

Budget Royale

Design Document

Team 9:

Katelyn Chen

Samuel Duprey

Andrew Liu

Deepika Ramesh

Table of Contents

| | |
|--|-----------|
| Purpose | 2 |
| Functional Requirements | 2 |
| Non-Functional Requirements | 4 |
| Design Outline | 5 |
| High level Overview | 5 |
| Protocol, Service, and Hosting Overview | 6 |
| Activity Flow Map | 7 |
| Design Issues | 8 |
| Functional Issues | 8 |
| Non-Functional Issues | 9 |
| Design Details | 11 |
| Class Design | 11 |
| Description of Classes and Interaction between Classes | 12 |
| Database Design | 14 |
| Description of Database Diagram | 14 |
| Sequence Diagrams | 16 |
| Sequence events: Logging In | 16 |
| Sequence Events: Creating Your First Personal Goal | 17 |
| Sequence Events: Inputting Spending Information | 18 |
| Sequence Events: Editing a Group Goal | 19 |
| UI Mockups | 20 |
| Example Nav Bar on User Page | 20 |
| Leaderboard | 21 |
| Input Transaction | 22 |
| View Transactions | 23 |

Purpose

Budgeting is an essential life skill, but factors such as a lack of financial literacy and minimal motivation can make it difficult for emerging adults to properly take care of their finances. Over the past couple years, the average debt per person in the United States has risen from \$50,090 in 2018 to \$59,580 in 2023 according to Business Insider. 75% of American teenagers believe they have a lack of knowledge in personal finance and 23% of adults ages 18-29 have debt on their credit cards 90 days overdue according to Annuity.org. Budget Royale aims to create a platform to help emerging adults properly manage their finances. The added feature of competition will incentivize users to compete with their friends, take better care of their finances, and come back to the platform.

While there are services to help with budgeting such as Goodbudget, PocketGuard, and Mint, they lack the user engagement to create consistent and long-term budgeting. Goodbudget allows the user to sync their goals with more than one person. PocketGuard focuses more on bills, and helps users keep track of budgeting their income to pay recurring bills. Mint links cash accounts, credit cards, and investments to one account to track progress.

While these platforms allow users to create financial goals and track their spending, they don't really incentivize users to come back. It's easy for a user to be motivated one week and track their spending, and then the next week be lazy to input their spending and goals. That's what sets apart Budget Royale from other budgeting apps. Budget Royale adds an element of competition using data analysis that makes a task like money management more fun. With Budget Royale, users can input and track savings, income, expenditures, and goals in a centralized location, as well as join groups to compete with friends or users on the application, fostering competition among community members.

Functional Requirements

1. Logging In

As a user,

- a. I want to be able to register for a Budget Royale account.
- b. I want to be able to login to my Budget Royale account.
- c. I would like to be able to reset my password if forgotten.

2. Easy Access to Budget Royale

As a user,

- a. I would like to be able to access Budget Royale both on mobile and non-mobile devices.
- b. I want an easy to remember domain name to access Budget Royale.
- c. I want to be able to send feedback to the developers through a form.

- d. I would like to have access to a navigation bar to be able to toggle between the different pages.
- 3. Inputting data
 - As a user,
 - a. I would like to be able to input my spendings by category.
 - b. I would like to be able to specify the currency I am using.
 - c. I would like to be able to edit and delete my financial goals.
 - d. I would like to create custom categories of spending.
 - e. I would like to input my financial goals.
- 4. Analytics
 - As a user,
 - a. I would like to view a breakdown of my spendings by category of transactions.
 - b. I would like to view my progress percentage to achieve a financial goal.
 - c. I would like to view analytics of my cumulative spending from week to week.
 - d. I would like to be able to view how much I saved in total per week.
 - e. I would like to view my statistics on a dashboard.
 - f. I would like to see a category breakdown of my cumulative spending for the week.
 - g. I would like to be able to toggle on and off specific categories on the analytics dashboard to be able see more specific information about my spending.
 - h. I would like to view analytics of my cumulative savings to see any trends.
 - i. I would like to view graphs of my cumulative spending and saving to see any trends.
 - j. I would like to export my graphs.
- 5. Creating competition groups of users
 - As a user,
 - a. I would like to be able to create a group to compete with.
 - b. I would like to join or search for a group to compete with.
 - As a group admin,
 - a. I would like to create custom group goals.
 - b. I would like to be able to edit and delete group goals.
 - c. I would like to be able to add and remove group members.
- 6. Keeping users motivated to come back
 - As a user,
 - a. I would like to be able to win badges or icons from competitions.
 - b. I would like to be able to view the badges/icons on a dashboard
 - c. I would like the loser from the previous week to have an L next to their username on the scoreboard.
 - d. I would like to see a scoreboard for the group.
 - e. I would like to be able to opt out of the competition features.

7. Invites

As a user,

- a. I would like to be able to add other users as my friends.

As a group admin,

- a. I would like to generate invitational QR codes for my group.
- b. I would like to be able to generate an invite link for my group.

8. Notifications

As a user,

- a. I would like the option to enable weekly emails regarding Budget Royale.

Non-Functional Requirements

1. Performance

As a developer,

- a. I would like to have our website up 24 hours a day to support users at any time and in any time zone.
- b. I would like to support the large number of users the application is built for, with 100 simultaneous requests

2. Server

As a developer

- a. I would like the server to be able to support real time user activities.
- b. I want the database to be able to store information like a user's login details, financial goals, spendings, and savings.
- c. I want to be able to get information from the database to create analytics and visualizations for the user.
- d. I want to be able to add information that the user inputs or reflect user actions into the database.

3. Security

As a developer,

- a. I would like to use SSL encryption through PostgreSQL.
- b. I would like to use options such as remote access, user authentication, and configuring allowed hosts.
- c. I would like to create options that allow users to choose what information will be shared to other users on the platform.

4. Design

As a developer,

- a. I would like to easily build project source files to deploy from our Git repository.

5. Usability

As a developer,

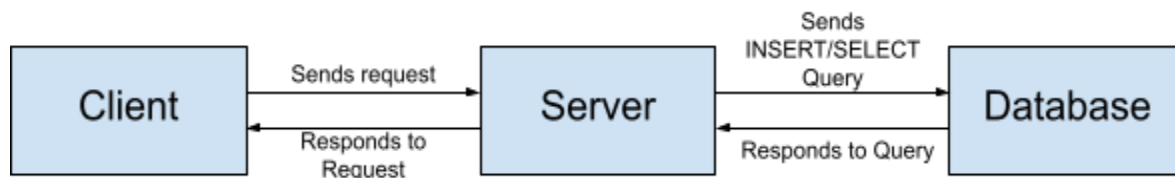
- a. I would like the application to be pleasing to look at.

- b. I would like the application to be easily navigable.

