

School of Science and Technology

**ITX3999
IT Project**

Autumn/Winter term

2018/2019

Date: 26-April-2019

Supervisor: Ms. Chinnu George

Student Name: Syed Daniyal Ahsan

Student ID Number: M00608913

Campus: Dubai

Title: Budget Living

School of Science and Technology

Student Name: **Syed Daniyal Ahsan**

Student Id No: **M00608913**

Module number: **ITX3999**

I hereby confirm that the work presented here in this report and in all other associated material is wholly my own work. I confirm that the report has been submitted to TURNITIN and that the TURNITIN results are on CD attached to this report. I agree to assessment for plagiarism.

Signature:

Date: **26-April-2019**



Abstract

The importance of budgeting is known to almost everyone, each month people think of saving money but do not achieve that due to over-spending which happens because there is no track of how much money is being spent where, so an application like '**Budget Living**' is a very helpful companion that not only tracks their expenses, but also helps the people set a limit on each category of expense, and allow them to see the savings and statistics from the past three month as well.

The project being developed is an android application and everything related to the project is present in the report. The report is divided into **9** chapters:

The **introduction** section talks about the need and scope of this application, where the **background** section talks about the existing applications related to budgeting. The **requirements & analysis** section looks at the need of the users and displays list of functional and non-functional requirements, which then allow the analysis to be carried out for the application in the analysis section. The **design** phase consists of the UML diagrams and wireframes that show the logic and layout of the application. **Implementation** stage displays the development phase. Along with that the **testing** section is linked with implementation stage as all the development code is being tested for making sure all logic is in proper shape. The **demonstration** section consists of screenshot of every screen to show how the application works and, in the end the **conclusion** stage talking about things learned and what different ways could be used to develop this application.



Table of Contents

CHAPTER NO.	TITLE	P AGE NO.
Chapter 1	INTRODUCTION.....	08
	1.1 Introduction to the Project.....	08
	1.2 Problem Addressed	08
	1.3 Project Scope and Purpose.....	08
Chapter 2	LITERATURE SURVEY.....	09
Chapter 3	SOFTWARE SPECIFICATIONS	12
	3.1 Software Development.....	12
	3.2 Requirements.....	13
	3.2.1 Functional Requirements.....	13
	3.2.2 Non-Functional Requirements.....	15
	3.2.3 Software and Hardware Requirements.....	16
	3.2.4 User Interface Requirements.....	17
	3.3 Test Plan.....	17
	3.4 Analysis.....	18
Chapter 4	DESIGN.....	20
	4.1 Design Phase.....	20
	4.2 ER Diagrams	20
	4.3 Class Diagrams.....	21
	4.4 Sequence Diagrams.....	22
	4.5 Wireframes.....	24
	4.6 Screens.....	07



Chapter 5	IMPLEMENTATION.....	34
5.1	Source Code.....	34
Chapter 6	TESTING.....	39
6.1	Test Plans.....	39
6.2	Test Cases	39
6.3	Important Checklist.....	43
6.4	Error and Success Functionality Test.....	44
Chapter 7	DEMONSTRATION.....	48
7.1	Activity Flow.....	48
Chapter 8	CONCLUSION.....	50
Chapter 9	APPENDIX.....	51
9.1	Quick Start Guide.....	51
9.2	Test Cases Results.....	52
9.3	Project Meeting Logs.....	54
9.4	References.....	57



Table of Figures

Figure #	Figure	Page Number
3.1.1	Software Development Life Cycle	12
3.2.1.1	Functional Requirements	13
3.2.2.1	Non-Functional Requirements	15
4.2.1	ER Diagram	20
4.3.1	Class Diagram	21
4.4.1	Sequence Diagram	22
4.4.2	Sequence Diagram	23
4.5.1	Wireframes (Screens 1-3)	24
4.5.2	Wireframes (Screens 4-6)	25
4.5.3	Wireframes (Screens 7-9)	26
4.6.1	Splash Screen	27
4.6.2	User Login/Registration Screen	27
4.6.3	Home Screen	28
4.6.4	Income Activity Screen	28
4.6.5	Expense Screen	29
4.6.6	Scheduling Notification	29
4.6.7	Current Month Expenses Screen	30
4.6.8	Current Month Detailed Expenses	30
4.6.9	Future Task Reminder Screen	31
4.6.10	Setting Limit On Expenses Screen	31
4.6.11	Selection Of Monthly Report Screen	32
4.6.12	Monthly Statistics Screen	32



4.6.13	Monthly Statistics Graphically Screen	33
4.6.14	Notification Received	33
5.1.1	Source Files	34
5.1.2	User Login/Registration Code	35
5.1.3	Limits Crossed Code	35
5.1.4	Transfer Data from One Activity to other	36
5.1.5	Enter Expense Code	36
5.1.6	Enter Income Code	37
5.1.7	Repeating Alarm Code	37
5.1.8	Get Expense from Database Code	38
5.1.9	Calendar Activity Code	38
6.4.1	Different Passwords Entered	44
6.4.2	Username/Password Missing	44
6.4.3	Existing User	44
6.4.4	Registration Success	44
6.4.5	Fields Missing	45
6.4.6	Wrong Details	45
6.4.7	Login Successful	45
6.4.8	Task Added	45
6.4.9	Empty Income	46
6.4.10	Income Added	46
6.4.11	Expense Not Accepted	46
6.4.12	Expense Added	46
6.4.13	Improper Income	47
6.4.14	Limit Not Accepted	47
6.4.15	Limits Accepted	47



6.4.16	Limit Crossed On Expense Category	47
9.1.1	How To Run The Application	51
9.1.2	How to Build Applications' APK	52
9.2.1	Test for multi-tasking	52
9.2.2	Test for app permissions	53
9.2.3	Test for getting calls in app use	53



Chapter 1 – Introduction

1.1 - Introduction to Project

Recently, studies show that noting your spending helps create financial awareness which makes it easier to identify where the money is being spent, along with that it helps to stick to a budget. The main idea that brought this development into action is the point that is mentioned above. Since keeping track of spending on a spreadsheet or sheet of paper can be challenging to manage, therefore this project will be an **android application** which will be more convenient for users as they do not have to do much rather than just entering the amount of each expense made during a day and the application will keep a track of these expenses, calculate the total expense being made during the month and then compare with the limit being set by the user to make sure the limit is not being crossed that is being set for each month. The user can also use this application to view the statistics of expenses as a graphical representation. Furthermore, the application can also help users set notifications and reminders for tasks that might arise in the future.

1.2 - Problem Addressed

There are many ways to keep track of expenses being made, if it is done using a sheet of paper it could result in wastage of many papers, while if managing is done by hiring someone to keep a track of spending done is wastage of money, and if this tracking is done using a spreadsheet then there is a loss of time. Since time, money and paper are valuable the best way to track your expense is using a mobile application that is free, saves time as all you need to do is input your expenses and helps save papers as there is no need of paper while using an application.

1.3 - Project Scope and Purpose

The main purpose and scope are to develop an android application that will allow users will have a platform where they can track their expenses. The users can log-in using their details and then set a budget limit for different categories of spending. This will allow them to be able to see how their expenses are divided based on different categories and where all they are spending their money. The application will give out a notification every day so that the users do not forget to add their expenses for the day. In the end, it will also help users to set reminders for future expenses which would be coming up.



Chapter 2 – Literature Survey

Why are budget applications not so widely used?

The need for tracking how much money is spent at a point in current time is a must, with expenses increasing so often, having a budget to limit spending becomes vital. Before understanding a budget tracking application, it is important to know what having a budget means. According to (Mymoneycoach.ca, 2018), a budget is keeping a track on the amount of money being spent so that the amount being spent is within limits and balancing with the amount we receive as our income. Making a budget helps create a line on how much money can be spent without following in debt. People do not track their expenses, and that leads them to go overboard and sometimes lead them to have debts. Knowing what a budget is, a budget app is getting everything related to budget onto an application where the user can have their advisor. This medium will allow noting how much is being spent and then allowing proper tracking of expenses and knowing when the user is spending a lot on something.

Researches have been made to find the strengths and weaknesses of these applications. (Bell, 2018) have been on the supporting side while (Theinternationalfinance.com, 2018) has been giving the weaknesses of using these applications. Strengths for these applications are as (Bell, 2018) says, these applications help you set a goal limit to limit the amount of spending a user can make. They also allow the user to save up funds for some other expense that might be coming in future. One of the key points of these applications is that they help the user from going above the limit being set. Users can also see the total amount of money that is available to them. (Theinternationalfinance.com, 2018) however, has the weaknesses of these applications, they talk about how an emergency expense will not be forecasted by the application. Time is valued as money, and setting limits on each category, entering data for expenses can be time-consuming, so a lot of time is being wasted. Sometimes, because of these applications, a user must overcompensate on his spending to stay under the limit. Using this application to set a limit for a person who has a family is sometimes going to cause a lot of chaos as getting other members of the family to be disciplined and follow the budget is almost impossible. Considering both the strength and weakness, the strengths, however, overpower the weaknesses because of the importance of tracking your budget. According to (The Balance, 2018), applications have become a need because they help a user know where and how they are spending their money. They help create awareness to save money and allows a user to stick to a budget with constant notifications.

Researches have been done on existing systems that exist regarding budget and financial applications, one of the research done by (Thanapal et al., 2018) talks about



the existing systems which have been developed. (Thanapal et al., 2018) tells that budget applications have made it easier for people to manage their incomes and expenses using phones rather than a sheet of paper, they also allow setting categories for expenses but the reason the application fail is that they are not able to give any reminders regarding expense being missed for a day and do not have any options to set limit on spending. Continuing the discussion about importance of budgeting applications, (Phil and Vinodhini, 2016) talks about how people complain that managing budgets through registers, spreadsheets, and sticky notes is almost a time-consuming way of keeping track of expenses made, therefore an application to monitor all the expenses is a must need.

However, there currently are some limitations or issues which the budgeting applications have just as off at this current point in time, in addition to that object over there currently exists the main reason for whatever nonsensical reason they do not gain a lot of popularity worldwide. (Long, 2018) shows the biggest issues currently found in the applications. Starting with few of the most famous reasons, the budget applications can be disregarded very easily. There is no doubt that these applications do their best to help a user stay on track by giving out notifications, but these notifications can just be ignored, and that makes these applications very easy to be ignored. One of the other issues is that these applications cannot control a person, even though they show a person what their limit on spending is. Putting expenses in the correct category is very time consuming, and the user can just put any item spending on some other category causing a problem in the total spending of that category. The advantage of this application is that users focus and pay more attention to what they are doing, so possible reason for these applications failing is that nothing new regarding design or updates once the application gets released, this would result in users getting bored of the application.

In this study, the gaps in budget applications are also discussed from the points that are given by (Clearbridge Mobile, 2018) and (Appy Pie, 2018). Both the authors have identified three points that are common in between them; these include forgetting the backend and service integration of the application, putting less attention on the functionality, and just getting the application ready to be used as soon as ready. Cross development, not present, cross development is that the application should be able to run in all mobile operating systems rather just one. Some applications do not be successful because the developers only target one type of operating systems. A gap identified is that there is no update. There are almost little or no updates on budget applications. (Clearbridge Mobile, 2018) talks about one more gap, this is the mistake done by developers/heads who misinterpret the difference between applications and websites. (Appy Pie, 2018) also points out another gap which is that full concentration and time is given to the UI of the application and has been kept as a priority over the functionality of the application. These identified gaps emphasize why these budget tracking applications are not widely used around the world.



Considering the number of smartphone users in the USA, considering the number of users in the USA because they are a country that has a huge number of smartphone users according to (Statista, 2018).

Therefore moving in detail of budget application users in US, as stated by (Barba, 2018) 63% of people in use a budget application and considering the need of this application, this number was expected because people need to use these apps on a daily basis but also looking at the current issues, gaps that are found in these budgeting applications, It explains why these applications do not succeed on a high scale rate everywhere.

After total amount of smartphone users, the next big question is why need an application rather than a website. According to (Venkata et al., 2014), the mobile applications downloaded in a year has been on the rise starting from 2010, increasing from 10.9 million to 76.9 million in 2014, and keeps on increasing every year. However, there are reasons why many people have issues with using applications, a survey done with 500 US citizens showed that 96% people got frustrated while using applications because of unnecessary ads, application lagging, slow responses, and high battery usage. One of the most important reasons for people using application is '**performance**,' 98% of the people chose performance as the main reason to use an application. According to (Harrison et al., 2013) the other few important reasons for applications failing is '**connectivity**' and '**small screen size**' and therefore an application that works without internet and works with all screen sizes will always attract users.

An application will exist as there are many people who want to use mobile applications rather than websites as stated by (Jay et al., 2014). Considering the limitations, gaps, total smartphone, mobile users and the importance of a budget to increase the number of people using these applications, an application (**Budget Living**) will be developed, the priority is to have maximum performance for the application which will attract more users, the next priority is to address the current issues and look more in detail about the current issues, limitations, and gaps. The application will be like an own personal financial advisor, who gives reminders and constant updates on the spending done by the user along with new features being updated and brought frequently so that the user does not get bored using the application. The current gaps need to be looked with much detail; more work will be going in functionality rather than the UI/UX part of the app. The development work will be planned properly to make sure work is done on every single aspect of the application. Some information regarding the next update, new features will be shown to make the users feel that they can expect more to come out from this application.



