**SW Engineering CSC 648/848**

**Section 04 Team 01**

**KeepUp App**

**Team List:**

|  |  |  |
| --- | --- | --- |
| **Hruthika** |  | **Team Lead** |
| **Hansley** |  | **Backend Lead** |
| **Carlos** |  | **Frontend Lead, GIT master** |
| **Iza** |  | **Scrum Master** |
| **Malavya** |  | **Member** |
| **Christian** |  | **Member** |
| **Tay** |  | **Member** |

**Milestone 2 Document**

**Revision History Table**

|  |  |  |
| --- | --- | --- |
| Revision ID | Revision Date | Revised By |
|  |  |  |
|  |  |  |
|  |  |  |

**1.Data Definitions V2:**

Revision from Milestone 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Primary Data Name** | **Data Name** | **Data Definition** | **Use** | **Comments** |
| Registered User | Username | Can be email id/ profile name | needed to signup / login | has to be unique for each user |
|  | Password | A combination of letters, numbers and special characters | needed to signup / login | must be strong |
| Notepad | Notes title | The name given to the respective notes | To give your notes a title for easy readability | can be checked, updated, deleted |
|  | Content | The data of the notes added | To store the notes taken | can be checked, updated, deleted |
| Expense Tracker | Income | The income amount and other related details | To store the income details added | can be checked, deleted |
|  | Expenses | The expense amount and other related details | To store the expense details added | can be checked, deleted |

**2.Functional Requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Priority Level** | **Feature** | **Description** |
| 1 | 1 | User Registration | New users should be able to register |
| 2 | 1 | User Login | Existing users should be able to log in |
| 3 | 1 | Add Notes | Registered users should be able to create new notes |
| 4 | 1 | Update Notes | Registered users should be able to update notes |
| 5 | 1 | Delete Notes | Registered users should be able to delete notes |
| 6 | 1 | View all notes | Registered users should be able to view all notes |
| 7 | 1 | Add Income | Registered users should be able to add new income |
| 8 | 1 | Delete Income | Registered users should be able to delete added income |
| 9 | 1 | View Incomes | Registered users should be able to view all incomes |
| 10 | 1 | Add expense | Registered users should be able to add new expense |
| 11 | 1 | Delete Expense | Registered users should be able to delete added expense |
| 12 | 1 | View Expense | Registered users should be able to view all expenses |
| 13 | 1 | Income-Expense dashboard | A income vs expense graph shall be displayed based on user input and also total balance should be displayed |
| 14 | 1 | Relaxer - Breather | Visual breathing aid with inhale exhale instructions |
| 15 | 1 | Relaxer - Audio | Relaxing audio clips and meditation instructions |
| 16 | 2 | Relaxer - Exercises | Muscle relaxation exercises |
| 17 | 3 | Relaxer - Notification | Users must be notified to take time to relax and use the relaxer feature |
| 18 | 3 | User App Notification | Any updates / changes to the app shall be notified to users via email |