Design Outline

High level Overview

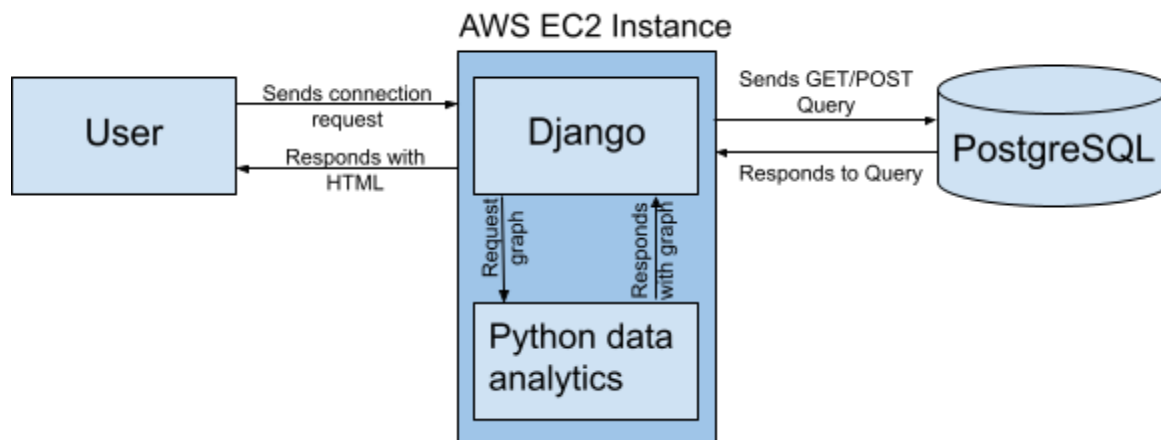
This project will be hosted on a website, allowing multiple clients to be interacting with the server at the same time. When the user interacts with the client (eg. inputting an expense), the client will send a request to the server. In order to fulfill the request from the client, the server will send a INSERT/SELECT query to the database (eg. adding the expense to the database). The database will fulfill the query from the server and then send the updated data back to the server.



1. Client
 - a. The client is where the user will interact with our system.
 - b. The client will send requests to the server.
 - c. Tables, user analytics, and interactive components are displayed to clients through UI features.
2. Server
 - a. The server handles requests from the client.
 - b. The server will transfer information on user spending, saving, and progress to appropriate relational database tables.
 - c. The server calls SQL inserts or selects to read and write to the database
 - d. Information from the database is used to create analytics and visualizations for the user.
 - e. The server will respond to requests from the client.
3. Database
 - a. The database stores information like a user's login details, financial goals, spendings, and savings.
 - b. User inputs or user actions are reflected in the database through the server.

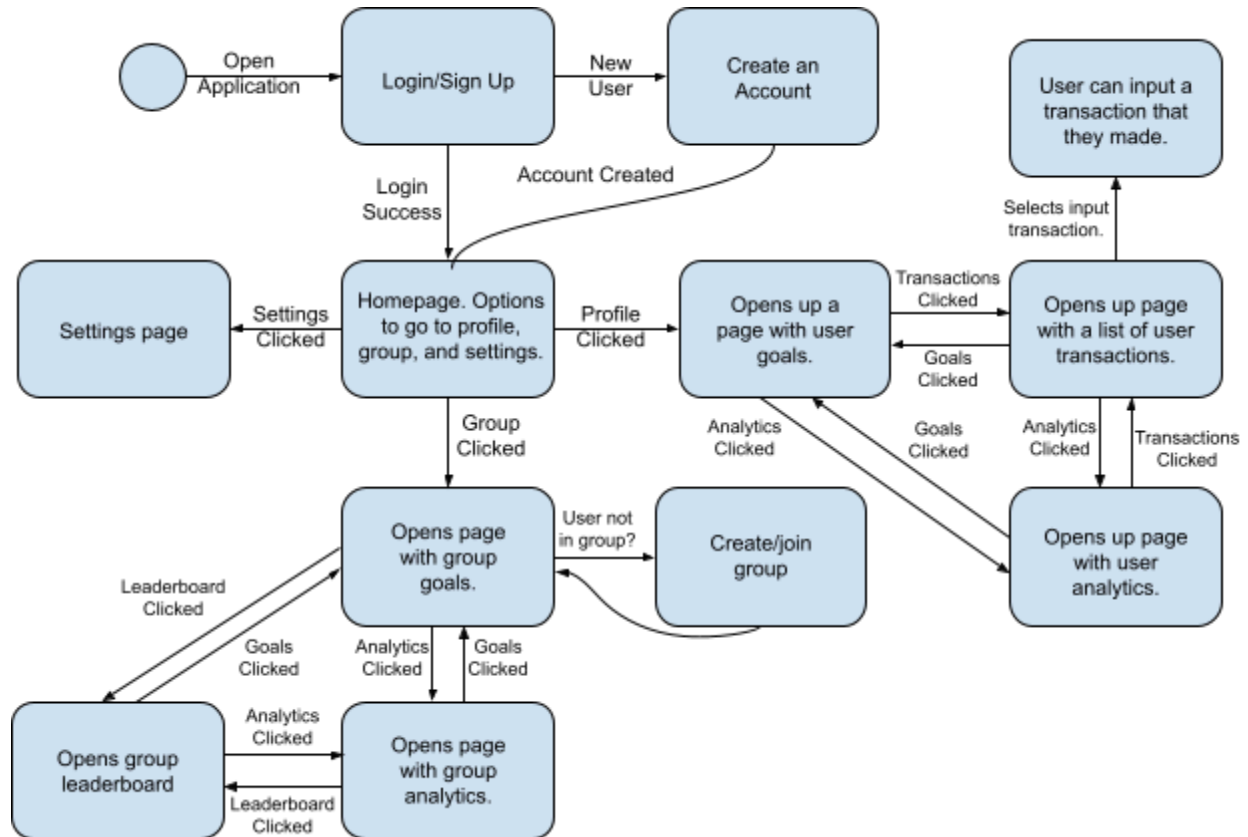
Protocol, Service, and Hosting Overview

Using Django as our web framework helps us to streamline the process of displaying our data. We are using python for our graph creation and backend so we are using Django to easily connect to our code. We have set up Jenkins and Docker to utilize continuous delivery of our project. When we push our code to main, our website is automatically updated after successfully building. When pushed to GitHub, Jenkins recognizes this and builds the image from the Docker file. On success, it will deploy to our AWS instance. The Dockerized Django application runs on AWS's EC2, which are virtual machines hosted on AWS's cloud. The database we use to store user information is AWS RDS (Relational Database Service), which is run on the PostgreSQL engine. The Django application and the database is connected via an endpoint. Through SQL queries and built in Django methods, we can read and write user information.



Activity Flow Map

The design of the application is intended to be simple, while consolidating different pages to be accessible through navigation bars. Users must have an account, so they can log in or sign up with an email account, username, and password. The homepage has options to access profile, group, and settings. Each of these pages has navigation bars, and options which can be clicked on to access the different features of the application. These features include inputting and viewing transactions, viewing graphs and analytics, as well as creating goals.



Design Issues

Functional Issues

What information does a user need to create an account?

- Option 1: Username + Password
- **Option 2: Username + Email + Password**

Decision: We need usernames and passwords to give users unique identities, which they can use to join groups. We decided to include emails for verification or resetting a password, and a mode of contact through which users can be added to groups. The emails will act as an unique key we can use inside the databases to identify specific users in queries.

What categories of spending should the users be able to pick from?

- Option 1: A pre-set list of categories
- Option 2: The user will custom make all their categories
- **Option 3: A pre-set list of categories with an option to make custom categories**

Decision: We wanted to create a generic list of categories for users to pick from such as groceries, entertainment, housing, and insurance so users didn't need to worry about creating custom categories. However, different users have different needs. For example, a user may have a pet snake so they may want to create a spending category for expenses towards their snake. By giving both options, we give the user more flexibility when it comes to tracking their spendings.

How should the content for Budget Royale be displayed?

- Option 1: One page which displays graphs and analytics
- **Option 2: Multiple pages that the user can access through a nav bar**

Decision: It could take a long time for the user to find the specific graph they might be looking for, and would also make the page look overcrowded if everything was placed on one page.

