Last edited by **Denzil Correa** 1 year ago

Basic prototype

BudgetWidget

Overview

Welcome back! This blog conveys to our readers a detailed overview of our second milestone, which is the Basic Prototype version of our android application - BudgetWidget Covering every aspect and event by giving insights through user stories, analysis, illustrations of various drafted diagrams, software development architecture, and App preview along with our unyielding journey spanned across this process, to impeccably demonstrate our application in a precise, yet uncomplicated manner.

Essential Requirements of the Project

- [ ] Andriod Application Developement

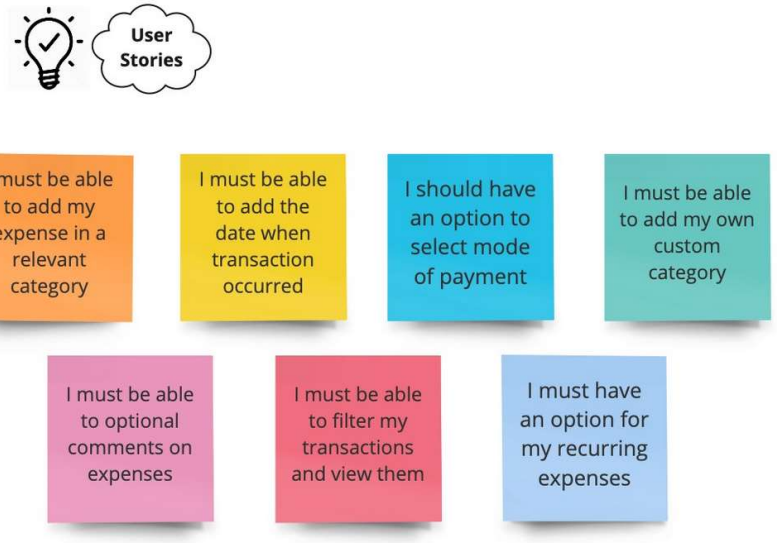
Essentials	Necessary	Nice to Have
Different categories	Icons	Currencies +plus conversion
Add Date.Month.Year	Delete entries	Help menu
Filters (*by category, Month, etc*)	Budget threshold	Analytics
Additional notes per entry	Repeated transactions	
Mode of payment		

- [ ] Create User Stories
- [ ] Draft Diagrams (*Case, Activity, Class Diagrams*)
- [ ] Documentation

Application Preview

User stories

In software development, a user story is an informal, natural language description of features of a software system. They are written from the perspective of an end-user or user of a system, and maybe recorded on index cards, Post-it notes, or digitally in project management software. In other words; User stories are short, simple descriptions of the requirement specified by the customer of the system. They typically follow a simple template:
As a < type of user >, I want < some goal > so that < some reason >



USER STORIES
As a User, *I want to sign up to log in to the app.*
As a User, *I want to log in to the app to add my expense.*
As a User, *I want to enter expenses into different categories.*
As a User, *I want to know my balance.*
As a User, *I want to add a date for a particular expense.*
As a User, *I want to add additional notes for an expense entry.*
As a User, *I want to enter the mode of payment for an expense.*
As a User, *I want to have icons for different functions in the application.*
As a User, *I want to delete a particular entry.*
As a User, *I want to set my budget plan.*
As a User, *I want to schedule my repeated transactions for every month.*

Use Case Diagrams

A Use-case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. A Use-case is a methodology used in system analysis to identify, clarify, and organize system requirements.

In other words,

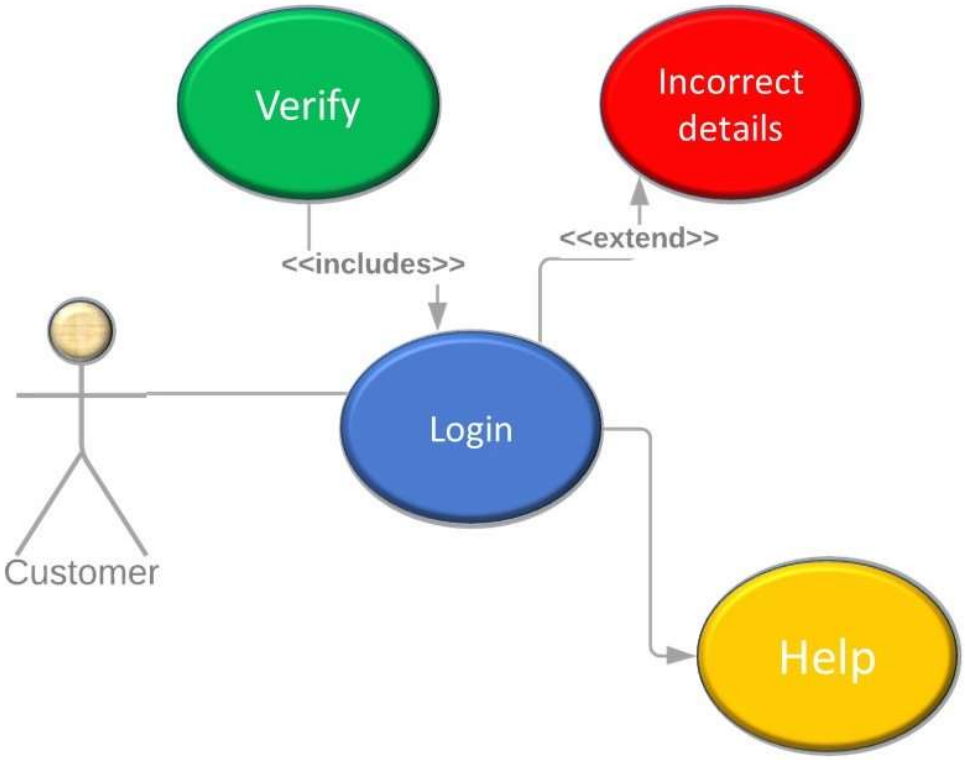
The use case diagrams represent the core parts of a system and the workflow between them. In the below diagram, the user upon opening the application will have two options:

To add a new expense

To view an already added expense.

If the user clicks the former option i.e to add a transaction, he has to next choose or create a category in which his expense was used.

Use-Case Diagram - Login



The above Use-case diagram shows the basic login functionality where the user enters the login address and the data is interpreted to verify along with an option of help if the user is unsure how to proceed with the App.

Use-Case Diagram - Add Transaction

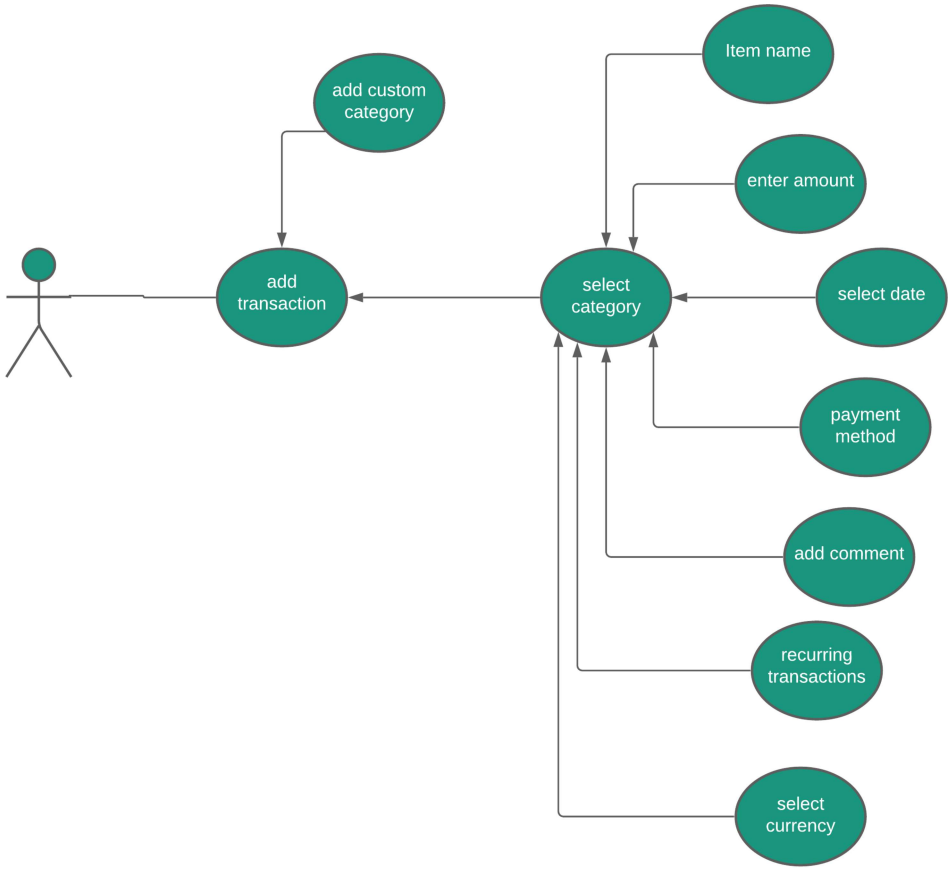


fig. 2 Use-case diagram shows that the user needs to select Add New Expense and fill in the necessary data such as Category, Item Name, Date, Amount, Payment method, Recurrence setting, comment, and currency type manually.

