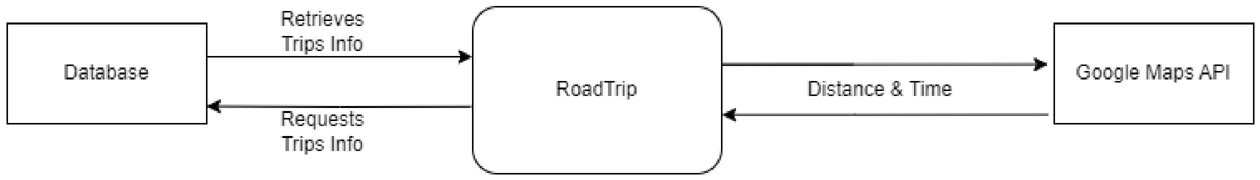
Roadtrip

Group 3

By Deeksha Marpadaga, Gabriel Dunlop, Matthew de Ridder,
Nate Mirman, Rob Salmon

1. **Project Overview**
   GitHub Link: https://github.com/natemirman/ITSC4155_MDSp24_Group3

   RoadTrip helps alleviate the chaos of travel by simplifying the planning stage. RoadTrip looks to solve the problem of travel by having an all-in-one software to organize your vacations. This includes routing, food stops, trip creator and saver and store destinations, and distance traveled. Our stakeholders include hikers, bikers, and frequent travelers. This software is designed to make planning travel an enjoyable experience for them. The goal is to have a way for stakeholders to have a place to plan and hold all their trips.



2. **Architectural Overview**

   RoadTrip will use a homepage as a main hub for navigating the website. It will have routes for planning or changing trips. Users will have access to view and keep track of their trips on the webpage. Having a page both for planning and a separate one for view to allow ease of use and understandability. Styles of map viewing and using were compared but simplistic straightforward map view and access were chosen to help users. For the website we will be using a Google Maps API hosted on a page with Javascript and CSS, this will be connected to a SQL Database to store the key points of the trip and display them when called. With the ability to easily use maps in an understandable way presented with a clear CSS and features of Javascript to streamline it.

   Our application uses the Model-View-Controller Architecture. This architecture was used because there is a clear distinction between data management, presentation logic, and user interaction in our travel app.

   Model: In this code, the SQLite database and the db.py file serve as the model. The db.py file contains functions to interact with the database, including creating tables, inserting data, querying data, and deleting data.