Chapter 3 – Software Specifications

3.1 – Software Development

According to (Software Testing Material, 2019), the Software Development Life Cycle (SDLC) is the life cycle that is followed every time a software project is to be created. The cycle allows the final product to be efficient, and to meet the high-quality expectations that the user has. (Existek.com, 2019) tells that the SDLC has many models which include; Agile, Waterfall, V-Shape, Iterative, and Spiral. The model we are considering in our case is **Waterfall**. In this model, there are six phases, and each phase needs to be completed to move forward with the project. This model is the most used approach in SDLC. The **Waterfall** approach in SDLC has six phases, all these six phases are equally important. The six phases of the Software Development Life Cycle are as shown in the image below:

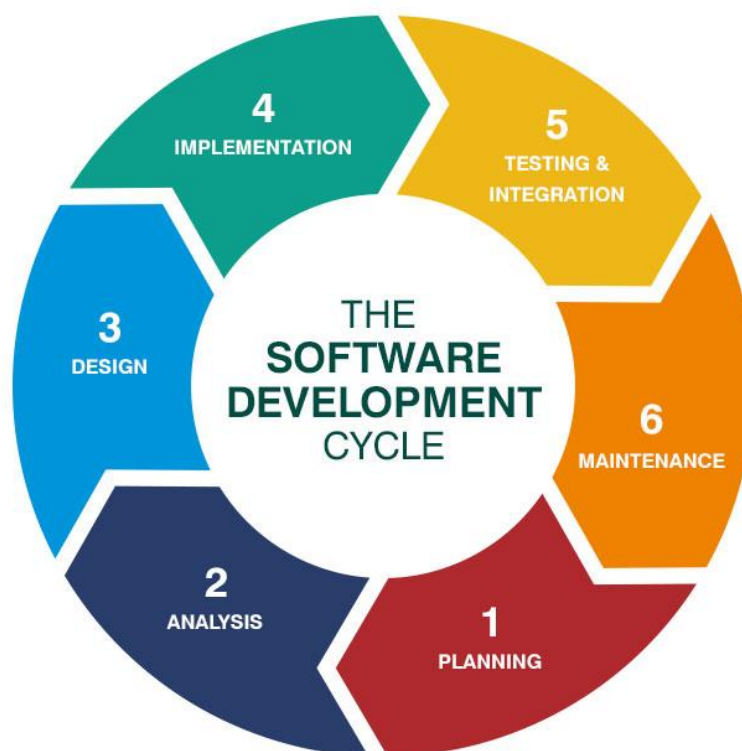


Figure 3.1.1: The Software Development Life Cycle (Software Testing Material, 2019).

The six phases mentioned in the figure have all been used to complete the '**Budget Living**' application. The project started with planning the requirements, then analyzing using the research and requirements, following that with design and implementation. Once the application was ready, it was put through testing and once



passed all the tests, it was ready to be used by users. All the phases being mentioned here are being described in the pages below.

3.2 – Requirements

One of the most important phases in software development is requirements, requirements help understand the expectation that the user have. These users are those who will use the application. Requirements help value the features that are important, and a must to have. They also help neglect features that are not of any value to the users. There are two types of requirements, these include **functional** and **non-functional** requirements (SearchSoftwareQuality, 2019).

3.2.1 – Functional Requirements

According to (Techopedia.com, 2019), Functional Requirements are based on the functionality and working of the application/software. These requirements are normally the inputs and outputs of the system. These types of requirements are important because they note the functionalities that the system should achieve during execution. The following image are the functional requirements shown in brief for this application, while the table shows a descriptive description about these requirements.

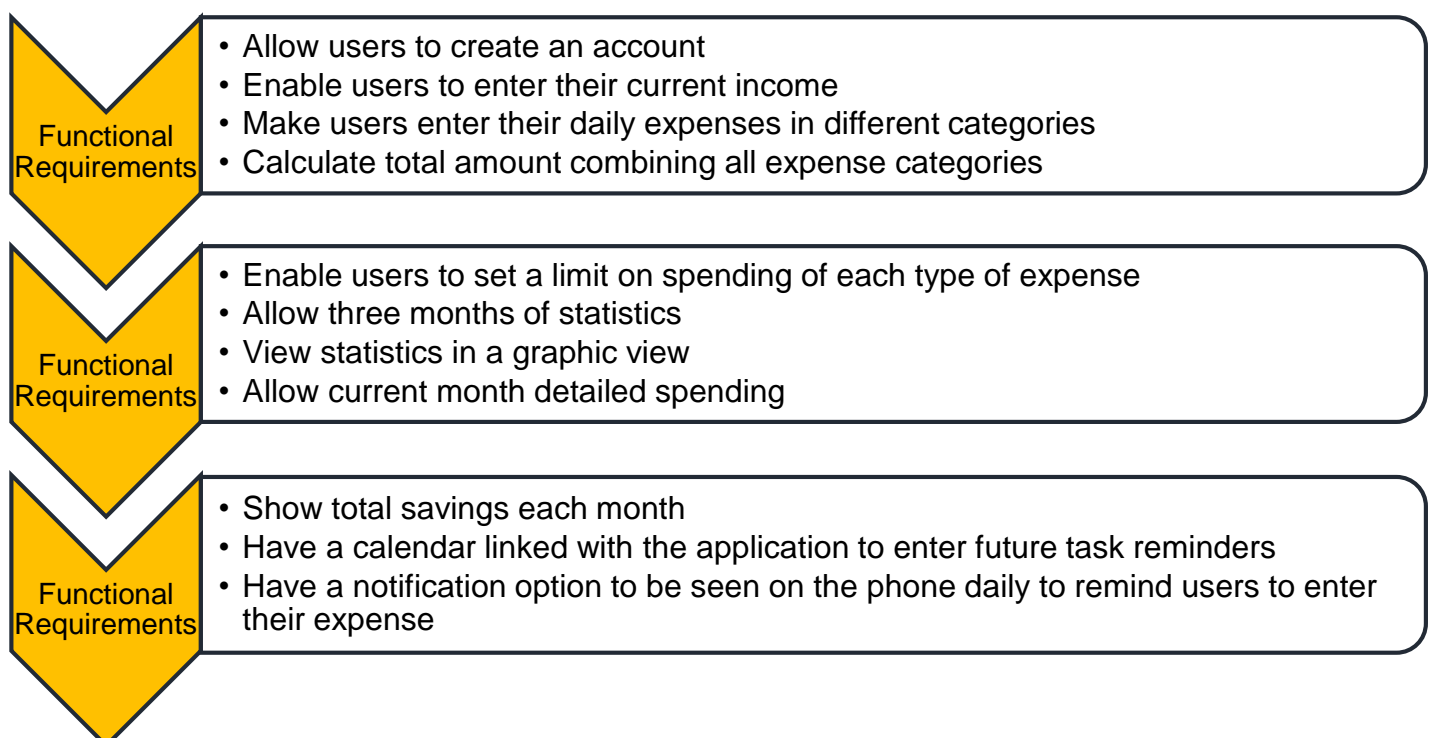


Figure 3.2.1.1 Functional Requirements



Functional Requirement	Importance
Allow users to create an account	<ul style="list-style-type: none">– Allowing users to create an account so that privacy is maintained– Many people can use the application on the same phone
Enable users to enter their current income	<ul style="list-style-type: none">– Will help the application to get a reference point from which all the expenses made can be subtracted
Make users enter their daily expenses in different categories	<ul style="list-style-type: none">– This will allow the users to enter their everyday expense to keep track of their spending
Calculate total amount combining all expense categories	<ul style="list-style-type: none">– Calculating total amount will help know how much money was being spent daily
Enable users to set a limit on spending of each type of expense	<ul style="list-style-type: none">– Allowing users to set a limit will allow the budget to be maintained as they will be notified when they cross the maximum amount set for spending in a category
Allow three months of statistics	<ul style="list-style-type: none">– This will allow users to see what their spending in the past three months is and how much money has been they have saved each month
View statistics in a graphic view	<ul style="list-style-type: none">– This will allow users to view their spending graphically rather than just numbers
Allow current month detailed spending	<ul style="list-style-type: none">– Give users detailed information about current month on how much has been spent in total
Show total savings each month	<ul style="list-style-type: none">– Help users know the amount being saved
Have a calendar linked with the application to enter future task reminders	<ul style="list-style-type: none">– To allow the user to be reminded of tasks that will be coming up in future
Have a notification option to be seen on the phone daily to remind users to enter their expense	<ul style="list-style-type: none">– This will notify the user that they need to input their expenses, it will help them remind about the application if they forget to enter expenses



3.2.2 – Non-Functional Requirements

According to (Request.com, 2019), Non-Functional requirements are very important for an application because these types of requirement determine how the system should operate. These requirements are constraints that are put up on the system to ensure that the application works in the most efficient way possible. Every requirement that does not come under the list of functional requirements, fall under the non-functional requirement. The following image are the non-functional requirements shown in brief for this application, while the table shows a descriptive description about these requirements.

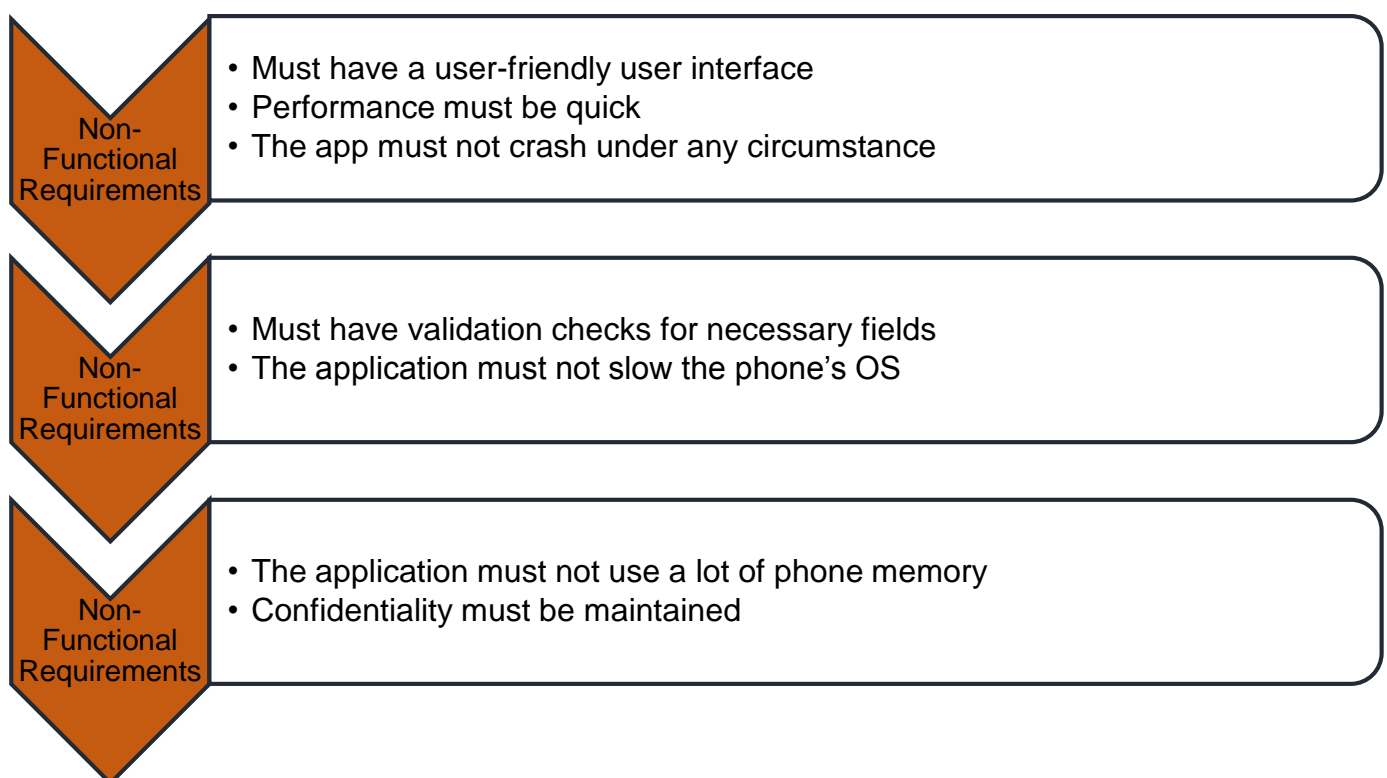


Figure 3.2.2.1: Non-Functional Requirements



Non-Functional Requirement	Importance
Must have a user-friendly user interface	<ul style="list-style-type: none">– The application should be friendly so that the users enjoy using it and find it easy to use
Performance must be quick	<ul style="list-style-type: none">– Quicker performance will indicate more efficiency
The app must not crash under any circumstance	<ul style="list-style-type: none">– Application crashing would indicate poor programming and missing of logic which would mean the application is not well coded
Must have validation checks for necessary fields	<ul style="list-style-type: none">– Validation to ensure all data being added is in correct format.– Validation to ensure that all the necessary data are present
The application must not slow the phone's OS	<ul style="list-style-type: none">– Application should allow other applications being run on the side to ensure that user can multi-task
The application must not use a lot of phone memory	<ul style="list-style-type: none">– Total amount of phone memory used should be less because the more the memory the phone takes, the less interest users will have in downloading the application
Confidentiality must be maintained	<ul style="list-style-type: none">– Privacy breach is what everyone fears, and if the data is not being secured then it would result in users being less interested in using the application and a more secure system would allow users to feel their data is in correct hands.

3.2.3 – Hardware and Software Requirements

Software Requirements: Android Studio, Android Emulator, SQLite

Hardware Requirements: Android Phone, Laptop to Phone USB Connector



3.2.4 – User Interface Requirements

User Requirements help make the application usable for users, the user requirements for Budget Living is mentioned below:

-Application must be user friendly with all test shown properly

-Application should notify the user on the task that has been performed

-The application should not lag and slow the phone's RAM

-Response time should be fast

-Application's icons must be those that match standards

-The layout for the application should remain a constant

3.3 – Test Plan

Every application needs a plan for testing so that once the implementation method is complete; these plans can be carried out to make sure the working of the system is in correct form. Some of the basic tests are shown in the table below, the detailed plan for testing is shown in section in **6.2**

Test Plan	Importance
Installation is done properly	If the installation is done properly, it might cause some of the functionality to not function, and even cause the application to crash suddenly
All the fields where user input is required has validation checks on it	Validation on fields is a must for every application, because without validation faulty values could be entered into the application and that would result in a reduced efficiency of work or could give wrong outputs



Application works while multi-tasking	Application should work even when there are multiple tasks working, that would indicate that the application is not using much of the phone's RAM
Database is kept secure	Security is key to success of an application, the more secure the data is, the more users will tend to use the application and trust it with their data
Friendly user interface	UI should be useable, attractive, and very simple to understand for every user so that they can benefit from the application
Response time check	Response time needs to be quick to show that the application performs all processes quickly
All content is visible	Making sure no fields are getting cut or not appearing properly which would make the application lose its aesthetic appearance

3.4 – Analysis

Going through the requirements, the main features needed on '**Budget Living**' is the ability to allow users to enter their income and expenses. One of the important features that would be a must is allowing users to set a limit on the amount of expenses for each month. After referring some research papers, it is important that most of the time is spent focusing on the functionality of the application rather than the UI/UX. Budget applications can be ignored or forgotten, therefore notification on the phone is a functionality that will be required to remind the users. Some expenses must be remembered from an earlier date; therefore, a reminder functionality will be an additional point that can be added to this application.

