Boom Budget Expense Tracking Application

CTEC3451: Development Project

Author: Lawrence Molokwu, P2514538

Supervisor: Salimah Mohamed,

De Montfort University, Leicester, UK

Words: 8197

Table of Contents

Acknowledgements....3

Chapter1 - Introduction....4

1.1- Project Background....4

1.2- Motivation and academic objectives....5

1.3-Project Overview....6

1.4 - Project Management:....8

Chapter2- System Components & Design....9

2.1- User interface and user experience....9

Chapter 3- Development Lifecycle....15

3.1- Getting the requirements:....15

3.2- Design:....15

3.3 - Implementation:....15

3.4 -Testing:....15

3.5 -Deployment:....15

3.6 -Continuous Improvement and Post-Deployment:....16

Chapter 4- Implementation....16

4.1 -Backend and security....16

4.2- Verification and Validation....18

4.3 -System History....22

4.4- System Analytics 24
Chapter 5 - Testing 27
5.1 -Junit test........ 27
5.2 -Instrumentation testing 27
Chapter 6- critical Analysis:........ 30
6.1 - Introduction 30
6.2 - Software and tools used. 30
6.3 - Research and Development Environment 30
6.4 - Libraries used. 31
6.5 - Major Dependencies Explained: 32
6.6 - Layout Design in Android Studio........ 32
6.7 – Risk assessment........ 33
6.8 - Challenges and issues 33
6.9 - Achievements........ 34
6.10 – Future Recommendations:........ 34
Chapter 7 -Conclusion........ 35
Appendices........ 35
Appendix – A 35
Appendix – B 36
Appendix – C 37
Appendix – D 38
Appendix – E 39
Appendix – D 39
Appendix – E 40
Appendix – F........ 40
References........ 41

Acknowledgements

My deepest thanks go out to the people, organisations, and resources that were so helpful in the creation of this budget tracking programme. My sincere gratitude goes out to the Stack Overflow community primarily. The various developers that have contributed to Stack Overflow have been quite helpful in offering answers and insights, especially when facing difficult programming problems involving Android Studio and Java (Stack Overflow, n.d.). On this site, there is a truly admirable culture of information sharing. In addition, I want to thank Educatree for their excellent YouTube instructional on "How to connect Android with Firebase database". For the application's database functionality, the tutorial's step-by-step instructions and clear explanation of how to integrate Firebase were essential (Educatree, 2018).

I owe Nilesh Technology a debt of gratitude, especially for their YouTube video titled "Android Healthcare Project | Android beginner Project | Tutorial." This tutorial acted as a useful resource and source of inspiration, especially when it came to comprehending the best practises for project structure and design in Android applications (Nilesh Technology, 2022).

Finally, I would want to thank the larger Java and Android Studio communities such as reddit and java forum. Throughout the development process, the documentation, forums, and user contributions have been crucial tools. My ability to create this application has been influenced by these communities' dedication to open collaboration and knowledge-sharing.

In conclusion, the foundation of our endeavour has been the knowledge and generosity of these communities and their resources. All parties involved have my sincere gratitude.

Chapter1 - Introduction

1.1- Project Background

Mobile phones have evolved into an essential aspect of daily life in the modern world, which is characterised by a rapid technological change (West & Mace, 2010). Over the past ten years, mobile applications have become a focus in technology. The widespread use of smartphones and tablets is blamed for this trend (Statista, 2021). The popularity of mobile applications is fuelled by a number of factors. One essential element is the broad accessibility of high-speed internet, which has improved the usability and accessibility of mobile applications, particularly social media applications (Chen, 2018). Mobile applications' extensive range of uses, which include communication, entertainment, health, gaming, productivity, commerce, and money, is another consideration (Kumar & Gupta, 2016).

Applications for tracking budgets have become more popular in the world of finance, especially because they make it easier for users to keep track of and control their spending and revenue. This makes it easier to stick to a budget and encourages the achievement of financial goals. Fenton (2014) found that 19% of Britons do not keep track of their monthly spending. This emphasises the difficulty in efficiently managing spending that many people encounter. A person's cash flow can be negatively impacted by a lack of spending tracking, which is especially important for prospective entrepreneurs. According to Suurd (2020), cash flow budgets are essential for keeping track of financial transactions for both personal and commercial finances, and they require ongoing supervision for a business to be effective and profitable.

By enabling users to register accounts, log in, and keep track of their budgets, the application under review has been created to address the issues. Users of the application can enter their expenses on a weekly or monthly basis with the added feature of preserving these facts. Additionally, users have access to notes and analytics that give them information on how their spending compares to their spending plan.

It is crucial to remember that this application serves everyone and is not just for commercial organisations. A versatile tool for managing personal finances and promoting financial literacy, this inclusive approach broadens its application (Perry & Reilly, 2018).

1.2- Motivation and academic objectives

The introduction of smartphones has completely changed how people manage their daily lives, including their finances. Eighty-one percent of Americans own a smartphone, and a sizeable percentage of them utilise mobile applications for a variety of things, including banking, according to Pew Research Centre (2019). Due to the growing use of smartphones, there is a huge market opportunity for creating useful applications. Financial management's essential component of budget tracking is sometimes seen as time-consuming and unpleasant. Gathergood (2012) stated that bad monetary management might result in unfavourable results including debt and a lower quality of life. The creation of a budget tracking software tries to solve this issue by giving users a practical and effective tool for handling their personal finances.

Android Studio was selected as the Integrated Development Environment (IDE) for creating this application because of its dependability and widespread recognition among developers for creating Android applications (Soyinka, 2018). Because of its dependability and numerous libraries that make application development easier, Java, a flexible and popular programming language, was chosen to create the application (Oracle, 2020). The database for this application was chosen, and it is a Firebase offering from Google. For its scalability, real-time updates, and simple interaction with Android applications, Firebase is acclaimed (Karim & Ding, 2020). YouTube tutorials played an enormous influence in the learning process as well. The "Android Healthcare Project beginner Tutorial" and "How to connect Android with Firebase database" educational resources from Educatree and Nilesh Technology, respectively, were crucial in giving practical knowledge and hands-on experience in creating applications using Android Studio, Java, and integrating with Firebase.

From the programming standpoint the main objective was to gain understanding of using Android studio and java. Integrating Google Firebase as the backend database for effectively storing and retrieving user data is one of the main development goals. According to Kamir & Ding (2020), Firebase is renowned for its real-time database, scalability, and simplicity of interaction with Android applications. Implementing Real-time Data synchronization was key programming objective, Users must have access to real-time data synchronisation to have current knowledge of their financial situation. Real-time synchronisation in mobile applications enhances the user experience by delivering timely information, claim Ciman, Gaggi, and Avesani (2017). In the fast-paced financial environment, this tool can assist consumers in making quick, informed financial decisions. Creating an intuitive and user-friendly interface is one of the main goals of programming. The significance of user-friendly design in mobile health applications is highlighted by a study by Lazard and Mackert (2015). By analogy, this pertains to applications for tracking your spending, as consumers are more inclined to interact with an app that has a user-friendly UI. Increased budgeting practises may be adopted as a result, which could enhance both the stability of the economy as a whole and the financial health of everyone.

Finally, evaluating how the budget tracking software affects users' financial literacy is one of the applications main goals. The necessity of financial literacy for making wise financial decisions is essential in the 21st century(Huston ,2010). This application seeks to determine whether it aids in increasing financial literacy, which may in turn result in improved monetary management at the individual and societal levels, through user input and analysis. The application also aims to determine whether users' spending habits have changed behaviourally. Tools that offer insights on expenditure, in accordance with Thaler and Sunstein (2008), can encourage users to adopt better financial practises. The societal impact of the application in terms of encouraging a saving culture and fiscal responsibility might be better understood by evaluating this element.

1.3-Project Overview

Boom is a budget expense tracker application that allows a user to track expenses and compare their expense to the budget set by the user . The user can input their expenses and see their expenditure history, and this can also be compared to the budget set by the user . The user can add expenses daily and the total amount weekly, today, and monthly is updated as the user progresses.

Development Environment:

System Software: Android Studio

Programming Language: Java

Database: Firebase Real Time Database

Scope

Only small retailers and those using it at home may use this programme. This application falls short of all the standards set by more advanced companies. This programme has a good market potential because it can be used by anyone who wants to budget their money, save for future investments, and many other things. The budget management process will be helped by this tool. This software addresses the issue of pen and paper budget management and can assist customers in managing their future savings.

Major Features:

- Users can Login
- Users can register.
- Users' data is updated.
- Users can add expenses daily.
- Users can track Weekly and monthly expenses.
- Users can set a Monthly Budget
- Users can compare expenses with Budget and how much they have saved.
- Users can also view their history.
- Users can View Monthly and daily analytics which will be represented using a pie chart.
- Users can delete account and change password.

Operating Environment:

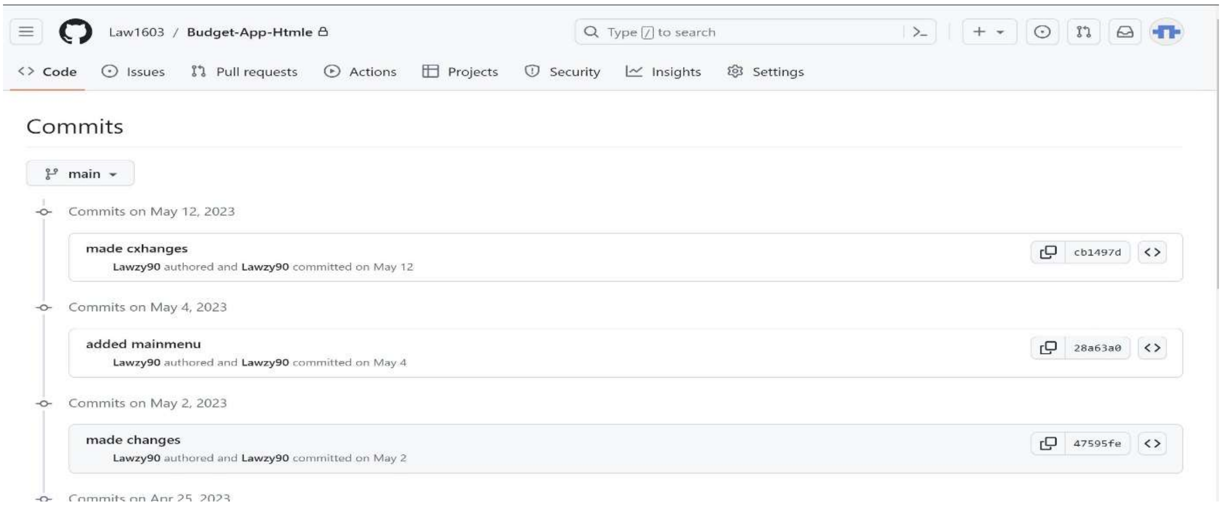
	Development Environment	**Operational Environment**
HARDWARE REQUIREMENTS	Processor: intel core i5 RAM: 8 GB or more Hard disk: 80 GB or more	RAM: 1 GB or more ROM: 4 GB or more Processor: 1 Ghz Dual Core or Higher

SOFTWARE REQUIREMENTS	Android Studio 4.1.2 or higher	Network connectivity
	Android Application Package (.apk)	Android 4.0 or higher

1.4 - Project Management:

The "Boom" budget expense monitoring application's development lifecycle, which adheres to the Agile methodology, is characterised by an iterative and incremental process that prioritises adaptability, cooperation, and ongoing input. This approach guarantees that the programme adjusts to changes and effectively meets user expectations.

- In the initial stage the project's groundwork is laid during the project planning phase. A summary of the project's aims and objectives is given.
- Stakeholders are named, and their tasks and obligations are laid down.
- A development roadmap is made after determining the project's scope.
- The tools and technologies are chosen, such as Firebase for the backend database and Android Studio for app development. These decisions are made based on factors including performance, security, and scalability.



Commits

main

Commits on May 12, 2023

made cxhanges
Lawzy90 authored and Lawzy90 committed on May 12 cb1497d

Commits on May 4, 2023

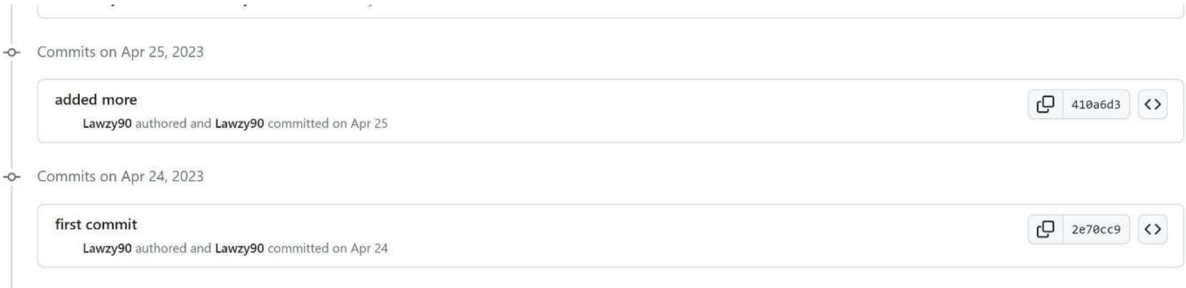
added mainmenu
Lawzy90 authored and Lawzy90 committed on May 4 28a63a0

Commits on May 2, 2023

made changes
Lawzy90 authored and Lawzy90 committed on May 2 47595fe

Commits on Apr 25, 2023

Figure 1- GitHub commits example.



Commits on Apr 25, 2023

added more
Lawzy90 authored and Lawzy90 committed on Apr 25 410a6d3

Commits on Apr 24, 2023

first commit
Lawzy90 authored and Lawzy90 committed on Apr 24 2e70cc9

Figure 2- GitHub commits example.

Chapter2- System Components & Design

2.1- User interface and user experience

The goal of the budget monitoring software is to assist users in managing their money well by keeping track of their budgets, expenses, and revenue. To provide the functionality of the app and guarantee a seamless user experience, the system components are essential. A user-friendly User Experience (UX) ensures the application's effectiveness, and the User Interface (UI) is crucial for capturing users' attention (Lallemand, Gronier, & Koenig, 2015). The application uses XML files for UI component layout. The use of the Recycler View for list-based expense presentation ensures an effective, scalable list that improves performance (Google, 2021). The Floating Action Button is placed in a way that facilitates quick access to the form for entering expenses. A typical design strategy that reduces the possibility of unintentional input is the usage of Alert Dialog to capture user input (Nielsen, 2014). The Progress Dialog updates the user on the status of the processing, keeping them informed (Hoober, 2014).

The "activity_login.xml" XML layout file serves as the representation of the user interface for the login screen in an Android application. It includes a number of components intended to improve user experience and make login easier. A Linear Layout serves as the layout's main component and arranges its child views vertically. The login screen is made visually appealing and user-friendly for users thanks to this design decision, which guarantees a linear flow of elements. Users can start the login procedure by clicking the login button. The button's language makes its purpose crystal apparent, giving consumers a simple next step to take. To give users who do not already have an account a way to access the registration screen, a Text View element with the words "Don't have an Account, Signup Here" has been added. This text view serves as a connection to the registration process, enhancing user comfort and usability. During the login process, a Progress Dialog is used to give the user visual cues. By letting users know that the login process is active and prohibiting them from trying to log in more than once at the same time, this design choice enhances user experience.

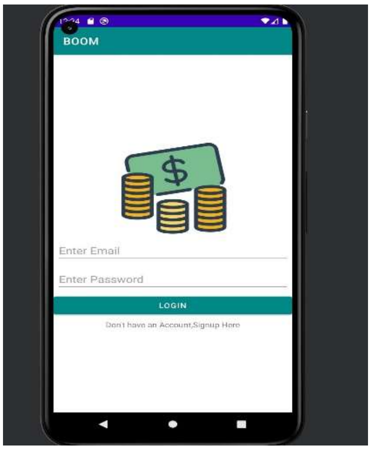
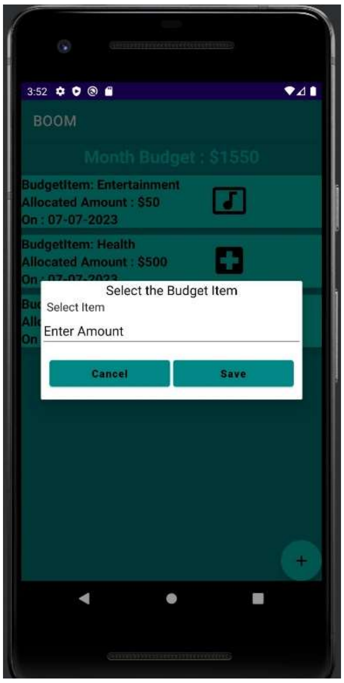


Figure 3- App Login screen



Month Budget : $1550

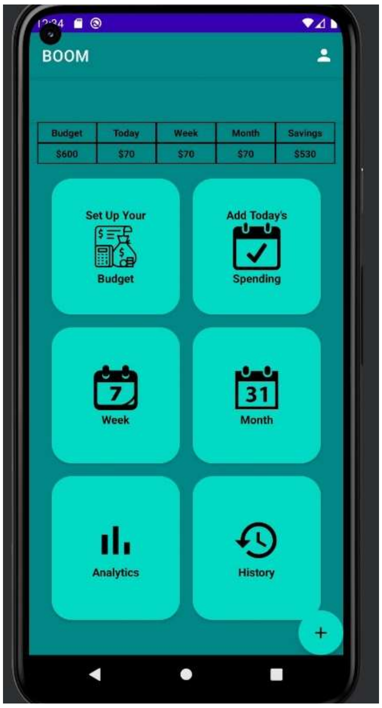
BudgetItem: Entertainment
Allocated Amount : $50
On : 07-07-2023

BudgetItem: Health
Allocated Amount : $500
On : 07-07-2023

Figure 4- Budget UI

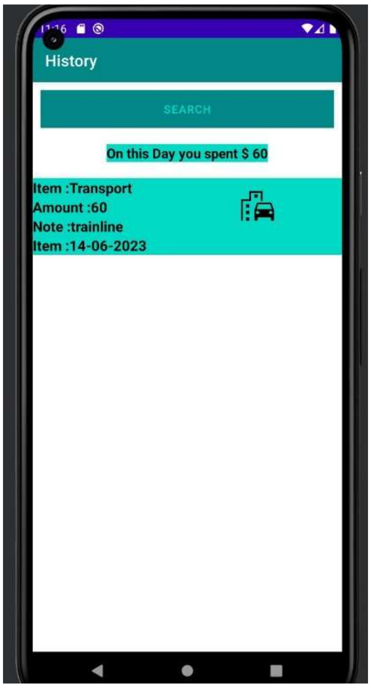
The user-friendly registration tool attempts to offer a seamless experience. Users can fill out the appropriate input boxes with their email address and password. If any of the fields are left blank, error notices will be shown. A progress dialogue that shows that the registration process is ongoing is displayed when the registration button is pressed. Users are sent to the app's main feature after completing the registration process successfully. A toast message is displayed to alert the user if there is a problem with registration.

