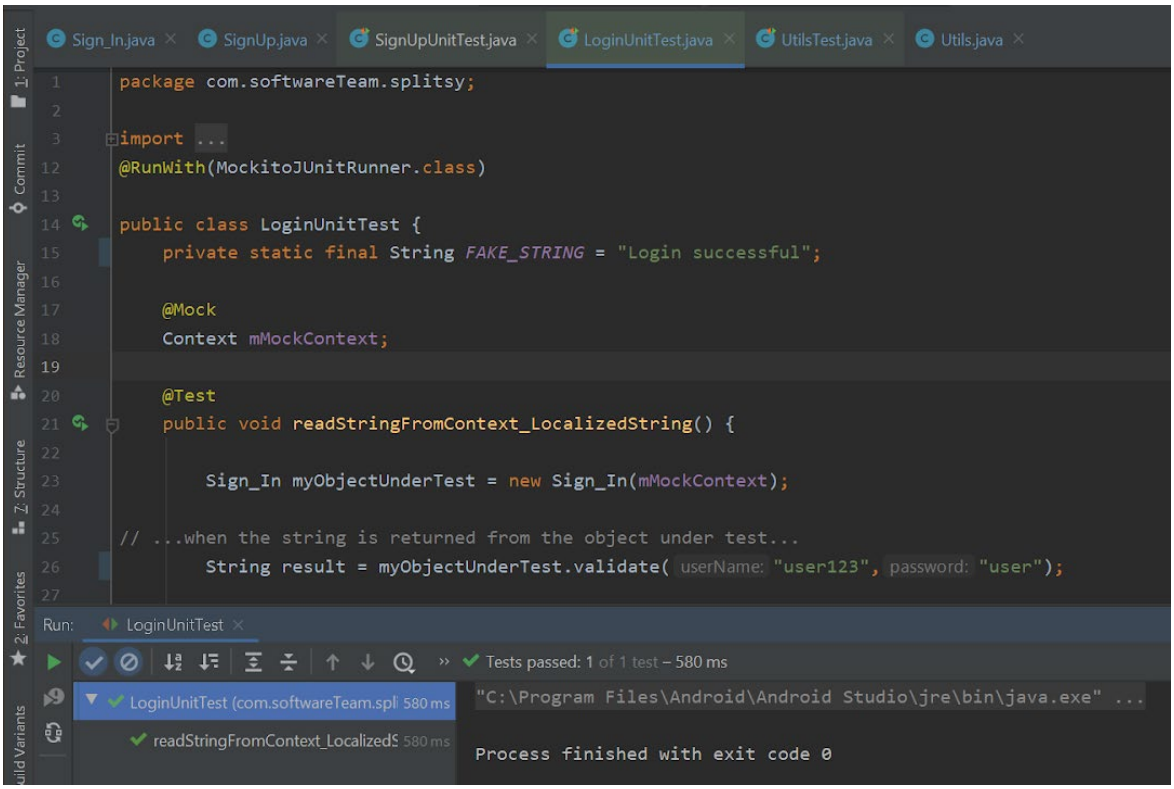


Figure 33: Unit test for email and password verification using Mockito with valid data.

Test 4.b Email and password verification with invalid test data

We compared the user's email and password to validate login using a mock object. It was unsuccessful because the test email along with the mock string does not match with the input email address (see figure 35).

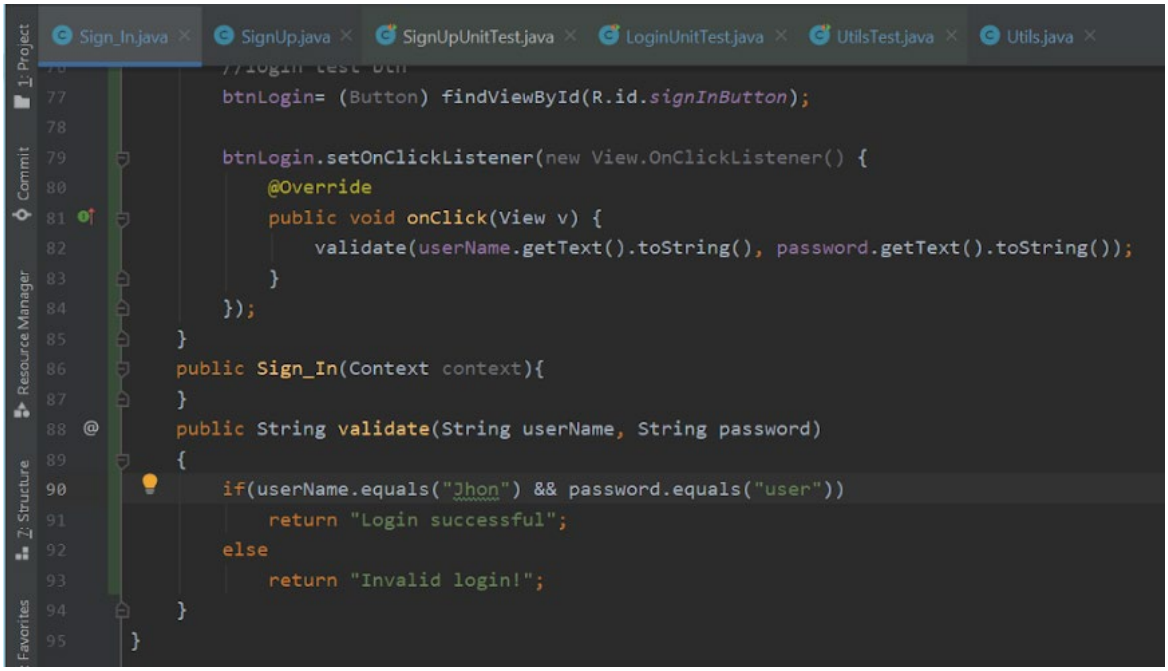


Figure 34: Unit test for email and password verification using Mockito with invalid data.

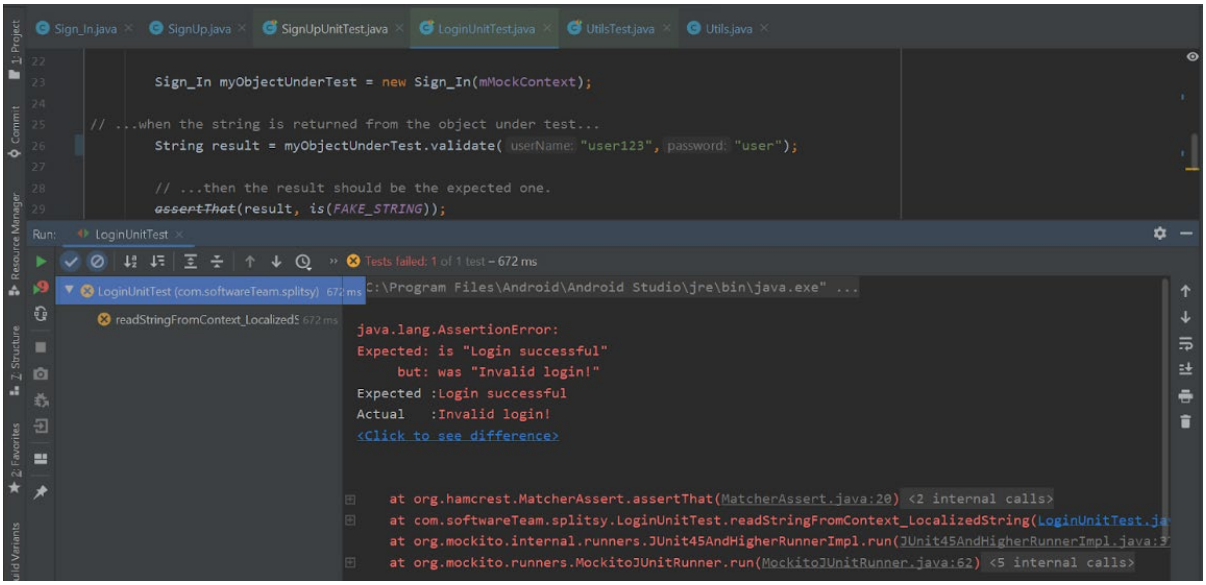


Figure 35: Unit test for email and password verification using Mockito with invalid data.

Test 4.c Sign in verification with Mockito string

This test was unsuccessful because the email and the password input matches with the test email and password data, but the output of the test does not match with the mock string (see figure 37).

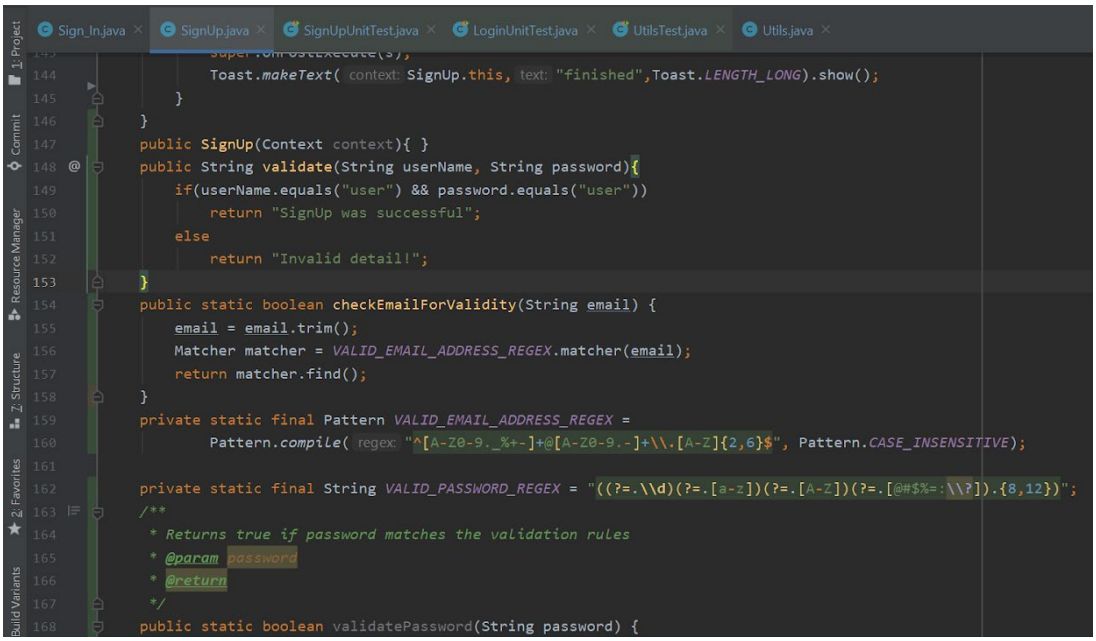


Figure 36: Unit test for sign in verification using Mockito string.

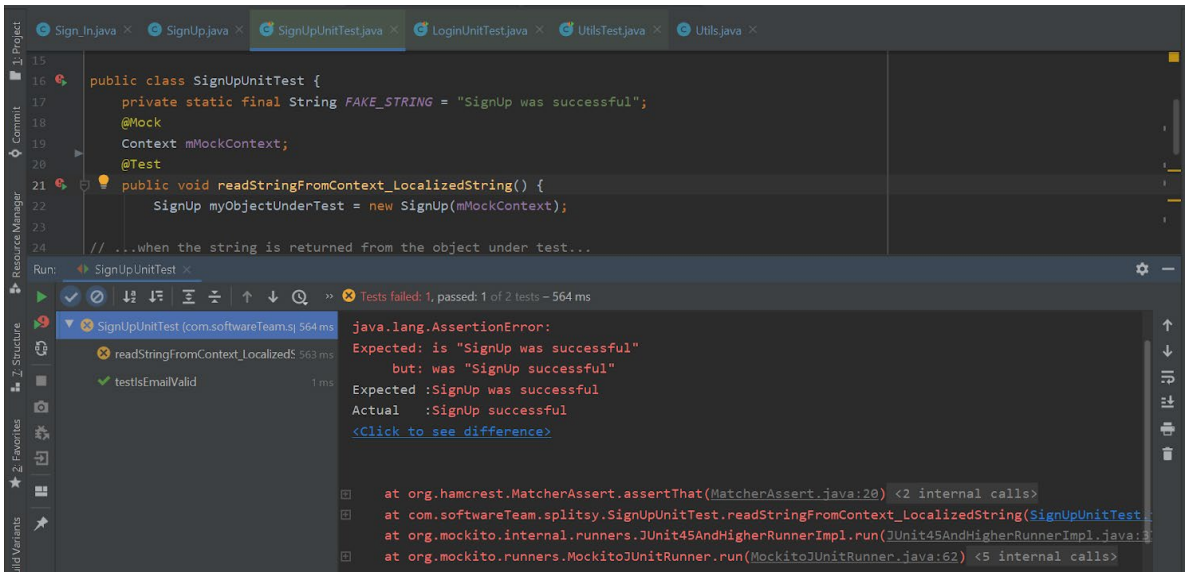


Figure 37: Unit test for sign in verification using Mockito string.

The unit testing demonstrated that our current product had fulfilled the minimal viable product's login validation functional requirements. We tested all possible user inputs on our login page. For the email input, we tested the email pattern and presence of whitespace. For the password input, we tested if the input field is empty or filled with text. Overall, our test methods showed expected results for each particular component. We managed to register users and save data to the database, but we did not implement email validation on our signup page. We will continue to improve login validation using the sanitisation method for our input data and hashing method for password. With the implementation of new activities and features, unit tests should continue effectively.

Integration Testing

Integration testing is the next level of product testing, following on from unit-level tests. At this stage, the units that were tested individually are combined and assessed as a group. These tests aim to verify that the modules work as intended when they are integrated together. Integration testing is a form of black box testing – the internal working of the software is not considered, only the input and output. In a larger project, integration testing would usually be performed by a dedicated testing team, rather than by the developers that wrote the code. As we are such a small team, we do not have members who don't write code, but we will disregard the inner workings of our code for the purposes of these tests.

There are different approaches that can be taken in regard to integration testing. These are the Big Bang approach, and the Incremental approach [44]. In the Big Bang approach, all modules are developed first and then tested together. In the Incremental approach, two or more related modules are combined and tested as soon as they have been developed. As more modules are completed, they are integrated in with the existing modules until all units have been added.

The Big Bang approach is suitable for smaller systems but has several disadvantages. It can be tough to troubleshoot the cause of problems as the whole system is tested at once, and testing cannot begin until all modules have been developed. In contrast, the Incremental approach can make it easier to locate the source of problems, as a smaller group of modules are tested at one time. Testing can also begin sooner, although testing may take longer overall with this method.

The Incremental approach can be further split into the Top-down, Bottom-up, and Sandwich approaches [44]. These sub-approaches define the order in which the modules are built and integrated in. In the Top-down approach, the top-most modules are tested first, and lower modules later. In the Bottom-up approach, the lowest modules are tested first, and the higher modules later. The Sandwich approach is a combination of both methods, used to approach a target layer from both directions.

The Top-down approach concentrates on building and testing the highest-level modules of the product first – the modules that all lower units will be connected by. This can lend a greater understanding of the system itself, and essential modules are tested early on, but basic user functionality is not tested until later. The Bottom-up approach focuses on building and testing the lowest-level modules first. This allows an early focus on testing user-facing modules but can lead to difficulties when connecting the components together later. The Sandwich approach attempts to gain the benefits of both methods while minimising their limitations. This can make it useful for very large projects, which the other methods may be less suitable for. However, this method is very costly in terms of time and effort required.

We chose to take an Incremental approach to our integrated testing. As we are using an Agile work methodology, this approach reflected our plan to build the project iteratively to best make use of our limited timeframe. One of the weaknesses of Big Bang integration testing is that there is a large delay before the testing itself can begin [44] - this in particular was something we wanted to

avoid. The sub-approach we used was the Bottom-up approach. From the market research we performed for our project proposal, we knew that the most common criticism of competing apps was that they are too complicated. The Bottom-up approach allowed us to focus on usability from the beginning of the project and perform user testing at the earliest possible time.

In this table we will test the interactions between the components that allow the user to create an account, sign in, and then sign out again. We will be testing each of the steps that must be taken to achieve this, as well as some optional steps that the user may take, but are not essential. Non-essential tests are towards the end. The user begins on the Sign In page.

Test Case ID	Test Case Objective	Test Case Description	Expected Result	Result	Pass / Fail	Comments
001	Navigate to Sign Up page.	Click "Sign Up" text or accompanying icon.	Application navigates to Sign Up page.	As expected.	Pass	
002	Create an account from Sign Up page.	Select Email box and enter user's email (usertest@gmail.com). Select Password box and user's password (Password123). Check the box to agree to ToS. Press the Continue button.	Application navigates back to the Sign In page.	As expected.	Pass	
003	Sign In using new account details.	Select Email box and enter user's email (usertest@gmail.com). Select Password box and enter user's password (Password123). Press the Sign In button.	Application navigates to Enable Fingerprint page.	As expected.	Pass	
004	Enable fingerprint sign in.	Check the permission box and press the fingerprint icon.	Application navigates to Fingerprint Scan page. User scans fingerprint. Application navigates to New Bill page.	Nothing happens upon clicking fingerprint icon.	Fail	This feature requires the fingerprint API which is not yet implemented.
005	Log Out user.	Press Account icon on the bottom navigation bar. Then press "LOG OUT" text.	Application navigates to the Sign In page.	As expected.	Pass	
006	Sign In using incorrect account details	On Sign In page, select Email box and enter an incorrect email. Select Password box and enter an incorrect password. Press the Sign In button.	Application does not change page. Displays "Incorrect login information" text.	As expected.	Pass	
007	Navigate from Sign Up page back to Sign In.	On Sign Up page, press "Sign In" text.	Application navigates back to Sign In page	As expected.	Pass	
008	Deny fingerprint sign in.	On Enable Fingerprint page, press "Deny".	Application navigates to New Bill page.	As expected.	Pass	
009	Navigate to Forgot Password page.	On Sign In page, press "Forgot Password?" text or accompanying icon.	Application navigates to Forgot Password page.	As expected.	Pass	
010	Use Forgotten Password function.	Select email box and enter user's email. Press Submit button.	Application navigates to Email Sent page.	As expected.	Pass	
011	Return to homepage from Email Sent page.	On Email Sent page, press the Home button.	Application navigates back to Sign In page.	As expected.	Pass	

Figure 38: Sign up, login and logout testing results.

In the next table, we will test the interactions between the components that allow the user to view their Past Bills. We will begin on the New Bill page, having just signed in. To reach this point perform test 003, followed by test 008.

Test Case ID	Test Case Objective	Test Case Description	Expected Result	Result	Pass / Fail	Comments
101	Navigate to Past Bills page.	Press Past Bills icon on the bottom navigation bar.	Application navigates to the Past Bills page.	As expected.	Pass	
102	Open Filter/Sort menu.	Press the purple filter icon in the top right of the page.	Filter/sort menu appears in the centre. The page behind darkens.	As expected	Pass	
103	Select a filter or sort to apply.	Press the empty circle next to the desired filter or sort.	The selected circle becomes filled in. The data displayed on the past bills is altered to match the selection.	The selected circle becomes filled in. However,	Fail	The filter functionality is not yet implemented.
104	Close the Filter/Sort menu.	Press outside the menu, on the darkened Past Bills page behind.	Filter/Sort menu disappears. The Past Bills page returns to normal brightness.	As expected.	Pass	
105	View Bill Details.	Press the name of a bill.	Application navigates to the Bill Details.	Nothing happens.	Fail	The Bill Details functionality is not yet implemented.
106	Navigate from Past Bills page to New Bill page.	Press New Bill icon on the bottom navigation bar.	Application navigates to the New Bill page.	As expected.	Pass	
107	Navigate from Past Bills page to Account page.	Press Account icon on the bottom navigation bar.	Application navigates to the New Bill page.	As expected.	Pass	

Figure 39: View past bills testing results.

In the next table, we will test the interactions between the components that allow the user to make changes to their account information. As before, we will begin on the New Bill page, having just signed in. To reach this point perform test 003, followed by test 008.

Test Case ID	Test Case Objective	Test Case Description	Expected Result	Result	Pass / Fail	Comments
201	Change registered email	Select Account page icon from bottom navigation bar. Select “Change Email” text. Application will navigate to Change Email page. Select New Email box and enter new user’s email (usertest123@gmail.com). Select Confirm Email box and enter new user’s email (usertest123@gmail.com). Press the Continue button.	Application will navigate to Email Changed confirmation page. Press Continue Button. Application will navigate back to Account page.	As expected	Pass	
202	Change password	Select “Change password” text. Application will navigate to Change password page. Select Current password and enter the current password. Select New Password box and enter new password (Password1234). Select Confirm New Password box and enter new password (Password1234). Press the Continue button.	Application will navigate to Password Changed confirmation page. Press Continue Button. Application will navigate back to Account page.		Fail	
203	Edit card	Select “Edit Card” text. Application will navigate to Edit card page. Select Expiry Date box and enter new Expiry Date (09/26). Press the Continue button.	Application will navigate to Edited Card confirmation page. Press Continue Button. Application will navigate back to Account page.	Application does not have a Finish Button. Cannot continue the test.	Fail	Instructions do not reflect the options available on the application.
204	Change nickname	Select the pencil icon. Select “Nickname” box and type your new nickname.	Nickname inside Nickname box will be updated.	Nothing happened when the pencil icon was selected.	Fail	

Figure 40: Change account details testing results.

As you can see, our application passed the vast majority of the integration tests. We believe this shows that the modules we have created so far work well together. Where our application has failed tests, in most cases this is due to a component having not yet been implemented. These failures will naturally be resolved as further units are developed and integrated. While we could have avoided testing the modules that have not yet been added, we felt it was important to test every

feature and link we had. This meant we could see clearly why some links are not yet working and ensured we had not overlooked anything.

In one case, test 203, we found a test was failed due to the test instructions not matching the available options presented on the application. This was useful as it made us assess why we had designed the test in this way, and how the application differed to our expectations. Using this information, we could judge if the test was at fault or if the application itself needed changing. These observations could then inform future development and testing.

Through integration testing, we reassured ourselves that our implemented modules work together appropriately and intuitively. The black box nature of our tests helped us cover a large amount of code, while giving straightforward results that make it easy to detect the existence of errors. This testing level was of particular importance, as our project is a complex piece of software being built simultaneously by multiple people.

UI Testing

After finishing the user interface development and implementing sign in functionality on the first page, we started user interface testing.

There were many reasons we decided to conduct UI testing; firstly, it ensures application usability, which is the most important aspect of application development since the only way users of this application can interact with it is through the graphical user interface. A malfunctioning UI would be a massive obstacle for users [45], hence it needed to be tested thoroughly. Secondly, the UI of is fundamental since it's what the user sees; visual components not behaving correctly may influence users negatively and cause them to switch to a competitors app.

We checked two different things: how the application responded to user action performed via the touch screen and whether the visual elements were displayed correctly and work as intended. Many of the visual elements were tested during development, however as we had multiple people working on the XML files, we decided to test this aspect again to ensure consistency throughout the pages. We further broke down the elements that needed to be tested into 5 different sections: consistency, spelling, typography, behaviour of interactive elements and adaptability.

We decided to conduct this testing manually. Automated tests ignore everything other than the specific aspect they are testing, meaning they could miss other errors on pages that we would notice when doing the tests manually. This would allow our tests to be much more extensive and through.

To conduct this testing, we used an Android phone with a 6-inch screen and 1440x3040 resolution, since this is the phone size we earlier found that most Android users have. We wanted to test the UI on a high-resolution phone screen to ensure the images and text would not be pixilated when viewed on a high-resolution screen.

We decided that testing all the pages would not be a good use of our limited time, and instead tested the main pages of the applications that users wound spend most of their time on. Here you can see our testing logs:

Members: brai001, dcard001, ddoch001, msous001, ypaks001

Page Name	Type	Test ID	Expected Behaviour	Steps	Actual Behaviour	Result	Notes	Todo
Sign In	Consistency	1.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Colours, font style and sizes are consistent.	Pass		
	Spelling	1.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	1.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
		1.4	Links are recognisable.	1. Check that the 'Forgot Password' link is distinguisable from the text. 2. Check that the 'Sign In' link is distinguisable from the text	Links are recognisable.	Pass		
	Behaviour	1.5	Sign in button goes to the 'New Bill' page when a valid login is entered.	1. Enter valid login details. 2. Click 'Sign In'	Sign in button goes to the 'New Bill' page when a valid login is entered.	Pass		
		1.6	Sign in button displays an error message when a invalid login is entered.	1. Enter invalid login details. 2. Click 'Sign In'	Sign in button displays an error message when a invalid login is entered.	Pass	Error message too general; could be 'incorrect email' or 'incorrect password' to give users more information.	Make the error messages more informative.
		1.7	Forgot Password button goes to the Forgot Password page.	1. Click 'Forgot Password'.	Forgot Password button goes to the Forgot Password page.	Pass		
		1.8	Sign Up button goes to the Sign Up page.	1. Click 'Sign Up'.	Sign Up button goes to the Sign Up page.	Pass		
		1.9	Toggle button on password field switches password visibility.	1. Enter password. Click the eye icon on the password field	Toggle button on password field switches password visibility.	Pass		
	Adaptability	1.10	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any problems.	Pass	Viewed design on smaller phone size on Android Studio.	
Sign Up	Consistency	2.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Colours, font style and sizes are consistent.	Pass		
	Spelling	2.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	2.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
		2.4	Links are recognisable.	1. Check that the 'Sign In' link is distinguisable from the text.	Links are recognisable.	Pass		
	Behaviour	2.5	Continue buttons goes to the 'Account Details' page.	1. Enter new email and password. 2. Click 'Continue'	Continue buttons goes to the 'Account Details' page.	Pass		
		2.6	Sign In link goes to the 'Sign In' page.	1. Click 'Sign In'.	Sign In link goes to the 'Sign In' page.	Pass		
		2.7	Toggle button on password field switches password visibility.	1. Enter password. 2. Click the eye incon on the password field	Toggle button on password field switches password visibility.	Pass		
	Adaptability	2.8	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design should adapt to being on a 5 inch screen without any problems.	Pass	Viewed design on smaller phone size on Android Studio Emulator.	
Account	Consistency	3.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Colours, font style and sizes are consistent.	Pass		
	Spelling	3.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	3.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
		3.4	Links are recognisable.	1. Check that the 'Change Email' link is distinguisable from the text. 2. Check that the 'Change Password' link is distinguisable from the text 3. Check that the 'Add new card' link is distinguisable from the text 4. Check that the 'Edit card details' link is distinguisable from the text	Links are recognisable.	Pass		
	Behaviour	3.5	Nickname field displays the users username.	1. Check that the nickname field displays the nickname you set when registering.	Nickname field is empty.	Fail		
		3.6	Edit button on the nickname field allows the user to edit their username.	1. Click the edit icon on the nickname field.	Edit button doesn't respond.	Fail		
		3.7	Email field displays the users email address.	1. Check that the email field displays the email you registered with.	Email field is empty.	Fail		
		3.8	Change email link takes the user to the 'Change Email' page.	!. Click the 'Change Email' link.	Change email link takes the user to the 'Change Email' page.	Pass		
		3.9	Password field displays the users password.	1. Check that the password field displays your accounts password.	Password field is empty.	Fail		
		3.10	Toggle button on password field switches password visibility.	!. Click the eye icon on the password field.	Button doesn't respond.	Fail	Toggel password itself works, but the data isn't retrieved.	
		3.11	Change password link takes the user to the 'Change Password' page.	1. Click the 'Change password' link under the password field.	Change password link takes the user to the 'Change Password' page.	Pass		
		3.12	Card details field displays the users card number.	1. Check that the card number field displays the card number you added when registering.	Card details field is empty.	Fail		
		3.13	Add new card links takes the user to the 'Add new card' page.	1. Click the 'Add new card' link under the card details field.	Add new card link takes the user to the 'Add new card' page.	Pass		
		3.14	Edit card button takes the user to the 'Edit Card' page.	1. Click the 'Edit Card' link under the card details fields.	Edit card button takes the user to the 'Edit Card' page.	Pass		
		3.15	Logout button logs the user out and goes back to the 'Sign In' page.	1. Click the 'Logout' link.	Logout button logs the user out and goes back to the 'Sign In' page.	Pass		
		3.16	Settings icon takes the user to the 'Settings' page.	1. Click the settings icon on the right hand corner.	Settings icon takes the user to the 'Settings' page.	Pass		
	Adaptability	3.17	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any	Pass	Viewed design on smaller phone size on Android Studio	

Figure 41: UI testing logs part 1.