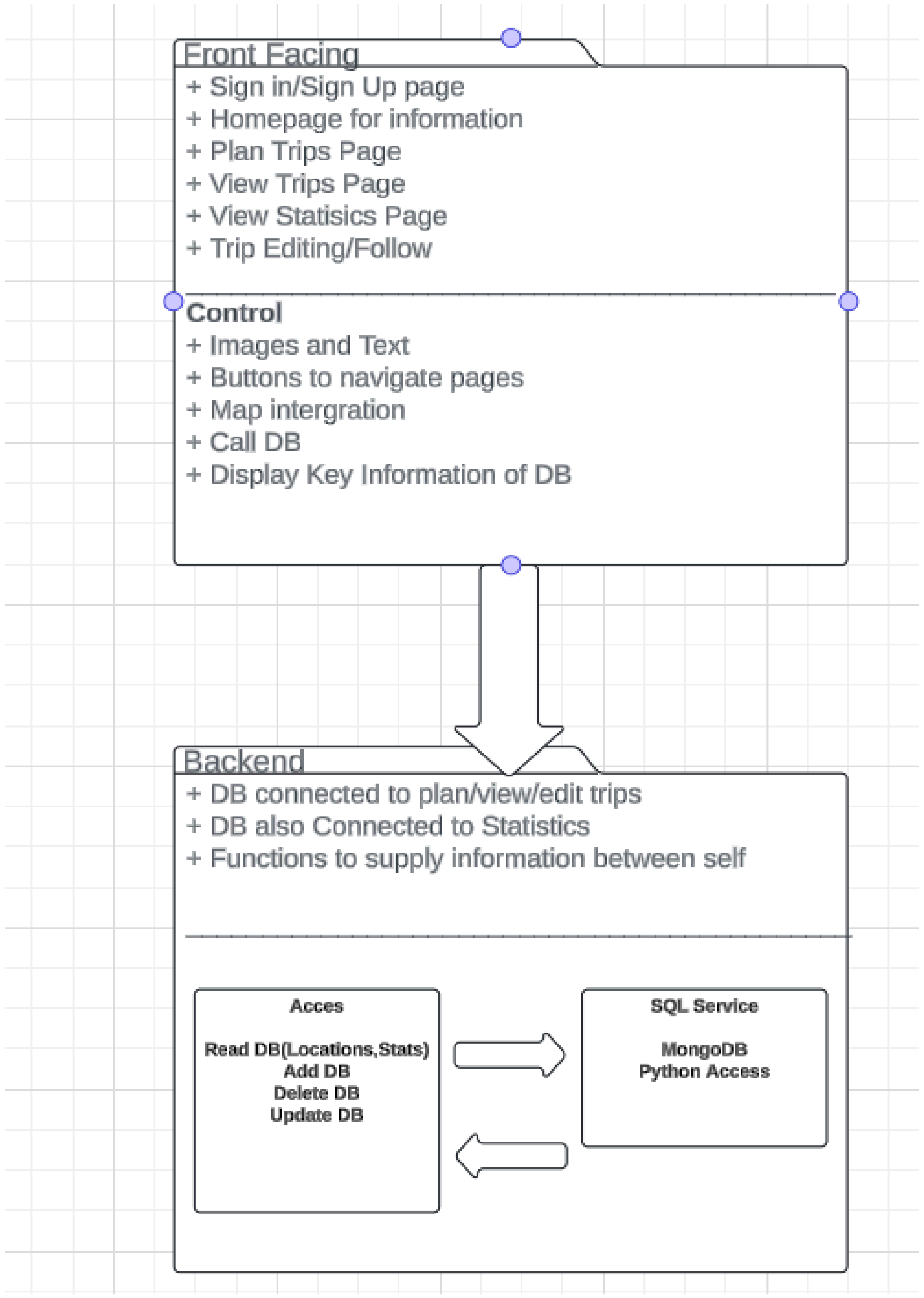
   View: HTML templates serve as the views. These HTML templates are responsible for presenting data to the users and capturing user input through forms and buttons. Also CSS is used for styling to enhance the presentation of the views.

Controller: The Flask application acts as the controller in this architecture. It defines routes for different URLs and HTTP methods, such as GET and POST. The routes handle incoming requests, interact with the model (database), retrieve or manipulate data, and render the appropriate view templates as responses. The controller also contains logic for redirecting users to different pages, handling form submissions, and processing data before storing it in the database.