Creating multiple pages, accessible through a nav bar, will make it easier for users to find what they're looking for and make each page less cluttered.

How will results from a competition be displayed?

- **Option 1: Badges/Icons**
- Option 2: Email notifications

Decision: Email notifications could be annoying to receive on a weekly/monthly basis. Using badges and icons on a scoreboard would indicate progress in a way that is appealing to the user. This will also help keep the user motivated and give them something to strive towards.

Does the user need to join a competition group?

- **Option 1: The user can opt in**
- Option 2: Yes

Decision: Some users may not want other people to view their financial data, and they may just want to use our application as a financial planning tool. We allow the users to opt in to competition features so it gives users more flexibility when it comes to our application.

Should users be able to have multiple financial goals?

- Option 1: No
- **Option 2: Yes**

Decision: Initially, we thought that each user would have a specific budgetary goal in mind when they joined the application, and would follow this spending or saving goal when they tracked their transactions. However, we realized that by keeping track of spending and saving goals, the user would be able to quantify their transactions, as well as label them under categories.

What functionality should a user be able to do regarding their transactions?

- Option 1: Creating a transaction
- Option 2: Creating/editing a transaction
- Option 3: Creating/deleting a transaction
- **Option 4: Creating/editing/deleting a transaction**

Decision: We wanted users to have as much flexibility as possible when it came to their transactions. We wanted to take into account that users may have typos when it comes to adding transactions. We didn't want users to have to recreate their whole transaction if they wanted to change something. Additionally, users may want to delete transactions if they accidentally create one or for some other reason. So, we decided that allowing users to both create, edit, and delete transactions would be the best for the user.

Non-Functional Issues

What backend language should we use?

- Option 1: Java
- Option 2: C
- **Option 3: Python**

Decision: Although all our team members had worked with Java and C in our classes at Purdue, a main part of our project involved data analysis and visualization. Python has a lot of built-in libraries such as Plotly for generating graphs based on data and Pandas for data analytics so we decided it would be the best fit for our project.

What web framework should we use?

- **Option 1: Django**
- Option 2: React

Decision: All of our team members had worked with React before, so this was the first web framework that popped to mind when we were considering which web framework to use.

However, displaying graphs and data analytics is an important component of our web application. Django works better with Python compared with React, and Python's built-in libraries were more useful to our application than JavaScript. Through the simple use of HTTP requests, the Django and graph generation python scripts can be run on the same EC2 virtual machine, allowing for a faster implementation between the two components.

What database should we use?

- Option 1: SQLite
- **Option 2: PostgreSQL**
- Option 3 : MongoDB

Decision: For our application we needed a database in order to store user data for our project. We first needed to decide if we needed a relational database or document based. We decided that our application would work better with a relationship database, as we need to store information about competition groups and each user's data. This eliminated MongoDB and other NoSQL options from our pool of choices. The database should also be hosted on a separate server and machine than our Django application, so that a downtime on the application doesn't jeopardize the database. If we wanted to expand, multiple AWS EC2 instances can be connected to one AWS RDS. RDS has built-in PostgreSQL options. We also really appreciated PostgreSQL's security features and support for complex queries. Through heavy consideration, we found PostgreSQL to be the option that best suited our project.

How should we host our application?

- Option 1: Github Pages
- **Option 2: AWS**
- Option 3 : Localhost

Decision: we wanted to be able to have users from other devices all connect to our website so we decided to host it on an external site instead of localhost. We decided on AWS to host our application because of the streamlined process of connecting it to our client and database. It also allows easy setup of continuous delivery through Jenkins so we can connect our repository to build through Jenkins and automatically deploy to our AWS instance.

What devices should the user be able to access our application from?

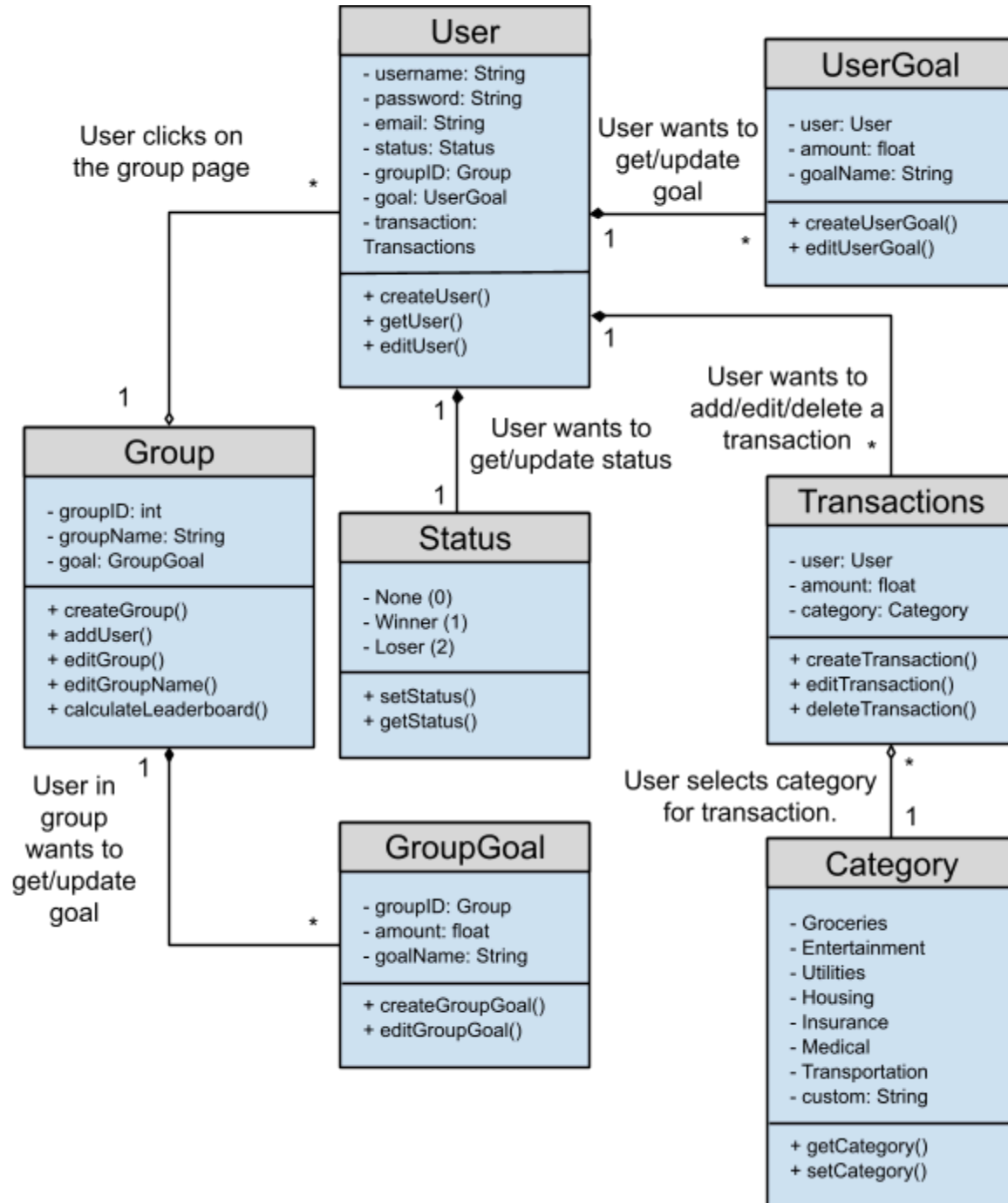
- Option 1: Computer
- Option 2: Mobile Device
- **Option 3: Both a computer and a mobile device**

Decision:

We want Budget Royale to be as accessible to the user as possible. Since our target audience is emerging adults, we know many of them use their phones just as much as their computers. Because of this, we decided that we wanted to create a website so it's usable both on a computer and on a person's mobile device.

