Bus Tracker

Group X
Matthewit Dechatech
Jonathan Kleehammer
Ashley Torres
Tristan White

SOFTWARE DESIGN DOCUMENT
Version 1.0

INDEX

1. Introduction

   This document is going to cover what out software product is, as well as its functionalities, so that readers are able to visualize the architecture of the product.

   Our product is a bus tracker that will inform users where the current location of a bus is for a specified route. We will find the nearest bus in relation to the user and the expected times the bus will reach its stops.

   The reason were are writing a software design document is because designers need to check if they are filling the requirements given to them by the client. It can save valuable time for engineers as they are simply refer back to their design document and make sure they are staying on track.

   The document is organized by having an introduction first, then getting more specific as you read onward. After this introduction, you'll be able to see an overview of the software's architecture in section two graphically with a brief explanation which will give you a broad sense of how the software works. Afterwards, section three presents a use case diagram which helps map out system requirements which is then followed by subsections of tables showing specific flows of important actions. Section four is our class diagram that shows how the classes will be interacting with each other. Section five features our state transitions that users can transition between. Section six contains example screens for the users. Then we wrap things up in section sevens conclusion which explains what we have learned and our difficulties while working on the project.

2. Overview of architecture

   This application uses a server client architecture This has a web app made with React access a backend made with Django. This in turn accesses a PostgreSql database which contains the data needed for the app.

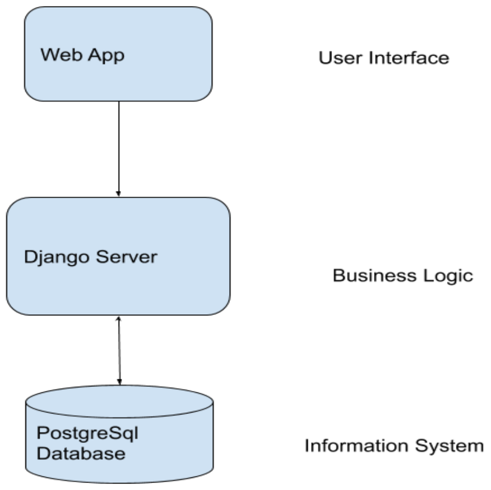
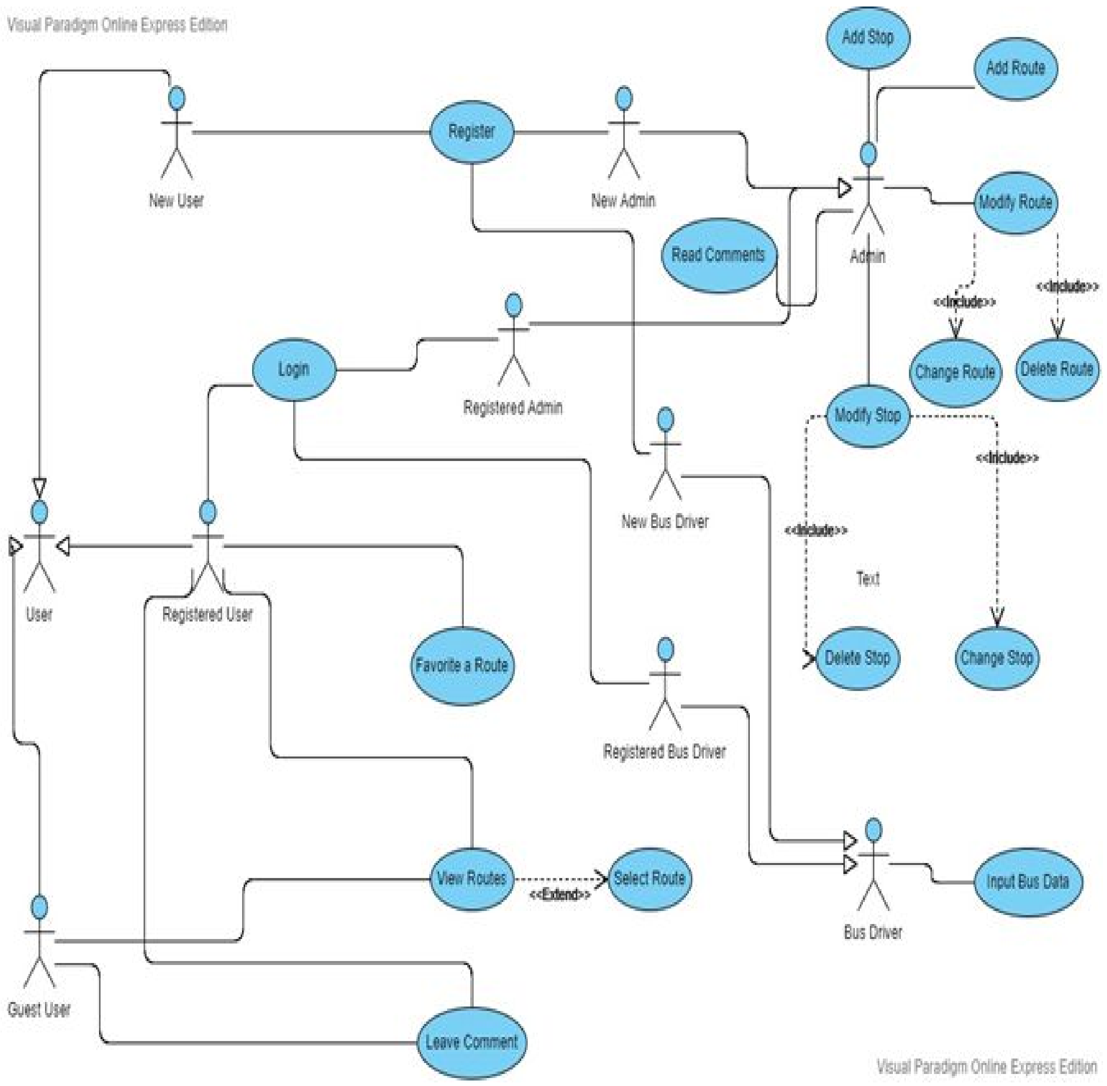