2.1 Subsystem Architecture

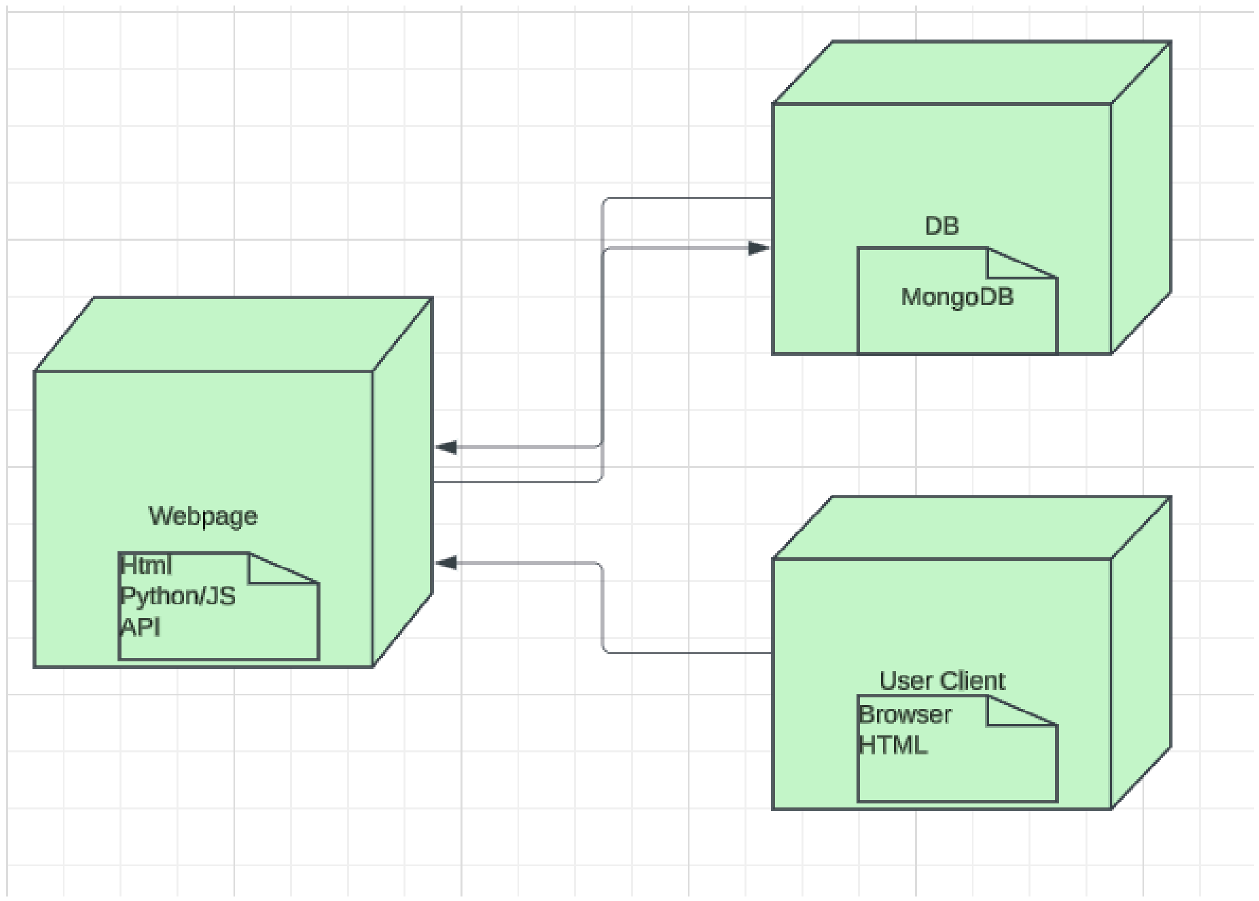
There are two main layers with connections to different parts of the page and app. The front-facing usable layer is made up of what the user sees and inputs into the page. It is what users will see and use while designing and looking at trips and using the application. On the back side, there is a layer of running and data storage that connects the front-facing pages and provides the data and connectivity for users to see a map and add and remove trips from it, and call the information up layer.

UML Diagram:


Front Facing
+ Sign in/Sign Up page
+ Homepage for information
+ Plan Trips Page
+ View Trips Page
+ View Statisics Page
+ Trip Editing/Follow
Control
+ Images and Text
+ Buttons to navigate pages
+ Map intergration
+ Call DB
+ Display Key Information of DB
Backend
+ DB connected to plan/view/edit trips
+ DB also Connected to Statistics
+ Functions to supply information between self
Acces
Read DB(Locations,Stats)
Add DB
Delete DB
Update DB
SQL Service
MongoDB
Python Access

2.2 Deployment Architecture

This software will run on a single processor.



2.3 Data Model

All the data for the trips will be stored in a MongoDB server. Information includes the trip name, all the stops, the distance and the estimated time. The user will build trips and name them and the parts of them will be stored in the database as text to build the same trip when asked to review it. Other important parts that can be saved like time and distance will also be available to see.

