CS/SE/CE 3354 Software Engineering

Team 8 Project Deliverable 2

Comet Commuter Carpool

Group Members

- Aman Balam - Comparison of your work with similar designs, Use Case Diagram slides
- Vincent Jones - Project Scheduling, Title, Objective, and Timeline slides
- Neal R Kapadia - Conclusion, Class Diagram slide
- Shivani Kumar - Cost, Effort and Pricing Estimation, Slides Design, Cost Estimation and Architectural design slides
- Alan Edward Roybal - Test plan, Prototype creation, Prototype slide
- Aarush Shintre - Cost, Effort and Pricing Estimation, Cost estimation and Requirements slide
- Andy Weng - Test plan, Sequence Diagram slides

Repository URL

https://github.com/AarushShintre/3354-Team8

Feedback From Proposal

The feedback from our project proposal is that we did well. The stakeholders were satisfied and wanted us to continue on with the project.

Software Process Model

For the development of our Comet Commuter Carpool web application, our team has chosen to use the Waterfall software process model. The Waterfall model requires each phase to be completed before the next begins in a linear fashion. The phases of the Waterfall model include requirement gathering, system design, implementation, testing, deployment, and maintenance. We chose the Waterfall model as it aligns well with the timeframe and structure of deliverables of the semester long project. Thus, creating a clear roadmap for our team to look towards. The main reason we chose the Waterfall model is that it emphasizes upfront planning and the creation of documentation. Since the Waterfall model has specific stages for planning that must be completed before moving forward, we can ensure our idea is fully fleshed out and everyone is on the same page before diving deeper. Our project requires a well-defined set of features, including user profile management, ride scheduling, payment handling, and safety tracking. By clearly defining these requirements early in the process, we reduce the risk of miscommunication and ensure that all team members share a unified understanding of the project objectives. Another advantage of the Waterfall model is the ease of accountability and role assignment. This is something we've already done for deliverable 1, so it fits well into the nature of the Waterfall model. Each member of our team has clearly delegated responsibilities, ranging from backend and frontend development to database management and testing. With a sequential process, team members can focus on completing their designated phase while maintaining a clear path to the next step. For example, once requirements and design diagrams are finalized, the frontend and backend teams can begin implementation of a design the entire team agrees on. This reduces the risk of conflicts or rework during later stages. The Waterfall model also supports a robust testing phase that follows implementation. Since our application deals with user authentication, personal identifiable information (PII), and potential payment integration, it is essential to thoroughly test all features before deployment. Testing at a defined stage allows us to systematically identify and correct errors based on a final product, rather than a prototype. While iterative models like Agile offer flexibility for changing requirements, the Waterfall model is linear and better fitted for a short term project, making it the ideal choice for a semester project with clearly defined goals. By following the Waterfall approach, our team can progress from requirement analysis to deployment, ensuring that Comet Commuter Carpool is both functional and reliable.

Requirements

Functional Requirements

1. The system shall allow users to log in securely using email/password or phone OTP.
2. Users shall be able to create and edit their commute profile, including their regular schedule and route points.
3. The application shall match users with other commuters based on location and overlapping schedules.
4. Users shall communicate via a secure in-app chat after accepting contact requests.
5. The system shall send notifications for ride invites and driver arrivals using user-selected methods (in-app, SMS, email).
6. The system shall calculate and display ride costs based on distance and rider count, supporting digital payments.
7. Users may request extra stops on a ride, with costs automatically adjusted.

Non-Functional Requirements

Product Requirements

- *Performance*: The app should respond to core actions (login, matching, messaging) within 3 seconds.
- *Dependability*: At least 99% system uptime should be expected.
- *Security*: All personal and session data must be encrypted; access must be limited to authorized users.
- *Usability*: Users shall navigate key features with no more than five clicks.

Organizational Requirements

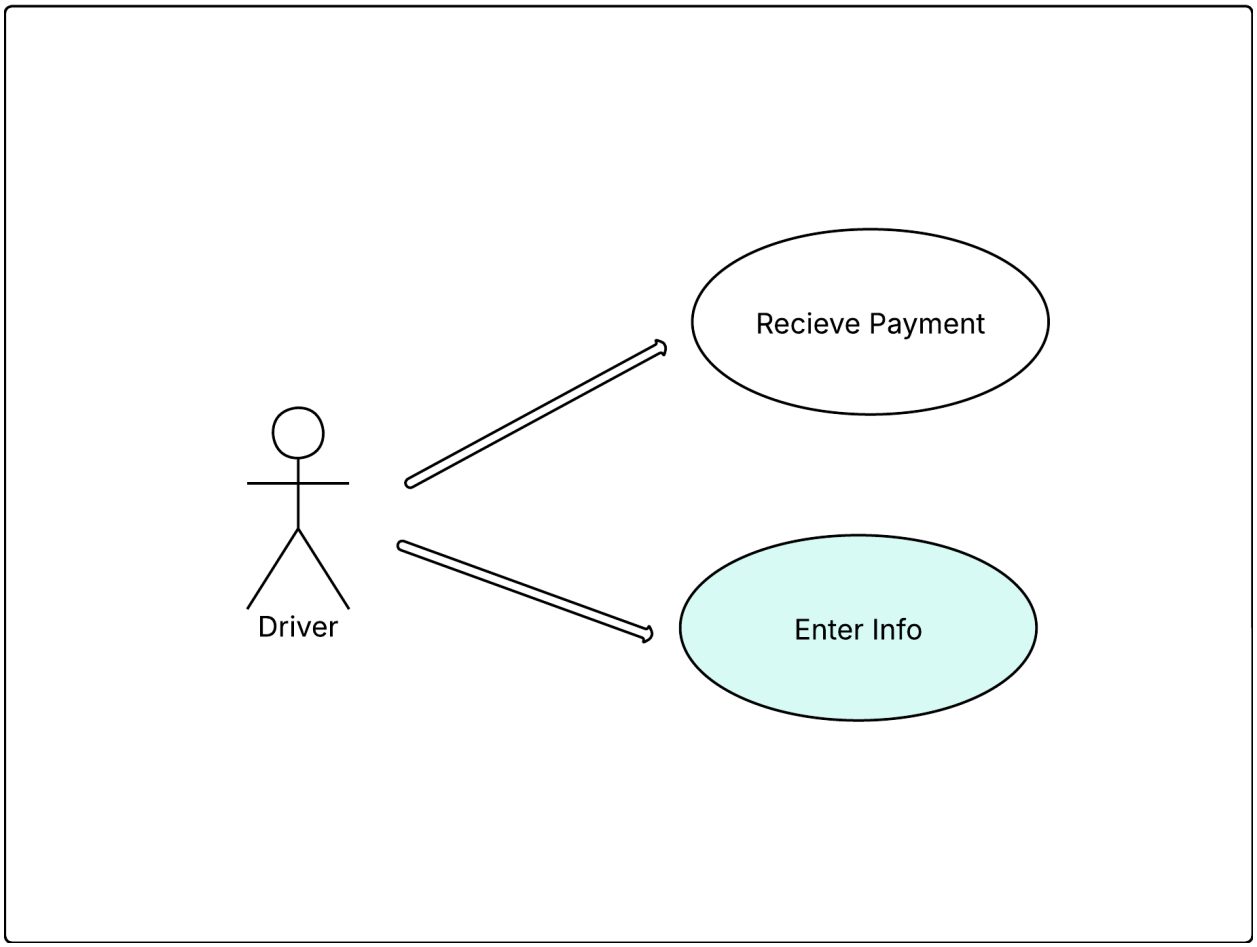
- *Operational*: Announce maintenance periods at least 24 hours ahead.

