Better Boulder Buses

By:

Kian Feiz GITHUB: Kife1762, optix12

Arman Mokhlesi GITHUB: ArmanM1, ArmanM3

Reed Shisler GITHUB: ReedShis

Riley Rasizer GITHUB: rjrasizer

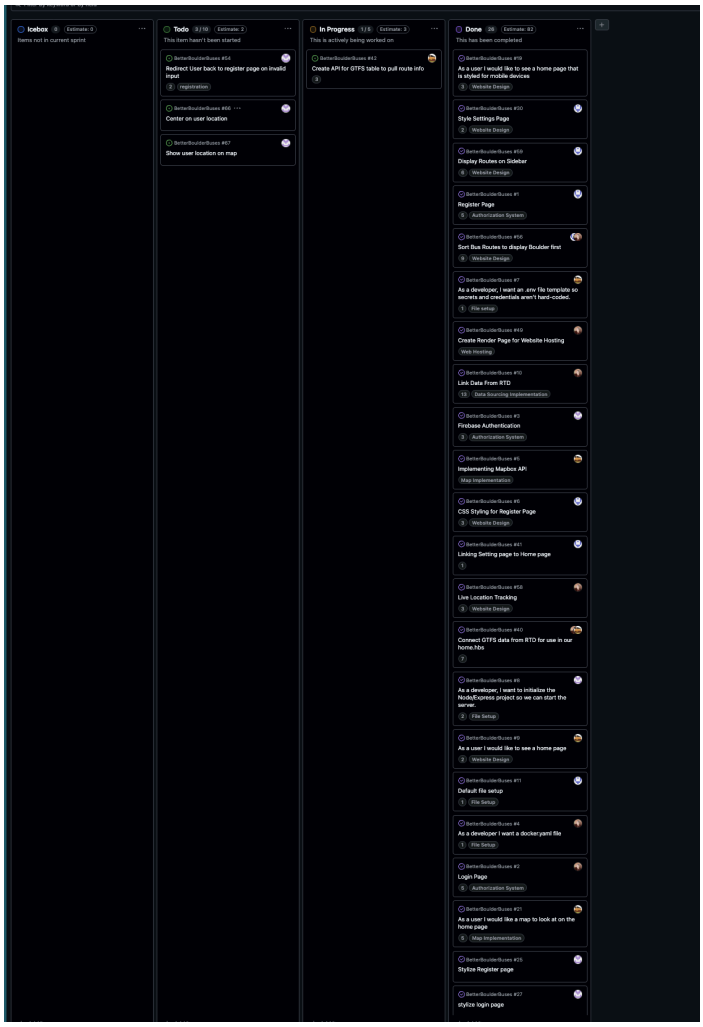
PROJECT DESCRIPTION

Better Boulder Buses is a free, Boulder-based public transportation website that allows users to view bus routes, live locations of buses, and bus schedules, along with route planning. A user is shown the nearest bus routes relative to their live location and can search for other routes in the Boulder area. One of the core features of the website is live bus time viewing, along with departure times for each stop. A user can intuitively find the bus route they want to take and find exactly when they need to be at the stop to catch that bus. To complement that feature, there is a closest stop walking directions implementation to show how a user can get to where they need to be.

The goal of Better Boulder Buses is to make transportation more accessible for students and residents of Boulder. It allows users to easily find how they can navigate around Boulder using public transportation and removes the confusion from bus times and stops.

PROJECT TRACKING

https://github.com/users/rjrasizer/projects/3



VCS
https://github.com/rjrasizer/BetterBoulderBuses

VIDEO

BetterBoulderBuses.mp4

PROJECT CONTRIBUTIONS

ARMAN - ArmanM1 and ArmanM3

I first created the initial wireframe and created the skeleton for the sign-up HBS files. I mainly focused on the backend, where I made the routes appear on the map. I implemented the mapbox api to display full-screen on our page. I additionally created a simple algorithm to calculate the estimated pose of the bus's location. Modifying the algorithm slightly, I also added the functionality to see what time the bus will appear at all other stops. Finally, I implemented zooming on the user, walking to bus routes, finding the closest bus route, and populated the sidebar with all bus routes, and I created a submenu which shows the times and other stops (all of which are clickable and give you the route to get to the stop).

KIAN - Github: optix12 and Kife1762

I developed and fully styled the login, register, and settings pages. Within the settings page, I implemented functionality to change user login information and access common FAQ about the website. I implemented a system to sort all bus routes by distance to the user and styled the sidebar to display these. I implemented a search system to find all routes pertaining to search queries. I styled buttons on the main page for a clean user user-intuitive design. I styled the logout page and routed to login/register pages, respectively. I developed the UAT test plan and carried out those tests to gather data on improvements the team should work on.