The target audience will be from students who have a part-time job to adults working on a full-time basis; this application will help them keep track of their savings and expenses. They will get to know how much they are spending along with giving detailed data about where and when they have been spending more. This application will have the above-mentioned people as target audience because no one more than them will be having to worry about their money, because students are earning need to spend wisely to carry out their monthly expenses while saving money for rent,



fees, and other spending's. While, adults have a family to take care of and thus they need to be more careful while carrying out their expenses.

According to (Statista, 2018) and (Simon, 2018), the market share of Android users [Google] is way more than iOS users [Apple] and other software users. The margin of users is huge. Therefore, choosing android as a platform would be a safer choice to familiarize the public with the application and once there are enough users for Android, develop the iOS version so that the gaps currently that many applications do not have cross-development gets reduced. This would be a good strategy as well, as it would allow time for the development of the application on another operating system.

The '**Budgeting Living**' application will be a free application as a new application needs to make users allow them to get used to the application without the need to pay anything. Having the application unpaid will allow more users to download the application and will be a good strategy from the marketing point of view. The application can have advertisements within the application in the start as a monetization strategy.



Chapter 4 – Design

4.1 – Design Phase

One of the most important phases in the software development cycle is the ‘**design**’ phase. After going through the requirements and analysis for the system, the logic will be converted to design showing data flow, users, and the limitations of the system. They allow us to see a detailed working of the system in a visual state rather than just text. There are many ways to execute designing; the **Waterfall** model is being considered for the execution, and the methods to carry out the method is UML (Unified Modeling Language) Models (Oer.nios.ac.in, 2019). We are considering **ERD, Class, and Sequence Diagrams** for the design phase of our system and along with the UML Diagrams, the wireframes for the screens are also shown in this section.

4.2 – ERD (Entity Relation Diagram)

Each entity in an ER Diagram is a component of data that has their own properties. ER Diagram helps point out the relationship between each entity. The ER Diagram 1 below is for the ‘**Budget Living**’ system

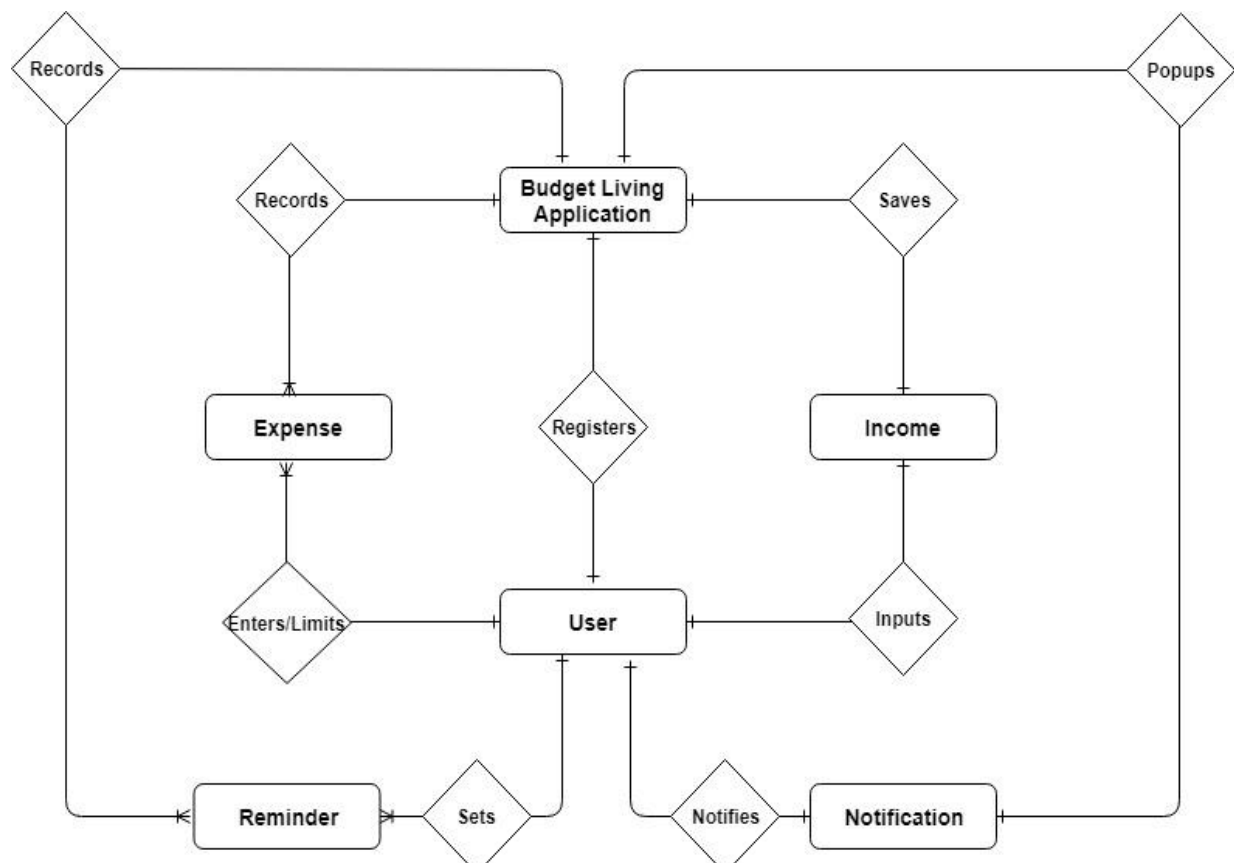


Figure 4.2.1: Entity Relation Diagram



The ERD in (Figure 4.2.1) shows the there is a user who registers in the application ‘**Budget Living**’. The application will have options for income, expense, limits, notifications, and reminder which the user can access once they have registered to use the application.

4.3 – Class Diagrams

Class Diagrams are used to model the structure of a system. They help pinpoint the attributes, operations, and the relationship between objects. These diagrams are one of the most useful diagrams in UML modeling (Lucidchart, 2019). The Class Diagram below is for the ‘**Budget Living**’ system

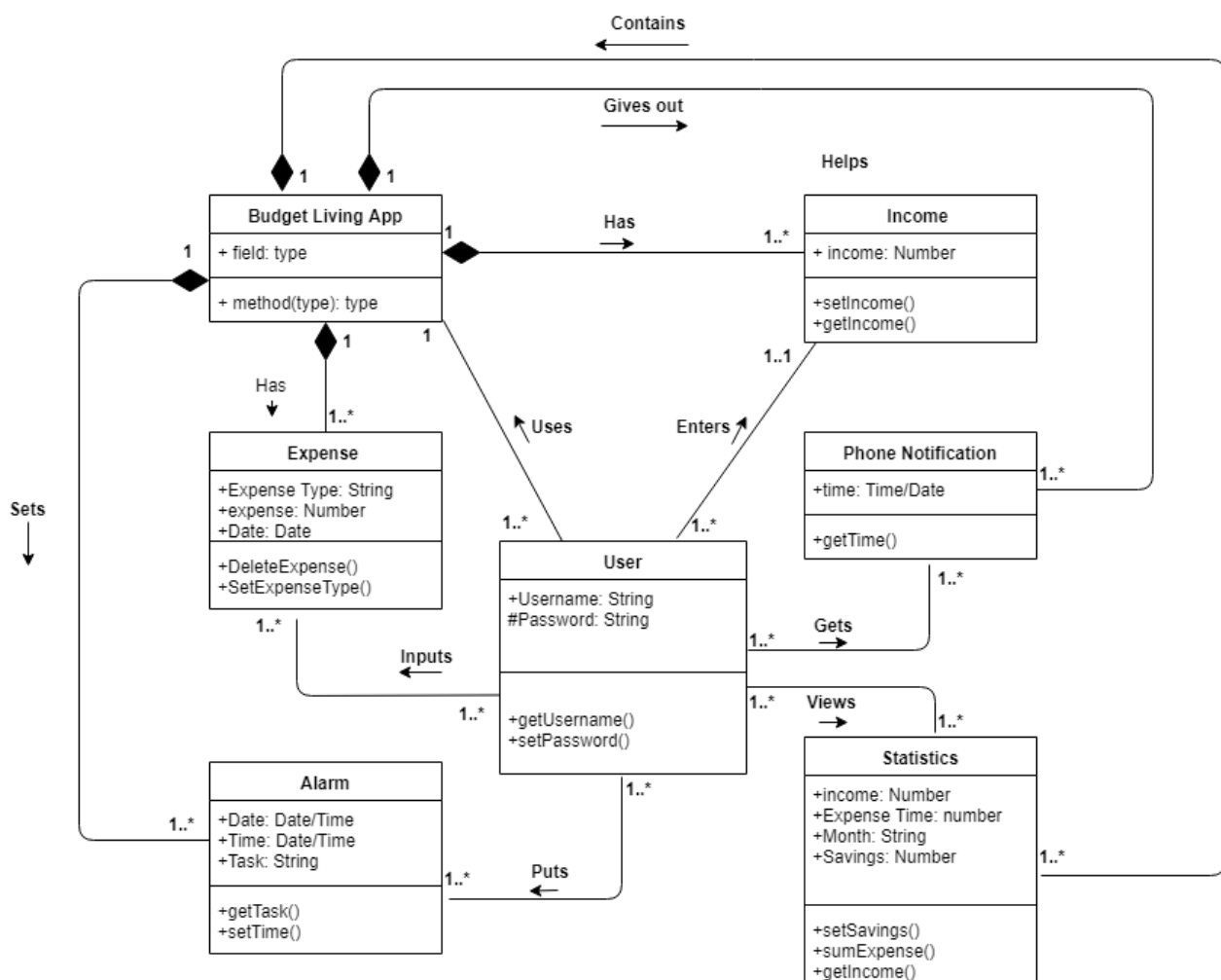


Figure 4.3.1: Class Diagram



The **class diagram** in (Figure 4.3.1) uses the logic that has been used in ERD, the classes have been displayed, and shows the interaction that will be happening in between all the classes.

4.4 – Sequence Diagram

According to (Agilemodeling.com, 2019), Sequence Diagram are the most famous UML model that is used for the design phase, it not only shows what all interactions will happen in the system, but also shows the order of interaction as well as how the system will behave in between these interactions. The Sequence Diagrams shown below are for the '**Budget Living**' system.

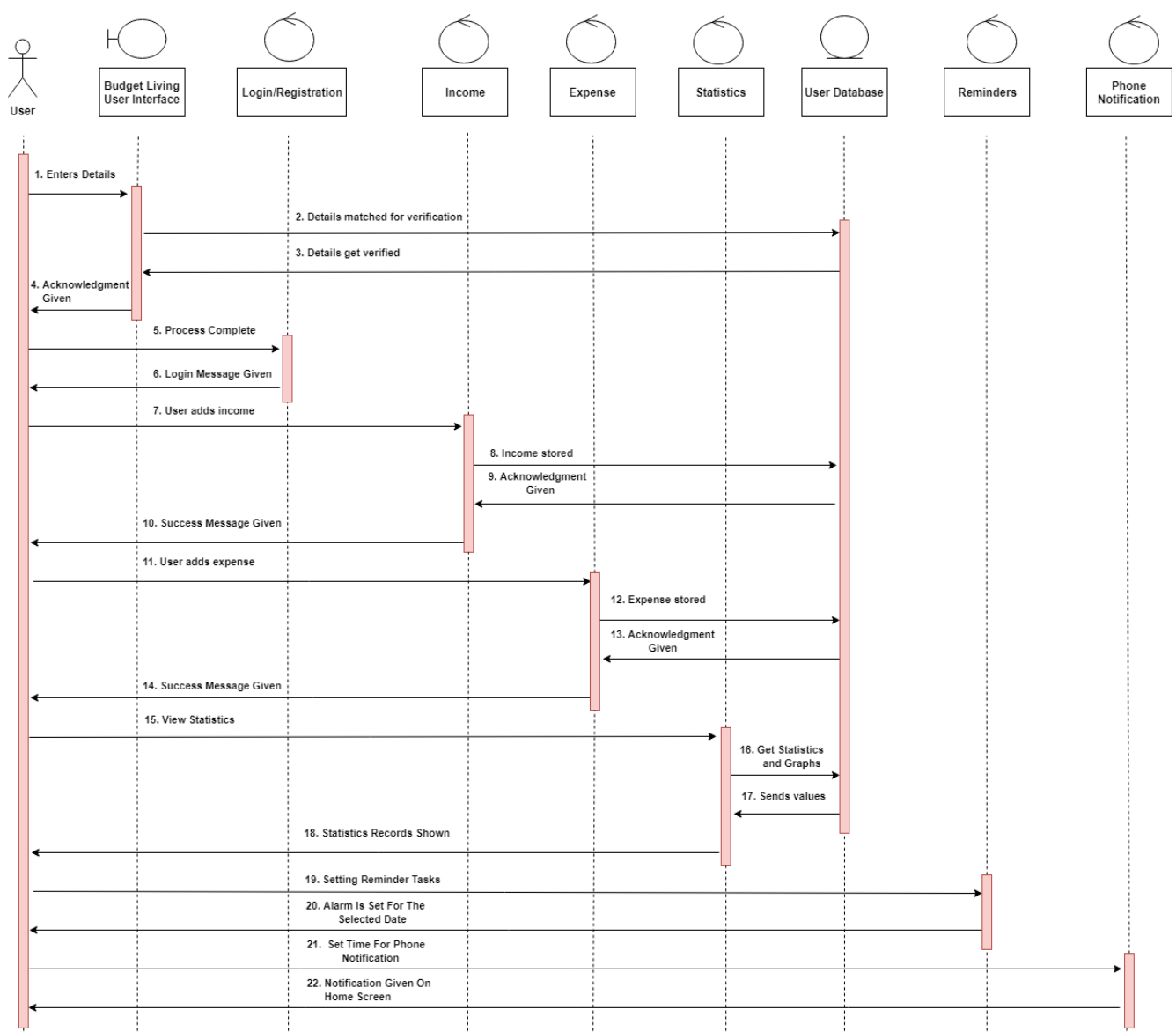


Figure 4.4.1: Sequence Diagram

The sequence diagram shown in (Figure 4.4.1) shows exactly all the general interactions the user would be having with the



application. This involves the user interaction with the application's interface, the processes that are present, and the application's interaction with the database.

