Team 14 - Project Backlog

Gabe Segura, Ben Lilley, Logan Cover, Wonho Lee, Ryan Rittner

Problem Statement

Between friends, roommates, and families, keeping track of who owes what, when this occurred, and trying to repay debts via equal value purchases can be difficult. BillMates aims to eliminate this confusion by providing a streamlined way to track and pay expenses on an individual and/or group basis, coordinate events and future expenses, and even help plan shopping trips.

Background Information

Audience:

Billmates aims to provide a painless way for students, professionals, world travelers, and roommates to split the cost of everyday purchases. This is especially useful for college students and young professionals who live with others on separate incomes, and want to make sure that bills are getting split fairly.

Modern Platforms and Competitors:

There already exist applications that handle monetary transactions and dividing financial responsibility in group and individual settings. Splitwise and SettleUp are finance tracking websites/apps that allow users to create groups and assign debts owed throughout an activity or trip. Venmo and Paypal are existing online payment services that allow users to transfer money from their bank accounts and keep a history of all transactions.

Limitations:

Although expense keeping websites already exist, none of them integrate a way to plan shopping trips, aggregate expenses across multiple events, and payout existing debts from your bank account all in one. Venmo and Paypal only allow you to make one payment at a time, and do not necessarily allow a user to keep an aggregate tab. Splitwise does not integrate a way to plan out shopping trips between families and roommates, or a calendar view of upcoming scheduled expenses. Our goal is to synthesize all these features into one application for user convenience.

Requirements (Backlog)

Functional:

1. As a user, I would like to register for my BillMates account.
2. As a user, I would like to sign in to my BillMates account.
3. As a user, I would like to manage my BillMates account.
4. As a user, I would like to have my account stay logged in (for a period of time) using a session cookie.
5. As a user, I would like to receive an email notification if an unauthorized user is trying to access my account
6. As a user, I would like to link my Venmo account with my BillMates account.
7. As a user, I would like to delete my BillMates account.
8. As a user, I would like to be able to change my password and account information.
9. As a user, I would like to create a BillMates group.
10. As a user, I would like to invite friends to my BillMates group with a link.
11. As a user, I would like to be able to join a BillMates group from a provided link.
12. As a group manager, I would like to be able to edit my BillMates group settings.
13. As a group manager, I would like to be able to delete my BillMates group.
14. As a group manager, I would like to be able to kick users from my BillMates group.
15. As a group manager, I would like to be able to archive my BillMates group.
16. As a user, I would like to be able to add a pre-set tag to my transaction to be used for analytics.
17. As a user, I would like to be able to submit an expense request.
18. As a user, I would like to be able to fulfill an expense request with Venmo.
19. As a user, I would like to be able to verify a completed expense request through BillMates.
20. As a user, I would like to be able to comment on an expense request fulfillment and submission so that users know what each expense request is related to.
21. As a user, I would like to be able to input the payment method used to fulfill an expense request.
22. As a user, I would like to be able to report an illegitimate expense request so that the group manager can review it.
23. As a group manager, I would like to be able to void an illegitimate or mistaken expense request.
24. As a user, I would like to include and disclude people from a group expense.
25. As a user, I would like to be able to assign specific amounts owed from a member in an expense request so that expenses can be split non-uniformly.
26. As a user, I would like to be able to create a shopping list in a group.
27. As a user, I would like to be able to add items and other group members to my shopping list.
28. As a user, I would like to have a “checkbox” to mark items in the shopping list that have been purchased
29. As a user, I would like to be able to schedule an event in a group.
30. As a user, I would like to be able to view group expense analytics.

31. As a user, I would like to be able to view a group calendar showing events and past transactions.
32. As a user, I would like to be able to have multiple groups, and change groups with ease
33. As a user, I would like to have a home screen that shows the total amount I owe among all the groups I am a part of.
34. As a user, I would like to be able to see individual analytics that show specifics about the history of my expenses.

Non-Functional:

Architecture

We intend to split the work between a frontend and backend team which will allow us to divide work more evenly amongst the group. Since it can take a different amount of time to create the same feature from a front or backend context, splitting our attention in this way allows us to work on multiple features while waiting for the other to finish. For the frontend, we will use NextJS for the framework. NextJS is a powerful javascript Framework based on React but has more added features such as Babel. Also, we will use Redux for state management. In terms of the backend, we plan to primarily use Python for API calls and data manipulation, with MongoDB as our primary database tool.

Security

Considering BillMates will have payment integration through Venmo, security is a critical aspect of our platform. Considering we will be using php to run our website, we intend to use Laravel, which contains features that helps prevent common malicious attacks that are directed towards websites, like SQL injections. Venmo integration on our platform will also require 2-factor authentication to ensure that no unauthorized users can access and modify user’s venmo accounts. We will also implement a notification system that will let a user know if an unauthorized user is trying to access their BillMates account. From here, the account owner will be able to change their password to prevent future successful attacks. Another security feature that we would like to add is to limit the people that have access to the API, which would be done by using a JWT token. To manage logged in users, we will keep users logged in for a period of time using session cookies.

Usability

The interface will be designed such that the average user will be able to quickly see expenses owed for every group that they are in. Additionally, navigation icons will be present to indicate where features such as the groups, analytics, shared calendar, and pay out will reside. Moreover, users will be able to link their Venmo account so that they will not have to use two different applications to accomplish our goal. In terms of device useability, we will focus our development on a web based application. The goal is for the site to be able to run 24/7, and handle at least 100 simultaneous requests as a beta test.

Deployment

Given that we can develop the front and backend separately, we plan to deploy each in their own respective processes as stated in above sections. The frontend will be deployed through Vercel since it provides its own NextJS deployment model. In terms of the backend, we plan to use AWS which provides easy integration with Python and PHP requests. To keep track of development, we plan to use git/GitHub to track changes and keep a history of features as they are added.

Scalability

MongoDB offers assistance in upscaling smaller processes/databases, allowing us to expand our user base as traffic increases. As well, AWS is able to easily handle upticks in requests as they come. This will allow for an easy and seamless transition into larger scale development.

Performance

Given that the platform is used by groups of people, it is likely that many people will be on the website at the same time. As a result, the website should be able to run 24/7, and handle at least 100 simultaneous requests as a beta test, with under 1 second of lag for normal requests, such as paying someone back, or inputting payment requests.