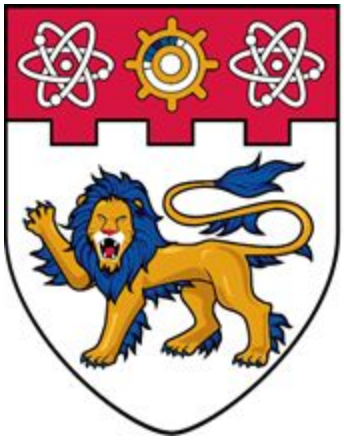


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ3002 Advanced Software Engineering Project:

System Requirements Specification - CashTrack

Name:

Ravishankar Amrita (U1822377F)

Datta Anusha (U1822948G)

Kumar Mehul (U1822146E)

Alex Leong (U1921599D)

Nicklaus Tan (1403385F)

Elliott Ong (U1922981C)

Daniel Loe (U1921408A)

S Sri Kalki (U1921575L)

Class: B2

School of Computer Science and Engineering (SCSE)

Revision History

Author(s)	Date	Description of Changes	Version #
Daniel Loe S Sri Kalki Elliott Ong Nicklaus Tan Alex Leong Datta Anusha Kumar Mehul Ravishankar Amrita	3 February 2021	Preliminary Version	1.0
Datta Anusha Kumar Mehul Ravishankar Amrita	13 February 2021	Revising changes for entire documentation	1.1
Daniel Loe S Sri Kalki Elliott Ong Nicklaus Tan Alex Leong Datta Anusha Kumar Mehul Ravishankar Amrita	14 February 2021	Finalized Version	1.2

1 Table of Contents

2 Problem Statement

The past few years have seen an exponential growth of e-commerce, digital payments (including contactless), instant payments, and cash displacement. As exciting as this paradigm shift is, it has introduced its own set of drawbacks. Since most of our payments have taken a digital turn, it is only intuitive to expect a digital solution to splitting bills which integrates seamlessly with this payments ecosystem.

Runtime Terror conducted an extensive survey on Singaporeans in the age groups 18 - 55. We identified the great effort & inconvenience while settling shared bills, the potential of data analysis on spending patterns, and the difficulties in keeping record of your own expenses. Our survey shows that 66.7% [B.1.1] of Singaporeans almost never volunteer to pick the check for the table in a group setting. The reason being sighted by most is either that they find splitting bills a cumbersome process or that they are unsure about whether people will pay their share eventually [B.1.2]. Furthermore, a staggering 77.1% of individuals say that they are almost always put into an awkward situation as they do not wish to ask someone for the money they are owed [B.1.3]. We aim to address all of these issues through our proposed application CashTrack.

CashTrack is a web-based expense tracker that acts as a one stop destination to track personal expenses, ease the process of resolving shared bills and view comprehensive insights into your spending patterns.

3 Overview

3.1 Background

Studies show that the ongoing shifts toward e-commerce, digital payments (including contactless), instant payments, and cash displacement have all been significantly boosted in the past six months. [B.2.1] This significant boost has been motivated by the existence of a global pandemic encouraging individuals to either use contactless payment or digital payment. According to data gathered by AksjeBloggen, the global digital payments market is expected to continue rising in the following years, reaching $6.7 trillion value by 2023. According to the same research, more than 6.1 billion people will use digital payments by 2023. [B.2.2] As faster and real time payments are gaining stream, the need for a seamless expense lifecycle is growing exponentially as well.

Our survey identified the great effort & inconvenience while settling shared bills, potential of data analysis on spending patterns, and difficulties in keeping record of your own expenses. Subsequently, when our diverse group of survey participants were asked whether they would make use of an expense tracker application, that enables an effortless and efficient process for splitting bills and setting reminders for individuals who owe them money, 85% of individuals voted 7 or above on how likely they would be to use such an application. [B.1.4]

Thus, the Runtime Terror team aims to develop CashTrack, an expense tracker application, to ease the process of splitting bills amongst friends, tracking the status of the payments owed and more.

3.2 Overall Description

CashTrack provides an integrated platform solution that enables users to collectively record, analyse and settle their monetary transactions between each other. The following are the primary features of this application:

1. **Personal Expense Tracking**

   CashTrack allows tracking of an individual's expenditure, more specifically how much they owe and are owed by other people, as well as their own expenditure such as food, travel, etc. Users are able to add personal expense records by navigating to the 'Personal Expenses' page. They can categorise these personal expense records by using category 'tags' such as food and travel.

2. **Shared Expense Tracking**

   As users expand their social circle, frequent get-togethers are quite likely, in these scenarios, splitting the bill can become a frequent event. The application thus allows splitting a bill, tracking the payment status for the bill, while actively monitoring and notifying the parties involved. Users are able to add expense records with friends and family, for bills where either the entire payment was made by the user or the user lent them money. Users adding the record can choose the category of the record, use the automatic split by ratio/percentage/shares option, and also choose to add some comments for everyone to see.

3. **Insights into Personal and Shared Expense Spending Patterns**

   The application performs a comprehensive analysis of users' expense data over time or by categories. This aims to reveal any underlying spending patterns or interesting insights into the user's payment trends. Further, it helps the user to effectively visualise their transactions histories for all both personal and shared expenses.

   Finally, the application also provides a chat feature to increase user engagement as well as to clarify doubts regarding shared expense transactions.

4 Investigation & Analysis Methodology

4.1 System Investigation

As our web application is a relatively novel concept in the market, our investigation will primarily focus on the functions / features the users require and desire within our application, how these features can be implemented and delivering a quality user experience at par with the users' expectations of the application.

4.2 Analysis Methodology

Analysis methodology encompasses business analysis, requirement analysis, data analysis, process analysis and CashTrack's application architecture:

- **Business Analysis:** The budget and sponsorship requirements are detailed in the project proposal.
- **Requirement Analysis:** Users will be able to access the application using any devices' (Personal Computers, Smartphones) web browser that has internet connectivity.
- **Data Analysis:** All the user's data (such as expense records, account details) is stored in the remote cloud database. The application can retrieve user's data to perform data analytics to provide the user with insights about their spending patterns and transaction trends.
- **Process Analysis:** This can be derived from system interfaces, individual process decomposition and data/process flow analysis.
- **Application Architecture:** The application user interface is designed to be as intuitive as possible to provide a smooth user experience, enabling users to make use of the application functionalities, effortlessly.

4.2.1 Feasibility study and requirement elicitation

1. Forming a team of talented individuals with experience and knowledge in full stack web development, cloud data services and data security (Front-End & Back-End Engineers).
2. Conduct interviews and surveys with identified potential users to get opinions and feedback on features that will be included in the web application.
3. A Feasibility and Risk Assessment study will be conducted to select solutions that are the most viable and feasible based on the results of the interviews.

The interviews and feedback are crucial in determining the current and future system requirements.

4.2.2 System analysis and requirements specification

4.2.2.1 Performance and analysis of problem using Unified Modelling Language (UML)

A UML diagram to represent a comprehensive overview of the CashTrack system's model. The system consists of different levels and components when users are acting as a Payer / Payee for grouped expense records or as regular users creating their own personal expense records and viewing their expense data analytics. The system consists of the following features:

- Users will be able to create personal expense records with category tagging.
- Users will be able to update personal expense records.
- Users will be able to delete personal expense records.
- Users will be able to set a personal expense limit with an optional reminder when approaching the limit.
- Users (Payee) will be able to create expense records with one or more individuals or with a group.
- Users (Payee) will be able to update these expense records.
- Users (Payee) will be able to delete these expense records.
- Users (Payee) will be able to add comments for these expense records.
- Users (Payee) will be able to acknowledge payment of these expense records by the payers after it has been received.
- Users (Payer) will be able to upload image files as proof of payment for an individual or group expense record.
- Users (Payer) will be able to add comments for these expense records.
- Users (Payer) will be able to notify that a payment for this expense record has been made.
- Users will be able to chat with other users.
- Users will be able to view data analytics of their personal and group expense records.

UML Use Case graphical representation of systems functions:

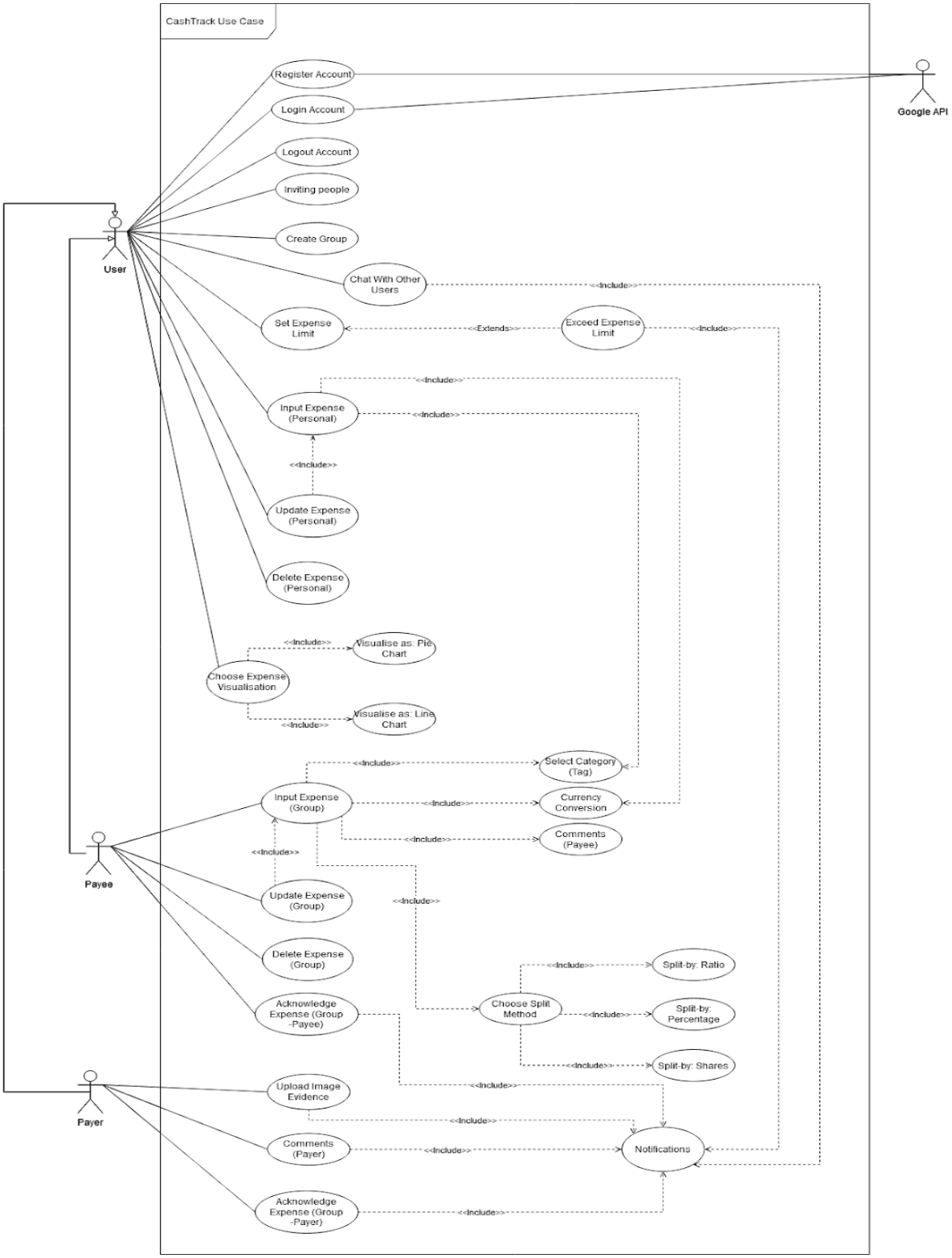


Figure : Use Case Diagram

4.2.3 Prototyping

Initial low fidelity prototypes will be generated to showcase a limited and functional prototype for the expense tracker system. This prototype will be a working example to facilitate the ideation during the design phase of the product life cycle. [A.1]

Following the relevant product design processes, these Lo-Fi prototypes will be refined to generate WireFrame design prototypes including interface with the cloud database system, which will serve to guide the implementation process of our application. [A.2]

This shall finally lead to the development of the first version Minimum Viable Product (MVP) prototype, which will be iteratively improved based on quality benchmarks set for this product. [A.3]

5 Constraints

5.1 Scalability

Our application, CashTrack, is built with an emphasis on enabling scalability, which has been accounted for and enforced by the following means:

1. **Scalable Development Tools: MEAN Stack**
   As the application load increases (number of users, number of expense records, data analytics increase), the application should adjust itself to handle the load accordingly. The
   choice of MEAN Stack ensures streamlined development to build scalable applications optimised for cloud deployment.