Figure 1. Server-Client architecture used in app

The server client architecture is chosen because it provides a database for the user to interact with. This allows the necessary information for both the riders and admins to be stored in an easy to access location. The Web application is the user interface because it needs to be used on desktops and mobile devices. A web app is best for this as it can be brought up in a browser regardless of the device

3. Use cases
   A. Use case diagram



B. Use case specifications

i. UC1 – View Route

Use Case Number:	UC-0001
Use Case Name:	View Route
Overview:	Rider views a route and the buses along the route
Type:	primary
Actors:	Rider
Pre-condition:	Rider is on the main screen

Main flow:	1. Rider is on the main screen 2. Rider clicks the route list button 3. The routes list slides out 4. The rider selects a route from the list which expands the stop list and shows
Post-condition:	The list of routes is now being displayed The selected route is highlighted on the map with bus positions shown in real time
Cross-reference:	

ii. UC2 - View Profile

Use Case Number:	UC-0002
Use Case Name:	View Profile
Overview:	Rider views their profile which shows favorite routes and stats
Type:	Secondary
Actors:	Rider
Pre-condition:	Rider is on the main screen
Main flow:	1. Rider is on the main screen 2. Rider clicks the user button in the top right 3. The rider is brought to their user page 4. The rider selects a route from the list which expands the stop list and shows
Alternate flow:	1. If the player isn't logged in they are brought to the login page
Post-condition:	The list of routes is now being displayed The selected route is highlighted on the map with bus positions shown in real time
Cross-reference:	