Design Details

Class Design



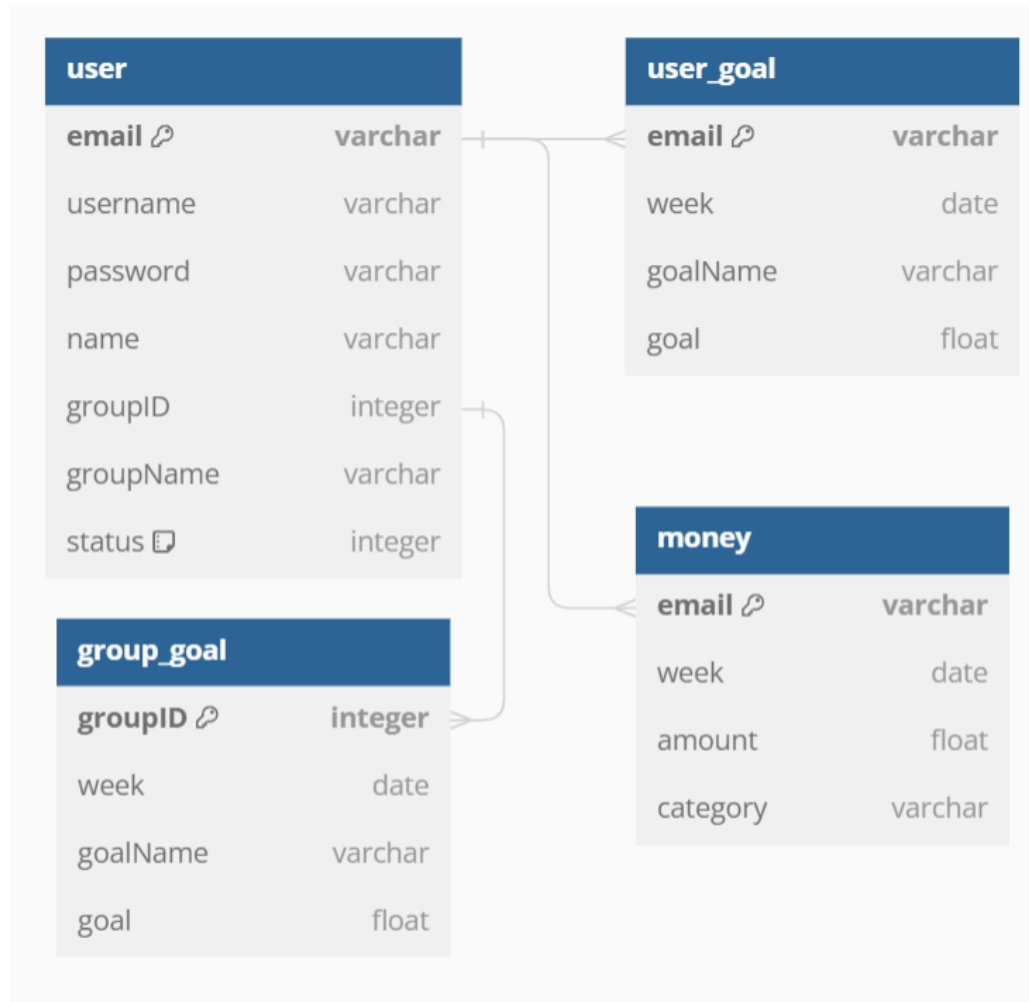
Description of Classes and Interaction between Classes

The classes are designed based on functions and databases in our application. Each class has a list of attributes linked to each object.

- User
 - A user is created when a person creates an account with Budget Royale.
 - When a user creates an account, they will input their email, username, and password. Their email will be used for notifications and for resetting their password if they have forgotten it.
 - A user's status indicates if they were at the top or bottom of the leaderboard of their group for the past week. If they were at the top, their status would be Winner (1). If they were at the bottom, their status would be Loser (2). If it's none of the above, their status will be None(0).
 - The user's groupID will indicate what group they have joined. Each group has a unique groupID. If the user hasn't joined a group yet, their groupID will be 0.
 - The user's transactions will be the transactions that the user has done for the week. This includes both expenses and income.
- Status
 - Each status is created when a user is created.
 - There are 3 types of status that the user can have: None (0), Winner (1), and Loser (2)
 - By default, the user's status is 0 - this is if they aren't in a group, or the first week of their competition.
 - As the competition progresses, status of winner and loser are assigned to users based on their analytics and progress.
 - People in a group who are not a winner (1) or loser (2) are assigned 0 as status.
 - Status determines whether the user will have a badge of a trophy (winner) or a L (loser) on their leaderboard.
- UserGoal
 - A user can create a spending/saving goal.
 - These goals include a target that they want to reach (amount) stored as a float.
 - The goal can also have a name (goalName), which details what the user wants to budget their money for.
- Transactions
 - Each transaction will be linked to the user through their email.
 - The user will input the amount that they spent/earned in amount.
 - The user will indicate which category their transaction pertains to.
- Category
 - The user will be given a list of categories to choose from for their transaction. The pre-set options are groceries, entertainment, utilities, housing, insurance, medical, and transportation.

- The user can also choose to create a custom goal which they can input.
- Group
 - A user can join/create a group along with other users.
 - Each group will be assigned a unique groupID for a primary key, to make groups easily distinguishable on the server side.
 - Groups can also be named by the users in the group
 - Users in the group can also create spending/saving group goals to be followed by the group.
 - A user can add another user to the group by sending an invitational QR code: this will employ the addGroup() feature.
 - The calculateLeaderboard() function creates a ranking of the users in the group, based on how well they are working towards the group goal for the week.
- GroupGoal
 - The group goals will be connected by the groupID to indicate which group it is for.
 - A user in the group can add/edit a group goal. When they do that, they need to specify the amount that they want for their goal which will be inputted as a float.
 - The user can also name the group goal which will be stored as a string.

Database Design



Description of Database Diagram

We plan on using PostgreSQL for our database.

- User
 - A user is created when a person creates an account with Budget Royale
 - When they sign up for an account, they input a username, email and password. The email is used for weekly updates and for resetting the password if the user forgot their password.
 - When the user signs up for an account, their default groupID is 0, meaning they haven't joined a group. Once they join a group, the groupID will be changed to reflect the group that they joined.