The **sequence diagram** seen in (Figure 4.4.2) is for two interactions, one is when a new user uses the application and wants to register an account where they interact with the interface and the process being involved, and when the interaction between the application and process with the database. The other interaction is the login process where the user is an already existing user and the process involved is same.

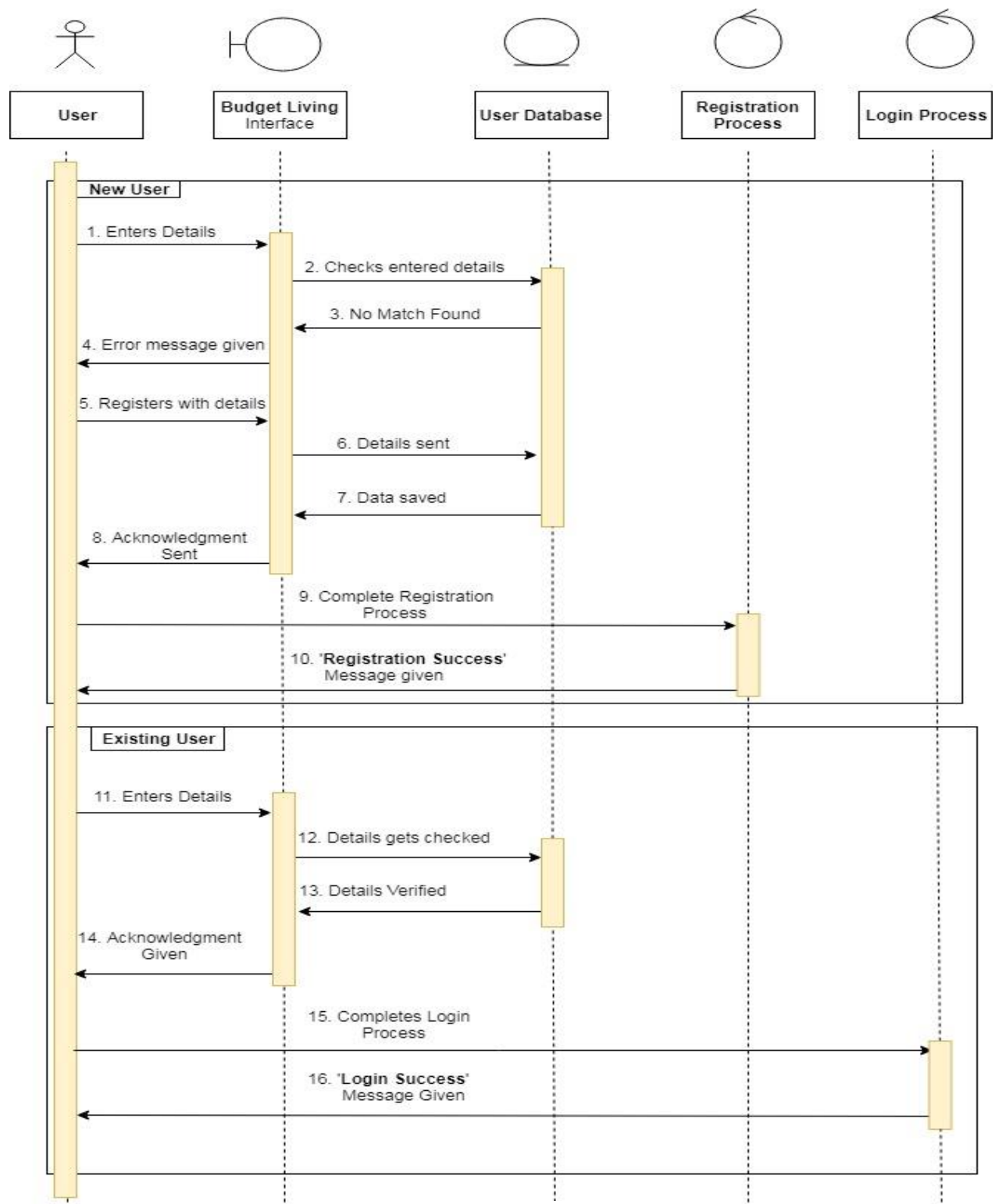


Figure 4.4.2: Sequence Diagram



4.5 – Wireframes

Wireframes are used to set a structure for the system, these can be sketches which help understand the layout that will be used for the system. They help give a basic understanding on where and how each component should be placed before the coding takes place (Experience UX, 2019). Below are the wireframes for each of the screens along with the description of these screens

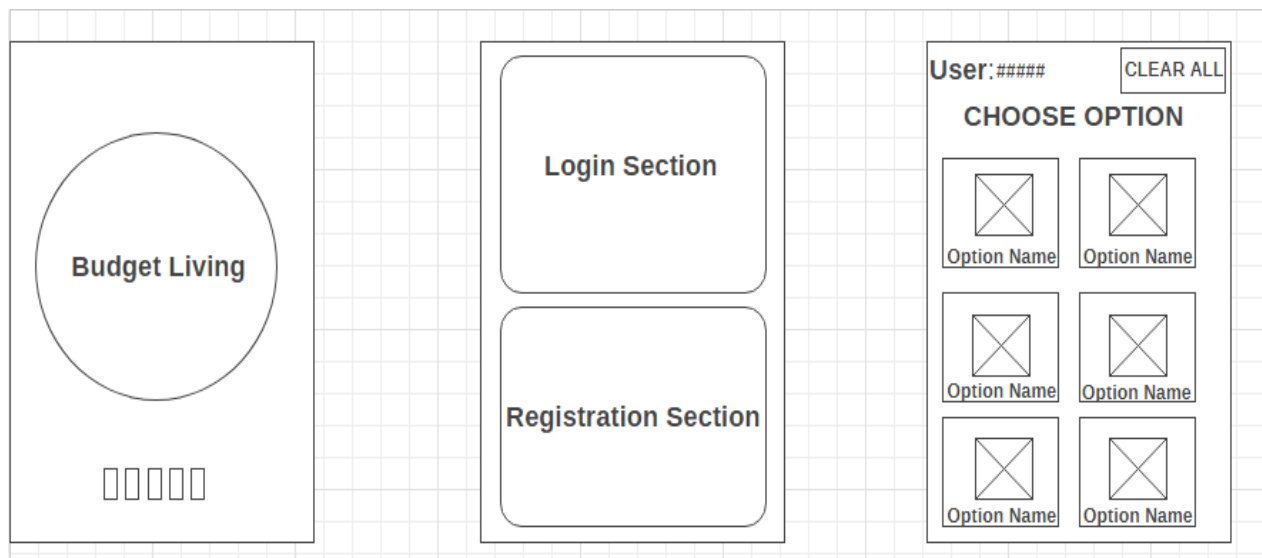


Figure 4.5.1: Wireframes (Screens 1-3)

- The first wireframe will be the splash screen that the user will see once they first open the application. There is a progress bar on the bottom of the screen that will show a loading progress and then refresh to a new screen after four seconds
- The second wireframe being seen here will be the next screen the users see, this screen is divided into two sections, one for login and the other for registration.
- The third wireframe is the home screen for the application which will have all the options available for the user that they can use to utilize the complete usage of the application, along with their username and an option to clear all data if required



<p>Monthly income?</p> <div>Income in AED</div> <p>CONFIRM</p>	<p>Expense Today?</p> <table><tr><td>Date</td><td>Choose Date</td></tr><tr><td>Option Name</td><td>AED</td></tr><tr><td>Option Name</td><td>AED</td></tr><tr><td>Option Name</td><td>AED</td></tr><tr><td>Option Name</td><td>AED</td></tr><tr><td>Option Name</td><td>AED</td></tr></table> <p>CONFIRM</p>	Date	Choose Date	Option Name	AED	Option Name	AED	Option Name	AED	Option Name	AED	Option Name	AED	<p>Reports</p> <p>LAST MONTH STATISTICS</p> <p>PRESENT MONTH STATISTICS</p> <p>LAST TO LAST MONTH STATISTICS</p>
Date	Choose Date													
Option Name	AED													
Option Name	AED													
Option Name	AED													
Option Name	AED													
Option Name	AED													

Figure 4.5.2: Wireframes (Screens 4-6)

- The first wireframe in this section is where the user will be asked to enter their income, and the only accepted result is a number, no text or special character is accepted here
- The next wireframe is the expenses screen, where the user will be able to see different categories and space next to them to fill out the total expense being made each day
- The third wireframe is the screen where users can choose to see the statistics from the options which include the last two months and the current month as well. In this screen, once an option is chosen, they can not only view the statistics in number, but also can see the statistics in a graphic view

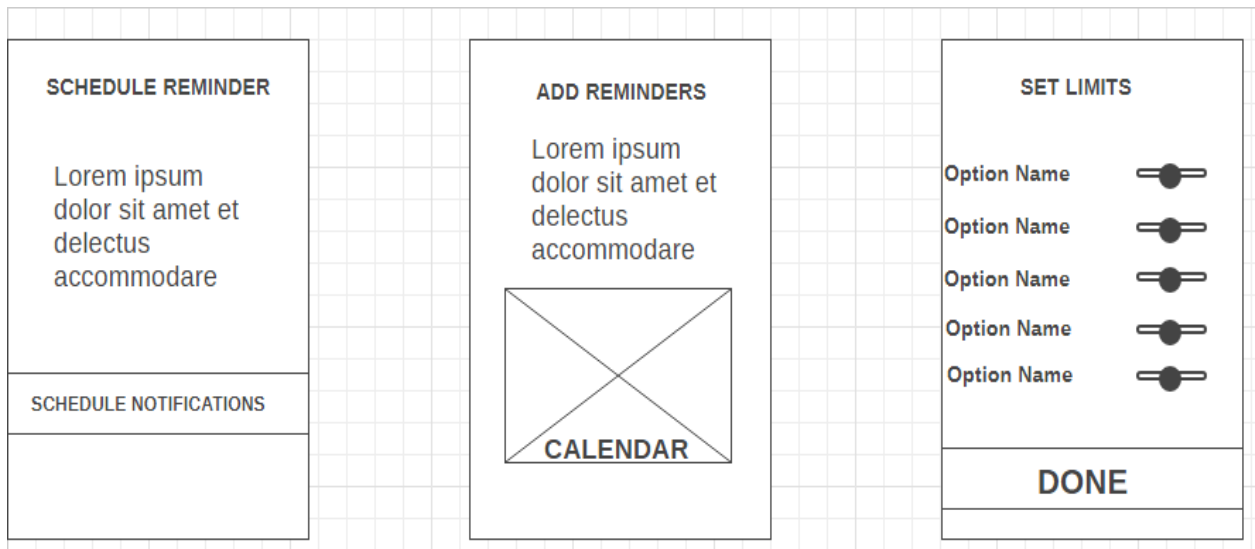


Figure 4.5.3: Wireframes (Screens 7-9)

- The first wireframe in this section is where the users can force the application to send them a notification at that very moment itself. The application already sends out one notification each day at a fixed time, but if the user wants to keep the notification on at that very moment, they can use this screen to get a notification pop up with the button click
- The next wireframe is for the user to set up future tasks using the calendar, once the calendar is clicked, it will link with the phone's default calendar and allow users to add tasks along with the date, so that it can be stored
- The last wireframe is the screen where the users can set a limit on the amount of expenses for each category, this is one of the most important screens as it will allow users to stay within the desired limit



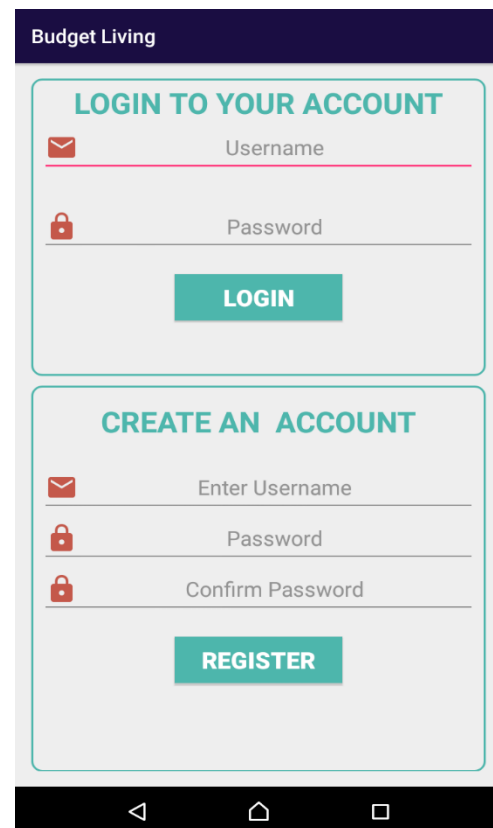
4.6 – Screens

The application is built of many classes (activities) which are interlinked with each other to have a complete working project. The following images are for all the screens which are designed by looking at the wireframes which are shown in Section 4.5

Figure 4.6.1: Splash Screen



Figure 4.6.2: User Login/Registration Screen



- The first two screens of the applications, which are two of the most important screens as the splash screen is the first screen the user sees, and as they say, “**first impression is the last impression**”, so the first screens should look attractive to the users. The login/registration screen is the most important as that activity allows users to get access to the application.