2.4 Global Control Flow

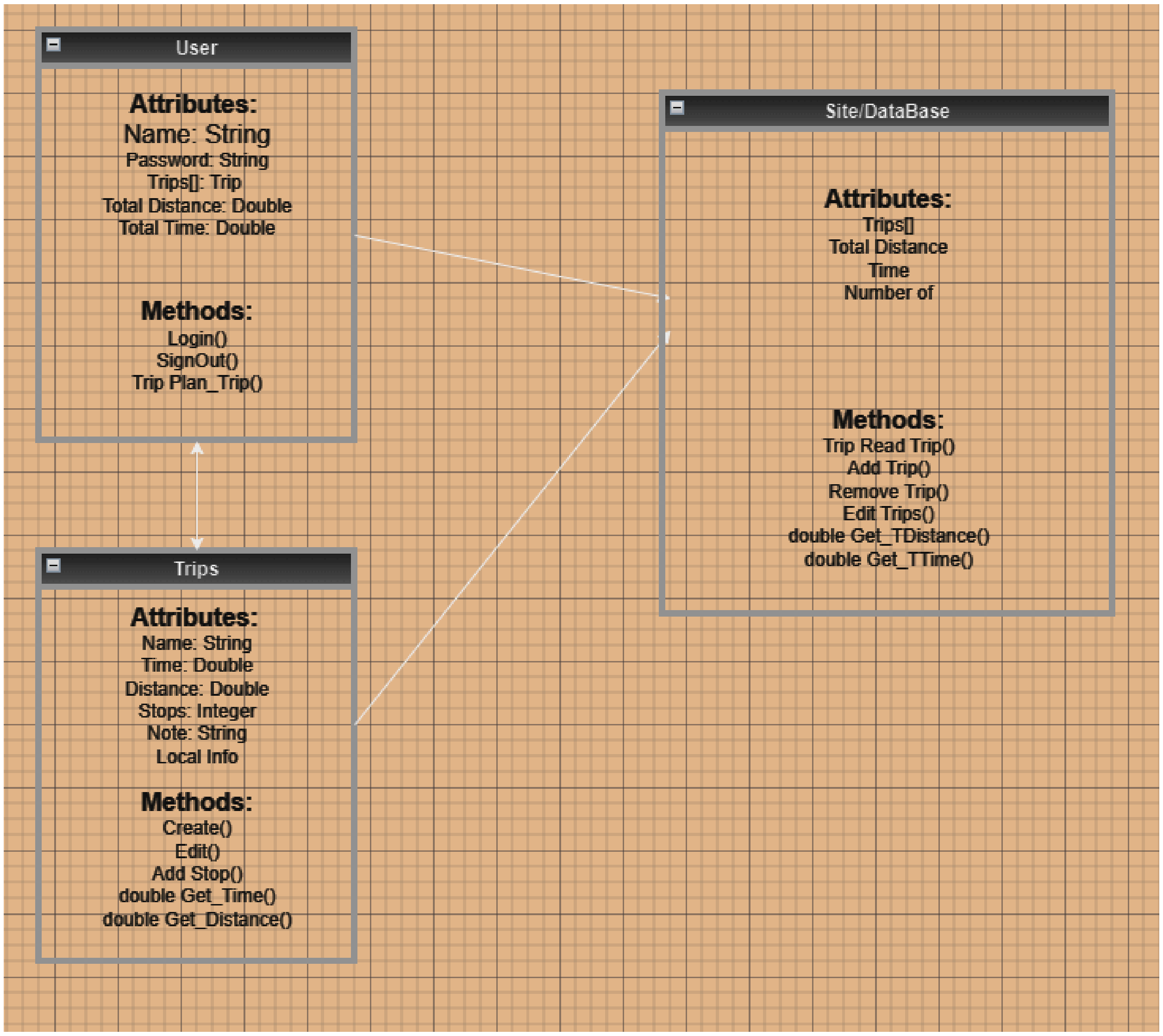
The system is an Event-driven system that executes events when doing one of: accessing the page, building or editing a trip, and then viewing it when wanting to follow it. The trips will be planned on real-time events as they should be on certain days and times to allow for planning around when the user would be on the trip and updating or following as events are triggered. The constraints on this of course are actual locations and times that haven't passed yet so the trip can exist in the APIs used to build it. The system will only offer the option to save and view those trips to signed-in users and will only let the plan be made for those without an account.

3 Detailed System Design

The system of RoadTrip is built using standard HTML using the Bootstrap framework for CSS and connectivity. Along with that to design and stylize the page and give the ability to interact with it others are used to provide outside details. The main feature is access to the Google Maps API which will be used to view maps of places and plan distances and times of trips coming from Google and their system. The control and use of these are backed by Javascript and Flask. This is then connected to a SQL database to store and call this information which is presented in Bootstrap's table format on the view trips page.

