Christopher Wang (ctwang3)
Christopher Varghese (cv12)
Siran Xianyu(sxianyu2)

Receipt Scanner and Bill Splitter Mobile Application

Pitch

Splitting the bill can be confusing, messy, and time-consuming and its hard to manage debt over an extended period of time. Billsplit allows you to take pictures of your receipts and send payment requests to your friends at the push of a button. Just snap a pic of the receipt, select who is splitting each item, and your friends will receive a quick link to pay you, and they don't even need to download the app.

Functionality

- Users can sign into their account using private credentials
- Users can scan their receipts into a easily viewable and savable format
- Users can add friend or delete friend in their friend list
- Users can go through digitized receipts and split it itemwise, Users can select which receipt items each person is attached to
- Users can input the exact split of each item using a percentage
- After adding people to each receipt and users can send them an invitation to pay them back

Advanced Functionality(if time permits)

- Multiple users can all upload receipts to an expense group and Billsplit will settle this into a final tab
- Users can get the result of the outstanding balance between one and the other through debt shuffling

Components

Front End:

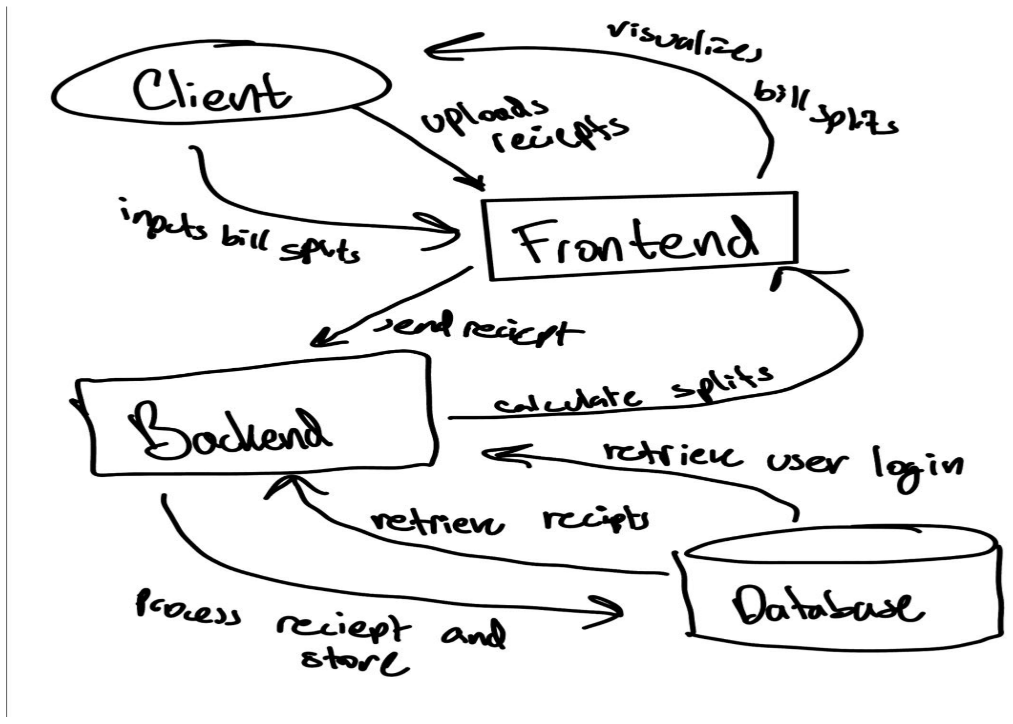
- **Functionality:** This part will be the interface where the user can interact with the application. It should have a clean and intuitive workflow.
- **Programming Language:** Javascript. This is a common frontend language that is also used with React Native.
- **Major Libraries Used:** React Native. This will allow the app to be run on both android and ios with a single codebase
- **Testing Methodology:** We will run both ios and android tests on our app. We will use Jest to create unit tests as well as live testing on a variety of virtual and real devices.
- **Interactions With Other Components:** This will rely heavily on the backend and database for retrieving information from the database and for general logic.

Back End:

- **Functionality:** This will govern the logic of the app such as retrieving data, computing bill splits, and sending payment requests to contacts.
- **Programming Language:** Python. This is a very good backend programming language since it is easy to use rest API. Python also has great libraries for ML which are useful for scanning receipts.
- **Major Libraries Used:** We will use DJango as the framework for the backend. This library uses python as the language. It has good compatibility with react native and uses Restful API. This will allow us to use the google OCR library for image processing and a payment processor like checkbook.io.
- **Testing Methodology:** We will write unit tests for the backend functions and mock requests for the API.
- **Interactions with Other Components:** This will be the middle layer between the frontend and database. The frontend will call backend functions to query the database and perform different functions. For example, after the user scans a receipt, that information will be sent to the backend, processed, and recorded in the database and can be queried at a later time.