Figure 4.6.3: Home Screen

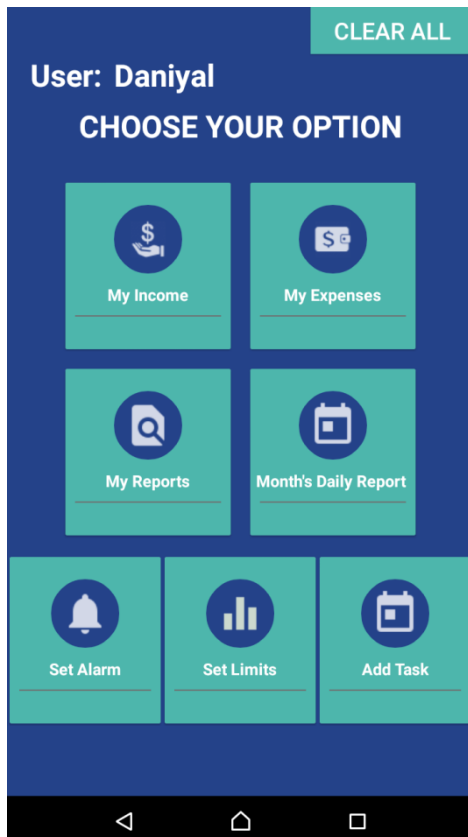
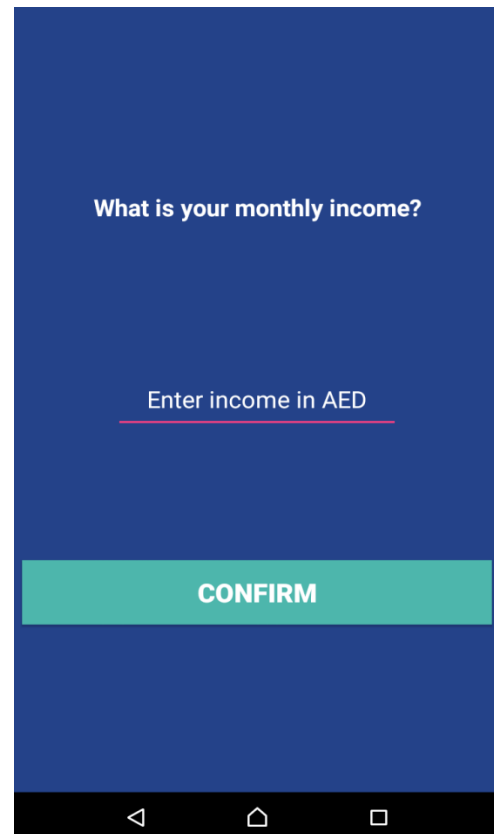


Figure 4.6.4: Income Activity Screen



- **Figure 4.6.3** shows all the options that are available in this application, these include the income, expense, monthly reports, current month's daily report, set notification, set limit, and further tasks activity. These are all the functionalities that are present in the application. **Figure 4.6.4** shows how the income activity is controlled.



Figure 4.6.5: Expense Screen

The Expense Screen features a dark blue background. At the top, the text "What was your expense today [in AED]?" is displayed. Below this, there are two columns of input fields. The left column contains buttons for "Date", "Food", "Shopping", "Car", "Rent", and "Miscellaneous". The right column contains corresponding input fields labeled "Choose Date", "Enter Food Expenses in AED", "Enter Shopping Expenses in AED", "Enter Car Expenses in AED", "Enter Rent Expenses in AED", and "Enter Other Expenses in AED". A large teal button labeled "CONFIRM" is positioned at the bottom of the form area. The screen is framed by a black bar at the bottom containing three white navigation icons: a back arrow, a home house icon, and a recent apps square icon.

Figure 4.6.6: Scheduling Notification

The Scheduling Notification screen has a dark blue background. At the top, the title "SCHEDULING A REMINDER" is shown. Below the title, a paragraph of text reads: "Click on the Daily Notification to have a notification on your phone to remind about the expenses. Notification will stay on the phone untill you clear it yourself". A teal button labeled "SCHEDULE DAILY NOTIFICATION" is located at the bottom of the screen. The screen is framed by a black bar at the bottom containing three white navigation icons: a back arrow, a home house icon, and a recent apps square icon.

- **Figure 4.6.5** displays how the user will enter input expenses and the different categories that are chosen are food, shopping, car, housing, and miscellaneous. **Figure 4.6.6** is scheduling activity that will allow users to force a notification onto their phone once they click the button.



Figure 4.6.7: Current Month Expenses Screen

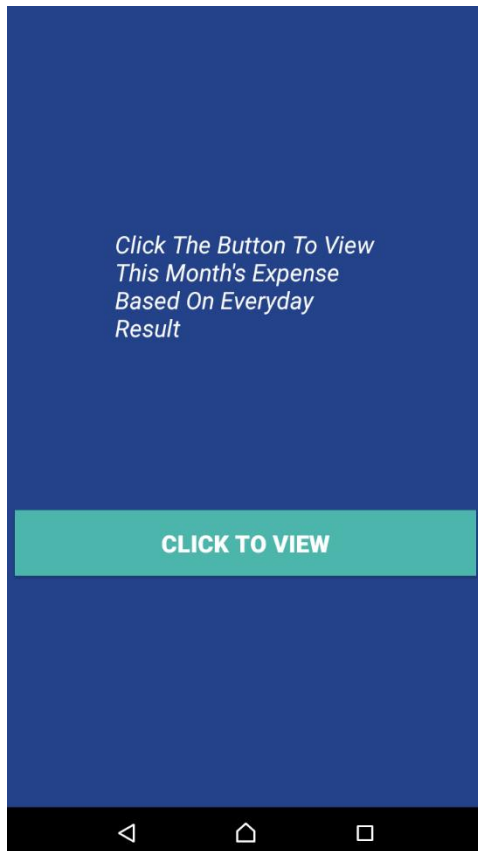


Figure 4.6.8: Current Month Detailed Expenses



- **Figure 4.6.7 & 4.6.8** are part of the same activity; this activity shows the total expense being made during the current month. The user clicks on the button which will allow a message to pop up displaying the date for expense along with the total expense being made in each category.



Figure 4.6.9: Future Task Reminder Screen

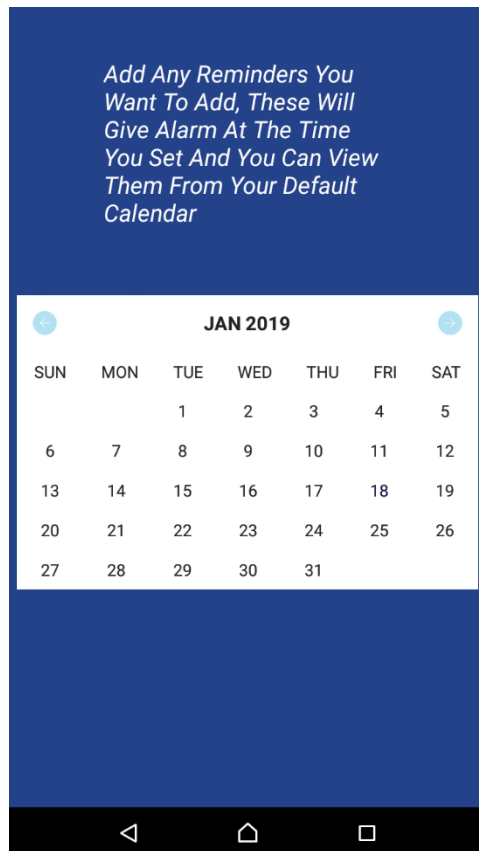
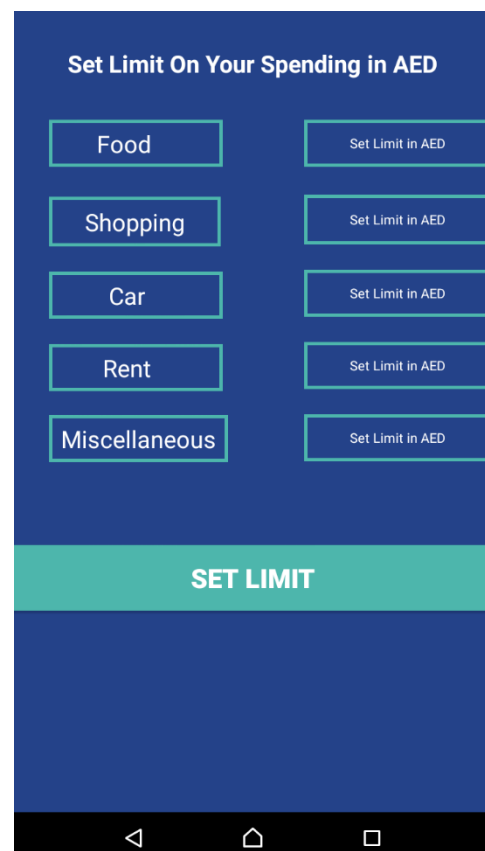


Figure 4.6.10: Setting Limit on Expenses Screen



- **Figure 4.6.9** shows the calendar where future tasks can be placed, while **Figure 4.6.10** displays the expenses categories broken down where the user can note down the limits for current month so that the budget can be maintained



Figure 4.6.11: Selection of Monthly Report Screen



Figure 4.6.12: Monthly Statistics Screen



- **Figure 4.6.11** shows the selection activity, only the month which has their expenses entered will have the button set as enabled, like for this case only the current month has expenses being mentioned, therefore making it the only working button. **Figure 4.6.12** shows the statistics for the month which is selected where all the income, savings, and every different expense is being displayed.



Figure 4.6.13: Monthly Statistic Graphically Screen

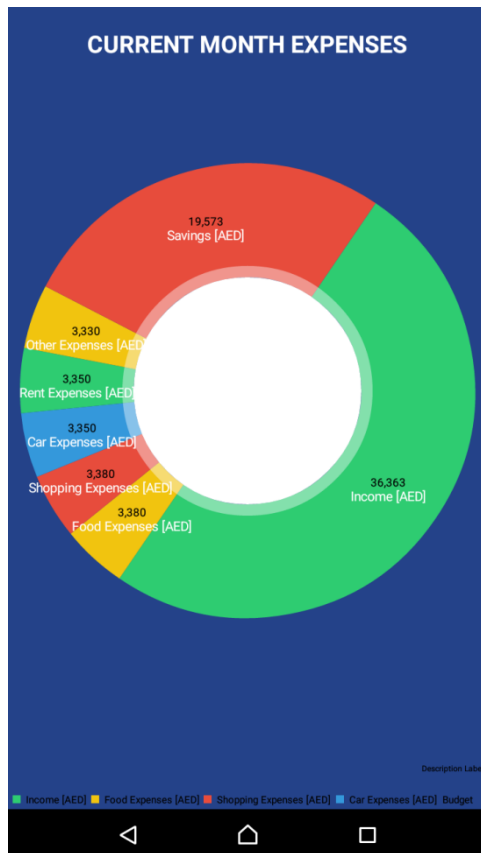
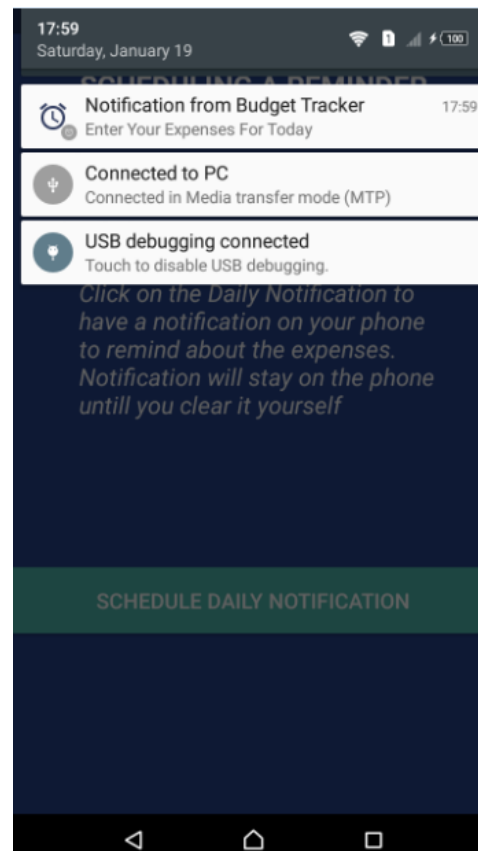


Figure 4.6.14: Notification Received



- **Figure 4.6.13** is a step ahead from where the activity from **Figure 4.6.12** will take the user; this activity takes all the information from the activity mentioned above, and then displays it in the form of a pie chart. **Figure 4.6.14** is a notification that the user can see on the phone when:
 - They click on the button to schedule notification
 - Wait till 23:00 (11pm) and get the automatic daily notification as that is the default time being set for the notification to be generated



Chapter 5 – Implementation

5.1 – Source Code

According to (Study.com, 2019), Implementation is the most important step in any software project as all the coding, development, and building of the project is being carried out in this phase. This is the step where the idea in mind becomes a reality. All the requirements collected are used as the ideas and a main product gets developed.

The images below show the java classes and the layout files that are used for the '**Budget Living**' application. Java folder has the java classes where the entire coding for the functionality is done, while the layout folder has layout files for the classes for designing purposes.