The main screen's user interface (UI) is made to be aesthetically pleasing, well-organized, and simple to use. Financial data, such as budgets, spending totals, and analytics, are presented using a combination of CardView and TableLayout components in the XML layout file activity_main.xml. The toolbar, icons, and text are all correctly formatted and aligned to improve reading and visual hierarchy. The use of colour, borders, and padding enhances the ability to distinguish between various sections and creates a unified appearance.



Budget	Today	Week	Month	Savings
$600	$70	$70	$70	$530

Figure 5- Main menu user interface

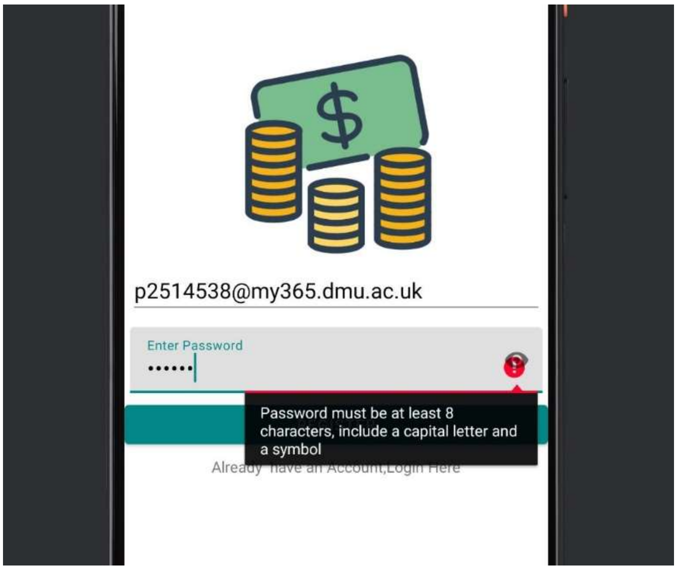


On this Day you spent $ 60

Item :Transport
Amount :60
Note :trainline
Item :14-06-2023

Figure 6- History Ui

- a daily, weekly, and monthly basis.

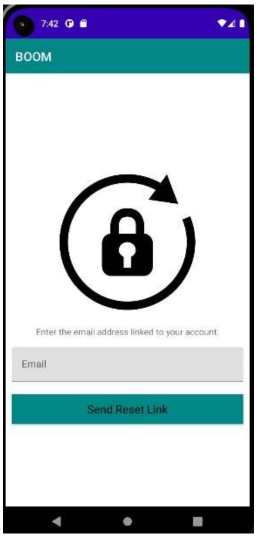


p2514538@my365.dmu.ac.uk

Enter Password: ••••••

Password must be at least 8 characters, include a capital letter and a symbol

Figure 7– Example of error handling in scenarios where the user’s password is not up to 8 characters.



Enter the email address linked to your account.

Email

Figure 8-– Forgot Password

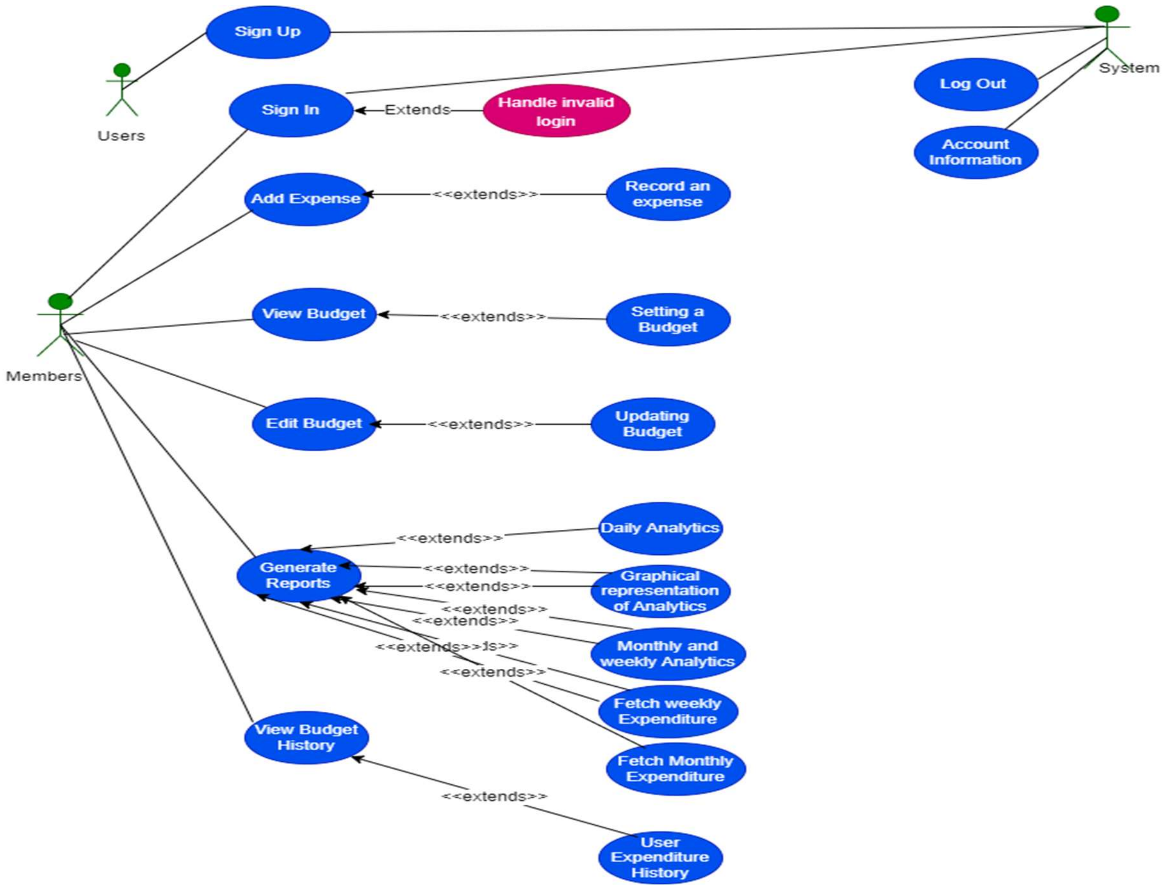


Figure 9- Use case diagram

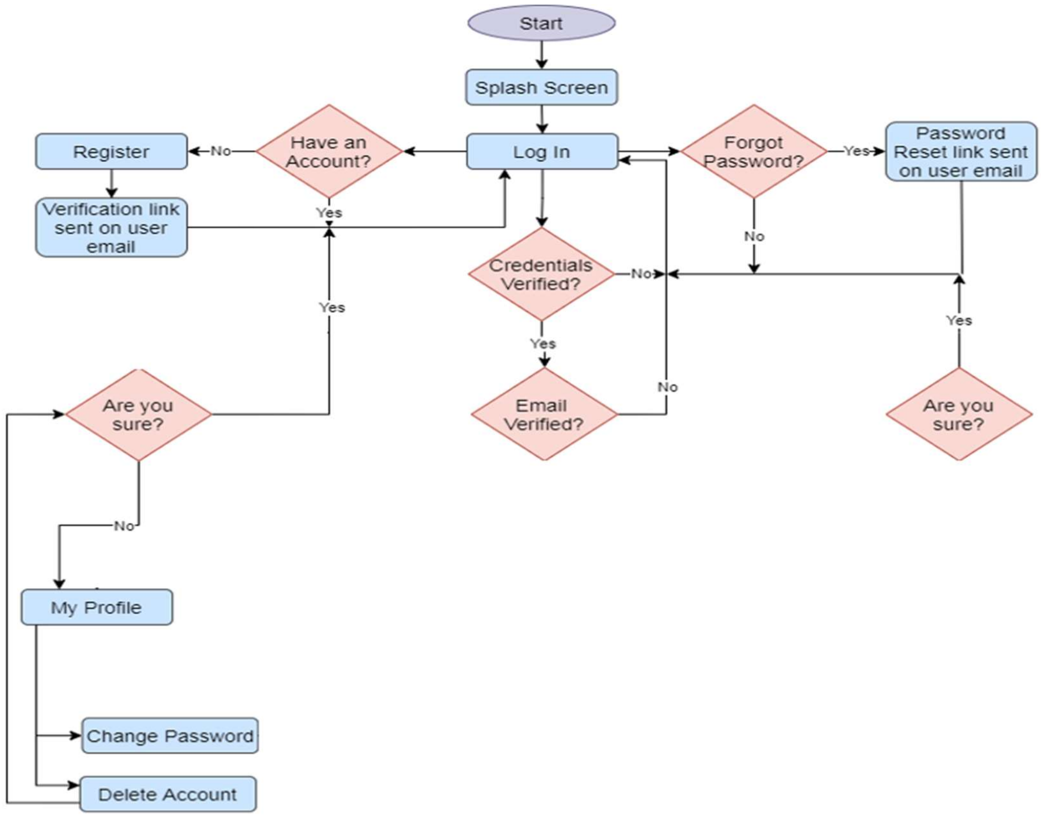


Figure 10- Application Flowchart

Chapter 3- Development Lifecycle

3.1- Getting the requirements:

- To better understand their requirements and expectations, this step involves engaging with potential users and stakeholders through interviews, surveys, and market research.
- The "Boom" app needs to have certain fundamental features and capabilities, like the ability to analyse costs on a daily, weekly, and monthly basis and display historical data in pie charts.
- A product backlog is frequently used to prioritise and document these requirements (See appendix A).

3.2- Design:

- The "Boom" application's architectural foundation is developed at this point.
- Data entities and their relationships are considered when designing the database structure. Because of its scalability and real-time synchronisation capabilities, Firebase Real-Time Database was chosen.
- The UI/UX design is given consideration, making sure it is simple and user-friendly. This includes creating pie charts and other data visualisation elements for historical analytics.

3.3 - Implementation:

- In the implementation phase, code is created using Java in Android Studio in compliance with the design specifications.
- The modular nature of the code ensures maintainability and reusability while conforming to best practises and standards.
- User authentication, data storage, and retrieval APIs are used to integrate with Firebase.

3.4 -Testing:

- The "Boom" application must pass this crucial stage to ensure that it adheres to the requirements and is free of serious problems.
- While integration testing focuses on interactions between modules, unit testing is done on individual components. See (Appendix B-D)

3.5 -Deployment:

- The application is packaged as an APK file appropriate for Android devices once it has passed all testing and quality checks.After that, it may be released on the Google Play Store or made available through other means.
- To improve security and user experience, features like account deletion, email verification, and forgotten password functionality are introduced during deployment.

3.6 -Continuous Improvement and Post-Deployment:

- The "Boom" application is regularly checked for problems after launch, and user feedback is gathered for further revisions.
- Users of the application can graphically analyse their spending patterns thanks to features like history statistics presented in pie charts.
- Effective budget management requires users to be able to track their expenditure on a daily, weekly, and monthly basis.

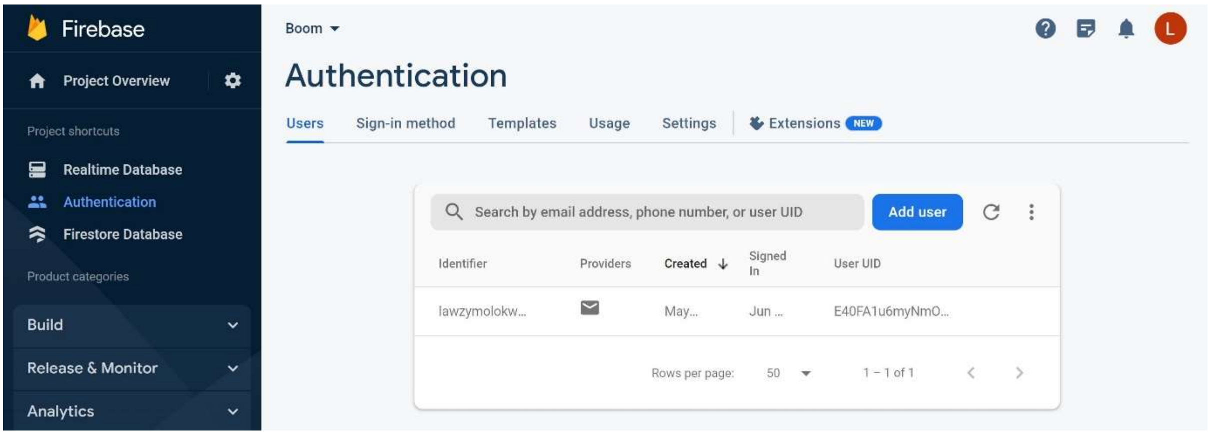
Chapter 4- Implementation

4.1 -Backend and security

As the backend service, Firebase is incorporated and offers various advantages like scalability, real-time synchronisation, and simplicity of integration (Firebase, 2021). For authentication, FirebaseAuth is employed. Using FirebaseAuth adds an extra degree of protection by guaranteeing that only authenticated users can access their c data. For the purpose of storing and retrieving data, Firebase Database is used. An application may maintain real-time data with minimal latency by utilising Firebase Realtime Database (Firebase, 2021). The security of the application depends heavily on Firebase Authentication. It makes sure that only users who have been verified are allowed access to data. To guard against potential security flaws such input manipulation, data validation is also used in the form inputs (OWASP, 2021).

The Java class file "LoginActivity.java," which provides the necessary backend logic and functionality for user authentication using Firebase, is a complement to the XML layout file. Security, database integration, and a flawless user experience are given top consideration while making implementation decisions in this class. User authentication is managed by the Firebase Authentication library. This option provides a safe and dependable means of managing user login information. Utilising the user's email and password that are safely saved in the Firebase backend, the signInWithEmailAndPassword() method is used to authenticate the user. Utilising TextUtils, the implementation incorporates email and password field validity checks. To confirm that the user enters both credentials, use the isEmpty() function. SetError() is used to present the user with the proper error message if any field is empty. Through this method, login attempts that are invalid or insufficient are avoided and security is improved.

Choosing Firebase Authentication allows for a more straightforward backend development and offers a secure and dependable user authentication solution. The completion listener enables proper handling of both successful and unsuccessful registration attempts, while the createUserWithEmailAndPassword() method provides a simple approach to create user accounts. The password and email that are entered are verified and securely sent to the Firebase server for account creation to ensure secure registration. The encryption and storage of user credentials are managed by the Firebase Authentication library. The text Password input type is also used in the password input field to hide the characters that users enter.



Identifier	Providers	Created	Signed In	User UID
lawzymolokw...	✉	May...	Jun ...	E40FA1u6myNmO...

Figure 11- Example of user credentials storage

Managing user identification, obtaining, and updating data from the Firebase Realtime Database, and carrying out computations based on user inputs are all part of the app's backend functionality. The MainActivity.java file configures event listeners to retrieve data from the database and initialises the Firebase references. It makes use of queries to retrieve spending totals for time frames, including today, this week, and this month. The implementation effectively manages data updates and retrieval, ensuring that the app is always accurate and responsive. The MainActivity.java file uses Firebase Authentication (mAuth) to identify users and retrieve their specific user ID (onlineUserID) to protect the security of user data. The secure database references (budgetRef, expensesRef, and personalRef) that are made using this user ID ensure that each user can only read and alter their own data. The application makes sure that user data is kept secure by utilising Firebase Authentication and appropriate database security policies.

To create a user-friendly and secure Budget tracker software, the activity_main.xml and MainActivity.java files' design and implementation choices were guided by this objective. The user experience is improved by the well-organized layout, simple interactions, and appropriate data retrieval techniques. Data security is guaranteed using secure database references and Firebase Authentication. Real-time updates and effective data management are provided via the integration of the Firebase Realtime Database. These choices add up to a strong and trustworthy Budget tracker tool.