iii. UC3- Register New User

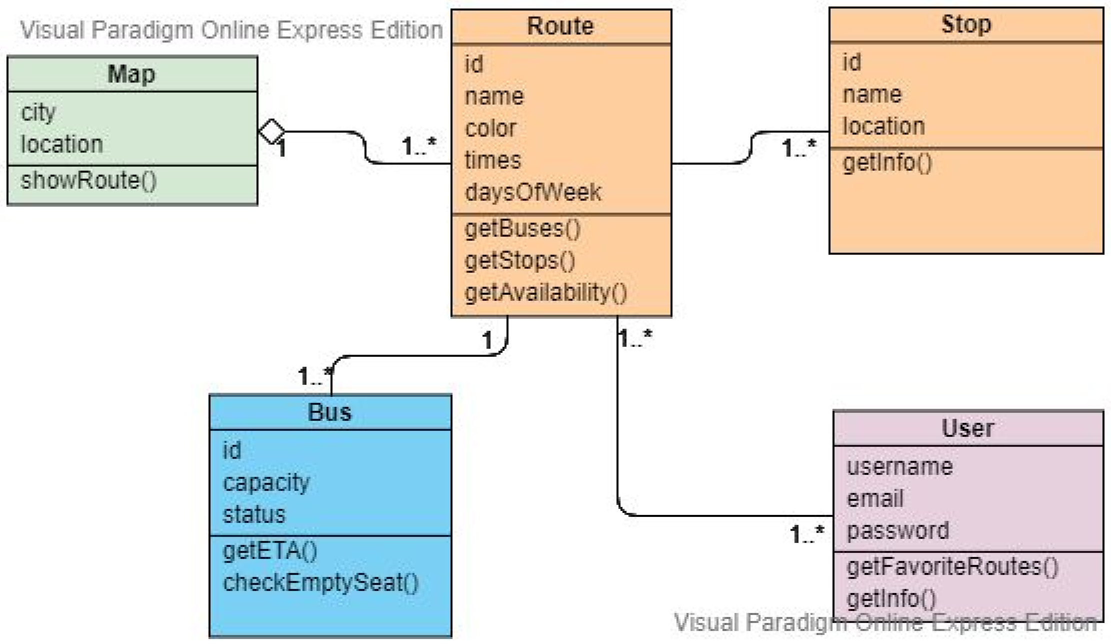
Use Case Number:	UC-0003
Use Case Name:	Register New User
Overview:	Rider wants to register a new account
Type:	Secondary
Actors:	Rider
Pre-condition:	Rider is on the main screen
Main flow:	1. Rider is on the main screen 2. Rider clicks the user button in the top right 3. The rider is brought to the login page which has a button for registering
Alternate flow:	1. If the player is already logged in they will be brought to their user page, from there they can logout to return to the login page
Post-condition:	The list of routes is now being displayed The selected route is highlighted on the map with bus positions shown in real time
Cross-reference:	

iv. UC4 - Favorite a Route

Use Case Number:	UC-0004
Use Case Name:	Favorite Route
Overview:	User selects a favorite route
Type:	Secondary
Actors:	User

Pre-Condition:	User has opened the sliding menu to view the lists of routes.
Main Flow:	1. User will scroll through the list and find the route they want to favorite. 2. Users will click the "Favorite" button. 3. If the user is logged in the route will save the route, else the system will prompt the user to log in.
Alternate Flow:	1. If the user doesn't want to log in to save the route, then they can click the 'X' to exit the menu.
Post-Condition:	True
Cross-reference	Check if the user is logged in

4. Classes and their interaction
    a. Class diagram



    b. Details of classes

i. C1 – Map Class

Attributes:

Attribute name	Type	Description
City	String	A variable used to store the city of the map.
Location	String	A variable used to store the location of the map.

Operations/Methods:

Method Name	Arguments passed	Expected return value	Description
showRoute()	Route	Null	A function used to have the map show the details of a route.

ii. C2 - Route Class

Attributes:

Attribute name	Type	Description
Id	String	A variable used to store the route ID.
Name	String	A variable used to store the route name.
Color	String	A variable used to store the route color.
Times	String list	A variable used to store route times.
DaysOfWeek	String list	A variable used to store available route days.

Operations/Methods:

Method Name	Arguments passed	Expected return value	Description
getBuses()	None	List of Bus	A function to retrieve the buses on the route.
getStops()	None	List of Stop	A function to retrieve the stops on the route.
getAvailability( )	None	True/False	A function to retrieve whether or not the route is available.

iii. C3 - Stop Class

Attributes:

Attribute name	Type	Description
ID	String	A variable used to store the stop ID.
Name	String	A variable used to store the stop name.
Location	String	A variable used to store the stop location.

Operations/Methods:

Method Name	Arguments passed	Expected return value	Description
getInfo()	None	String	A function used to retrieve stop info.

iv. C4 - Bus Class

Attributes:

Attribute name	Type	Description
ID	String	A variable used to store the bus ID.

Capaciry	Integer	A variable used to store bus capacity.
Status	String	A variable used to store bus status.