External Requirements

- *Regulatory*: Comply with GDPR and CCPA data protection rules.
- *Ethical*: Do not use user data for secondary purposes without consent.
- *Development*: Follow approved software lifecycle for releases.
- *Legislative*: Meet local transport and privacy legal requirements.
- *Accounting*: Generate digital receipts for payments.
- *Safety/Security*: Shall provide users with reporting and blocking tools for abusive behavior, and must verify student status of all users.

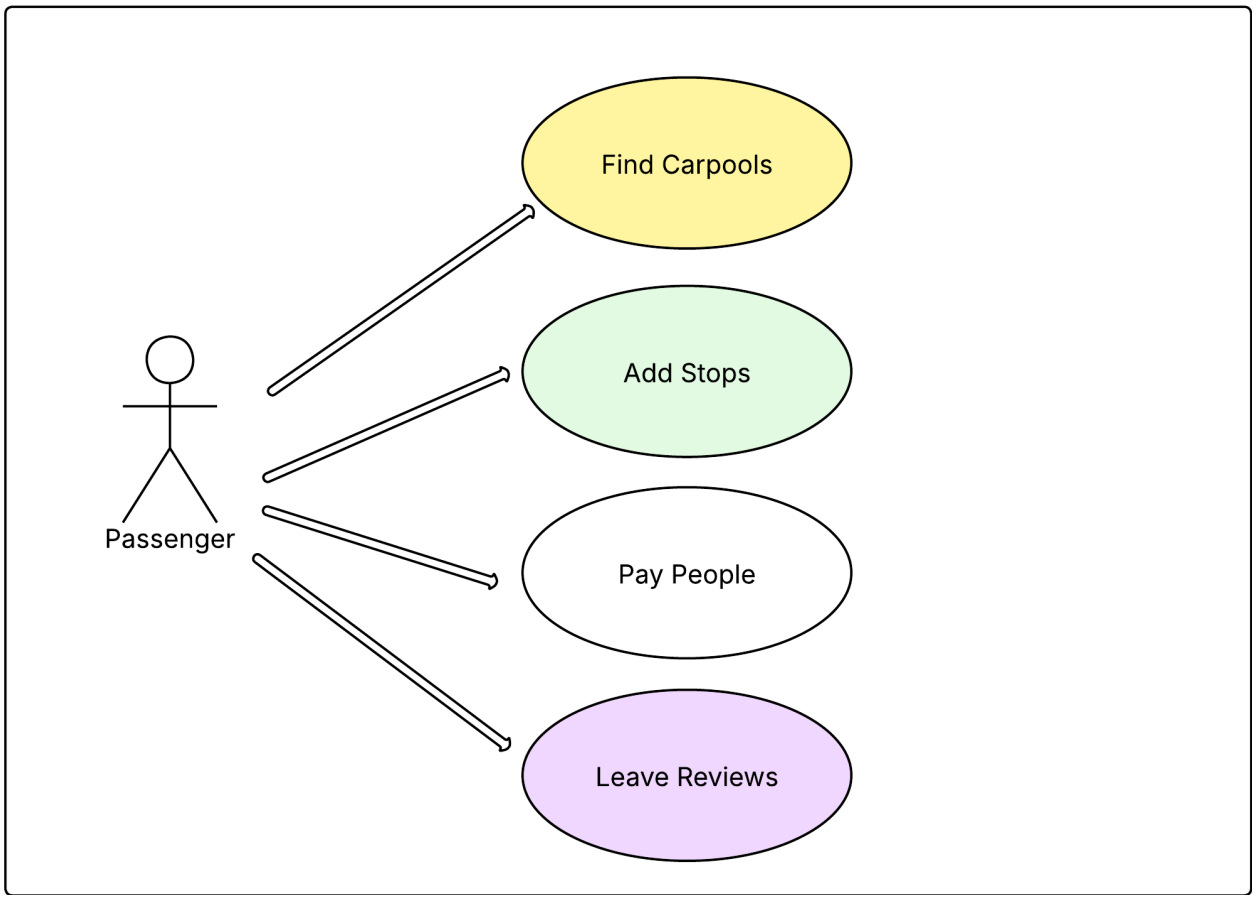
Use Case Diagrams

Driver Use Case Diagram

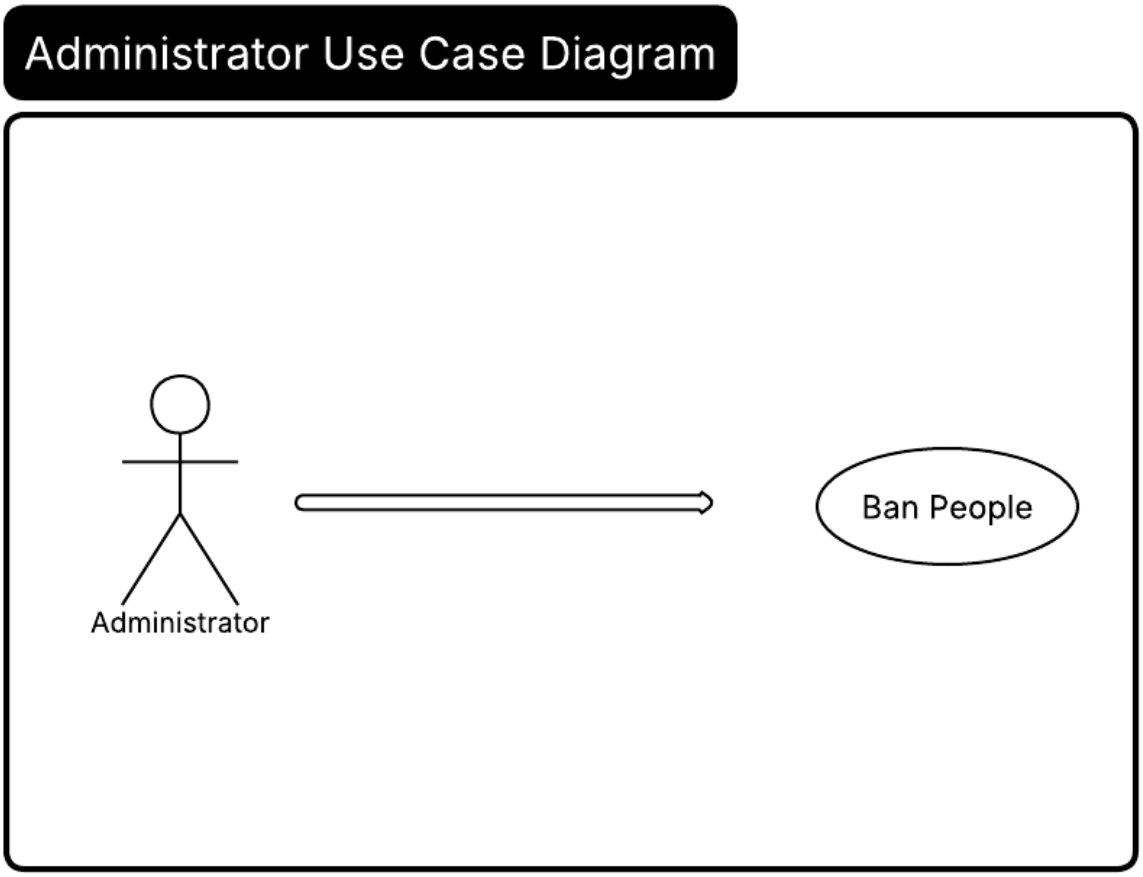


This is a use-case diagram demonstrating the functions a driver can access using the application.

Passenger Use Case Diagram



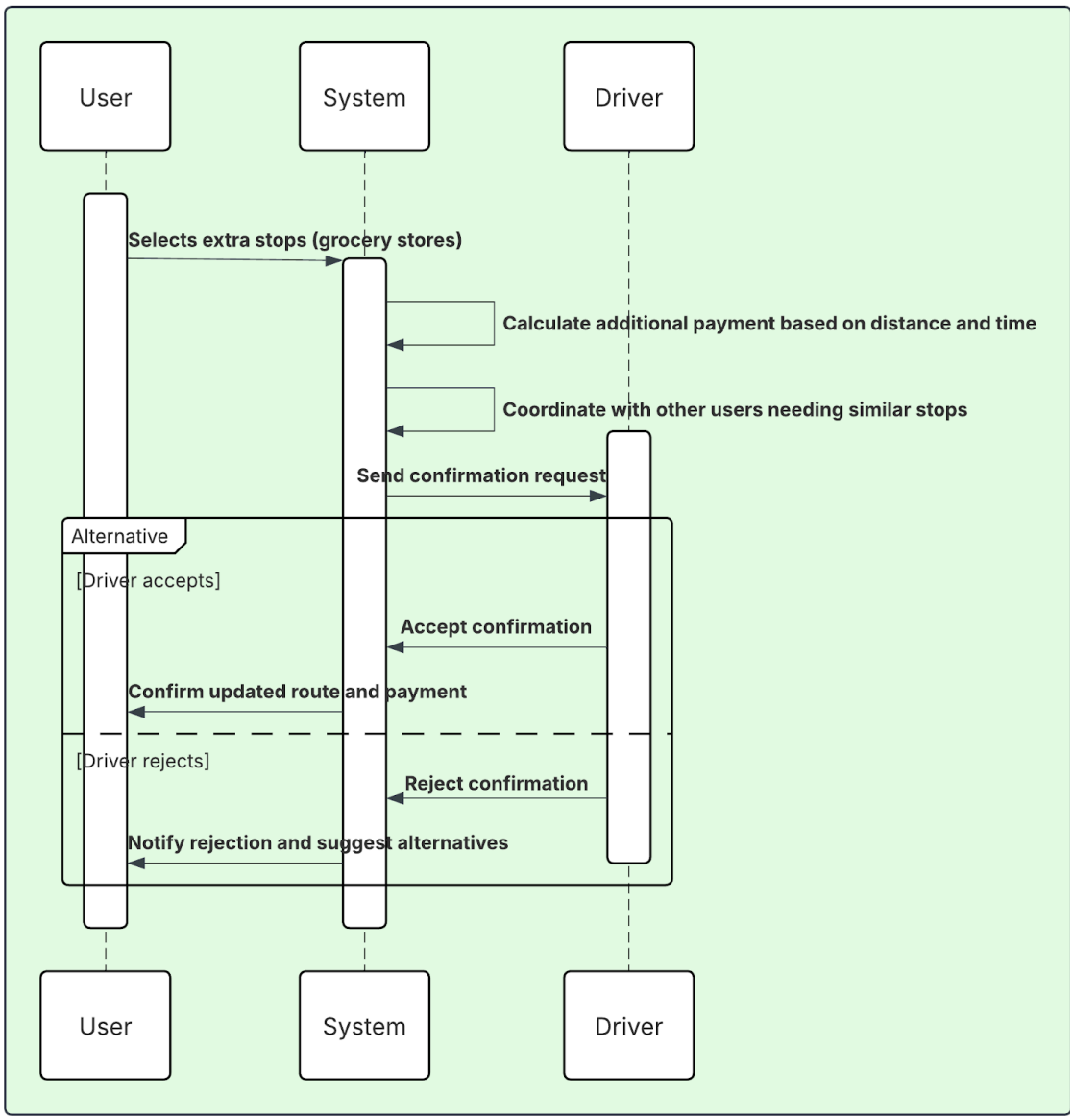
This is a use-case diagram demonstrating the features that a passenger can do with our application.