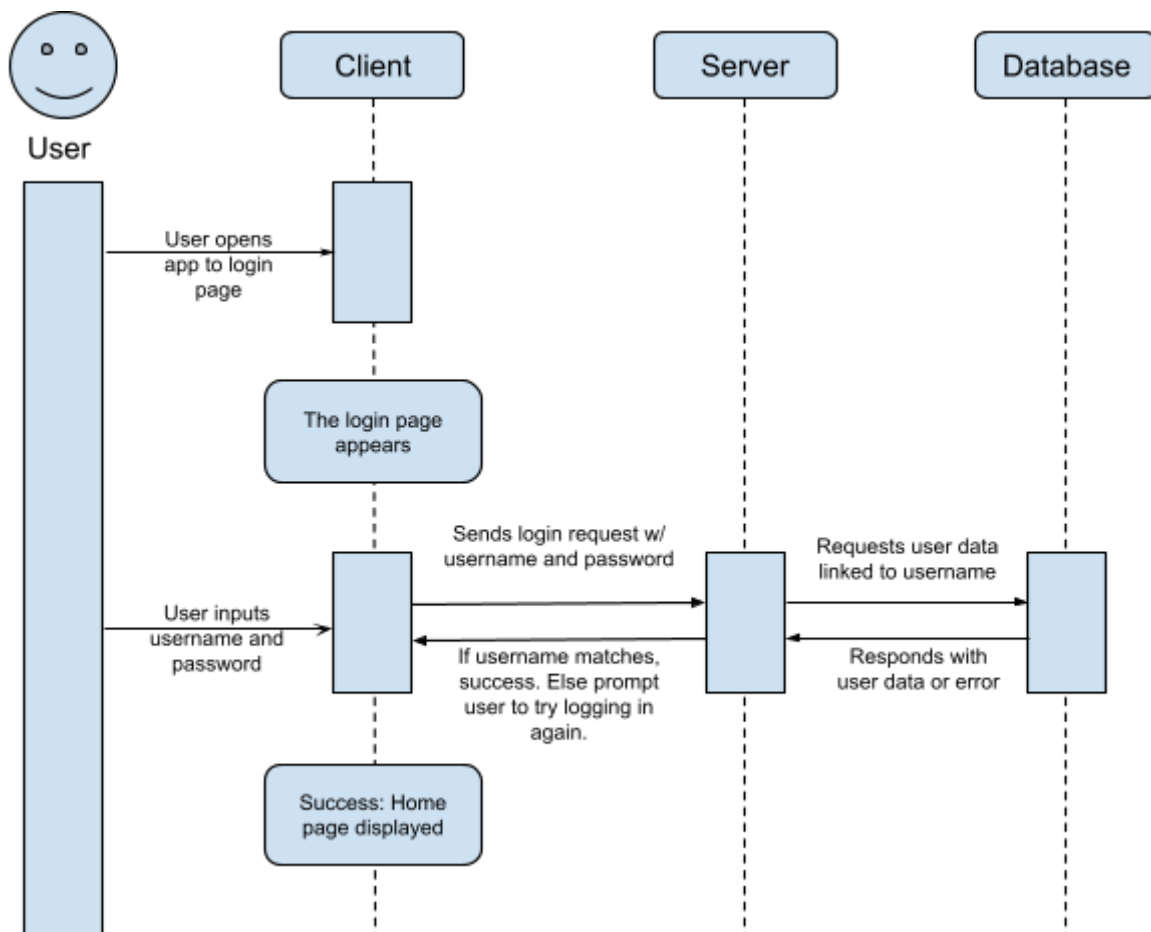
- The group name is also stored for users to personalize their groups to create a fun and more engaging experience with their friends.
- When a user signs up for an account, their default status is 0. If a user joins a group, that's when their status can change. If they are at the top of the leaderboard for the past week, their status will be 1 (winner) and they will have a crown next to their name for the next week. If they are at the bottom of the leaderboard for the past week, their status will be 2 (loser) and they will have a "L" next to their name for the next week.
- Money
 - When the user wants to input their savings or spendings, their email, which is their unique identifier, is associated with the transaction and will be used to connect it to their specific account.
 - When they log a transaction (either spending an amount or saving an amount), it tracks the week it was logged in by storing the date of the Sunday at the start of the week to group each transaction to the right time.
 - Each input, either savings or spendings needs a dollar amount to show the money being saved or spent to go towards the user's goal.
 - Each transaction needs a category to associate this with to allow the user to show specific spending to see what they spend the most on. This information is important for a variety of statistics showing overall spending for the week and displaying by category. If no category is included, it defaults to an empty category which will be all the extra transactions grouped into.
- User Goal
 - When the user creates a goal, the goal needs to be associated with the user so it contains the user email.
 - Goals are created for the upcoming week and last until the next week resets it. After the week it was created is over and the application resets, the goal progress will automatically reset. The user will have the option to delete or edit the goal in their profile when they log on again if they want to change their goals for the next week.
 - The goal is an amount of money that the user is trying to save or spend. As user transactions are imputed, the progress of the goal is updated based on the amount the user enters. The information inputted by the user throughout the week updates the progress of the goal.
 - The user can name their goal as well to personalize what they are saving for or limiting their spending on. This way they can add a personal touch to the goal and be more engaged in completing it.
- Group Goal
 - When a group goal is created, it is attached to a group through the group ID. The group ID is used to track which group the goal relates to.

- Similar to a user goal, group goals are created for the week and go until the next week resets them at which point the user can modify or delete any goals. Each user in the group can see the progress towards the group goal.
- The goal has an amount that represents the total spent or saved. The progress is calculated from the transactions of the users in the group. The progress is reset at the end of the week as well.
- The group members are able to set a name to the group goal to personalize it. This makes it more interesting and engaging and gives the users a more concrete idea and reason to reach their goal.

Sequence Diagrams

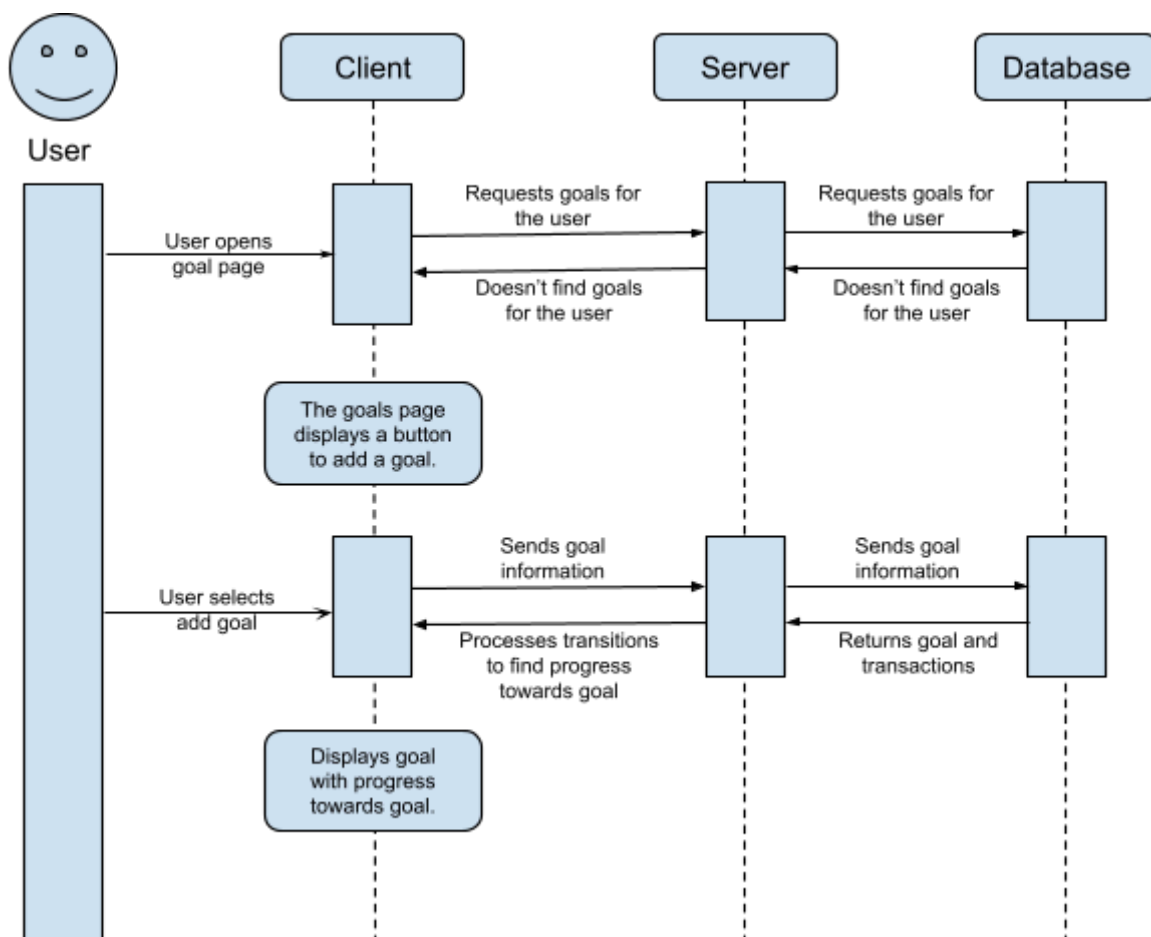
The following diagrams show the sequence of some events that the user will follow as they use the application, including logging in to the application, creating their first personal goal, and inputting spending information. These sequences use the client-server-database model, and show how queries are used on the backend depending on user interactions with the application.