- MongoDB can scale out horizontally via single large Replica Sets using one Primary and two Secondaries with heartbeat communication for up/down state with replication occurring to the secondaries via the oplog. It provides benefits like auto–sharding, replication and high availability, thus, it is easy and reliable to scale.
- Express.js handles requests, endpoints, cookies, handling routes, error handling and easy templating. Thus, allowing the application to increase in functionality effortlessly.
- Angular10 provides templating, data-binding, routing, and clean, modular and reusable architecture and security, Thereby, further allowing extension of application functionality.
- Node.js has a rich ecosystem of open source libraries and components, including an npm (Node.js package manager). Being a JavaScript-based runtime built on Chrome's V8 engine, Node.js compiles the JavaScript source code to native machine code before execution, which allows for building scalable and performant web apps.

2. **Development Practices for Maintainability:**

   As the complexity of our application grows, in terms of data, features and user traffic load, the application should be able to consistently enforce maintainability. Maintainability allows the software system or its components to be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. This would allow onboarding developers to learn and maintain the system easily and efficiently.

- The application is being developed keeping in mind the SOLID principles. Each class or module within the application will implement only a single feature/functionality of the

application, thereby implementing the **Single Responsibility Principle**. Each class or module is being written in such a way that it can be extended to implement additional functionality, thus adhering to the **Open-Closed Principle**. While structuring the application code, **Liskov Substitution Principle** has been kept in mind, as well. Furthermore, each interface provides only specific functionality, thereby sticking to the **Interface Segregation Principle**. To work according to the **Dependency Inversion Principle**, the dependencies between the code modules are going to be limited.

- The above practices will allow our application to have low-coupling and high cohesion. Low-coupling combined with modularity, provides the flexibility required to add new features to the application, in future, with less difficulty. It will improve code readability and will reduce the risk of breaking existing functionality when new features are implemented.

5.2 Proprietary hardware and software

Client-side requirements:

Client side requirements include standard Desktops and open-source software browsers such as Google Chrome. There are no special hardware or software requirements required of the user's local machine, to successfully run and access the application platform.

Server-side requirements:

Server side requirements for the application include remote backend server technologies like Google Cloud Platform (which uses Google Kubernetes) and MongoDB cloud data hosting. This consists of the proprietary hardware and software required to ensure application reliability and scalability.

5.3 Batch updates vs. (close) Real-time updates

CashTrack provides real time updates for chat messages, application notifications and all expense records created by the user.

However, the application is constrained to reflect shared expense record details in the form of batch updates. Similarly, the insights provided by the application on spending patterns will be in the form of batch updates as well.

These select batch updates are invoked by the client whenever the application browser's tab is refreshed.

5.4 Project Schedule

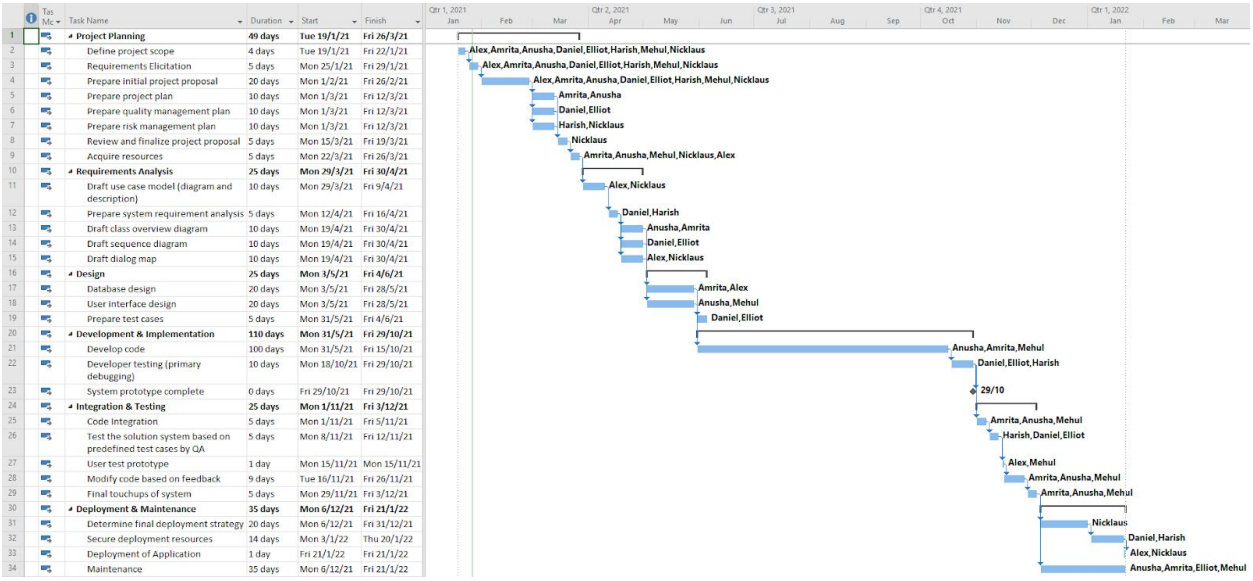
The CashTrack project is scheduled to be implemented and deployed within one year. Over the course of this development period, intermediary prototype launches have been scheduled before the final product is released, with the objective of receiving user feedback and beta testing results.

The first prototype is scheduled to release after 3 months, which will implement basic functionalities, interface and user interactions of the application.

The second prototype is scheduled to release after 9 months, which will include all functionalities implemented and tested in detail. The beta testing for this prototype will span the following 2 months to acquire quality feedback to refine the application.

Following the above process of extensive testing and feature refinement, the project will then proceed to the deployment phase, which is scheduled for the 11th month of the project. This will conclude with complete deployment and release application at the end of the final 12th month.

Attached below is a Gantt Chart to illustrate the described project schedule:



	Task Name	Duration	Start	Finish
1	**Project Planning**	**49 days**	**Tue 19/1/21**	**Fri 26/3/21**
2	Define project scope	4 days	Tue 19/1/21	Fri 22/1/21
3	Requirements Elicitation	5 days	Mon 25/1/21	Fri 29/1/21
4	Prepare initial project proposal	20 days	Mon 1/2/21	Fri 26/2/21
5	Prepare project plan	10 days	Mon 1/3/21	Fri 12/3/21
6	Prepare quality management plan	10 days	Mon 1/3/21	Fri 12/3/21
7	Prepare risk management plan	10 days	Mon 1/3/21	Fri 12/3/21
8	Review and finalize project proposal	5 days	Mon 15/3/21	Fri 19/3/21
9	Acquire resources	5 days	Mon 22/3/21	Fri 26/3/21
10	**Requirements Analysis**	**25 days**	**Mon 29/3/21**	**Fri 30/4/21**
11	Draft use case model (diagram and description)	10 days	Mon 29/3/21	Fri 9/4/21
12	Prepare system requirement analysis	5 days	Mon 12/4/21	Fri 16/4/21
13	Draft class overview diagram	10 days	Mon 19/4/21	Fri 30/4/21
14	Draft sequence diagram	10 days	Mon 19/4/21	Fri 30/4/21
15	Draft dialog map	10 days	Mon 19/4/21	Fri 30/4/21
16	**Design**	**25 days**	**Mon 3/5/21**	**Fri 4/6/21**
17	Database design	20 days	Mon 3/5/21	Fri 28/5/21
18	User interface design	20 days	Mon 3/5/21	Fri 28/5/21
19	Prepare test cases	5 days	Mon 31/5/21	Fri 4/6/21
20	**Development & Implementation**	**110 days**	**Mon 31/5/21**	**Fri 29/10/21**
21	Develop code	100 days	Mon 31/5/21	Fri 15/10/21
22	Developer testing (primary debugging)	10 days	Mon 18/10/21	Fri 29/10/21
23	System prototype complete	0 days	Fri 29/10/21	Fri 29/10/21
24	**Integration & Testing**	**25 days**	**Mon 1/11/21**	**Fri 3/12/21**
25	Code integration	5 days	Mon 1/11/21	Fri 5/11/21
26	Test the solution system based on predefined test cases by QA	5 days	Mon 8/11/21	Fri 12/11/21
27	User test prototype	1 day	Mon 15/11/21	Mon 15/11/21
28	Modify code based on feedback	9 days	Tue 16/11/21	Fri 26/11/21
29	Final touchups of system	5 days	Mon 29/11/21	Fri 3/12/21
30	**Deployment & Maintenance**	**35 days**	**Mon 6/12/21**	**Fri 21/1/22**
31	Determine final deployment strategy	20 days	Mon 6/12/21	Fri 31/12/21
32	Secure deployment resources	14 days	Mon 3/1/22	Thu 20/1/22
33	Deployment of Application	1 day	Fri 21/1/22	Fri 21/1/22
34	Maintenance	35 days	Mon 6/12/21	Fri 21/1/22

6 Operational Requirements

6.1 Help Desk Support

System users have the ability to contact support within the application through email to either ask for assistance in questions that are too specific or technical in nature and have not been answered in the FAQ section or to report a specific bug/error. Such technical issues can range from application errors, system downtime inquiries or even account lock-out assistance.

Moreover, in line with the Personal Data Protection Act 2012 (PDPA), there must be processes in place for system users to request a full copy of their personal data stored across all CashTrack servers. This request must be serviced accurately and effectively within seven working days.

6.2 Application Services and Technical Support

Programmers and application developers will have to request access to the source code of the application to address any bugs or make system enhancements whenever necessary.

Both the Network Administrator's and Database Administrator's contribution and support are essential to maintain a 24/7 system uptime to ensure maximum availability of the system.

6.3 Administration Features

Various levels of access and security are implemented for the different types of users throughout the system. The system administrator will have the highest security clearance, allowing them full access to details of user (payer and payee) accounts. This complete access does not extend to private and sensitive user data, which is stored using end to end encryption (E2EE) to ensure data integrity and security. The system administrator will also have the rights to flag/disable any user accounts that violate the CashTrack code of conduct.

User accounts however only have access to their own individual details, and cannot access other users' accounts. This is enforced through user authentication which has been implemented via Google OAuth access delegation. Additionally, there are also different levels of access between the payers and payee, one key feature being the modification of the amount owed (amount payer is supposed to pay) which is only available for the payee.