Operations/Methods:

Method Name	Arguments passed	Expected return value	Description
getETA()	Stop	Time	A function used to get a bus' ETA to a stop.
checkEmptySeat()	None	True/False	A function used to check if the bus is full or not

v. C5 - User Class

Attributes:

Attribute name	Type	Description
Username	String	A variable used to store the username.
Email	String	A variable used to store the user's email.
Password	String	A variable used to store the user's password.

Operations/Methods:

Method Name	Arguments passed	Expected return value	Description
getFavoriteRoutes()	None	List of Route	A function used to retrieve the user's favorite routes.
getInfo()	None	String	A function used to retrieve user info.

vi.

c. Sequence diagrams

Diagram #1: This shows the communication between the user, UI, map, and server for how a route is viewed.

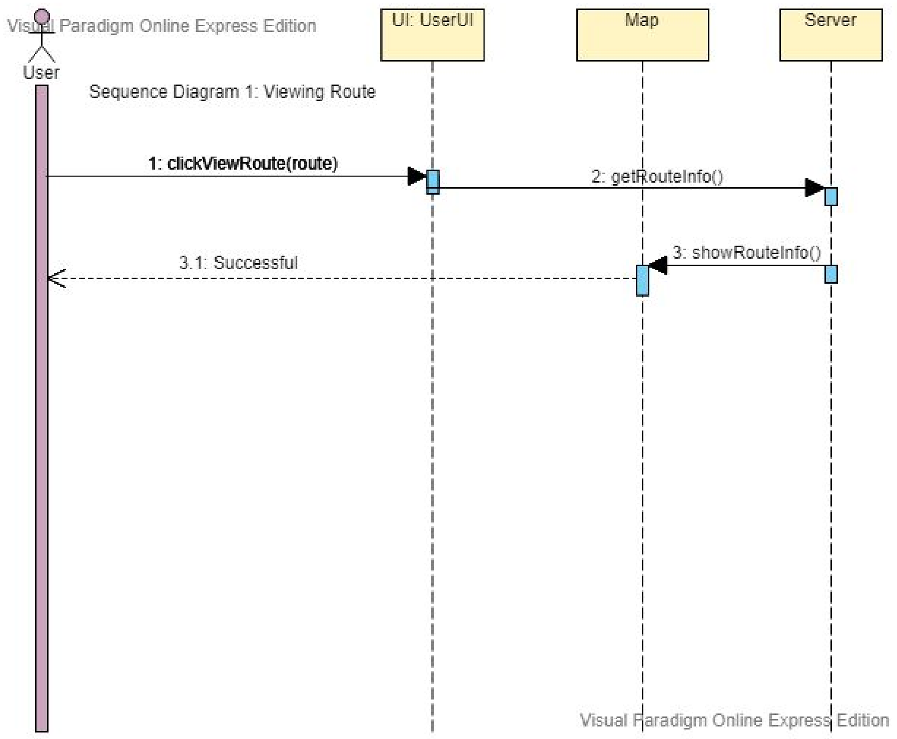

Visual Paradigm Online Express Edition
User
UI: UserUI
Map
Server
Sequence Diagram 1: Viewing Route
1: clickViewRoute(route)
2: getRouteInfo()
3: showRouteInfo()
3.1: Successful
Visual Paradigm Online Express Edition

Diagram #2: This shows the communication between the user, UI, and server for how profile info is viewed.