Sequence events: Logging In



This sequence describes how a user who has an account logs into the application. The user opens the website to log in, and the login page appears. At this point, the user inputs their name and password. The client sends this to the server, which sends a request to verify if this matches with information in the database. The database responds with the appropriate response of user data, or an error if no match is found. If user data is returned, the client displays the homepage. Else, the login page is displayed again, and the cycle repeats till the user data is found and success is returned.

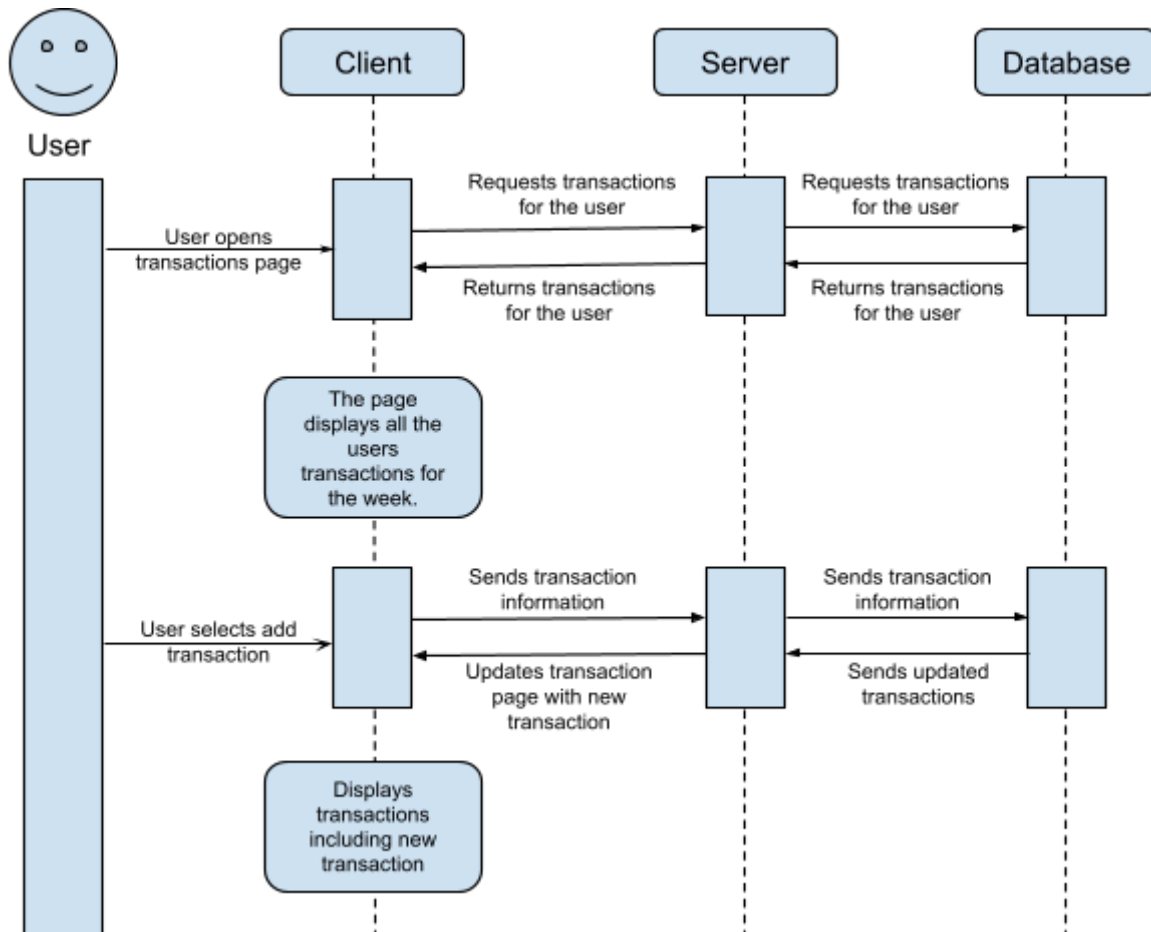
Sequence Events: Creating Your First Personal Goal



This diagram shows the sequence of creating your first personal goal and all the relationships between the client, server, and database and how information is passed between them. This process starts with the user opening the goal page. This will then call the server to request the information of the goals of the user which will then query the database to return the information. If the user already has personal goals, it will return them, however in this sequence, the user does not have any goals, so the database does not find any previous goals and the server repeats this to