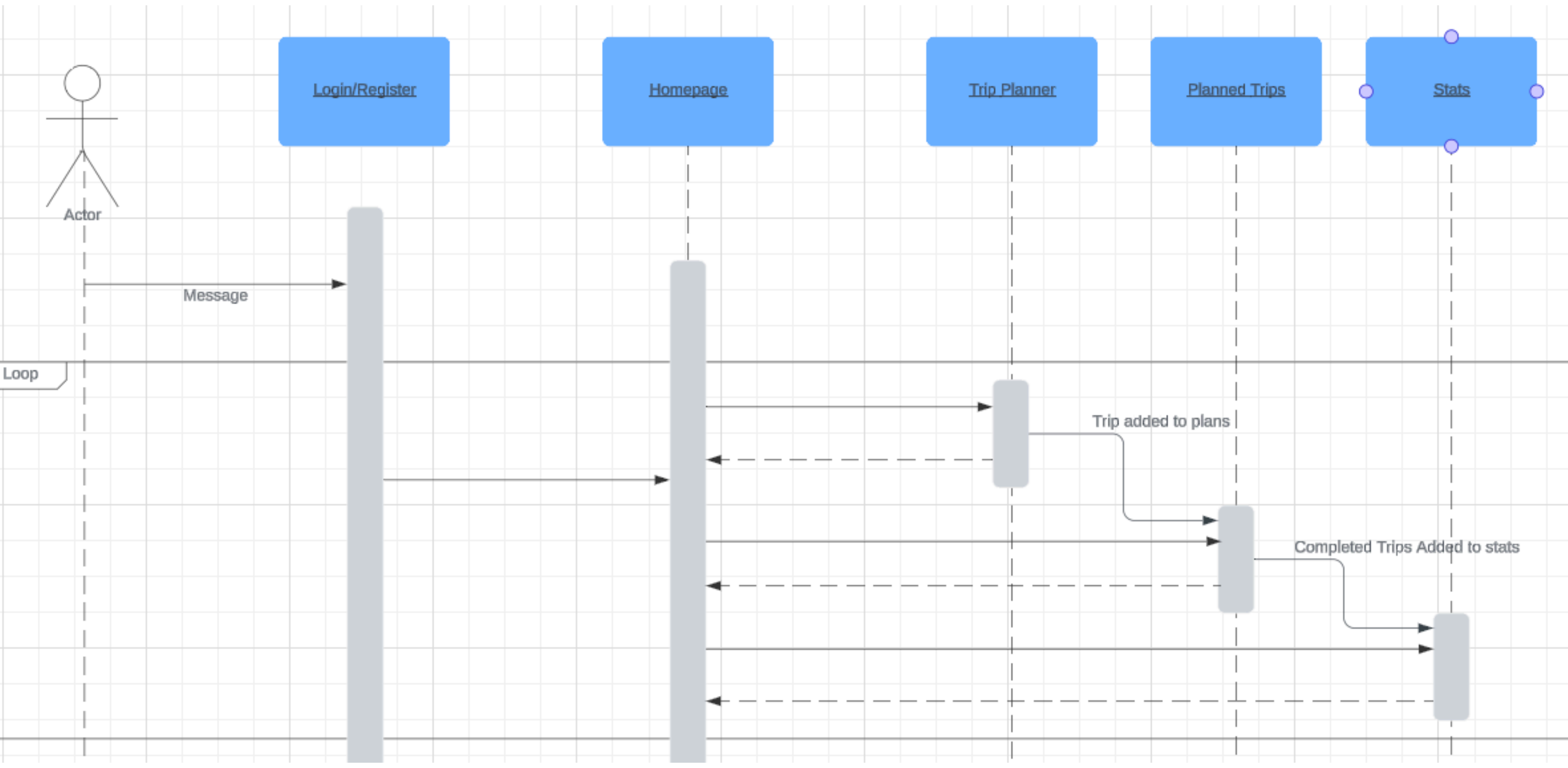
3.1 Static view

Users will be stored with information about Username, password, trips, total time, total distance, and other trip statistics that are feasible to track and bring up for detailed information on their use of the app and trips. This will be connected to the web application that by extension reaches for a trip class in the database that will hold the name, time, distance, stops, and other information on currently saved trips and be used as the base for planning new trips. The web application will hold the forward-facing information of both of these sides to make an easy viewing experience of the trips and statistics for trip enthusiasts.



3.2 Dynamic view

When a user joins they are prompted to log in/sign up if applicable or continue as a guest to access the main webpage. On the main page, there is an option to plan, view, or see statistics of trips. They can then choose based on their needs to either plan or view trips and edit or follow as needed. When the time there will be focus brought to a current trip and important statistics.


Login/Register
Homepage
Trip Planner
Planned Trips
Stats
Actor
Message
Loop
Trip added to plans
Completed Trips Added to stats