Use-Case Diagram - View Transaction

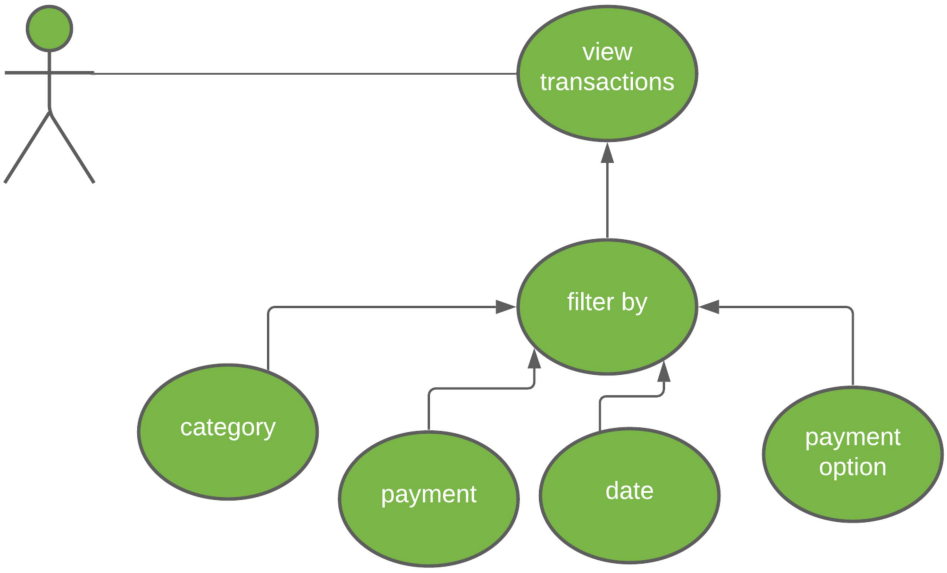
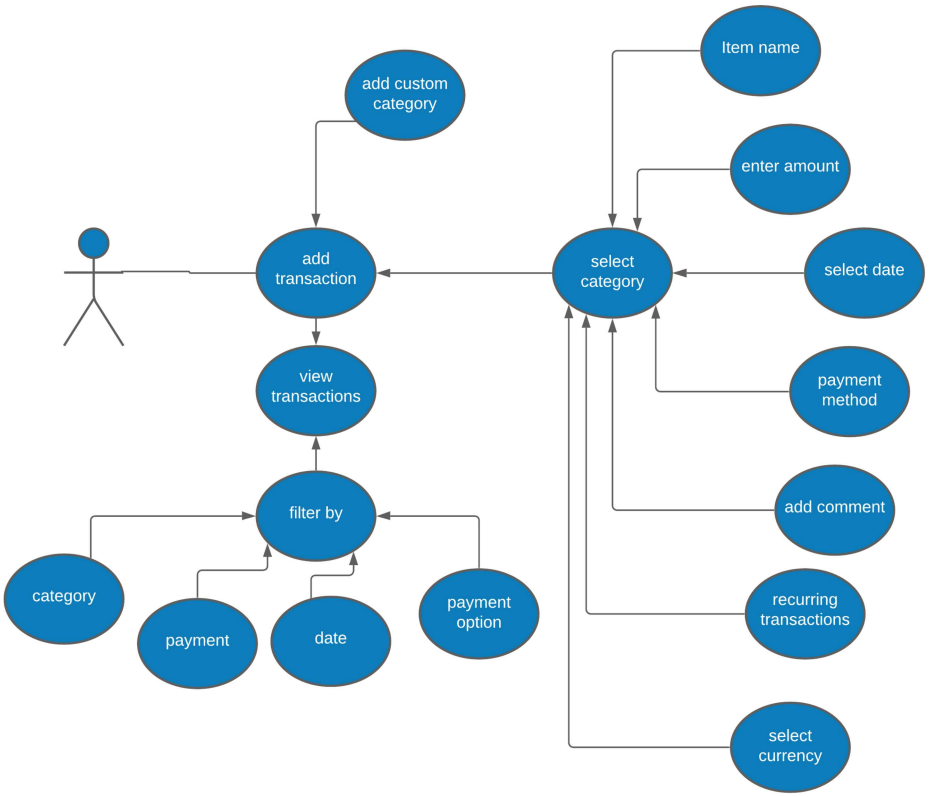


Fig. 3, Use-case diagram shows that the user can select View Transaction History and filter by providing the data like Category, Date, Amount, and Payment option.

Use-Case Diagram - Overview.



Lastly, fig. 4 - illustrates a combined - Use case diagram of fig. 2 and fig. 3.

Requirement Analysis

Understanding and structuring requirements play a key role in building a suitable output. During Requirements Analysis we identify all functional requirements (*WHAT should the system do?*) and all non-functional requirements (*How good should it do particular functions/tasks?*)
Input, processing, and output are being **key functions.** Team interactions, questionnaires, Document analysis observations, and prototyping is the **key methods.**

Once the requirements were gathered, we tried to identify and sort requirements from the perspective of a user's expectation, beginning with the most essential ones to the least.

In addition to that, we tried to achieve anything that could be added to the below categories. (*such as in this case double click instead of single click to exit the app, and Log out floating button having a pop up "are you sure?" to prevent mishap*). This was coupled with the existing BudgetWidget app to make the life of the user easier.