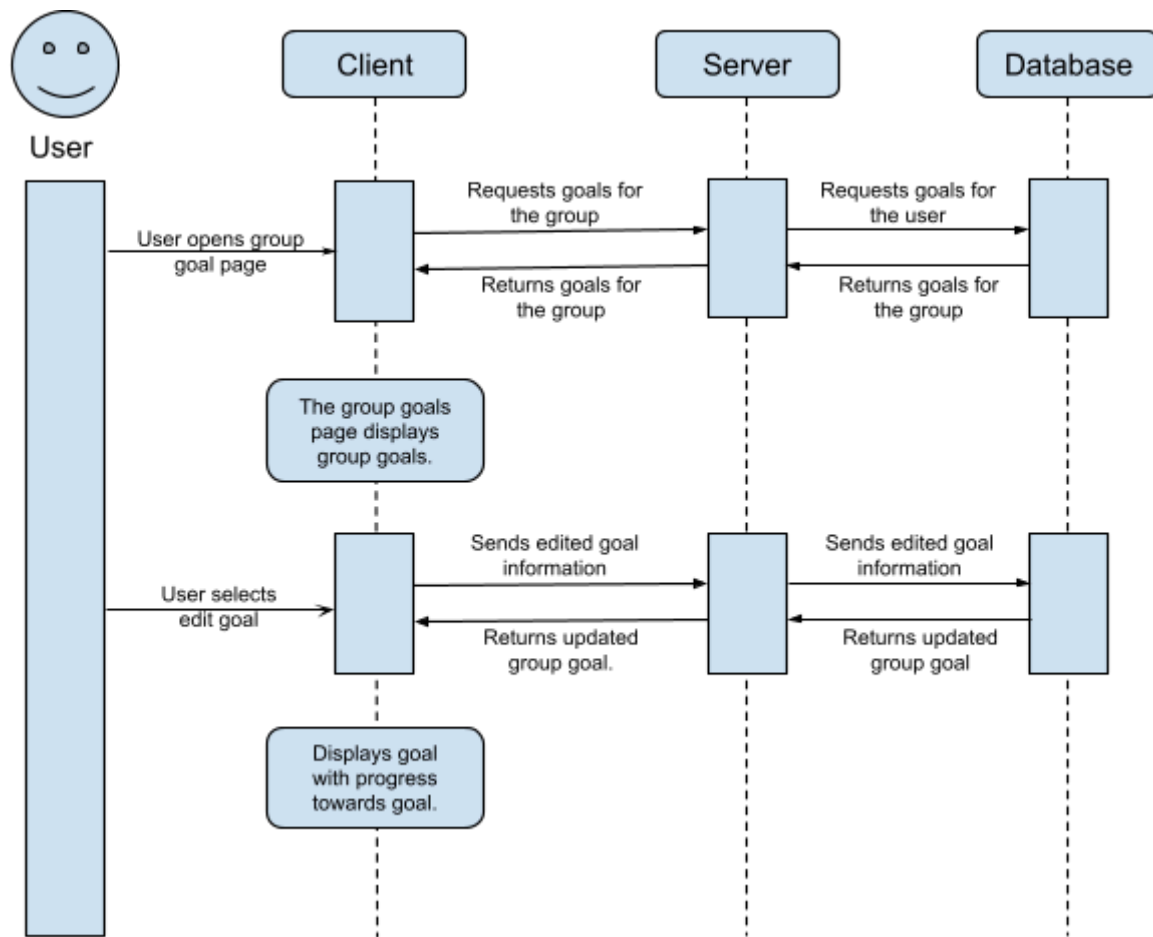
the client. From there, there is an add goal button which the user will then select. The user will input the information of their goal where it will be sent to the backend and then database for it to be stored. After the user creates the goal, the database returns the transactions for the server to calculate the weekly progress towards the goal just created. The client then displays this goal and the progress on their goal page.

Sequence Events: Inputting Spending Information



This sequence diagram shows the interactions between the client, server, and database when the user inputs spending information. In order to input information, the user must first be on the transactions page. When a user opens the page, this request is relayed to the server, which then calls queries on the database, selecting for this particular user's transactions from the database. If successful, the database will return the information back to the server, and be displayed on the client's side. Similarly, when the user adds transactions, this request is relayed to the server, an insert method is called to add the data to the database. The database will also return the user's data, reflecting the newly added information. This will be displayed to the client.

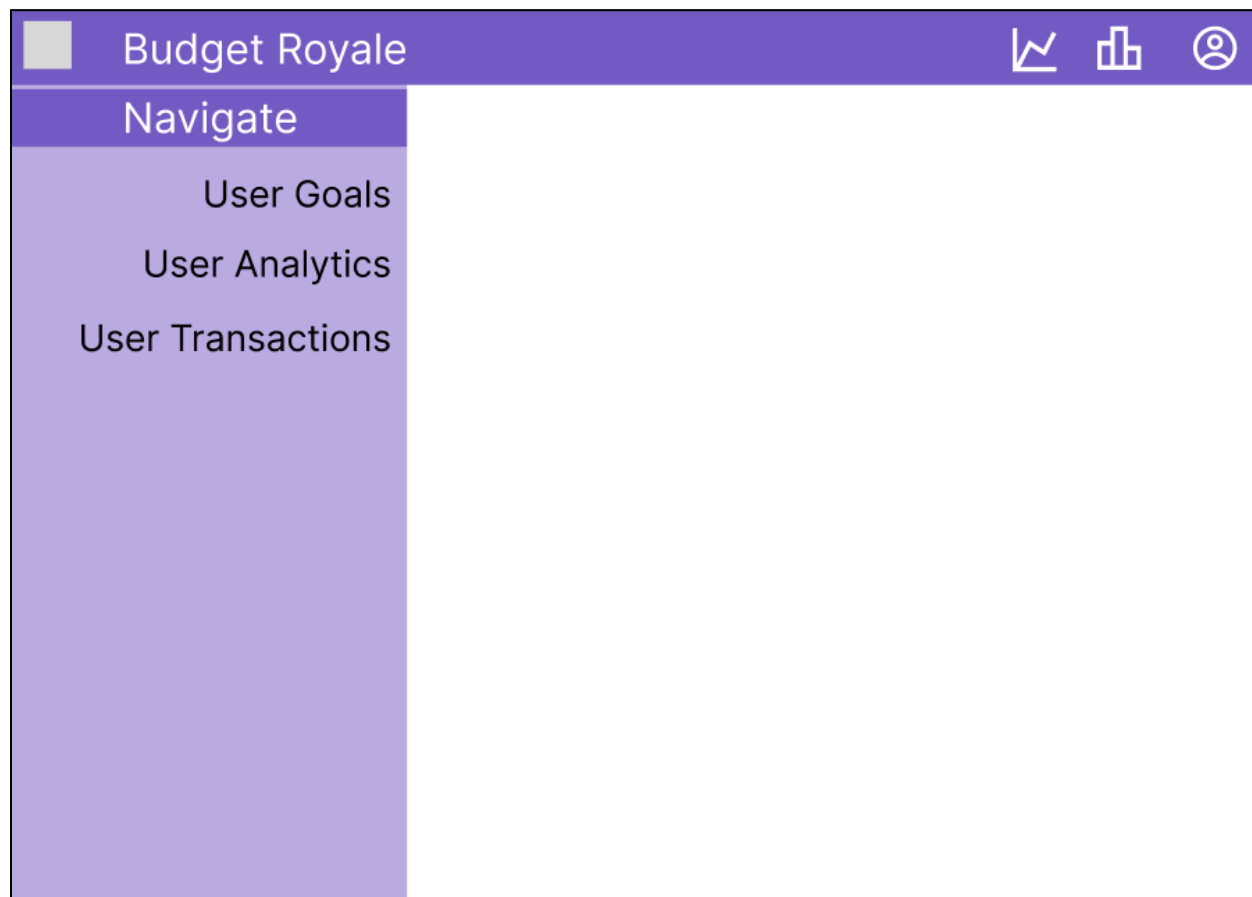
Sequence Events: Editing a Group Goal



This event is for a user editing a group goal. First, the user opens the page for the group goal. The client requests the group goal page information from the server and the server requests that information from the database. Once the database sends back that information, the group goals page will now display the current group goals. The user then selects the edit goal button on the client side. The client sends the edited goal information to the server which then sends it to the database. The database sends back the updated goal information and the updated goals page is now displayed on the client for the user.