Members: brai001, dcard001, ddoch001, msous001, ypaks001

Page Name	Type	Test ID	Expected Behaviour	Steps	Actual Behaviour	Result	Notes	Todo
Settings	Consistency	4.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Design is not consistent.	Fail	'High contrast mode' button is not in line with the other buttons.	Change high contract mode text and toggle position.
	Spelling	4.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	4.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
	Behaviour	4.4	High contract mode toggel switches the application to a high contract color scheme.	1. Switch the 'High Contrast Mode' toggle on.	Toggel switches but colours do not change.	Fail		
		4.5	Dark mode toggel switches the application to a dark colour scheme.	1. Switch the 'Dark Mode' toggel on.	Toggel switches but colours do not change.	Fail		
		4.6	Fingerprint access toggel allows the user to log in with their fingerprint instead of their password.	1. Click the 'Fingerprint Access' toggel on. 2. Close the app 3. Log back in using your fingerprint	Toggel switches but login doesn't allow user to use their fingerprint.	Fail		
		4.7	Notifications toggel stops/starts notifications for joined/created bills.	1. Switch the 'Notifications' toggel on.	Toggle switches but no notifications are received.	Fail	This behaviour has not yet been implemented.	
		4.8	Export data button exports all user data and downloads it to their device.	1. Click the 'Export data' icon.	Button does not respond.	Fail		
	Adaptability	4.9	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any problems.	Pass	Viewed design on smaller phone size on Android Studio Emulator.	
Past Bills	Consistency	5.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Design is not consitant.	Fail	Filter icon colour is not consistent with the rest of the UI.	Change colour of filter icon.
	Spelling	5.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	5.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
	Behaviour	5.4	Users past bills are displayed in status and date order; pending bills at the top and complete underneath.	1. Check that the bills are organised depending on status and date; pending first and then date order.	Past bills are not displayed in a particular order.	Fail	This behaviour has not yet been implemented.	
		5.5	Clicking on a bill takes you to a new page with all the bill details.	1. Click on the name of one of the bills.	There is no response.	Fail		
		5.6	Clicking on the filter icon opens the filter window.	1. Click on the filter icon.	Clicking on the filter icon opens the filter window.	Pass		
		5.7	Adding a filter reorders the bills accordingly.	1. Click on the filter icon. 2. Select a filter. 3. Click the close button.	Past bills are not reordered in any way.	Fail	This behaviour has not yet been implemented.	Implement filter functionalities.
		5.8	Clicking outside the filter window should close it.	1. Click on the filter icon. 2. Click on the grey area outside the filter window.	Clicking outside the window closes it.	Pass		
	Adaptability	5.9	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any problems.		Viewed design on smaller phone size on Android Studio Emulator.	
Create New Bill	Consistency	6.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Colours, font style and sizes are consistent.	Pass		
	Spelling	6.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	6.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
	Behaviour	6.4	'Bill Name' field accepts input.	1. Click the 'Bill Name' field. 2. Enter some text.	'Bill Name' field accepts input.	Pass		
		6.5	'Pay By Day' field accepts a date only.	1. Click the 'Pay By Day' field. 2. Enter a date in the format on the screen.	'Pay By Day' field accepts a date only.	Pass		
		6.6	'Number of member to join' accepts a number between 1 and 50.	1. Click the drop down menu arrow. 2. Select a number between 1 and 50.	'Number of member to join' accepts a number between 1 and 50.	Pass		
		6.7	Can select 'By Item' for the 'Split Type'.	1. Click on 'By Item'.	Able to select 'By Item' for the 'Split Type'.	Pass		
		6.8	Can select 'Equally' for the 'Split Type'.	1. Click on 'Equally'.	Able to select 'Equally' for the 'Split Type'.	Pass		
		6.9	When you select one option for 'Split Type' the other gets unselected.	1. Click on 'By Item'. 2. Click on 'Equally'.	When you select one option for 'Split Type' the other does get unselected.	Pass		
	Adaptability	6.10	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any problems.	Pass	Viewed design on smaller phone size on Android Studio Emulator.	
Navigation	Consistency	7.1	Colours, font style and sizes are consistent.	1. Compare all elements to ensure consistency.	Colours, font style and sizes are consistent.	Pass		
	Spelling	7.2	There are no spelling errors.	1. Check all text for spelling errors.	There are no spelling errors.	Pass		
	Typography	7.3	Text is readable.	1. Check that all text is readable.	Text is readable.	Pass		
	Behaviour	7.4	Clicking the 'New Bill' button opens the 'New Bill' page.	1. Click on the 'New Bill' icon.	Clicking the 'New Bill' button opens the 'New Bill' page.	Pass		
		7.5	When on the 'New Bill' page the icon on the navigation bar is filled white.	1. Click on the 'New Bill' icon.	When on the 'New Bill' page the icon on the navigation bar is filled white.	Pass		
		7.6	Clicking the 'Past Bills' button opens the 'Past Bills' page.	1. Click on the 'Past Bills' icon.	Clicking the 'Past Bills' button opens the 'Past Bills' page.	Pass		
		7.7	When on the 'Past Bills' page the icon on the navigation bar is filled white.	1. Click on the 'Past Bills' icon.	When on the 'Past Bills' page the icon on the navigation bar is filled	Pass		
		7.8	Clicking on the 'Account' button opens the 'Account' page	1. Click on the 'Account' icon.	Clicking on the 'Account' button opens the 'Account' page.	Pass		
		7.9	When on the 'Account' page the icon on the navigation bar is filled white.	1. Click on the 'Account' icon.	When on the 'Account' page the icon on the navigation bar is filled white.	Pass		
	Adaptability	7.10	Design should adapt to being on a 5 inch screen.	1. Run the application on a phone with a 5 inch screen size.	Design adapts to being on a 5 inch screen without any	Pass	Viewed design on smaller phone size on Android Studio	

Figure 42: UI testing logs part 2.

We have designed 73 UI tests, distributed through the 5 different aspects. Due to the type of application that we have designed, there are many interactive elements on each page. This led to a majority of tests being based on behaviour, as reflected on the following pie chart.

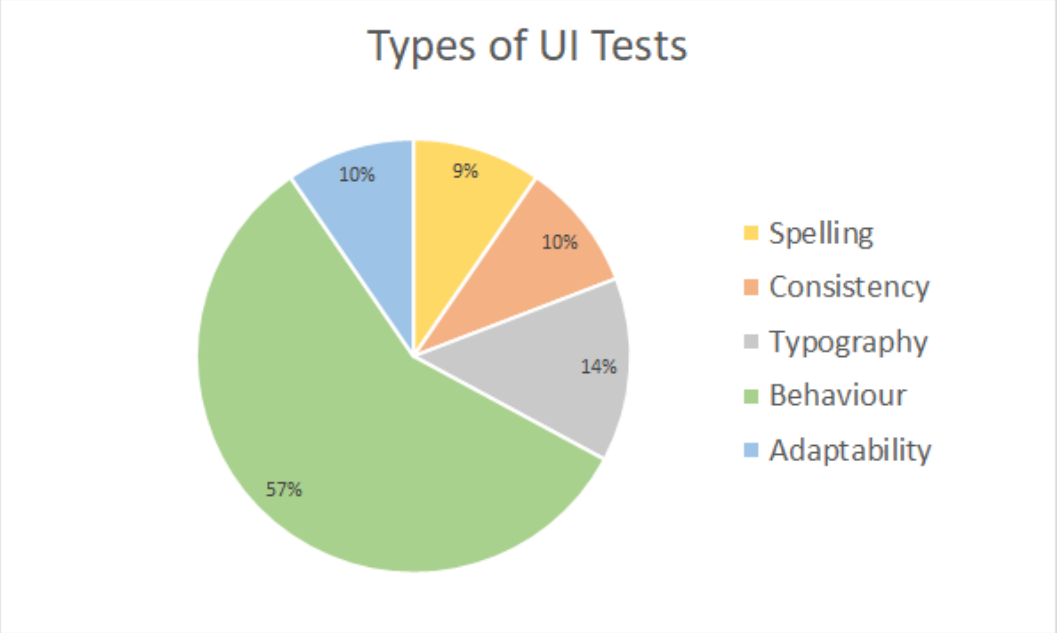


Figure 43: Types of UI testing distribution pie chart.

Moreover, the graph below provides an overview of the number of tests run and the number of tests failed in each section. Looking at this overview, we were able to determine that we only had two problems in two areas: consistency and behaviour. In total, only 22% of tests (16) failed, giving our application a 78% success rate in terms of UI testing.

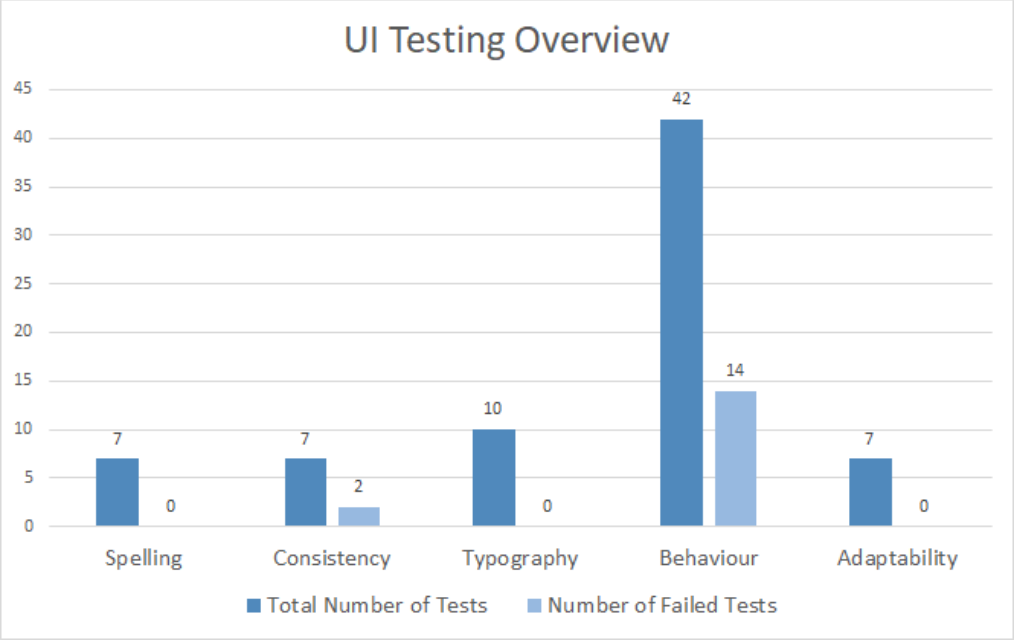


Figure 44: Total number of UI tests and fails for each category bar graph.

There were only two instances where we had issues with consistency: test 4.1, where the high contrast mode title and switch were not in the correct position on the 'Settings' page and test 5.1, where the filter icon colour did not match the rest of the UI. Both of these issues were simple to fix, and we did so immediately.

As you can see clearly in figure 44, our main issues were regarding the behaviour of the app. Out of a total of 73 tests, 57% of them (42) were testing behaviour. Of this 57%, 33% of tests failed (14 total). As shown in the logs, many of the behaviours tested failed as we have not yet implemented them using Java. Many of these fails were expected since we knew before starting testing that most functionalities had yet to be implemented.

No other areas had any failing test, allowing us to determine that we should have been more focused in terms of implementing behaviours, as this was the cause of 88% of our total failed tests.

User Testing

User testing is a process where participants who fit the target audience of a product perform a set of tasks using it to evaluate the user interface and functionality of the product [46]. We decided to conduct user tests to give us an external insight on how our is performing, if we had any opportunities for optimizations and whether our application appealed to users [47]. We went about user testing by asking users to perform a number of typical tasks that were a part of our MVP.

While remote testing may have been faster, we decided against it since we wouldn't be able to observe the user's reactions as well as we would in person, and it would make it more difficult to ask them to expand on any comments they may have [48]. This would of be a massive loss in information.

Additionally, since our application is not available on the Google Play Store, we would have to assist participants in downloading it using the applications Android Package Kit (APK), which is the standard file format for Android applications [49]. This would involve users having to set their phones into developer mode, override security settings to allow the installation of unknown applications and manually install the APK through a 3rd party application downloaded from Google Play Store [50]. While this process should be easy, our target audience are not necessarily technologically knowledgeable and may require assistance. Additionally, the need to use a 3rd party application could compromise our commitment to ethical development since we could not assess the security of the source code. Conducting testing in person also eliminates this problem, making it the superior choice.

We decided to use a mix of moderating and unmoderated in-person user testing. Moderated user tests consist of guiding the participant through the application and answering any questions they may have, while unmoderated user testing consists of allowing the participant to complete any tasks and answer questions in their own pace. We found a balance between the two by allowing users to take their time to complete tasks and answer a set of questions after they had finished, while encouraging them to think out loud so we could answer any questions and ask them to expand on any comments they may have.

This method was ideal as we were able to observe users, giving us a better understanding of the journey users take when using the application and allowing us to take note of any mistakes they came across to fix in the next iteration of our application[51]. By allowing users to answer a set of questions after they had completed all the tasks, we were able to obtain a genuine opinion on how they overall felt about the experience.

Below you can see the notes taken regarding the participants questions and/or comments during testing.

Participant ID	Parrticipant Age	Comments taken during testing process
1	18	▪The text in the password data field doesn't diappear when you start typing, its annoying. ▪There is no confirmation that you have signed up, its confusing. ▪The pound sign disappears when entering the details on the bill details page, it looks bad.
2	27	▪Sign up icon could be at the Sign in page ▪Your account fields are empty ▪Past bills page seems crowdy ▪Likes the theme
3	21	▪You can see white text in the email field on sign up. ▪Text on nickname field is on 2 lines, looks unprofessional. ▪No conformation that sign up has been successful. ▪Hom instead of Home on after I make the bill. ▪Clicking on a past bill should do something
4	23	▪It was easy to go from one page to another
5	25	▪When they signed up, there was no confirmation message
6	20	▪App seems to be missing pieces, on the past bills in particular
7	28	▪Filters don't do anything ▪Some things look like links but are only cosmetic ▪Test instrucutions had spelling errors

Figure 45: User testing comments and questions table.

As you can see in figure 45, our testing sample consisted of participants who were in our target audience: all participants were between 18 and 30 years old. We ensured that there was a diverse range of ages from our target audience, so our feedback was reflective of our whole target audience.

Our first task consisted of asking users to complete the sign-up process. This process consists of users registering their email, setting a password and nickname, and entering their bank details. As shown in figure 46, most participants found this process relatively easy. Participants 1, 3 and 5 expressed confusion due to the lack of explicit confirmation message after the process was completed, which we believe to be the reason not all participants were fully satisfied with the ease of completing the sign-up process. Taking this feedback into consideration, in the next iteration of our application we would implement a confirmation page at the end of this process to ensure users don’t experience any confusion.

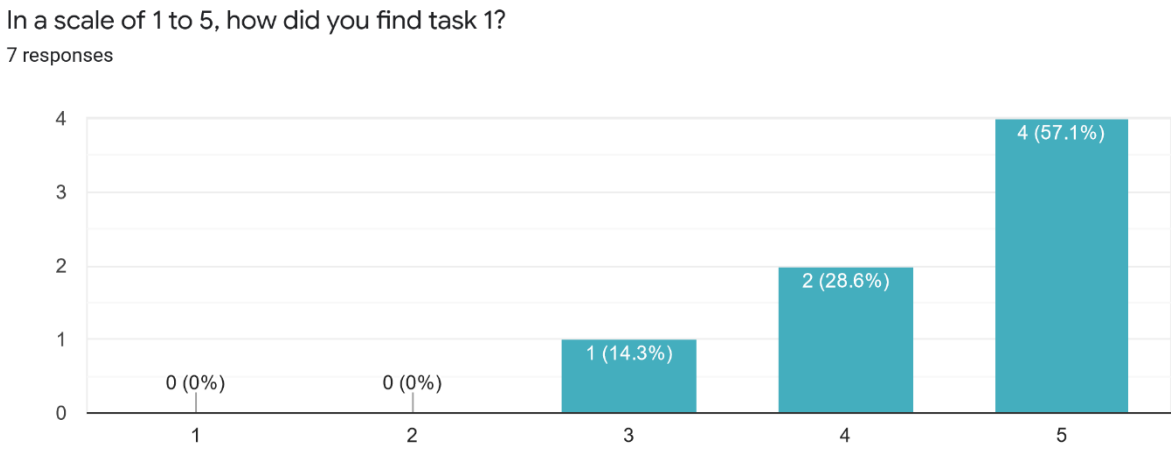


Figure 46: User testing questionnaire question 1 results (1 is difficult and 5 is easy).

Our second task consisted of asking users to login with the same details they entered when signing up in task 1. As shown in figure 47, this was a smooth process that left all participants satisfied. Going forward, we would not have to make any changes to this process.

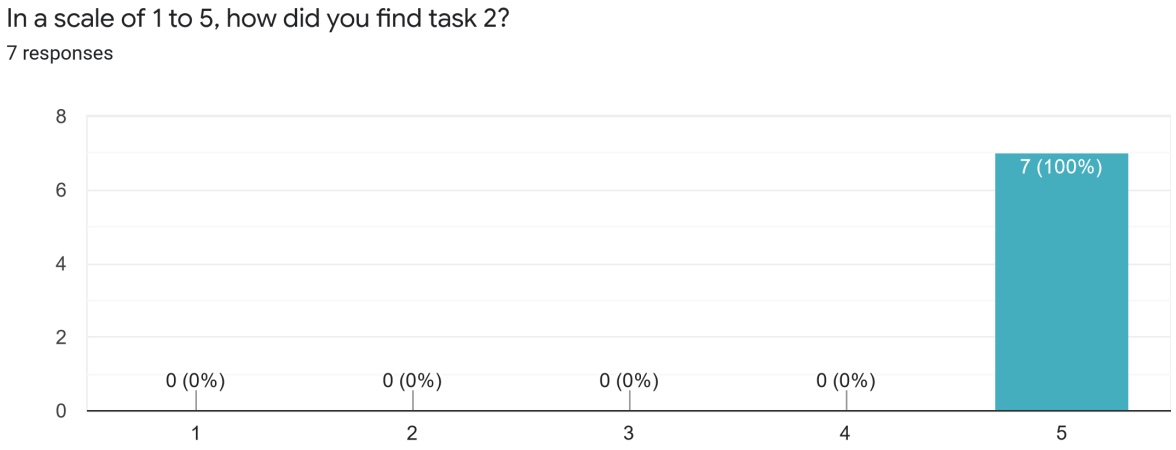


Figure 47: User testing questionnaire question 2 results (1 is difficult and 5 is easy).

The third task asked users to create a new bill that was split equally. Figure 48 shows the answers given by users in regards to the easy of this task. A majority, 57%, of users rated this task 4 out of 5. We believe this is die to multiple small factors; while there were no bugs encountered during this test, many participants verbally pointed out small errors, shown in figure 45. These included the sterling symbol disappearing when entering the bill amount, the text on the home button spanning two lines and thus reading "Hom" instead of "Home" and so on. We believe these are the reasons not all participants were completely satisfied with the ease of this process. Going forward, we would fix these small errors to give users the most pleasant experience possible.

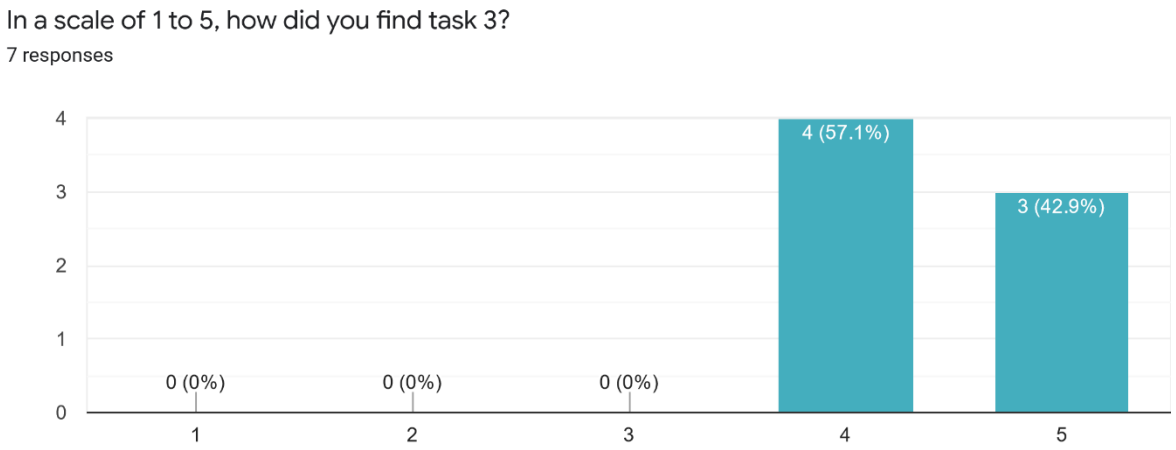


Figure 48: User testing questionnaire question 3 results (1 is difficult and 5 is easy).

Task 4 consisted of asking users to check a past bill. As shown in figure 49, there was a bigger ranger of responses in regards to the ease of this task compared to the previous ones. As shown in figure 45, many participants expressed confusion in regards to this page: Participant 3 voiced disappointment when clicking on a past bill didn't do anything, participant 6 commented on past bills feeling incomplete and participant 7 remarked on how the filters do not work. We believe these problems to be due to the fact we didn't implement most of this page's functionalities using Java. Given this feedback, this would be a top priority in the next iteration of this application.

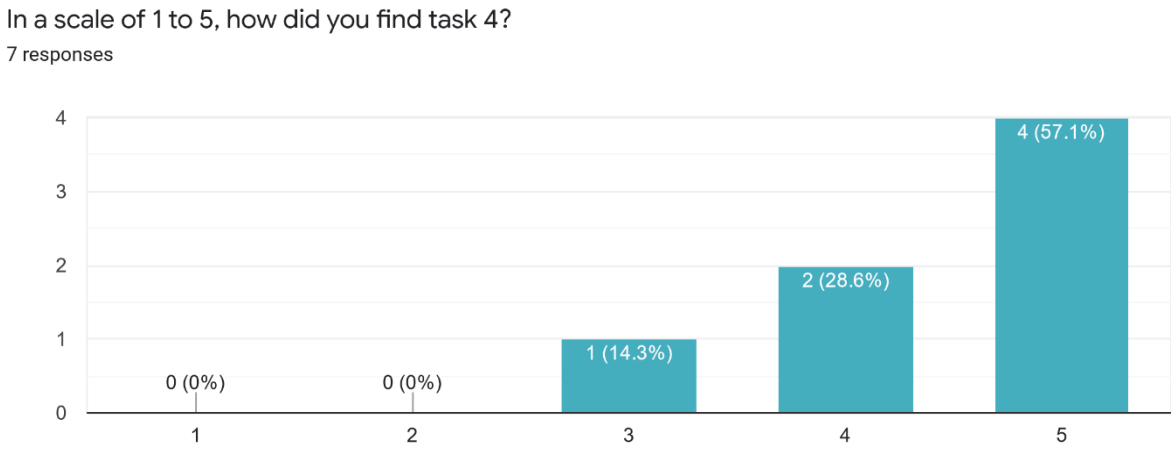


Figure 49: User testing questionnaire question 4 results (1 is difficult and 5 is easy).

Task 5 asked participants to logout. Users expressed no confusion in regards to this instruction and completed the task with ease. As shown in figure 50, this was a straightforward process and requires no changes going forward.

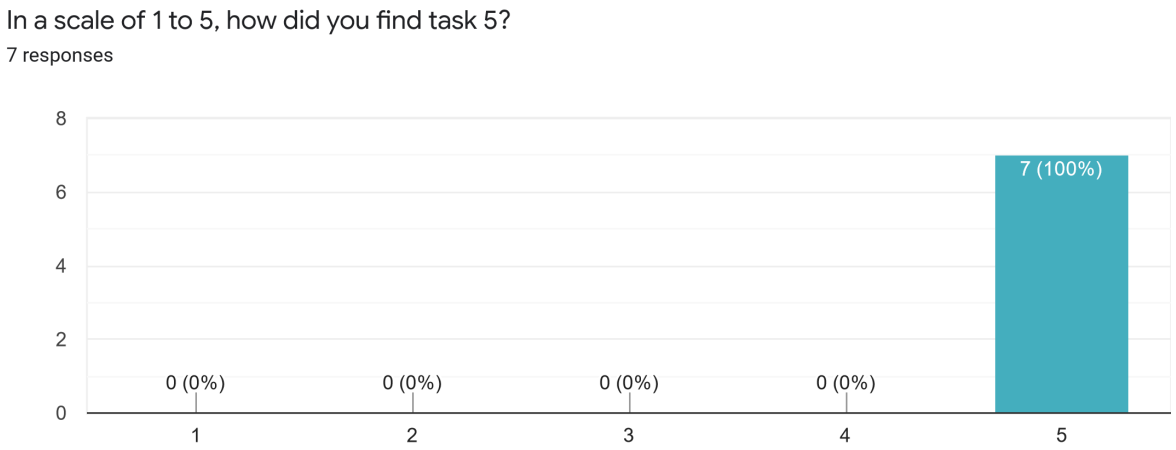


Figure 50: User testing questionnaire question 5 results (1 is difficult and 5 is easy).

Task 6 asked users to retry the sign in functionality but use incorrect login information. We wanted to test this process this way to guarantee it would fail gracefully and check that it is clear for users why this is the case. Figure 51 shows that all but one participant found this process to be extremely easy; during the testing process, all users were able to recognise the error message in response to their action. Going forward, we would make no changes to the error catching procedure of this functionality.

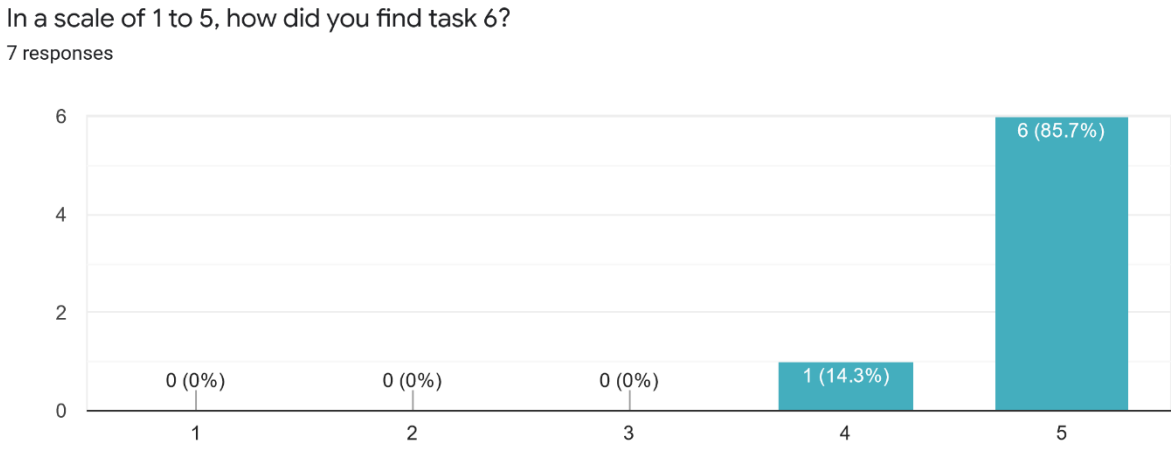


Figure 51: User testing questionnaire question 6 results (1 is difficult and 5 is easy).

The last two questions were open questions to allow for users to express their general thought and suggestions. Figure 52 shows the list of answers participants left when asked to describe their experience with the application. All feedback was mostly positive and there were only 3 participants who drew attention to errors and incomplete functionalities.

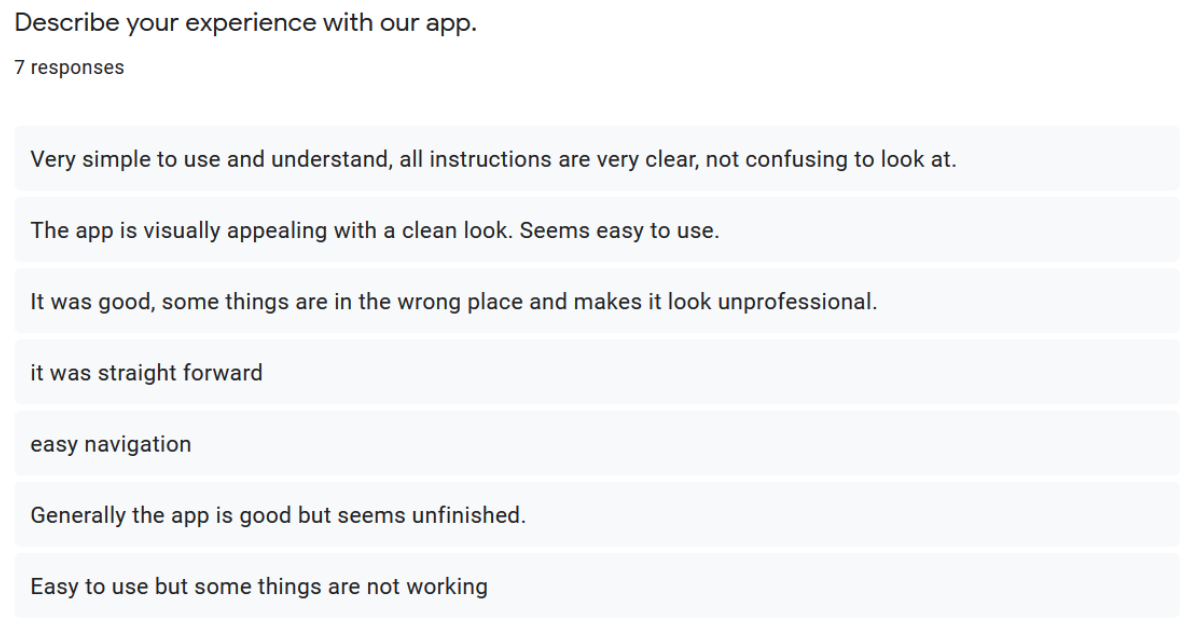


Figure 52: User testing questionnaire question 7 answers.