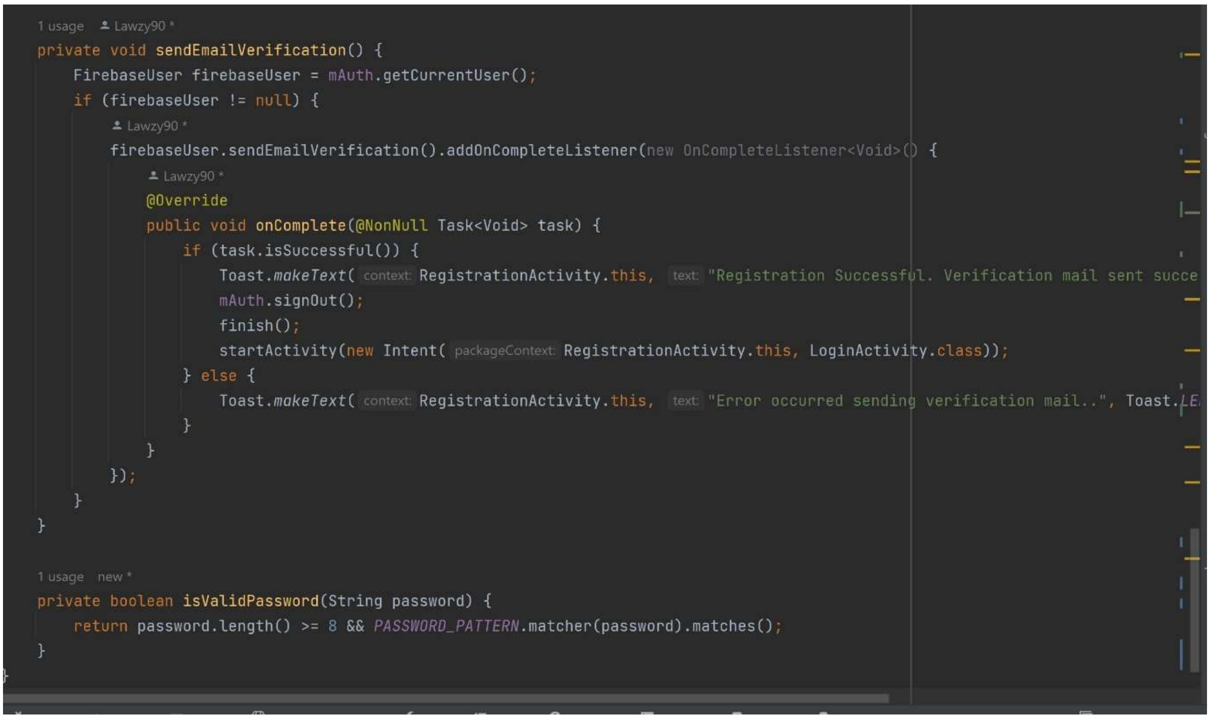
To retrieve expense information from the database, ‘HistoryActivity.java’ code communicates with the backend component of the programme. It stores and retrieves expenditure items using Firebase Realtime Database. The programme creates a connection with Firebase, gets the ID of the current user, and then gets the spending items for the chosen date. It is dependent on the Firebase authentication mechanism, which makes sure that only authorised users can view their cost history. For storing and retrieving expenditure items, it also combines Firebase Realtime Database. To get expenses based on the chosen date, the code uses Firebase database references and queries. The user can then peruse the displayed data in the RecyclerView when it has been fetched.

4.2- Verification and Validation

A crucial component of application security is authentication. It makes sure that only users who have registered can access resources and features. The authentication backend used in this code is Firebase Authentication (Smith, 2016). Services like user registration and authentication, which are essential for security, are provided by Firebase Authentication. When a user registers, the application communicates with Firebase, which subsequently manages the login processes and securely stores the user credentials. An email verification is sent when the registration is complete to make sure the user is using a valid email address that they have access to.

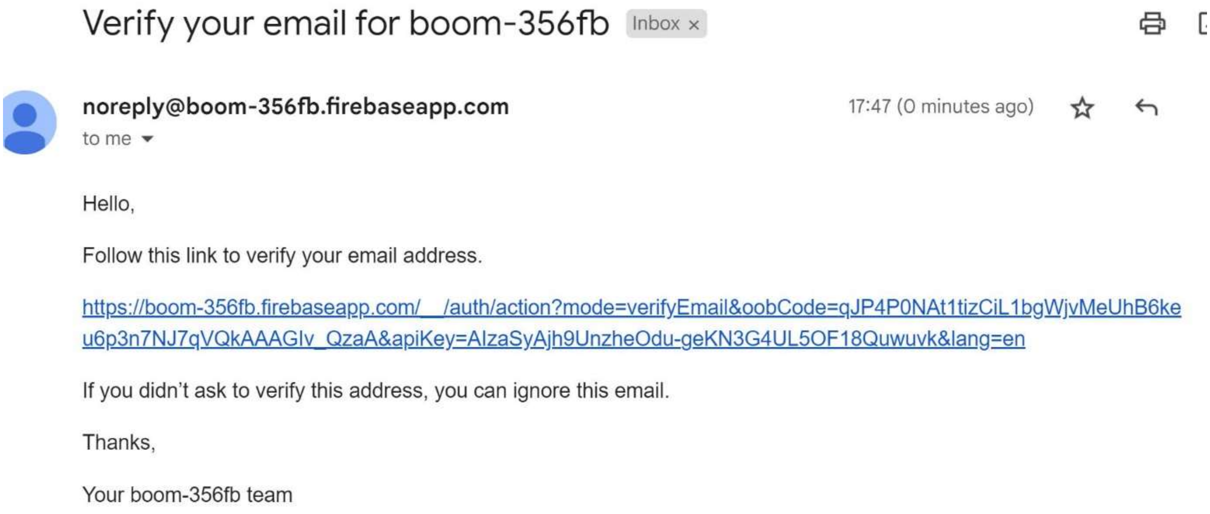
To secure user accounts, it is essential to have a strict password policy. The code enforces a policy requiring passwords to be at least eight characters long and contain both a capital letter and a symbol. These standards can prevent typical password attacks like brute force or dictionary attacks, according to the National Institute of Standards and Technology (NIST) (Grassi, Garcia, & Fenton, 2017). Entropy of the password is increased by requiring at least eight characters, which makes it more difficult for attackers to guess. Assuring complexity by incorporating capital letters and symbols into the password policy reduces the dangers associated with using simple passwords (Seitz, 2018).

Regex expressions are used to enforce the password policy. Pattern matching is a common application for this method, which also works well to verify the strength of passwords (Stubblebine, 2011).



```
1 usage  Lawzy90 *
private void sendEmailVerification() {
    FirebaseUser firebaseUser = mAuth.getCurrentUser();
    if (firebaseUser != null) {
        Lawzy90 *
        firebaseUser.sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {
            Lawzy90 *
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText( context: RegistrationActivity.this,  text: "Registration Successful. Verification mail sent succe
                    mAuth.signOut();
                    finish();
                    startActivity(new Intent( packageContext: RegistrationActivity.this, LoginActivity.class));
                } else {
                    Toast.makeText( context: RegistrationActivity.this,  text: "Error occurred sending verification mail..", Toast.LE
                }
            }
        });
    }
}

1 usage  new *
private boolean isValidPassword(String password) {
    return password.length() >= 8 && PASSWORD_PATTERN.matcher(password).matches();
}
```

Figure 12-– Registration Email verification and password verification



Verify your email for boom-356fb

noreply@boom-356fb.firebaseapp.com

17:47 (0 minutes ago)

to me

Hello,

Follow this link to verify your email address.

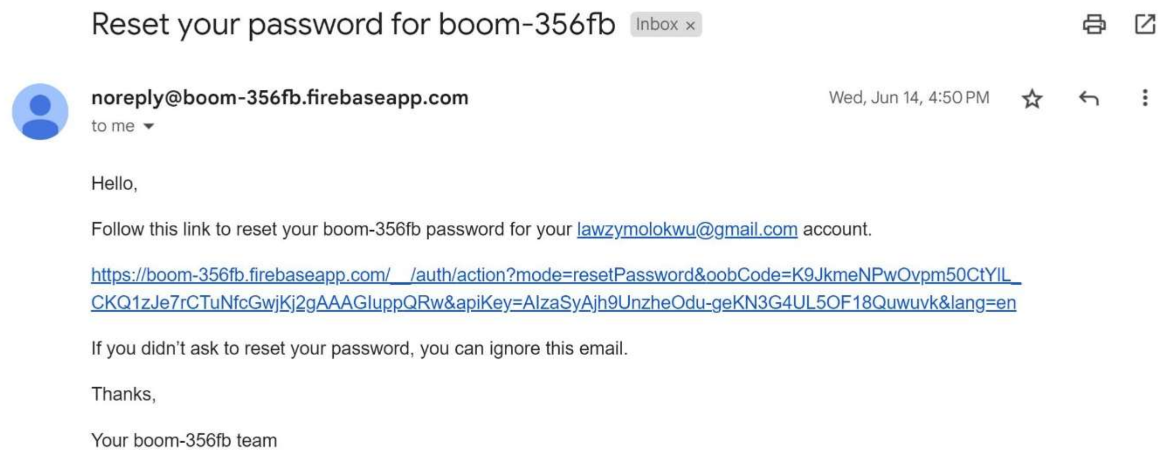
https://boom-356fb.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=qJP4P0NAt1tizCiL1bgWjvMeUhB6keu6p3n7NJ7qVQkAAAGlv_QzaA&apiKey=AIzaSyAjh9UnzheOdu-geKN3G4UL5OF18Quwuvk&lang=en

If you didn’t ask to verify this address, you can ignore this email.

Thanks,

Your boom-356fb team

Figure 13- Example of email sent to the user after successful registration.



Reset your password for boom-356fb

noreply@boom-356fb.firebaseapp.com

Wed, Jun 14, 4:50 PM

Hello,

Follow this link to reset your boom-356fb password for your lawzymolokwu@gmail.com account.

https://boom-356fb.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=K9JkmeNPwOvpm50CtYlL_CKQ1zJe7rCTuNfcGwjKj2gAAAGluppQRw&apiKey=AIzaSyAjh9UnzheOdu-geKN3G4UL5OF18Quwuvk&lang=en

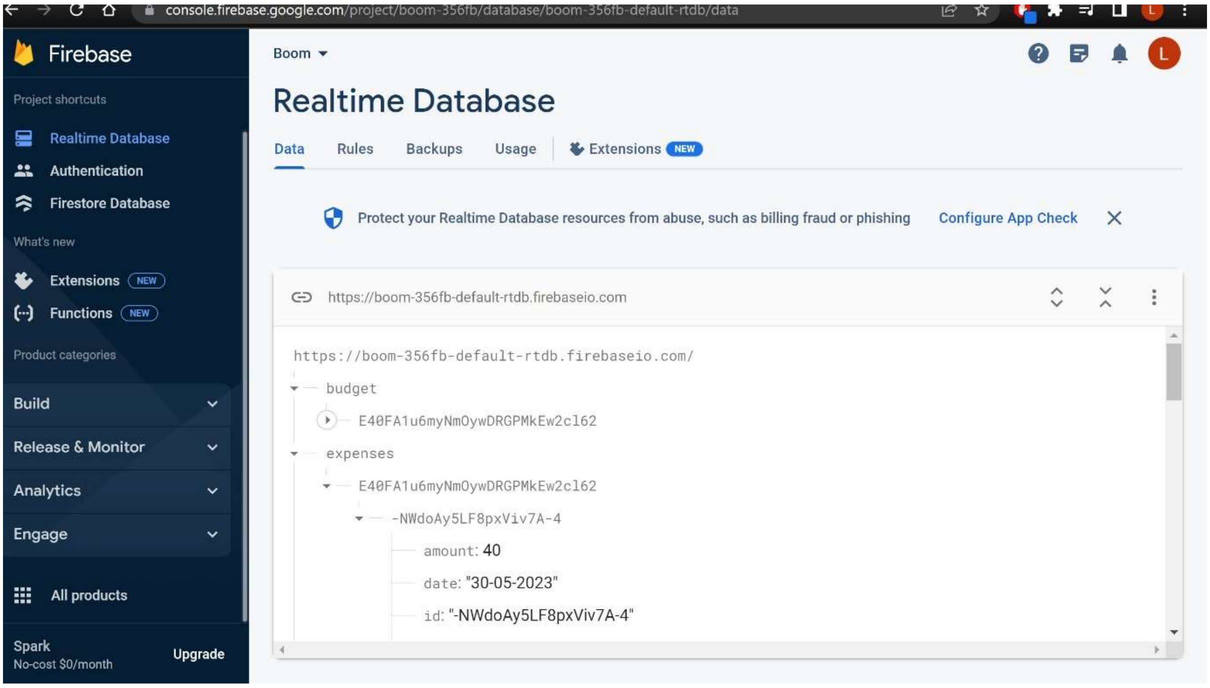
If you didn't ask to reset your password, you can ignore this email.

Thanks,

Your boom-356fb team

Figure 14- Reset password email example.

To store and retrieve data, the app incorporates the Firebase Realtime Database. The MainActivity.java code creates references to the proper database nodes and makes use of event listeners to instantly acquire data. The data is subsequently processed and shown via the user interface (UI). The database integration offers consumers a smooth experience by enabling effective data management, real-time changes, and synchronisation across various devices.



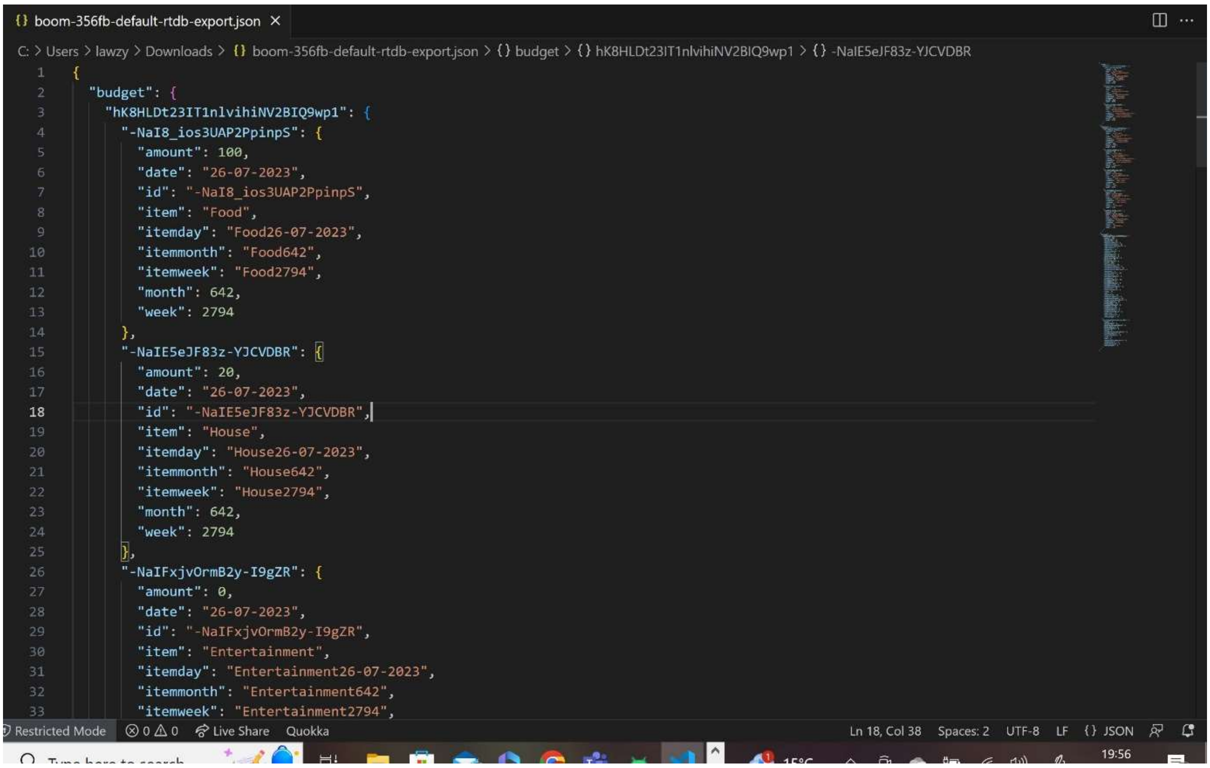
Realtime Database

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://boom-356fb-default-rtdb.firebaseio.com

```
https://boom-356fb-default-rtdb.firebaseio.com/
  budget
    E40FA1u6myNmOywDRGPMkEw2cl62
  expenses
    E40FA1u6myNmOywDRGPMkEw2cl62
      -NWdoAy5LF8pxViv7A-4
        amount: 40
        date: "30-05-2023"
        id: "-NWdoAy5LF8pxViv7A-4"
```

Figure 15 – Database in Firebase



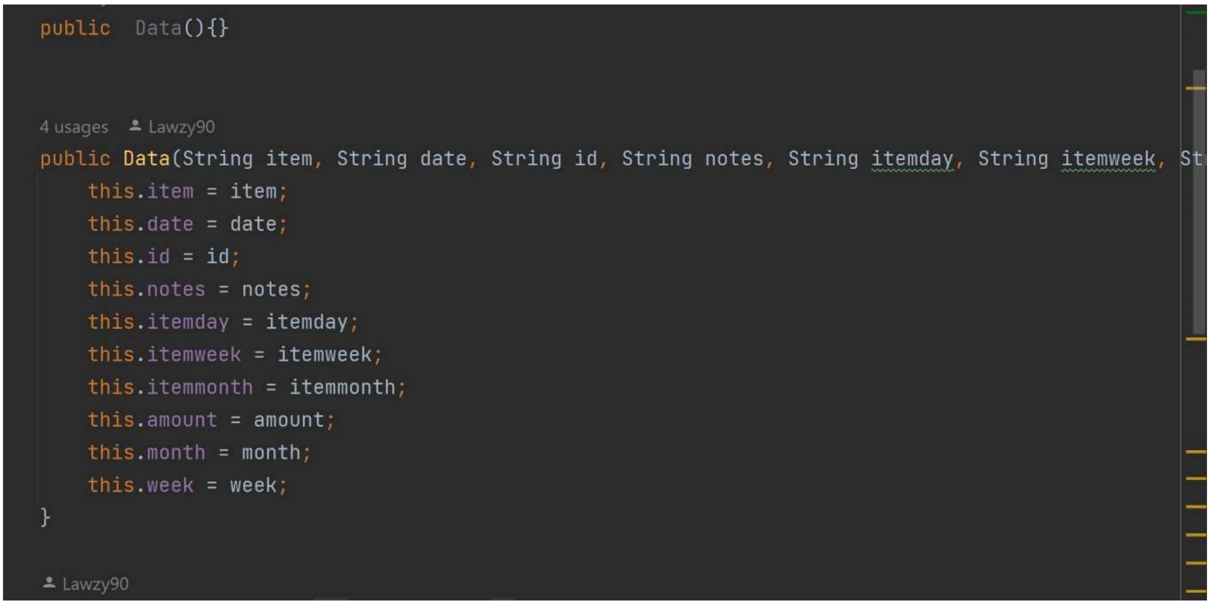
boom-356fb-default-rtdb-export.json
C: > Users > lawzy > Downloads > boom-356fb-default-rtdb-export.json > {} budget > {} hK8HLDt23IT1nlvihiNV2BIQ9wp1 > {} -NaIE5eJF83z-YJCVDBR
Restricted Mode
0 0
Live Share
Quokka
Ln 18, Col 38
Spaces: 2
UTF-8
LF
{} JSON
19:56

```json
{
  "budget": {
    "hK8HLDt23IT1nlvihiNV2BIQ9wp1": {
      "-NaI8_ios3UAP2PpinpS": {
        "amount": 100,
        "date": "26-07-2023",
        "id": "-NaI8_ios3UAP2PpinpS",
        "item": "Food",
        "itemday": "Food26-07-2023",
        "itemmonth": "Food642",
        "itemweek": "Food2794",
        "month": 642,
        "week": 2794
      },
      "-NaIE5eJF83z-YJCVDBR": {
        "amount": 20,
        "date": "26-07-2023",
        "id": "-NaIE5eJF83z-YJCVDBR",
        "item": "House",
        "itemday": "House26-07-2023",
        "itemmonth": "House642",
        "itemweek": "House2794",
        "month": 642,
        "week": 2794
      },
      "-NaIFxjvOrmB2y-I9gZR": {
        "amount": 0,
        "date": "26-07-2023",
        "id": "-NaIFxjvOrmB2y-I9gZR",
        "item": "Entertainment",
        "itemday": "Entertainment26-07-2023",
        "itemmonth": "Entertainment642",
        "itemweek": "Entertainment2794",
```

Figure 16- database as JSON file

To store and retrieve spending data, the TodaySpendingActivity.java backend code interacts with Firebase, an online database service. To retrieve or add expense items, the code connects to Firebase, gets the current user's ID, and then retrieves or adds the appropriate items. The code also contains data manipulation algorithms to compute totals and date-related data. To save and retrieve expenditure items, the programming incorporates Firebase Realtime Database. Each user has their own collection of expenses, and the database is organised based on their own ID. The programme makes queries to gather costs unique to the current date and then computes totals using the information it has obtained.