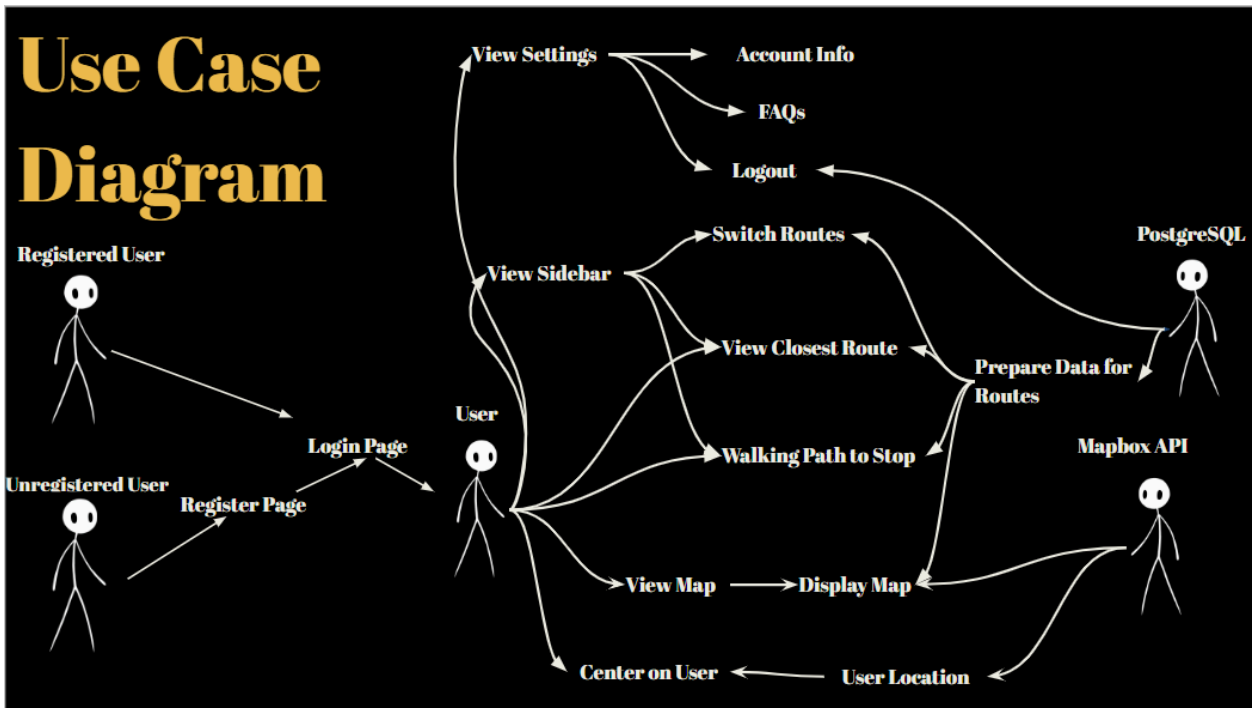
REED - Github: ReedShis

I tackled integrating the GTFS bus data, adding the zipped dataset from RTD, and creating the automated process to load all transit data into our database. This automation took all information like stops, routes, and timing, which was crucial for being able to query properly and load data onto our Mapbox interface. Additionally, I did many of the deliverables, like setting up render hosting and deploying our database from my local system to render, as well as implementing various test cases. In terms of user features, I implemented live user tracking and assisted with live bus tracking, as well as making sure our routes populated the map and displayed without redundancy.

RILEY - Github: rjrasizer

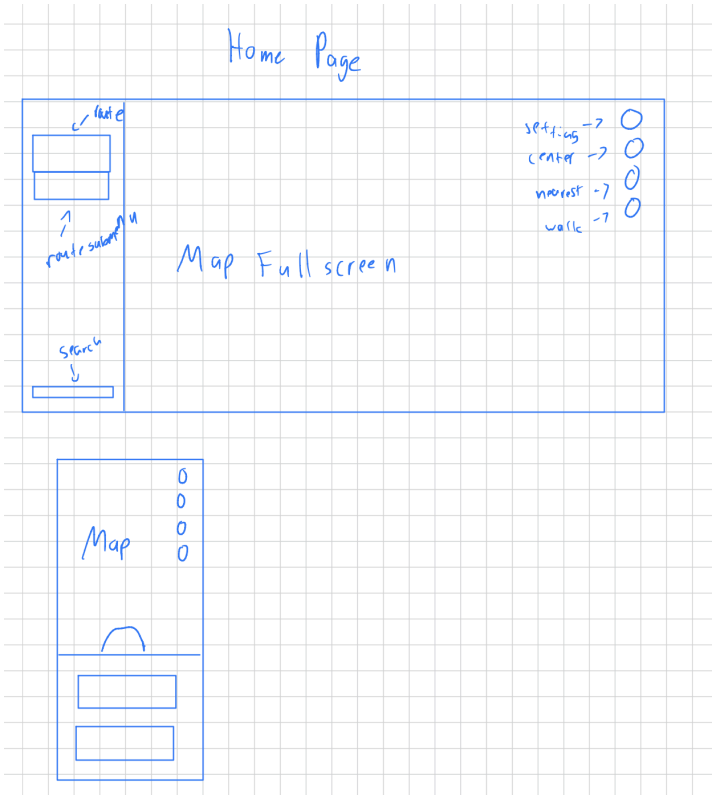
Developed and implemented file setup by creating handlebars layouts, styling sheets, and setting up basic initial database usage. Implemented API routing between login, logout, register, settings, and home pages, along with proper test cases and functions to ensure routes were properly working at all times. Implemented working sidebar functionality for both desktop and mobile users, along with styling for sidebar and other home page implementations. Also worked alongside Arman to implement working sidebar buttons that properly displayed associated routes whenever clicked.