When asked for any suggestions to improve the application, users mainly expressed interest in new features rather than incomplete ones, as seen in figure 53. While all good ideas that would add to the overall appeal of the application, many of these suggestions would not add to the overall functionality of the application. Due to this, the next iteration of this application would mainly be focused on thoroughly completing the implementations of existing functionalities rather than adding those suggested.

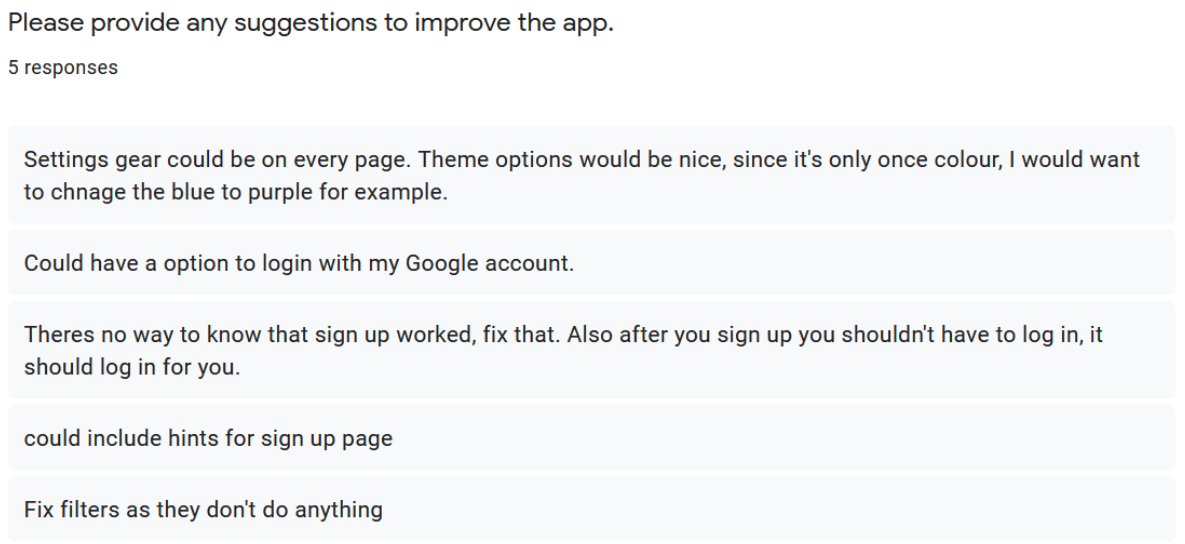


Figure 53: User testing questionnaire question 7 answers.

This testing process gave us valuable insight on the usability of our application. Usability was one of our main focuses, as this was the area we believed our competitors failed in, since during market research we had found that many of the already existing applications had many reviews suggesting they weren't user friendly.

Overall, we found that the functionalities that we had fully implemented had little to no problems. This was our expected result since we had conducted thorough user testing on our high-fidelity prototype before beginning development. Since we followed our prototype extremely closely, we believed that we would not have many issues in this regard. Our main issues were regarding functionalities that we had not yet fully implemented; going forward, this would be our main focus.

While we knew this could be an issue, due to the challenges that we faced and the consequent time constraints, we aimed to focus on a narrow scope. This would allow for chosen functionalities to be implemented fully, unlike with a wide scope, where the implementations would have been less complete given our time frame. Although this was a risky decision, we believed that it paid off, since our user testing shows that participants were generally very satisfied with our application and did not spend a lot of time focusing on features that had yet to be implemented.

Prototyping & Iteration

XML & Consistency

The implementation of the user interface is one of the many iterative processes we undertook when developing this application. The first step of this iterative process was to implement the user interface through XML files. To perform this with the maximum efficiency, we decided to conduct a sprint, where we split the group into 2 sub-groups of 2 people. By promoting the use of Agile methodology, results would be delivered swiftly and simultaneously. Thus, in our M7 meeting, XML files were assigned, and their aspects were discussed so later the merge would be natural and smooth; taking our previously developed high-fidelity prototype as a baseline, activity file names, elements naming format, their individual constraints, text sizes and positions for each element were reviewed within the group. From this meeting resulted directives for all 36 XML files and both groups worked on them simultaneously, as a second and third iteration, each subgroup in their own subbranch in GitHub, as shown in figure 54.

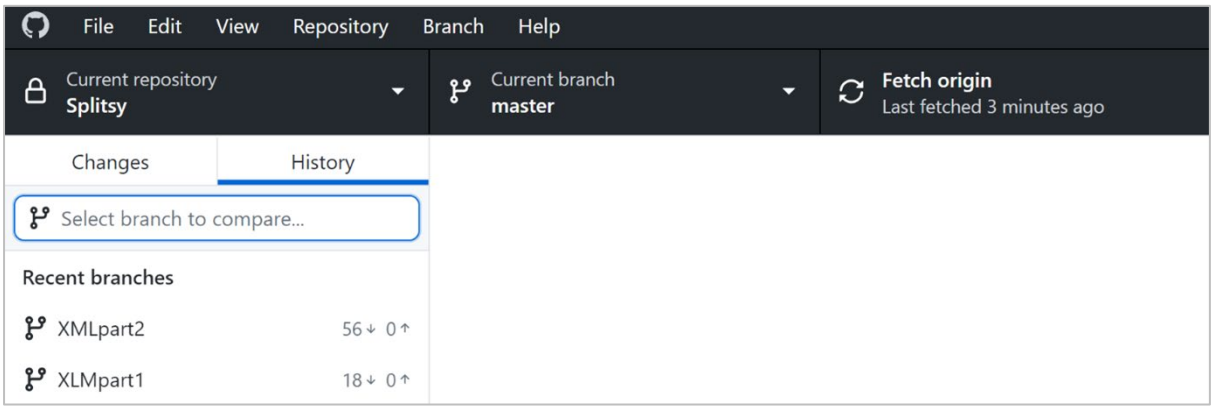


Figure 54: Branches XMLpart 1 and 2 as part of subgroups working alongside.

Before we started development of these pages, for optimal productivity and consistency, we created files of common use for both sub-groups such as drawable

buttons template files (figure 55) and our accent colour was added to the colours XML file for a consistent use across the application (figure 56).

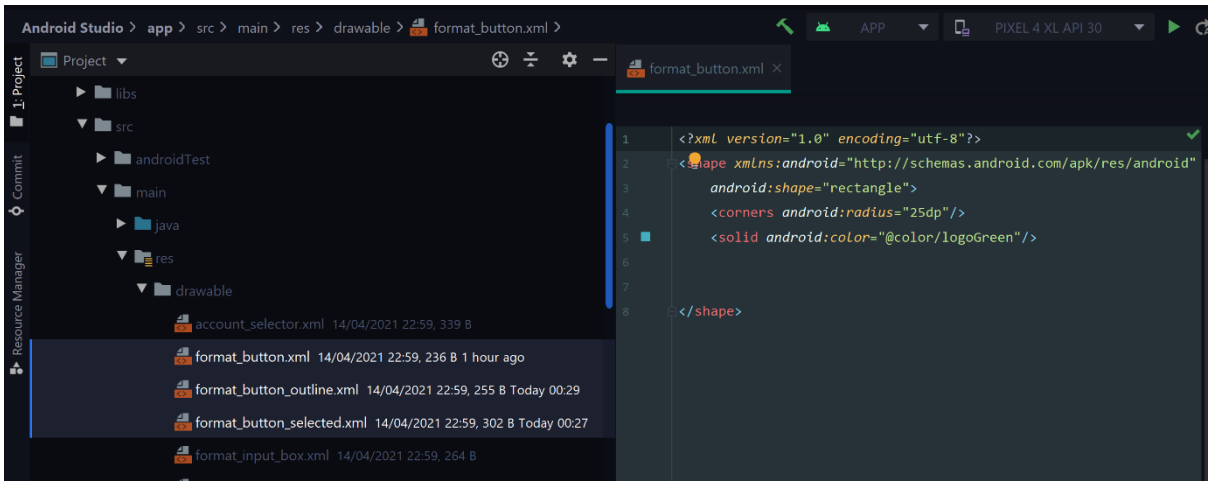


Figure 55: Template XML files for buttons used across multiple files.

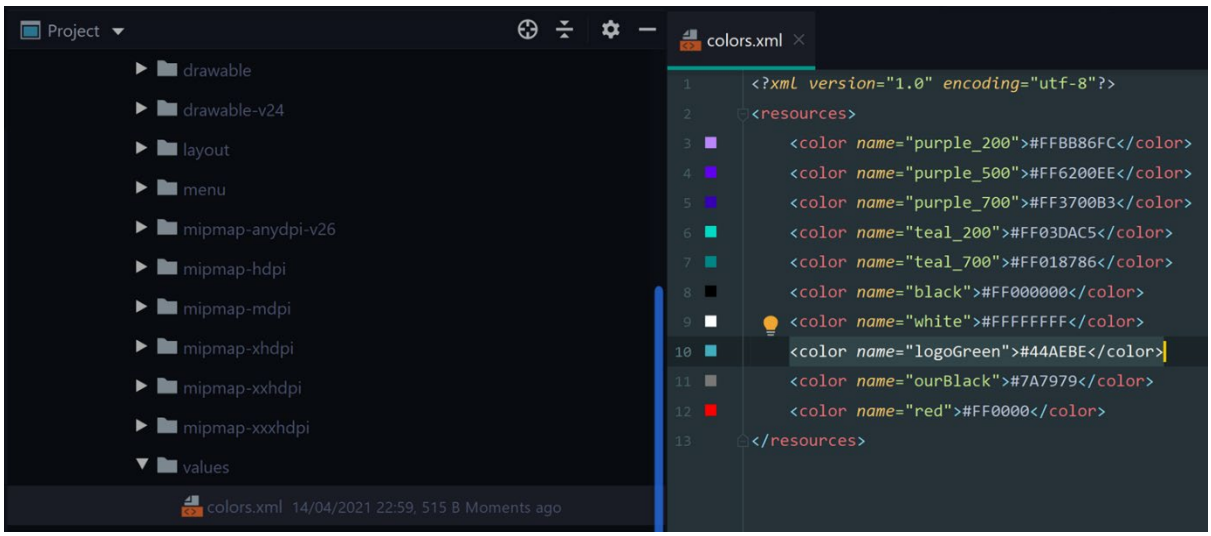
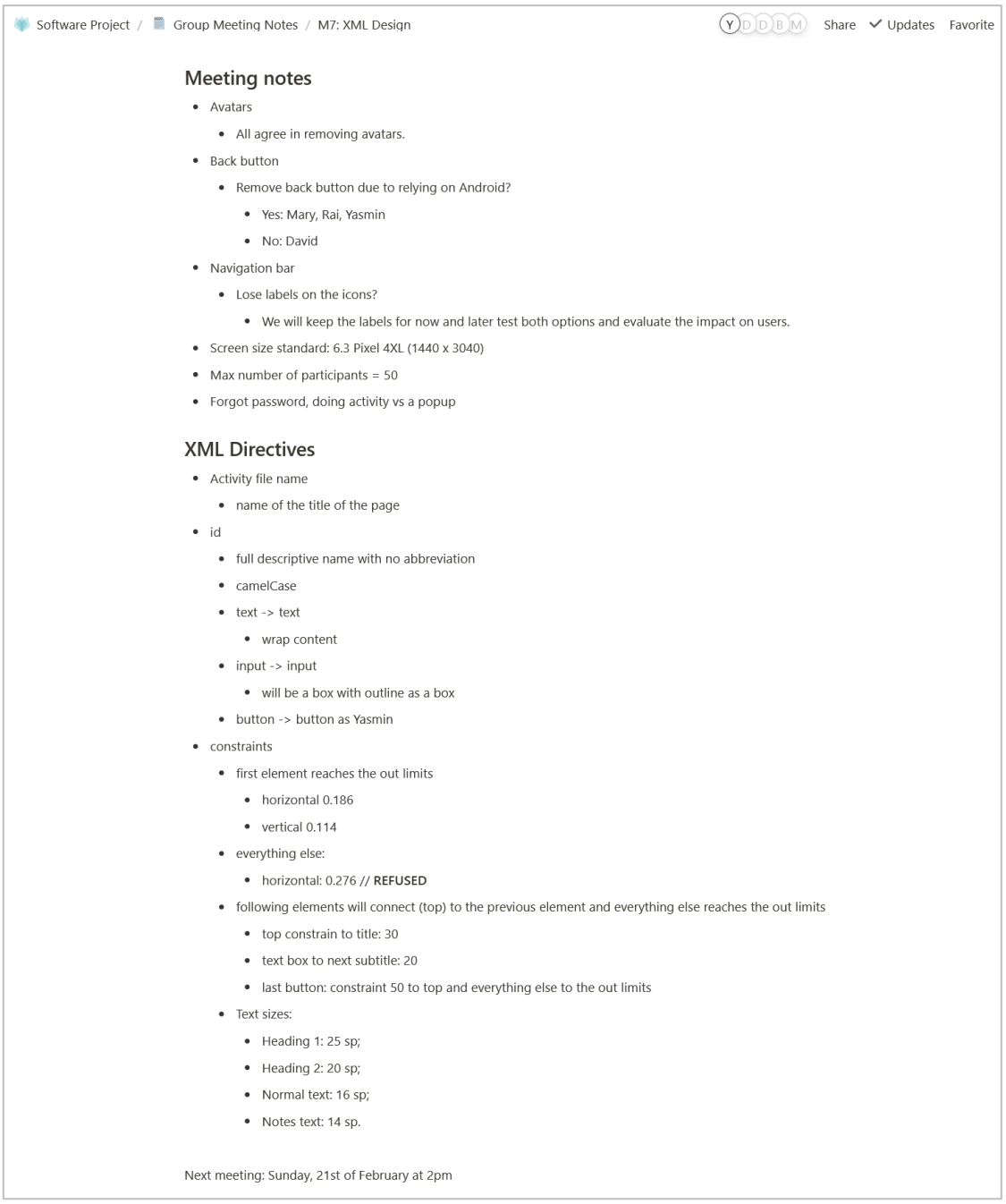


Figure 56: Logo green colour added to colours XML file.

During these iterations, icons and input field hints were implemented for better usability and readability. "*Consistency in UI design is concerned with making sure elements in a user interface are uniform*" [52]. We tried to use a consistent layout and colour scheme with good contrast with appropriate font size to make our application convenient to use. This also helped to improve user's assumption regarding the user interface creating a sense of familiarity and reliability.

Upon completion of the previous iterations, the following iteration consisted of two members, one from each sub-group, verifying consistency across all the pages in each respective branch, following the directives set previously (figure 57); only then branches "XMLpart1" and "XMLpart2" were merged into the same branch as shown in figure 58.



Software Project / Group Meeting Notes / M7: XML Design

Share Updates Favorite

Meeting notes

- Avatars
  - All agree in removing avatars.
- Back button
  - Remove back button due to relying on Android?
    - Yes: Mary, Rai, Yasmin
    - No: David
- Navigation bar
  - Lose labels on the icons?
    - We will keep the labels for now and later test both options and evaluate the impact on users.
- Screen size standard: 6.3 Pixel 4XL (1440 x 3040)
- Max number of participants = 50
- Forgot password, doing activity vs a popup

XML Directives

- Activity file name
  - name of the title of the page
- id
  - full descriptive name with no abbreviation
  - camelCase
  - text -> text
    - wrap content
  - input -> input
    - will be a box with outline as a box
  - button -> button as Yasmin
- constraints
  - first element reaches the out limits
    - horizontal 0.186
    - vertical 0.114
  - everything else:
    - horizontal: 0.276 // **REFUSED**
  - following elements will connect (top) to the previous element and everything else reaches the out limits
    - top constrain to title: 30
    - text box to next subtitle: 20
    - last button: constraint 50 to top and everything else to the out limits
  - Text sizes:
    - Heading 1: 25 sp;
    - Heading 2: 20 sp;
    - Normal text: 16 sp;
    - Notes text: 14 sp.

Next meeting: Sunday, 21st of February at 2pm

Figure 57: Notes from our M7 meeting with the directives agreed by the group.

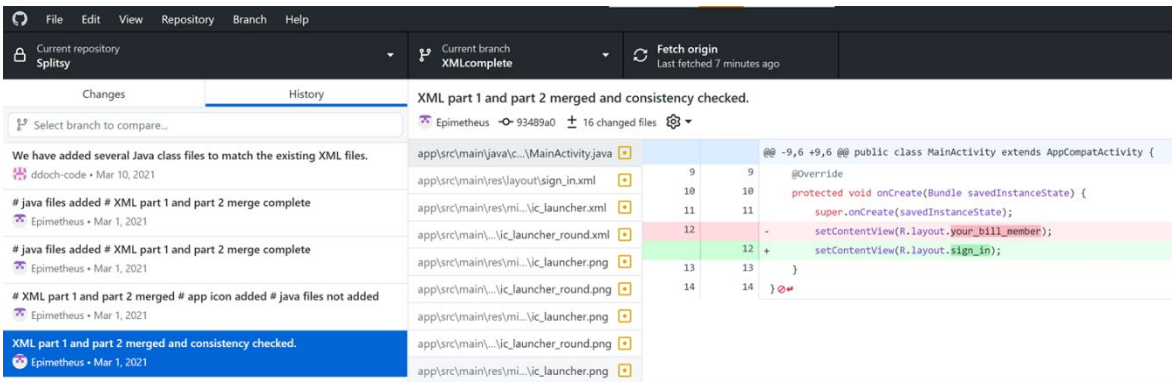


Figure 58: Pull request of the merging of XMLpart1 and XMLpart2 branches.

In the last iteration of this process, we linked the XML files with Java files (figure 58) to add the application functionalities and perform instrumentation and UI tests.

Camera API

For the implementation, we created a prototype using Kotlin which is a newly created language inspired by Java [53]. Kotlin is faster to compile and is lightweight. Each block of code is small compared to Java and easy to develop [54], meaning it reduces time in error detection in programs such as an Android Studio. Since Kotlin is interoperable with Java, we were able to configure the dependencies required to run Kotlin in the Gradle folder. We added all the permissions needed to use CameraX API in the Android Manifest folder.

We started developing the action listener, the preview section, the access for the camera in all types of devices, handling an exception not to interrupt the application lifecycle and finally setting up a path to store images, which you can see in figure 59.

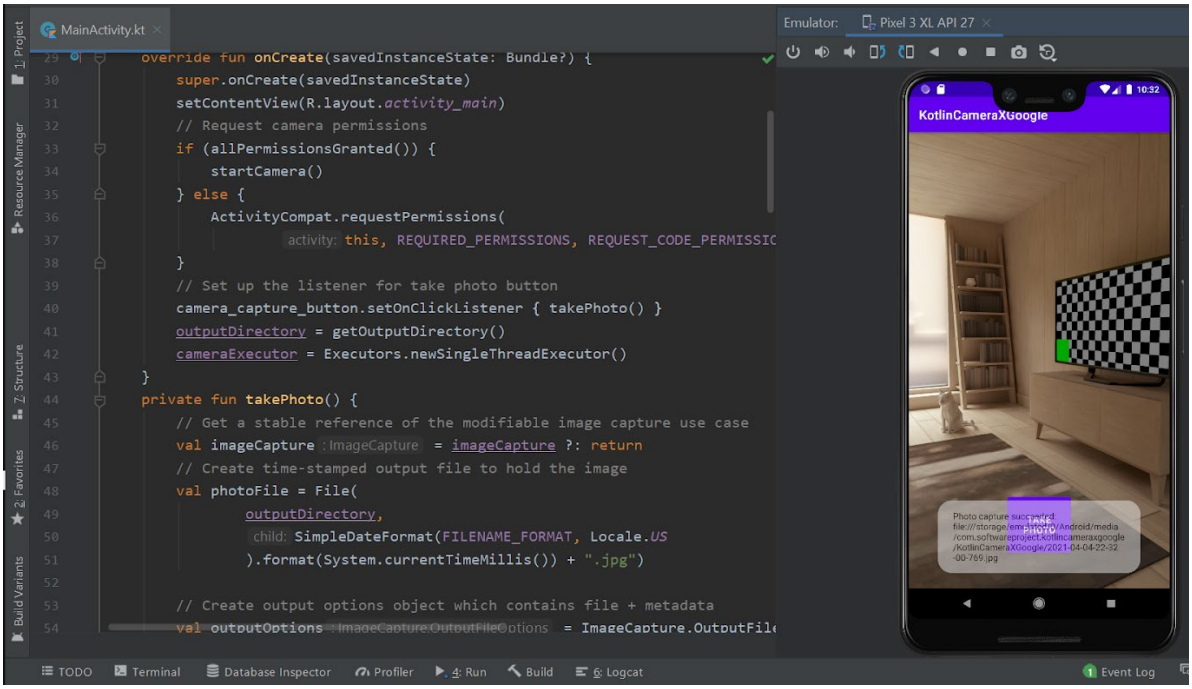


Figure 59: Camera API implementation part 1 snippet.

We were able to take pictures using the actual physical device and save them to our internal storage, as shown in figure 60. We have not yet implemented all the camera features, such as auto-focus to sharpen the focused frame and snap a clear image, and filter to enhance colour and minimise reflections.

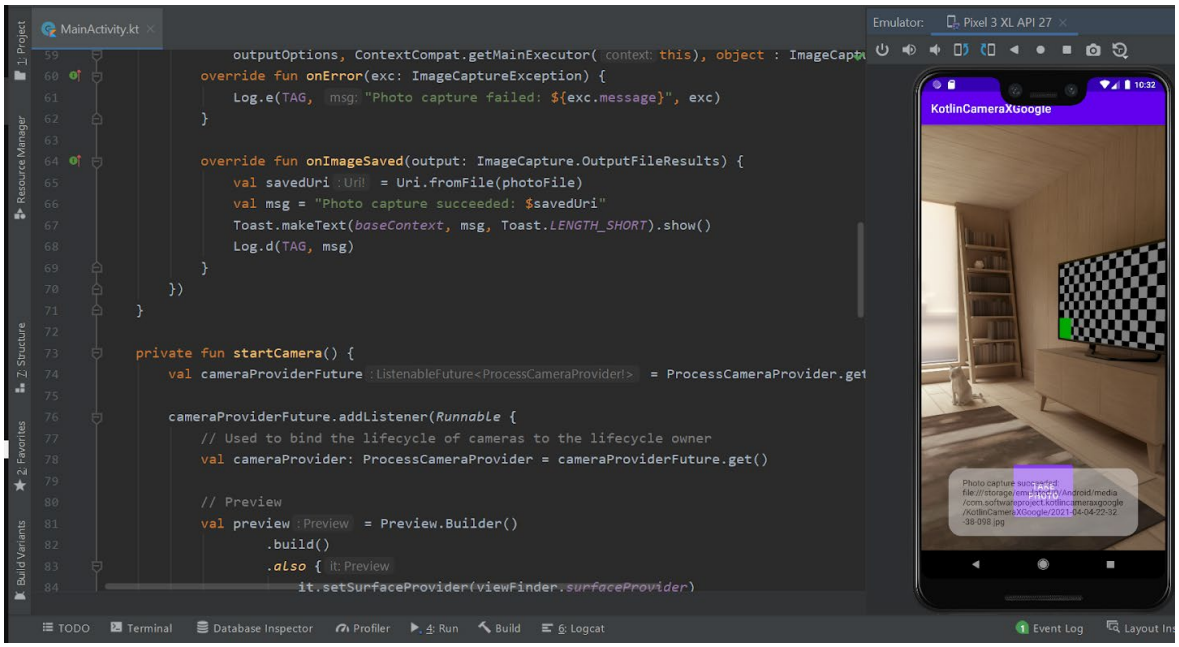


Figure 60*: Camera API implementation part 2 snippet.*

This prototype has not been integrated with our final application for now, as we focused on other more essential features first, such as creating database connections.

Technical Analysis

As we had no prior experience developing an Android application, we had to research and study the implementation of each area prior to starting the implementation. During this stage, we mainly used the documentations for the relevant technologies to ensure we understood the basics on how to get started. We then looked at working implementations similar to the ones we were expecting to achieve on tech blogs and code sharing websites such as GitHub. Despite this, during development, we ran into multiple problems.

For most problems, we were able to resolve them by checking the documentation again. If this proved futile, we then searched for similar errors on websites such as StackOverflow, where we were able to apply the advice given to our own implementation.

While this method worked well for most tasks, some implementations required specific knowledge on how the different technologies we were using interacted with each other, so the initial research was not adequate to develop a working implementation.

JDBC Implementation

We experienced problems when trying to form a connection from our application to our database through JDBC. As mentioned before, we worked in an iterative manner. This task was allocated to a sub-group while other members worked on other aspects, as usual when implementing Agile methodology.

Our first iteration consisted of connecting to a local database through a dummy project on IntelliJ, instead of Android Studio, in order to ensure that our already existing application code would not interfere with the connection. This implantation was successful, as shown in figure 61.

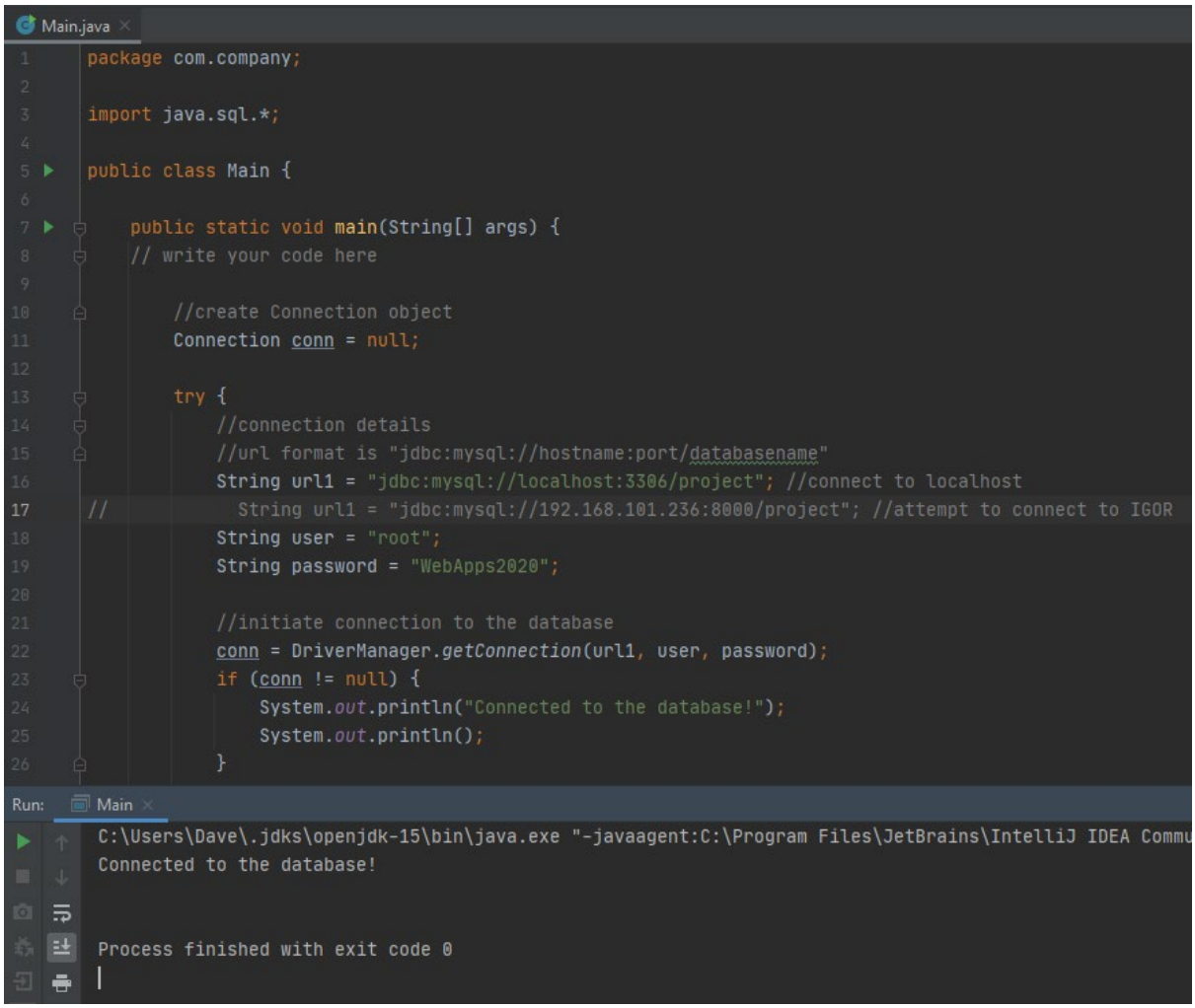


Figure 61: IntelliJ dummy project successfully connecting to local database.