6.4 System hardware fail over and routine backup

System will be required to perform bi-weekly data backup of all users data on the system to prevent any loss of user data in the event of a system failure.

Backup routines are scheduled to be carried out on a fixed date bi-weekly by the development team unless there are new features being implemented or when there are bugs / errors to be fixed in the system.

6.5 Audit Trail

An audit trail will log all expense records in the System. This includes the time and date of expense creation, currency, updating of expenses, comments, and time and date of payment acknowledgements.

Further, an access token will be generated by OAuth whenever a user logs in, this token will be used to track all user activity within that session.

7 Functional Requirements

7.1 General User:

7.1.1. System must allow User to register an account.
7.1.2. System must allow User to log in.
    7.1.2.1. System must allow User to log out.
7.1.3. System must allow User to invite people to the application.
7.1.4. System must allow User to create a group involving two or more users.
    7.1.4.1. System must allow User to invite other users to the group.
    7.1.4.2 System must allow User to remove other users to the group.
7.1.5. System must allow User to chat with other users.
    7.1.5.1. System must send new chat message notifications in real-time.
7.1.6. System must allow User to set a desired personal expense limit amount for given time frame
    7.1.6.1. System must allow the User to set a percentage for the expense limit, which when reached must notify the User of the same.
    7.1.6.2. System must send a notification to User when User exceeds the set limit.
7.1.7. System must allow User to input their personal expenses
    7.1.7.1. System must allow User to select the category/type of inputted expense.
    7.1.7.2. System must allow User to update/modify details of selected personal expense records.
    7.1.7.3. System must allow User to delete selected personal expense records.
7.1.8. System must allow User to view data analytics of their expense records.
    7.1.8.1. System must be able to visualise spending patterns based on personal expenses by categories using pie charts.
    7.1.8.2. System must be able to visualise spending patterns based on shared expenses with another user over time using line charts.
7.1.9. System must allow a User to delete their CashTrack account.

7.2 Payee:

7.2.1. System must allow User to split an expense, by creating a shared expense record, with one or more individuals or a group.
7.2.2. System must allow User to split the expense Equally, by Percentage, by Shares, by custom amounts, or with a currency exchange feature.
7.2.3. System must allow User to acknowledge payment from another User (Payer).
    7.2.3.1 System must send a notification to User when image evidence has been added to the shared expense record they created.
7.2.4. System must send a notification to User when an expense they created receives a comment.

7.2.5. System must allow the User to make comments for any expense record they are involved in.

7.2.6 System must allow Users (Payee) to update/delete a shared expense record.

7.3 Payer:

7.3.1. System must allow User to Settle payment with User (Payee)

7.3.1.1. System must allow User to upload image evidence

7.3.2. System must allow User to comment on an expense that they are involved in

7.3.3. System must send a notification to User when an expense they are involved in receives a comment.

7.3.4 System must send a notification to the User when an expense they are involved in is edited or deleted.

7.3.5. System must send a notification to the User (Payer) when another User (Payee) acknowledges their payment for a shared expense.

8 Non-Functional Requirements

Reliability

- The system must maintain an uptime of 99.999% to minimise user inconvenience and maximise reliability.
- In the case of a system network issues, the system's backup server shall be utilized.
- After a system reboot, the full system functionalities must be restored within 5 minutes.
- Crucial data such as user accounts and all data must be backed up bi-weekly at 3am on Friday on Google Cloud Platform and MongoDB respectively.

Usability

- The system must be able to offer informative feedback to users.
- The system must reduce short-term memory load.
- The system must allow for easy reversal of actions.
- The system must strive for consistency.
- The system must have a Frequently Asked Question (FAQ) that enhances the user's experience and provides quick information.

Performance

- Upon user-initiated query, the system must process and display the results to users within 2 seconds.
- The system must be able to handle a load of 50000 users at any one time.
- The system must strive to keep response time within 3 seconds to prevent the users from experiencing latency.
- The system should not take more than 7 seconds to query and insert a newly created user account to the database.

Supportability:

- The database must be replaceable with any commercial product supporting non-relational data queries.
- The system must also exhibit portability and support integration with similar development tools at the time of system upgradation or migration.

9 Process Requirements

9.1 Database Transaction

The system must be able to fetch from, update and save data to the cloud MongoDB. The communication with the database system will be handled by the ExpressJS backend server through REST API endpoints which will be called from the frontend of our application.

9.2 Data Validation

The backend system must perform validation on the data being saved and the data being fetched from the database to ensure that the data adheres to the established structure and matches the database schema. The system must perform error handling on missing data, wrong data, incorrect user input or mismatch of actual response from the database and the expected response. While performing CRUD operations, all data must be processed by the ExpressJS server to match the expected format before passing forward to the frontend application or database.

9.3 Data Integrity

Each datapoint in the database must be saved with a unique identifier to make it searchable and modifiable by the backend. If any record is added to, updated, or deleted from the database, cascading must be done to ensure that all the relevant datapoints reflect the same change accurately and consistently to maintain integrity.

9.4 Performance

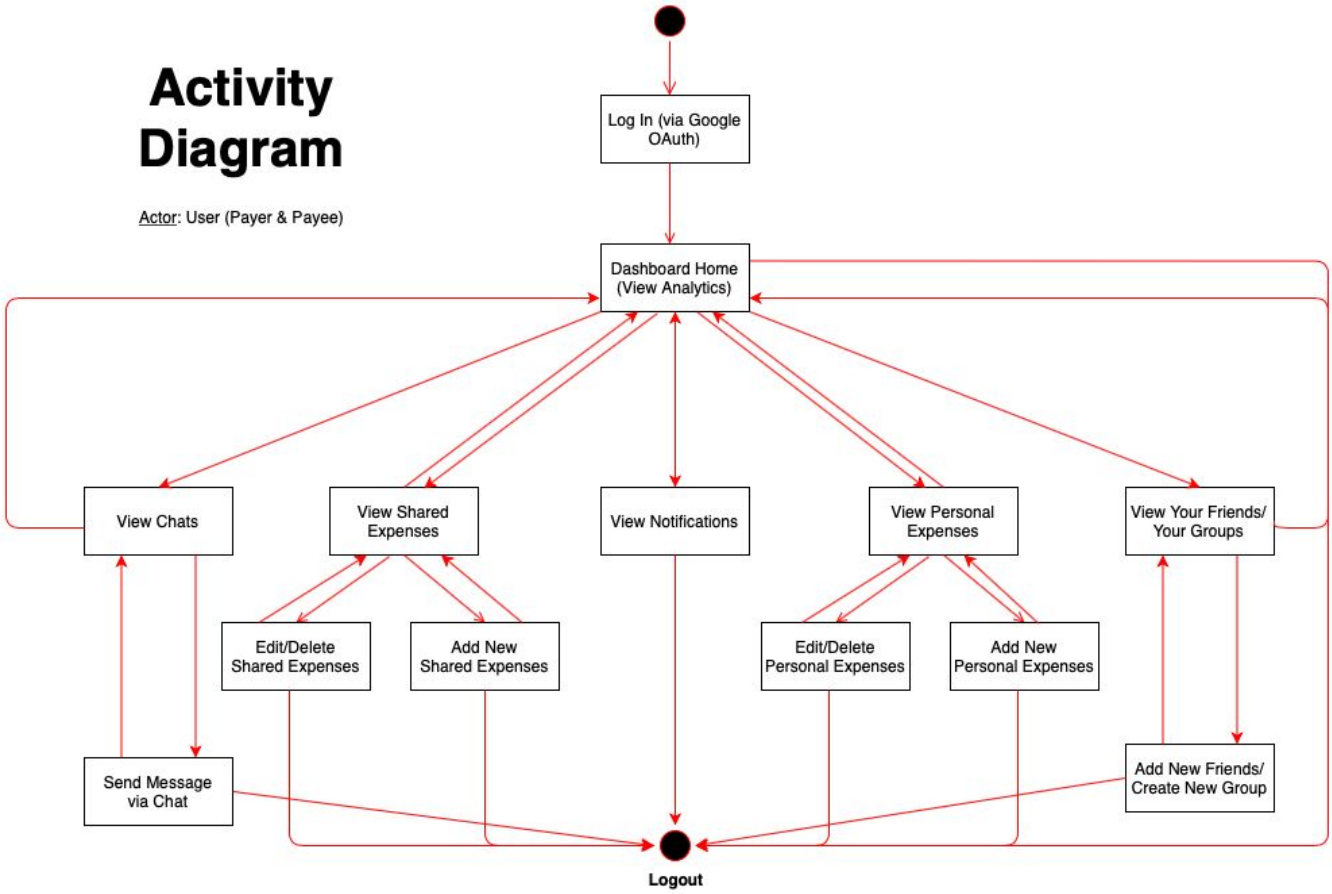
The expense records must be updated in the database within 2 seconds of it being added/modified by the user, to facilitate real time updates. If the same record is updated by many users at the same time, the database must queue and process every update, to resolve concurrency and reflect the most accurate data representation for the user.

The frontend application must provide a seamless lag-free user experience, maintaining 99.99% upkeep. The maximum time for any API call by the frontend application must not be more than 500ms, and the loading icon must not be shown to the user anytime greater than 1.5 seconds.

9.5 Data Repository

The MongoDB database must store all the relevant records relating to user information, expense records, chats and analytics data. This database must be hosted on a cloud server to be accessible remotely from anywhere with valid credentials, and not be restricted by localhost.

9.6 Activity Diagram



10 Input Requirements

10.1 Gmail Account Login Details - The application provides login through the user's Google account, and thus the user must provide their Google account login details to start using the application

10.2 Set personal expense limit (Optional) - The User must be able to set a personal expense limit and must be able to set a reminder to be notified when a certain percentage of the expense limit has been reached

10.3 Personal Expense Record Details - While inserting a personal expense record, the user will have to input the following information about the expense:

- Label - A text element
- Category - A dropdown to pick from the following options:
    - Food
    - Travel
    - Entertainment
    - Shopping
    - Others
- Amount - An amount in SGD

10.4 Shared Expense Record Details - While creating a shared expense record, the user will have to input the following information about the expense:

- Label - A text element
- Category - A dropdown to pick from the following options:
    - Food
    - Travel
    - Entertainment
    - Shopping
    - Others
- Amount - An amount in SGD
- Payer(s) - A multi-select dropdown displaying all the other application users that the particular user has added to their 'friend list.'
- Choose whether to split or "you owe" - A dropdown to select between "You owe *UserX* money" or "Split the bill"
- Split By Method (Choose) - A five-tab pop up allowing the User to choose how to split the bill. The tabs are as follows:
    - Split equally
    - Split by shares - User must enter shares of split for each of the friends in this expense
    - Split by percent - User must enter percentage of split for each of the friends in this expense

    - Split custom - User must enter specific amounts of split for each of the friends in this expense
    - Split Currency - User must enter the currency they want to convert to SGD
- Attachments (Optional) - The User(Payee) can choose to upload a picture of the bill.
- Comments (Optional) - The User(Payee) can optionally add comments to the expense

10.5 Comments by User(Payer) - The User(Payer) can optionally add comments to a shared expense record he/she is a part of

10.6 Acknowledgement by Payer if money sent - The User(Payer) can choose to upload a proof of payment and must press the ‘Settle Up’ button in the expense record after paying the author of the bill

10.7 Acknowledgement by Payee if money received - The User(Payee) must acknowledge that the money has been received and press the ‘Settle Up’ button to mark the expense record as ‘Settled’

11 Output Requirements

11.1 Expense Records Summary

The application will display personal and group expense records. Users shall also be able to view their data analytics of spending categories in the form of a Pie Chart, and personal spending patterns or shared transaction histories with friends in the form of a Line Chart.