The Data.java file is designed to work with the backend database. It is implemented using Firebase Realtime Database .The class has attributes for item, date, id, notes, and amount to represent the information necessary for an expense item.



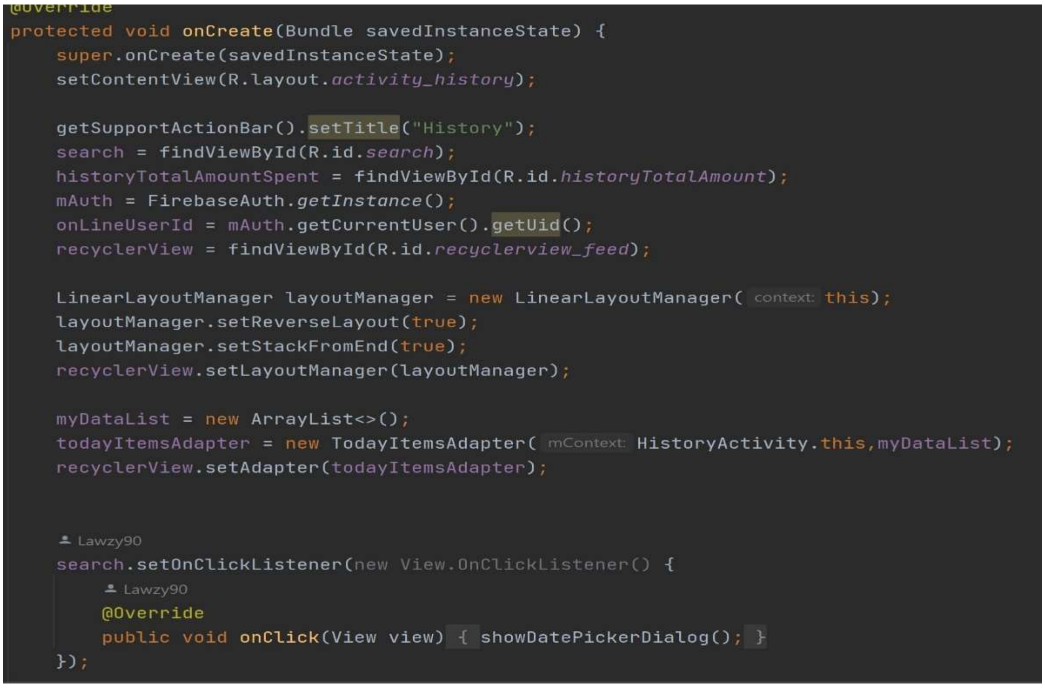
```java
public  Data(){}


4 usages   Lawzy90
public Data(String item, String date, String id, String notes, String itemday, String itemweek, St
    this.item = item;
    this.date = date;
    this.id = id;
    this.notes = notes;
    this.itemday = itemday;
    this.itemweek = itemweek;
    this.itemmonth = itemmonth;
    this.amount = amount;
    this.month = month;
    this.week = week;
}

 Lawzy90
```

Figure 17-data class code snippet with properties and amount to represent the relevant data for an expense item.

4.3 -System History

The HistoryActivity extends AppCompatActivity, a more up-to-date version of the previous Activity class. This enables improved app navigation and content engagement.

The overridden method onCreate is called when the activity is created. The code sets the layout and retrieves references to UI elements inside of the onCreate function.

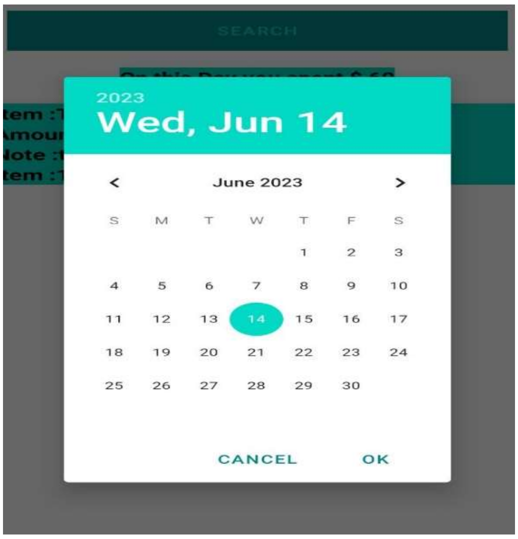


```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_history);

    getSupportActionBar().setTitle("History");
    search = findViewById(R.id.search);
    historyTotalAmountSpent = findViewById(R.id.historyTotalAmount);
    mAuth = FirebaseAuth.getInstance();
    onLineUserId = mAuth.getCurrentUser().getUid();
    recyclerView = findViewById(R.id.recyclerview_feed);

    LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
    layoutManager.setReverseLayout(true);
    layoutManager.setStackFromEnd(true);
    recyclerView.setLayoutManager(layoutManager);

    myDataList = new ArrayList<>();
    todayItemsAdapter = new TodayItemsAdapter( mContext: HistoryActivity.this,myDataList);
    recyclerView.setAdapter(todayItemsAdapter);


     Lawzy90
    search.setOnClickListener(new View.OnClickListener() {
         Lawzy90
        @Override
        public void onClick(View view) { showDatePickerDialog(); }
    });
```

Figure 18-On create method.

When the search button is clicked, a date picker dialogue is displayed. By using showDatePickerDialog, the date picker enables users to select a certain date. The selected date is obtained in the onDateSet method, and the Firebase database is searched to obtain the associated expenditures. The query looks for entries that match the given date and sorts the data by date. This offers a simple method for choosing a date.

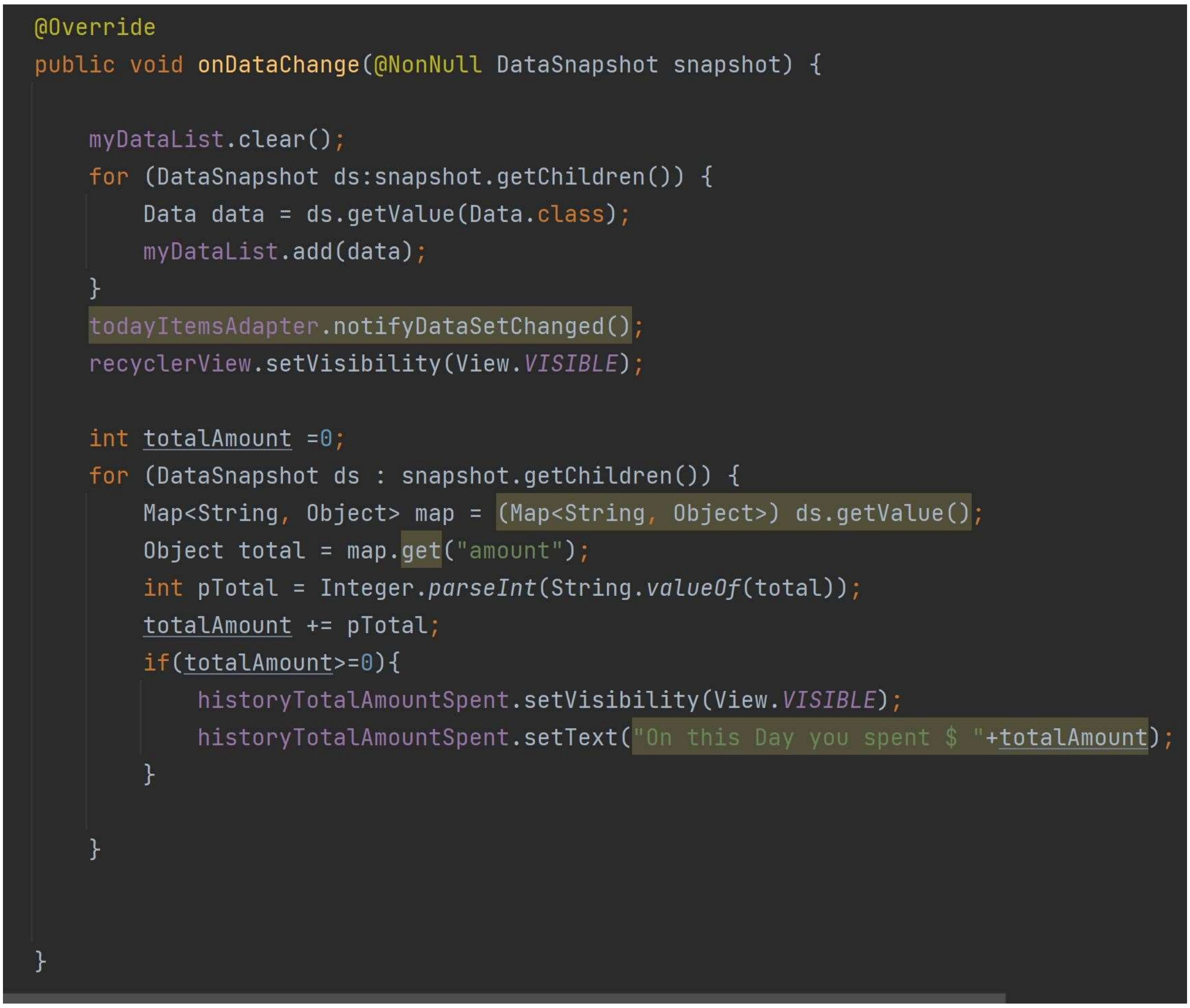
Greater flexibility is possible, and the user is always viewing the most recent data thanks to the option to obtain data in real-time based on the chosen date. Compared to manually entering a date in text format, it is far more intuitive.



S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figure 19-Example of date picker UI

After successfully retrieving data, the onDataChange function is invoked. The procedure initially clears the data that is currently in myDataList.The next step is to loop through the data snapshots that were retrieved, add them to myDataList, and alert the adapter that the data set has changed. Iteratively going over the data and adding up the quantities results in the total amount spent. Along with a text message, the total sum is shown.



```
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {

    myDataList.clear();
    for (DataSnapshot ds:snapshot.getChildren()) {
        Data data = ds.getValue(Data.class);
        myDataList.add(data);
    }
    todayItemsAdapter.notifyDataSetChanged();
    recyclerView.setVisibility(View.VISIBLE);

    int totalAmount =0;
    for (DataSnapshot ds : snapshot.getChildren()) {
        Map<String, Object> map = (Map<String, Object>) ds.getValue();
        Object total = map.get("amount");
        int pTotal = Integer.parseInt(String.valueOf(total));
        totalAmount += pTotal;
        if(totalAmount>=0){
            historyTotalAmountSpent.setVisibility(View.VISIBLE);
            historyTotalAmountSpent.setText("On this Day you spent $ "+totalAmount);
        }

    }


}
```

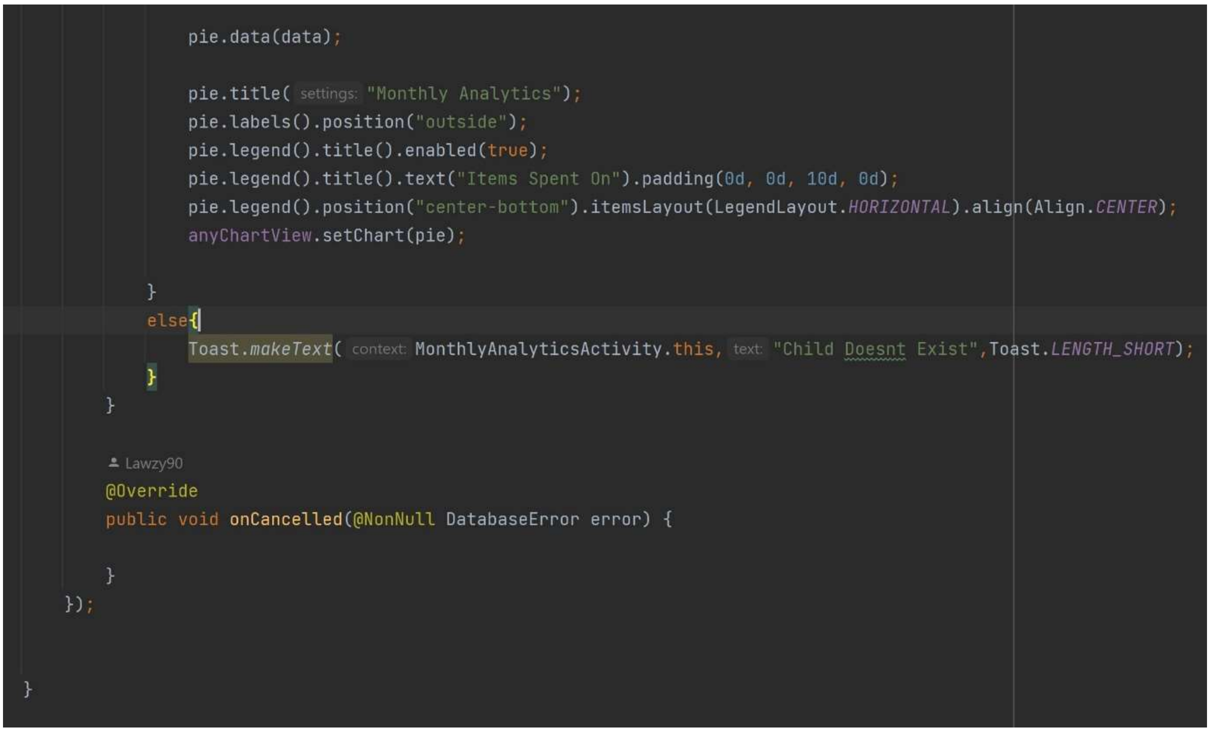
Figure 20- OnData Change function implementation.

4.4- System Analytics

The Analytics feature's user interface prioritises simplicity, clarity, and data visualisation. To display the user with the pertinent data, the XML layout file combines linear layouts, relative layouts, text views, image views, and chart views. The feature's charts show how expenses are distributed among various categories, allowing users to immediately understand their spending habits. The layout is created to be both logical and aesthetically pleasing, improving the user experience.

The backend for the Analytics feature, such as the weeklyAnalytics, monthlyAnalytics, and todayAnalytics, is the Java class with ChooseAnalyticsActivity. To retrieve spending information and determine the overall expenses for each category, it interfaces with Firebase Realtime Database. To guarantee secure access to user-specific data, the class makes advantage of Firebase Authentication. It uses queries to retrieve expenses for the most recent month and uses calculations to figure out how much was spent overall across all categories. The AnyChart library is then used to create charts using the collected data via the LoadGraph() method. To retrieve spending information for the current user, the Analytics feature communicates with the Firebase Realtime Database. It gets the user's unique identifier (UID)'s stored expenditure records. The structure of the spending data includes fields for the item name, amount, and month.

By executing pie.data(data), the data is passed to the pie chart."Monthly Analytics" is an example of the title of the chart.The pie chart's labels for each slice are set up to appear outside of the chart. This can aid in making it obvious what each component stands for without overcrowding the chart area.The chart's legend is activated and set up. The "Items Spent On" legend title is displayed in the bottom centre of the chart. Users can quickly comprehend the significance of the various sections of the chart with the aid of a legend.By executing anyChartView.setChart(pie), the chart is subsequently assigned to anyChartView. The pie chart will appear on the UI as a result.



```
                pie.data(data);

                pie.title( settings: "Monthly Analytics");
                pie.labels().position("outside");
                pie.legend().title().enabled(true);
                pie.legend().title().text("Items Spent On").padding(0d, 0d, 10d, 0d);
                pie.legend().position("center-bottom").itemsLayout(LegendLayout.HORIZONTAL).align(Align.CENTER);
                anyChartView.setChart(pie);

            }
            else{
                Toast.makeText( context: MonthlyAnalyticsActivity.this, text: "Child Doesnt Exist",Toast.LENGTH_SHORT);
            }
        }

        Lawzy90
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });


}
```

Figure 21-example of how pie chart is implemented to view analytics.