The next iteration consisted of following the same process but connecting to the database on our virtual server instead. While connecting to the local database worked flawlessly, this was not the case when implementing a connection to the database hosted on the virtual server. When we changed the server address, username, and password we were presented with an error. As shown in figure 62, there was a communications link failure, and the driver had not received any packets from the server.

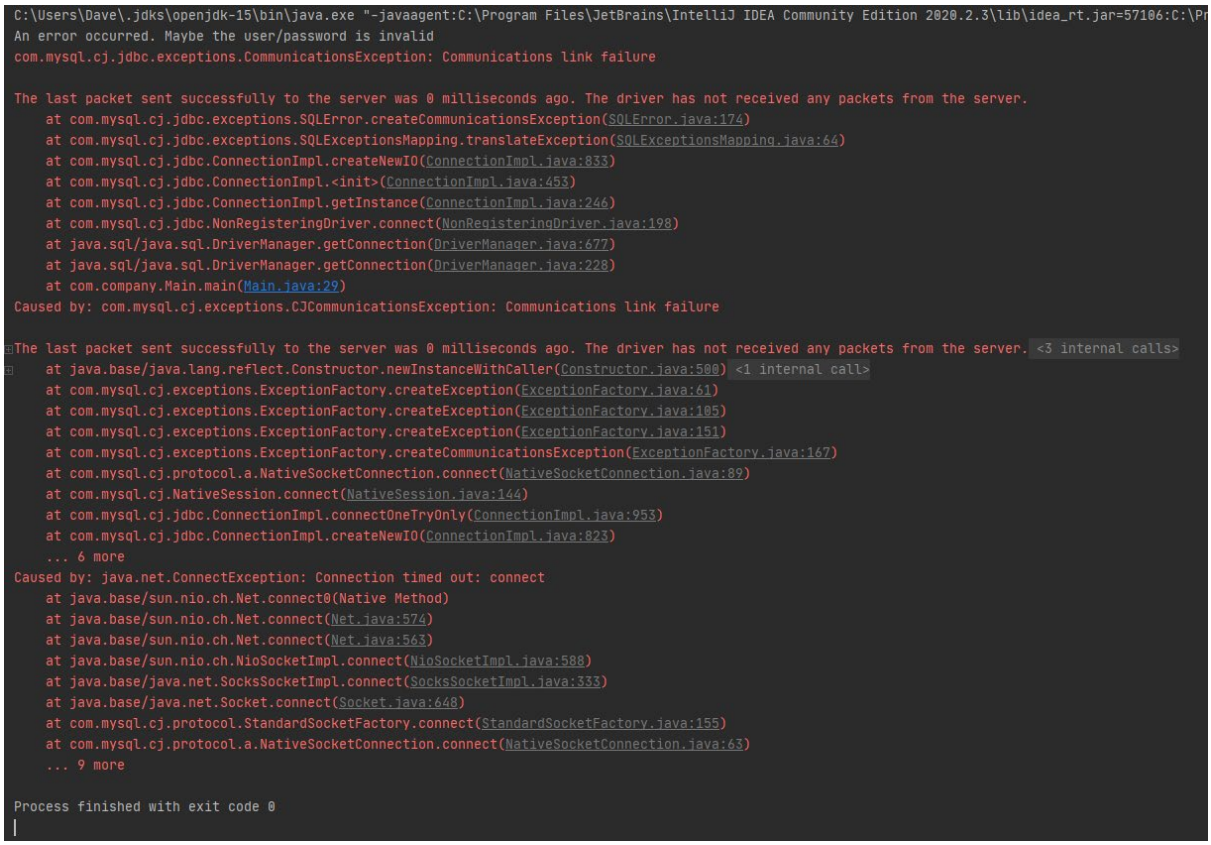


Figure 62: Error message received when attempting to connect to virtual server database.

At this point, we researched for a solution. By reviewing implementations of JDBC [55], we confirmed that our implementation for connecting to a local database could be altered to connect to a database hosted on a virtual server by simply changing the database address. Although this was the method we implemented, we were still unable to connect to the database and we could not identify the cause of the problem. We sought support on this matter and were advised to consider using SQLite. Through our research [56] we found that this was not a viable solution due to the restrictions on the use of data types, the lack of support for XML formats, SQLite's inability to scale successfully, among others. Instead, we decided to reallocate this task as other members of the group would have a fresh perspective of the problem.

The new sub-group attempted to connect to the database through a mock Android Studio project and was also unsuccessful. We started troubleshooting and noticed, through research, that some JDBC versions could be incompatible with Android Studio [57], so we chose version 5.0.4 as it was known to be compatible. With compatibility between JDBC and Android Studio ensured, we still could not connect to the database, so we checked if there was a security protection in place. We performed a ping test, as shown in figure 63, and found that no packets were able to be delivered.

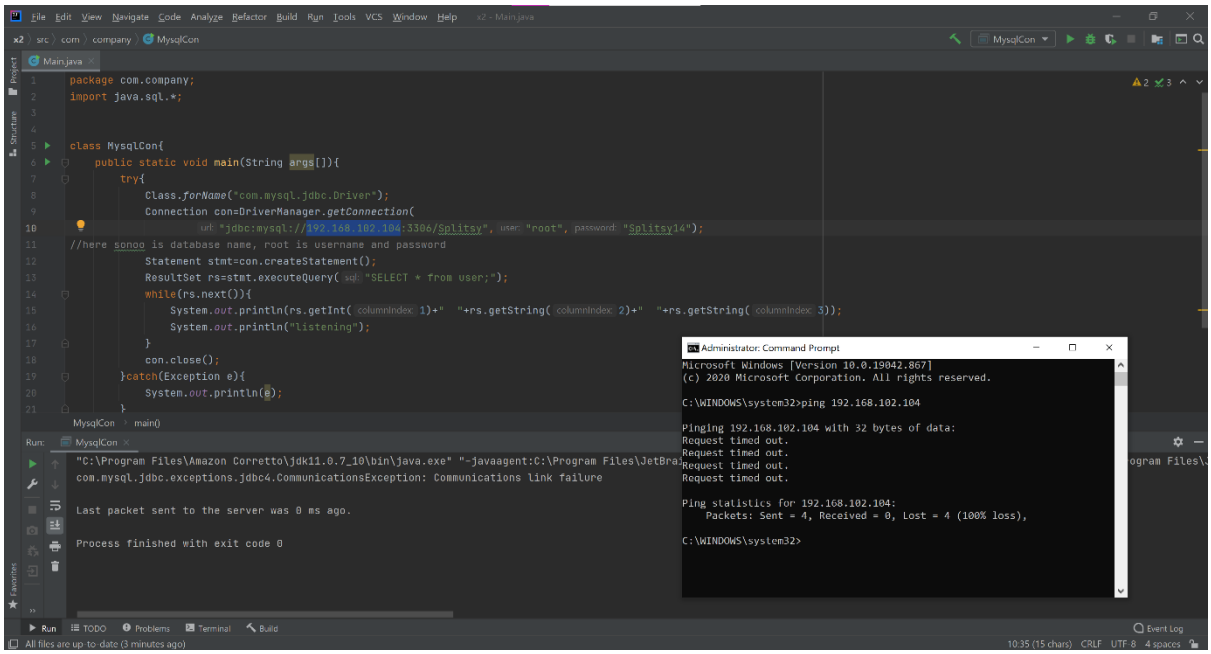


Figure 63: Ping test to virtual server IP address.

This made us confident that there was a security problem, hence we contacted the database administrator, Eamonn Martin, as shown in figure 64. We were informed that the virtual servers cannot be accessed directly from outside the Goldsmiths network. He suggested using the MySQL server running on the department IGOR server instead and provided us the necessary access and steps.

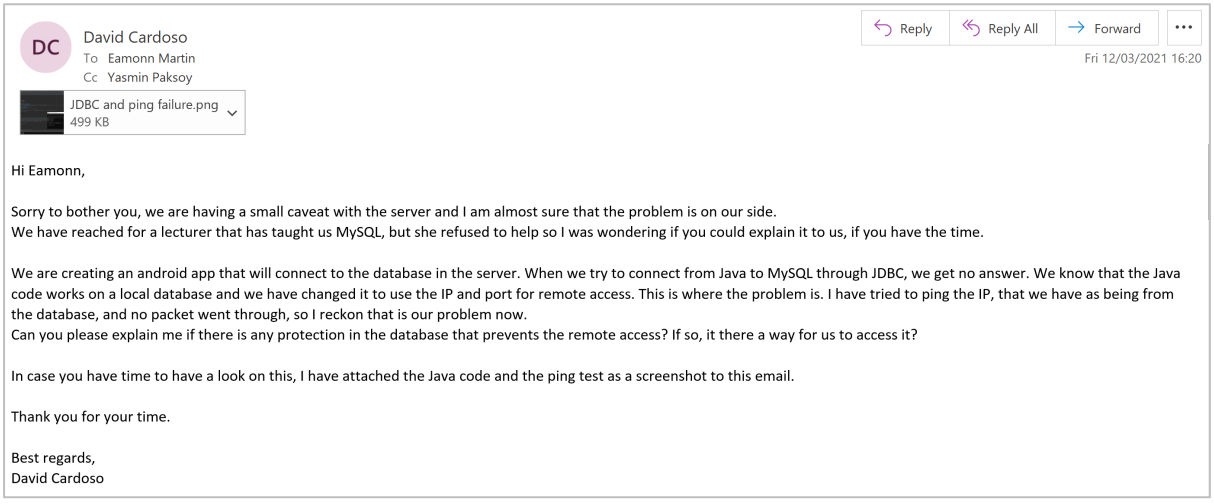


Figure 64: Email sent to database administrator, Eamonn Martin.

We followed his advice and replicated our database on the IGOR server. First, we attempted to connect to this database via a dummy project on IntelliJ. As shown in figure 65, this was successful, and we were able to retrieve the data to the command line.

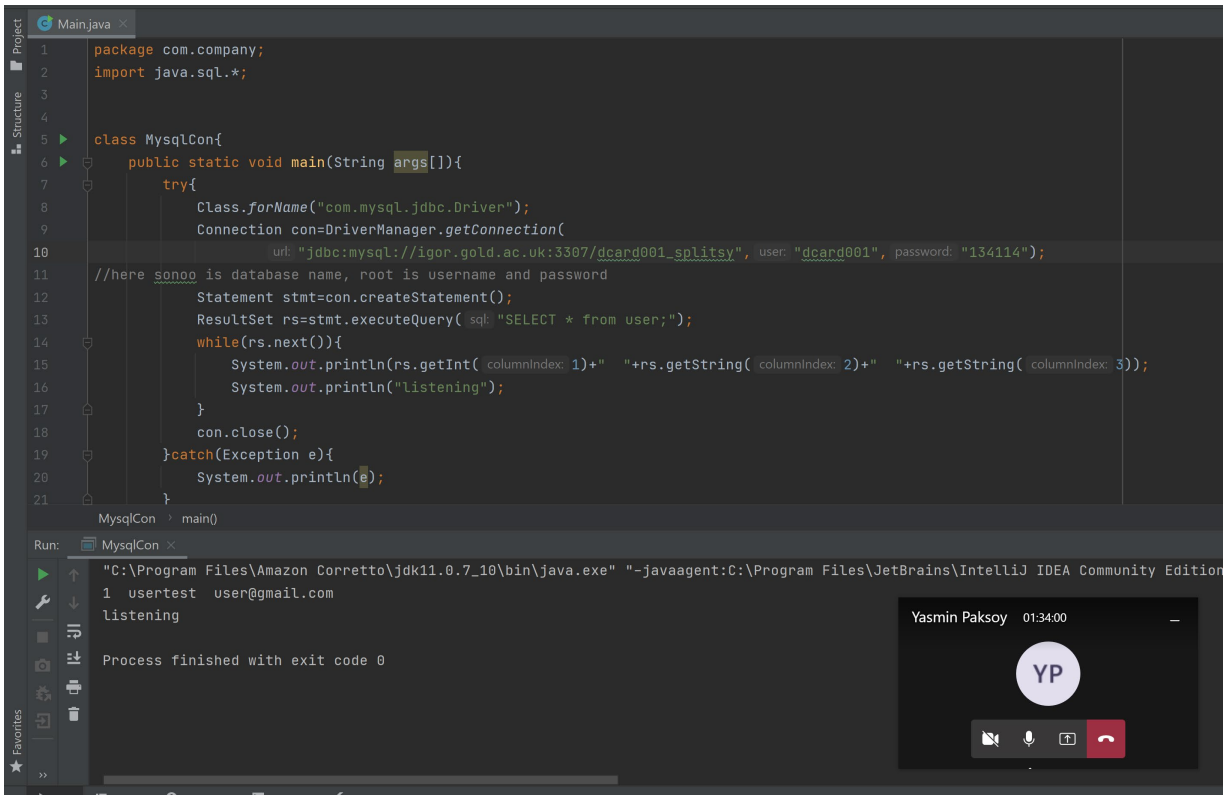


Figure 65: IntelliJ dummy project sucessfully connecting to IGOR database.

We the tried changing the IP address and password on our mock Android Studio implementation to verify that it would also work. This allowed for a successful connection to the database, as shown in figure 66.

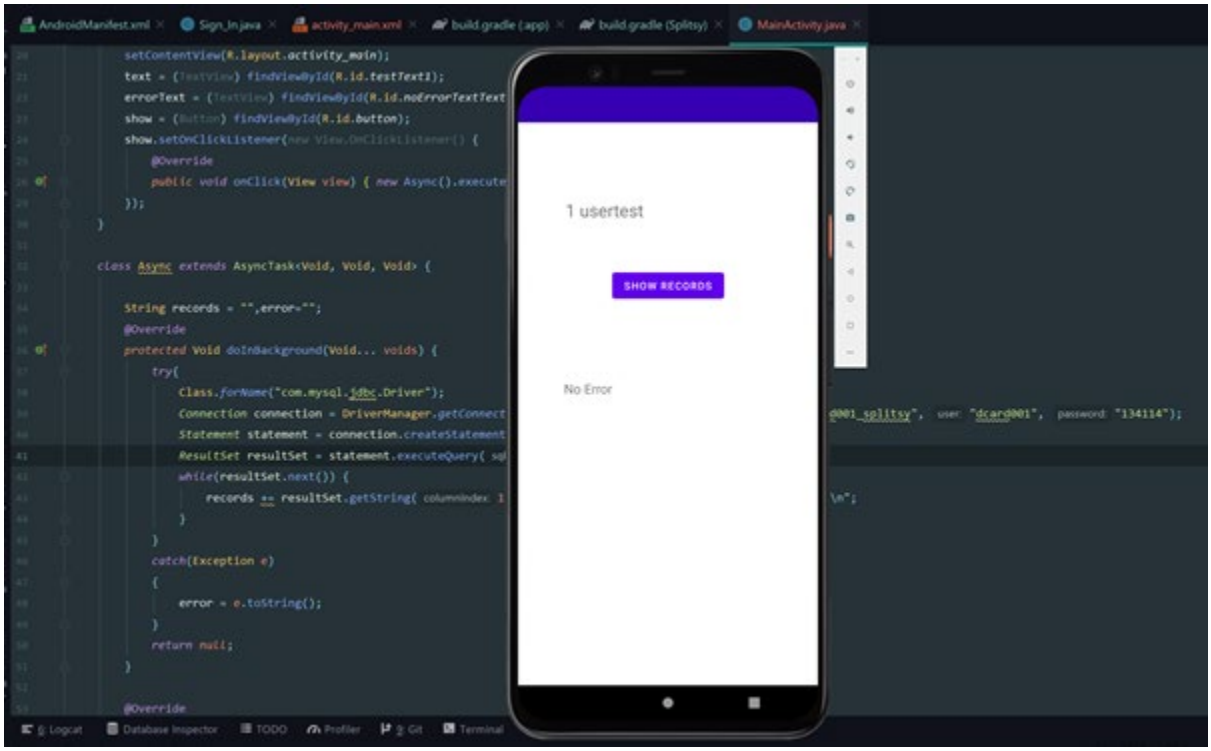


Figure 66: Android Studio dummy project connecting to IGOR database.

Subsequently, we were able to replicate this process in our application. While this should have been a smooth process, we encountered a Class Not Found Exception, as shown in figure 67.

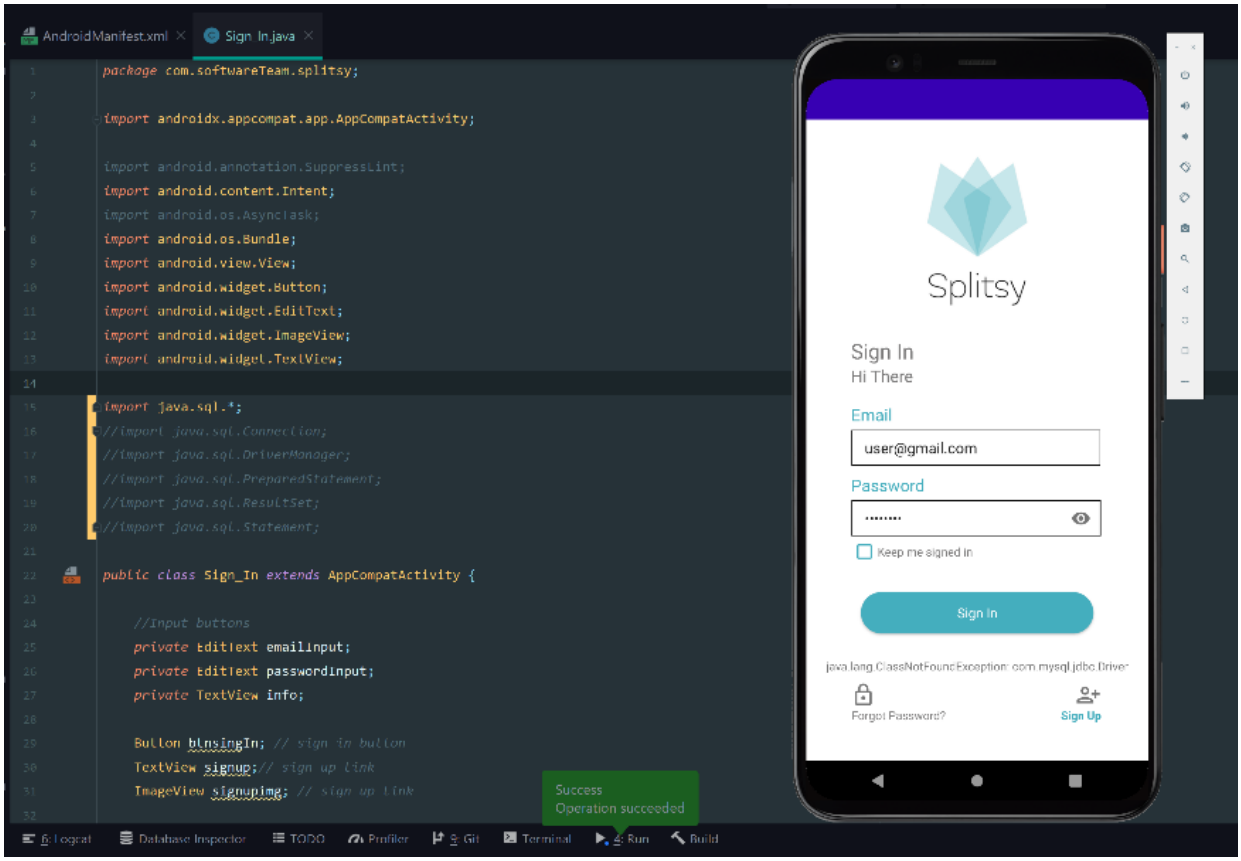


Figure 67: Emulator showing Class Not Found Exception error.

We were not able to find a fault in our implementation and so continued researching different causes for this error. We soon noticed that all JDBC implementations we reviewed used a AsyncTask [55], and not a standard Java function. An AsyncTask does "background operations on a background thread and updates on the main thread" [58], meaning that we could connect to the database before executing the Sign In functionality, through the second thread, where the AsyncTask is executed. This method does not disrupt the main thread and cause an error. Therefore, we used a AsyncTask class on our implementation and were successful.

Linking XML and Java

Our application contains a navigation bar at the bottom of many pages. We did research to familiarise ourselves with the basic implementation of this functionality, and followed the graphic illustrations given in our project proposal when creating our design. This was implemented through the use of XML, by creating drawable resource files for the selected and non-selected icons, and three selector files to switch between them (new_bill_selector.xml, past_bills_selector.xml, and account_selector.xml). In the selector files we set the "state-checked" method to change the icons when they are selected, as shown in figure 68.

styles.xml | account_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:drawable="@drawable/ic_account_filled"
        android:state_checked="true"/>
    <item
        android:drawable="@drawable/ic_account_outlined"
        android:state_checked="false"/>
</selector>
```

Figure 68: account_selector.xml snippet.

After our first XML implementation, we launched the emulator to test the design and found the icons were too small compared to the rest of the page. The small size meant it was difficult to identify each option, as seen in figure 69. This could make the application less accessible to those who have visual impairments, so it was something we wanted to improve.

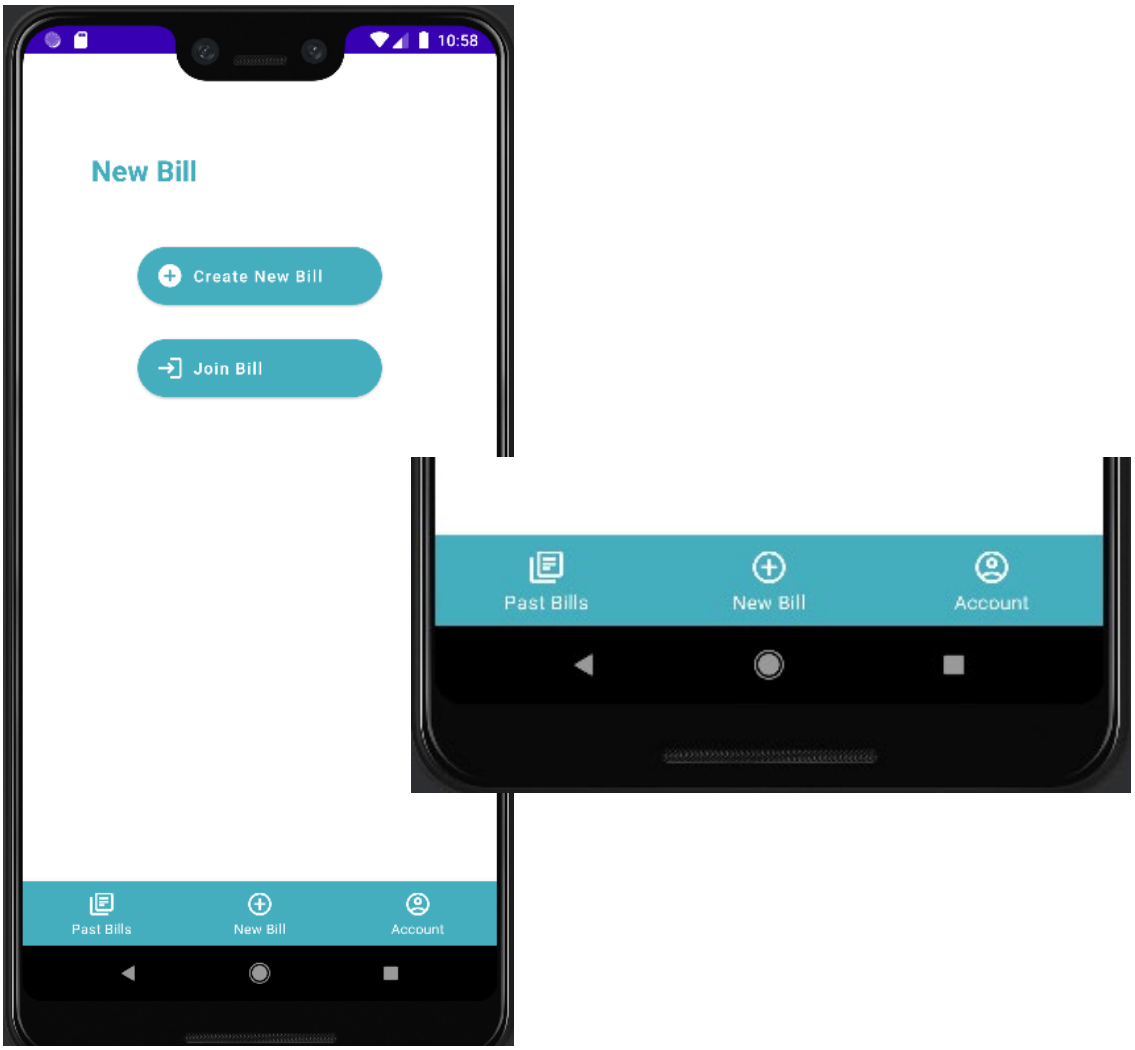


Figure 69: Original navigation bar with small icons.

To fix this, we decided to increase the size of the icons in the XML file. However, this method caused the icons to overlap the titles [59], as shown in figure 70 below. This meant it was still difficult to understand each icon's purpose. We agreed that the icons alone did not clearly convey the option they represented, so although eliminating the titles would have resolved the problem, it was not an option.

Figure 70: Icons overlapping titles on navigation bar.

Our solution was to implement a values resource XML file called dimen.xml [60], as seen in figure 71. Much like other values resource files, this file is useful to store dimensions that are used in multiple places across your application. It allows you to change these common values in one spot, rather than updating each place individually. We used this file to set the dimensions of each element of the navigation bar by overriding the existing parameters for the icons and text [61]. This ensured they were of the correct size relative to the page and each other.

nav_bar.xml | dimen.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <dimen name="design_bottom_navigation_elevation">8dp</dimen>
    <dimen name="design_bottom_navigation_shadow_height">1dp</dimen>
    <dimen name="design_bottom_navigation_icon_size" tools:override="true">40dp</dimen>
    <dimen name="design_bottom_navigation_margin_top" tools:override="true">5dp</dimen>
    <dimen name="design_bottom_navigation_label_padding" tools:override="true">8dp</dimen>
    <dimen name="design_bottom_navigation_text_size" tools:override="true">14sp</dimen>
    <dimen name="design_bottom_navigation_active_text_size">14sp</dimen>
    <dimen name="design_bottom_navigation_item_min_width" tools:override="true">56dp</dimen>
    <dimen name="design_bottom_navigation_item_max_width" tools:override="true">96dp</dimen>
    <dimen name="design_bottom_navigation_active_item_max_width">168dp</dimen>
    <dimen name="design_bottom_navigation_active_item_min_width">96dp</dimen>
    <dimen name="design_bottom_navigation_height" tools:override="true">70dp</dimen>
</resources>
```

Figure 71 dimen.xml file snippet

Below in figure 72 you can see our finished design for the bottom navigation bar once the dimensions resource file had been applied to it.