Following this, to analyse and assess we segregated the requirements into 3 categories

◆

Application Development(*essential*)
Application development included the Essential followed by Necessary and then Nice to have' prioritized accordingly in meeting the requirements.

◆

Diagrams and User stories (*necessary*)
Consists of drafting User stories and Class / Activity Diagrams of the Application.

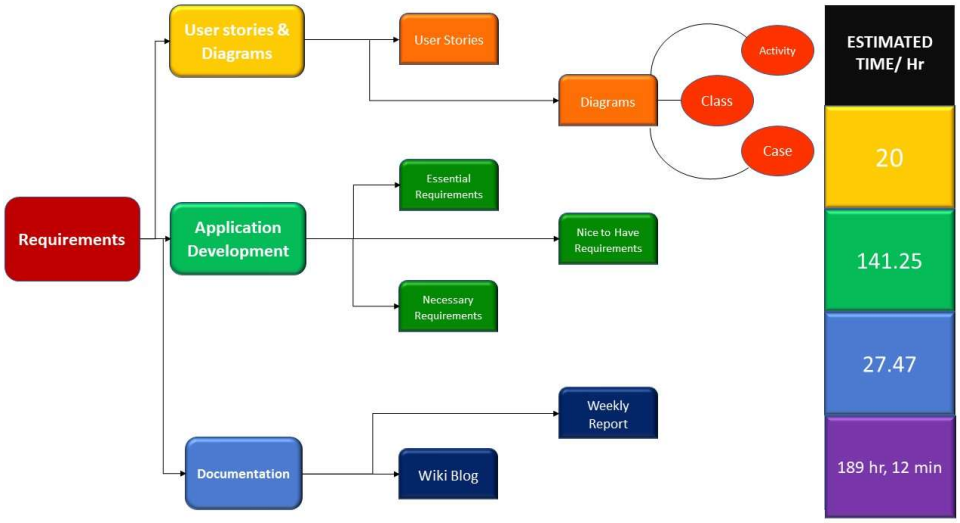
◆

Documentation (*desirable*)
Keep track of updates and progress, analyze accordingly to produce a weekly report, and write a wiki Blog to link between categories - Application Development and Diagrams\User Stories.

Estimated Time:

Particulars	**header**
Application development: 30 hours per week. [*Which is 30/7 (33 days)*]	141 hours 25 min

Particulars	header
Diagrams and user stories	20 hours
Documentation: [*2.5 hours per week (Weekly report) + 16 hours (wiki: Basic prototype)*]	27 hours 47 min
Total time	**189 hours 12 minutes**



Wrong assumptions

Fortunately, so far none of our team members made wrong assumptions that could lead to a critical or overall mishap which could have made us suffer in making major changes in return caused us not to have met the requirements set on time. Thanks to our initial approach in thorough development strategies and coordination. We were able to keep wrong assumptions to a minimum, although not eradicate them.

And Some of them are as follows,

The estimated time of work was assumed to be less but took overall 1.5 times more than the actual estimation in Application development and Documentation precisely. Upon analyzing we noted we didn't add system error/ interruption time in our calculation which we were compelled to do so when we had an overlap to re-estimate our time frame.

Direct commit changes to the main branch on GitLab were initially assumed which would have caused many issues. Therefore, we immediately rectified it by creating a sub-branch to make sure we review codes before pushing them to the main branch. This gave us room for correction and improvisation.

Weekly reports were initially assumed to have been uploaded in the wiki blog, upon clarifying, uploading reports in the Git repository and setting a link to it on the blog was enough.

History View of only the amount and purpose of expenses/ transaction validated was assumed to be enough but was later accommodated by mode of payment and the date.

Constraints

The entire App had to be designed in a way that we could accommodate all the requirements in the App Therefore constructing a layout/design accordingly to meet the requirements was our major constraint as none of the team members had experience in developing an Andriod App

Secondly, The emulator kept crashing and was unaware of how to troubleshoot the issue. after analyzing the issue was overcome by looking into Logcat, which directly takes you to the line of code where the issue is, and that finally solved it, Otherwise was constantly assuming it as of it to be a hardware issue.

And lastly, we faced some difficulty in understanding the difference between aggregation and composition and usage of them in the class diagrams and were not able to clearly understand the topic of different types of association and accompanied multiplicities. Nonetheless, these queries were cleared after attending the ISEE classes and also going through the descriptions, examples, tutorials, and reference of all types of Unified Modeling Language (UML): Use case diagrams, Class, Package, Component, Composite Structure diagrams, Deployments, Activities, Interactions, Profiles, etc. (uml-diagrams.org)

These were some of the constraints we faced during the process of developing our Basic Prototype Applications.