11.2 Notification

11.2.1. User Personal Expense Limit

The application must send a notification notifying users when they have reached the specified percent of their set expense limit. The application must also send a notification when the user exceeds their personal limit.

11.2.2. Payee Group Expense

The Payee must receive a payment notification from the Payer with an acknowledgement action by Payee, after which the shared expense can be marked as resolved.

11.2.3. Payer Group Expense

An acknowledgement request from the Payer must be sent to the Payee, when the Payer has settled up the full amount of the shared expense record.

11.2.4. Payer Group Expense Reminder

The application must send a reminder notification notifying users that they are yet to settle a shared expense record.

11.2.5. Group Expense Created / Edited / Deleted

The application must display a notification to members (Payers) of a group expense upon creation / updation / deletion of the group expense.

11.3. Chat

Upon selecting a User to converse with, the application must display the messages from the conversation prior between them if the User has had any past conversations with them, and all future messages should be reflected in real time.

11.4. Success and Error messages

The application must give informative feedback, and display the appropriate success and error messages respectively upon user inputs.

12 Hardware Requirements

Any internet enabled device with a web browser will be able to access this application. The hardware must establish a secure and stable connection with the internet before the application can be accessed.

12.1 Network

Network connection to the internet is established primarily via 2 types of devices. The first instance is that of a personal computer, for which a network interface card/network adapter is required for the PC to receive network connection. The second device type is mobile devices, such as tablets or smartphones, which require the use of a modem chip with cellular capabilities to facilitate connection to the internet.

12.2 Client Devices

The client can use any internet enabled device, equipped with a browser and an internet connection to run CashTrack. On that note, the minimum device requirements to access the application for the identified devices are as follows:

For Computers:

	Minimum Requirements	Recommended Specification
Processor	Intel ® Dual Core ™	Intel ® Core™ i5
Memory	4 GB RAM	
Internet	Broadband internet connection	
Resolution	1024 x 768 minimum display resolution	

For Smart Devices:

	Recommended Specification
Memory	512 MB RAM
Resolution	640 x 960 minimum display resolution

12.3 Device Capabilities

CashTrack can be run on two types of devices (Personal Computer & Smart Device). Smart devices accessing CashTrack should have touch input capabilities for users to input relevant data and navigate

through the features of CashTrack. Personal Computers accessing CashTrack should have a keyboard to input relevant data and a mouse to navigate through the features of CashTrack.

Moreover, in order for users to be able to access CashTrack, each device must establish a stable internet connection. Further, our web application shall be cross site responsive, which will ensure that web content is flexible to fit varying device screen sizes and provides a consistent experience to all users.

12.4 Production Support Systems

12.4.1 Uninterruptible Power Supply (UPS)

One of the production support systems to be used for this project is the Uninterruptible Power Supply (UPS). As a key feature in computing, it acts as a contingent power source when the regular power source fails [B.2.3]. This way, data loss is averted and productivity is boosted as there is no fear of a possible power disruption. One of ways a UPS is implemented is through a Standby Power System (SPS). [B.2.3]

Unstable electrical current can prove detrimental to the system, as noise and power surges can occur [B.2.4]. By connecting the SPS to the server, the power line is constantly being observed. Once a problem is discovered, the SPS will switch to battery power. This way, the website hosting server can be online at all times and users will be able to access it without any interruption.

12.4.2 Cloud Infrastructure

The use of cloud infrastructure is an additional development support scheme that is going to be utilized for this project. Using the cloud to store our information offers multiple benefits, such as cost effectiveness and rent utility constraints. As host programs can be migrated to other available servers in the event that a server fails, the cloud is considered secure.

12.4.3 Other Methods

Additionally, we aim to set up servers at multiple geographical locations. It is imperative that the CashTrack website has rapid content rendering. From a user’s perspective, an unpleasant user experience would prevent the user from returning to our application. This emphasizes the need to host the website on servers in locations that are closer to the target audience.

13 Software Requirements

13.1 Client Operating System

The minimum operating system requirements, for the two primary client devices, to successfully interface and access our application, CashTrack are as follows:

For Computers:

	Minimum Requirements	Recommended Specification
Operating System	Windows®7 / macOS 10.13	Windows®7/ Windows®8/ Windows®10 64-bit / macOS 10.13.5 or higher

For Smart Devices:

	Recommended Specification
Operating System	iOS 12.0 or higher / Android 8.0 or higher

13.2 Client Application

The latest release of each of the latest two supported major versions of:

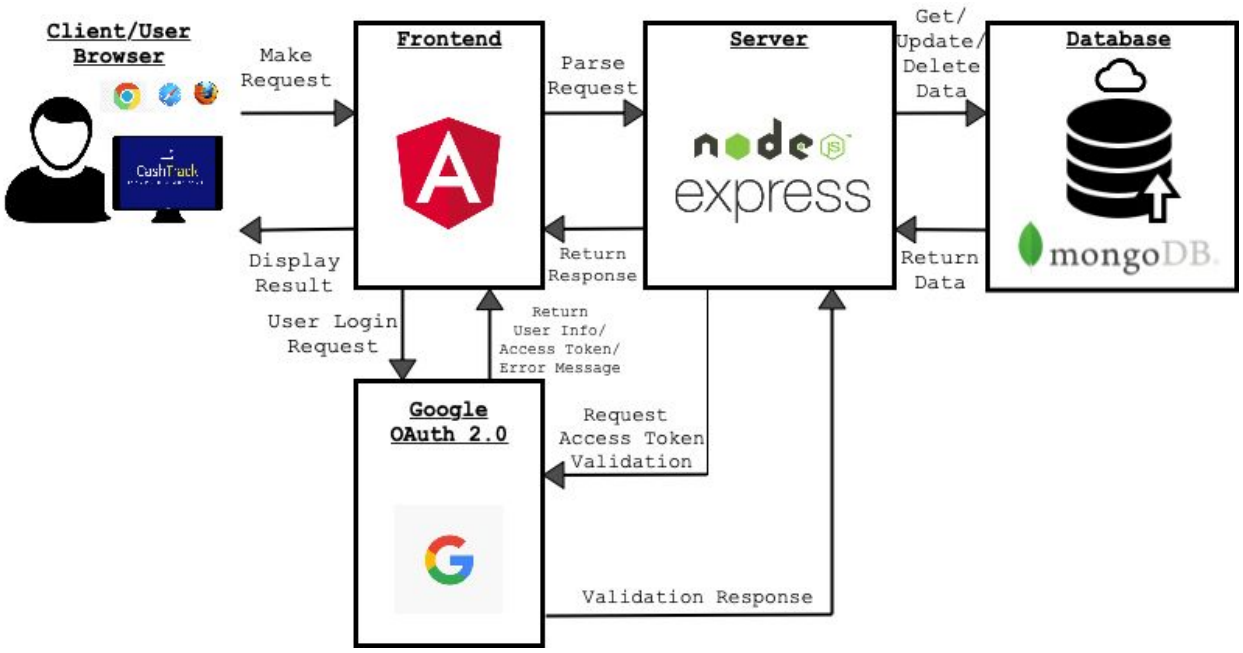
- Desktop browsers:
  - Google Chrome
  - Firefox
  - Safari
  - Microsoft Edge
  - Opera
- Mobile browsers:
  - Safari for iOS
  - Chrome for Android

14 Deployment Requirements

The deployment of our application, CashTrack, ensures remote application accessibility, scalability and reliability as the application evolves and grows.

The Angular Frontend and the Node Express Server shall be hosted on a Google Cloud Platform's App Engine Instance. GAE is a Platform-as-a-service i.e. PaaS, which ensures automatic scalability and a seamless service experience. The Google OAuth service shall also be utilized, which shall be accessed by the frontend by using APIs hosted and maintained by Google. Moreover, the REST APIs will be accessible via HTTP requests and the User will be able to access the application frontend by using an HTTP URL in a web browser. This URL will be defined later in the production stage. Further, MongoDB will be hosted in a cloud server which is accessible via Mongo drivers for Node by providing the database credentials.

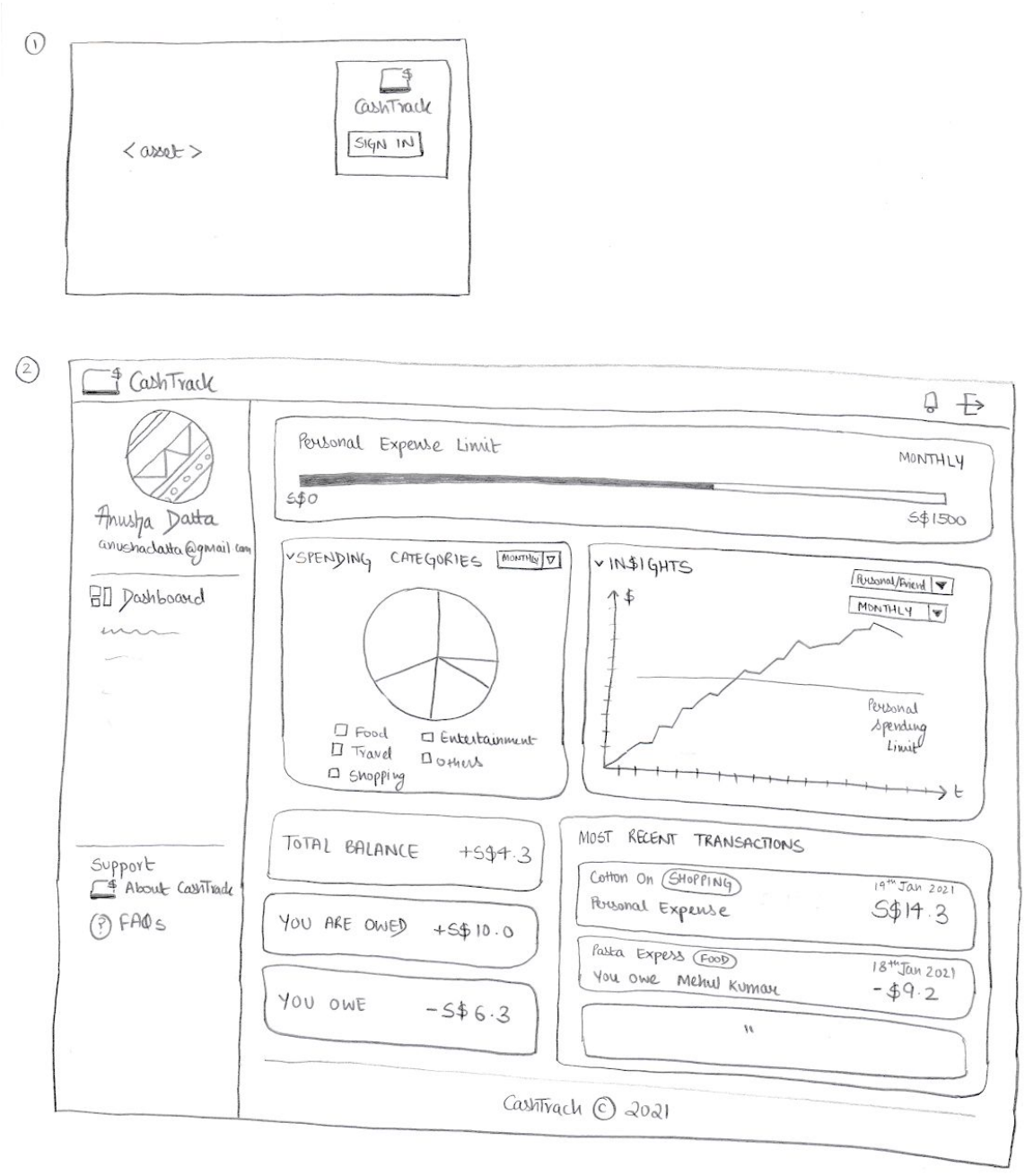
The interactions between the various components of our software development stack, MEAN and Google OAuth API 2.0 (used for User Authentication), are visualised in the diagram below.