**3.UI Mockups and UX Flow:**

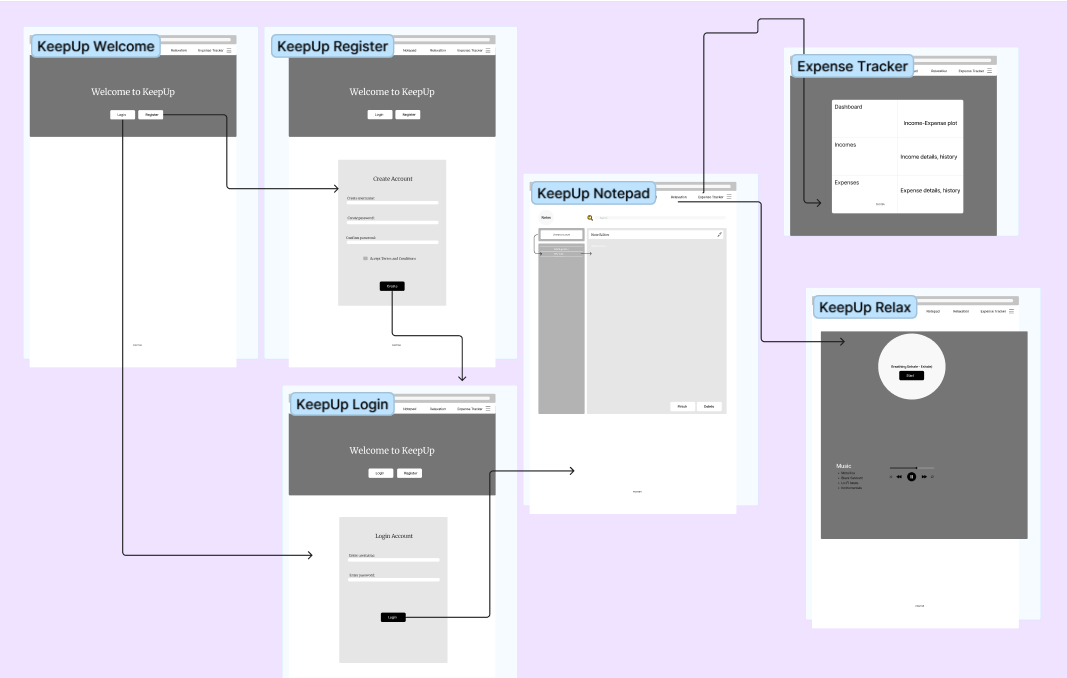
****

Fig 1 a: Wire Image

Summary of Team Discussions:

After through discussions on the priority of user stories, we have collectively come to a conclusion that the features and functionalities of the KeepUp app are going to be as depicted in Fig 1. The user has to mandatorily sign up in order to access the app and once he logs in, he could use all the 3 features and every functionality of these features.

Figma link: [KeepUp App – Figma](https://www.figma.com/file/yEnhBTVElJPDqwQYo6wrEs/KeepUp-App?node-id=0-1&t=Qn1iZ57Ca8hsJhuk-0)

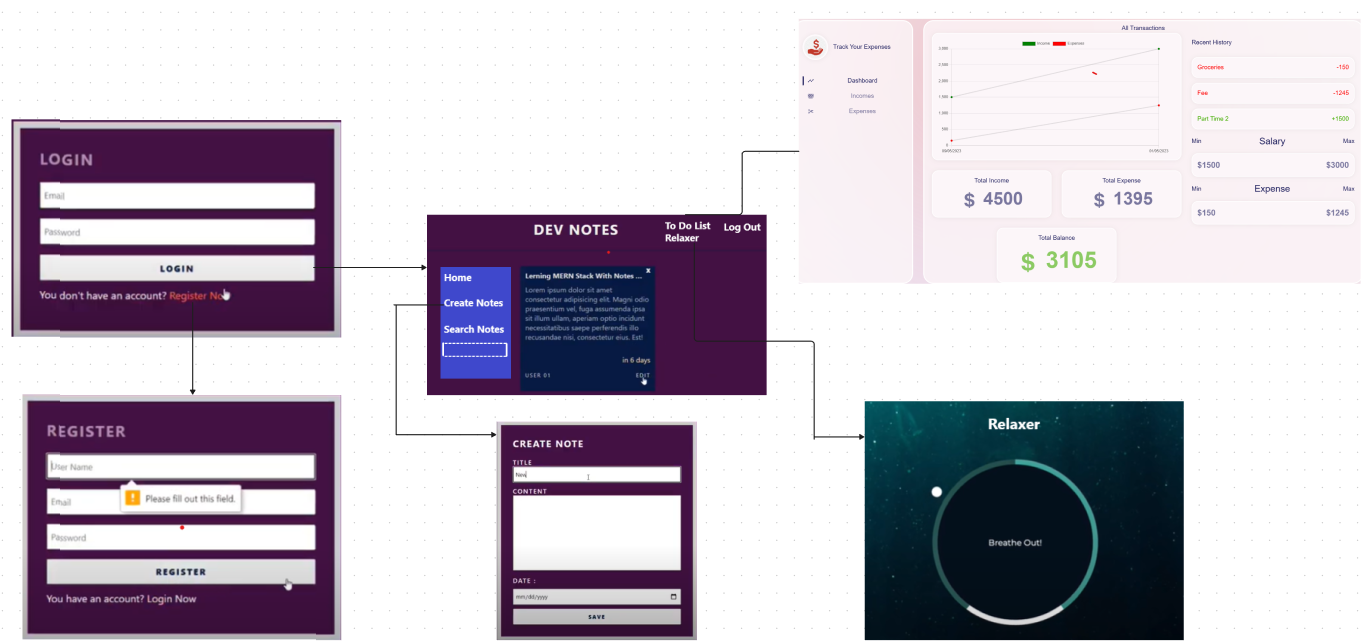


Fig 1b: Prototype Image

**4.High level Architecture, Database Organization:**

Users: Notepad: Expense Tracker:

Email\_id title Income

password content Expense

**Add/ Delete/ Search Architecture:**

|  |  |
| --- | --- |
| Add/ Delete for Notepad | When users wants to add or remove or update notes |
| Search for Notepad | When users wants to search for a particular notes with its title |
| Add/ Delete for Expense Tracker | When users wants to add or remove an income/expense added |

Technical feasibility of DB operations:

After thorough planning and discussions, we collectively concluded that the above mentioned DB operations are the most feasible to implement. The only feature that may need a search feature is the Notepad. We will be storing the titles of all notes the user enters in the DB along with its contents and hence, once the user searches for a titled notes, the closest search results will be displayed and then the contents of the notes can be fetched from there.