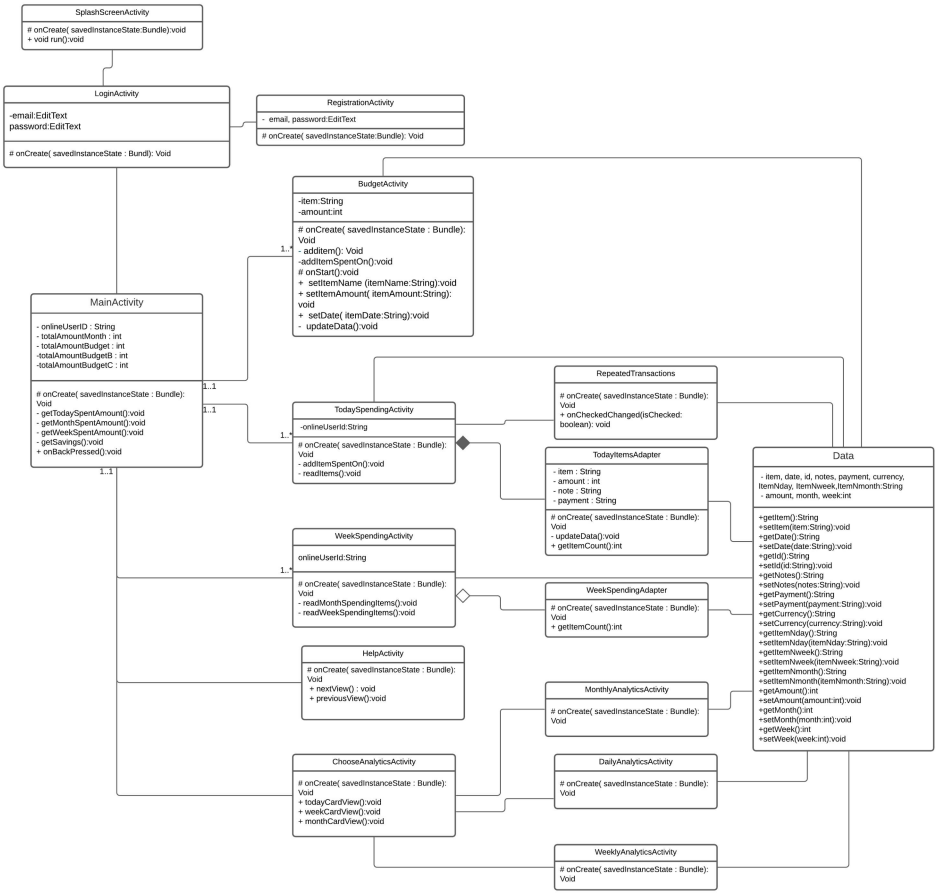
Class Diagram

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

Class diagrams are the best way to illustrate a system's structure in a detailed way.

They are based on the principles of object orientation and can be implemented in various phases of a project.

Class diagrams are a vital part of any software development project and they form the foundation of all software products.



ACTIVITY	DESCRIPTION
LoginActivity:	*This class provides the user with all the fields required for a user to login into the application.*
SplashScreenActivity:	*This class displays an animation of our app logo at the start of the application.*
RegistrationActivity:	*This class helps a new user to register so that he can log in to the app in the future.*

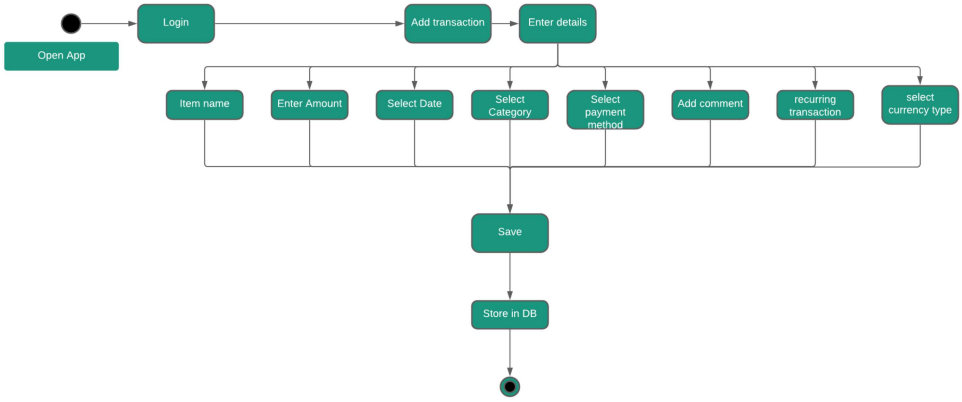
ACTIVITY	DESCRIPTION
MainActivity:	*This class creates a user interface where the user can view various features available with the application. This class also creates a dialogue box where the user can view a brief overview of the transactions.*
BudgetActivity:	*This class helps the user to add different types of items and the amount of money the user is planning to spend on that particular item. The user can also update and delete the item.*
TodaySpendingActivity:	*This class helps users to add different types of transactions with details like item name, mode of payment, currency type, and notes.*
TodayItemsAdapter:	*This class helps in retrieving the data from the database and displaying the data on the Today spent view.*
RepeatedTransactions:	*This class helps the user to add a recurring transaction. So that the transaction repeats itself after the said period of time.*
WeekSpendingActivity:	*This class displays all the transactions done by the user in that week.*
WeekSpendingAdapter:	*This class helps in retrieving the data from the database and displaying the data on the week spent view.*
ChooseAnalyticsActivity:	*This class presents the user the amount of money he has spent on various activities during a day, week, and month.*
DailyAnalyticsActivity:	*This class provides the user with information about his daily spending with the help of graphical representation.*
MonthlyAnalyticsActivity:	*This class provides the user with information about his monthly spending with the help of graphical representation.*
WeeklyAnalyticsActivity:	*This class provides the user with information about his weekly spending with the help of graphical representation.*
HelpActivity:	*This class helps the user to understand the working of the various features of the application using images.*
Data:	*This class obtains all the data entered by the user in various places and stores it in the database.*