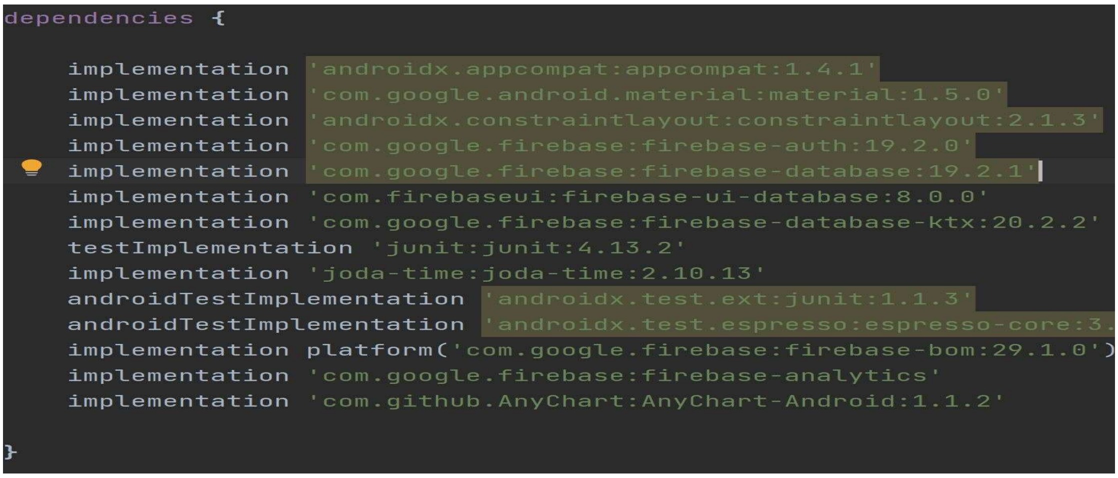
The user sees a Toast notification with the phrase "Child Doesn't Exist" if the snapshot is missing. This is a straightforward technique to let the user know that the data is not accessible. Although the onCancelled() method has been implemented, when the listener is unable to receive data, this function is called.

The Java programme retrieves pertinent expenses and determines the overall amount spent in each category by making a query to the database based on the current month. The smooth data retrieval and analysis made possible by this connection gives users precise insights into their purchasing patterns.

The Analytics feature uses Firebase Authentication to protect user privacy and data security. Before accessing their analytics, users must verify themselves using their credentials. This guards against unauthorised access to private financial data. To further safeguard user data from unauthorised access, the Firebase Realtime Database rules are set up to limit read and write access to only authenticated users.

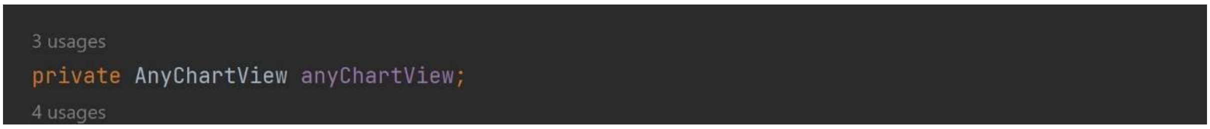
The layout's focus is on providing information in a clear, orderly manner. The combination of linear and relative layouts enables simple scaling across various screen sizes and flexible placement. The chosen colour palette improves readability and helps distinguish distinct sections visually.

Visually pleasing and interactive charts are made using the AnyChart framework. Pie charts are used because they are an effective way to display proportional data and make it simple for consumers to see how expenses are distributed across various categories. The charts are placed prominently to maximise their exposure and user effect. The project's build.gradle(Module:App) file needs to include a dependency for the AnyChart library. It is necessary to declare the AnyChartView element to display the chart in the XML layout. Here is an illustration.



```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'com.google.firebase:firebase-database:19.2.1'
    implementation 'com.firebaseui:firebase-ui-database:8.0.0'
    implementation 'com.google.firebase:firebase-database-ktx:20.2.2'
    testImplementation 'junit:junit:4.13.2'
    implementation 'joda-time:joda-time:2.10.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.
    implementation platform('com.google.firebase:firebase-bom:29.1.0')
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.github.AnyChart:AnyChart-Android:1.1.2'

}
```

Figure 22-Chart dependency



```
3 usages
private AnyChartView anyChartView;
4 usages
```

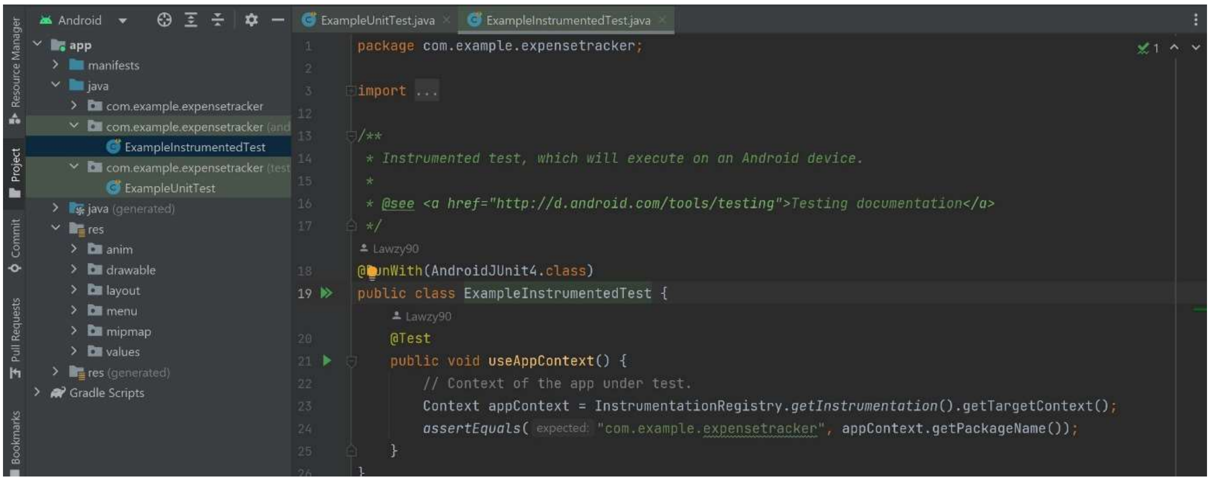
Figure 23-Analytics charts implementation

Chapter 5 - Testing

To assure the dependability and functioning of the programme, testing was carried out using Android Studio. This was done to find and repair errors, check intended behaviour, and confirm the app's compatibility with various Android versions and devices by developing and running tests inside the development environment. This results in a more reliable and user-friendly application. Below are some of the tests conducted:

5.1 -Junit test

A framework for creating and executing automated tests in Java is called JUnit testing. It is done to ensure that code is valid by developing test cases that examine if desired and actual behaviour match. JUnit testing enhances code quality, finds errors earlier, and acts as a backup plan for refactoring and future changes.

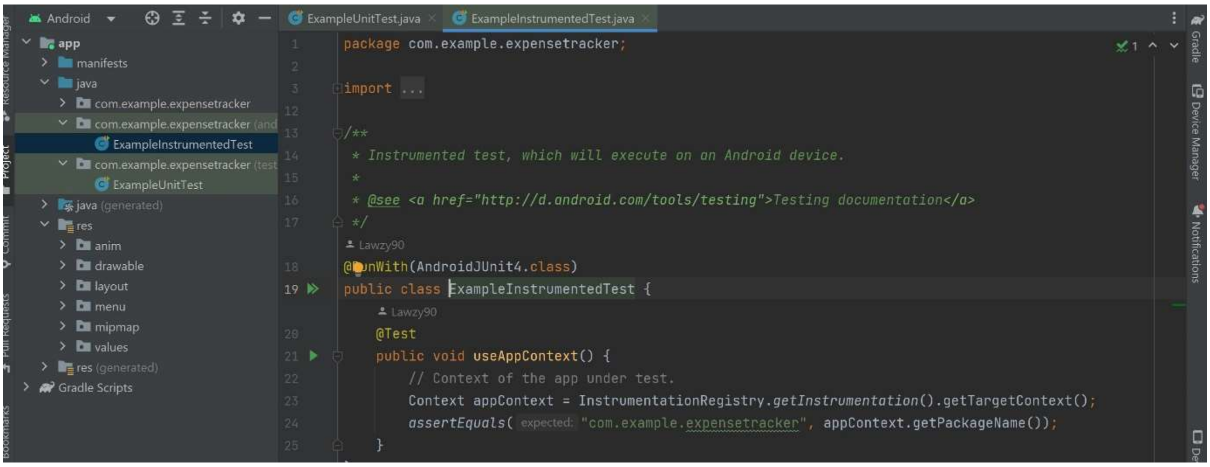


```
package com.example.expensetracker;

import ...

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
Lawzy90
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    Lawzy90
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals( expected: "com.example.expensetracker", appContext.getPackageName());
    }
}
```

Figure 24- Junit test carried out on “Boom” app.

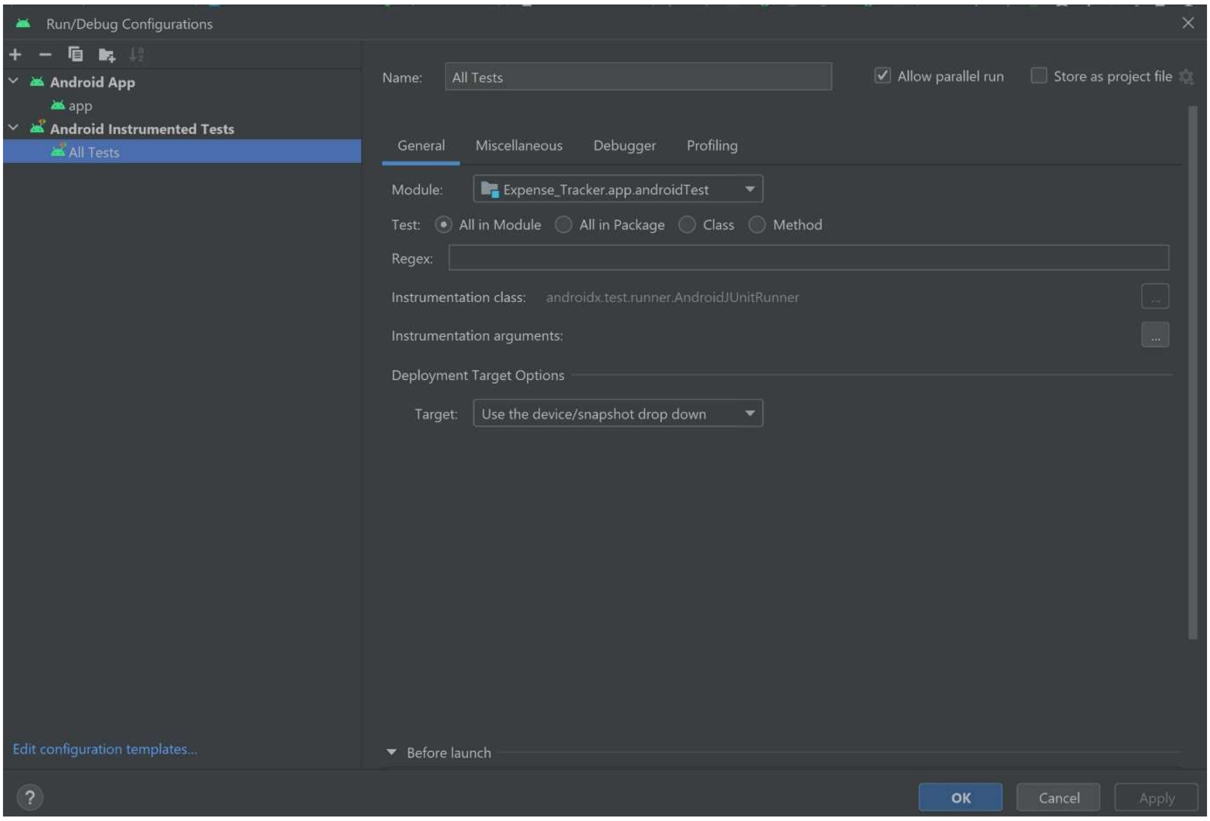
5.2 -Instrumentation testing

To verify the behaviour of an app's user interface and its interactions with the underlying system components, instrumentation testing, sometimes referred to as UI testing or functional testing, is conducted in the Android development process. On an actual or simulated Android device, tests are run. Instrumentation tests increase trust in the overall app quality and user experience, help to guarantee that the app works correctly in a real-world setting, and help to identify UI regressions. The testing was run across different virtual devices to ensure the usability of the device on different android versions and configurations.



```
package com.example.expensetracker;

import ...

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals( expected: "com.example.expensetracker", appContext.getPackageName());
    }
```

Figure 25



Name: All Tests

☑ Allow parallel run □ Store as project file

Module: Expense_Tracker.app.androidTest

Test: ● All in Module ○ All in Package ○ Class ○ Method

Regex:

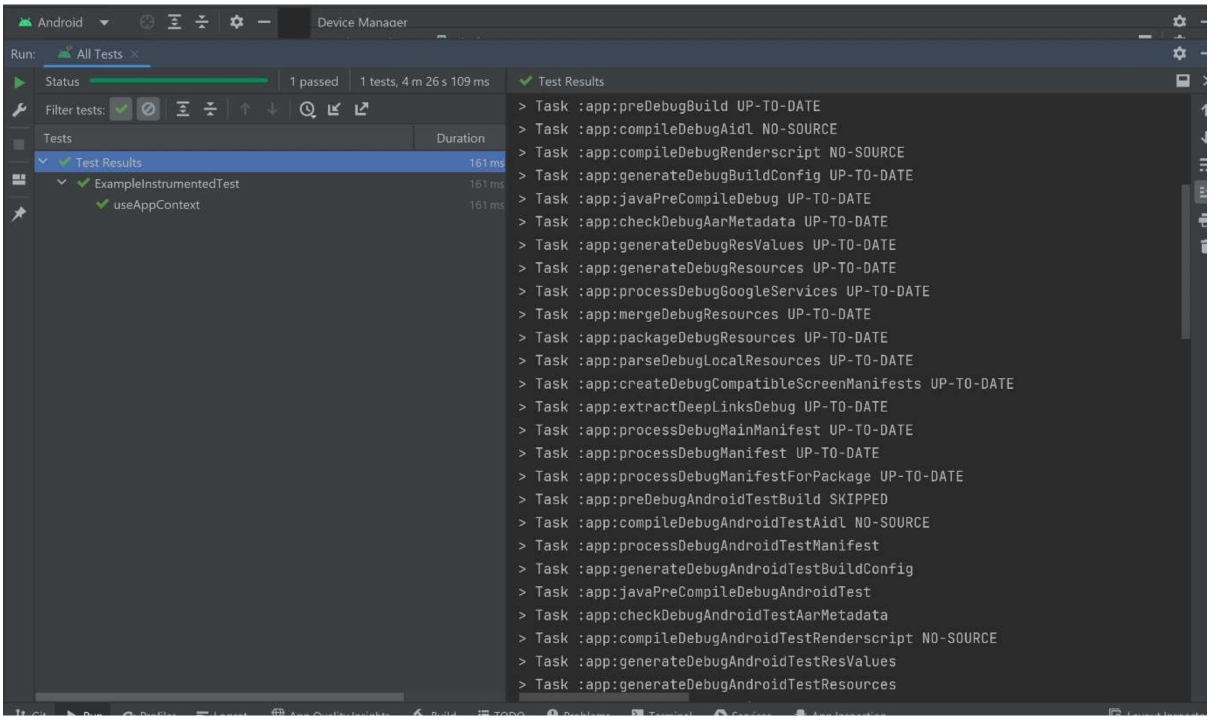
Instrumentation class: androidx.test.runner.AndroidJUnitRunner

Instrumentation arguments:

Deployment Target Options

Target: Use the device/snapshot drop down

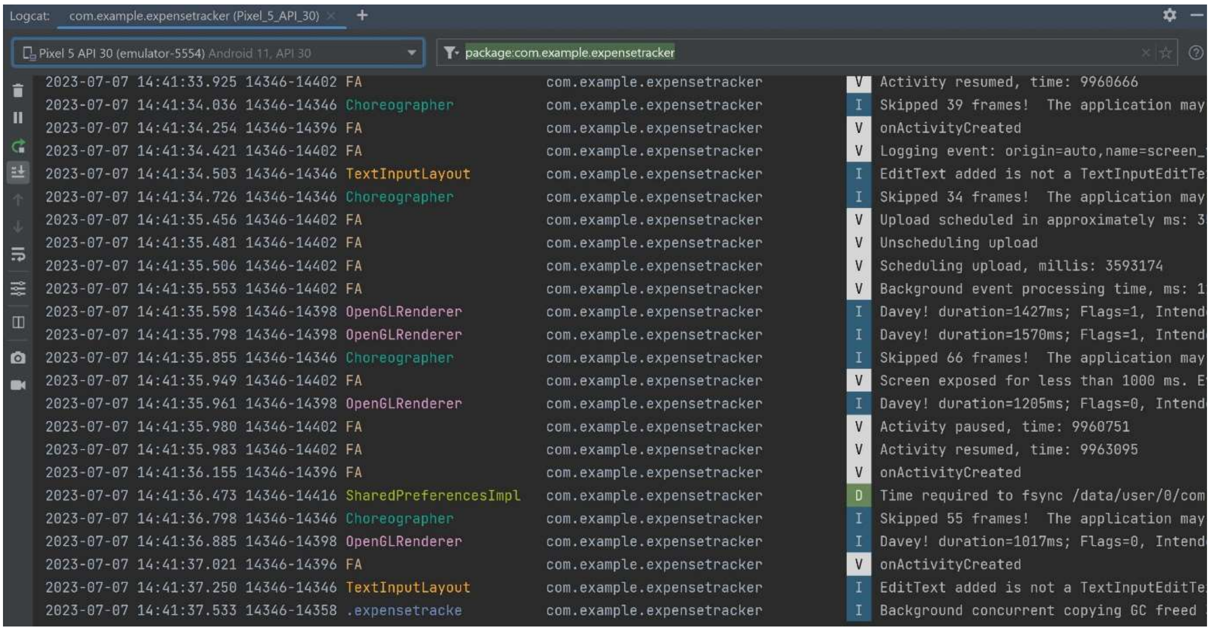
Figure 26-Instrumented Testing Implementation and environment .



```
Status    1 passed    1 tests, 4 m 26 s 109 ms

Test Results                    161 ms
  ExampleInstrumentedTest       161 ms
    useAppContext               161 ms
```

```
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:compileDebugAidl NO-SOURCE
> Task :app:compileDebugRenderscript NO-SOURCE
> Task :app:generateDebugBuildConfig UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:generateDebugResources UP-TO-DATE
> Task :app:processDebugGoogleServices UP-TO-DATE
> Task :app:mergeDebugResources UP-TO-DATE
> Task :app:packageDebugResources UP-TO-DATE
> Task :app:parseDebugLocalResources UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinksDebug UP-TO-DATE
> Task :app:processDebugMainManifest UP-TO-DATE
> Task :app:processDebugManifest UP-TO-DATE
> Task :app:processDebugManifestForPackage UP-TO-DATE
> Task :app:preDebugAndroidTestBuild SKIPPED
> Task :app:compileDebugAndroidTestAidl NO-SOURCE
> Task :app:processDebugAndroidTestManifest
> Task :app:generateDebugAndroidTestBuildConfig
> Task :app:javaPreCompileDebugAndroidTest
> Task :app:checkDebugAndroidTestAarMetadata
> Task :app:compileDebugAndroidTestRenderscript NO-SOURCE
> Task :app:generateDebugAndroidTestResValues
> Task :app:generateDebugAndroidTestResources
```

Figure 27-Testing outputs.