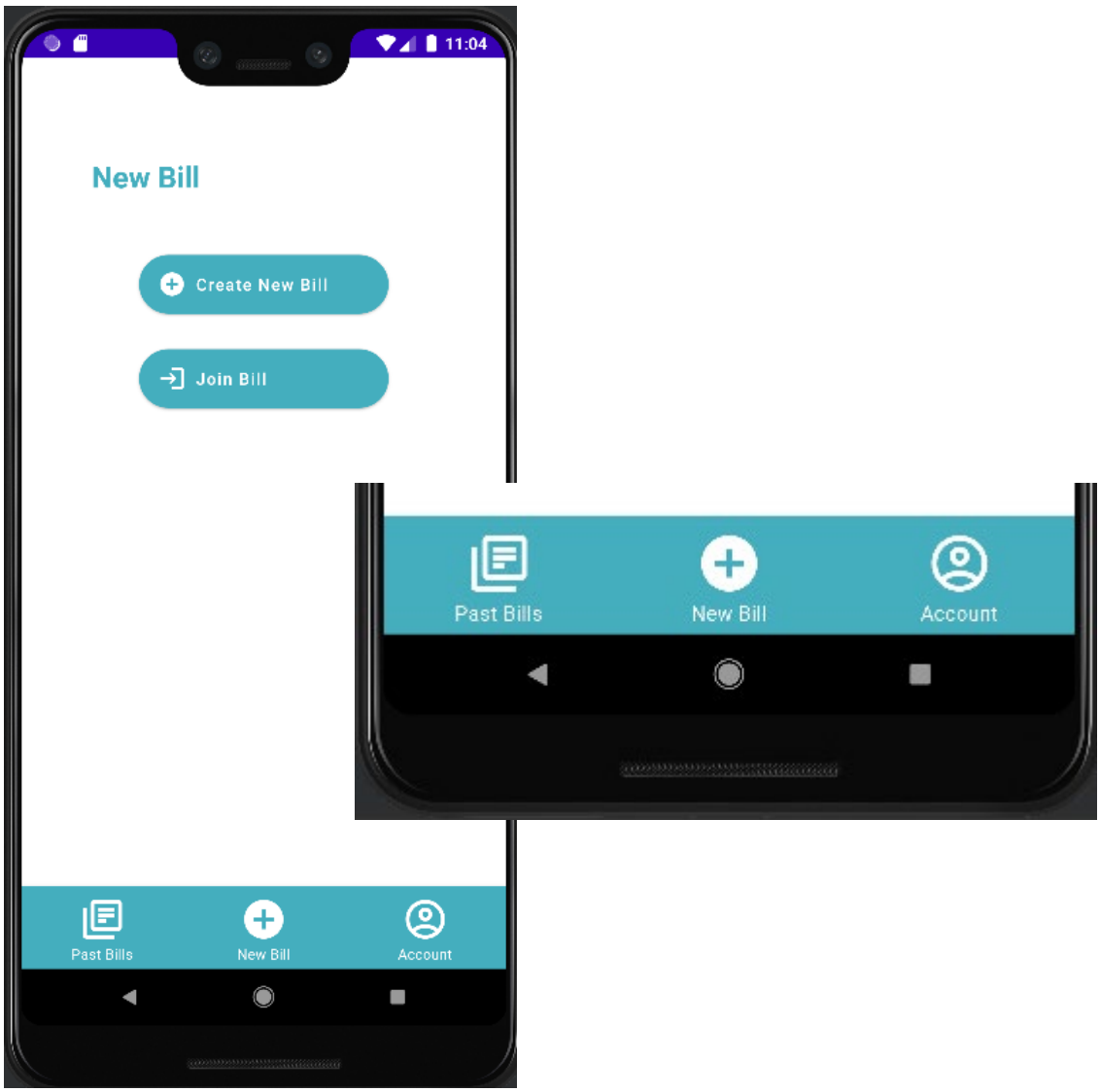


Figure 72 Fixed navigation bar.

After finishing the XML design of the navigation bar, we created the Java files, which are responsible for the functionality of the navigation bar. After our first Java implementation, we ran into a problem. Upon starting the emulator, the application would crash, as shown in figure 73. We looked at the XML layout file that was causing the crash and commented out each UI element in turn to find out which block of code was responsible. When we found it, we were able to ascertain that using a certain background style was causing the error. When we replaced this with another background the problem was resolved.

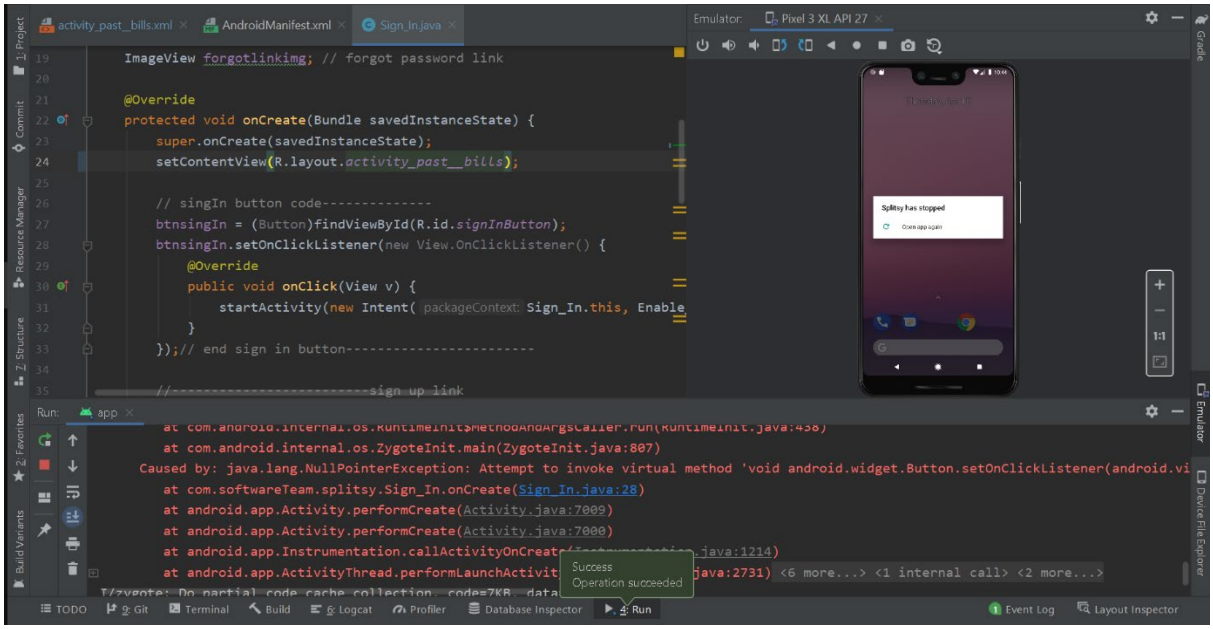


Figure 73 Emulator throwing error.

Moving on, we encountered another problem. Although the application no longer crashed, the navigation bar did not work correctly. When the icons were clicked, they would become highlighted, but the page would not change. Although the navigation bar was clearly responding to being pressed, the actual navigation did not take place. To determine if the issue was with our implementation, we created a straightforward navigation button in a new activity to try and understand what was going wrong. This is shown in figure 74.

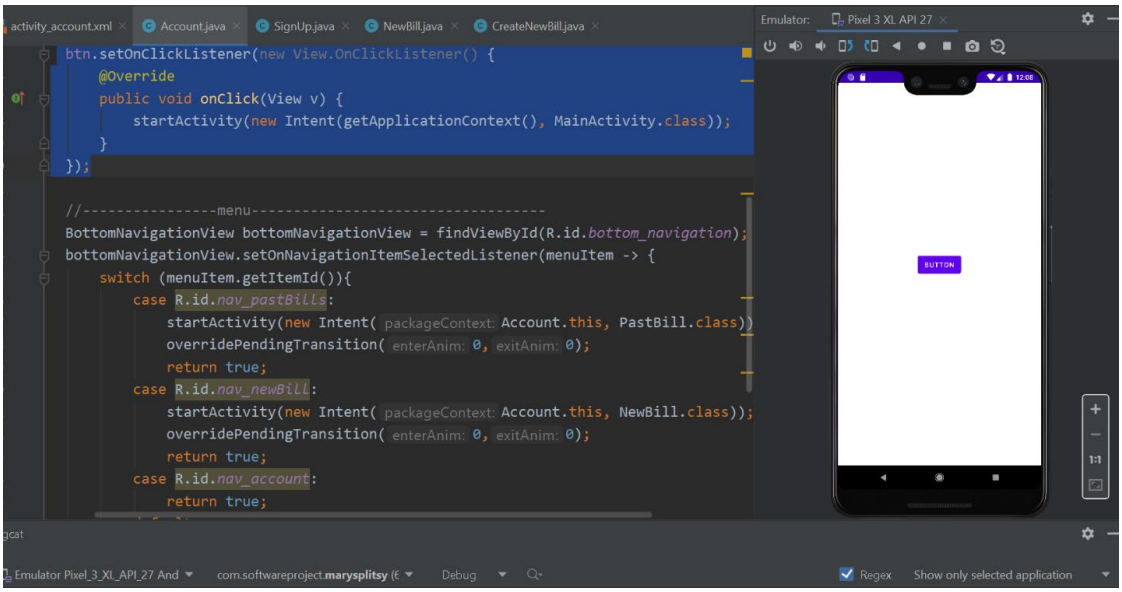


Figure 74: Simple button implementation snippet.

We found that even this simple implementation did not work as it was supposed to. We concluded that our pages were not linked together correctly; however, we could not locate where the error had been made. All pages were declared in the AndroidManifest.xml file and were correctly linked with the equivalent XML layout resource files. We also double-checked:

- The correct layout IDs were referred to when using the findViewById() function.
- The setOnClickListener() functions were correctly formatted.
- The button itself was correctly initialised and formatted.
- The button view was correctly cast to a Button object to have access to the above functions.

When we started our project, we began with one activity that had both Java and XML files. After we had implemented our XML layouts, we created the relevant Java files which we linked to each activity and then declared in the manifest. We now believe this method was the root cause of the navigation bar problem.

Instead, we experimented with creating a new activity directly via the File menu (File -> New -> Activity). When doing this, Android Studio would automatically create a connected Java file. We could fill in the code and add the layout XML, and the navigation bar would then work correctly. When we recreated our pages in this manner, the issue was resolved.

In summary, the navigation bar implementation challenged our research and coding skills to overcome the issues we faced. Learning about the creation of the Java files has helped us better understand how the Java and XML elements in Android Studio relate to one another.

Radio Buttons on XML

On the "Create New Bill" page, we needed to implement a pair of buttons that allow the user to select whether they would like to divide the bill equally or split it by item. The user should not be able to continue until they have chosen one of

these options. Only one button should be able to be selected at once and pressing the other button should deselect the currently selected one.

This functionality could be implemented using radio buttons. However, as we only need two options, we decided to implement it using standard button views, for which we had already created Drawable Resource Files (format_button_outline.xml). These files helped give our button a more attractive design, as shown in figure 75:

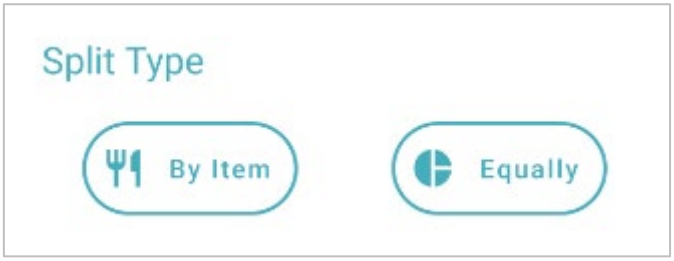


Figure 75: The split type buttons.

The logic for these buttons was simple to make. Two Boolean variables are created (one for each button) and initialised as false. When a button is pressed, its Boolean is set to true while the other is set to false, to ensure only one can ever be true at a time. When a user presses the "Continue" button, the page they move on to depends on which variable is true. If no button has been pressed, neither is true and the user cannot move on, as shown in figure 76.

```
// ----- create new bill button-----
createbtn = (Button) findViewById(R.id.scanCreateButton);
createbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if (equallySelected) {
            startActivity(new Intent( packageContext: CreateNewBill.this, EnterBillDetails.class));
        }

        else if (byItemSelected) {
            startActivity(new Intent( packageContext: CreateNewBill.this, ScanYourReceipt.class));
        }

    }
});// end of create new bill button-----
```

Figure 76: CreateNewBill.java - Continue button code.

However, it was far more challenging to implement the visual indication of which button is selected. We wanted the button to become filled in with the outline colour, while the text and icon turn white – essentially, the colours of the button should invert. We created a new Drawable Resource File (format_button_selected.xml) to serve this purpose. Switching between the designs dynamically during runtime, in response to the user's actions, was the challenging part.

We found from our research that there were a couple of ways that we could achieve the effect we wanted [62]. The first method would be by using XML. This method involves creating a 'selector' XML file which would contain references to different Drawable Resource Files. The selector would be set as the button's background in the layout resource file; the visible background would be set by the selector depending on the state of the button. We implemented a prototype using this method by creating the file button_selector.xml, which can be seen below in figure 77:

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/format_button_selected"
        android:state_pressed="true" android:state_enabled="true" />

    <item android:drawable="@drawable/format_button_outline" android:state_enabled="true" />
</selector>
```

Figure 77: button_selector.xml.

The second method was to implement the effect programmatically, using Java. In this approach, we would directly alter the properties of the button using code inside the CreateNewBill Java class. We implemented a prototype of this method; when the button is clicked, the drawable is switched and the text/icon colours are changed. As seen in figure 78, when the "Equally" button is pressed, its attributes are changed to the 'selected' colours, while the "By Item" button is changed to use the default colours.

```java
// ---- EQUALLY button ----
equallybtn = (Button) findViewById(R.id.equallyButton);
equallybtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //set Booleans
        equallySelected = true;
        byItemSelected = false;

        //set Equally button attributes to selected
        equallybtn.setBackgroundResource(R.drawable.format_button_selected);
        int imgResource = R.drawable.ic_pie_chart_white;
        ((Button)equallybtn).setCompoundDrawablesWithIntrinsicBounds(imgResource, top: 0, right: 0, bottom: 0);
        ((Button)equallybtn).setTextColor(getColor(R.color.white));