Activity Diagram

An activity diagram is an important behavioral diagram in the UML diagram to describe dynamic aspects of the system. An activity diagram is essentially an advanced version of a flow chart modeling the flow from one activity to another.

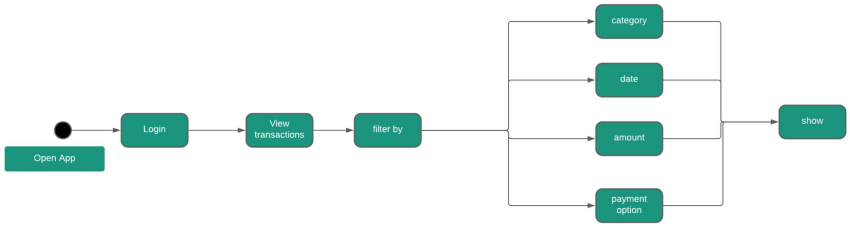
Activity Diagrams describe how activities are coordinated to provide a service that can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve several different things that require coordination, or how the events in a single-use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use case coordinates to represent the workflows, therefore describing the steps performed in a use case diagram.

Add Transaction Activity



The Add-in transaction activity user needs to follow, this is in reference to fig.2 of the Use case diagram.

View Transaction Activity



The View transaction activity user needs to follow, this is in reference to fig. 3 of the Use Case diagram.

Activity - Overview

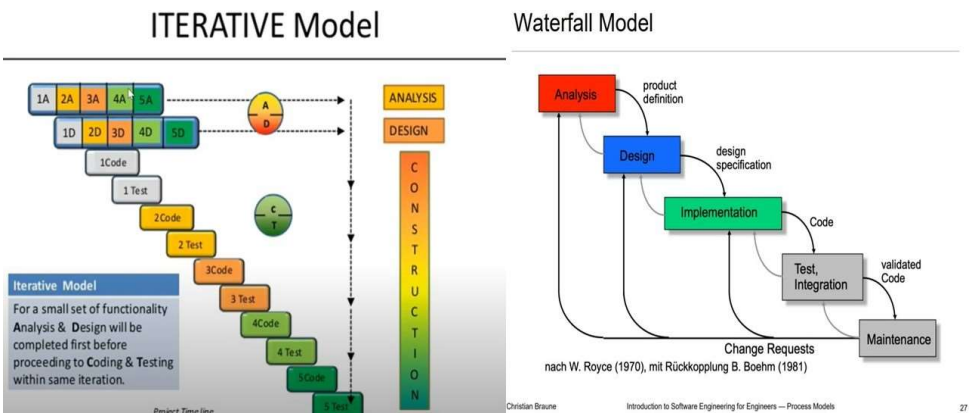
A combined Overview of the two Activities diagram from above and this is in reference to fig. 3 of Use case diagram.

Development Strategy - Managing and Monitoring Work.

Approach

Soon as the team gathered requirements, we distributed tasks among team members according to categories of requirements which we segregated previously. Our approach was to analyze and ensure all essential requirements of the App are completed first but also simultaneously start analyzing necessary requirements and further nice to have requirements.

On the other hand, work parallel on documents progress (*weekly report*) and collect data to analyze and design User stories and Diagrams.
Therefore used a combination of the Waterfall Model and the Agile model: Iterative and Incremental (*features keep on adding*) to achieve our requirements for the basic prototype.



> *On the right you can see analyses design and construction, On the left, you can see 1A 2A, .. 5A these are the analysis we need to do as part of the project. And 1D, 2D, .. 5D designs for the following analysis*

What is an *Iterative Model*?

The iterative model as opposed to the waterfall model where you complete 100% analysis before moving to design and completing 100% design before moving to Coding. In the Iterative model: we split the requirements into various categories and assign them respectively.