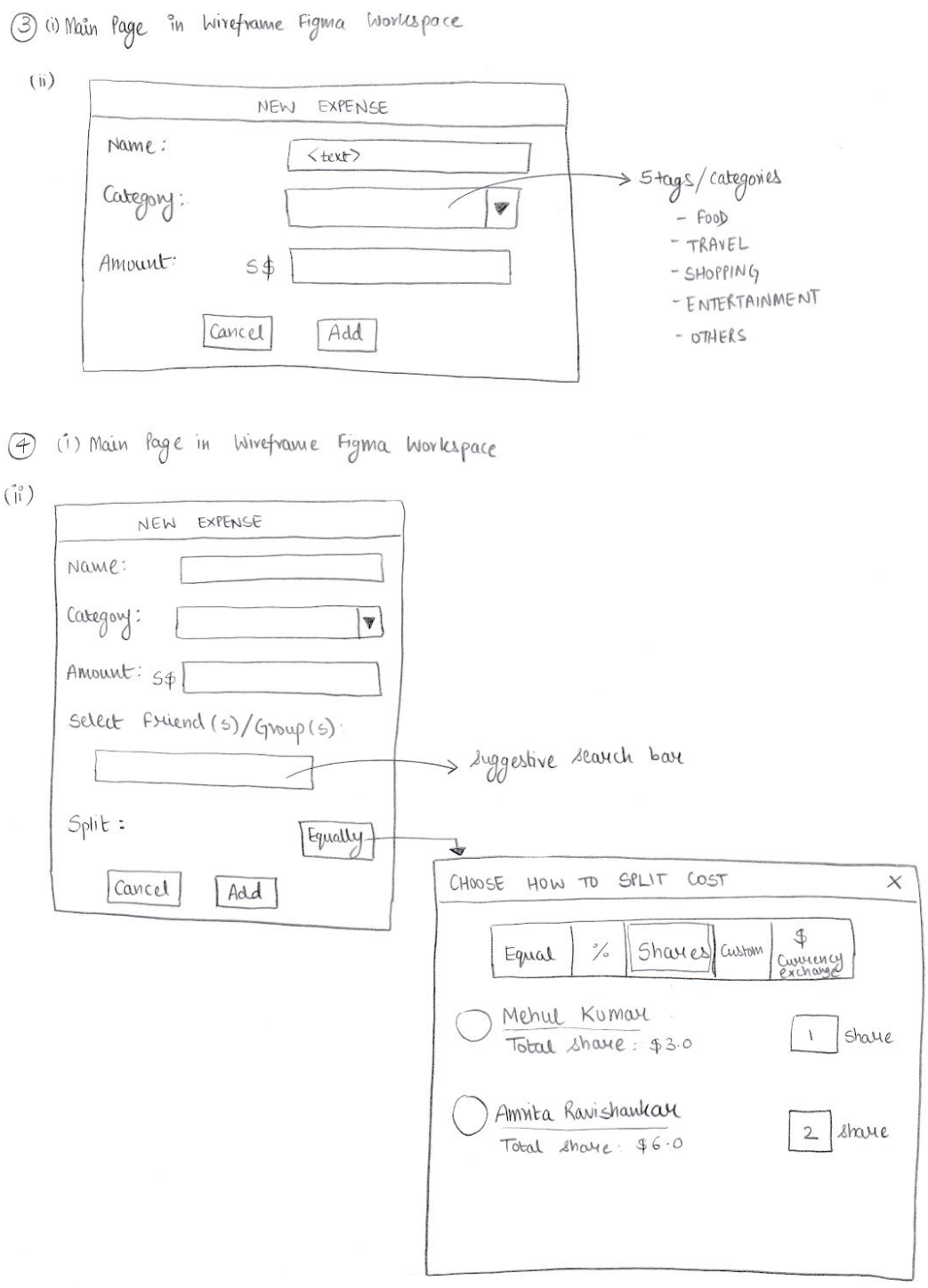
Appendix A: Prototype

[A1] Lo-Fi Prototypes

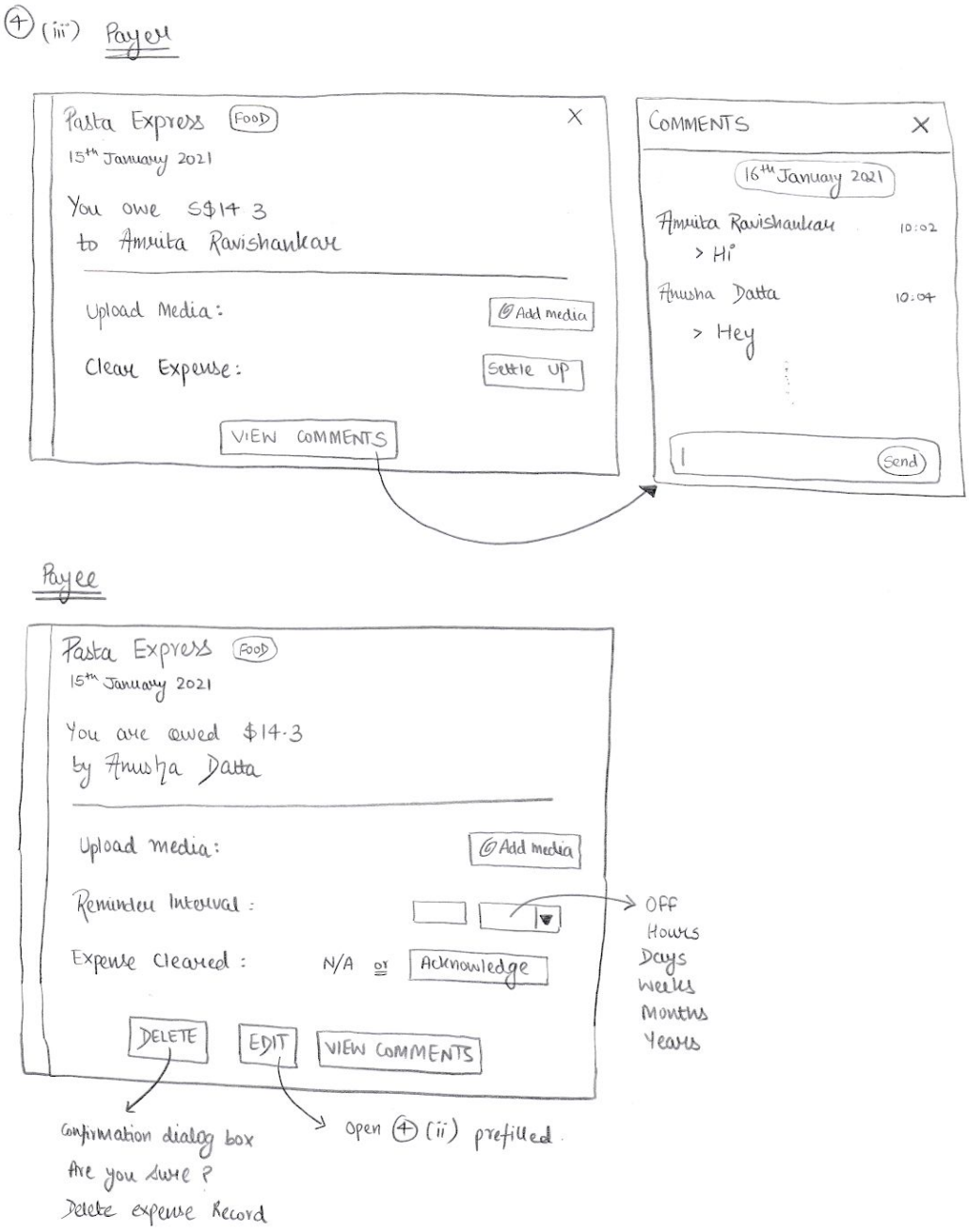
Landing Page and Login



Input Expense

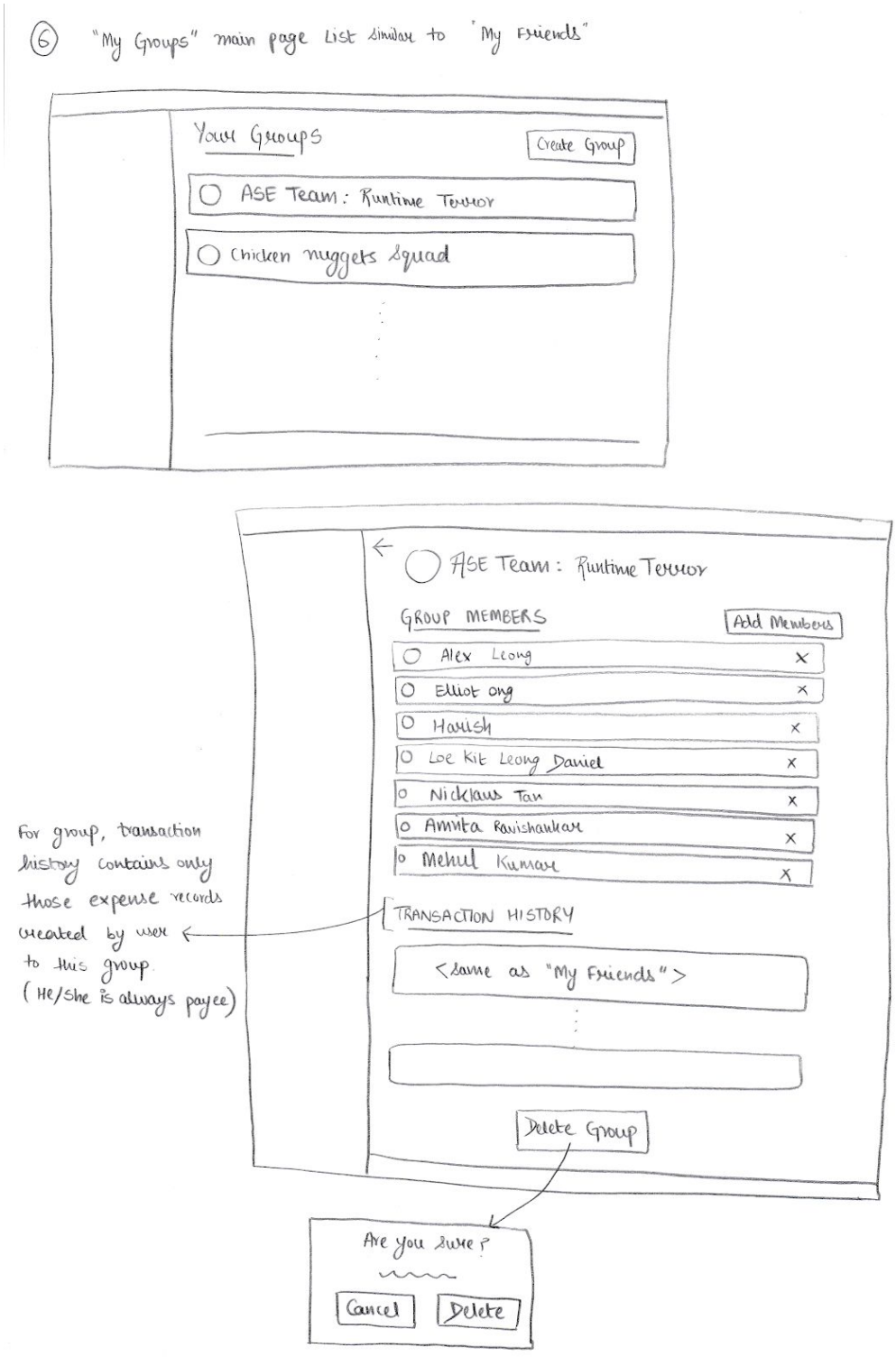


Payer and Payee Share Expense Record View

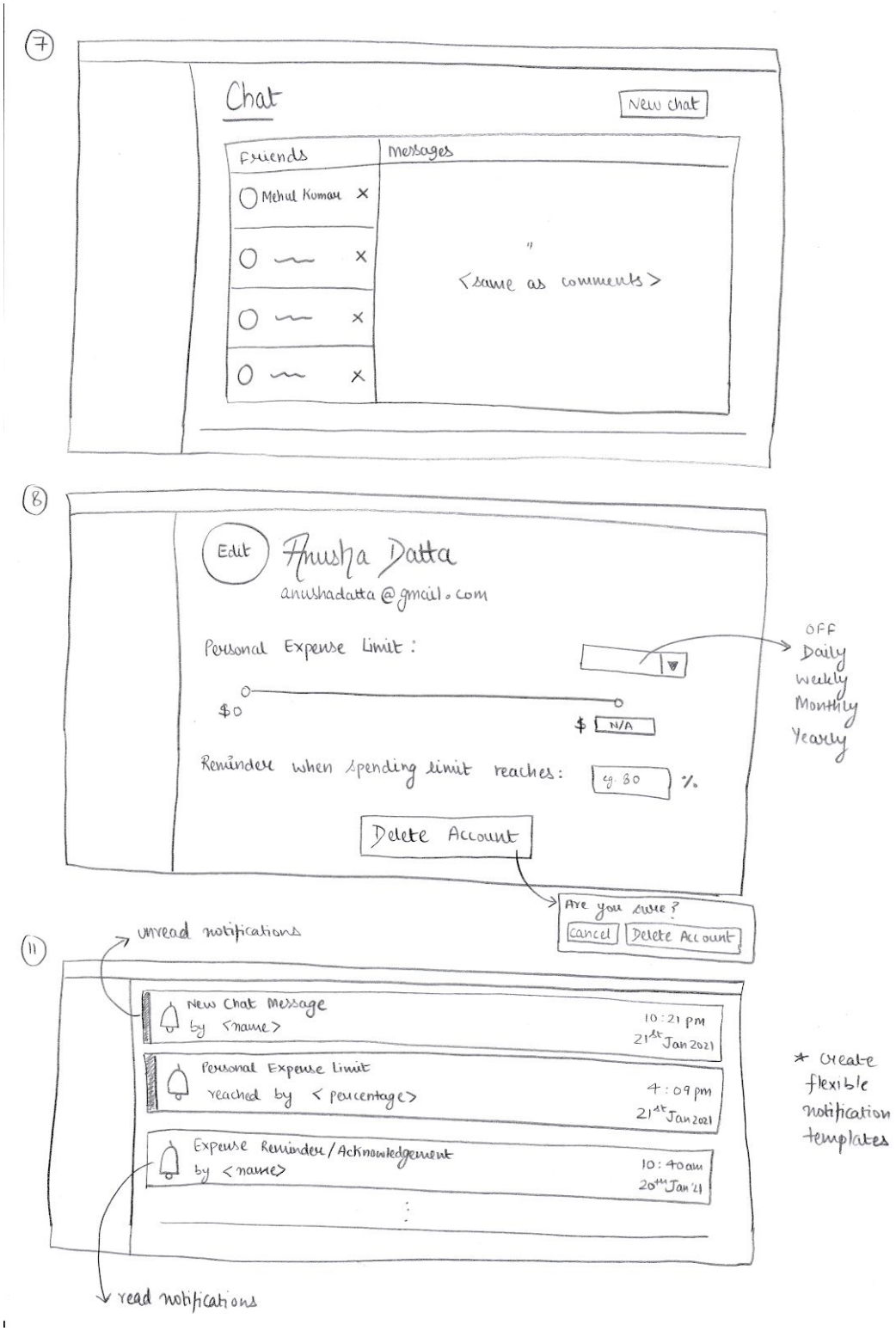


Friend Details

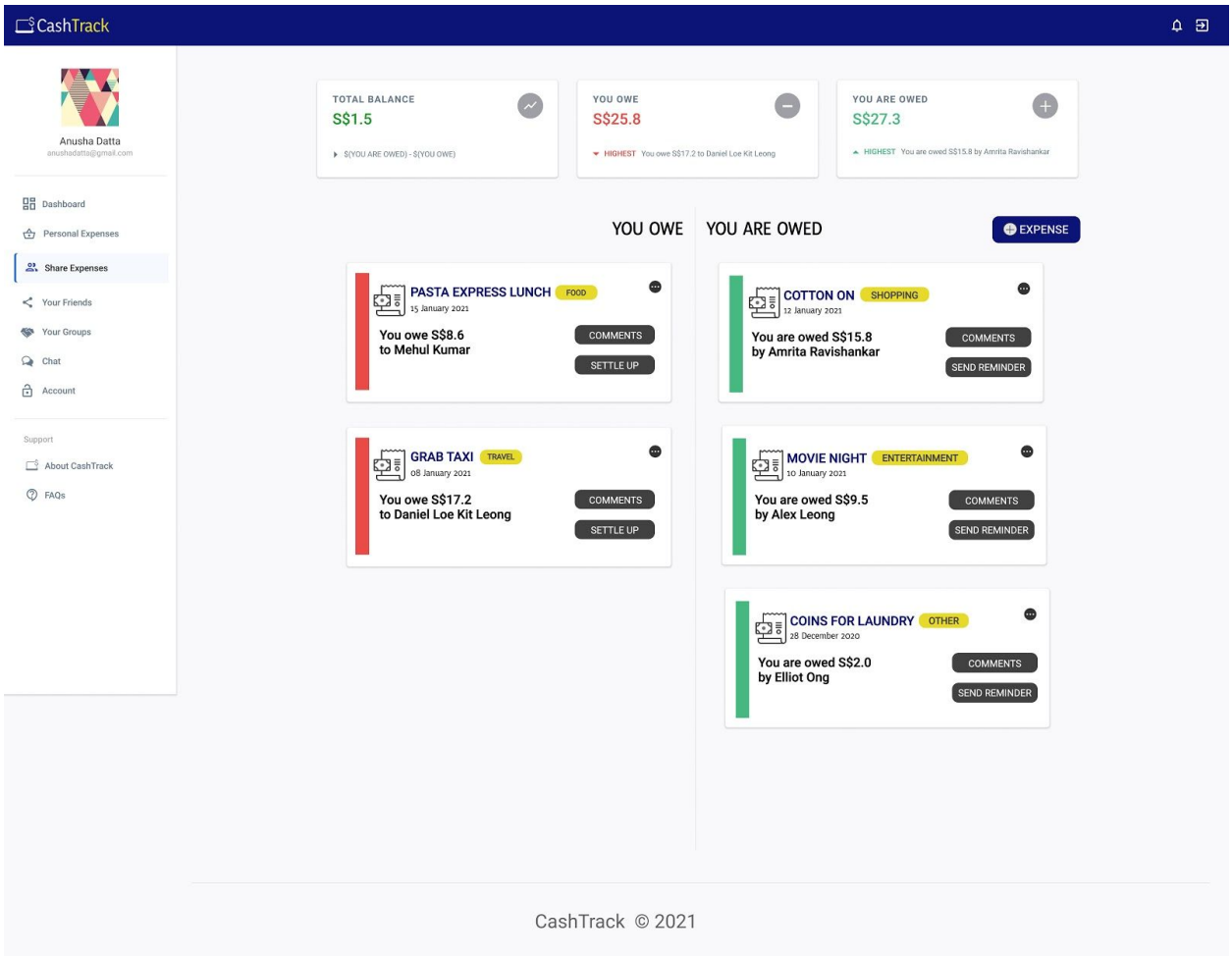
Group Details



Chat and User Account Information

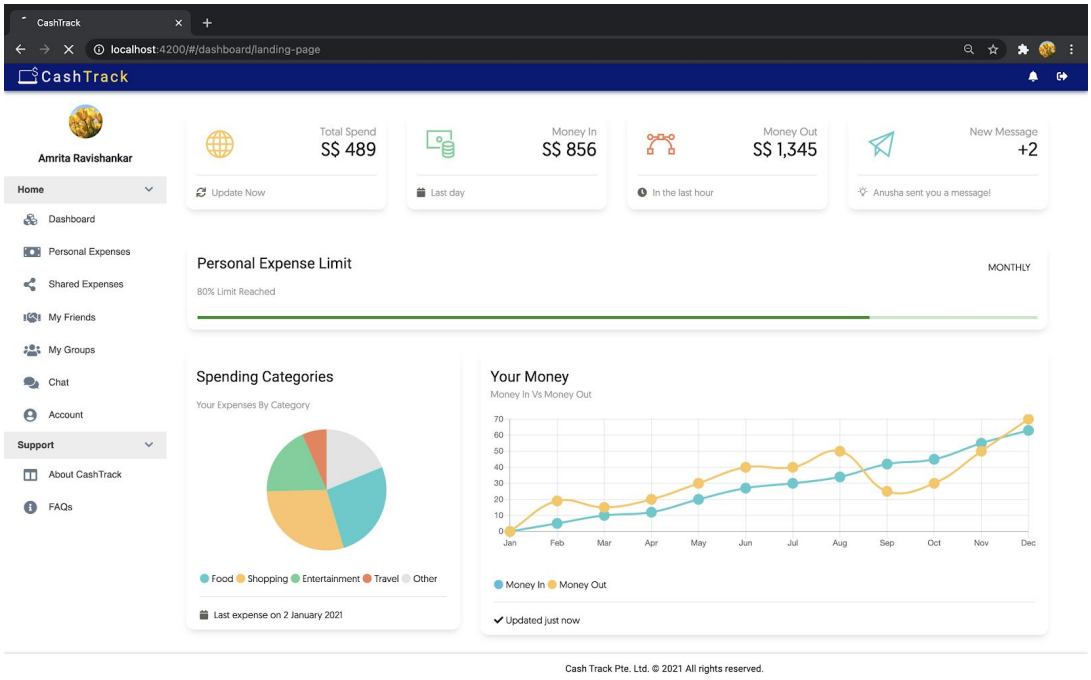


[A2] WireFrame - Shared Expenses Page

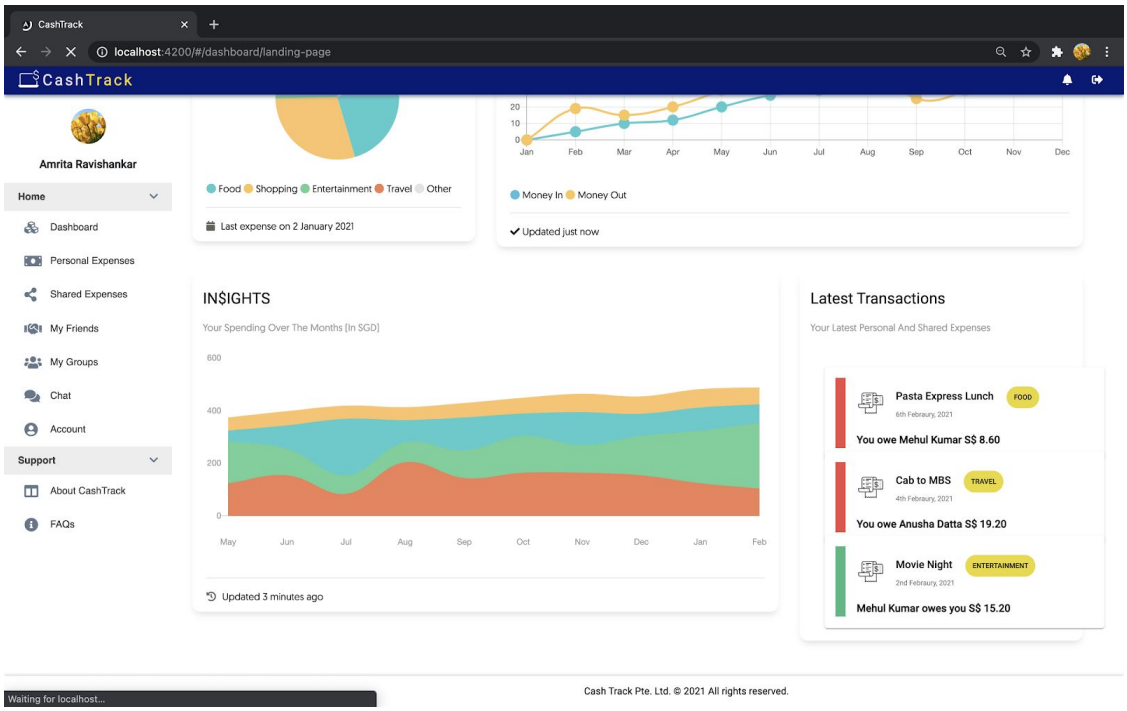


[A3] Prototype Screenshots

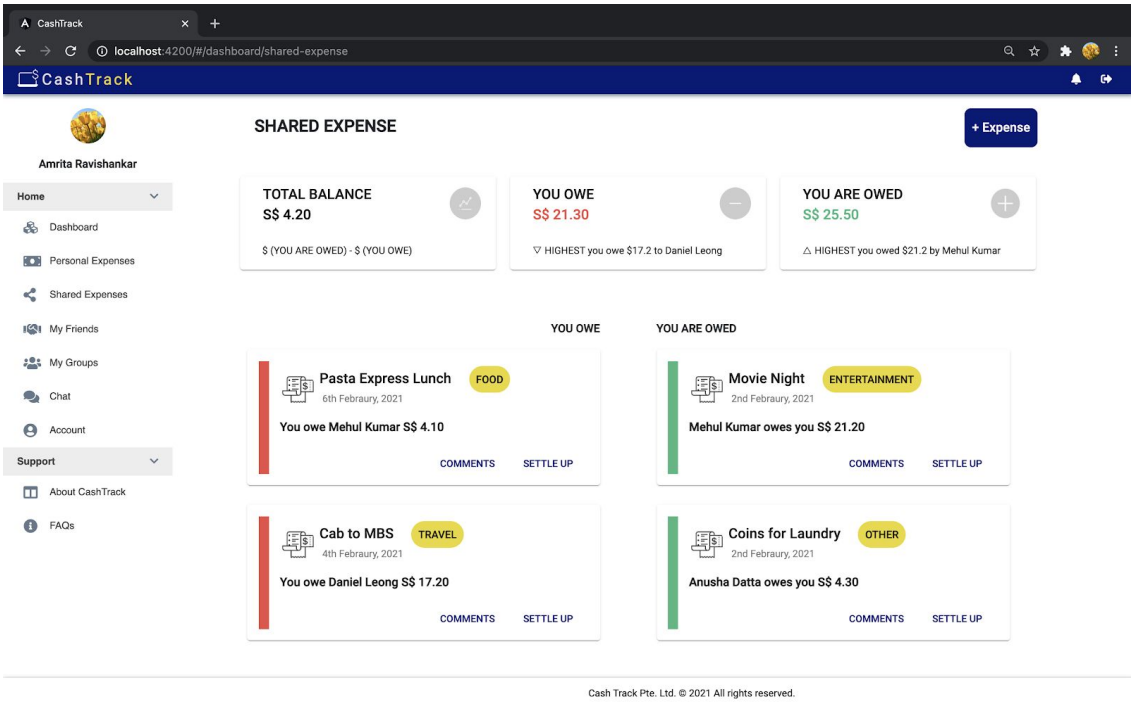
Landing Page Screenshot 1



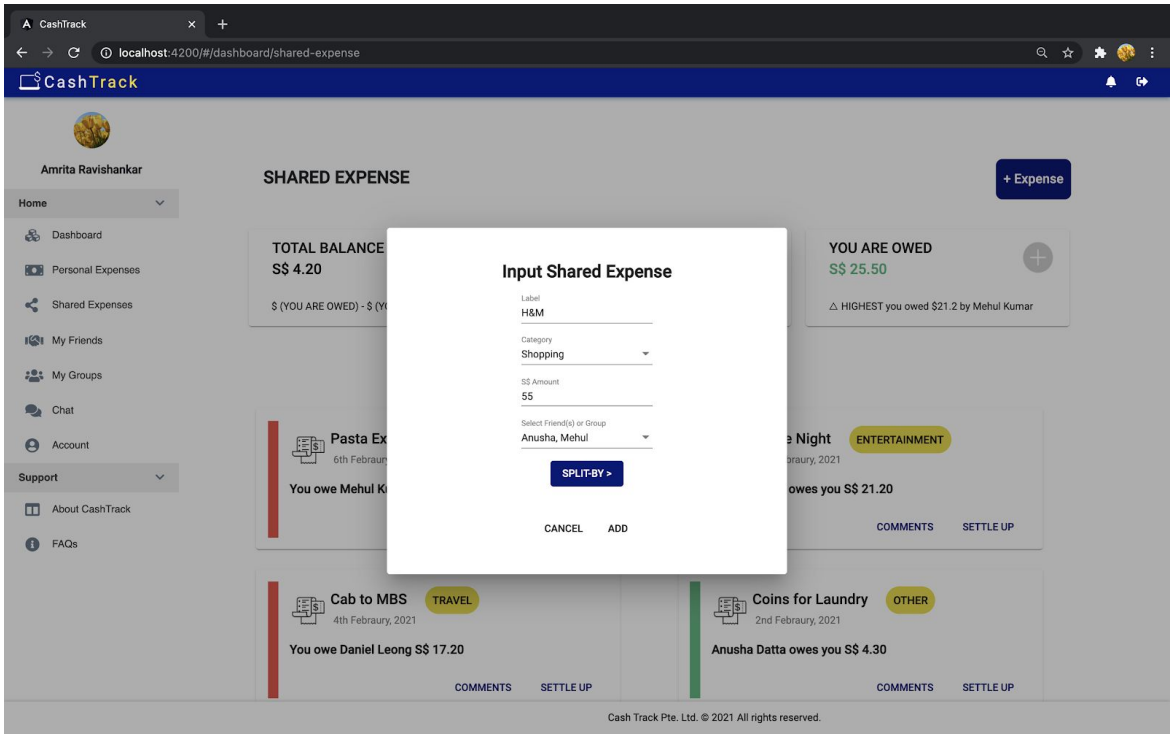
Landing Page Screenshot 2



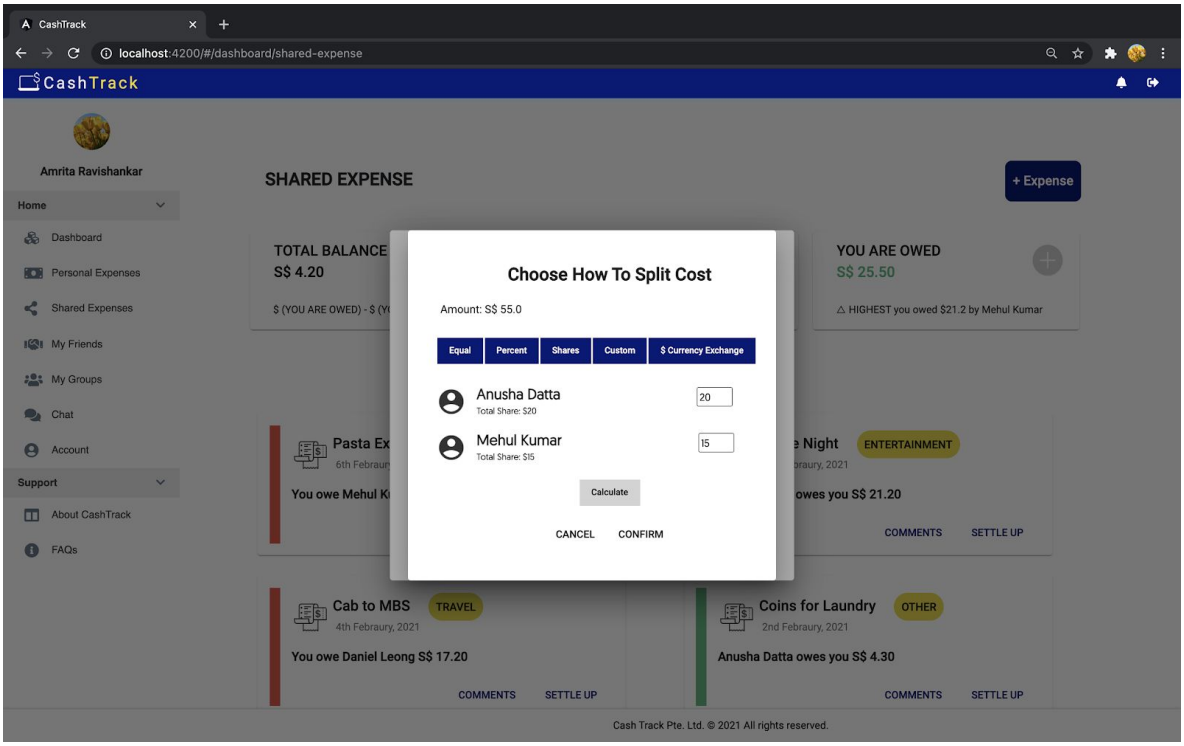
Shared Expenses Page - Overview



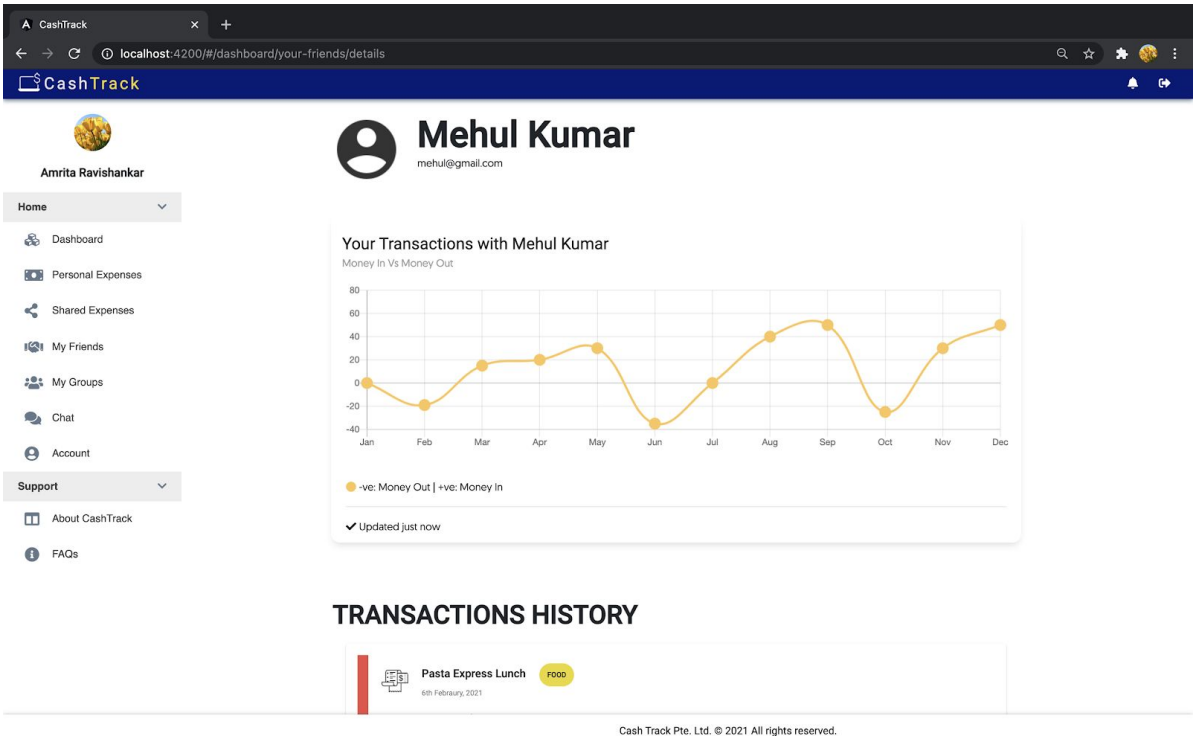