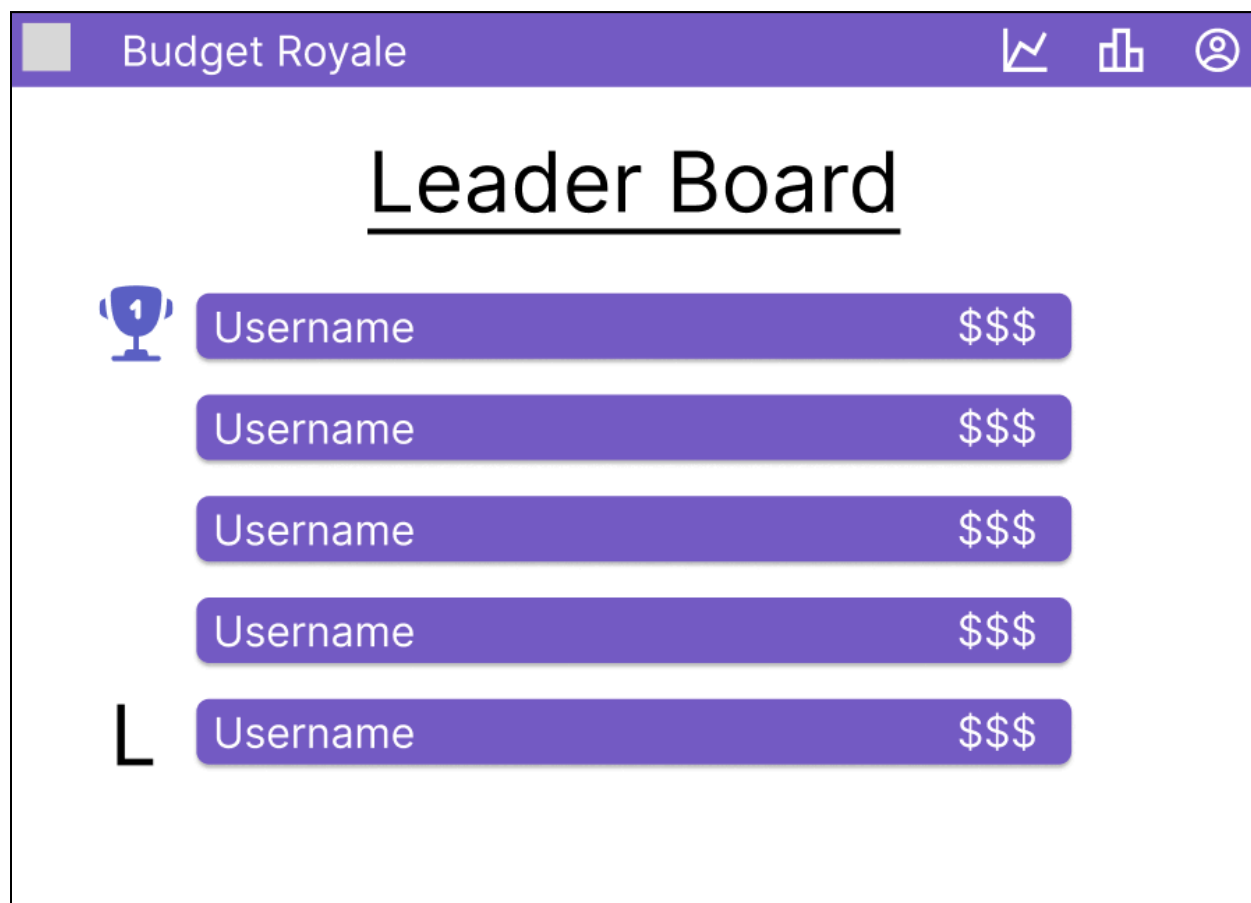
UI Mockups

Example Nav Bar on User Page




This is a potential page we could have, that routes to User Goals, User Analytics, and User Transactions. This page is reached when the user clicks on their profile, at the top right of the screen. The side nav bar is collapsable, and shows the different parts of a profile, like analytics, graphs, and inputting transactions.




Leaderboard



This is our general concept of the leaderboard page where all users in a group are ranked on their progress. The past week's loser is marked with an L and the last winner is marked by another icon.

Input Transaction

 Budget Royale



Transaction

spending

☒

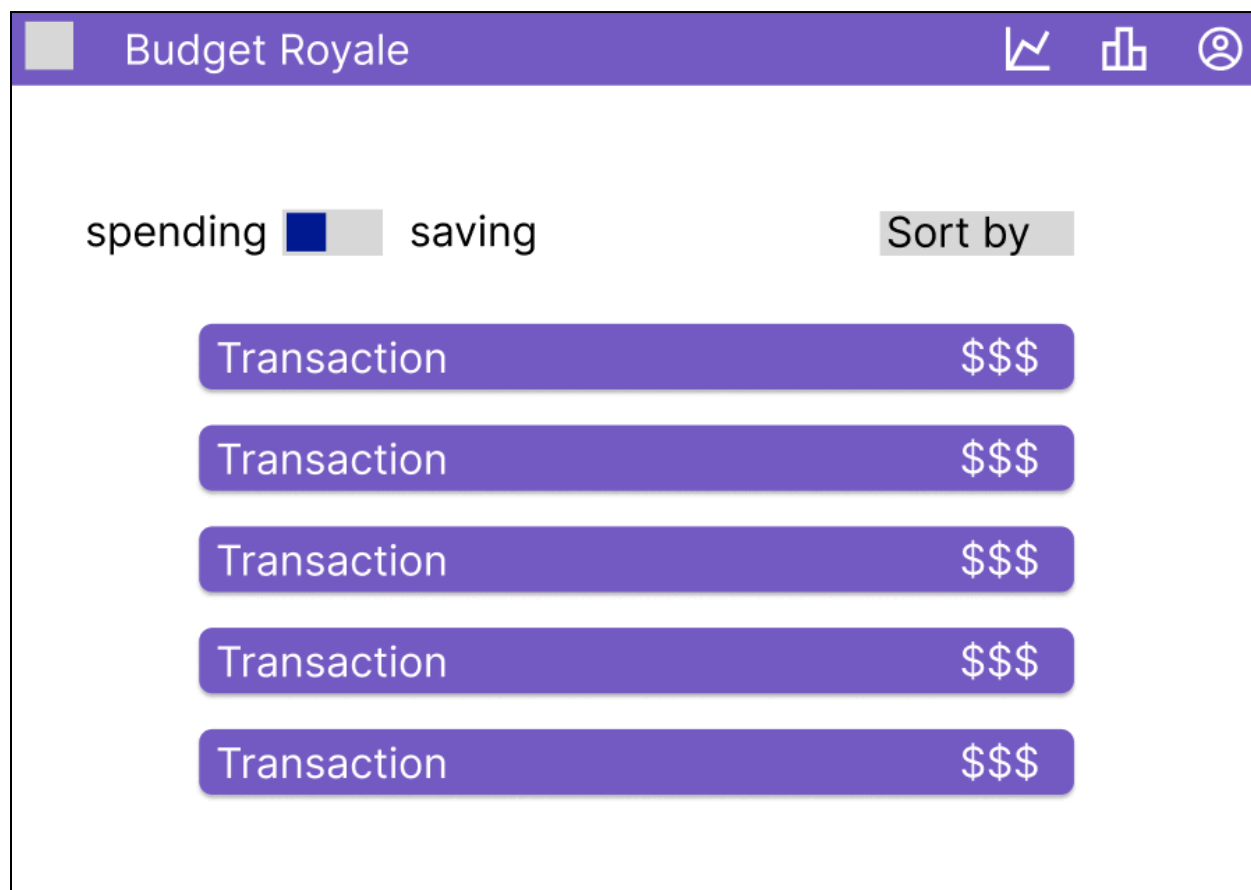
saving

Category

Amount

Users can input the transactions they have made, which will be inputted into the database and used to calculate and track their progress. Transactions can be in the form of spending/saving money, which can be toggled. A category can be chosen based on preset values, or a custom category can be created. The amount they spent/saved is inputted by the user, and pushed to the database.

View Transactions



This is a concept of our view transaction page where a user can see all the transactions they have imputed. This includes any money they gained and saved or have spent. They can see it by category as well or sort by different factors.