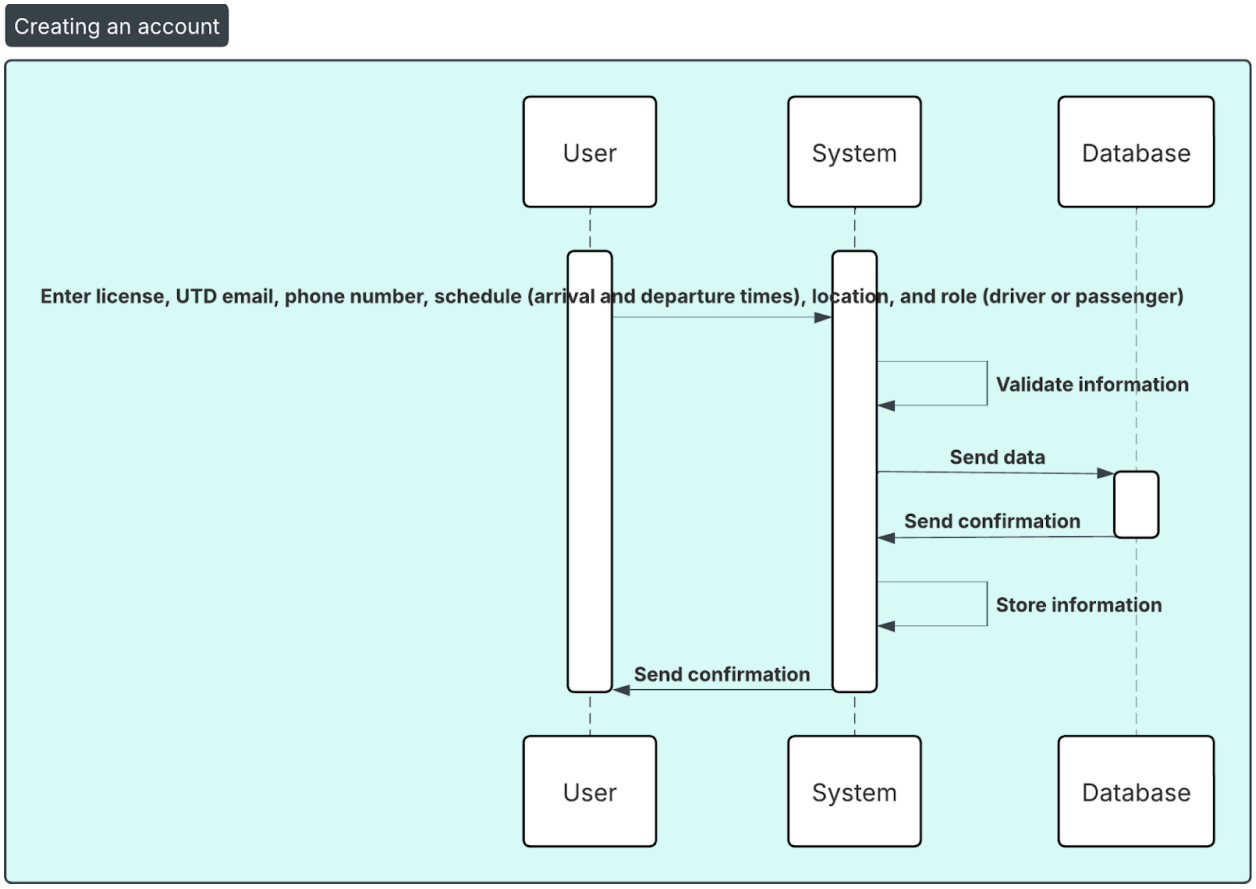
This is a use-case diagram that demonstrates the Administrators who can ban people who violate the rule and policies.