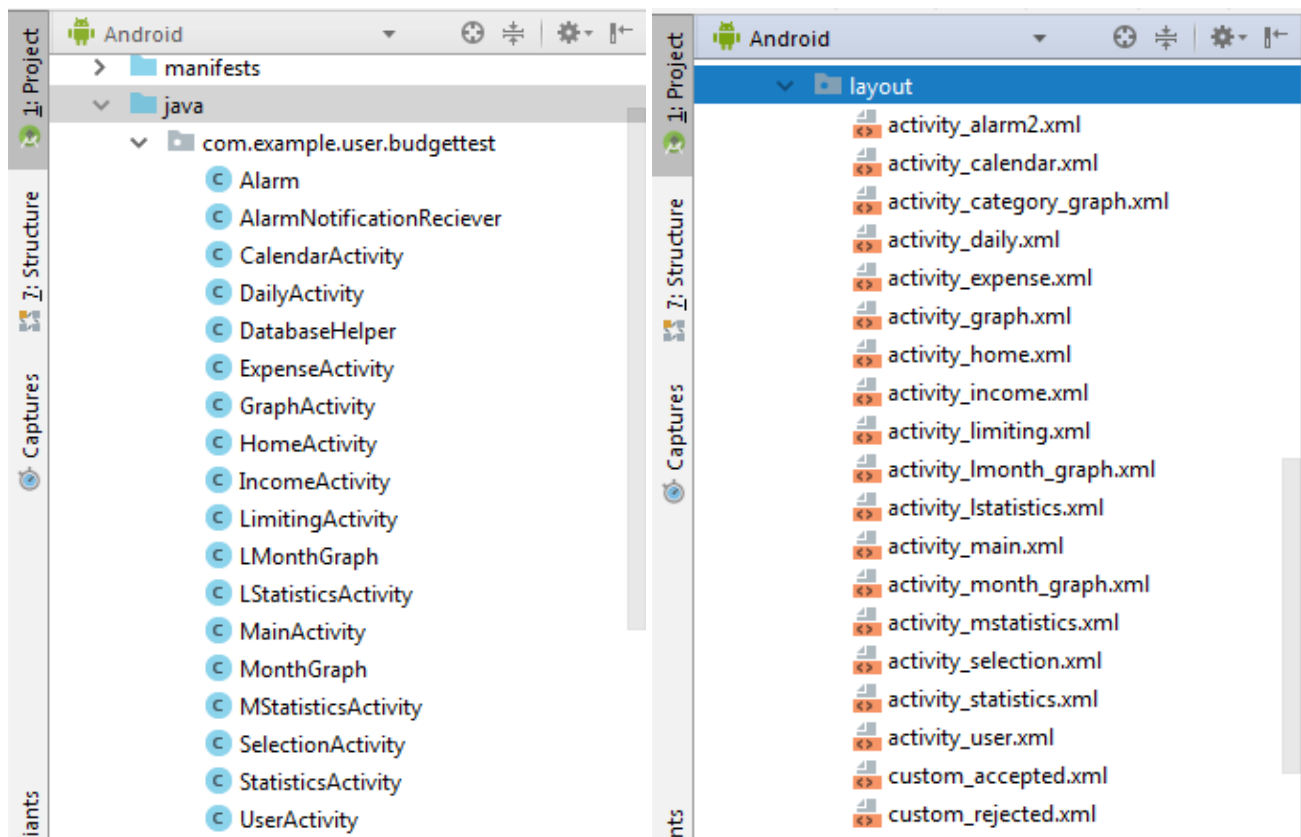


Figure 5.1.1: Source Files

Below are few of the source codes that make up the application are shown below



```
public boolean insert(String username, String pass) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("username", username);
    contentValues.put("pass", pass);
    //insert into table users
    long ins = db.insert( table: "users", nullColumnHack: null, contentValues);
    if (ins == 1) return false;
    else return true;
}

//method to check mail
public boolean checkMail(String username) {
    SQLiteDatabase db = this.getReadableDatabase();
    //cursor to check whether user with username typed is present or not
    Cursor cursor = db.rawQuery( sql: "Select * from users where username=?", new String[]{username});
    if (cursor.getCount() > 0) return false; // if more than 0 then not accepted
    else return true; //else accept it
}

// method to check password
public boolean checkMailPassword(String username, String pass) {
    SQLiteDatabase db = this.getReadableDatabase();
    //cursor to check whether user with username & pass typed is present or not
    Cursor cursor = db.rawQuery( sql: "Select * from users where username=? and pass=?", new String[]{username, pass});
    if (cursor.getCount() > 0) {
        return true; // if more than 0 then user is valid
    } else return false; // else not found
}

// Get username data from users table
public Cursor getAllData() {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res = db.rawQuery( sql: "select username from users", selectionArgs: null);
    return res;
}
```

Figure 5.1.2: User Registration/Login Code

```
//set alarm to give warning message
public void giveLimitAlarm(String category) {
    android.support.v7.app.AlertDialog.Builder builder = new android.support.v7.app.AlertDialog.Builder( context: HomeActivity.this);
    //Instantiate view using Layout Inflater
    View view = LayoutInflater.from(HomeActivity.this).inflate(R.layout.custom_rejected, root: null);
    //Set title and image
    TextView title = (TextView) view.findViewById(R.id.title);
    ImageButton imageButton = (ImageButton) view.findViewById(R.id.image);
    TextView customBoxTitle = view.findViewById(R.id.customBoxTitle);
    customBoxTitle.setText("You Have Crossed The Limit You Set For This Month");
    title.setText("Limit Crossed On " + category + " Expenses");
    imageButton.setImageResource(R.drawable.ic_limit_cross);
    //set button to move forward
    builder.setPositiveButton( text: "Okay, Noted", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
        }
    });
    //builder to set view and display
    builder.setView(view);
    builder.show();
}
```

Figure 5.1.3: Limits Crossed Code



```
//set alarm to give warning message
public void giveLimitAlarm(String category) {
    android.support.v7.app.AlertDialog.Builder builder = new android.support.v7.app.AlertDialog.Builder( context: HomeActivity.this);
    //Instantiate view using Layout Inflater
    View view = LayoutInflater.from(HomeActivity.this).inflate(R.layout.custom_rejected, root: null);
    //Set title and image
    TextView title = (TextView) view.findViewById(R.id.title);
    ImageButton imageButton = (ImageButton) view.findViewById(R.id.image);
    TextView customBoxTitle = view.findViewById(R.id.customBoxTitle);
    customBoxTitle.setText("You Have Crossed The Limit You Set For This Month");
    title.setText("Limit Crossed On " + category + " Expenses");
    imageButton.setImageResource(R.drawable.ic_limit_cross);
    //set button to move forward
    builder.setPositiveButton( text: "Okay, Noted", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
        }
    });
    //builder to set view and display
    builder.setView(view);
    builder.show();
}
```

Figure 5.1.4: Transfer Data from One Activity to another Code

```
// Method to insert into SQLiteDatabase
public void InsertDataIntoSQLiteDatabase(){
    // if date is not blank, accept the expenses that is being entered and save into database
    if (!date.equals("")) {
        SQLiteDatabaseQueryHolder = "INSERT INTO expense (months, username, date, food, shopping, car, rent, others)
        sqliteDatabaseObj.execSQL(SQLiteDatabaseQueryHolder);
        giveExpenseSuccess();
    }
    else {
        //otherwise call faulure method
        giveExpenseFailure();
    }
}
```

Figure 5.1.5: Inserting Expense Code



```
// insert data into SQLiteDatabase
public void InsertDataIntoSQLiteDatabase() {
    //if income is added, insert into the database
    if (!income.equals("")) {
        SQLiteDatabaseQueryHolder = "INSERT INTO income (username, income) VALUES('" + vincome + "','" + income + "');";
        sqLiteDatabaseObj.execSQL(SQLiteDatabaseQueryHolder);
        //run the method if income is added
        giveIncomeSuccess();
    } else {
        //run the method if income is not added
        giveIncomeFailure();
    }
}
```

Figure 5.1.6: Inserting Income Code

```
// method to set repeating notification
private void setRepeatedNotification(int ID, int hh, int mm, int ss) {
    //call alarm manager to get service
    AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
    Intent alarmIntent = new Intent( packageContext: MainActivity.this, com.example.user.budgettest.AlarmNotificationReceiver.class);
    alarmIntent.putExtra( name: "ID", ID);
    //create a pending intent
    PendingIntent pendingIntent = PendingIntent.getBroadcast( context: MainActivity.this, ID, alarmIntent, PendingIntent.FLAG_UPDATE_CURRENT);
    //get calendar instance to set time
    Calendar calendar = Calendar.getInstance();
    Calendar now = Calendar.getInstance();
    calendar.set(Calendar.HOUR_OF_DAY, hh);
    calendar.set(Calendar.MINUTE, mm);
    calendar.set(Calendar.SECOND, ss);
    //check whether the time is earlier than current time. If so, set it to tomorrow. Otherwise, all alarms for earlier time will fire
    if(calendar.before(now)){
        calendar.add(Calendar.DATE, amount: 1);
    }
    //wake up the phone to get notification
    alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pendingIntent);
}
```

Figure 5.1.7: Repeating Alarm Code



```
//Cursor with statement to get monthly car expense for last month
public Cursor getMonthlyCar() {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursorssss = db.rawQuery( sql: "select sum(car) from expense where date like '%" + month + "/" + year + "'", selectionArgs: null);
    return cursorssss;
}

//Cursor with statement to get monthly rent expense for last month
public Cursor getMonthlyRent() {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursorssss = db.rawQuery( sql: "select sum(rent) from expense where date like '%" + month + "/" + year + "'", selectionArgs: null);
    return cursorssss;
}

//Cursor with statement to get monthly expenses on others for last month
public Cursor getMonthlyOthers() {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursorssss = db.rawQuery( sql: "select sum(others) from expense where date like '%" + month + "/" + year + "'", selectionArgs: null);
    return cursorssss;
}
```

Figure 5.1.8: Get Expenses from Database Code

```
public void onDateSelected(Date date) {
    //Create instance of calendar
    Calendar cal = getInstance();
    //set new intent that opens calendar with title and address to be added for event
    Intent intent = new Intent(Intent.ACTION_EDIT);
    intent.setType("vnd.android.cursor.item/event");
    Intent calendarIntent = new Intent(Intent.ACTION_INSERT);
    calendarIntent.setData(CalendarContract.Events.CONTENT_URI);
    calendarIntent.putExtra(CalendarContract.Events.TITLE, value: "Enter Title");
    calendarIntent.putExtra(CalendarContract.Events.EVENT_LOCATION, value: "Enter Venue");
    cal = Calendar.getInstance();
    calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, cal.getTime().getTime());
    calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, cal.getTime().getTime());
    startActivity(calendarIntent);
    // give Calendar Success message using function
    giveCalendarSuccess();
}
```

Figure 5.1.9: Calendar Activity Code



Chapter 6 – Testing

6.1 – Test Plans

A testing plan is where different approaches are made to a situation, to make sure there are not any loopholes in the system. Testing helps rectify any mistakes in the development phase and makes sure that the final product meets the user satisfaction level. The five types of test being considered in this project are:

- Functionality
- Performance
- Security
- Usability
- Compatibility

These test plans were identified in section 3.6. **Functionality** test is a must as it allows us to check whether all the functionalities are working properly or not. **Performance** test checks the application's performance and makes sure it is causing no problem to the phone. **Security** test is one of the most important tests because security is a basic problem found in almost every application, and a test to make sure the application is secure is needed. **Usability** and **Compatibility** tests are needed so that it is made sure that the application can be understood by everyone and accessed by every user no matter what the phone size is. All these tests combine to assure that the application is fully working and ready to be used by users. Along with the test plans, there is an important checklist section being covered in this section.

6.2 – Test Cases

Test Case	Test Type	Expected Outcome	Actual Outcome	P(Pass)/ F(Fail)
Validate whether all mandatory fields are set as required	Functionality	If activity is submitted without mandatory fields, error message given	As seen in section 4.6, there are custom boxes that pop up when mandatory fields are missing.	Pass (Results in section 6.4)



Validate whether the phone allows multi-tasking, calls, SMS, and notifications from other applications	Functionality	The application should show any notification, call, or SMS on the top of the screen as notification	When the application receives a notification from another application and the user clicks on it, the application gets minimized	Pass (Results in section 9.2)
Validation that the application is installed correctly	Functionality	When the user downloads this application, a message saying "Application Installed Successfully" should be displayed	As the user installs the application a message is given out saying "Application Installed Successfully"	Pass
Validate that the application checks the data type of data being entered to make sure it is in correct form	Functionality	The keyboard that appears on screen should be set in such a way that there is no chance of getting faulty values	The application has text type and number type validation being kept on activities to make sure the keyboard shown is off the format needed	Pass (Results in section 6.4)



To validate that the application performs as per the requirements when it has many applications running in the background	Performance	The application should have no effect in the way of running when there are many background applications running	The application moves smoothly even when there are around 6-7 applications running in the background	Pass (Results in section 9.2)
Validate that the response time is up to the mark	Performance	The application's response time should be up to the mark required by users	The application's response time is in seconds which helps user get the required work done in seconds	Pass
Validation that any unauthorized person cannot have access to the data of any other user	Security	The application should not allow any other unauthorized access to the application	The person using the application can only see their own data as there is no chance of data being loss because it is kept secure and not anywhere on the cloud where any hacker could access	Pass



Ensuring that the zoom-in, zoom-out feature works fine for the pie charts	Usability	The graphs should allow zooming-in and zooming-out on the sectors of the chart	Each sector from the chart can be clicked and get zoomed-in and zoomed-out	Pass
Ensuring that the text sentences, graphs, text boxes are all readable	Usability	The application's text and everything else should be in readable form for the audience	All the texts, text boxes, graphs and every content are carefully being seen into to make sure the size, font, and color are all in a way that the user can read it easily	Pass
Checking that the UI is same for every device	Compatibility	The application's UI must be same no matter what the screen size of the device is	Application has everything in a well-managed layout, so the design remains same in all devices and nothing stays inaccessible	Pass
Checking that the call functionality works while the application is in use	Compatibility	The application should allow incoming calls even when the application is in use	The application first shows the notification of call, and then opens the call screen	Pass (Results in section 9.2)



Validation that the application does not ask for permission for tasks that it will not be using	Security	The application should work without asking for extra permissions that is not required.	The application asks for permission to get when the phone ID gets activated to give notification that there is an incoming call if any, also asks permission to access SD card as the data for the application is stored on it	Pass (Results in section 9.2)
-------------------------------------------------------------------------------------------------	-----------------	----------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------

6.3 - Important Checklist

Task	Checked/Not Checked
Installation of application takes minimum time	Checked
Splash screen moves to another screen automatically after four seconds	Checked
Back key pressed on the home screen should not close the application	Checked
No effect on the usage of the application while it is on charge	Checked
No interruption of keypad opening in between the application where it is not required	Checked
High Performance level of application	Checked
Clearing all data method is well defined	Checked
Uninstallation of application takes minimum time	Checked



6.4– Error and Success Functionality Test

The application is incomplete without error and success messages, the figures below show all the pop-up messages that pop up, along with when each of them appear on the screen.

Figure 6.4.1: Different Passwords Entered

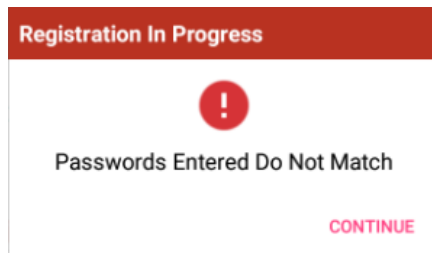
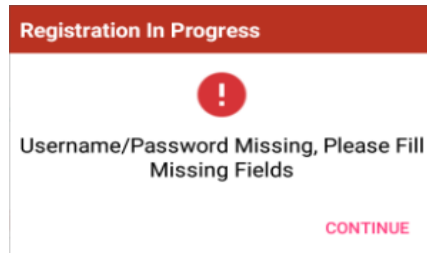


Figure 6.4.2: Username/Password missing

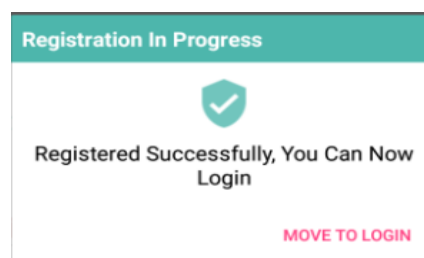


- **Figure 6.4.1 & 6.4.2** are both for the Registration purpose, if the user sets their password and confirms the password with some other password, the error message as shown in **Figure 6.4.1** will be popped out, while if the user forgets to enter one of the fields, then the error message shown in **Figure 6.4.2** is shown to the user.