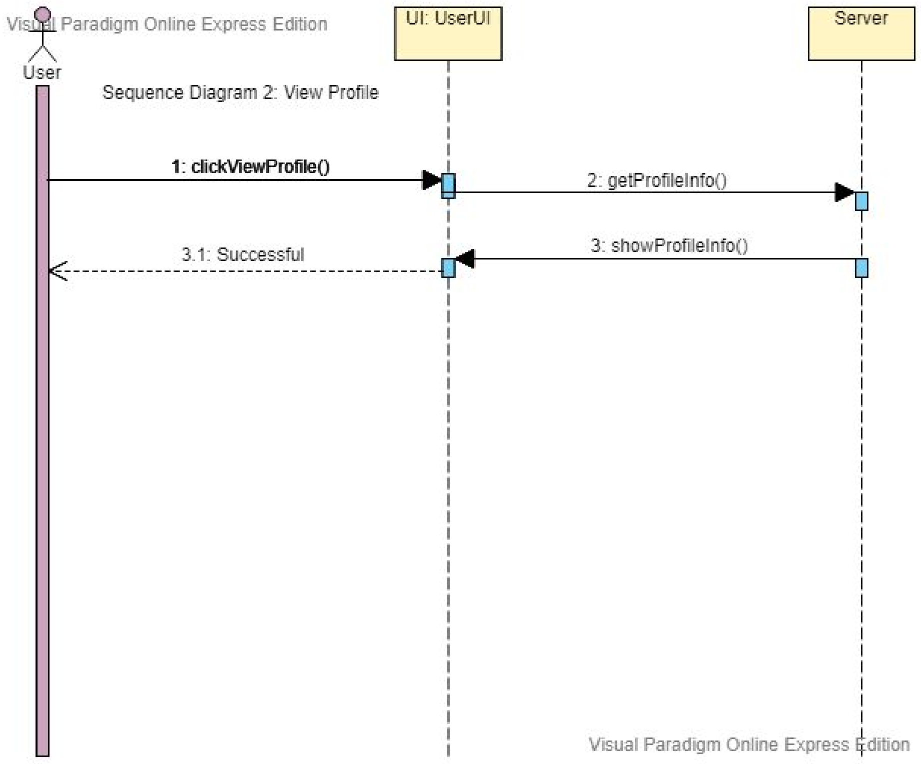

Visual Paradigm Online Express Edition
User
UI: UserUI
Server
Sequence Diagram 2: View Profile
1: clickViewProfile()
2: getProfileInfo()
3: showProfileInfo()
3.1: Successful
Visual Paradigm Online Express Edition

Diagram #3: This shows the communication between the user, UI, and server for how a new user is registered.

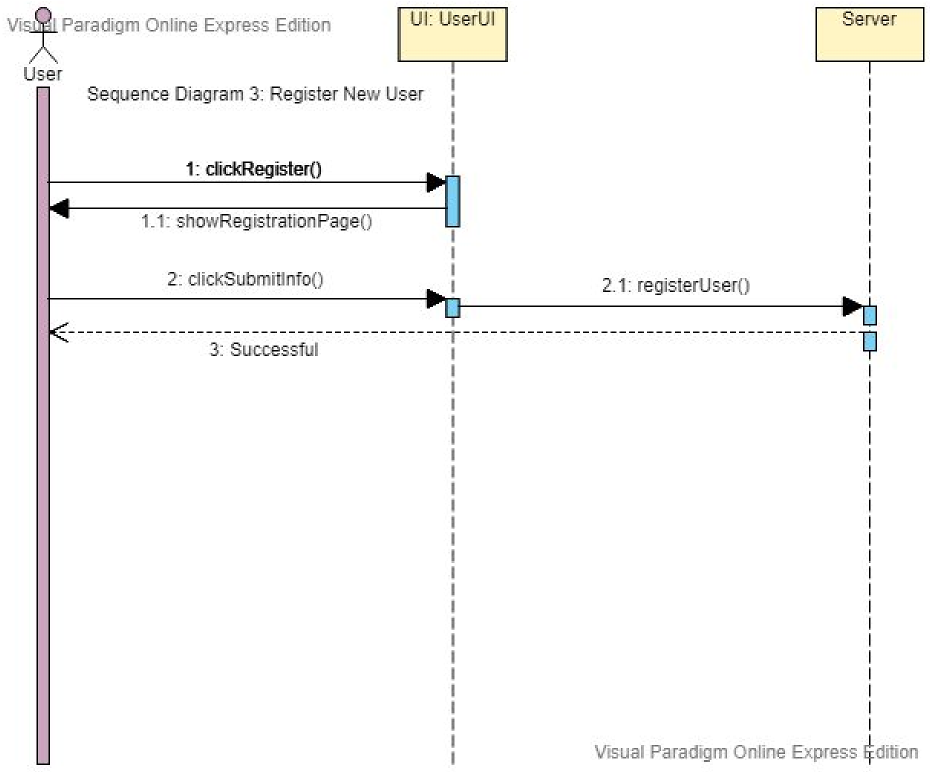

Visual Paradigm Online Express Edition
User
UI: UserUI
Server
Sequence Diagram 3: Register New User
1: clickRegister()
1.1: showRegistrationPage()
2: clickSubmitInfo()
2.1: registerUser()
3: Successful
Visual Paradigm Online Express Edition

Diagram #4: This shows the communication between the user, UI, and server for how a route is favorited.