Sequence Diagrams

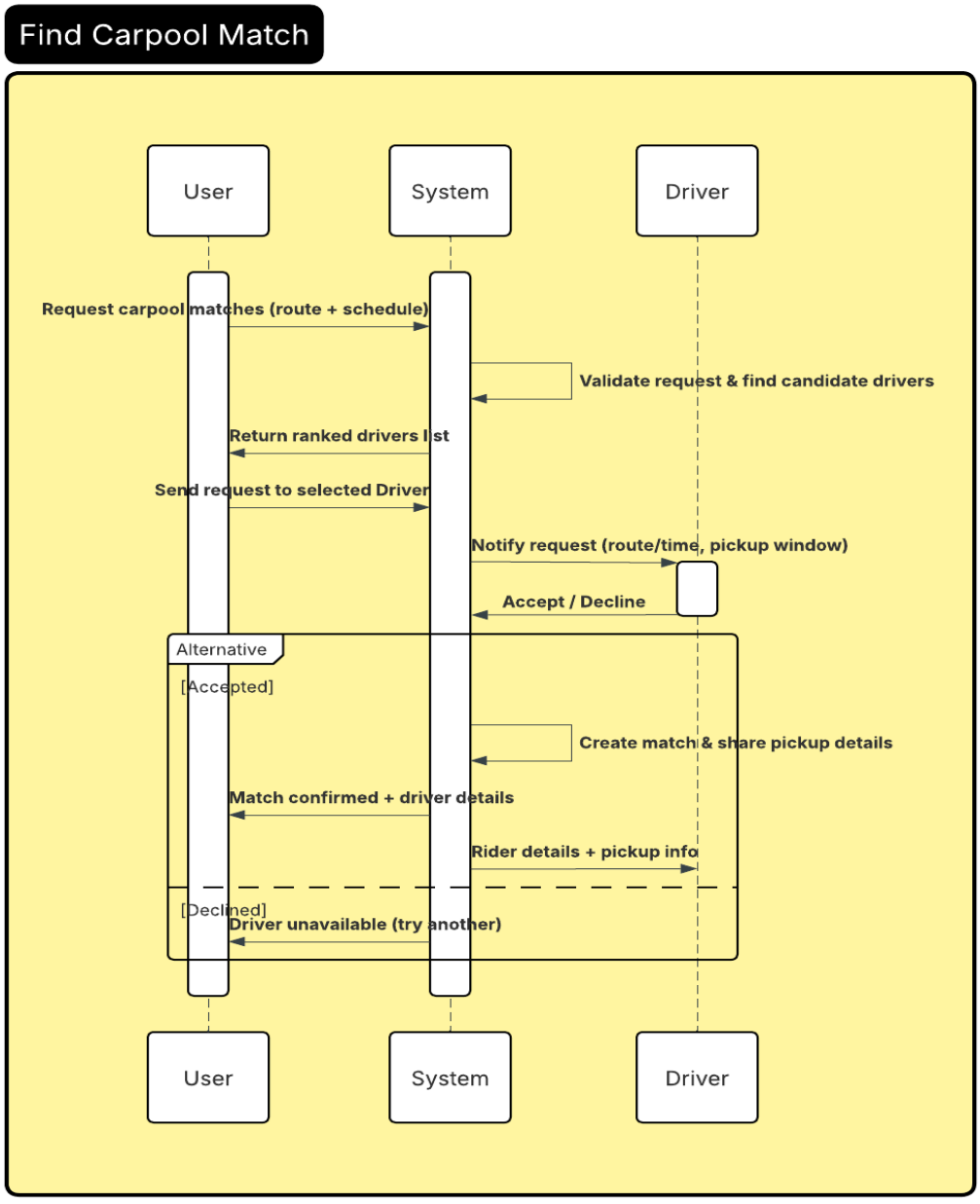
Add Extra Stops



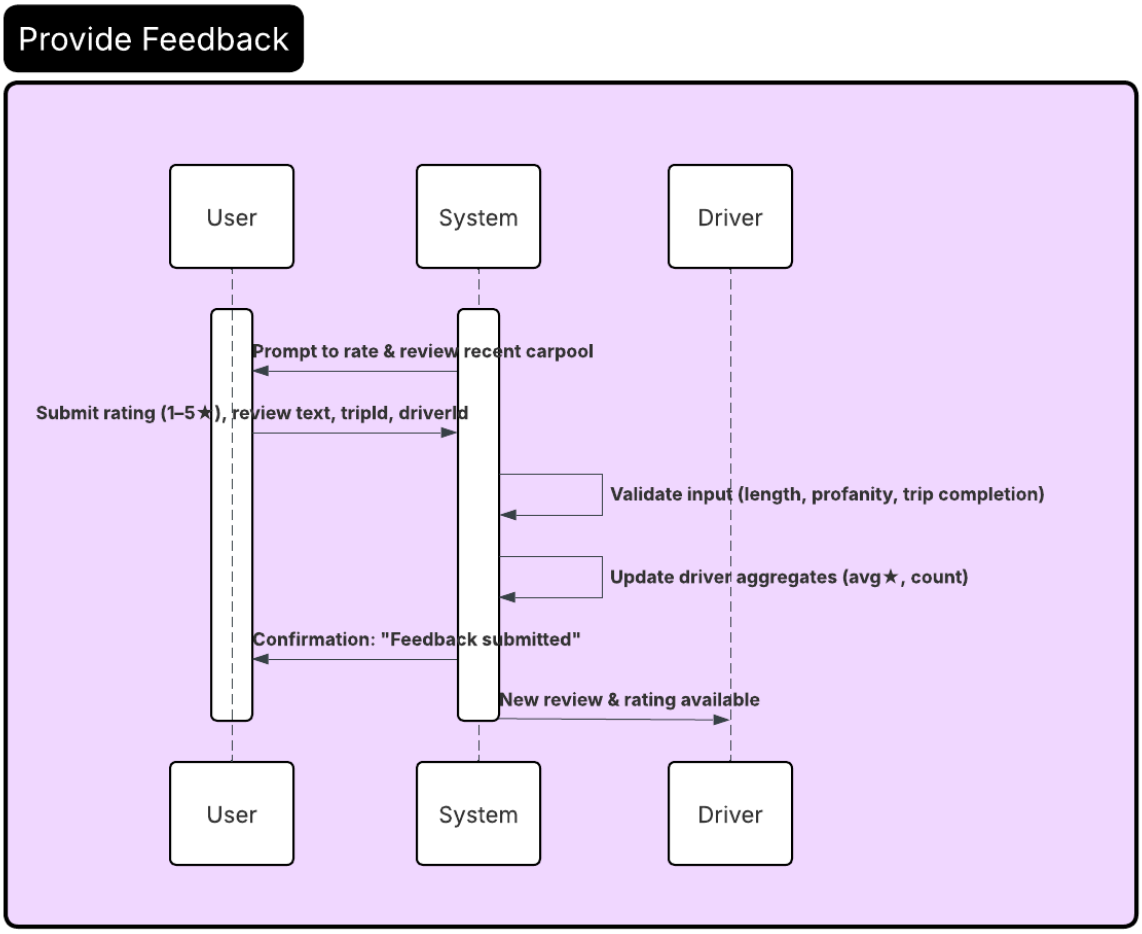
This Sequence Diagram illustrates the process of a user adding additional stops and coordinating with the driver.



This sequence-diagram shows the process of creating an account to sign up for the application.

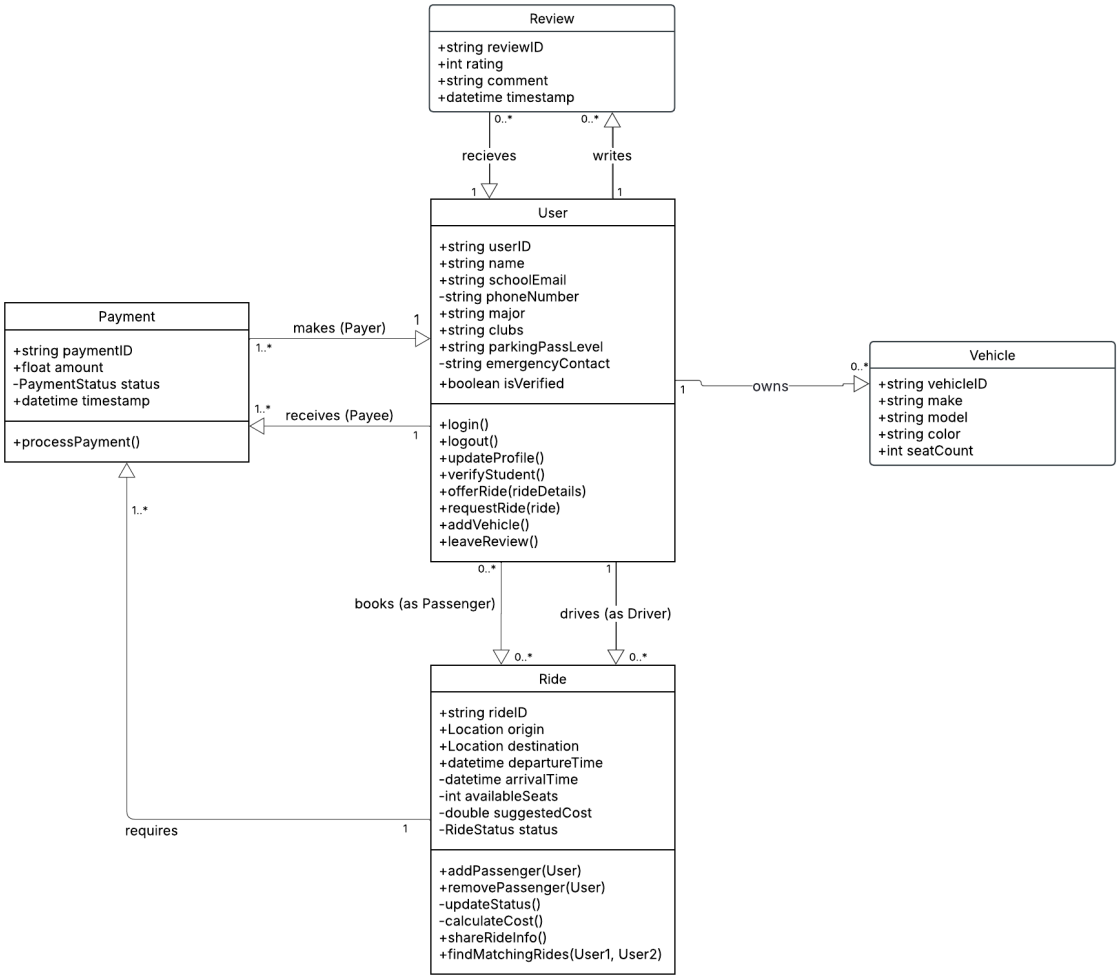


This is a sequence diagram that demonstrates the process of finding a carpool match between user and driver.