Shared Expenses Page - Input Expense



Shared Expense Page - Split By Choices



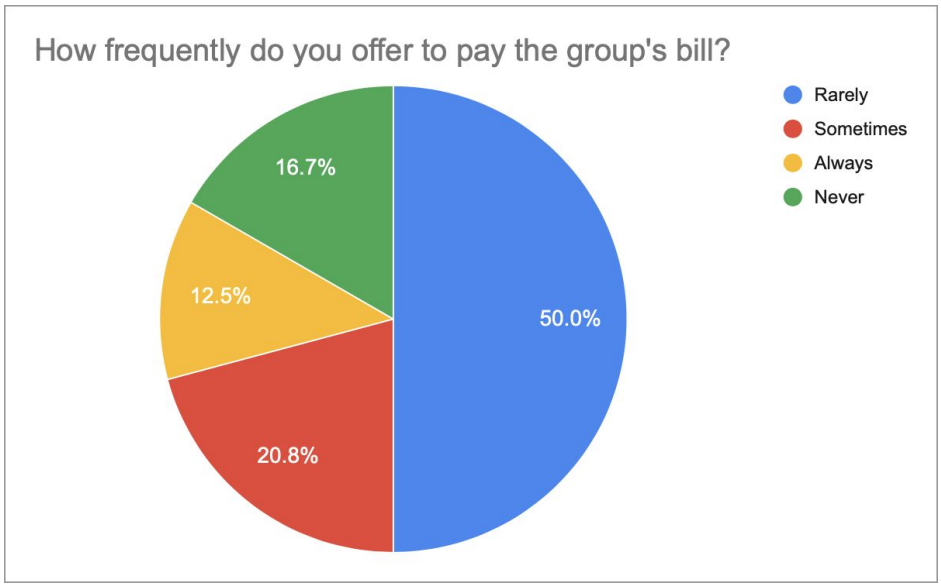
Details of transactions with a friend



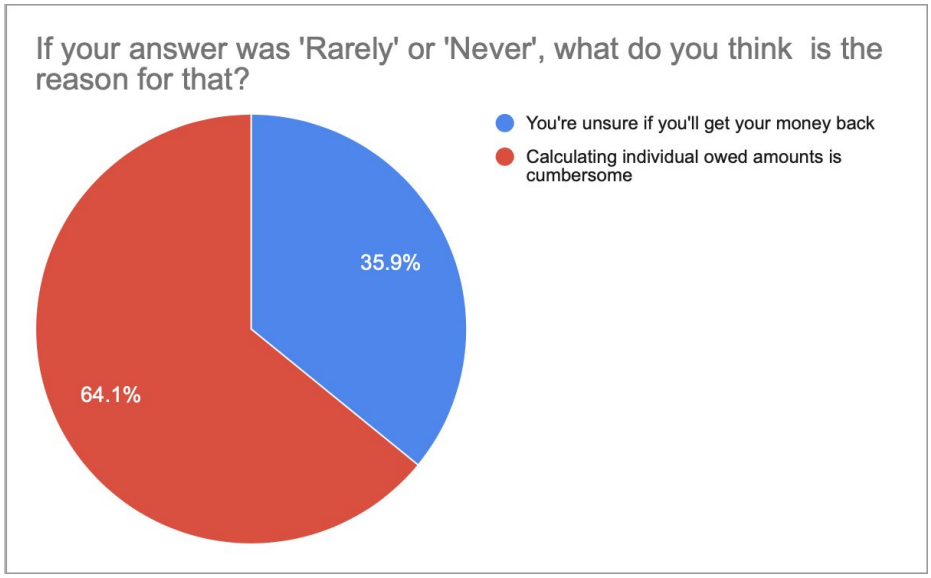
Appendix B: References

[1] Graph Data - Survey

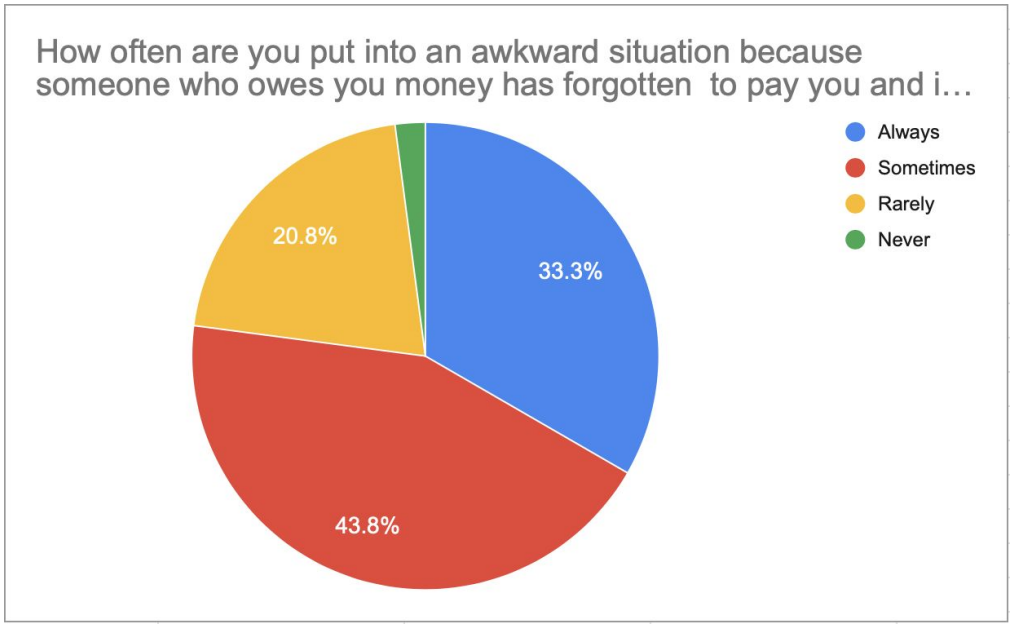
[1.1]



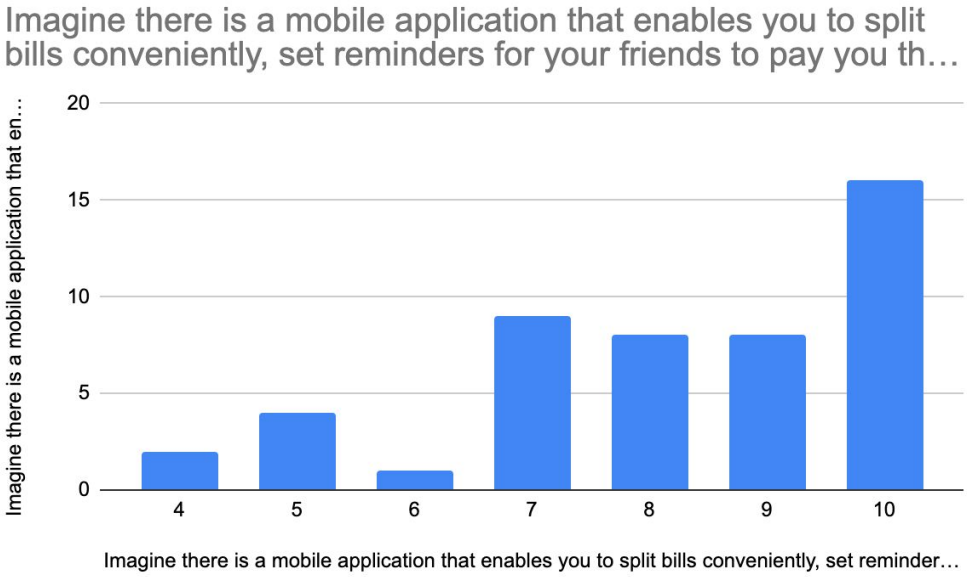
[1.2]



[1.3]



[1.4]



[2] Sources of Information

[2.1] McKinsey & Company. “The 2020 McKinsey Global Payments Report.” McKinsey & Company, Oct. 2020.

[2.2] Rob Clymo “More than 6.1 billion people will use digital payments by 2023.” TechRadar online publication, 12 May, 2020.

[2.3] (n.d.). How UPS (Uninterruptible Power Supply) Systems Works. (n.d.). Retrieved from: https://www.electricalengineeringtoolbox.com/2017/07/how-ups-uninterruptible-power-supply.html

[2.4] (n.d.). Common issues with power supply. Retrieved from: http://www.captech.com.au/2016/05/06/common-issues-with-power-supply/

Appendix C: Data Dictionary

Term	**Definition**
System	CashTrack application
User	Person using CashTrack application.
User (Payee)	User who created a Group Expense Record and is the owner of the group expense record. This user will be receiving payment from User (Payer).
User (Payer)	Users who are invited to a group expense record. This user will be making payment to User (Payee)
Expense Record (Personal)	An expenditure record that users can create to keep track of their spending.
Expense Record (Group)	An expenditure record that includes the user (Payee) and other user(s) (Payer).
Expense Limit	A value that the user can set so that the system can notify them when they are approaching their limit.
MongoDB	A source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.
Angular10	A JavaScript-based open-source front-end web framework.
ExpressJS	A back end web application framework for Node.js, designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.
URL	A Uniform Resource Locator (web address) is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.
API	Application Programming Interface. A computing interface that defines interactions between multiple software intermediaries

GAE	Google App Engine is a Platform as a Service and cloud computing platform for developing and hosting web applications in Google-managed data centers.
PaaS	Platform as a Service is a cloud computing model where a third-party provider delivers hardware and software tools to users over the internet