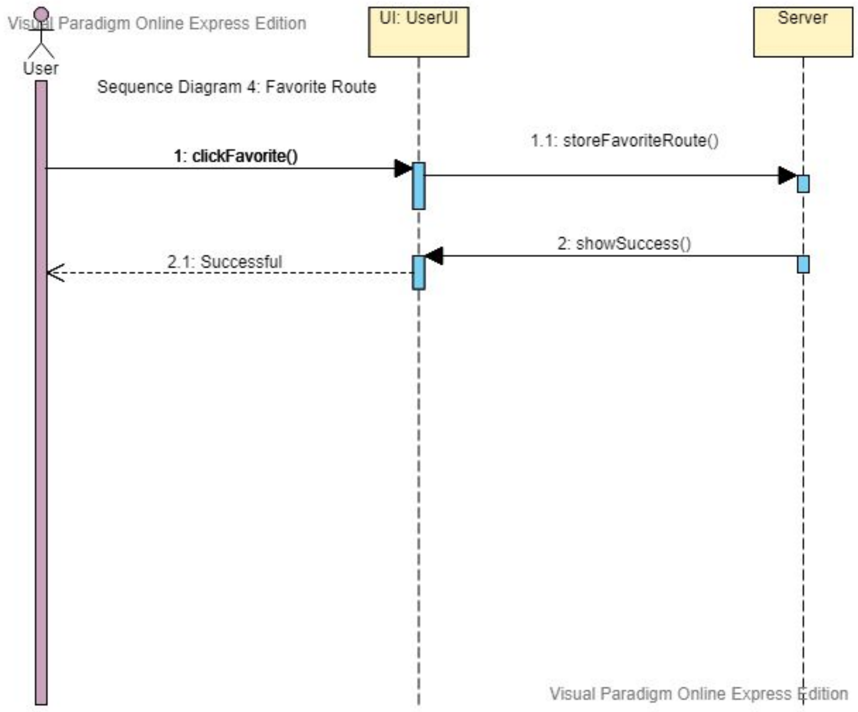
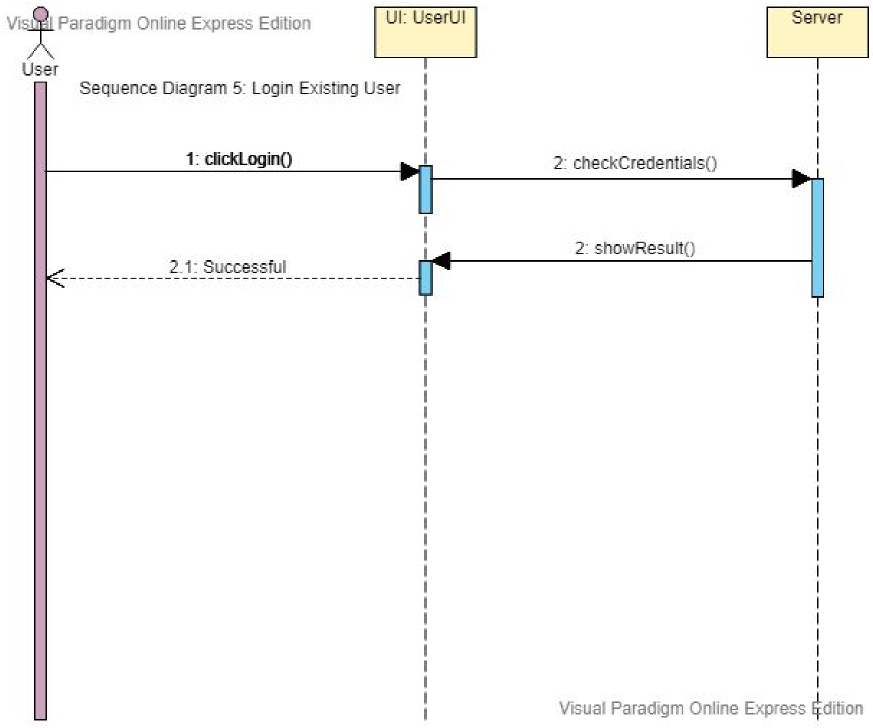