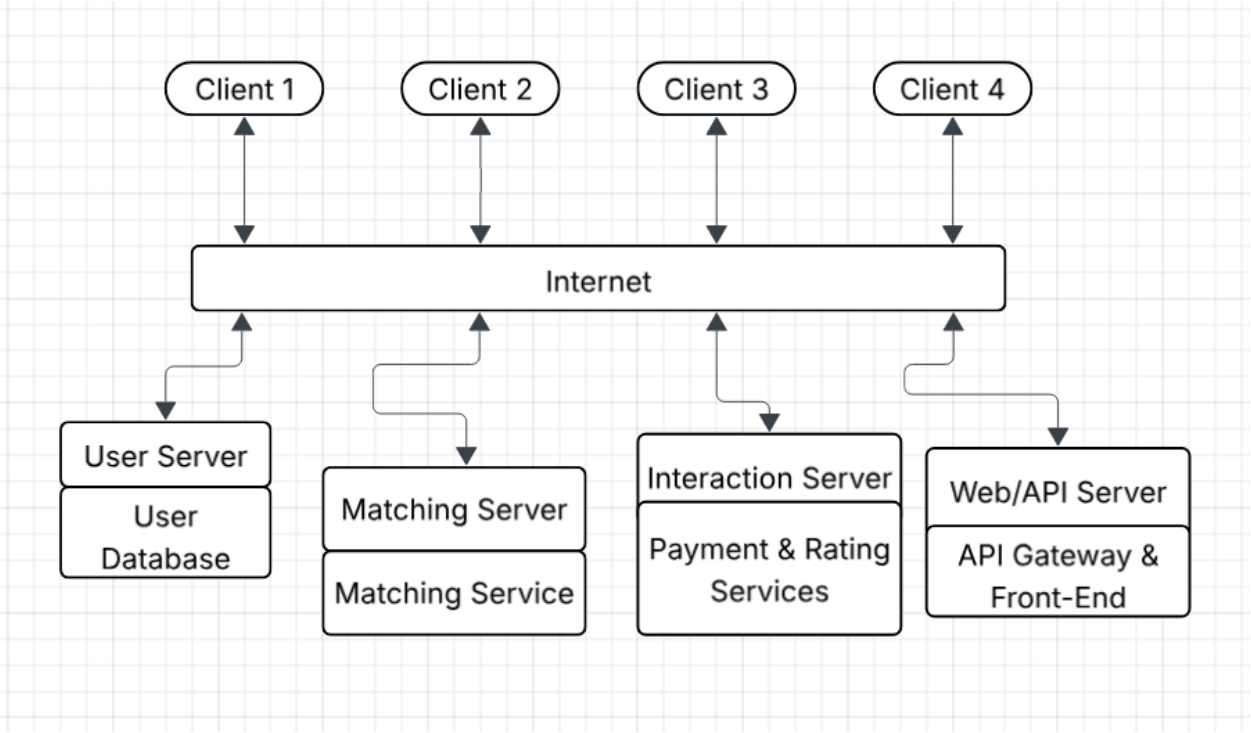


This is a sequence diagram that shows the process of how users provide feedback to drivers for the trip.

Class Diagrams



Architectural Design



Our application will follow a client-server architectural style as shown above. Users interact with the client interface, which communicates with different parts of the backend server for data processing and ride matching.

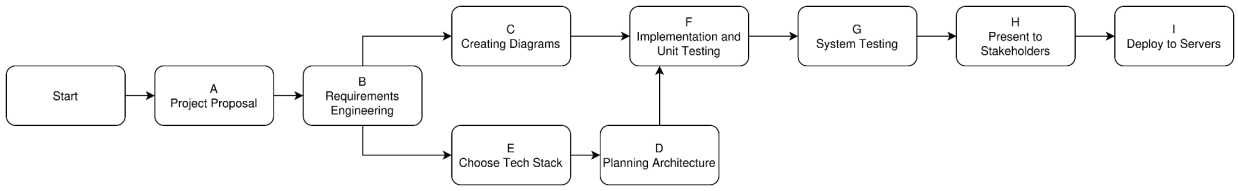
Project Scheduling

For our project, we need to complete the following activities. I have also included estimates for the amount of time they would take and the dependencies

Activity	Description	Preceding activity	Activity Time (weeks)
A	Project Proposal	None	1
B	Requirements Engineering	A	1
C	Creating Diagrams	B	1
D	Planning Architecture	E	1.5
E	Choose Tech Stack	B	0.5
F	Implementation and	C, D	2

	Unit Testing		
G	System Testing	F	1
H	Present to Stakeholders	G	0.5
I	Deploy to Servers	H	1

We chose a waterfall software process model, so our schedule is very linear with only a few things we can do in parallel. Below is the Activity on node diagram of our schedule.



Using this, we can calculate the latest and earliest start and finish times to determine the critical path

Earliest and Latest times									
A		B		C		D		E	
0	1	1	2	2	3	2.5	4	2	2.5
0	1	1	2	3	4	2.5	4	2	2.5
F		G		H		I			
4	6	6	7	7	7.5	7.5	8.5		
4	6	6	7	7	7.5	7.5	8.5		

From this, we determined that the critical path is: A → B → E → D → F → G → H → I
And the total time would be: 1 + 1 + 0.5 + 1.5 + 2 + 1 + 0.5 + 1 = 8.5 weeks

Project Schedule:

Task	Start Date	Due Date
Project Proposal	Nov 3, 2025	Nov 7, 2025
Requirements Engineering	Nov 10, 2025	Nov 14, 2025

Creating Diagrams	Nov 17, 2025	Nov 21, 2025
Choose Tech Stack	Nov 17, 2025	Nov 19, 2025
Planning Architecture	Nov 19, 2025	Nov 28, 2025
Implementation and Unit Testing	Dec 1, 2025	Dec 12, 2025
System Testing	Dec 15, 2025	Dec 19, 2025
Present to Stakeholders	Dec 29, 2025	Dec 31, 2025
Deploy to servers	Dec 31, 2025	Jan 7, 2026

The schedule assumes we work 8 hours a day and we do not work on weekends. Also, it assumes that we take the week off for Christmas.

Start Date: Nov 3, 2025
End Date: Jan 7, 2025

Cost, Effort, and Pricing Estimation

Below is the table we used to calculate the number of function points allocated for our project. The breakdown of the counts for each category is as follows.

User Input (8): Login, Signup, Ratings & Reviews, Gas Price Estimation, Edit Profile, Accept a Match, Provide Application Feedback, Search & Filter Matches
User Output (6): List of All Matches, List of Filtered Matches, View an Individual Profile, Suggested Gas Split Price, Notification Page, Driver Rating Overview
User Queries (6): Search Users, Filter by Time, Filter by Location, Check Reviews, Filter by Shared Extracurriculars, Filter by Rating
Data Files (5): Users, Match Records, Notifications, Feedback, Ratings & Reviews
External Interfaces (4): Google Maps API, UTD SSO Login, Email Service, Gas Prices API

	Function Category	Complexity				Count x Complexity
		Count	Simple	Average	Complex	
1	Number of user input	8	3	4	6	32
2	Number of user	6	4	5	7	30

	output					
3	Number of user queries	6	3	4	6	36
4	Number of data files and relational tables	5	7	10	15	50
5	Number of external interfaces	4	5	7	10	40
		Gross Function Point				188

We used the table below to calculate our PCA.