Figure 6.4.3: Existing User



Figure 6.4.4: Registration Success



- **Figure 6.4.3** shows the error pop-up the user will see if they enter a username that is already in use, they will be asked to try with a different username and **Figure 6.4.4** shows that registration has been completed successfully and the user can now login and access the application



Figure 6.4.5: Fields Missing

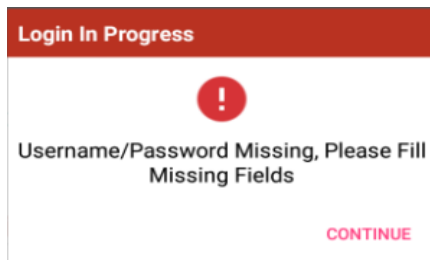
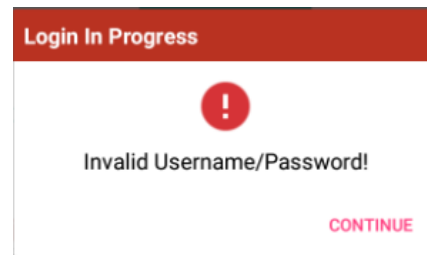


Figure 6.4.6: Wrong details



- **Figure 6.4.5** shows the error message that will pop up if either the username or password is being left empty. This message will allow the users to know they cannot access the application without entering both their details. **Figure 6.4.6** displays when a user enters their details which do not match with any data on the database

Figure 6.4.7: Login Successful

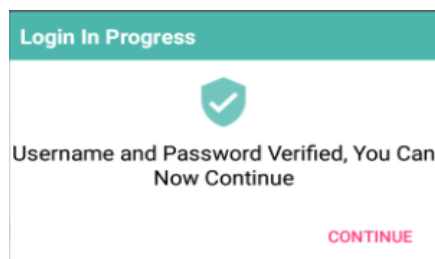
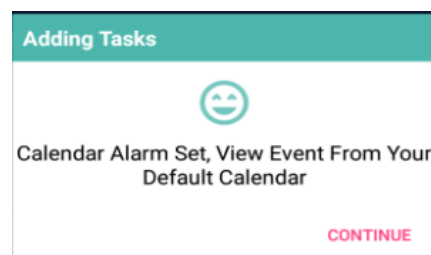


Figure 6.4.8: Tasks added



- **Figure 6.4.7** is the message that appears when details entered are correct and login process is made successful, while **Figure 6.4.8** is where users use the calendar to set the tasks for future and once that is done, a message is shown up on the activity to notify them that the event has been created on the calendar.



Figure 6.4.9: Empty Income

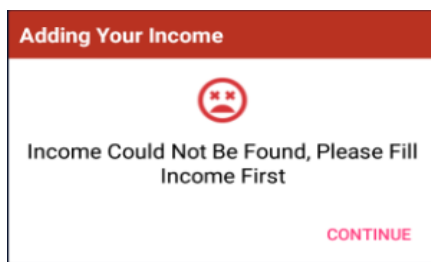
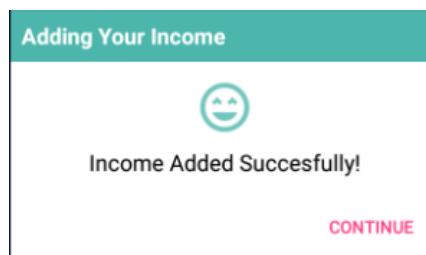


Figure 6.4.10: Income Added



- **Figure 6.4.9 & 6.4.10**, both are for income adding process, if the user leaves the income field empty and submits, then the pop-up message from **Figure 6.4.9** appears, while if the user adds income and then clicks submit, then the pop-up message from **Figure 6.4.10** appears.

Figure 6.4.11: Expense Not Accepted

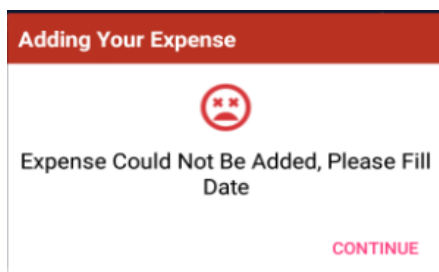
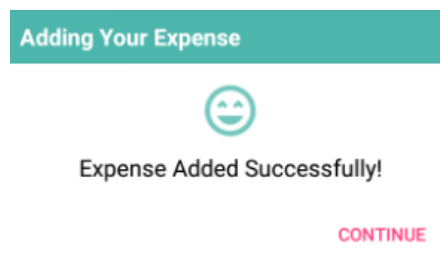


Figure 6.4.12: Expense Added



- **Figure 6.4.11 & 6.4.12**, both are from the expenses process. If the user does not mention the date for expense, then pop-up from **Figure 6.4.11** is appeared on the activity. When the user enters the date along with expenses, then the pop-up from **Figure 6.4.12** is shown



Figure 6.4.13: Improper Income

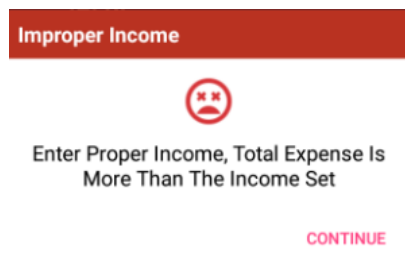
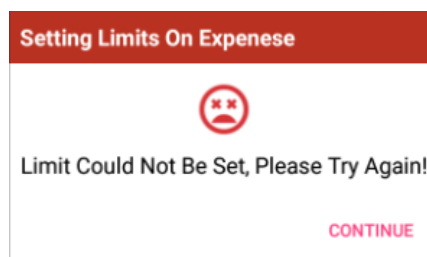


Figure 6.4.14: Limit Not Accepted



- **Figure 6.4.13** is an error message shown when the user has set the income value as improper and as a result the savings being made are shown as negative, this message is given out to notify the user that there is an improper income value set and they need to change that. **Figure 6.4.14** shows the limits not being set because the user kept all the limits for expenses as zero, and that is not allowed while setting limits

Figure 6.4.15: Limits Accepted

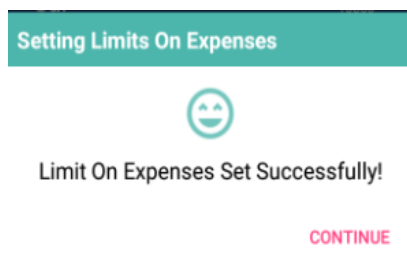


Figure 6.4.16: Limit Crossed on Expense Category



- **Figure 6.4.15** shows a positive pop-up displaying message that the limit set has been set successfully and the application will now manage the expense being made to make sure they are kept under the limit. **Figure 6.4.16** however, shows a warning message that the limit has been crossed for the category the user has set limit for. In this figure, the example is taken from the **food** category.



Chapter 7 – Demonstration

7.1 – Activity Flow

The application launches with a splash screen and displays two options for the user; a login section and a registration section.

If the user is doing registration first, there are two possibilities:

- **Approved and Rejected**

The registration can get rejected if the user:

- Misses to enter either the username or password
- Enters an existing username
- Mismatches Password and Confirm Password

But, if the registration can get approved when the user:

- Enters a unique username
- Has both the passwords matching

Once the registration gets approved, the user will be guided to the login section where the user can login. The login can be rejected if the user:

- Enters wrong username/password
- Forgets to enter username/password

The login is approved when user: Types the correct username and password

Once login is authorized, then the user will have many options, these include:

- Adding Income
- Setting Expenses
- Setting Limit
- Use Calendar for future tasks
- Set Daily Notification
- View Daily Expense Report
- View Monthly Statistics

If the user goes to the income activity, the income could be either accepted or rejected.

It gets accepted when:

- There is an integer value entered.



It gets rejected when:

- The income field is kept empty.

If the user goes to the expense activity, the expenses could be either accepted or rejected.

It gets accepted when:

- The date for spending is given.

It gets rejected when:

- The date for expenses is missing.

If the user goes to the setting limit activity, the limits could be either accepted or rejected.

It gets accepted when:

- All the categories are not stored as **zero**.

It gets rejected when:

- All the categories are kept as zero.

If the user selects daily notification activity, the application:

- Sets notification on their phone to remind themselves they need to enter their spending.

If the user selects monthly statistics activity, the application:

- Views Monthly Statistics for the past three months. The statistics although, do not get shown if the income does not get added and if the income is not a proper value.

If the user selects daily expenses activity, the application:

- Allows viewing Daily Expenses Report in a message box where all expenses made in each category of expense is being shown along with the total expense being displayed on screen.

If the user selects calendar activity, the application:

- Opens default calendar where the future tasks can be stored onto the default calendar.



Chapter 8 – Conclusion

Budget Living application is the first application that I have developed for Android devices, the idea of developing an application based on budgeting came in my mind because of hearing many people talk about how they had spent more than they were expecting and that could cause financial problems. So that is why Budget Living was being developed to help people monitor their expenses. During this project I have learned that how important it is to know the logic behind each action before starting to code. One of the most important lesson learned was that every problem has a reason, and till you cannot pinpoint the reason, you should not start coding as it would result in some illogical code being printed out.

The Waterfall model for this project proved to be very helpful as it allowed to carry out the project in a very systematic order. It allowed completing a whole phase to get to the next stage. Once the wireframes were sketched, based on the logic and structure that were used from the UML Diagrams. Since there are more and more improvements being made in the software section, if I had to recreate this application, I would use **Zoho Creator** to develop this application. The reason why I would use **Zoho Creator** is because that is a low-code app development platform, which would allow the application to be built faster than the time it usually takes to develop a full-working application.

There are plans in the future for this application which includes getting a version released on the Google Play Store for people to use and then once it becomes famous within the Android community, then develop the iOS version and put it on the Apple Store for iOS users to use the application too.



Chapter 9 – Appendix

9.1 – Quick Start Guide

To run the application:

1. Run Android Studio
2. Open Main Activity.java
3. Run the Main Activity
4. Choose either the device attached with USB (making sure developer tool is on) or choose a virtual device to simulate the application
5. Click Ok
6. Wait for gradle to build
7. Use the application