For example, a part of the project requirement is split into 1,2,3,4,5 parts (*from the figure above*) where 1A is analysing part 1 of the project requirement and move the part that is analysed i.e. 1A to Design – 1D and simultaneously start analysis part of the project requirement 2A while designing 1D, so therefore from the fig above, when the 1A,1D part it is pushed to coding and then testing. The 4th part of the analysis (4A) and the 3rd part of the design (3D) is already been completed when the 1st part of the coding done. This has a huge impact on the project timeline. Thus, we implemented this in meeting the Application Development requirement and Documentation

For the User Stories < Diagrams > and blog,

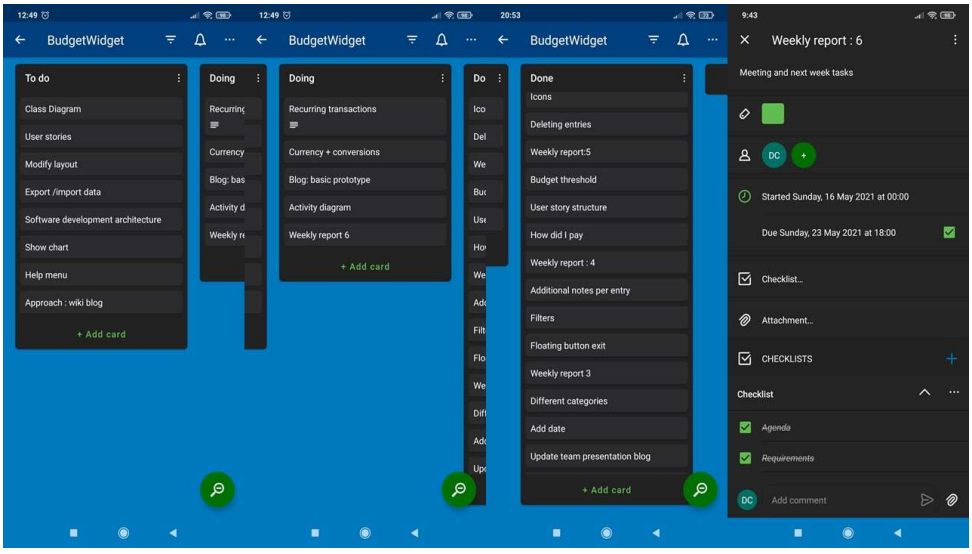
We implemented the *Waterfall model*: were all the tasks are analysed completely and designed and structured accordingly.

Managing and Monitoring

Successful monitoring of the progress needs efficient segregation of requirements to be assigned to the team members and we used the Trello app and labeled 3 categories

🔴 To-Do,
〽️ Doing,
✅ Done.

were given to know the status of the progress accordingly.



Once a task is created in the To-do columns then transferred to the doing column once one starts working on the task. At last, after the task is done, it is reviewed and transferred to the Done column. All these requirements and issues are categorized into the three mentioned columns that are present and their due dates assigned to each of us.

Following,
Abhirup and Anjan took the responsibility of being the Product owner and the SCRUM master.

User stories, Class, and Activity diagrams were assigned to Harsha and Sreekar.

Documenting: weekly reports and blog updates were assigned to Denzil.

Responsibilities and related developments for each member were decided and worked upon weekly. We set weekly targets and outlined them in our weekly report to have a progressive outcome and environment amongst ourselves.

As the development progressed, issues/ constraints were identified and channeled to team members to analyze, sort, and resolve with a systematic deadline. This way the every team member was aware of the development and its issues arising in all sectors.

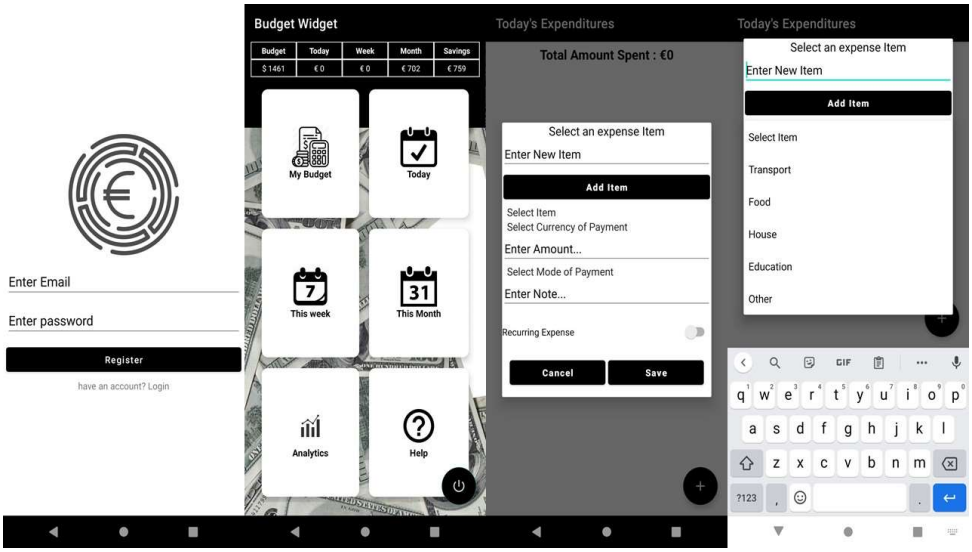
Weekly meeting: A healthy and reasonable - two meetings were scheduled per week. One on every Monday at 18:00 (or 16:00 in case someone is not able to make it @ 18:00). which helped us have an overview of completed tasks from the previous week, record it and review for improvements. Another meeting was set on Wednesday after the tutorial session @ 11:30 to plan next week's tasks accordingly.