Logcat:
com.example.expensetracker (Pixel_5_API_30)
Pixel 5 API 30 (emulator-5554) Android 11, API 30
package:com.example.expensetracker

```
2023-07-07 14:41:33.925 14346-14402 FA                    com.example.expensetracker    V  Activity resumed, time: 9960666
2023-07-07 14:41:34.036 14346-14346 Choreographer         com.example.expensetracker    I  Skipped 39 frames!  The application may
2023-07-07 14:41:34.254 14346-14396 FA                    com.example.expensetracker    V  onActivityCreated
2023-07-07 14:41:34.421 14346-14402 FA                    com.example.expensetracker    V  Logging event: origin=auto,name=screen_
2023-07-07 14:41:34.503 14346-14346 TextInputLayout       com.example.expensetracker    I  EditText added is not a TextInputEditTe
2023-07-07 14:41:34.726 14346-14346 Choreographer         com.example.expensetracker    I  Skipped 34 frames!  The application may
2023-07-07 14:41:35.456 14346-14402 FA                    com.example.expensetracker    V  Upload scheduled in approximately ms: 3
2023-07-07 14:41:35.481 14346-14402 FA                    com.example.expensetracker    V  Unscheduling upload
2023-07-07 14:41:35.506 14346-14402 FA                    com.example.expensetracker    V  Scheduling upload, millis: 3593174
2023-07-07 14:41:35.553 14346-14402 FA                    com.example.expensetracker    V  Background event processing time, ms: 1
2023-07-07 14:41:35.598 14346-14398 OpenGLRenderer        com.example.expensetracker    I  Davey! duration=1427ms; Flags=1, Intend
2023-07-07 14:41:35.798 14346-14398 OpenGLRenderer        com.example.expensetracker    I  Davey! duration=1570ms; Flags=1, Intend
2023-07-07 14:41:35.855 14346-14346 Choreographer         com.example.expensetracker    I  Skipped 66 frames!  The application may
2023-07-07 14:41:35.949 14346-14402 FA                    com.example.expensetracker    V  Screen exposed for less than 1000 ms. E
2023-07-07 14:41:35.961 14346-14398 OpenGLRenderer        com.example.expensetracker    I  Davey! duration=1205ms; Flags=0, Intend
2023-07-07 14:41:35.980 14346-14402 FA                    com.example.expensetracker    V  Activity paused, time: 9960751
2023-07-07 14:41:35.983 14346-14402 FA                    com.example.expensetracker    V  Activity resumed, time: 9963095
2023-07-07 14:41:36.155 14346-14396 FA                    com.example.expensetracker    V  onActivityCreated
2023-07-07 14:41:36.473 14346-14416 SharedPreferencesImpl com.example.expensetracker    D  Time required to fsync /data/user/0/com
2023-07-07 14:41:36.798 14346-14346 Choreographer         com.example.expensetracker    I  Skipped 55 frames!  The application may
2023-07-07 14:41:36.885 14346-14398 OpenGLRenderer        com.example.expensetracker    I  Davey! duration=1017ms; Flags=0, Intend
2023-07-07 14:41:37.021 14346-14396 FA                    com.example.expensetracker    V  onActivityCreated
2023-07-07 14:41:37.250 14346-14346 TextInputLayout       com.example.expensetracker    I  EditText added is not a TextInputEditTe
2023-07-07 14:41:37.533 14346-14358 .expensetracke        com.example.expensetracker    I  Background concurrent copying GC freed
```

Figure 28- Example of app debug logcat on virtual device

In addition to JUnit tests and Instrumentation testing, a number of other tests listed in Appendix A were used to evaluate the app's verification, validation, and usability. These tests covered a range of topics, including user experience, performance evaluation, security analysis, and functional correctness. Through the execution of these exhaustive tests, a full assessment of the app's quality, dependability, and user happiness was made, guaranteeing that it adhered to set criteria and requirements.

Chapter 6- critical Analysis:

6.1 - Introduction

The Agile methodology, recognised for its gradual and iterative approach, is being scrutinised for how well it promotes adaptability, teamwork, and ongoing feedback. Software choices like Android Studio, Java, and Firebase are examined for compatibility and effectiveness in accomplishing the objectives of the application. Additionally, the effectiveness of GitHub and OneDrive in promoting a collaborative atmosphere for development and research is examined. These platforms have been used for code backup and documentation, respectively. Additionally, the methods and resources employed in building the layouts of the app within Android Studio are assessed for their impact on the user experience.

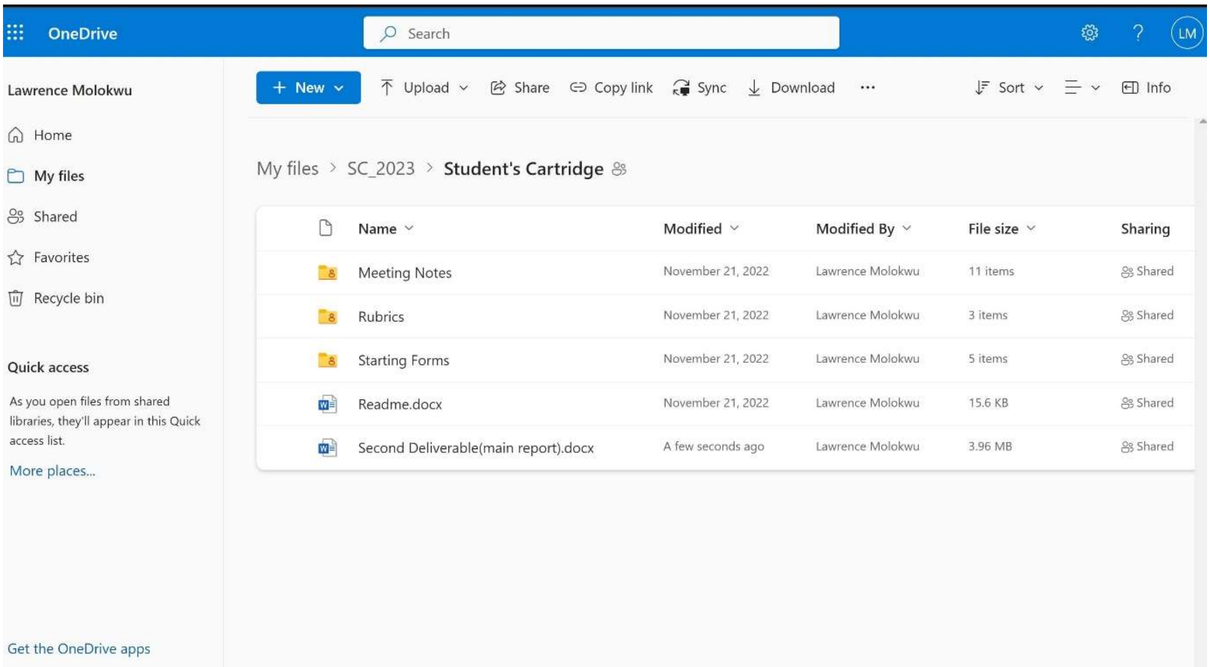
Finally, the use of libraries and tools, such as the AnyChart library, is examined critically for how they contribute to the expansion of functions and the facilitation of data visualisation. This analysis tries to clarify the effects and efficacy of the development methods and tools selected in the production of the "Boom" application.

6.2 - Software and tools used.

The "Boom" application's core technologies include Android Studio, Java, and Firebase. An Integrated Development Environment (IDE) from Google designed exclusively for Android app development is called Android Studio. According to Majumdar et al. (2018), Android Studio provides a variety of tools for code editing, debugging, and testing, all of which are crucial for creating reliable applications. Java's use as a programming language is supported by its object-oriented structure, which encourages code reuse and maintainability. Oracle (2021) claims that Java has a sizable ecosystem and is renowned for its powerful memory management and cross-platform capabilities. Java is therefore a sensible option for a programme that is meant to grow and change over time. As the backend service, Firebase provides several features, including authentication, real-time database, and analytics. Firebase’s real-time database is particularly relevant for a budget-tracking application, where timely synchronization of data is critical.

6.3 - Research and Development Environment

The "Boom" application's research and development environment is made up of GitHub and OneDrive. For version control and code backup, GitHub is used. It is feasible to efficiently collaborate and track code changes by utilising GitHub. Documents from research are saved to OneDrive and shared with supervisor. Using cloud storage services like OneDrive makes sure that documents are accessible from anywhere and protected from data loss. Additionally, the development process uses a variety of data sources, including Stack Overflow, YouTube, and the Firebase community, for help and learning. In line with the Agile methodology's emphasis on adaptation and evolution, these platforms are essential for problem-solving and ongoing learning.



As you open files from shared libraries, they'll appear in this Quick access list.

My files > SC_2023 > Student's Cartridge

Name	Modified	Modified By	File size	Sharing
Meeting Notes	November 21, 2022	Lawrence Molokwu	11 items	Shared
Rubrics	November 21, 2022	Lawrence Molokwu	3 items	Shared
Starting Forms	November 21, 2022	Lawrence Molokwu	5 items	Shared
Readme.docx	November 21, 2022	Lawrence Molokwu	15.6 KB	Shared
Second Deliverable(main report).docx	A few seconds ago	Lawrence Molokwu	3.96 MB	Shared

Figure 29- Student one drive cartridge

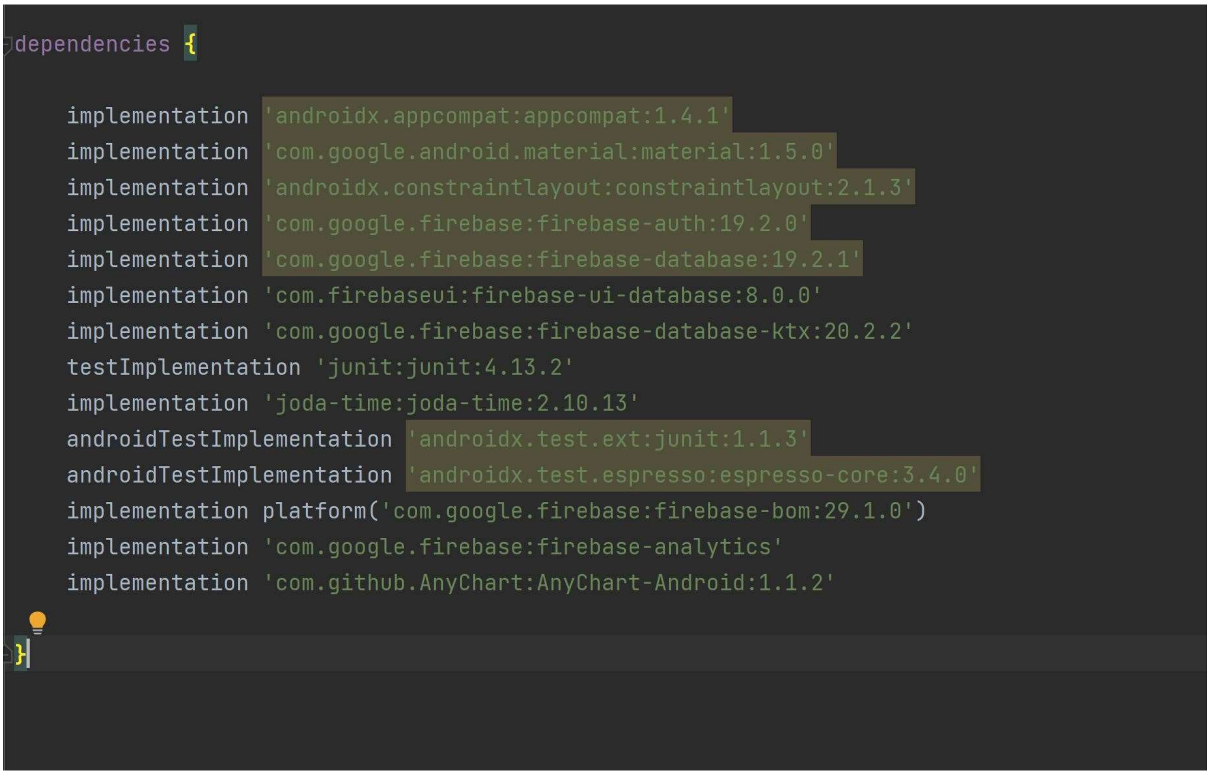
6.4 - Libraries used.

The build. Gradle file, which is a crucial component of the Gradle build system that Android Studio uses, implements the dependencies and libraries.

Effective dependency management is possible thanks to the build. Gradle file. Gradle can automatically download and integrate the resources the application needs by defining the libraries and dependencies it needs. Developers no longer need to manually download and handle library files, thanks to this. Developers can define the version of the library or dependency they want to use in the build. Gradle file. This is crucial since different libraries may have various functionalities or incompatibilities between versions. Gradle employs a number of methods that are optimised for reducing build times, such as incremental builds. Gradle can intelligently decide which components of the build tree need to be rebuilt when dependencies are indicated in the build.gradle file, thereby saving time.

The code for the external libraries is kept apart from the code for the application by adding dependencies to the build.gradle file. Because of this separation, it is simpler to update or replace libraries without affecting the application's code.

The AnyChart Library is employed to facilitate the creation of visually appealing charts within the application. It is essential for the history analytics feature, where users can see their spending habits in pie charts.



```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'com.google.firebase:firebase-database:19.2.1'
    implementation 'com.firebaseui:firebase-ui-database:8.0.0'
    implementation 'com.google.firebase:firebase-database-ktx:20.2.2'
    testImplementation 'junit:junit:4.13.2'
    implementation 'joda-time:joda-time:2.10.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation platform('com.google.firebase:firebase-bom:29.1.0')
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.github.AnyChart:AnyChart-Android:1.1.2'

}
```

Figure 30– dependencies for Anychart library.

6.5 - Major Dependencies Explained:

- implementation 'com.google.firebase:firebase-auth:19.2.0' - Firebase Authentication library to facilitate user authentication through various means such as email, Google, etc.
- implementation 'com.google.firebase:firebase-database:19.2.1' - This library enables interaction with Firebase Real-time Database, allowing for the storage and retrieval of user data.
- implementation 'joda-time:joda-time:2.10.13' - Joda-Time is used for better handling and formatting of date and time.
- implementation 'com.github.AnyChart:AnyChart-Android:1.1.2' - As mentioned earlier, this is used for creating interactive and visually appealing charts for history analytics.

6.6 - Layout Design in Android Studio

Any programme must have user-friendly and intuitive layouts. The layout editor in Android Studio is employed in the "Boom" application. For designing layouts, Android Studio offers drag-and-drop capabilities, XML editors, and a preview pane. With the use of these characteristics, developers can produce responsive layouts that change to accommodate various screen sizes and orientations. Hussain et al. (2012) assert that an application's layout directly affects user satisfaction and usability3. The layout of the application must be both aesthetically beautiful and useful as well as responsive to match user expectations.

6.7 – Risk assessment

The development of the application involved risk assessment heavily. Early on, a number of hazards were recognised, including technical difficulties, tool restrictions, and database compatibility. While some risks were successfully managed, such as the substitution of Firebase for MongoDB due to compatibility difficulties, others became known during the course of the project. Performance concerns with the Android Studio emulator and communication issues with the first supervisor were a couple of them. To quickly resolve concerns, future projects would benefit from more thorough risk assessment and contingency planning, particularly in terms of tool and technology selection.