        //set By Item button attributes to default
        byitembtn.setBackgroundResource(R.drawable.format_button_outline);
        int imgResource2 = R.drawable.ic_fork_green;
        ((Button)byitembtn).setCompoundDrawablesWithIntrinsicBounds(imgResource2, top: 0, right: 0, bottom: 0);
        ((Button)byitembtn).setTextColor(getColor(R.color.logoGreen));
    }
});
```

Figure 78: CreateNewBill.java - Equally button code.

We elected to use the second method in our source code. There should be no difference in performance for either option, especially given the simple changes we are making [63]. The deciding factor in the end was ease of implementation. Our familiarity with Java (and relative unfamiliarity with XML), coupled with the

directness and effectiveness of method two made it more attractive to us. As there were only 2 states that the buttons could be in, the programmatic approach was feasible, whereas if we had several different states, it could have made our code convoluted and less elegant. In figure 79, you can see the output of our final implementation:

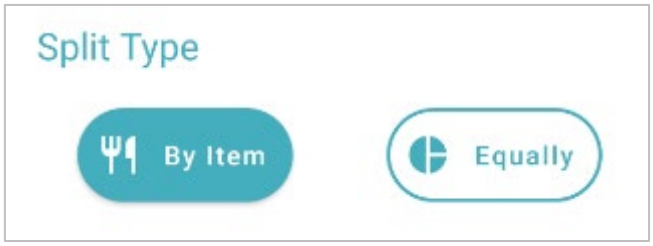


Figure 79: "By Item" split type button selected.

Ethical Audit

"A code of ethics is a guide of principles designed to help professionals conduct business honestly and with integrity" [64]. It outlines how colleagues should behave and approach issues such as safety and conflicts of interest to maintain the organisation's core values and standards [65].

We acted with respect and integrity, and in an ethical manner with colleagues and stakeholders. We acquired practical knowledge and experience regarding the Software Development Life Cycle (SDLC) which includes planning and analysis, design, development, testing, deployment, and maintenance of software applications. By working as a team from the first phase of SDLC, we could deliver our product on time fulfilling specified software requirements. As a group we performed peer review regarding our code structure and documentation to detect and troubleshoot errors. We ensured user requirement specifications were well presented and documented. The design was thoroughly evaluated before the start of the development phase to better understand the target audience's needs, monitor the project's progress towards requirements efficiently and enhance productivity.

We explored several resources like journal papers, articles, books, websites, and blog posts to compare different tools and technologies and gather useful and valid information relevant to Android development. All the references are cited within the source files and the project report. The code of Ethics encourages us "to volunteer and contribute to public education concerning the discipline" [66]. Hence, we decided to make our project public through GitHub and contribute to the open-source community [67].

This application relies on the ability to store user's personal information such as email address and password. This sensitive information must be kept confidential and secure in a database. Therefore, biometric authentication and encryption of user data, as well as extensive testing of the application for vulnerabilities would be the area to focus on if developed further. The Data Protection Act of 2018 states "the information is kept for no longer than is necessary" [68]. Thus, when storing data, these guidelines will be followed.

All user testing was conducted on adults and no minors were involved. While the main target audiences of this application are adults, it can be used by minors, on the condition that they have a children's account.

We have carried out Unit, Integration and User Interface tests for our application to maintain the quality of code and assess the user interface along with its functionality. This helped us to discover and fix defects before the delivery to the end-users and improve the application's quality, performance, and reliability. We have tried to develop safe and reliable software within a reasonable time maintaining the core principle of ethics.

Evaluation

Our Performance

As a group, this was our first-time planning, designing, and developing a software product from start to finish. We had to learn and apply many new processes and concepts as we worked. In this section we will assess how effectively we did this for the various facets of our project.

Planning

On this second part of our software project, our first point of order was to outline a plan on how we would implement everything that we have done in the first part of this project. Equipped with all the management experience gathered during the previous 3 months, we decided to reuse tools that have proven their value, such as Notion, Microsoft Teams and WhatsApp. As we planned the development of the application, we realised that new demands would require new tools therefore we choose Android Studio and GitHub as our main tools alongside those aforementioned.

Despite the use of enhanced planning skills from the first part, we struggled in starting the whole process of development as it was not clear what was the right path to follow. In addition, Android Studio and GitHub soon proved to be high complex tools that demanded wide research so that we would not end up producing a major flaw later. At this point was clear that the group was disoriented but with thorough research, trial and error on the new tools and guidance from our supervisor, we soon mastered these tools and started making progress towards our MVP.

From the forefront of our work in Android Studio, we demonstrated good management and problem anticipation skills by recognizing the importance of android aspects such as API, SDK, and screen size. By tackling this right at the beginning of the development, we prevented major incompatibilities later down the road for both the application and users since we assured that all the features could be accessed by the majority of users with a suitable layout for the average screen size.

Design

We laid out a comprehensive design for our system in our proposal. As we built our project, we found that some of the design choices and decisions we had made were flawed or did not fit our project as well as we had initially believed.

In our proposal, we stated we would be using MongoDB to implement our back-end database. However, during implementation we realised that a non-relational database may not best serve our needs, and that a relational database would be more suitable for our project's purposes. Consequently, we implemented our database using MySQL for the intrinsic benefits of relational databases in conjunction with features like a wide range of data types that would enrich our database, and subsequently n the options of development on the application. Despite our proposal being supported by exhaustive research, upon reaching the implementation stage we realised that we could use a better solution. In order to achieve the highest standards, we had the courage and flexibility to change such a fundamental system element as our application database. Now, in retrospect, we are sure that we made a great decision proved by the ERD and the overall functionality of our application. We do not see this as an error and fixing but rather as a natural improvement from the theorical realm into the tangible world.

Continuing on the subject of changes, we realized that we made an error when it came to our choice of Payment API. In our proposal, we outlined that we would use Noodlio Pay as our payment API and yet during development we found that Noodlio was no longer maintained. In fact, it had not been updated for several years, a key piece of information missed while researching our proposal. This meant we had to decide on an alternative API to use instead; we switched to Stripe – the API that Noodlio is based upon. This extra research took valuable time that could have been used elsewhere. If we had more thoroughly researched Noodlio for our proposal, we may have been able to realise it was unsuitable earlier on.

Although our design was solid for the most part, there were errors that could have been avoided. The payment API is also essential for full functionality and electing to use outdated software was an elementary mistake. Errors are a crucial but natural part of an effective learning so despite being preventable, this mistake has taught us how to perform better research.

System Development

Agile

We adapted well to working in an Agile manner, worked as individuals and sub-teams to produce progress on multiple tasks concurrently. This meant planning, implementation and testing could take place simultaneously, even over separate modules. Our weekly group meetings allowed frequent reviews of the current state of the project, and the opportunity to discuss and react to changing needs or problems encountered; in addition, the weekly meeting with a supervisor helped to attest decisions and reassure that we were on the right path. Between meetings, frequent communication and collaboration amongst group members ensured fluid progress towards our objectives. For this project, we utilize two Agile frameworks: Kanban and Scrum.

Key elements of the Kanban framework were used to good effect. We created a virtual Kanban boards on Notion to keep track of each of our sprints. Through the use of this, we could easily visualise the progress made towards these critical objectives and apply our efforts where they were most needed to keep development moving smoothly.

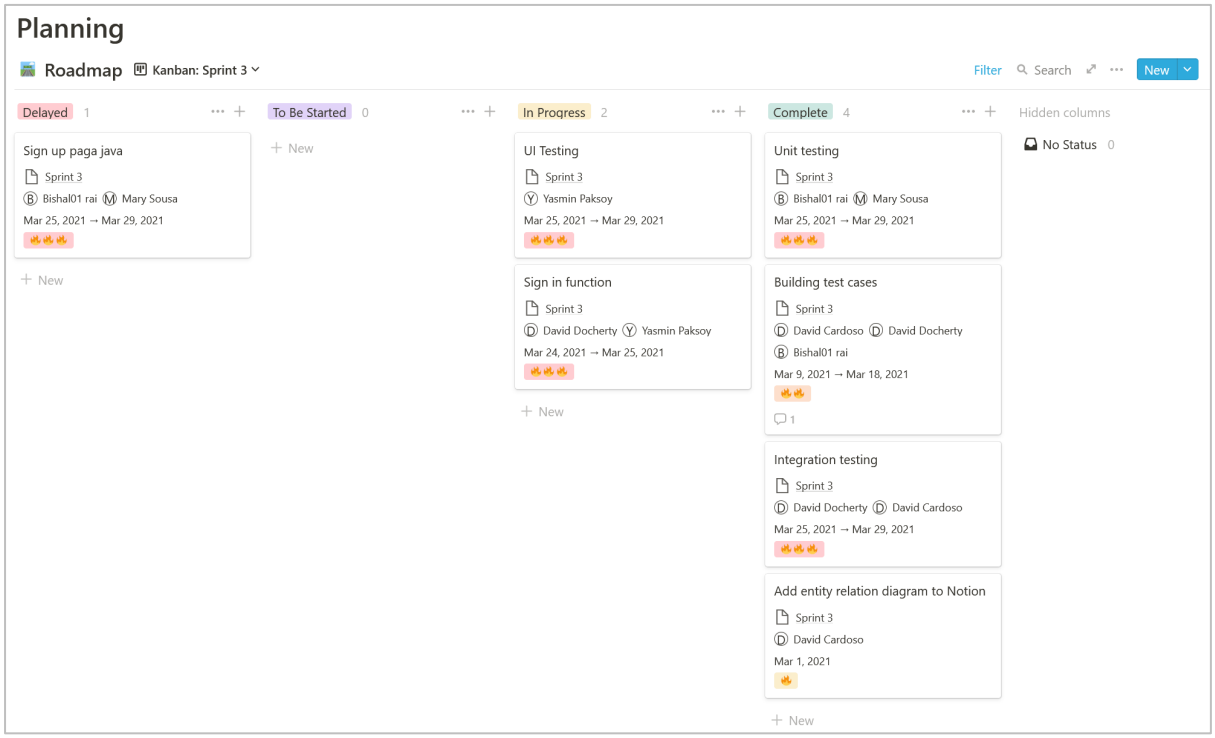
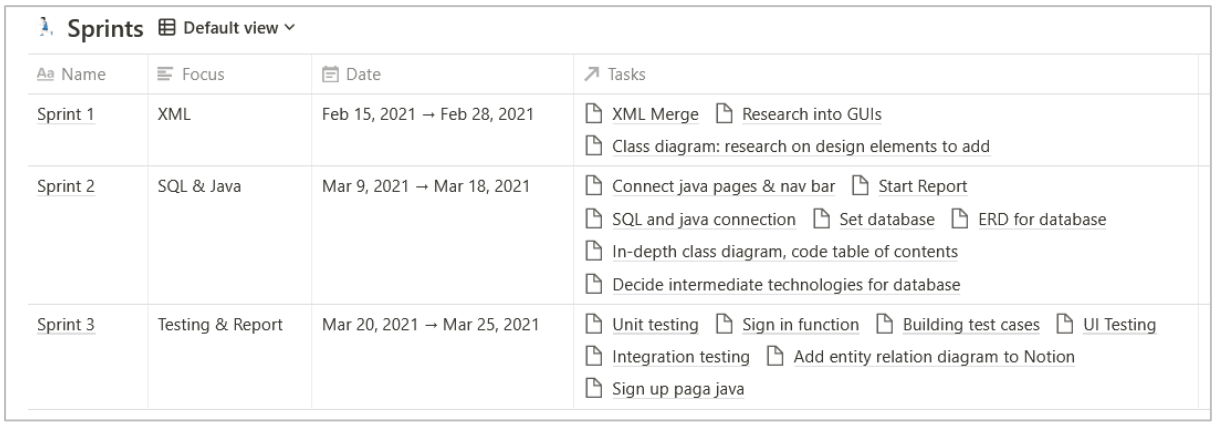


Figure 80: Sprint 3 Kanban board.

However, as time went on, we used the Kanban boards less and less up to the point where they no longer represented accurately the current state of work. The problem here was not the if the framework chosen was fit for the purpose; we did not make the best use of it and hence did not harvest the results we were hoping for.

We made solid use of some Scrum framework features; we used sprints to concentrate on accomplishing important tasks in a fixed timeframe, allowing critical functionalities and dependencies to be established when they were needed. However, we could have done better, particularly in terms of our Sprint reviews. While we would review the work we had completed during the sprint, less focus was put on analysing the sprint process itself. This meant that our sprinting did not noticeably change or improve as the project went on. As a result, we missed out on one of the key benefits to scrum sprints.



Sprints | Default view

Name	Focus	Date	Tasks
Sprint 1	XML	Feb 15, 2021 → Feb 28, 2021	XML Merge, Research into GUIs, Class diagram: research on design elements to add
Sprint 2	SQL & Java	Mar 9, 2021 → Mar 18, 2021	Connect java pages & nav bar, Start Report, SQL and java connection, Set database, ERD for database, In-depth class diagram, code table of contents, Decide intermediate technologies for database
Sprint 3	Testing & Report	Mar 20, 2021 → Mar 25, 2021	Unit testing, Sign in function, Building test cases, UI Testing, Integration testing, Add entity relation diagram to Notion, Sign up paga java

Figure 81: Sprints undertaken, and their corresponding tasks.

We used sprints only at specific times, rather than as our standard work process week-to-week. In 'true' Scrum, sprints are a constant and cyclical method of working at all times. We also failed to make effective use of other tenets of the Scrum framework, such as artifacts and team roles (such as product owner and scrum master). This meant we did not make full use of this framework.

Although we succeeded in generally working according to Agile principles, and reaped most of the according benefits, we struggled to use them to their greatest effects. By using a combination of both Scrum and Kanban frameworks, we gave ourselves access to the strengths of both, but also made it correspondingly more difficult to fully utilise all of tools they provided. Scrum would have produced better results if we had designated a Scrum master, a member responsible for overseeing at all times the current effectiveness of the framework. The reason for the lack of this role was our effort to work always as a group and not having a person overseeing it. It is clear now that a scrum master would improve the overall efficiency of the group, contrary to our stance at the time. Likewise, it is also clear that we neglected the full use of Agile at all times. For instance, despite one of the core strengths of Agile being the constant testing, we only mainly user tested at the very end of the project. Had we kept faithful to Agile core and we would have gathered much more benefits from it.

GitHub/Version Control

The online service GitHub was used for our version control system and source code management needs. There facilities were used effectively for the most part despite the initial struggles. For being an unknown complex tool for every member, our first approach to GitHub was to research how to master it. This task proved to be a real challenge as despite the abundance of information online, the majority of it is either abstract or lacks enough depth from a complete beginner's perspective. As we were fully aware that the incorrect use of this tool would likely produce major setbacks along the development process, the group used a full meeting as a joined learning method where we shared our research findings and tried to perform common tasks in a dummy repository. This provided us with enough knowledge to move into to the next process.

With error prevention as out focus, we developed many features of our application in parallel by using different GitHub branches to implement these in isolation. By

doing so, we preserved each branch from others so possible problems could be dealt with in an isolated environment, our modules were organised, and ensured we had backups of our software; a merge would only happen after both branches were complete. The branching and merging of our units generally went smoothly.

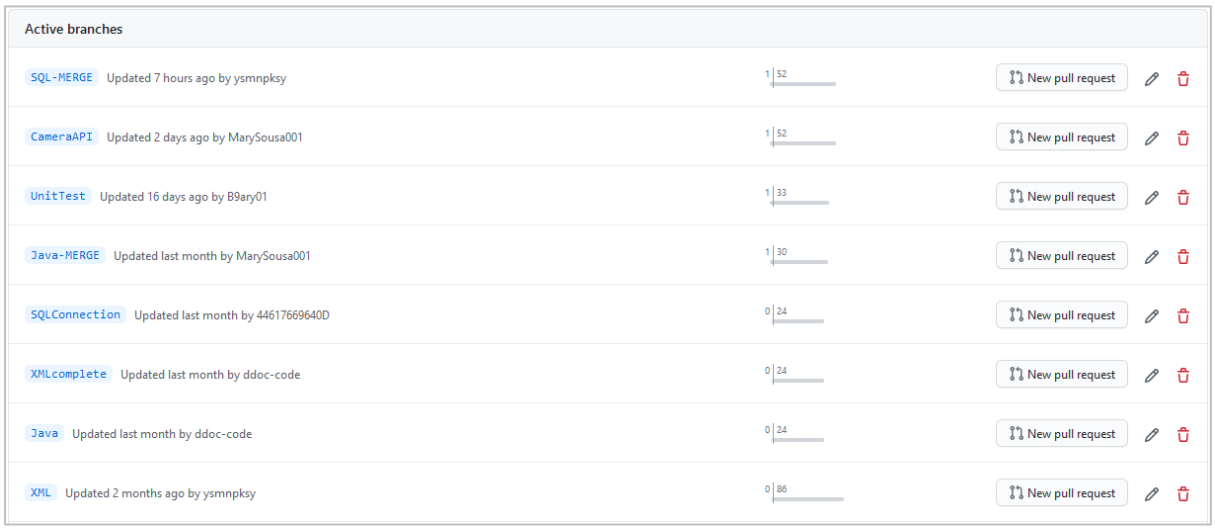


Figure 82: GitHub branches ordered by most recent activity.

Due to our unfamiliarity with version control, we occasionally made mistakes while using this service. These were minor, but hindered development somewhat. Mistakes made included commits being pushed to the wrong branches, different branches being incorrectly merged, and in one instance, a second repo being made meaning we had to work out how to change our remote repo settings. This also unfortunately meant that we were unable to merge our last branch, SQL-MERGE, with the master branch, as they both have different commit histories due to SQL-MERGE being made in a different repository. **Due to this problem, our current working branch is SQL-MERGE.** These errors lead to inefficient use of our limited time. As our familiarity with GitHub increased, these mistakes happened less.

XML

We quickly got to grips with using XML to design layouts for our application. This showed to be an easy task due to the drag and drop feature of Android Studio Here, as a group we pondered how this process would be deployed and debated details such as rules of consistency upon creating activity pages on Android Studio. This proves, once more, our commitment for efficient productivity in a methodical and reliable way.

In the interest of efficiency, we created resource files to contain elements that we used very commonly. This allowed us to use a simple reference to these files each time we needed them, rather than having to implement them multiple times in different files. In addition, we created template files for buttons also for the same reasons: efficiency and creativity in problem solving, characteristic of developers. This let us save time and ensured high consistency between different modules of our application. By doing this, we demonstrated good planning, error prevention and proactivity in efficiency, enhancing our work method as developers.

Structure

The structure of our code was generally good, although it took us some time to become accustomed to working in Android Studio. Android development is complex, and something that was new to all of our group members. We wrote our filenames in a clear and consistent format, and as a result it is easy to tell which XML files link to which Java classes. For example, the Java class "Your_Account.java" refers to the XML layout file "activity_your__account.xml" for its content view. We also did our best to observe good coding conventions such as proper casing and ensured that files were well commented to aid comprehension. Overall, we did well at working in a proactive way to help avoid problems later on.

We implemented our code in such a way as to minimise the number of bugs we produced. This took the form of testing as we coded to ensure our execution achieved the goals we had in mind. We also created templates for frequently used components such as buttons, which could be adapted to use when implementing new units. This had the effect of increasing consistency and decreasing the chances of errors being made by coding the same block of code over and over again. Along with the validation we implemented to control the input the application receives, our stringent standards helped ensure our application was resistant to error states.

Some parts of our structure are less ideal. Some comments and variable names have small misspellings or incorrect casing issues, but these are not severe enough to interfere with the code's readability. Some filenames have similar problems. For example, several of our XML files, including the above mentioned "activity_your__account.xml", have extra underscores used to separate two words. Although a very minor oversight, this does mean our XML filenames have a less consistent format, with some names being slightly different than you might expect.

Testing

When testing we were thorough, comprehensive, and methodical. Our testing was done at multiple levels to fully assess our application as it was built; both functional and non-functional tests have been performed. The former measured how well our application met the functional requirements we had laid out in our proposal, whilst the latter was used to validate aspects that lay outside the functional scope. Doing this ensured we had a good overview over all facets of our projects progress, including non-technical parameters such as usability.

In keeping with our Agile methodology, testing was done iteratively. Results from our tests was incorporated into the next version of our product. This not only improved our application but helped us to keep useability at the forefront of our development process. Different levels of testing were carried out by different members of our group. This gave us a great deal of flexibility in when tests were performed. Nevertheless, we made mistakes. Not only were we meant to test throughout the whole development, as supported by Agile, but we also should have tested more. Additionally, our user tests were limited due to restrictions imposed by the current pandemic situation. If not for this, we would have

conducted a more expressive number of user testing, in person and at a distance as to cover every scenario as much as possible.

Technical Analysis

JDBC

Our efforts to successfully establish a connection using JDBC were a major focus during development. This middleware was essential to link our front-end application to our back-end database. We found it challenging to implement this. On several occasions we encountered difficulties that slowed our progress, and our Agile work processes aided us in beating these issues.

For example, when a sub-group was struggling to make progress on this task, we reassigned it to another sub-team. The new team could review it with fresh eyes, while the old group could refocus their skills in other areas. This prevented bottlenecks from forming in our workflow, and by doing this we showed a good comprehension of task management and prioritisation.

Building this component iteratively helped ensure we had a good understanding of how it worked and could systematically eliminate possible causes of the problems we faced. Only when we had ruled out all potential errors on our part and realised the probable cause did we approach the Goldsmiths database administrator, to confirm our suspicions and learn how to implement the intended method that should be used to connect. With his advice, we were able to successfully establish the link. Our methodical approach here ensured we made the best use of both our time, and that of others.

This was definitely the biggest challenge of our project. To solve it we applied our knowledge of databases, networks, and Java in conjunction with problem solving. Without this connection the whole concept of the application would become invalid therefore we used all our skills and time in order to be successful. For this, we believe that we have surpassed ourselves and enhanced our technical and soft skills that we will use in the future.

Handling Merges

For being a group using Agile methodology, we end up having a merge as a result of a sprint creating XML files in subgroups. At this point, decisions made previously while planning showed to be decisive between a good and bad outcome. Due to our methodical way of coding and implementing, we expected to have a smooth merge with only a few aspects to resolve. However, the merge encountered 70+ errors. Although each subbranch worked on its own, when merged they broke. This can be read as small problems pilling up until becoming big enough to inhibit correct functioning. Our solution ended up being a manual merge which required more time but ended up better suiting the purpose.

Overall, our technical skills have been challenged beyond expected with some wins and some defeats. Moving forward, we will retain the knowledge we gained by learning from our mistakes and will know what to do differently.

Teamwork

Since the very beginning of this project, we have built a strong sense of team that translated into this development phase as we did not experience the natural constrains of total unknown people working together for the first time. The first weeks of this second part were especially hard since we struggled to find the best approach to pursue our goal: deliver the best product that we could. Throughout all these weeks and steps, three key aspects were present: time management, communication, and productivity.

Time management

Time management is a very important factor that contributes when delivering a product. To keep track of our overall progress, we used a Gantt chart. Unlike our proposal where we used the Gantt chart actively, in the second part of the project we ended up undervaluing it. This important tool was reviewed and updated only a few times, far from the regularity that it should have been used with. As we focused more on solving intricate issues such as JDBC, we did not realise that important management aspects such as the Gantt chart were being left on hold. Even so, analysing retrospectively we would have reached most of our internal deadlines if not for the initial struggle of finding the correct start and the JDBC issue.

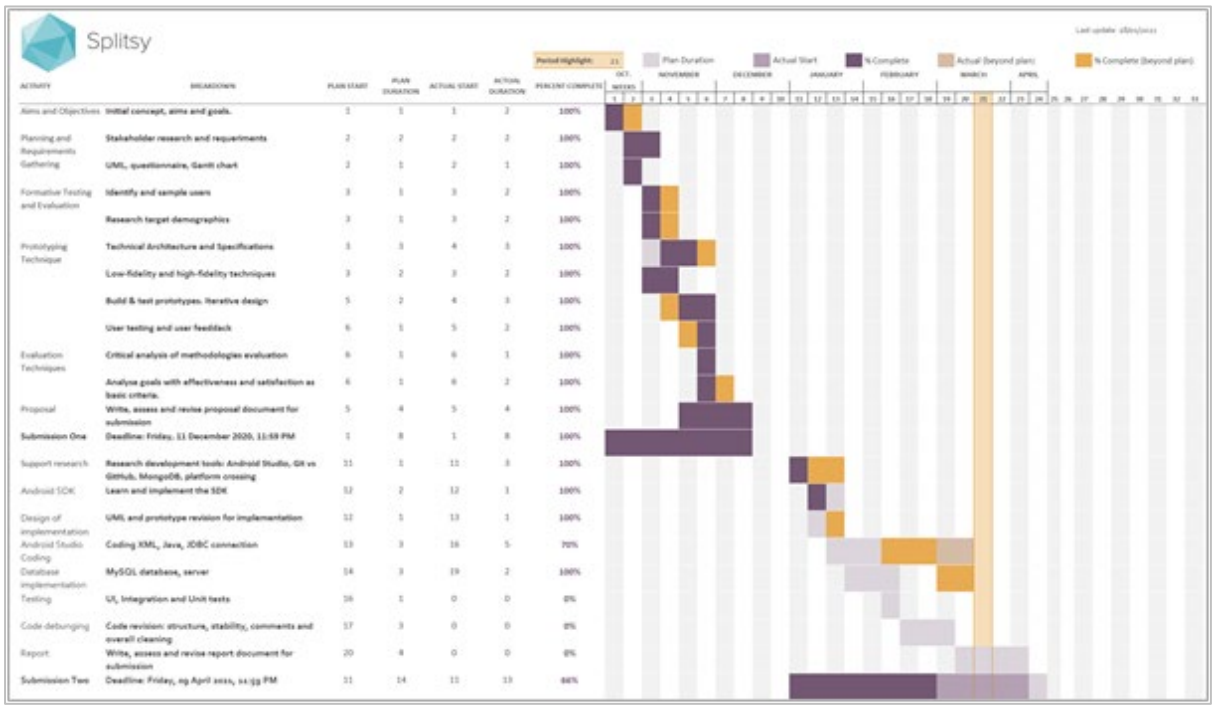


Figure 83: Updated Gantt chart, dated 17/03/21.

Our regular meetings and reviews allowed us ample opportunities to check our progress was proceeding as planned. As mentioned before, our Agile workflow meant we ensured that tasks that were proving difficult were focused on early, rather than allowing them to become bottlenecks. To keep track of our individual time committed to the project, we created time trackers on Notion where group

members could record the hours they worked each week. These can be seen below:

Mary

Time Tracker

Week	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
Week 12	3	2.25	5.25	7	1.75	No
Week 13	4	5	9	7	-2	Yes
Week 14	3	6.5	9.5	7	-2.5	Yes
Week 15	5	3.25	8.25	7	-1.25	Yes
Reading Week	12	3	15	7	-8	Yes
Week 16	4	2	6	7	1	No
Week 17	6	3	9	7	-2	Yes
Week 18	10	1.5	11.5	7	-4.5	Yes
Week 19	16	3.75	19.75	7	-12.75	Yes
Week 20	14	2.5	16.5	7	-9.5	Yes
Week 21	14	3.5	17.5	7	-10.5	Yes
Week 22	14	8	22	7	-15	Yes
Week 23		1.5	1.5		-1.5	Yes

Figure 84: Mary Sousa time record table

Rai

Time Tracker