*Note: As we will be using MongoDB and the data is stored as a collection, this makes it much easier to perform the CURD operations with simple query operators.*

API’S: Our backend technology is Express JS. So, we shall be making API calls using Axios.

**5.High Level UML Diagrams:**

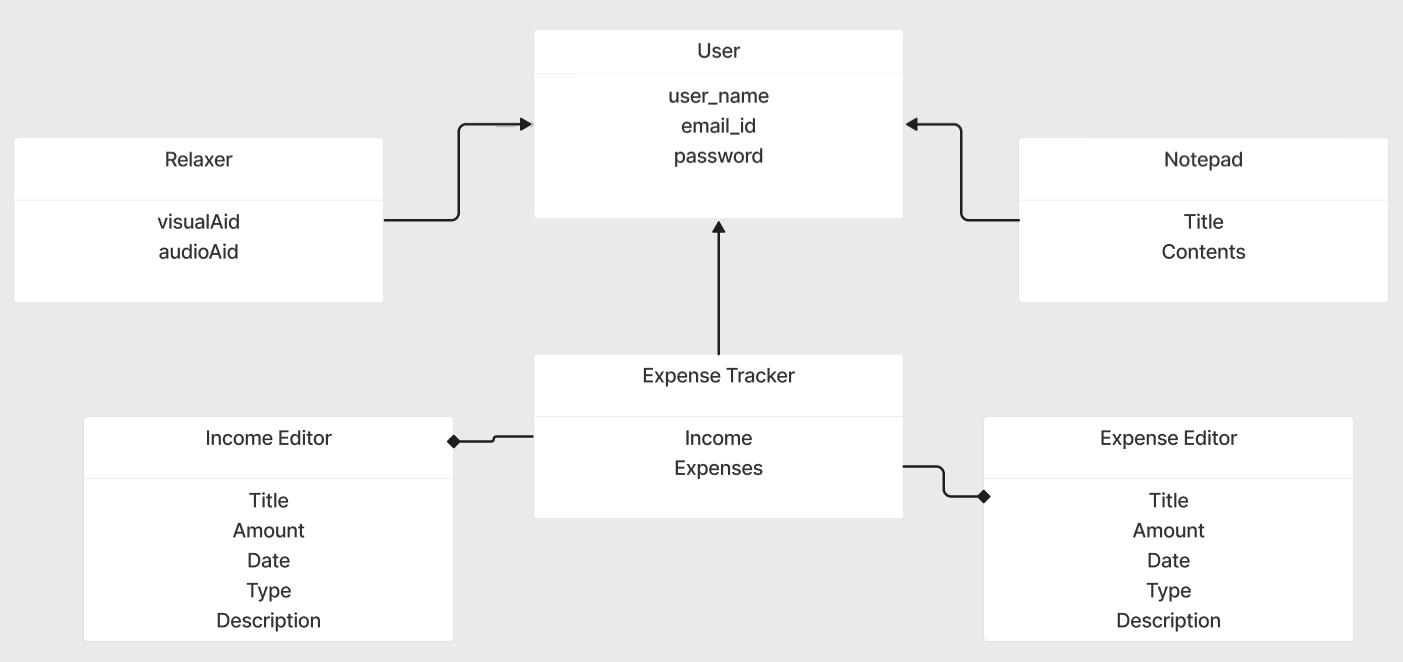
****

Fig 2: UML Class Diagram

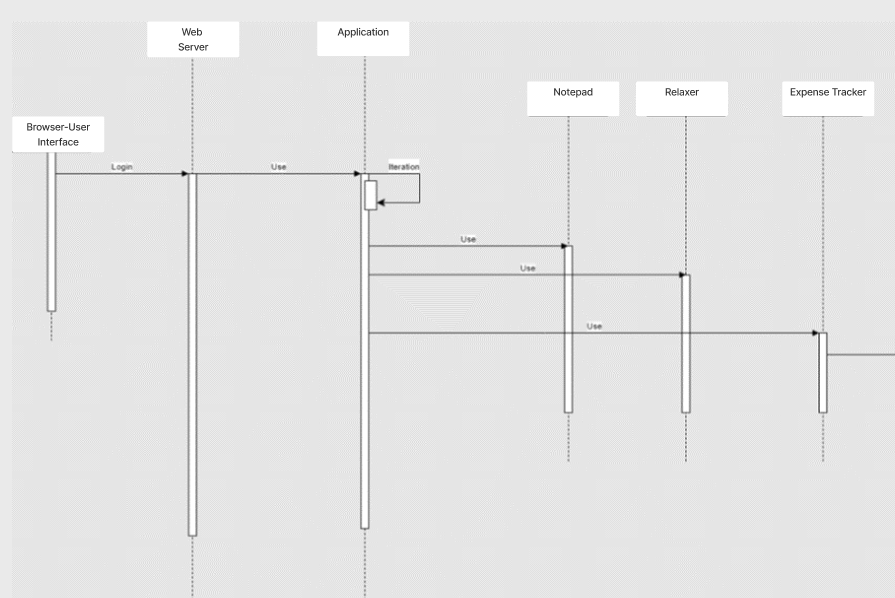
****

Fig 3: High Level Sequence Diagram

**6.Key Risks:**

**Skill Risks & Mitigation Plan:**

>Many of us are unfamiliar or unskilled in some of the technologies utilized in this project - noticeably towards backend development.

>This will gradually be alleviated as we learn and apply these technologies as the project

develops.

>In cases where learning these technologies has proven more difficult than anticipated,

we have many resources to turn to such as online videos, tutorials, and of

course fellow team members.

>On the off chance we cannot grasp and fully utilize a technology in time, there exists

The possibility to retroactively change it due to our agile workflow. Though this is only a

last resort given our time constraints.

>In the development of this project, team members may struggle with implementing certain features or debugging code.

>Members should refer to the documentation or reference thoroughly and remember

That the roles assigned aren’t strictly rigid; asking other members never hurts.

>In case the issue is still not resolved, a meeting should be scheduled to help in its

resolution.

**Schedule Risks:**

>Our schedules differ greatly, which may cause missing a key meeting or event.

>In which case, those who have missed a meeting may be briefed by fellow team

Members on what they missed or refer to the scrum master’s documentation.

>If scheduling conflicts become too invasive, an alternative solution would be to

Arrange two different meetings: one for the frontend team and one for the backend.

However, this is a last resort as the added overhead would only dampen the team’s

workflow.

>A team member may fall behind on their task, leaving us an unfinished product with

which other members are forced to fill in the gaps or present an unfinished - but likely to

be more coherent - product.

>Ensure that the schedule is easily adjustable and that the assigned task can be picked up

by another member.

>If a member believes they are falling behind or are unable to deliver their part of the

project on time, they should communicate that sentiment as soon as possible so other

members may then be able to handle their work more readily.