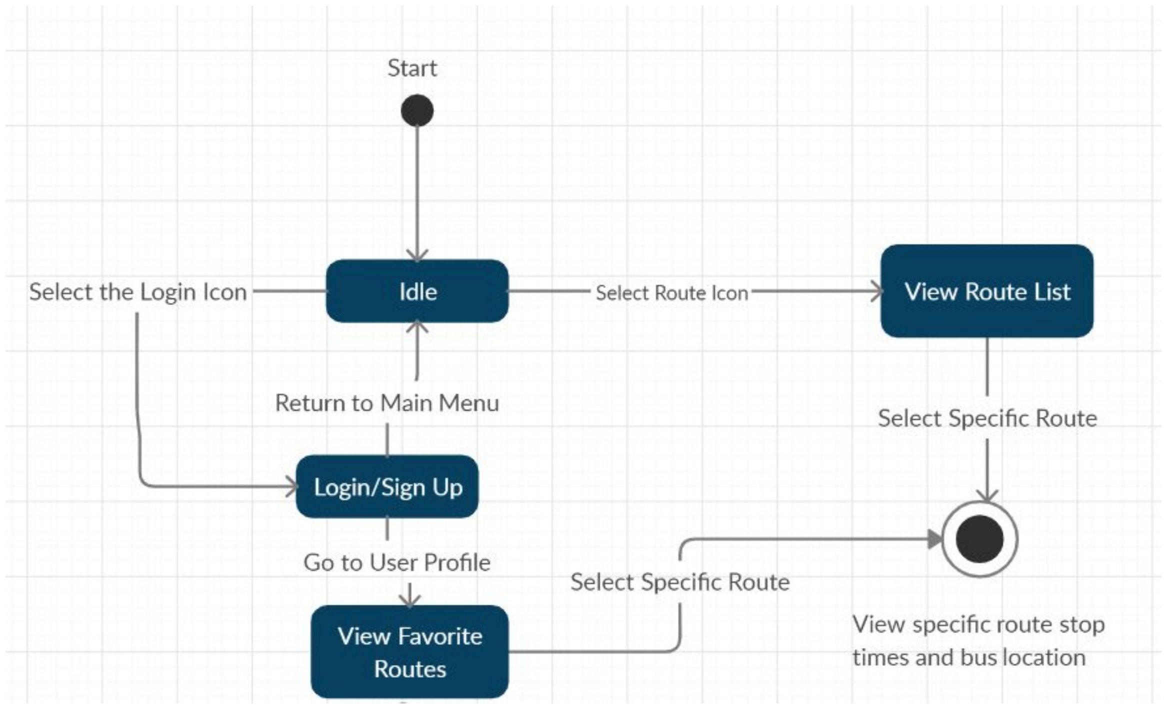