Week	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
Week 12	6	2.25	3	7	4	No
Week 13	8	5	3	7	4	No
Week 14	5	6.5	3	7	4	No
Week 15	7	3.25	3	7	4	No
Reading Week	8	3	3	7	4	No
Week 16	5	2	3	7	4	No
Week 17	6	3	3	7	4	No
Week 18	3	1.5	3	7	4	No
Week 19	4	3.75	3	7	4	No
Week 20	5	2.5	3	7	4	No
Week 21	6	3.5	3	7	4	No
Week 22	7	8	3	7	4	No
Week 23	7	1.5	3	7	4	No

Figure 85: Bishal Rai time record table

David

Time Tracker

Week	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
18/01 - 24/01	1	2.25	3.25	7	3.75	No
25/01 - 31/01	2	5	7	7	0	Yes
01/02 - 07/02		6.5	6.5	7	0.5	No
08/02 - 14/02	2	3.25	5.25	7	1.75	No
15/02 - 21/02	9.5	3	12.5	7	-5.5	Yes
22/02 - 28/02	5	2	7	7	0	Yes
01/03 - 07/03	7	3	10	7	-3	Yes
08/03 - 14/03	3	1.5	4.5	7	2.5	No
15/03 - 21/03	8	3.75	11.75	7	-4.75	Yes
22/03 - 28/03	6	2.5	8.5	7	-1.5	Yes
29/03 - 04/04	4	3.5	7.5		-7.5	Yes
05/04 - 11/04	13	8	21		-21	Yes
12/04 - 18/04	20	1.5	21.5		-21.5	Yes

+ New

Figure 86: David Cardoso time record table

Yasmin

Time Tracker

Week	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
Week 12	1.5	2.25	3.75	7	3.25	No
Week 13	1	5	6	7	1	No
Week 14	2	6.5	8.5	7	-1.5	Yes
Week 15	2	3.25	5.25	7	1.75	No
Reading Week	10	3	13	7	-6	Yes
Week 16	6	2	8	7	-1	Yes
Week 17	6	3	9	7	-2	Yes
Week 18	7.5	1.5	9	7	-2	Yes
Week 19	10.5	3.75	14.25	7	-7.25	Yes
Week 20	7	2.5	9.5	7	-2.5	Yes
Week 21	10	3.5	13.5	7	-6.5	Yes
Week 22	15	8	23	7	-16	Yes
Week 23	20	1.5	21.5	7	-14.5	Yes

+ New

Figure 87: Yasmin Paksoy time record table

Dave

Time Tracker

Week	Database ...	Individual Time	Meeting Time	Total Time	Goal	Time Left	Goal Reached
01/03 - 07/03	Week 17	6	3	9	7	-2	Yes
08/03 - 14/03	Week 18	8	1.5	9.5	7	-2.5	Yes
15/03 - 21/03	Week 19	3	3.75	6.75	7	0.25	No
22/03 - 28/03	Week 20	4	2.5	6.5	7	0.5	No
29/03 - 04/04	Week 21	3.5	3.5	7	7	0	Yes
05/04 - 11/04	Week 22	2	8	10	7	-3	Yes
12/04 - 18/04	Week 23	7	1.5	8.5	7	-1.5	Yes

Figure 88: David Docherty time record table

We aimed to spend 7 hours a week working on the project per individual. As you can see from our data, in aggregate we achieved this goal 35 out of 59 possible times. This translates into a success rate of 59.32%. While this figure is not bad, it could be higher. The most likely cause for this were factors external to the project that consumed an unexpected amount of time and resources. Despite this, we believe that we achieved a solid rate, and this is shown in our stable workflow and meticulous record-keeping.

Communication

Communication is key in project development – particularly in Agile. One of the twelve principles outlined in *The Manifesto for Agile Software Development* is "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation" [69]. Obviously during a global pandemic, face-to-face communication is disincentivised. Nevertheless, we have done our best to work as closely to it as currently possible.

Our group meetings, as well as most of our collaborative work, took place using Microsoft Teams. This allowed interaction that is virtually face-to-face, granting us the benefits that come with quick and clear communication. We also used the proposal's WhatsApp group to stay in contact at all times for small doubts. Between these methods, we were able to connect in almost any way that would be possible (for example) in an office. The good availability of all our members meant we could 'drop in' on each other whenever needed, to ask questions or request advice.

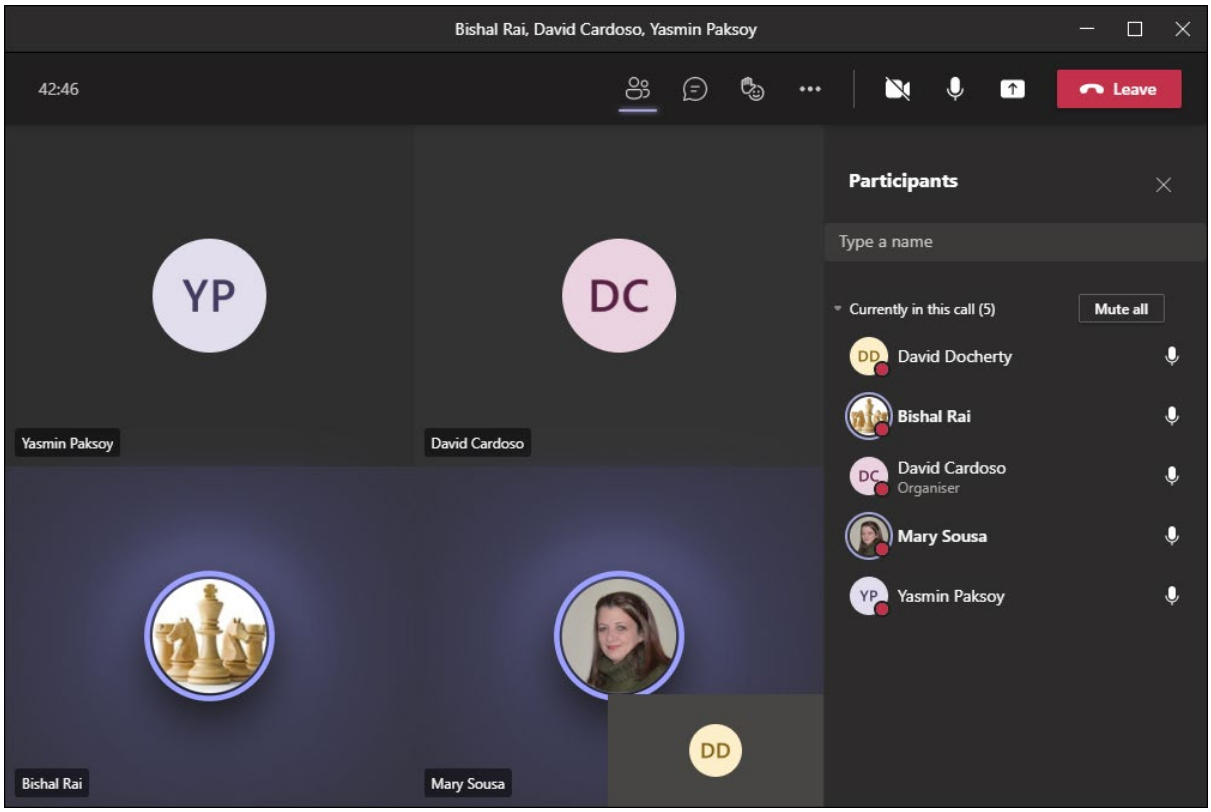


Figure 89: Microsoft Teams meeting in progress.

Meetings took place regularly and began promptly. The attendance of all our group members was excellent, and everyone was present for the vast majority of meetings. In the meetings, all members were able to give and receive advice in equal measure. In the interest of teambuilding, social conversations were encouraged after the project topics were addressed which proved to build up the team connection and morale.

Our communication could have been superior in some respects. Our main flaws were reluctance in asking help in a reasonable time without dragging the issue beyond expected; hiding minor errors that could pile up into a major problem or accountability. Despite this, the group have shown empathy, professionalism and approachability which always resulted in reaching common ground, help being provided and the group moving forward.

Productivity

Productivity is closely linked with Project Management, and just as important. As our Project Management was excellent, group members had the tools they needed to foster high productivity. The great availability and effort given by team members meant that help was always available when needed. Combined with our Agile work methodology, this meant that we had the flexibility to adapt to obstacles and overcome difficult challenges. This allowed a good pace of constant progress which was never stymied for long. Furthermore, the preceding factors worked to ensure morale stayed high, which also promoted good productivity.

Overall, it became evident early on that we aimed too high with the MVP. Yet, we did not tremble and choose the easy path by downgrading our product into a much

easier outcome. Instead, we kept our MVP, focused our assets and efforts on what we have defined as the main needs and we tried our best to deliver the best product that we could have produced.

Our Product

Overall, we had a very ambitious MVP; during the planning phase of this project, we knew that implementing each aspect of our MVP would be difficult but believed it to be possible assuming we did not run into any problems. Unfortunately, as this was not the case, we have not been able to implement every aspect of our MVP.

Our current application includes allowing users to sign up and sign in using an email and password. Users are able to enter data and navigate around the application using on screen buttons and the navigation bar. The GUI has been implemented perfectly, following the high-fidelity prototype as closely as possible, and so the visual appeal of the application is very pleasing.

Being able to implement the UI by replicating our high-fidelity prototype has been extremely valuable since we had chosen the element sizes, font sizes and added descriptive icons in order to ensure that our application is accessible for users with disabilities. Using a constraint layout when implementing the UI has led to an overall respectable product, as potential users will not face issues due to their screen size.

Most inoperative features of our MVP can now be easily implemented, as the major components that were necessary to do so have been correctly set up. For example, the back-end database would be used for 8 out of the 11 tasks detailed as part of our MVP. Unfortunately, setting up the database took up a majority of our time, but since it has been fully implemented moving forward with coding the processes that implement it would be much more uncomplicated.

Nevertheless, our current product sadly does not achieve the main goal of our application; users are unable to split bills. The most crucial aspects to make this happen have still not been fully implemented: the camera and payment APIs. While this kind of software would not be practical to solve real-world problems, we are certain that we have built a strong foundation for further development of the application.

Moving forward, we would like to conclude implementing the features we outlined as part of our MVP while conducting more user testing to ensure the quality of our product, which would lead to a fully functioning product. There are other key aspects that are also missing; plain passwords are stored in our database and there is no encryption process when saving and retrieving data.

There are many features we outlined in our proposal that were not a part of our MVP that we would also like to implement. These includes features such as biometric login, high contrast mode, dark mode and allowing users to export all their data. User testing also gave us many suggestions that would help take our application to the next level; allowing users to sign up using their Google account, for example, would lead to a more sophisticated product.

Conclusion

All things considered, the group has used a distinct range of methodologies and tools to produce a client-based product of the proposed application. In just 6 months, while not working exclusively on this project, our group assembled and enhanced the technical and soft skills essential to purpose, develop and deliver a functioning Android application connected to an online database. This was achieved via appropriate planning, design, and system development though Agile while always looking for improvements in order to reach the closest as possible to our MVP. Even though we failed to fully achieve our MVP and this project pushed us out of our comfort zone, in the end we have grown as developers and every single member of this group stands by the final product with a feeling of pride and accomplishment.

User Guide

Please refer to file Splitsy User guide - ver1.0.1.2021

Online version: https://issuu.com/splitsy/docs/splitsy_-_user_guide.issu

Glossary

Agile: Used for describing ways of planning and doing work in which it is understood that making changes as they are needed is an important part of the job [70].

Android: An operating system used mainly for mobile devices that you control by touching the screen [71].

Android Studio: Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development [72].

API: Abbreviation for application programming interface: a set of rules that defines how a software program can request and receive information from other software, usually a website [73].

APK: An Android Package Kit (APK for short) is the package file format used by the Android operating system for distribution and installation of mobile apps [74].

Branch: A branch in Git is a way to keep developing and coding a new feature or modification to the software and still not affecting the main part of the project [75].

Emulator: An emulator is a hardware device or software program that enables one computer system (also known as a host) to imitate the functions of another computer system (known as the guest) [76].

Git: Git is a content management and tracking system developed by Linus Torvalds, creator of Linux. It includes a directory that continuously changes as codes are added throughout application or website development. Git also tracks revisions that are performed on stored data [77].

GitHub: GitHub is an open source, cloud-based repository hosting service that allows developers to store, manage, track, and control changes to their code [78].

Java: A cloud-based, object-oriented programming language that is designed to have as few implementation dependencies as possible [79].

JDK: The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development [80].

Kanban: The Kanban Method is a means to design, manage, and improve flow systems for knowledge work. The method also allows organizations to start with their existing workflow and drive evolutionary change [81].

Kotlin: Kotlin is an open source object-oriented and functional programming language that's often suggested as an alternative to JavaScript. Some describe it as a ‘streamlined’ version of Java [82].

Manifest: Every application project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set. The manifest file describes essential information about the application to the Android build tools, the Android operating system, and Google Play [83].

Merge: Merge is the process of combining the various versions of a file or folder. This feature is typically found in version control software as a fundamental operation that is responsible for reconciliation of changes of data in a file [84].

Minimum viable product (MVP): A minimum viable product is a version of a product with just enough features to be usable by early customers who can then provide feedback for future product development [85].

MongoDB: MongoDB is a document database which stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time [86].

MySQL: MySQL is a full-featured relational database management system (RDBMS) that competes with the likes of Oracle DB and Microsoft's SQL Server [87].

Repository: Repositories in GIT contain a collection of files of various different versions of a Project. These files are imported from the repository into the local server of the user for further updates and modifications in the content of the file [88].

Scrum: Scrum is a process framework used to manage product development and other knowledge work. Scrum is empirical in that it provides a means for teams to establish a hypothesis of how they think something works, try it out, reflect on the experience, and make the appropriate adjustments [89].

SDK: A software development kit (SDK) is a set of tools used for developing applications provided by hardware and software providers. SDKs are usually comprised of application programming interfaces (APIs), sample code, documentation, etc [90].

SDLC: SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace, and alter or enhance specific software [91].

SQL: A query language used for accessing and modifying information in a database. Some common SQL commands include "insert," "update," and "delete" [92].

XML: Stands for "Extensible Markup Language." XML is used to define documents with a standard format that can be read by any XML-compatible application [93].

References

[1] Android Developers. (2020). | Android Developers. [online] Available at: https://developer.android.com/guide/topics/manifest/uses-sdk-element#ApiLevels. [Accessed 10 Apr. 2021].

[2] Medium. (n.d.). Medium. [online] Available at: https://medium.com/androiddevelopers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd. [Accessed 10 Apr. 2021].

[3] StatCounter Global Stats. (2019). Mobile & Tablet Android Version Market Share Worldwide | StatCounter Global Stats. [online] Available at: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide. [Accessed 10 Apr. 2021].

[4] Android Developers. (2019b). Support different screen sizes | Android Developers. [online] Available at: https://developer.android.com/training/multiscreen/screensizes. [Accessed 13 Apr. 2021].

[5] StatCounter Global Stats. (n.d.). Screen Resolution Stats Worldwide. [online] Available at: https://gs.statcounter.com/screen-resolution-stats#monthly-202004-202103. [Accessed 16 Apr. 2021].

[6] DeviceAtlas. (2019b). Viewport, resolution, diagonal screen size and DPI for the most popular smartphones. [online] Available at: https://deviceatlas.com/blog/viewport-resolution-diagonal-screen-size-and-dpi-most-popular-smartphones. [Accessed 13 Apr. 2021].

[7] Techopedia.com. (2019). What is a Minimum Viable Product (MVP)? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp. [Accessed 6 Apr 2021].

[8] digitalcommunications.wp.st-andrews.ac.uk. (2021). What is a minimum viable product? | Digital Communications team blog. [online] Available at: https://digitalcommunications.wp.st-andrews.ac.uk/2021/04/08/what-is-a-minimum-viable-product/. [Accessed 15 Apr. 2021].

[9] techterms.com. (n.d.). XML (Extensible Markup Language) Definition. [online] Available at: https://techterms.com/definition/xml. [Accessed 2 Feb. 2021].

[10] Android Authority. (2019). An introduction to XML for new Android developers – the powerful markup language. [online] Available at: https://www.androidauthority.com/introduction-to-xml-968598/. [Accessed 2 Feb. 2021].

[11] beginnersbook.com. (2018). Advantages and Disadvantages of XML. [online] Available at: https://beginnersbook.com/2018/10/advantages-and-disadvantages-of-xml/. [Accessed 2 Feb. 2021].

[12] Android Developers. (2019). Support different screen sizes | Android Developers. [online] Available at:

https://developer.android.com/training/multiscreen/screensizes. [Accessed 2 Feb. 2021].

[13] MongoDB. (2020). Mongodb Vs Mysql. [online] Available at: https://www.mongodb.com/compare/mongodb-mysql. [Accessed 10 December 2020].

[14] MongoDB. (2019). MongoDB and MySQL Compared. [online] Available at: https://www.mongodb.com/compare/mongodb-mysql. [Accessed 2 Apr. 2021].

[15] Dev.mysql.com. 2021. MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL?. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Accessed 16 Apr. 2021].

[16] Smartdraw.com. 2021. Entity Relationship Diagram (ERD) - What is an ER Diagram?. [online] Available at: https://www.smartdraw.com/entity-relationship-diagram/. [Accessed 16 Apr. 2021].

[17] www.tutorialspoint.com. (n.d.). JDBC Tutorial - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/jdbc/. [Accessed 11 Marc 2021].

[18] noodlio (n.d.). Noodlio Pay - Smooth Payments with Stripe API: How To Use the API. [online] RapidAPI. Available at: https://rapidapi.com/noodlio/api/noodlio-pay-smooth-payments-with-stripe/details [Accessed 13 Apr. 2021].

[19] Aryan (2020). What exactly does Stripe do? A Beginners Guide on Stripe Payments. [online] Subscription Management & Recurring Billing Software for your Business. Available at: https://payvoice.io/what-exactly-does-stripe-do-a-beginners-guide-on-stripe-payments. [Accessed 13 Apr. 2021].

[20] Strojny, D. (2016). Stripe vs PayPal: Who should you choose? [online] Memberful. Available at: https://memberful.com/blog/stripe-vs-paypal/. [Accessed 13 Apr. 2021].

[21] Kinsta. (2019). *Stripe vs Square: Which Payment Gateway Should You Use in 2021?* [online] Available at: https://kinsta.com/blog/stripe-vs-square/#:~:text=Stripe%20was%20founded%20in%202011. [Accessed 6 Apr. 2021].

[22] Android Developers. (n.d.). Android 11—Week 6—Jetpack. [online] Available at: https://developer.android.com/courses/pathways/android-week6-jetpack. [Accessed 13 Apr. 2021].

[23] xda-developers. (2020). How to find the Android Version Distribution statistics in Android Studio. [online] Available at: https://www.xda-developers.com/android-version-distribution-statistics-android-studio/. [Accessed 9 Apr. 2021].

[24] Android Developers. (n.d.). *CameraX overview*. [online] Available at: https://developer.android.com/training/camerax. [Accessed 1 Apr. 2021].

[25] Poetker, B. (2019). How to Use Portrait Mode on iPhone and Android. [online] learn.g2.com. Available at: https://learn.g2.com/portrait-mode#:~:text=Portrait%20mode%20is%20a%20phone. [Accessed 13 Apr. 2021].

[26] Cervantes, E. (2021). What is HDR in photography? Let us teach you how it's done. [online] Android Authority. Available at: https://www.androidauthority.com/what-is-hdr-photography-1021421/. [Accessed 13 Apr. 2021].

[27] Vicente, V. (2021). How Does "Night Mode" Work on Smartphone Cameras? [online] How-To Geek. Available at: https://www.howtogeek.com/702941/how-does-night-mode-work-on-smartphone-cameras/. [Accessed 13 Apr. 2021].

[28] DataFlair. (2020). Android Studio Vs Eclipse - Which is Better for Android Developers? [online] Available at: https://data-flair.training/blogs/android-studio-vs-eclipse. [Accessed 6 Apr. 2021].

[29] TrustRadius. (n.d.). Android Studio vs IntelliJ IDEA. [online] Available at: https://www.trustradius.com/compare-products/android-studio-vs-intellij-idea. [Accessed 6 Apr. 2021].

[30] Android Developers. (2019). Meet Android Studio | Android Developers. [online] Available at: https://developer.android.com/studio/intro. [Accessed 4 Apr. 2021].

[31] Android Developers. (2019). Android Studio features | Android Developers. [online] Available at: https://developer.android.com/studio/features. [Accessed 4 Apr. 2021].

[32] Vittana. 2020. 15 Advantages and Disadvantages of a Waterfall Model. [ONLINE] Available at: https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model. [Accessed 22 Jan. 2021].

[33] Anon, 2020. 5 benefits of agile project management you must know about. Apiumhub. Available at: https://apiumhub.com/tech-blog-barcelona/benefits-of-agile-project-management/. [Accessed 11 Dec. 2020].

[34] Alexandra Altvater, 2017. What is Agile Methodology? How It Works, Best Practices, Tools. [online] Stackify. Available at: https://stackify.com/agile-methodology/. [Accessed 22 Jan. 2021].

[35] Anon, 2019. Scrum. Agile Alliance. Available at: https://www.agilealliance.org/glossary/scrum/. [Accessed 6 Dec. 2020].

[36] Anon, 2019. What is Kanban? Agile Alliance. Available at: https://www.agilealliance.org/glossary/kanban/. [Accessed 6 Dec. 2020].

[37] Jackson, B. (2018). What Is GitHub? A Beginner's Introduction to... [online] Kinsta Managed WordPress Hosting. Available at: https://kinsta.com/knowledgebase/what-is-github/. [Accessed 22 Jan. 2021].

[38] Mutabazi, P. (2018). What Do the 16 digits Printed on a Debit Card Mean to You? [online] Available at: https://www.linkedin.com/pulse/what-do-16-digits-printed-debit-card-mean-you-patrick-mutabazi/. [Accessed 16 Apr. 2021].

[39] Android Developers. (n.d.). Specify the input method type. [online] Available at: https://developer.android.com/training/keyboard-input/style. [Accessed 16 Apr. 2021].

[40] www.guru99.com. (n.d.). Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE. [online] Available at: https://www.guru99.com/unit-testing-guide.html. [Accessed 10 Apr. 2021].

[41] Android Developers. (n.d.). Build local unit tests. [online] Available at: https://developer.android.com/training/testing/unit-testing/local-unit-tests. [Accessed 10 Apr. 2021].

[42] www.guru99.com. (n.d.). Junit Assert & AssertEquals with Example. [online] Available at: https://www.guru99.com/junit-assert.html. [Accessed 10 Apr. 2021].

[43] Pfaff, F. (2012). Unit tests with Mockito - Tutorial. [online] Vogella.com. Available at: https://www.vogella.com/tutorials/Mockito/article.html. [Accessed 10 Apr. 2021].

[44] Edureka. (2019). What is Integration Testing? | How to perform integration testing? [online] Available at: https://www.edureka.co/blog/what-is-integration-testing-a-simple-guide-on-how-to-perform-integration-testing/. [Accessed 7 Apr. 2021].

[45] AI-driven E2E automation with code-like flexibility for your most resilient tests. (2020). UI Testing: A Beginner's Guide With Checklist and Examples. [online] Available at: https://www.testim.io/blog/ui-testing-beginners-guide/. [Accessed 11 Apr. 2021].

[46] Omniconvert. (2019). What is User testing? Definition - Omniconvert. [online] Available at: https://www.omniconvert.com/what-is/user-testing/. [Accessed 11 Apr. 2021].

[47] Savvy Apps. (2018). Comprehensive Guide to App User Testing. [online] Available at: https://savvyapps.com/blog/how-to-user-testing-your-app. [Accessed 11 Apr. 2021].

[48] AGConsult. (2016). User testing: what, why and how? Everything you need to know about user testing on websites. [online] Available at: https://www.agconsult.com/en/usability-blog/user-testing-what-why-and-how/. [Accessed 12 Apr. 2021].

[49] The Custom Droid. (2020). Android 101: How to Install an APK on Android (Sideloading Apps). [online] Available at: https://www.thecustomdroid.com/how-to-install-apk-on-android/. [Accessed 12 Apr. 2021].

[50] Stack Overflow. (n.d.). android - Install custom .apk without adb/Eclipse. [online] Available at: https://stackoverflow.com/questions/6527579/install-custom-apk-without-adb-eclipse. [Accessed 12 Apr. 2021].

[51] UserTesting. (n.d.). Moderated Vs. Unmoderated Usability Testing: The Pros and Cons | UserTesting Blog. [online] Available at: https://www.usertesting.com/blog/moderated-vs-unmoderated-usability-testing. [Accessed 12 Apr. 2021].

[52] careerfoundry.com. (n.d.). Why Consistency Is So Incredibly Important In UI Design. [online] Available at: https://careerfoundry.com/en/blog/ui-design/the-importance-of-consistency-in-ui-design/. [Accessed 16 Apr. 2021].

[53] Singh Gill, N., 2021. Kotlin vs Java Comparison - Which One is Better? (2020). [online] XenonStack. Available at: https://www.xenonstack.com/blog/kotlin-andriod. [Accessed 6 Apr. 2021].

[54] Ababei, A. (2018). 8 Reasons to Use Kotlin Over Java for Android Development - DZone Java. [online] Available at: https://dzone.com/articles/what-are-the-biggest-advantages-of-kotlin-over-jav. [Accessed 6 Apr. 2021].

[55] ASH (2018). Android Connect MySQL Database Programmatically. [online] ParallelCodes. Available at: https://parallelcodes.com/android-connect-mysql-database/. [Accessed 17 Mar. 2021].

[56] EDUCBA. (2019). MySQL vs SQLite | Top 14 Differences You Should Learn. [online] Available at: https://www.educba.com/mysql-vs-sqlite/. [Accessed 17 Mar. 2021].