**Teamwork Risks:**

**>**As the project progresses, burnout may ensue, delaying progress, quality, and overall morale for the team.

>Frequent communication, meetings, and simply knowing what our assigned tasks are

and where to find resources to help with those tasks alleviates this issue.

>We all have different coding styles which may cause confusion when having to merge and integrate together into a whole application.

>All members should adhere to a certain coding style for all work conducted on this

project. Coding style documentation should be written and distributed as soon as the

coding phase begins to resolve this issue. Testing and code reviews should be conducted

to enforce this policy.

>Frontend and backend teams may run into conflicts as two different ‘subteams’ must learn to integrate and work coherently.

>Frequent communication and adhering to a certain coding style lightens the gravidity of

this issue.

>Individual team members may find other members difficult to work with, leading to negative sentiments beyond the scope of this project.

>All members should keep in mind that they’re all in the same boat, so when one member

sinks, all members may sink. Knowing this, the work environment should be kept

respectable even if such sentiments arise.

**7.Project Management:**

Apart from the meetings on Wednesdays in class, our team is having scrum meetings on Mondays. The meeting logs are being maintained by the scrum master. At the beginning of every Milestone, tasks are being assigned to each team member uniformly and an excel sheet with deadline for each task is being shared with the team. In the scrum meetings, every team member would be sharing their progress. The difficulties faced can be shared with the team not only during the team meetings but anytime over our team discord channel. This helps maintaining a healthy progress with the milestone while also eliminating any scope for miscommunication within the team.

Our team consists of 2 front end developers, 2 backend developers, 1 members working on the Data Base and 2 exclusively on the server maintenance. Apart from this categorization, each member also works with the other internal teams to coordinate for smooth flow of development.

For proper maintenance of repository, master branch and develop branches were created and it was made mandatory to create a pull request in order to merge to the master, else individuals can not directly upload files to master. This way, the main working code in master will not be disrupted by faulty one.