6.8 - Challenges and issues

- Emulator Performance and Crashes: One of the biggest issues was Android Studio's emulator's sluggish performance and frequent crashes. This can dramatically lower testing and development productivity. More resources may have been given to the emulator to address this problem, or alternative emulators with better performance could have been investigated. Additionally, the emulator's speed might have been enhanced by code optimisation and a reduction in resource-intensive app operations.
- Glitches: Due to the poor running environment , the emulator will often cause glitches when values are inputted in the system which will later reset to the original value when the app is re-run .This was not ideal especially when testing the apps input functionality.
- Limitations of Firebase's Trial Mode: After the trial period ended, using Firebase for user verification posed problems. It is critical to understand the constraints of trial versions and make appropriate plans. It would have been advantageous in this situation to investigate Firebase's pricing options and select the best one for the app's requirements.
- Database Choice: There were problems when MongoDB was used as the app's database in the beginning. For an application to function well, scale easily, and be simple to integrate, the right database must be chosen. In retrospect, making a more educated choice would have resulted from carefully investigating and assessing the available database solutions and their suitability for the app's requirements.
- Lack of Support and Communication from the First Supervisor: The success of any project depends on having a supportive and communicative supervisor. The first supervisor, however, failed to provide the essential encouragement and communication, which influenced the development process. This problem might have been resolved by looking for more assistance from other sources, including online forums or communities, or by asking for advice from a different supervisor.

- Virtual Device Creation Difficulties: Using Android Studio, it was tough to create virtual devices. This problem might have been solved by investigating different approaches or investigating different emulators to build virtual devices.

6.9 - Achievements

The "Boom" budget cost tracking application was successfully constructed using Android Studio and Firebase, despite the difficulties encountered. This indicates the potency of the selected technology and development tools. The app's backend database of choice turned out to be Firebase. Scalability, seamless connection with Android applications, and real-time data synchronisation were all made possible.

A positive element of the development process was the change to a new supportive supervisor who offered support and direction. Having a supportive boss makes it easier to overcome obstacles and encourages improved communication and teamwork.

6.10 – Future Recommendations:

Place a Strong Emphasis on Resource Allocation: By making sure that the development environment has enough CPU, memory, and disc space, you may prevent performance problems like sluggish emulators or crashes. Perform a thorough examination of the database possibilities, considering aspects like performance, scalability, community support, and licencing constraints. The evaluation will assist in determining which database solution is best for the project.

Implementing a user-friendly mapping system : User IDs are encrypted in the Firebase database of the Application, increasing security but making user tracking more difficult. Data management is hampered when usernames are encrypted and replaced with cryptic IDs in a downloaded JSON file. The development of a user-friendly mapping system connecting encrypted IDs to readable user information, as well as the provision of tools for controlled decryption within the administrative interface, are recommended improvements. These upgrades would streamline user data tracking without sacrificing data protection, striking a balance between security and accessibility. The administrative aspects of the programme will operate more effectively and be easier to use if these suggestions are fully implemented.

Chapter 7 -Conclusion

In conclusion, the "Boom" budget spending monitoring application goes through a well-planned development lifecycle that adheres to the Agile methodology. This lifespan includes phases including project planning, requirements gathering, design, implementation, testing, deployment, and post-deployment monitoring. For users looking to effectively manage their budgets, the application's spending tracking on several time periods and history insights in pie charts make it a helpful resource. Because of its user-centred design and strong functionality, it has the potential to become a vital tool for managing personal finances.

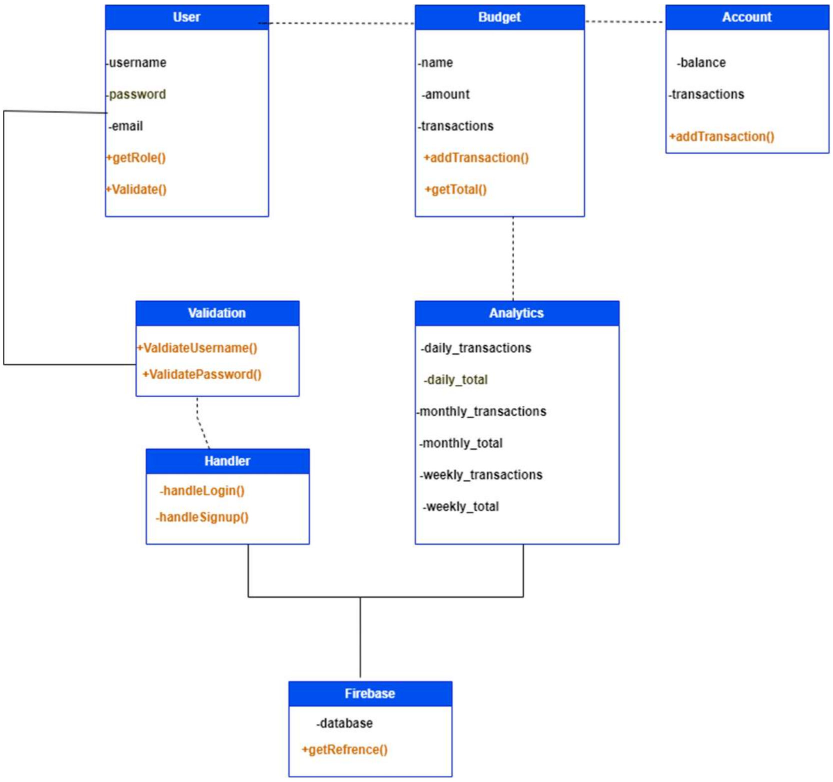
Appendices

Appendix – A

Test Case ID	Test Scenario	Test case Description	Expected Result	Passed
1	Login functionality and authentication	The test case verifies that when a user enters a valid email address and password, the app logs the user in.	The app logs the user in and displays the main screen	YES
2	User Registration	The test case verifies that when a user enters valid registration information, the app creates a new account and logs the user in.	The user should be successfully registered and logged in	YES
3	User logout	The test case verifies that when user logs out, the app correctly logs the user out	The test case verifies that when user logs out	YES
4	Firebase Database Connection	The test case verifies that the app can successfully connect to the Firebase database and perform basic read and write operations.	the app should be able to connect to the Firebase database, and data should be able to be read and written to the database without error.	YES
5	Successful Data Retrieval from Database	The test case verifies that the app can successfully retrieve data from the Firebase database	The app should successfully retrieve data from the Firebase database and display it to the user	YES
5	Updating a budget item	The test case verifies that when a user updates a budget item, the app correctly updates the corresponding data in the database and reflects the changes in the user interface	The budget item should be successfully updated, and the changes should be reflected in the user interface and the database	YES

6	**Deleting a budget item**	**The test case verifies that when a user deletes a budget item, the app correctly deletes the corresponding data from the database and removes it from the user interface**	**The budget item should be successfully deleted from the user interface and the database**	**YES**
9	**Updating expense item**	**The test case verifies that when a user updates an expense item, the app correctly updates the corresponding data in the database and reflects the changes in the user interface and total budget**	**The expense item should be successfully updated, and the changes should be reflected in the user interface, the database and the total budget**	**YES**
10	**Deleting expense item**	**The test case verifies that when a user deletes an expense item, the app correctly deletes the corresponding data from the database and removes it from the user interface and the total budget**	**The expense item should be successfully deleted from the user interface, the database, and the total budget**	**YES**
11	**Fetching weekly expenditure**	**The test case verifies that the app can successfully retrieve and display the weekly expenditure data from the Firebase database.**	**The app should be able to retrieve the weekly expenditure data from the Firebase database and display it to the user without error.**	**YES**
12	**Fetching monthly expenditure**	**The test case verifies that the app can successfully retrieve and display the monthly expenditure data from the Firebase database.**	**The app should be able to retrieve the monthly expenditure data from the Firebase database and display it to the user without error.**	**YES**
13	Error messaging	The test case verifies that the app's displays error messaging in a clear and consistent manner.	The app's displays error messaging should be clear and consistent, providing information about the error and how to resolve it in a user-friendly way	YES
14	**Data Visualization**	**The test case verifies that the app can visualize the analytics data in different formats such as charts**	**The app should be able to visualize the analytics data in formats such as charts, to help the user better understand the data.**	**YES**

Appendix – B

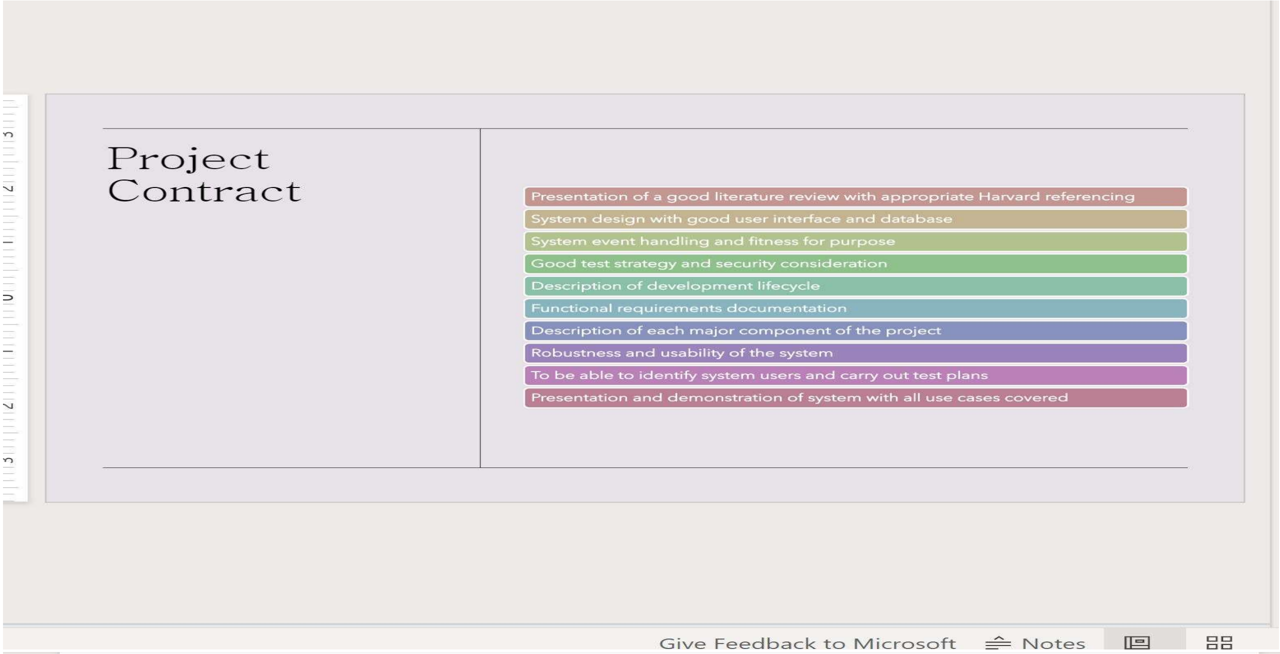


Appendix – C

[Project Contract]

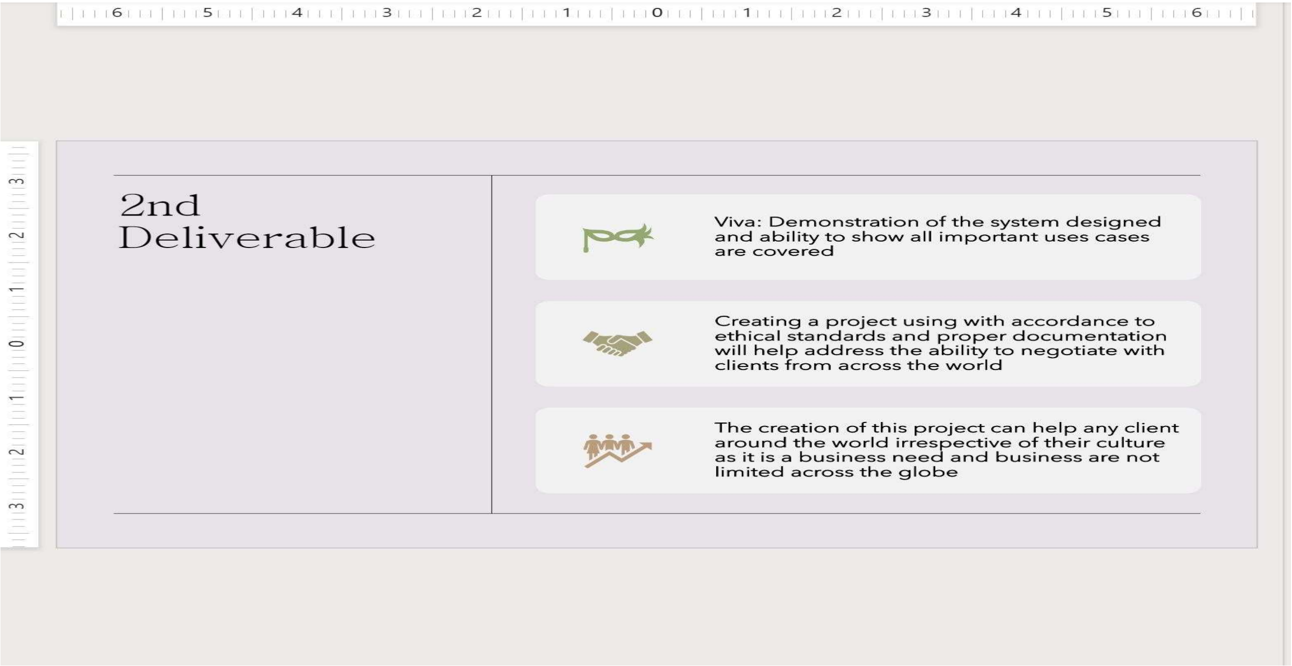
[Meeting Notes]

Appendix – D



Project Contract

- Presentation of a good literature review with appropriate Harvard referencing
- System design with good user interface and database
- System event handling and fitness for purpose
- Good test strategy and security consideration
- Description of development lifecycle
- Functional requirements documentation
- Description of each major component of the project
- Robustness and usability of the system
- To be able to identify system users and carry out test plans
- Presentation and demonstration of system with all use cases covered



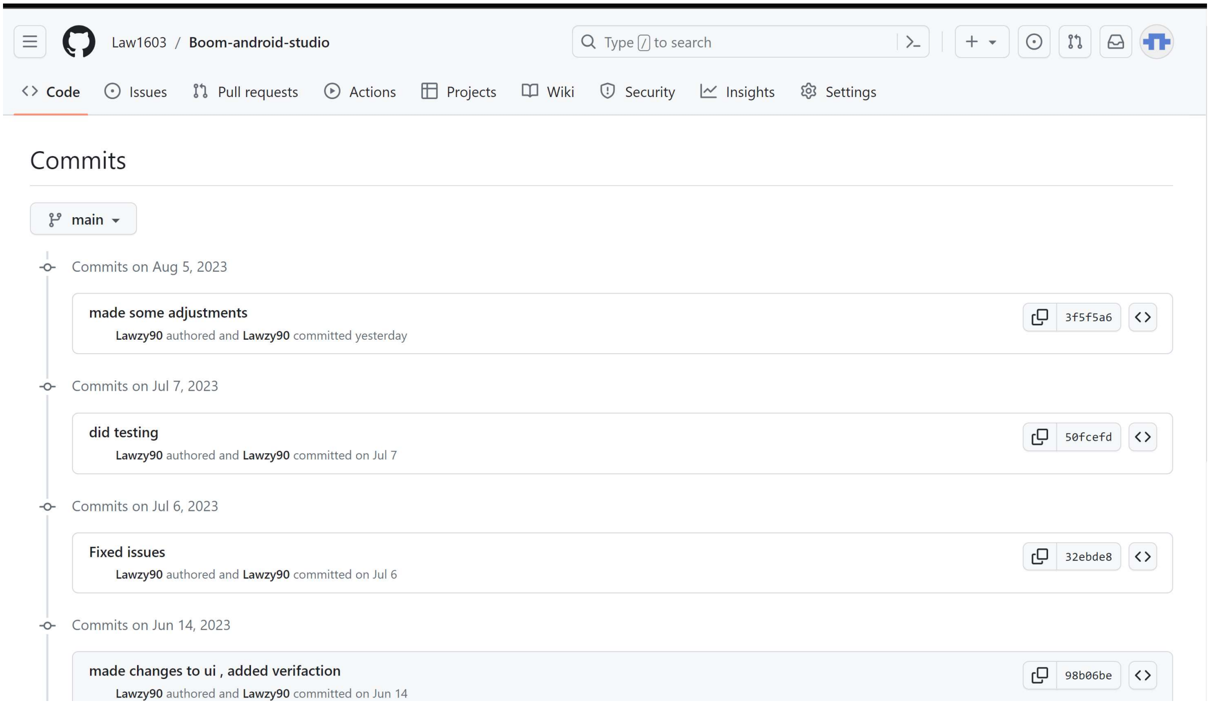
2nd Deliverable

Viva: Demonstration of the system designed and ability to show all important uses cases are covered

Creating a project using with accordance to ethical standards and proper documentation will help address the ability to negotiate with clients from across the world

The creation of this project can help any client around the world irrespective of their culture as it is a business need and business are not limited across the globe