Use Case Diagram

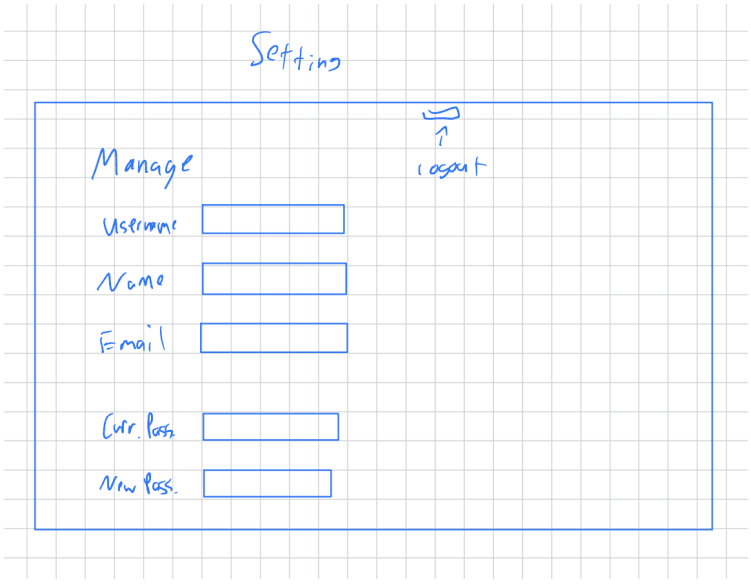


WIREFRAMES

Home page (mobile and desktop)



Settings page



TESTS

1. Valid User Registration and Account Creation

What was the user doing?

The user chose the register button from the login page. They then attempted to put their information into the email and password boxes. They tried to submit the registration form without completing all the required fields. The user finished by clicking the register button and navigating back to the login page.

User Reasoning

The user expected that all the regex requirements for the form live display and that the form would explain the constraints for password and email submission. Users tried to break the registration system by not following the provided requirements to see if there were bugs in the website.

Consistent Behaviour with the Use Case

The user did have consistent actions aligned with the use case and used the feedback provided by the application to be guided along the correct path to registration. The only stray was purposely trying to break the register page by failing to put correct data in or leaving forms empty to test the resilience of the program.

Deviation

The user was trying to stress test the register page by leaving forms empty and trying to input incorrect information. The user most likely did this to see if the requirements were soft and not actually enforced, testing the security of the web page.

Changes?

We did make changes to the feedback that we provided to the user to be absolutely clear throughout the entire process of registering. We noticed that the user didn't have a clear indication that their email must be in an exact format, so we added a catch in our routes and displayed a message in our JS.

2 . Invalid user registration/login

What is the user doing?
The user navigates to the registration and login pages. They try to sign in without one of the required forms. They try using dummy data in the boxes to see if it's actually checked against a database. They are redirected back to the page if incorrect.

Reasoning

We informed the user to stress test the register and login pages to find holes in them. The user's reasoning for this is to see if there's any way to

access the website without doing the proper registration/login, and their info is correctly matched in the database.

Behaviour

This behaviour is consistent with the use case, and we saw the user stress test in all the ways we predicted they would. Things like fake passwords, passwords that didn't match regex requirements, passwords that didn’t match emails, and incorrectly formatted emails.

Changes

We used the data gathered from this test to edit our catches in our routes for login and registration. We noticed that if you tried to sign up with an email that already has an account, the application would crash and fail to inform the user that the email was already in use. We fixed that and added a responsive message to show that they need a unique email for their account.

3. Route Choosing via The Sidebar

What?

The user went to the sidebar and navigated around their local routes. They used the search bar to filter for a specific bus and clicked on it. They then clicked the closest route button to get walking instructions to that bus stop.

Reasoning
The user's reasoning for this was to test the actual functionality and ui experience of finding a route they want, and getting the directions to get there. We wanted to test how intuitive it was for someone who had never seen the application to use the main functionality.

Behaviour:

The behaviour of the user was exactly what our use case predicted, and they were able to access all the features that we implemented in a normal manner. There was no major confusion, and they were able to pick a route and find the walking directions to get there.

Changes:
From this test, we noticed that the touch window for expanding the routes to see bus times was small, so we expanded it to the whole card to make interaction easy on a browser.