Communicating with each other was a big challenge due to the current pandemic crisis, but we chose to work our way through different communication platforms. Messengers, online Zoom meeting rooms, using screen sharing helped us track each other's work effectively. Every change to the project was informed to all the teammates during the first meeting, but when in urgency – through WhatsApp.

Overall these strategies along with managing the optimum use of our resources helped us overcome every challenge/ obstacle that we faced and forecasted.

App preview

Our final content, App preview to show an overview of the app experience, focusing on the app's core features and content aiming to tell a cohesive story that gives users a sense of the journey they'll experience when using our App.

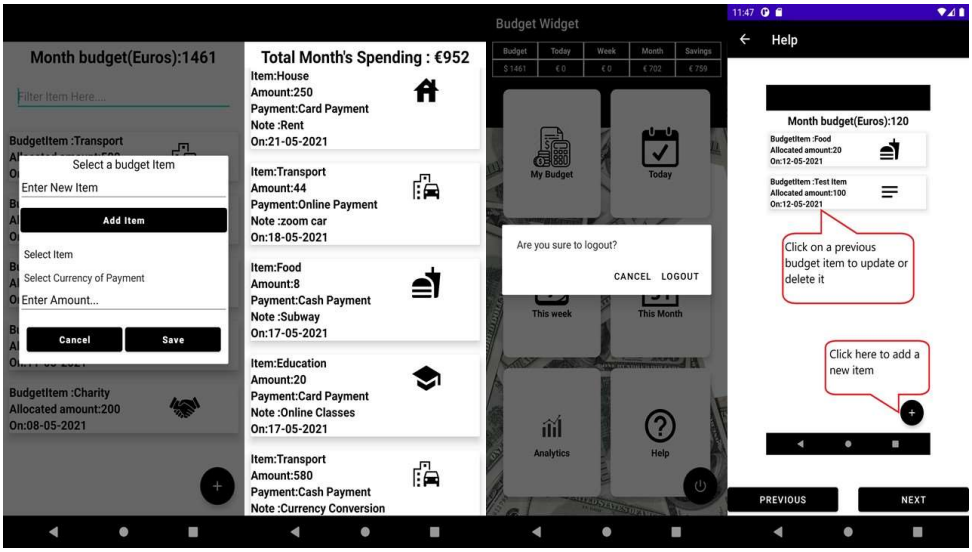


When the user opens the application, they are presented with a register page, (fig 1.1) where the user can register to access their account, if the user has already registered they can simply click the login function below to enter.

Once the user logins the app takes them to the main page, (fig 1.2) where several features are shown such as My Budget, Today, This week, etc. along with and a quick overview on the tab above describing the amount that is been spent with a margin titled budget.

On clicking Today, the user can add expenses (fig 1.3) quoting item, currency, amount, mode of payment along with a not an option to add it as a recurring expense. Once clicked the data will be stored in the expense which could be edited or deleted further.

Under Select expense item, some useful/common items are already made available (fig 1.4) for the user to choose/pick. Making it even more convenient for our users.



The user can also choose to have a monthly budget by clicking on My budget and selecting or adding an item, currency, and amount (fig 2.1) and save it. Which is reflected on the main page overhead as well.

Similar to a monthly budget, an overview of Monthly's spending can also be viewed (fig 2.2) where the user can access it by clicking This month from the main page. Where the user can also modify or delete items shown/added. > which is similar to the "This week" function from on the main page.

It's crucial for every app to have support /guide and therefore our help menu (fig 2.3) gives our user a good guide to make the best use of our application.

Lastly, we all know how annoying it can get when you accidentally or a part of your hand touches the log-out button and the app closes. You'd have to open the app and go through the log-in functionality over again and every time it happens. Thus, we made a small pop-up appear when pressed the log out floating button (fig 2.4) - where the user can confirm to log out or cancel if not.

Visual Glance

These days every movie, product, service is projected with a trial, trailer, advertisement, or a preview video to get the user a glance of the product/ a visual assurance per se. Thus, we finally present to you our basic prototype app in the short clip below.

0:00 / 2:46

App Preview

Hope you had a nice time reading through our blog. We like to continue to keep our readers interested, so do visit us back for our upcoming blog on Advanced Prototype.

Home Page