Visual Paradigm Online Express Edition
User
UI: UserUI
Server
Sequence Diagram 4: Favorite Route
1: clickFavorite()
1.1: storeFavoriteRoute()
2: showSuccess()
2.1: Successful
Visual Paradigm Online Express Edition

Diagram #5: This shows the communication between the user, UI, and server for how an existing user is logged in.


Visual Paradigm Online Express Edition
User
UI: UserUI
Server
Sequence Diagram 5: Login Existing User
1: clickLogin()
2: checkCredentials()
2: showResult()
2.1: Successful
Visual Paradigm Online Express Edition

5. State transition diagrams

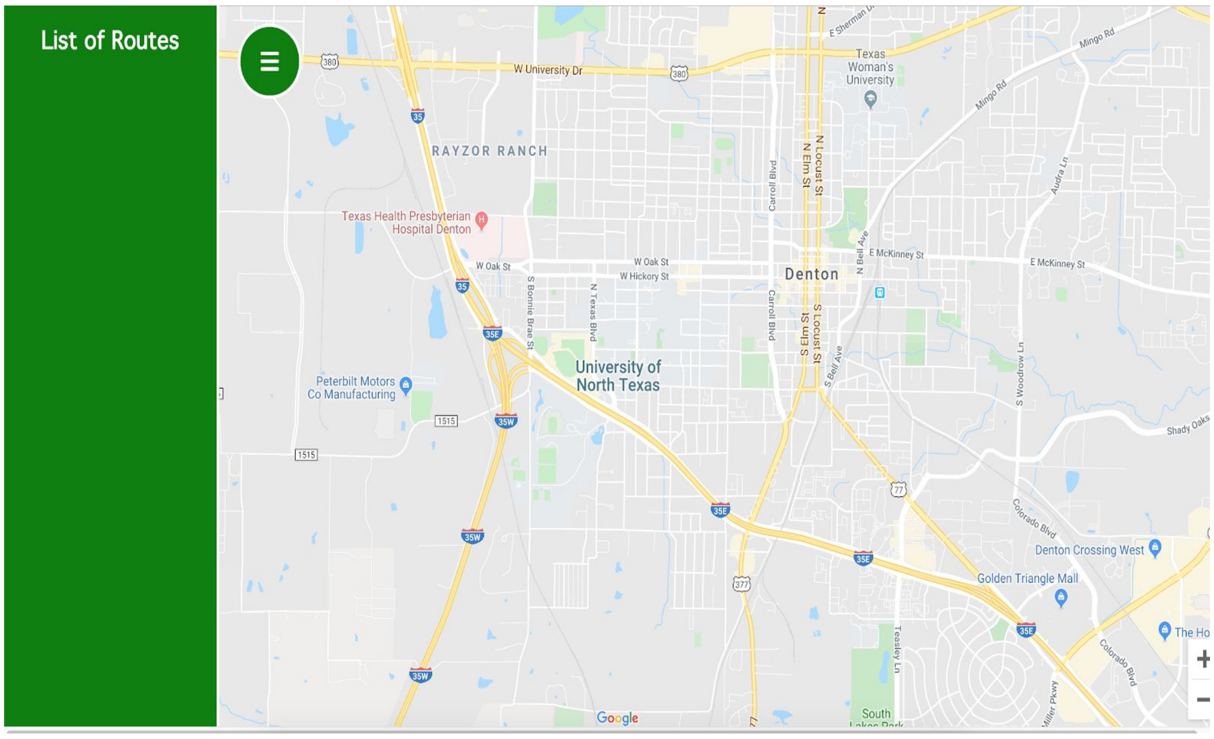


The idle state is the first state we enter, in this state we see a map centered on Denton with icons leading to the routes list and the login/sign up page. Depending on which icon the user picks it will take them to that respective page.

If the login/sign up page is chosen, then the user will have to enter their email address and password. If they choose to sign up, then they will enter their email, password, and password verification. The web application will either then return to the main menu and wait for the user to choose another/ action, or go to the user's profile to view the favorite routes. From there, the user can select a favorite route and view the route stop times and bus location.

If the user were to select the route icon, they are taking to a list of all bus routes. Once the user finds a route they want to view, they can select that route and they will see the route stop times and the bus location.

6. User interface design
   a. Screen 1 – Main Screen (Route List Expanded)
      Example:



The main routes screen is currently a work in progress. The Map on the screen will have the complete route displayed when the user clicks on the route in the drawer to the left. The drawer will contain the different routes in alphabetical order with the user favorites on top There will also be a profile button located on the page which will take the user to the profile page

b. Screen 2 - Profile page

This will contain the users settings along side the option to change them, and delete their account. If they don't have an account, or aren't logged in, a page with a sign up or login button will be shown.

c. Screen 3 - Admin page

This will only be accessible through a special URL with a special login. The admin page will contain access to the different databases of routes, buses, maps, users, and the stops. This will allow for the administrators to be able to create, update, search, and delete various pieces of data.

7. Conclusion

Through this design project, we have learned about how interconnected all our different classes and functions are. It's also helped us to visualize how everything will be placed in our web app to make a cleaner and more user-friendly page. With this design document, we will be able to refer back to the diagrams and explanations and make sure that we are on the right track.

In the beginning, we found that we didn’t know a whole lot about designing software. There was a lot of learning and coding at the same time occurring. We’ve learned to take our time while we’re coding to make sure that we are doing everything right. We also found that the project tools weren’t compatible with at least half of our groups team. Due to this, we have begun pair-programming in order to make sure that we do not get behind on our project.