[57] Stack Overflow. (n.d.). java - Android JDBC not working: ClassNotFoundException on driver. [online] Available at: https://stackoverflow.com/questions/7221620/android-jdbc-not-working-classnotfoundexception-on-driver/7221716#7221716. [Accessed 17 Mar. 2021].

[58] www.tutorialspoint.com. (n.d.). Android Tutorial - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/android[-asynctask-example-and-explanation. [Accessed 17 Mar. 2021].

[59] code.luasoftware.com. (n.d.). Android BottomNavigationView: Change Icon Size. [online] Available at: https://code.luasoftware.com/tutorials/android/android-bottomnavigationview-change-icon-size/. [Accessed 1 Apr. 2021].

[60] GeeksforGeeks. (2020). Bottom Navigation Bar in Android. [online] Available at: https://www.geeksforgeeks.org/bottom-navigation-bar-in-android. [Accessed 7 Apr. 2021].

[61] Suragch (2019). Using dimens.xml in Android. [online] Medium. Available at: https://suragch.medium.com/using-dimens-xml-in-android-10dec2fe485c. [Accessed 9 Apr. 2021].

[62] Stack Overflow. (n.d.). onclick - Make an Android button change background on click through XML. [online] Available at: https://stackoverflow.com/questions/4125774/make-an-android-button-change-background-on-click-through-xml. [Accessed 23 Mar. 2021].

[63] Stack Overflow. (n.d.). java - Android - Layouts performance: Programmatic vs XML. [online] Available at: https://stackoverflow.com/questions/35568058/android-layouts-performance-programmatic-vs-xml. [Accessed 23 Mar. 2021].

[64] Hayes, A. (2020). Code of ethics definition. [online] Investopedia. Available at: https://www.investopedia.com/terms/c/code-of-ethics.asp. [Accessed 11 Apr. 2021].

[65] Betterteam. (n.d.). Code of Ethics. [online] Available at: https://www.betterteam.com/code-of-ethics. [Accessed 11 Apr. 2021].

[66] ACM Ethics. (2018). Software Engineering Code. [online] Available at: https://ethics.acm.org/code-of-ethics/software-engineering-code/. [Accessed 11 Apr. 2021].

[67] ysmnpksy (2021). ysmnpksy/Splitsy. [online] GitHub. Available at: https://github.com/ysmnpksy/Splitsy. [Accessed 17 Apr. 2021].

[68] GOV.UK. (n.d.). Data protection. [online] Available at: https://www.gov.uk/data-protection. [Accessed 11 Apr. 2021].

[69] Agilemanifesto.org. (2001). Principles behind the Agile Manifesto. [online] Available at: https://agilemanifesto.org/principles.html. [Accessed 17 April 2021].

[70] Anon. (n.d.). AGILE: meaning in the Cambridge English Dictionary. Cambridge Dictionary. [online] Available at: https://dictionary.cambridge.org/dictionary/english/agile [Accessed 6 Dec. 2020].

[71] Anon. (n.d.). Android: meaning in the Cambridge English Dictionary. Cambridge Dictionary. [online] Available at: https://dictionary.cambridge.org/dictionary/english/android. [Accessed 6 Dec. 2020].

[72] Anon. (n.d.). Meet Android Studio: Android Developers. Android Developers. [online] Available at: https://developer.android.com/studio/intro. [Accessed 6 Dec. 2020].

[73] Anon. (n.d.). API: meaning in the Cambridge English Dictionary. Cambridge Dictionary. [online] Available at: https://dictionary.cambridge.org/dictionary/english/api/ [Accessed 6 Dec. 2020].

[74] Montegriffo, N. (2021). What is an APK file and how to install APKs on Android? | NextPit. [online] NextPit. Available at: https://www.nextpit.com/android-for-beginners-what-is-an-apk-file. [Accessed 12 Apr. 2021].

[75] Rajora, H. (2020). What is a Branch in Git and the importance of Git Branches?. [online] TOOLSQA. Available at: https://www.toolsqa.com/git/branch-in-git/. [Accessed 5 Apr. 2021].

[76] Techopedia. (n/d). What is an Emulator? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/4788/emulator. [Accessed 05 Apr. 2021].

[77] Techopedia. (2021). What is Git? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/28960/git. [Accessed 05 Apr. 2021].

[78] Webopedia. (2021). GitHub Definition & Meaning | Webopedia. [online] Available at: https://www.webopedia.com/definitions/github-definition-meaning/. [Accessed 05 Apr. 2021].

[79] Schildt, H. (2019). Java: a beginner's guide, New York: McGraw-Hill Education.

[80] Techopedia. (n.d.). What is Java Development Kit (JDK)? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/5594/java-development-kit-jdk. [Accessed 05 Apr. 2021].

[81] Anon. (2019). What is Kanban? Agile Alliance. [online] Available at: https://www.agilealliance.org/glossary/kanban/. [Accessed 6 Dec. 2020].

[82] Techopedia. (n.d.). What is Kotlin? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/34021/kotlin. [Accessed 05 Apr. 2021].

[83] Android Developers. (n.d.). App Manifest Overview | Android Developers. [online] Available at: https://developer.android.com/guide/topics/manifest/manifest-intro. [Accessed 05 Apr. 2021].

[84] Techopedia. (2021). What is Merge? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/1217/merge. [Accessed 05 Apr. 2021].

[85] Becker, R. (n.d.). What is a Minimum Viable Product (MVP)? -Definition from Techopedia. Techopedia.com. [online] Available at: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp. [Accessed 6 Dec. 2020].

[86] Anon. (n.d.). What Is MongoDB? MongoDB. [online] Available at: https://www.mongodb.com/what-is-mongodb. [Accessed 6 Dec. 2020].

[87] Techopedia. (2021). What is MySQL? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/3498/mysql. [Accessed 04 Apr. 2021].

[88] GeeksforGeeks. (2021). What is a GIT Repository? - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/what-is-a-git-repository/. [Accessed 04 Apr. 2021].

[89] Anon. (2019). Scrum. Agile Alliance. [online] Available at: https://www.agilealliance.org/glossary/scrum/. [Accessed 6 Dec. 2020].

[90] Techopedia. (n.d.). What is a Software Development Kit (SDK)? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/3878/software-development-kit-sdk. [Accessed 05 Apr. 2021].

[91] tutorialspoint.com (2019). SDLC Overview. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/sdlc/sdlc_overview.htm. [Accessed 05 Apr. 2021].

[92] Anon. (n.d.). SQL (Structured Query Language) Definition. SQL (Structured Query Language) Definition. [online] Available at: https://techterms.com/definition/sql. [Accessed 04 Apr. 2021].

[93] Anon. (n.d.). XML (Extensible Markup Language) Definition. XML (Extensible Markup Language) Definition. [online] Available at: https://techterms.com/definition/xml. [Accessed 04 Apr. 2021].

Bibliography

A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza and S. Z. Sarwar, "Agile software development: Impact on productivity and quality" 2010 IEEE International Conference on Management of Innovation & Technology, Singapore, 2010, pp. 287-291, doi: 10.1109/ICMIT.2010.5492703. [Accessed 09 Jan. 2021].

Abhiandroid.com. 2021. XML in Android: Basics And Different XML Files Used In Android | Abhi Android. [online] Available at: https://abhiandroid.com/ui/xml. [Accessed 5 Apr. 2021].

Android Developers. (n.d.). Android Jetpack. [online] Available at: https://developer.android.com/jetpack. [Accessed 11 Apr. 2021].

Android Developers. (n.d.). Buttons. [online] Available at: https://developer.android.com/guide/topics/ui/controls/button. [Accessed 21 Feb. 2021].

Android Developers. (n.d.). CameraX overview. [online] Available at: https://developer.android.com/training/camerax. [Accessed 12 Mar. 2021].

Android Developers. (n.d.). Specify the input method type. [online] Available at: https://developer.android.com/training/keyboard-input/style. [Accessed 16 Feb. 2021].

Barron, B. (2019). Stripe vs Square: Which Payment Gateway Should You Use in 2021? [online] Kinsta. Available at: https://kinsta.com/blog/stripe-vs-square. [Accessed 5 Apr. 2021].

Billie Keita. 2020. Agile Scrum Methodology: A Beginner's Guide to Agile Scrum. [online] Available at: https://www.invensislearning.com/blog/agile-scrum-methodology/#What_Is_a_Scrum_Master. [Accessed 22 Mar. 2021].

BragitOff.com. 2021. How to store the data from EditText in a variable? [SOLVED] - Android Studio - BragitOff.com. [online] Available at: https://www.bragitoff.com/2017/03/how-to-store-the-data-from-edittext-in-a-variable-solved-android-studio// [Accessed 15 Mar. 2021].

Brook Appelbaum. N/A. Agile Methodologies: A Beginner’s Guide | Planview. [online] Available at: https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/. [Accessed 22 Mar. 2021].

CodeProject. 2021. Getting values from edittext into string variable and apply validations in xamarin android. - CodeProject. [online] Available at: https://www.codeproject.com/questions/1080919/getting-values-from-edittext-into-string-variable. [Accessed 15 Mar. 2021].

Coding Demos. 2017. How to make Android button clickable to start a new page - Coding Demos. [online] Available at: https://www.codingdemos.com/android-button-clickable/. [Accessed 05 Apr. 2021].

Connecting to MongoDB from Java. 2021. Connecting to MongoDB from Java. [online] Available at: https://www.quickprogrammingtips.com/java/connecting-to-mongodb-from-java.html. [Accessed 1 Feb. 2021].

D. S. Janzen and H. Saiedian, "On the Influence of Test-Driven Development on Software Design," 19th Conference on Software Engineering Education & Training (CSEET'06), Turtle Bay, HI, USA, 2006, pp. 141-148, doi: 10.1109/CSEET.2006.25. [Accessed 26 Feb. 2021]

Damodaran B, D., Salim, S. and Vargese, S.M. (2016). Performance Evaluation of MySQL and MongoDB Databases. International Journal on Cybernetics & Informatics, 5(2), pp.387–394. [Accessed 18 Jan. 2021]

Dev.mysql.com. 2021. MySQL :: MySQL 8.0 Reference Manual :: 1.2.2 The Main Features of MySQL. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/features.html. [Accessed 16 Mar. 2021].

Developers, A., 2011. What is android. Dosegljivo: http://www. academia. edu/download/30551848/andoid--tech. Pdf. [Accessed 10 Dec. 2020]

Dipina Damodaran, B., Salim, S. and Vargese, S.M., 2016. Performance evaluation of MySQL and MongoDB databases. Int. J. Cybern. Inform. (IJCI), 5. [Accessed 17 Jan. 2021]

Enck, W., Octeau, D., McDaniel, P.D. and Chaudhuri, S., 2011, August. A study of android application security. In USENIX security symposium (Vol. 2, No. 2). [Accessed 14 Jan. 2021]

Goel, R., 2021. Why is prototyping important?. [online] zipBoard. Available at: https://blog.zipboard.co/why-is-prototyping-important-13150d76abc4. [Accessed 7 Feb. 2021].

Google Codelabs. (n.d.). Getting Started with CameraX. [online] Available at: https://codelabs.developers.google.com/codelabs/camerax-getting-started?hl=ja#0. [Accessed 16 Feb. 2021].

guides.codepath.com. (n.d.). Working with the EditText | CodePath Android Cliffnotes. [online] Available at: https://guides.codepath.com/android/Working-with-the-EditText. [Accessed 18 Feb. 2021].

I. Dalmasso, S. K. Datta, C. Bonnet and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 2013, pp. 323-328, doi: 10.1109/IWCMC.2013.6583580. [Accessed 03 November 2020]

JDBC Tutorial - Tutorialspoint. N/A. JDBC Tutorial - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/jdbc/index.htm. [Accessed 05 Mar. 2021].

Jetbrains, A. (n.d.). Corporate Overview in 4 pages. [online]. Available at: https://resources.jetbrains.com/storage/products/jetbrains/docs/jetbrains_corporate_overview_compact.pdf. [Accessed 2 Apr. 2021].

Kamthan, P., 2009. Ethics in software engineering. In Software Applications: Concepts, Methodologies, Tools, and Applications (pp. 2795-2802). IGI Global. [Accessed 23 Jan. 2021]

Lake, I. (2016). Medium. [online] Medium. Available at: https://medium.com/androiddevelopers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd. [Accessed 4 Apr. 2021].

Lake, I. (2018). Layouts, Attributes, and you. [online] Medium. Available at: https://medium.com/androiddevelopers/layouts-attributes-and-you-9e5a4b4fe32c. [Accessed 16 Mar. 2021].

Lee, S., 2012. Creating and Using Databases for Android Applications. International Journal of Database Theory and Application, 5(2). [Accessed 04 December 2020]

Lerner, M., 2007. 'How to overcome the relational – XML Impedance Mismatch:XML processing within the JDBC Sender Adapter', BUSINESS PROCESS EXPERT COMMUNITY, pp. 1-6.

Loeliger, J. and McCullough, M., 2012. Version Control with Git: Powerful tools and techniques for collaborative software development. "O'Reilly Media, Inc.". [Accessed 12 December 2020]

Login Page in Android Studio Source Code. N/A. Login Page in Android Studio Source Code. [online] Available at: https://www.11zon.com/zon/android/login-page-in-android-studio-source-code.php. [Accessed 16 Mar. 2021].

Louise Gaille. 2021. 15 Advantages and Disadvantages of a Waterfall Model – Vittana.org. [online] Available at: https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model. [Accessed 22 Mar. 2021].

Minh, N., CodeJava. 2020. Java connecting to MongoDB database examples. [online] Available at: https://www.codejava.net/java-se/jdbc/java-connecting-to-mongodb-database-examples. [Accessed 1 Feb. 2021].

MongoDB. 2021. Using MongoDB with Java | MongoDB. [ONLINE] Available at: https://www.mongodb.com/java. [Accessed 1 Feb. 2021].

MySQL Tutorial. 2021. An Essential Guide to MySQL Foreign Key By Practical Examples. [online] Available at: https://www.mysqltutorial.org/mysql-foreign-key/. [Accessed 04 Mar. 2021].

MySQLTutorial. 2021. What Is MySQL. [online] Available at: https://www.mysqltutorial.org/what-is-mysql/. [Accessed 16 Mar. 2021].

Nunkesser, R., 2018, May. Beyond web/native/hybrid: a new taxonomy for mobile app development. In 2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft) (pp. 214-218). IEEE. [Accessed 13 Jan. 2021]

Oracle. N/A. Lesson: JDBC Basics. [online] Available at: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html. [Accessed 05 Mar. 2021].

Ostrander, J., 2012. Android UI Fundamentals: Develop and Design. Peachpit Press. [Accessed 02 Feb. 2021]

Praveen Max. 2011. Login application using Java Swings and Mysql. | PraveenMax. [online] Available at: https://praveenmax.wordpress.com/2011/01/21/simple-login-application-in-java-using-swings/. [Accessed 20 Mar. 2021].

Ramon. 2020. How to Make Login Page in Android Studio [online] Available at: https://howcreateit.com/programming/android-studio/how-to-make-login-page-in-android-studio/. [Accessed 16 Mar. 2021].

Rogers, R., Lombardo, J., Mednieks, Z. and Meike, B., 2009. Android application development. O'Reilly. [Accessed 15 November 2020]

Sarojadevi, H. (2011). Performance testing: methodologies and tools. Journal of Information Engineering and Applications, 1(5), pp.5-13. [Accessed 12 Feb. 2021]

SearchMobileComputing. (2019). What is Android Studio? - Definition from WhatIs.com. [online] Available at: https://searchmobilecomputing.techtarget.com/definition/Android-Studio. [Accessed 4 Jan. 2021]

Shah, R. and Chettri, K. (2020). Android XML Layouts – A study of Viewgroups and Views for UI Design in Android Application. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp.184–197. [Accessed 04 Feb. 2021]

Shukla, A. (2017). Structured Programming, its Advantages and Disadvantages - IncludeHelp. [online] www.includehelp.com. Available at: https://www.includehelp.com/data-structure-tutorial/structured-programming-its-advantages-and-disadvantages.aspx. [Accessed 15 Mar. 2021].

Smartdraw.com. 2021. Entity Relationship Diagram (ERD) - What is an ER Diagram? [online] Available at: https://www.smartdraw.com/entity-relationship-diagram/. [Accessed 16 Apr. 2021].

Stack Overflow. 2012. Best practices: Layouts on Android (Programmatic vs XML). [online] Available at: https://stackoverflow.com/questions/9827819/best-practices-layouts-on-android-programmatic-vs-xml. [Accessed 23 Mar. 2021].

Stack Overflow. 2019. Programmatic UI design. [online] Available at: https://stackoverflow.com/questions/53906433/programmatic-ui-design. [Accessed 23 Mar. 2021].

StatCounter Global Stats. (n.d.). Mobile & Tablet Android Version Market Share Worldwide. [online] Available at: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide. [Accessed 6 Apr. 2021].

Taka, D., 2016. Quick Tip - Gradle and How It Works with Android Studio. [online] Sitepoint.com. Available at: https://www.sitepoint.com/quick-tip-what-is-gradle-and-how-does-it-work-with-android-studio/. [Accessed 1 Feb. 2021].

TrustRadius. (n.d.). Android Studio vs NetBeans. [online] Available at: https://www.trustradius.com/compare-products/android-studio-vs-netbeans. [Accessed 3 Apr. 2021].

WADIC. 2021. 6 Advantages and Disadvantages of the Waterfall Model | Wadic. [online] Available at: https://wadic.net/waterfall-model-advantages-disadvantages/. [Accessed 22 Mar. 2021].

Wikipedia. 2021. Java Database Connectivity - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Java_Database_Connectivity. [Accessed 22 Mar. 2021].

www.holadevs.com. (n.d.). [Solved] java | Adding an icon to the start of a TextView in. [online] Available at: https://www.holadevs.com/pregunta/66311/add-icon-at-the-start-of-a-textview-on-android. [Accessed 2 Mar. 2021].

www.quora.com. (n.d.). What is better, Android Studio, Eclipse, or NetBeans for Android development? - Quora. [online] Available at: https://www.quora.com/What-is-better-Android-Studio-Eclipse-or-NetBeans-for-Android-development. [Accessed 3 Apr. 2021].

Yang, S., Yan, D. and Rountev, A., 2013, May. Testing for poor responsiveness in Android applications. In 2013 1st international workshop on the engineering of mobile-enabled systems (MOBS) (pp. 1-6). IEEE. [Accessed 22 Feb. 2021]

Appendices

Steps to install APK

1. Go to Google Play store and download the application named "Apk Installer"
2. On your phone go to Settings
    a. Select Apps and notifications
    b. Select app access
    c. Install unknown apps
3. Open and download the link https://mega.nz/file/ePZUnDbR#ocgukhCqYPotd7pbLTEDmylXpOfXTJNeyMz9PY3L8hA
4. Open "Apk Installer" application
    a. Select Splitsy ver0.2
    b. Install the APK
5. On your phone's apps list, locate Splitsy and open it.

Notion Workplace

https://www.notion.so/Software-Project-5cc4fb1f81854f15a8427d354a863766

GitHub Repository

https://github.com/ysmnpksy/Splitsy

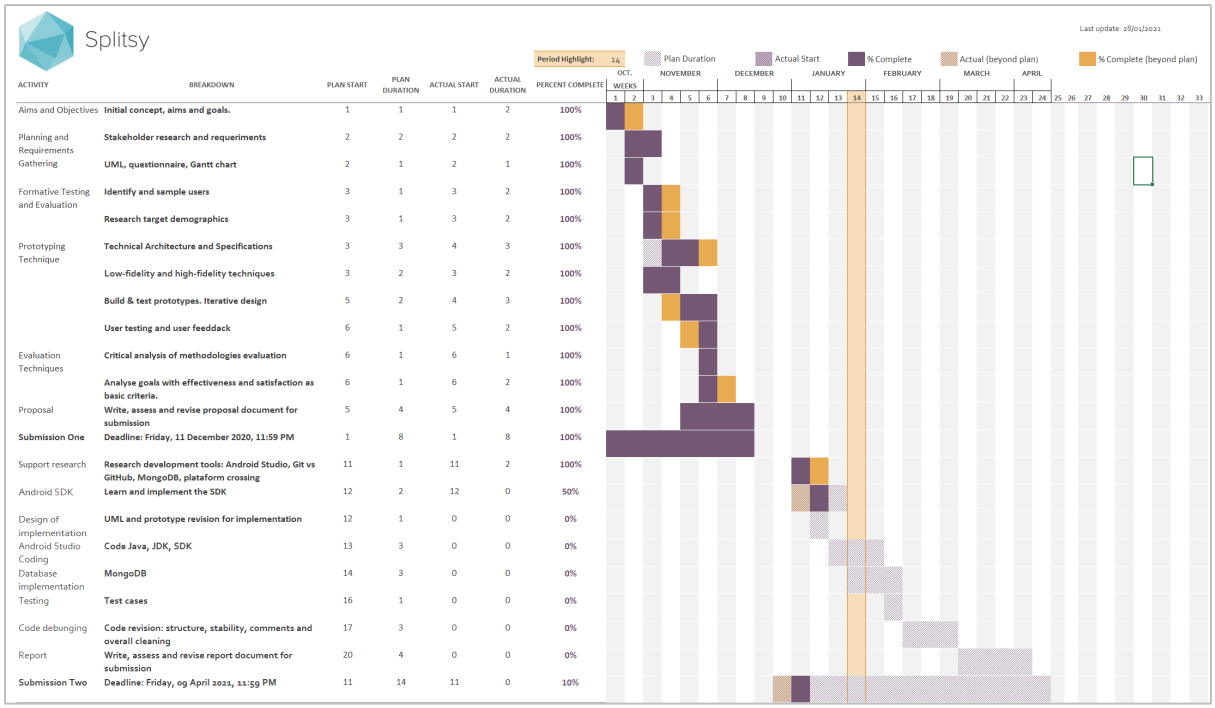


Figure 90: Initial Gantt Chart, dated 28/01/21.

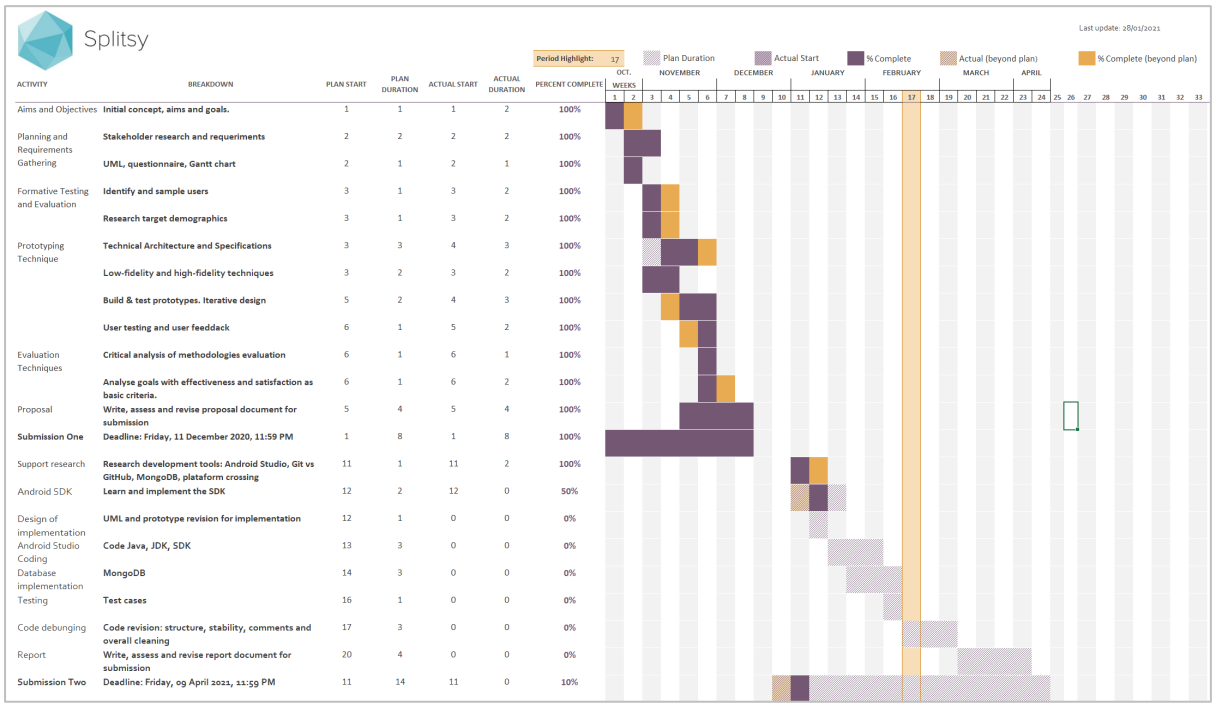


Figure 91: Updated Gantt chart, dated 22/02/21.

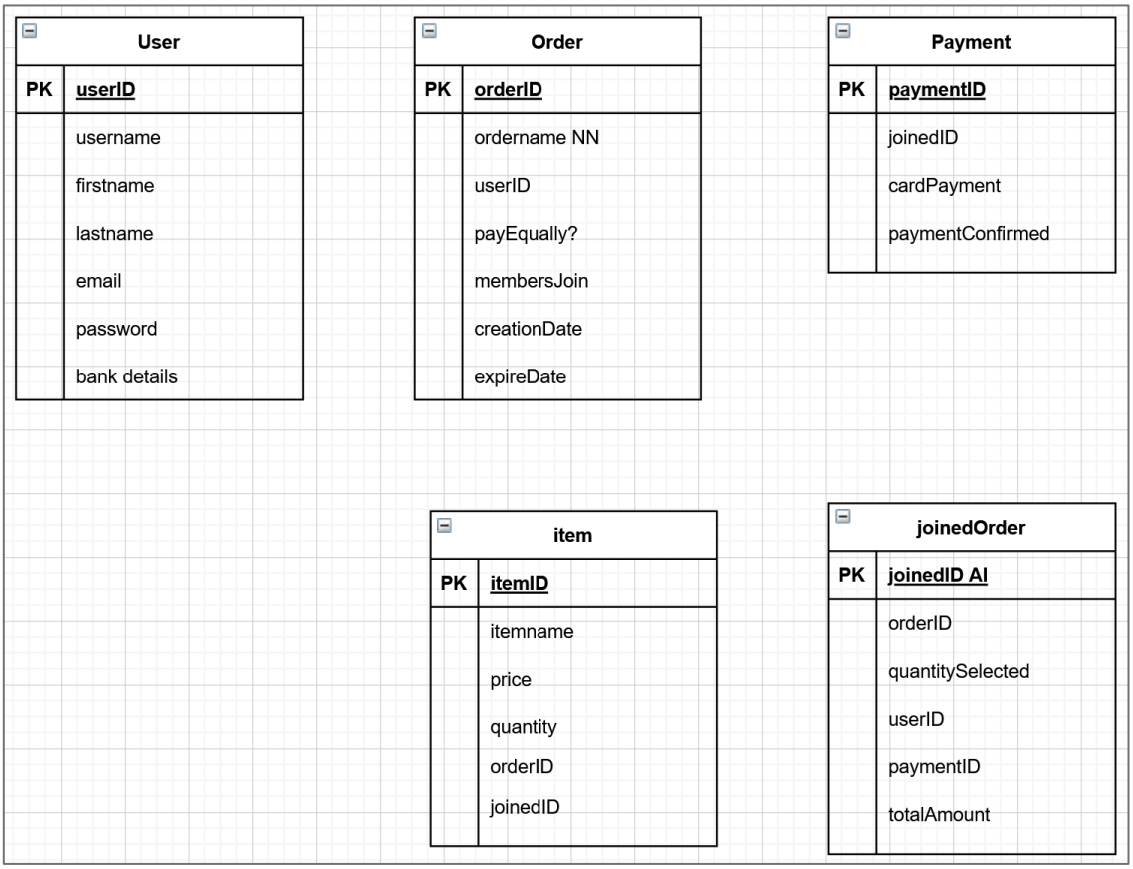


Figure 92: Initial ERD.

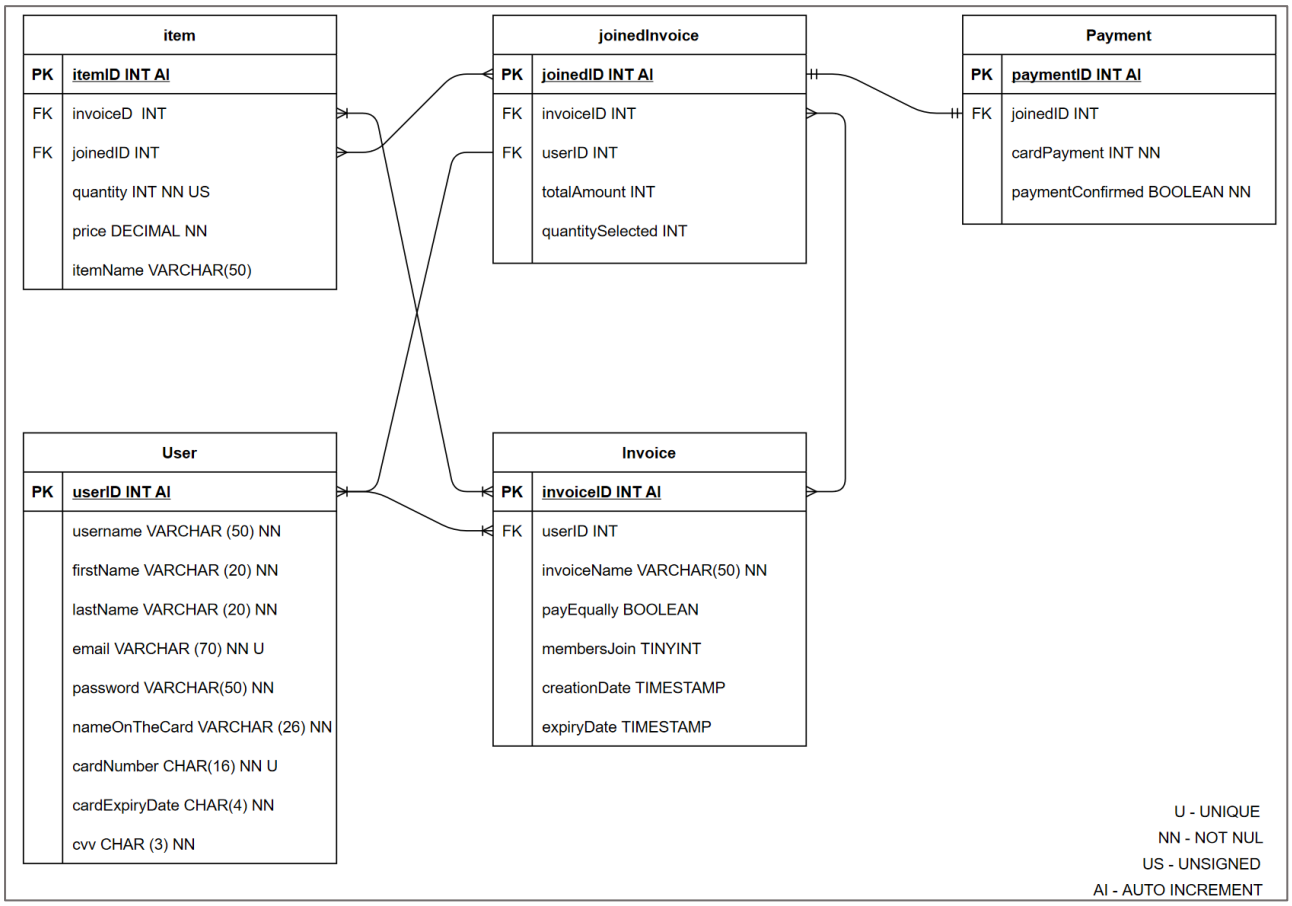


Figure 93: ERD version 2.

Sprints Default view

Name	Focus	Date	Tasks
Sprint 1	XML	Feb 15, 2021 → Feb 28, 2021	XML Merge, Research into GUIs, Class diagram: research on design elements to add
Sprint 2	SQL & Java	Mar 9, 2021 → Mar 18, 2021	Connect java pages & nav bar, Start Report, SQL and java connection, Set database, ERD for database, In-depth class diagram, code table of contents, Decide intermediate technologies for database
Sprint 3	Testing & Report	Mar 20, 2021 → Mar 25, 2021	Unit testing, Sign in function, Building test cases, UI Testing, Integration testing, Add entity relation diagram to Notion, Sign up paga java

New

Figure 94: Sprint planning table.

Members: brai001, dcard001, ddoch001, msous001, ypaks001

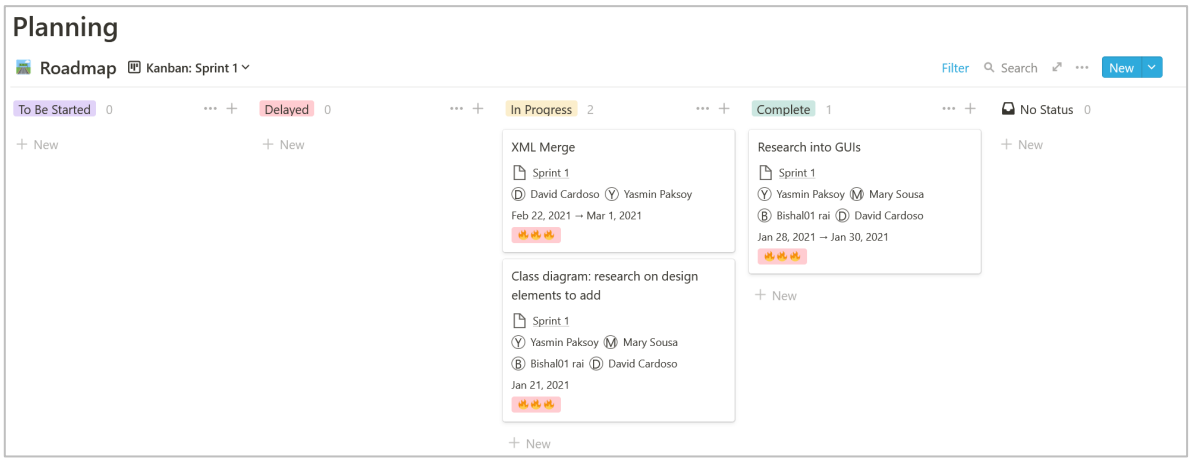


Figure 95: Sprint 1 Kanban board.

Members: brai001, dcard001, ddoch001, msous001, ypaks001

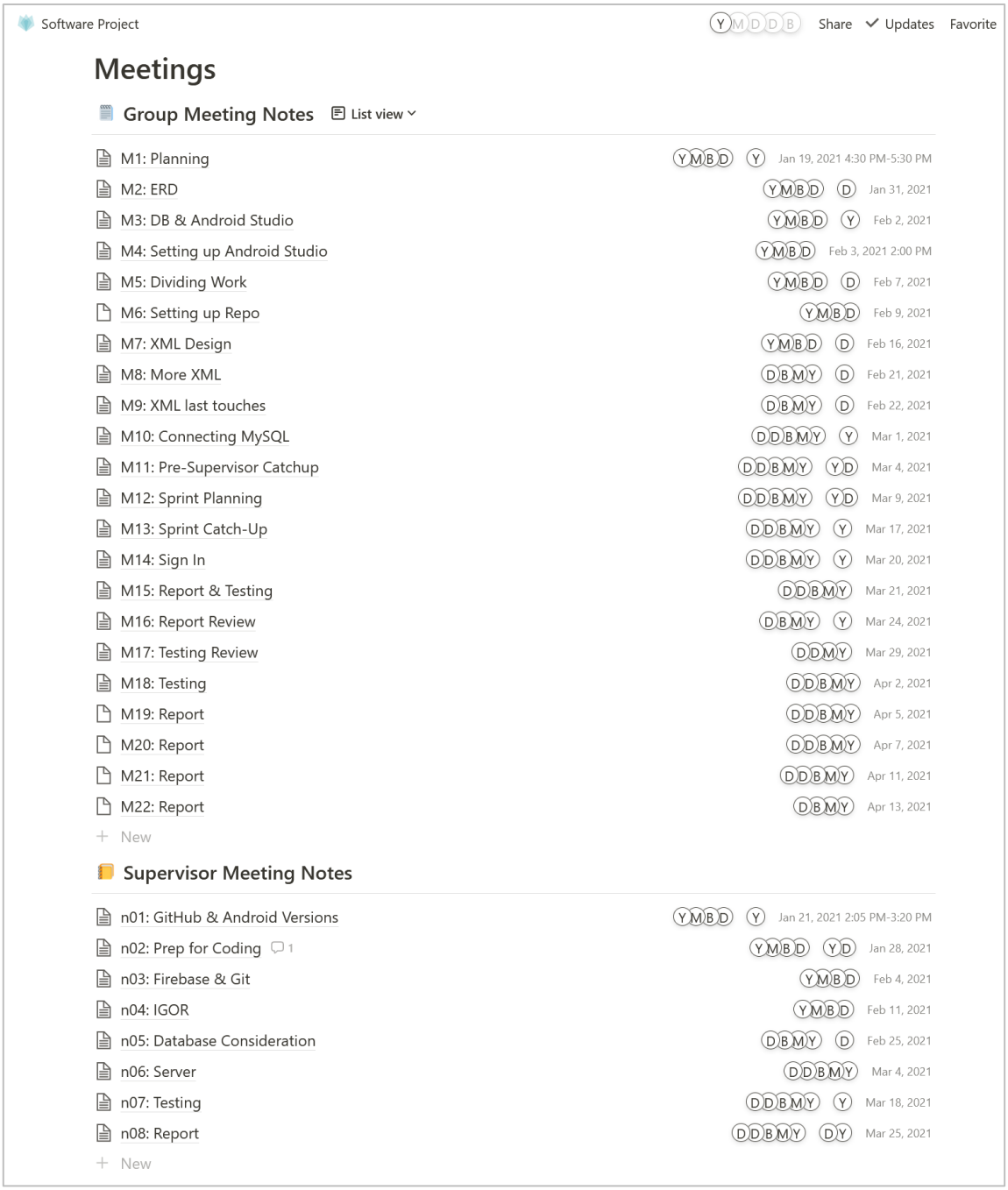
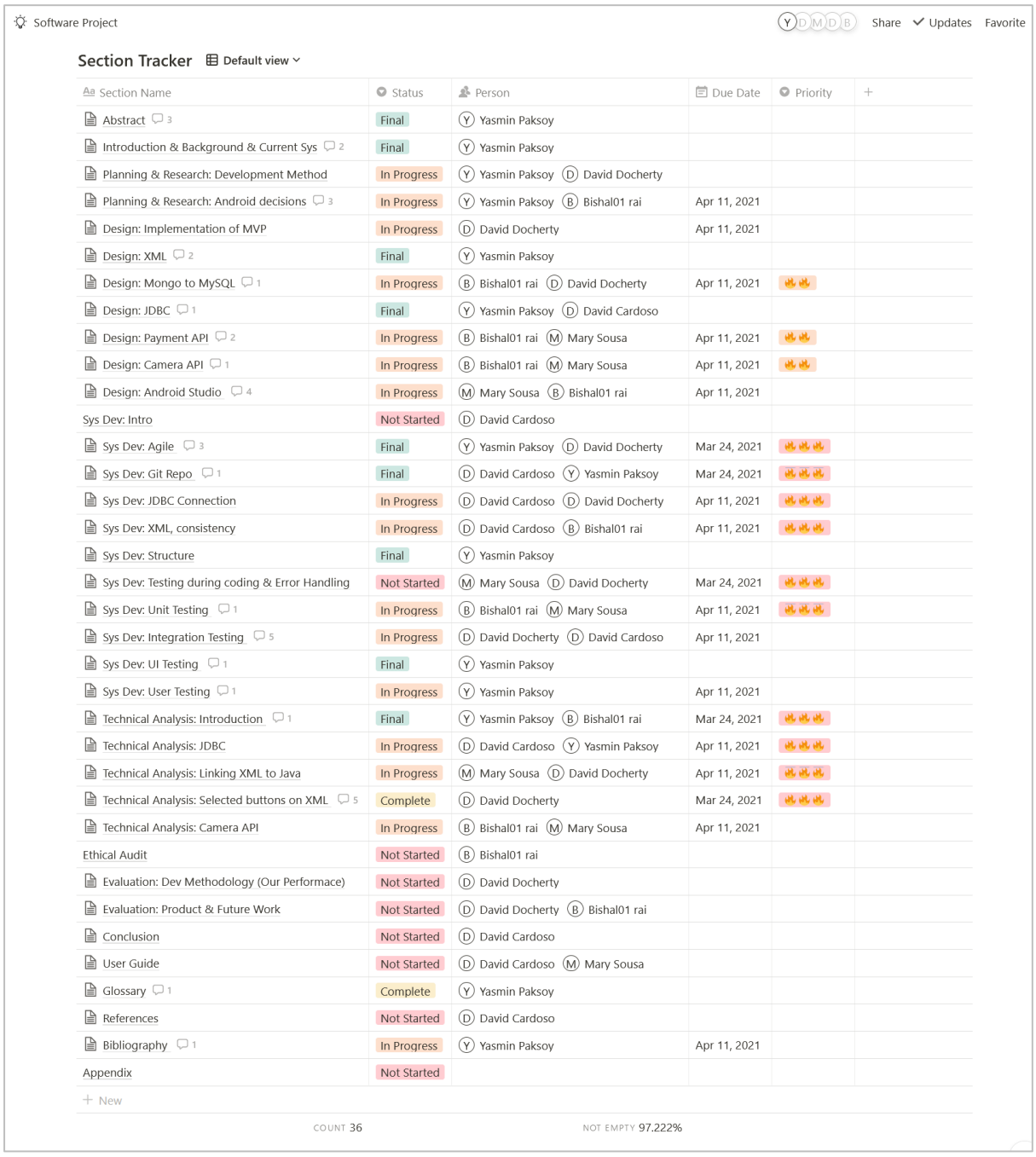


Figure 96: List of meeting notes.

Members: brai001, dcard001, ddoch001, msous001, ypaks001



Software Project

Share Updates Favorite

Section Tracker

Default view

Section Name	Status	Person	Due Date	Priority
Abstract 3	Final	Yasmin Paksoy		
Introduction & Background & Current Sys 2	Final	Yasmin Paksoy		
Planning & Research: Development Method	In Progress	Yasmin Paksoy, David Docherty		
Planning & Research: Android decisions 3	In Progress	Yasmin Paksoy, Bishal01 rai	Apr 11, 2021	
Design: Implementation of MVP	In Progress	David Docherty	Apr 11, 2021	
Design: XML 2	Final	Yasmin Paksoy		
Design: Mongo to MySQL 1	In Progress	Bishal01 rai, David Docherty	Apr 11, 2021	
Design: JDBC 1	Final	Yasmin Paksoy, David Cardoso		
Design: Payment API 2	In Progress	Bishal01 rai, Mary Sousa	Apr 11, 2021	
Design: Camera API 1	In Progress	Bishal01 rai, Mary Sousa	Apr 11, 2021	
Design: Android Studio 4	In Progress	Mary Sousa, Bishal01 rai	Apr 11, 2021	
Sys Dev: Intro	Not Started	David Cardoso		
Sys Dev: Agile 3	Final	Yasmin Paksoy, David Docherty	Mar 24, 2021	
Sys Dev: Git Repo 1	Final	David Cardoso, Yasmin Paksoy	Mar 24, 2021	
Sys Dev: JDBC Connection	In Progress	David Cardoso, David Docherty	Apr 11, 2021	
Sys Dev: XML, consistency	In Progress	David Cardoso, Bishal01 rai	Apr 11, 2021	
Sys Dev: Structure	Final	Yasmin Paksoy		
Sys Dev: Testing during coding & Error Handling	Not Started	Mary Sousa, David Docherty	Mar 24, 2021	
Sys Dev: Unit Testing 1	In Progress	Bishal01 rai, Mary Sousa	Apr 11, 2021	
Sys Dev: Integration Testing 5	In Progress	David Docherty, David Cardoso	Apr 11, 2021	
Sys Dev: UI Testing 1	Final	Yasmin Paksoy		
Sys Dev: User Testing 1	In Progress	Yasmin Paksoy	Apr 11, 2021	
Technical Analysis: Introduction 1	Final	Yasmin Paksoy, Bishal01 rai	Mar 24, 2021	
Technical Analysis: JDBC	In Progress	David Cardoso, Yasmin Paksoy	Apr 11, 2021	
Technical Analysis: Linking XML to Java	In Progress	Mary Sousa, David Docherty	Apr 11, 2021	
Technical Analysis: Selected buttons on XML 5	Complete	David Docherty	Mar 24, 2021	
Technical Analysis: Camera API	In Progress	Bishal01 rai, Mary Sousa	Apr 11, 2021	
Ethical Audit	Not Started	Bishal01 rai		
Evaluation: Dev Methodology (Our Performace)	Not Started	David Docherty		
Evaluation: Product & Future Work	Not Started	David Docherty, Bishal01 rai		
Conclusion	Not Started	David Cardoso		
User Guide	Not Started	David Cardoso, Mary Sousa		
Glossary 1	Complete	Yasmin Paksoy		
References	Not Started	David Cardoso		
Bibliography 1	In Progress	Yasmin Paksoy	Apr 11, 2021	
Appendix	Not Started			

New

COUNT 36 NOT EMPTY 97.222%

Figure 97: Report section tracker in progress on Notion.

Members: brai001, dcard001, ddoch001, msous001, ypaks001

Software Project

Section Tracker

Default view

Section Name	Status	Person	Due Date	Priority
Abstract 3	Final	Yasmin Paksoy		
Introduction & Background & Current Sys 2	Final	Yasmin Paksoy		
Planning & Research: Project Management 3	Final	Yasmin Paksoy, David Docherty		
Planning & Research: Android decisions 4	Final	Bishal01 rai	Apr 11, 2021	
Design: Implementation of MVP 6	Final	David Docherty	Apr 11, 2021	
Design: XML 2	Final	Yasmin Paksoy		
Design: Mongo to MySQL 5	Final	David Cardoso, Yasmin Paksoy	Apr 11, 2021	🔥🔥
Design: JDBC 1	Final	Yasmin Paksoy, David Cardoso		
Design: Payment API 2	Final	Bishal01 rai, Mary Sousa	Apr 11, 2021	🔥🔥
Design: Camera API 3	Final	Bishal01 rai, Mary Sousa	Apr 11, 2021	🔥🔥
Design: Android Studio 2	Final	Mary Sousa, Bishal01 rai	Apr 11, 2021	
Sys Dev: Intro 3	Final	David Cardoso		
Sys Dev: Agile 3	Final	Yasmin Paksoy, David Docherty	Mar 24, 2021	🔥🔥🔥
Sys Dev: Git Repo 1	Final	David Cardoso, Yasmin Paksoy	Mar 24, 2021	🔥🔥🔥
Sys Dev: XML, consistency 9	Final	David Cardoso, Bishal01 rai	Apr 11, 2021	🔥🔥🔥
Sys Dev: Structure	Final	Yasmin Paksoy		
Sys Dev: Testing during coding & Error Handling 2	Final	David Docherty, Yasmin Paksoy	Mar 24, 2021	🔥🔥🔥
Sys Dev: Unit Testing 1	Final	Bishal01 rai, Mary Sousa	Apr 11, 2021	🔥🔥🔥
Sys Dev: Integration Testing 11	Final	David Docherty	Apr 11, 2021	
Sys Dev: UI Testing 1	Final	Yasmin Paksoy		
Sys Dev: User Testing 13	Final	Yasmin Paksoy	Apr 11, 2021	
Technical Analysis: Introduction 1	Final	Yasmin Paksoy, Bishal01 rai	Mar 24, 2021	🔥🔥🔥
Technical Analysis: JDBC 3	Final	David Cardoso, Yasmin Paksoy	Apr 11, 2021	🔥🔥🔥
Technical Analysis: Linking XML to Java 3	Final	Mary Sousa, David Docherty	Apr 11, 2021	🔥🔥🔥
Technical Analysis: Selected buttons on XML 6	Final	David Docherty	Mar 24, 2021	🔥🔥🔥
Technical Analysis: Camera API 2	Final	Bishal01 rai, Mary Sousa	Apr 11, 2021	
Ethical Audit 4	Final	Bishal01 rai		
Evaluation: Dev Methodology (Our Performance) 11	Final	David Docherty, David Cardoso		
Evaluation: Product & Future Work 7	Final	David Docherty, Bishal01 rai		
Conclusion 5	Final	David Cardoso		
User Guide 7	Final	David Cardoso, Mary Sousa		
Glossary 1	Final	Yasmin Paksoy		
References	Final	Mary Sousa		
Bibliography 1	Final	Yasmin Paksoy	Apr 11, 2021	
Appendix	Final			

+ New

COUNT 35 NOT EMPTY 97.143%

Figure 98: Final section tracker on Notion.