Database:

- **Functionality:** This will contain all of the necessary data such as user log-in, receipts saved, and other necessary information.
- **Programming Language:** SQL. This is a stable language for querying information and is suitable for the scale of this project.
- **Major Libraries Used:** This will be run using MySQL. This is an open source relational database software that can be easily run on any cloud service or even a local machine. This is also natively compatible with Django and we will use mysqlclient as our driver.
- **Testing Methodology:** We will run a set of test queries and ensure that the database transactions match up to the intended behavior.
- **Interactions with Other Components:** The interaction structure will follow the MVC model. The frontend which will be implemented in react will display Views for the user. We will also create components that will be the Controller for the user. The frontend will interact with the backend which will manipulate the database, query it, and update the view for the user accordingly.



Continuous Integration:

- **Testing Library:** Will use pytest for the backend and the React Native Testing Library for the frontend.
- **Style Guide:** Use Black for python and Prettier for react native
- **Test Coverage:** Use coverage.py for python
- **Pull Request Workflow:** For pull requests, the other teammates can be added as reviewers to the PR and the first person to review it can simply approve it. When there are merge conflicts, teammates should meet and analyze the changes to make a final decision. If there are conflicting PR's we will sit down as a team and determine which one better suits and possibly join parts of them. Ultimately, PR's should be dealt with on a case by case basis.

Weekly Planning

- Week 1: Get shell of an app working with ReactNative, start learning all new skills needed. Finalize all of the decisions to use or not use certain libraries that are outlined in project proposal.
- Week 2: Design database architecture and discuss layout and normalization, Implement an instance of MySQL in google cloud and fill it with some dummy data.
- Week 3: Create a figma wireframe mockup of the important frontend views such as a log in page, a page to upload receipts, and a place to view debts, etc. Begin converting the log-in and account creation wireframes into react, writing tests in jest, and connecting them to the database.

- Week 4: Use the OCR library to create a function that can scan a receipt and format the contents in a uniform manner. Create a test suite with sample receipts and expected output.
- Week 5: Connect receipt scanning function to the wireframe view and write tests in jest for the frontend and in coverage.py for the backend. Finalize the UI layout and wireframes for the rest of the application including allowing users to select friends they want to split bills with.
- Week 6: Convert all wireframes to react components and work on backend functions like sending a link to pay for a balance. Also use jest to write unit tests for the frontend components.
- Week 7: Finalize frontend and have user log-in, receipt scanning, and sending out links to pay working. Run coverage.py for backend and develop tests where there are holes.
- Week 8: Run app on to a variety of virtual and real devices and adjust formatting to best fit many devices with differing screen sizes and shapes. Finalize any tests by running coverage.py and running app and codebase by reviewers and mentors.

Potential Risks

- Learning and becoming competent with languages we aren't strong in is difficult
  - Make an effort to use extra personal time to learn languages and libraries. Possibly pushed certain people back a week but other members can help also to catch them up.
- Using native features like the camera may be a difficulty in the front end. Since we are using react native and making the app cross platform, different devices will take pictures at different aspect ratios and resolutions.
- Building the backend algorithm to recognize text and numbers from a receipt will be challenging. We will need to do plenty of testing and have a clear cut strategy for handling failed cases built into the app.
- Since this is an ambitious project, the scheduling is very tight. We know that the core features are for a user to be able to log in, scan a receipt, and send a link to pay to their friends. If we are behind schedule we will prioritize completing these three goals over any other features. If we are struggling beyond schedule to allow users to scan a receipt image, we will change the frontend to allow users to manually input the receipts instead.

Teamwork

- The team will meet at least once a week. This will help to keep everyone on the same page and handle PR reviews. We will use discord so that we can separate the various components of the project into different channels and so that we can communicate with ease. Since our team is composed of people who are meeting each other for the first time, it's important to lay a strong foundation for communication. This will make sure that everyone knows what they are supposed to do and what everyone else is also doing.
- We will distribute tasks on a voluntary basis. This is because everyone has similar experience across the board and it would be good to dive into some of the libraries and languages used so that everyone can find what they are best at doing.