Appendix – E



Law1603 / Boom-android-studio

Type / to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

main

Commits on Aug 5, 2023

made some adjustments
Lawzy90 authored and Lawzy90 committed yesterday
3f5f5a6

Commits on Jul 7, 2023

did testing
Lawzy90 authored and Lawzy90 committed on Jul 7
50fcefd

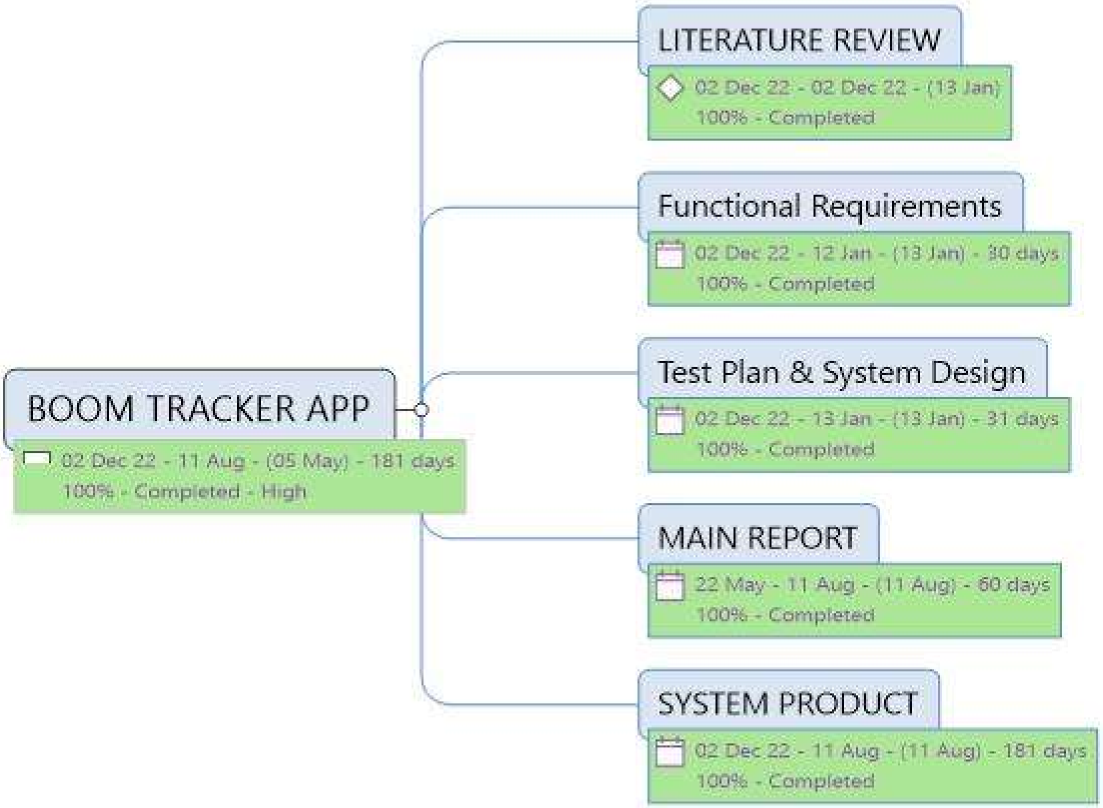
Commits on Jul 6, 2023

Fixed issues
Lawzy90 authored and Lawzy90 committed on Jul 6
32ebde8

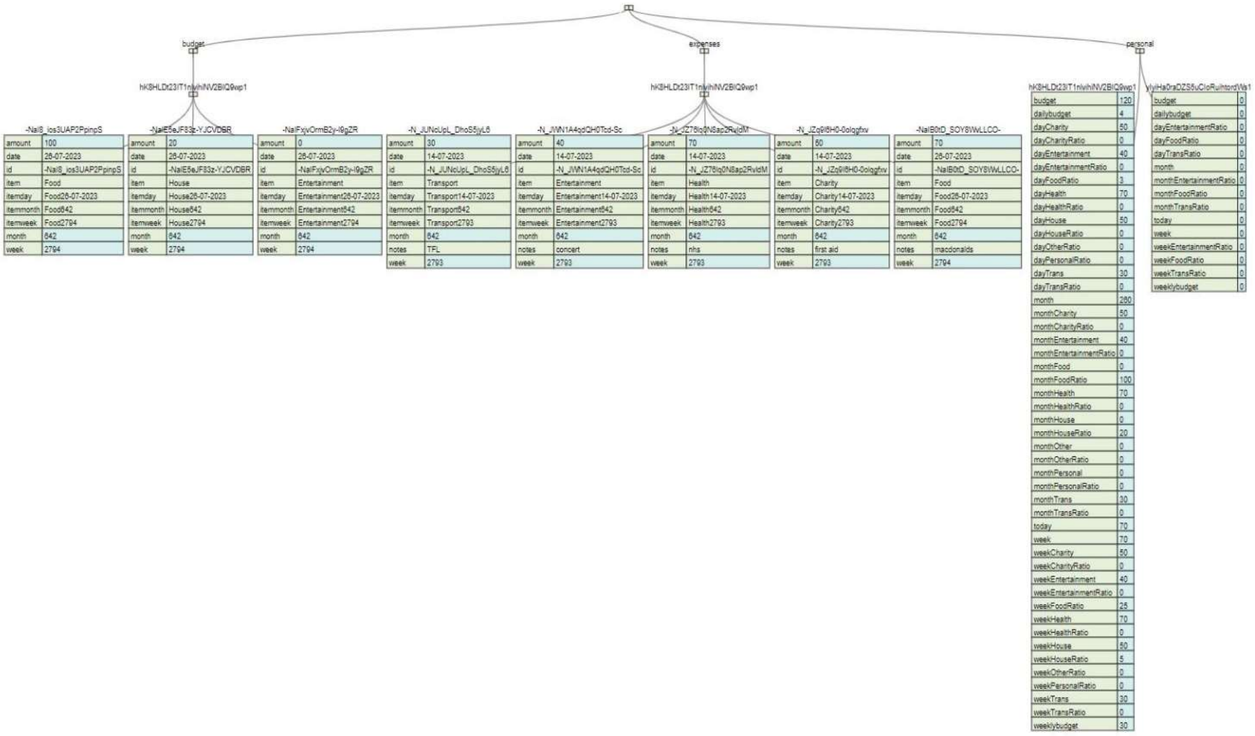
Commits on Jun 14, 2023

made changes to ui , added verifaction
Lawzy90 authored and Lawzy90 committed on Jun 14
98b06be

Appendix – D


BOOM TRACKER APP
02 Dec 22 - 11 Aug - (05 May) - 181 days
100% - Completed - High
LITERATURE REVIEW
02 Dec 22 - 02 Dec 22 - (13 Jan)
100% - Completed
Functional Requirements
02 Dec 22 - 12 Jan - (13 Jan) - 30 days
100% - Completed
Test Plan & System Design
02 Dec 22 - 13 Jan - (13 Jan) - 31 days
100% - Completed
MAIN REPORT
22 May - 11 Aug - (11 Aug) - 60 days
100% - Completed
SYSTEM PRODUCT
02 Dec 22 - 11 Aug - (11 Aug) - 181 days
100% - Completed

Appendix – E


budget
expenses
personal
hK8HLDt23IT1nlvihINV2BIQ9wp1
hK8HLDt23IT1nlvihINV2BIQ9wp1
hK8HLDt23IT1nlvihINV2BIQ9wp1
ylyiHa0raDZS5uCloRuihtordWs1

-NaI8_ios3UAP2PpinpS

amount	100
date	26-07-2023
id	-NaI8_ios3UAP2PpinpS
item	Food
itemday	Food26-07-2023
itemmonth	Food642
itemweek	Food2794
month	642
week	2794

-NaIE5eJF83z-YJCVDBR

amount	20
date	26-07-2023
id	-NaIE5eJF83z-YJCVDBR
item	House
itemday	House26-07-2023
itemmonth	House642
itemweek	House2794
month	642
week	2794

-NaIFxjvOrmB2y-I9gZR

amount	0
date	26-07-2023
id	-NaIFxjvOrmB2y-I9gZR
item	Entertainment
itemday	Entertainment26-07-2023
itemmonth	Entertainment642
itemweek	Entertainment2794
month	642
week	2794

-N_JUNcUpL_DhoS5jyL6

amount	30
date	14-07-2023
id	-N_JUNcUpL_DhoS5jyL6
item	Transport
itemday	Transport14-07-2023
itemmonth	Transport642
itemweek	Transport2793
month	642
notes	TFL
week	2793

-N_JWN1A4qdQH0Tcd-Sc

amount	40
date	14-07-2023
id	-N_JWN1A4qdQH0Tcd-Sc
item	Entertainment
itemday	Entertainment14-07-2023
itemmonth	Entertainment642
itemweek	Entertainment2793
month	642
notes	concert
week	2793

-N_JZ76lq0N8ap2RvldM

amount	70
date	14-07-2023
id	-N_JZ76lq0N8ap2RvldM
item	Health
itemday	Health14-07-2023
itemmonth	Health642
itemweek	Health2793
month	642
notes	nhs
week	2793

-N_JZq9l6H0-0olqgfxv

amount	60
date	14-07-2023
id	-N_JZq9l6H0-0olqgfxv
item	Charity
itemday	Charity14-07-2023
itemmonth	Charity642
itemweek	Charity2793
month	642
notes	first aid
week	2793

-NaIB0tD_SOY8WvLLCO-

amount	70
date	26-07-2023
id	-NaIB0tD_SOY8WvLLCO-
item	Food
itemday	Food26-07-2023
itemmonth	Food642
itemweek	Food2794
month	642
notes	macdonalds
week	2794

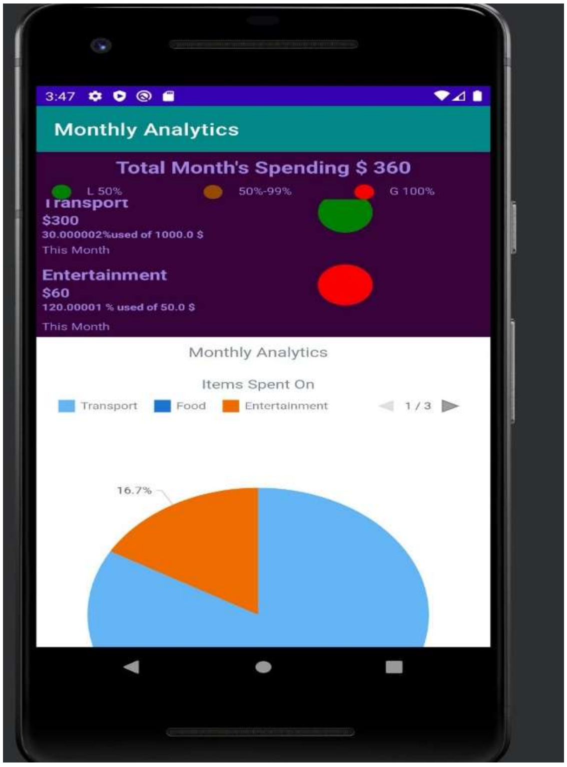
hK8HLDt23IT1nlvihINV2BIQ9wp1

budget	120
dailybudget	4
dayCharity	50
dayCharityRatio	0
dayEntertainment	40
dayEntertainmentRatio	0
dayFoodRatio	3
dayHealth	70
dayHealthRatio	0
dayHouse	50
dayHouseRatio	0
dayOtherRatio	0
dayPersonalRatio	0
dayTrans	30
dayTransRatio	0
month	280
monthCharity	50
monthCharityRatio	0
monthEntertainment	40
monthEntertainmentRatio	0
monthFood	0
monthFoodRatio	100
monthHealth	70
monthHealthRatio	0
monthHouse	0
monthHouseRatio	20
monthOther	0
monthOtherRatio	0
monthPersonal	0
monthPersonalRatio	0
monthTrans	30
monthTransRatio	0
today	70
week	70
weekCharity	50
weekCharityRatio	0
weekEntertainment	40
weekEntertainmentRatio	0
weekFoodRatio	25
weekHealth	70
weekHealthRatio	0
weekHouse	50
weekHouseRatio	5
weekOtherRatio	0
weekPersonalRatio	0
weekTrans	30
weekTransRatio	0
weeklybudget	30

ylyiHa0raDZS5uCloRuihtordWs1

budget	0
dailybudget	0
dayEntertainmentRatio	0
dayFoodRatio	0
dayTransRatio	0
month	0
monthEntertainmentRatio	0
monthFoodRatio	0
monthTransRatio	0
today	0
week	0
weekEntertainmentRatio	0
weekFoodRatio	0
weekTransRatio	0
weeklybudget	0

Appendix – F


3:47

Monthly Analytics

Total Month's Spending $ 360

● L 50% ● 50%-99% ● G 100%

Transport
$300
30.000002%used of 1000.0 $
This Month

Entertainment
$60
120.00001 % used of 50.0 $
This Month

Monthly Analytics

Items Spent On

References

- Chen, H. (2018). The comprehensive survey of mobile applications. Journal of Computer Networks, 120, 30-46.
- Fenton, S. (2014). Money management: Britons fail to keep track of their finances. Independent. Retrieved from https://www.independent.co.uk/
- Kumar, B., & Gupta, S. (2016). Mobile applications and their increasing utility. Journal of Business and Management, 18(2), 88-92.
- Oulasvirta, A., Rattenbury, T., Ma, L., & Raita, E. (2012). Habits make smartphone use more pervasive. Personal and Ubiquitous Computing, 16(1), 105-114.
- Perry, V. G., & Reilly, T. (2018). Financial literacy and the rising importance of financial education policy. Journal of Economic Perspectives, 32(3), 195-218.
- Statista. (2021). Number of smartphone users worldwide from 2016 to 2021. Retrieved from https://www.statista.com/
- Suurd, R. (2020). The importance of cash flow budgeting in small businesses. Journal of Small Business and Entrepreneurship, 37(4), 391-408.
- West, D. M., & Mace, M. (2010). Browsing as the killer app: Explaining the rapid success of Apple's iPhone. Telecommunications Policy, 34(5-6), 270-286.
- Gathergood, J. (2012). Self-control, financial literacy, and consumer over-indebtedness. Journal of Economic Psychology, 33(3), 590-602.
- Karim, M. R., & Ding, X. (2020). Google Firebase: A comprehensive study. International Journal of Computer Applications, 175(9), 26-30.
- Nielsen, J. (2012). Usability 101: Introduction to usability. Nielsen Norman Group. Retrieved from https://www.nngroup.com/
- Oracle. (2020). The Java® Language Specification. Oracle America, Inc. Retrieved from https://docs.oracle.com/
- Pew Research Centre. (2019). Mobile fact sheet. Retrieved from https://www.pewresearch.org/
- Soyinka, O. (2018). Android Studio 3.0 development essentials: Android 8 edition. Payload Media.
- Educatree. (n.d.). How to connect Android with Firebase database [Video]. YouTube. https://www.youtube.com/
- Nilesh Technology. (n.d.). Android Healthcare Project | Android beginner Project | Tutorial
- Karim, M. R., & Ding, X. (2020). Google Firebase: A comprehensive study. International Journal of Computer Applications, 175(9), 26-30.
- Ciman, M., Gaggi, O., & Avesani, P. (2017). Mobile apps for visually impaired people: The interplay between high customer expectations and personal experience. Universal Access in the Information Society, 16(3), 711-727.
- Lazard, A., & Mackert, M. (2015). E-health first impressions and visual evaluations: Key design principles for attention and appeal. Communication Design Quarterly Review, 3(4), 25-34.
- Huston, S. J. (2010). Measuring financial literacy. Journal of Consumer Affairs, 44(2), 296-316.

- Thaler, R. H., & Sunstein, C. R. (2008) 'Nudge: Improving decisions about health, wealth, and happiness', Yale University Press, New Haven, CT.

- *How to connect Android with Firebase database || Android and Firebase || Firebase + android #1* (2018) *YouTube*. Available at: https://www.youtube.com/watch?v=lnidtzL71ZA&list=PLO1TUgtj_6n149RoMQw6w9FzyHOnd94eG&index=51&ab_channel=Educatree (Accessed: 05 May 2023).
- *Android Healthcare Project | Android beginner project | tutorial | android | project | 2022-23* (2022) *YouTube*. Available at: https://www.youtube.com/watch?v=9CkpMm-n5iA&t=14422s&ab_channel=NileshTechnology (Accessed: 5 May 2023).
- Church, K., & de Oliveira, R. (2011). What is up with whatsapp? Comparing mobile instant messaging behaviors with traditional SMS. Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11. doi:10.1145/2037373.2037402
- Lallemand, C., Gronier, G., & Koenig, V. (2015). User experience: A concept without consensus? Exploring practitioners' perspectives through an international survey. Computers in Human Behavior, 43, 35-48. doi:10.1016/j.chb.2014.10.048
- Google. (2021). Recycler View | Android Developers. Retrieved from https://developer.android.com/guide/topics/ui/layout/recyclerview
- Nielsen, J. (2014). Dialog Boxes. Nielsen Norman Group. Retrieved from https://www.nngroup.com/articles/dialog-boxes/
- Hoober, S. (2014). Mobile User Experience: Limitations and Strengths. Nielsen Norman Group. Retrieved from https://www.nngroup.com/articles/mobile-ux/
- Firebase. (2021). Firebase Realtime Database. Retrieved from https://firebase.google.com/products/realtime-database
- OWASP. (2021). Input Validation. Retrieved from https://owasp.org/www-community/fundamentals/Input-Validation
- Smith, R. (2016) Getting Started with Firebase Authentication on Android, [online]. Available at: https://www.sitepoint.com/authentication-android-using-firebase/ [Accessed 14 June 2023].
- Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017) SP 800-63-3 Digital Identity Guidelines, National Institute of Standards and Technology. DOI: 10.6028/NIST.SP.800-63-3.
- Seitz, L. (2018) Password Policies. In: Menges, F., Fischer-Hübner, S. (eds) Human Factors in Information Security and Privacy. Human-Computer Interaction Series. Springer, Cham. https://doi.org/10.1007/978-3-319-92919-4_4.
- Stubblebine, T. (2011) Regular Expressions Pocket Reference: Regular Expressions for Perl, Ruby, PHP, Python, C, Java, and .NET. 2nd ed. O'Reilly Media.
- Google (n.d.). Advanced Test Setup. Available at: https://developer.android.com/studio/test/advanced-test-setup (Accessed: 7 July 2023).
- Majumdar, A., Kar, A., & Ghosh, S. K. (2018). Software Development Tools and Technologies for Mobile Apps. In Mobile Application Development, Usability, and Security (pp. 95-112). IGI Global. ↩
- Oracle. (2021). Introduction to Java programming language. Oracle Docs. https://docs.oracle.com/javase/tutorial/java/nutsandbolts/ ↩

- Hussain, A., Mkpojiogu, E. O., & Kamal, F. M. (2012). Usability and user experience: a survey among software professionals in Malaysia. Journal of computer science and software engineering, 1, 1-14. ↩