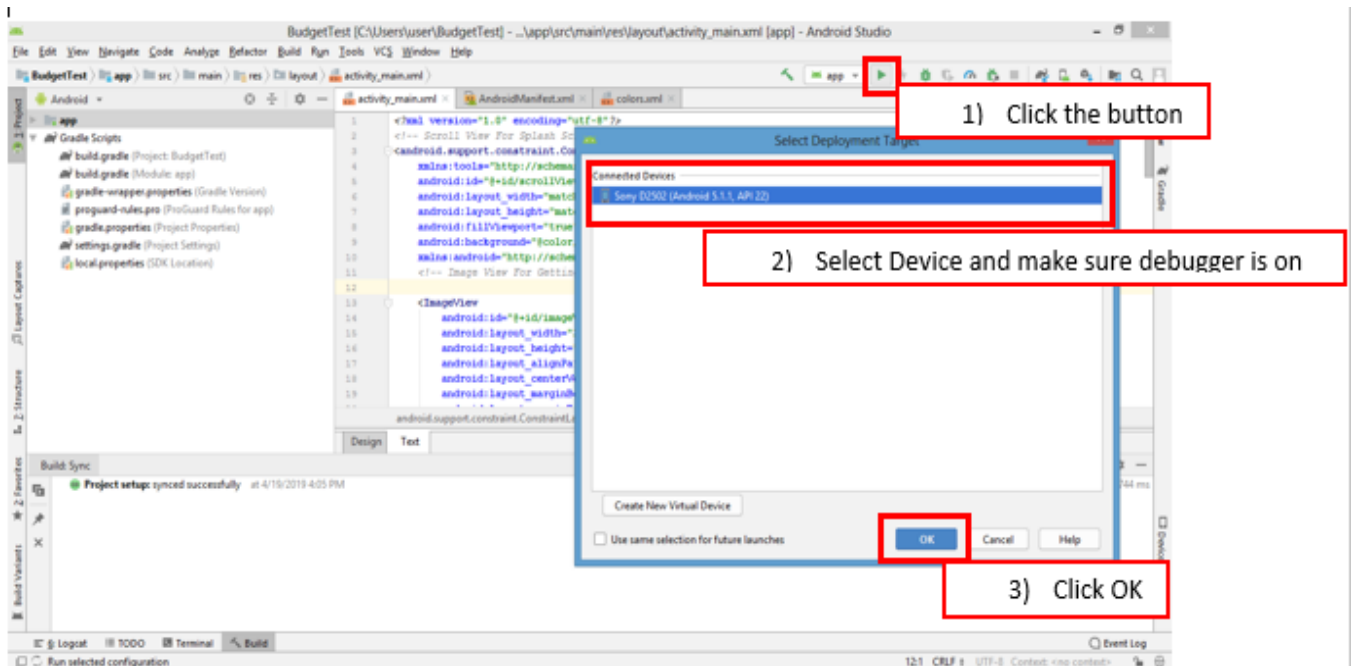


Figure 9.1.1: How to run the application

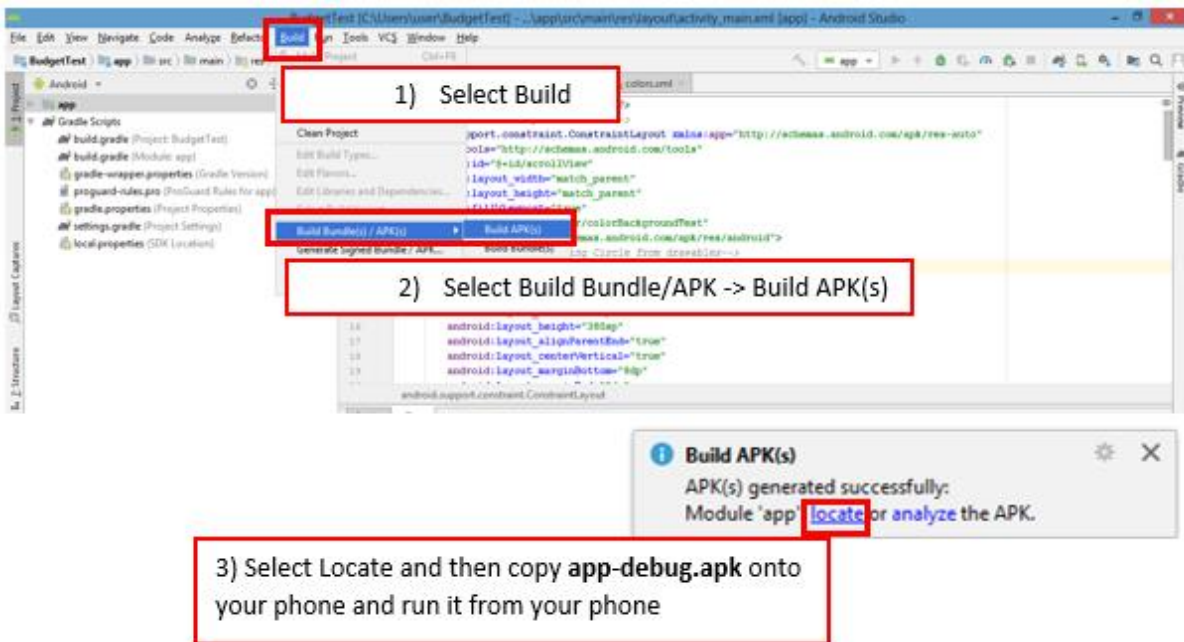


Figure 9.1.2: How to get application's APK

9.2 – Test Cases Results

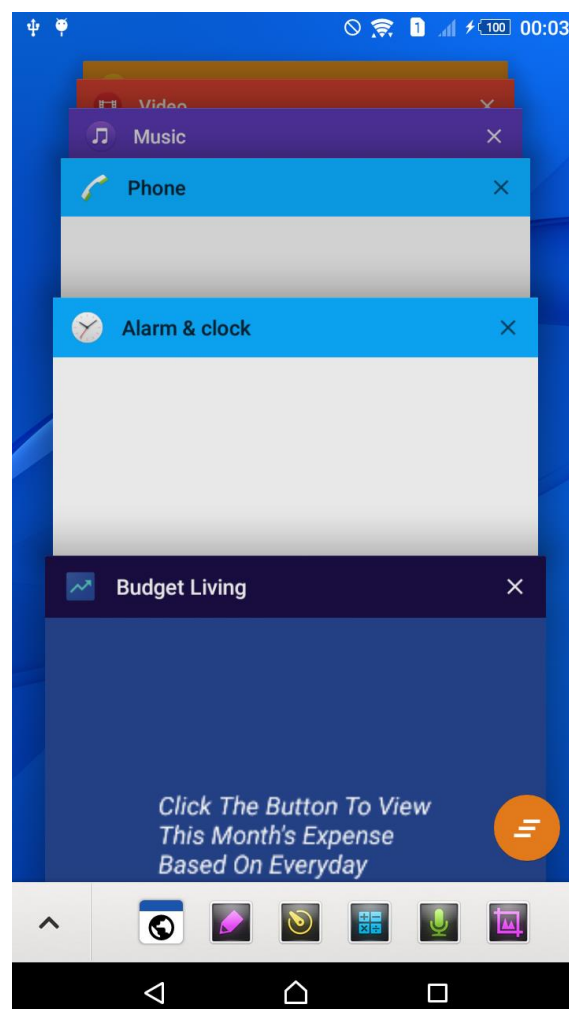


Figure 9.2.1: Test for multi-tasking

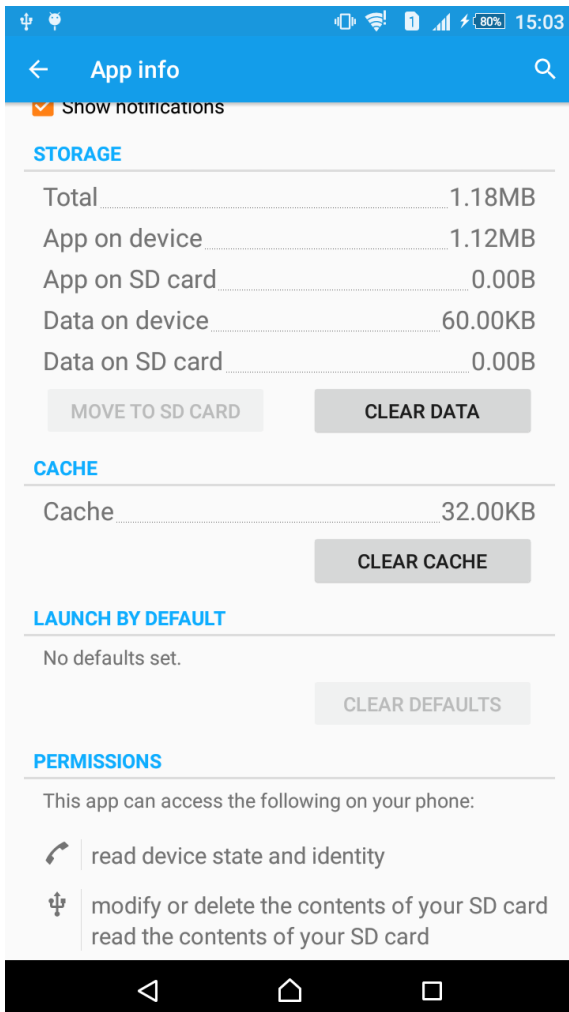


Figure 9.2.2: Test for app permissions

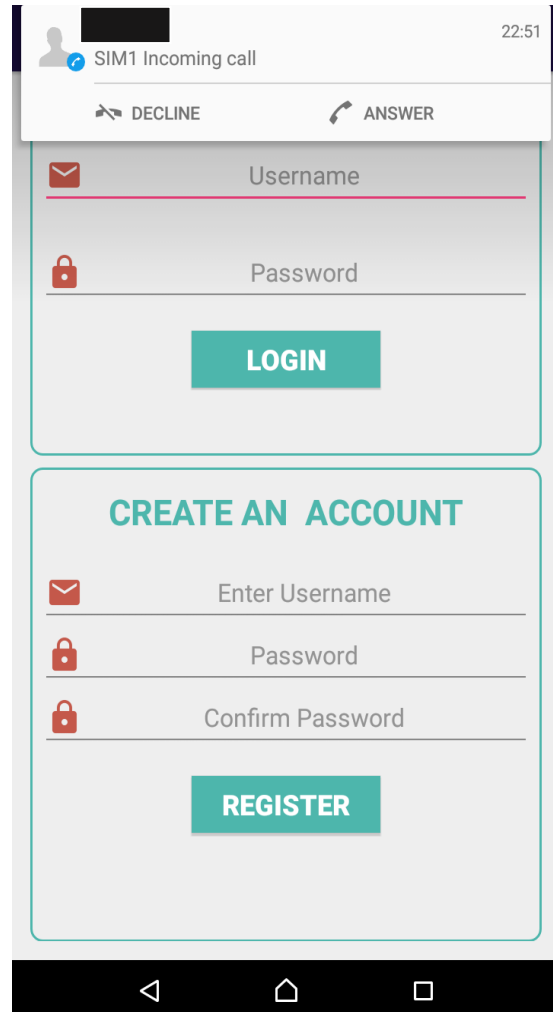







Figure 9.2.3: Test for getting calls in app use



9.3 – Project Meeting Logs

Date	Comments	Supervisor's Signature
18 Oct 2018	Requirements for Proposal	
20 Oct 2018	Feedback for draft Proposal	
04 Nov 2018	Literature Review & Brief discussion	
11 Nov 2018	Feedback for literature review draft	
12 Dec 2018	Literature review, Designs discussion	
26 Dec 2018	Requirements for Initial Draft	
07 Dec 2019	Get feedback for initial draft	
08 Dec 2019	Finalize initial draft.	



Date	Comments	Supervisor's Signature
30-Dec	Information on all contents	
8-Jan	Feedback on few completed chapters	
18-Jan	Feedback on report	
24-Jan	Report format discussed	
24-Feb	Format of report discussed	
4-March	Report work done & shown feedback	
14-March	Detailed feedback recieved	
29-March	Changes on feedback recieved	 2

[illegible]



9.4 - References

- Agilemodeling.com. (2019). UML 2 Sequence Diagrams: An Agile Introduction. [online] Available at: <http://www.agilemodeling.com/artifacts/sequenceDiagram.htm> [Accessed 18 Jan. 2019].
- Appy Pie. (2018). *9 Deadly Yet Common Mobile App Budgeting Mistakes to Avoid*. [online] Available at: <https://www.appypie.com/blog/common-mobile-app-budgeting-mistakes-to-avoid> [Accessed 23 Nov. 2018].
- Barba, R. (2018). *63% Of Smartphone Users Have At Least One Financial App* | Bankrate.com. [online] Bankrate. Available at: <https://www.bankrate.com/personal-finance/smart-money/americans-and-financial-apps-survey-0218/> [Accessed 23 Nov. 2018].
- Bell, A. (2018). *6 Reasons Why You NEED a Budget*. [online] Investopedia. Available at: <https://www.investopedia.com/financial-edge/1109/6-reasons-why-you-need-a-budget.aspx> [Accessed 23 Nov. 2018].
- Clearbridge Mobile. (2018). *5 Major Mobile App Budgeting Mistakes* | Clearbridge Mobile. [online] Available at: <https://clearbridgemobile.com/5-major-mobile-app-budgeting-mistakes/> [Accessed 23 Nov. 2018].
- Existek.com. (2019). [online] Available at: <https://existek.com/blog/sdlc-models/> [Accessed 20 Jan. 2019].
- Experience UX. (2019). What is wireframing | Experience UX. [online] Available at: <https://www.experienceux.co.uk/faqs/what-is-wireframing/> [Accessed 18 Jan. 2019].
- Jay, T., Sunil, K., and Sukanya, K. (2014) 'A Study Of Mobile Application Preferences Of Youth And Future Scope' *'Researchjournal's Journal of Marketing'* Vol. 2, no.3. Available at: https://www.researchgate.net/publication/262188801_A_Study_Of_Mobile_Application_Preferences_Of_Youth_And_Future_Scope [Accessed 28 Dec. 2018].
- Long, K. (2018). *5 Reasons Budgeting Apps Don't Work For Most People*. [online] Forbes. Available at: <https://www.forbes.com/sites/financialfinesse/2017/10/26/5-reasons-budgeting-apps-dont-work-for-most-people/#75af058448db> [Accessed 23 Nov. 2018].
- Lucidchart. (2019). UML Class Diagram Tutorial. [online] Available at: <https://www.lucidchart.com/pages/uml-class-diagram?a=0> [Accessed 18 Jan. 2019].
- Mymoneycoach.ca. (2018). *What is Budgeting and Why is it Important?* | My Money Coach. [online] Available at: <https://www.mymoneycoach.ca/budgeting/what-is-a-budget-planning-forecasting> [Accessed 23 Nov. 2018].
- Oer.nios.ac.in. (2019). Phases of System Development Life Cycle - NIOS. [online] Available at: http://oer.nios.ac.in/wiki/index.php/Phases_of_System_Development_Life_Cycle [Accessed 18 Jan. 2019].



- Phil, M. and Vinodhini, K. (2016) 'Personalized Expense Managing Assistant Using Android', International Journal of Computer Techniques, 3(2). Available at: <http://oaji.net/articles/2017/1948-1513926576.pdf> (Accessed 23 November 2018).
- Rachel, H., Derek, F., and David, D. (2013) 'Usability of mobile applications: literature review and rationale for a new usability model', 'Journal of Interaction Science', vol. 1, no.1. Available at: <https://link.springer.com/content/pdf/10.1186%2F2194-0827-1-1.pdf> [Accessed 28 Dec. 2018].
- SearchSoftwareQuality. (2019). *What is requirements analysis (requirements engineering)? - Definition from WhatIs.com.* [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/requirements-analysis> [Accessed 17 Jan. 2019].
- Simon, H (2018). *Android vs. iOS: Which smartphone platform is the best?* [online] Available at: <https://www.digitaltrends.com/mobile/android-vs-ios/> [Accessed 23 Dec. 2018]
- Software Testing Material. (2019). *Software Development Life Cycle - SDLC | Software Testing Material.* [online] Available at: <https://www.softwaretestingmaterial.com/sdlc-software-development-life-cycle/> [Accessed 18 Jan. 2019].
- Statista. (2018). *Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018| Statista.* [online] Available at: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/> [Accessed 23 Nov. 2018].
- Statista. (2018). *Number of smartphone users in the U.S. 2010-2022 | Statista.* [online] Available at: <https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/> [Accessed 23 Nov. 2018].
- Study.com. (2019). [online] Available at: <https://study.com/academy/lesson/implementation-coding-phase-in-sdlc.html> [Accessed 19 Jan. 2019].
- Techopedia.com. (2019). *What is a Functional Requirement? - Definition from Techopedia.* [online] Available at: <https://www.techopedia.com/definition/19508/functional-requirement> [Accessed 17 Jan. 2019].
- Thanapal, P., Patel, M., Raj, T. and Kumar, J. (2018). *Income and Expense Tracker.* [online] Indjst.org. Available at: <http://www.indjst.org/index.php/indjst/article/view/59166/46299> [Accessed 23 Nov. 2018].
- The Balance. (2018). *Why You Need to Track Expenses to Take Control of Your Finances.* [online] Available at: <https://www.thebalance.com/is-it-important-to-track-my-expenses-2385679> [Accessed 23 Nov. 2018].



- Theinternationalfinance.com. (2018). *Considering the Pros and Cons of Personal Budgeting*. [online] Available at: <http://www.theinternationalfinance.com/2012/08/considering-pros-and-cons-of-personal.html> [Accessed 23 Nov. 2018].
- Venkata, N., Divya, D., Taeghyun, K., and Manikanta, I. (2014) 'Factors Influencing Quality Of Mobile Apps: Role of Mobile App Development Life Cycle', *International Journal of Software Engineering & Applications (IJSEA)*, 5(5). Available at: <https://arxiv.org/ftp/arxiv/papers/1410/1410.4537.pdf> [Accessed 23 Nov. 2018]