PC Question	**Score**
Multiple Installations	1
Performance Critical	3
Heavily Utilized Operational Environment	3
Complexity of Input, Output, Files, & Inquiries	3
Complexity of Internal Processing	3
Reliable Backup and Delivery	4
Distributed Processing	4
Data Communications	4
Online Data Entry	4
Reusable Code	4
Conversion and Installation	4
Input Transaction Over Multiple Screens	5
Master Files Updated Online	5
Ease of Use	5

The total number of **gross function points** for our project, calculated from the sum of the weighted counts of all function categories, is 188.

A **processing complexity adjustment** of 1.17 was calculated by summing all 14 complexity factor scores, multiplying the sum by 0.01, and adding 0.65.

Adjusting the gross function points by the processing complexity yields 220 **function points** (FP = GFP * PCA).

We assumed a **productivity** of approximately 4 function points per person per week. By dividing the function points by productivity, we determined an **estimated effort** of 55 person-weeks.

The **project duration** is the estimated effort divided by team size, which in our case was 7 people. Therefore, we predict that our project will take 8 weeks to complete.

Hardware Cost

Server (1500) + Database (1000) + Backup (500) = 3000

Software Cost

OS or DB License (1000) + Development Tools (500) + Monitoring/Deployment Tools (300) = 1800

Personnel Cost

People (7) * Dollars/Week (1000) * Weeks (8) + Training/Onboarding (4000) = 60000

Test Plan

Find Driver Test Plan:
The purpose of this test plan is to verify whether a driver correctly matches a rider's carpool request. The system will check the driver's location, destination, and pick up time against the rider's start location, destination, and schedule.

What Will Be Tested:

- Test if a driver has the same location as the rider
- Test if a driver has the same destination as the rider
- Test if the driver's pick up time fall within the rider's time window

Expected:

If all conditions match, the method should return true. If one or more conditions do not match, the method should return false. If the input of rider or driver is missing, the method should throw an IllegalArgumentException.

Test Code

```
public class CarpoolMatchTest {
  @Test
  void positiveMatchTest() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     Driver driver = new Driver("Plano", "UTD", 9);
     assertTrue(CarpoolMatch.findDriverMatch(rider, driver));
  }
  @Test
  void negativeMatchTest_WrongLocation() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     Driver driver = new Driver("Dallas", "UTD", 9);
     assertFalse(CarpoolMatch.findDriverMatch(rider, driver));
  }
  @Test
  void negativeMatchTest_WrongDestination() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     Driver driver = new Driver("Plano", "Frisco", 9);
     assertFalse(CarpoolMatch.findDriverMatch(rider, driver));
  }
  @Test
  void negativeMatchTest_WrongTime() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     Driver driver = new Driver("Plano", "UTD", 11);
     assertFalse(CarpoolMatch.findDriverMatch(rider, driver));
  }
  @Test
  void edgeCase_ExactTimeMatch() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     Driver driver = new Driver("Plano", "UTD", 8);
     assertTrue(CarpoolMatch.findDriverMatch(rider, driver));
  }
  @Test
  void exceptionTest_NullInput() {
     Rider rider = new Rider("Plano", "UTD", 8, 10);
     assertThrows(IllegalArgumentException.class,
```

```
            () -> CarpoolMatch.findDriverMatch(rider, null));
    }
}
```

Testing Results

Test Case:

- Positive test cases: Driver matches the rider's start location, destination, and pick up time.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: start = Plano, destination = UTD, pickup time = 9

  True, driver meets all requirement
- Negative test cases: Driver did not match with rider.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: start = Dallas, destination = UTD, time window = 9

  False, different starting location
- Negative test cases: Driver did not match with rider.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: start = Plano, destination = Frisco, time window = 9

  False, different destination
- Negative test cases: Driver did not match with rider.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: start = Dallas, destination = UTD, time window = 11

  False, different schedule
- Edge cases: Driver and rider had the exact schedule.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: start = Plano, destination = UTD, time window = 8

  True, the driver's schedule is at the time window of the rider.
- Exception testing: null input.
    - Input:
        - Rider: start = Plano, destination = UTD, time window = 8-10
        - Driver: NULL

  Caught the expected expectation.

Comparison of your work with similar designs

One of our major competitors is Lyft. Particularly at major campuses such as USC, Duke, and OSU, Lyft offers custom made programs for late night rides after buses, trains, and shuttles. One

important thing to note however is that Lyft also has a program for bikes and scooters. Because of their large network they are also able to integrate the service seamlessly [1][2]. This is of some concern as we don't have as large of a network currently so it will be hard to compete with the various means of transport that are offered such as bikes and scooters. However, we are aiming to solve a different purpose entirely in the form of provide a way for students to commute and coordinate trips because UTD is largely a commuter school in the heart of DFW. We also have a few other unique differentiators in the form of allowing students to coordinate trips for other purposes of travel such as grocery shopping which the original driver has already planned. This helps because the student who would normally need a Lyft to the store and back would need to book two separate rides as well as worry about the availability of the Lyft drivers on the way back. Here the users will sort of accompany the driver on their trips and just get their stuff on the drivers schedule. This makes it easy for the driver as they aren't going out of their way to go on certain stops and are instead just adding people to the trip. The passengers also don't have to worry about securing a return ride back to university because they will be going with the driver. We do plan on having an option for drivers to take requests for students sort of similar to current day Uber and Lyft which is a similarity across platforms.

Another major competitor is Hitch. This is a popular option for students and behaves similar to Uber except for long distances [3]. Particularly this allows students to go from city to city for major stops and events. One such particular use case is for travel to other college campuses. Students from Texas A&M often use Hitch to drive to UT Austin for Halloweekend as an example. Hitch offers two plans, one for Station to Station where you meet your driver at a nearby location, the other one is Door to Door which is more private and a direct route [3]. Here you have the option of travelling with friends as well to split the costs and can make it more or less affordable depending on the option you choose. However, it is important to note that the same features can apply for our application. Plenty of UTD students go to the same places as other college students, particularly many UTD students go to UT Austin for Halloweekend and might need to go home to other areas such as Houston. In these cases the same situation applies. Since there are similar events for students and most students head home on the holidays which can include long distances, those that have a car and travel can pick up other students on the way. Moreover, plenty of students go to events such as Halloweekend which ensures availability and a sense of trust as they are going with UTD students. Overall given the nature of our application which is a rideshare application for UTD students there isn't really a need to differentiate between going long distances or staying local as it really depends on the drivers mood. In this sense we currently don't need to separate customer segments as Hitch and Lyft currently do. One thing that is interesting is that our popular locations aren't really emphasized in our application but are rather implied based on the drivers mood. It would be interesting to add this to our application such as a Dallas to Austin trip and then show availability based on who is going.

Another competitor is Waymo. Waymo provides self-driving cars which are available 24/7 to

travel to your location of choice [4]. This is unique since it removes the people entirely which can mean safer driving, as well as increased safety as you don't need to worry about the background of the person that is driving you or any other hazards. However, Waymo is relatively new and still does not service our location in DFW and has only extended itself to Austin and Atlanta recently. It is possible for Waymo to expand its distribution but they appear to be taking things carefully meaning that they are unable to compete in our market. Secondly, we aim to differentiate on the human aspect, since our drivers are from UTD, there is a sense of common ground and identity in a sense as driver and passenger can both talk about UTD life as well as have that implied trust and security among themselves. Another issue outside of distribution is price. On average at peak times Waymo can tend to cost about $9.50 more than Uber and $11 more than Lyft [5]. Considering our demographic is a college student, increased costs compared to our platform also makes choosing Waymo an issue. Moreover, for nearby trips to areas around UTD, it is hardly worth paying that hefty premium. Waymo could be a true competitor if it moves into the DFW marketshare and decreases prices to compete with us, but given that Waymo is targeting different locations at the moment that seems unlikely.

One of our key competitors is Uber. Uber has vast options to book rides. One such element that is similar to our platform is UberPool, UberPool has key elements of our business idea which is to allow riders to ride in the same direction as others [6]. This allows the drivers to pick up multiple passengers along the way for more profit, also allowing more people to ride Uber as well. However, it is important to note differences. UberPool is a commercial ride-sharing service focused on lowering costs for public urban transportation [6]. Meanwhile our platform is mainly for verified UTD students and staff with their identification to send rides to and from campus. UberPool also allows anyone to sign up with a phone and a payment method whereas for our platform the requirements are a lot stricter. Our overall algorithm and criteria for matching is different as well as we match based on a variety of factors such as matching schedules, parking pass zones, departure and arrival time, and majors and interests. We focus on the community and culture of our UTD community. In contrast UberPool mainly focuses on route efficiency and cost [6]. Another such thing is UberPool requires everything to be scheduled in advance which can sometimes prevent students from acting out on their situations. For example, if I am craving a midnight snack and have to go through the hassle of finding an UberPool or Uber and then need to wait for the person to arrive, I might lose interest. In contrast, we offer spontaneous matching without requiring scheduling as well as a scheduling option. This way our users can plan out their options or go as soon as possible. We also place an emphasis on our community and culture as reiterated above, whereas Uber and other competitors addressed are focused on a transactional relationship. Uber itself offers other options such as delivery of food in the form of UberEats for a fee, and does offer integration with popular food companies with deals and incentives such as a buy one get one free at McDonalds [7]. However, although we don't have the company backing we have other options such as not having an emphasis on tipping culture or gratuity which is

emphasized in Uber and other platforms.

One commercial competitor is Scoop. Scoop is primarily a corporate carpooling system designed to connect commuters at major corporations for daily commutes to work [8]. We on the other hand are mainly for UTD students. Our target market differs with Scoop being focused on working professionals and we are focused on UTD staff and students. The verification systems are similar with Scoop having a employee verification system to make sure that you work at the company while we do the same with making sure you are a verified UTD student. A key difference is that Scoop focuses on planned commutes in the form of morning drives and evenings back, and users need to confirm a time by the evening for a ride the next day [8]. We as mentioned offer flexible planned and on demand scheduling for our UTD students depending on class schedules or spontaneous trips. The matching algorithm for Scoop is the most similar to ours as it takes in the drivers schedules into account as well as the workplace locations, we do the same with class scheduling but take it a step further with parking zones as well as student preferences based on clubs, interests and hobbies. Scoop's platform is also similar with company culture and networking while we do the same at UTD to bring students together and improve school spirit. Overall, Scoop is by far the most similar to what we are trying to achieve at UTD however we have differences in target markets and more specialization for our UTD students.

Conclusion

The Comet Commuter Carpool has progressed from the initial proposal phase to a more detailed architectural design and feasibility analysis. The Waterfall software process model has allowed us to have a clear linear progression starting from requirements gathering to system design. This made sure that all functional requirements like ride matching, profile management, and safety integrations were defined before estimation began, allowing us to actually estimate the product more easily than if we had chosen an agile method.

We used the Function Point estimation method to confirm the project's feasibility in terms of financial commitment. With a projected timeline of about 8.5 weeks and an estimated total cost of $64,800, the project is feasible to complete with a 7 person team. We chose a Client-Server architecture which aligned with our needs for centralized data management of user schedules and routes, while securing any personal identifiable information. The comparison with other ride scheduling and ride share apps like Lyft, Uber, and Scoop showed that our product fills a clear gap by providing carpooling for UTD students. While these more commercialized companies focus on efficiency, our platform provides a unique value based on campus security and social connection with other UTD students.

Deviations from Original Plan

One change that we made since the project proposal is that we split up the use cases and sequence diagrams between Aman and Andy. This decision was made because the sequence

diagrams turned out to be a lot more work than the use case diagrams, thus it required more people to share and complete the workload.

In addition, the GitHub repository shows it is owned by Aarush, while Vincent created the original repository. Ownership had to be transferred to Aarush so he could add the rest of the team. While we understand this was not the most efficient way to handle this, we had to split the GitHub tasks because different people had to do each task.

Finally, we included more than the three to four use cases in our documentation because we wanted to fully flesh out the application's different functions. However, we only created the sequence diagrams for four of them. This aligns with the professor's instructions that we only need three to four use cases and a corresponding sequence diagram for each of them, allowing us to be comprehensive in design while meeting the requirements.

In conclusion, the Comet Commuter Carpool is robust, accurately estimated financially, and tailored to the needs of the University of Texas at Dallas students. The team was able to adapt to logical challenges during the planning phase, which resulted in a cohesive and actionable software design document.

References

[1] "Ride Smart Around Campus with Lyft," *Lyft Blog*, https://www.lyft.com/blog/posts/ride-smart-around-campus-with-lyft (accessed Nov. 11, 2025).

[2] "Higher Education Transportation Solutions," *Lyft Business*, https://www.lyft.com/business/industries/higher-education (accessed Nov. 11, 2025).

[3] "Hitch | City to City Uber & Long Distance Rideshare/Carpool App," *Hitch*, https://www.hitch.com/ (accessed Nov. 11, 2025).

[4] "Waymo - Self-Driving Cars - Autonomous Vehicles - Ride-Hail," *Waymo*, https://waymo.com/ (accessed Nov. 11, 2025).

[5] "Here's How Much A Waymo Ride Will Cost You Compared To Uber And Lyft," *SlashGear*, https://www.slashgear.com/1993145/waymo-robotaxi-ride-cost-vs-uber-lyft/ (accessed Nov. 11, 2025).

[6] "What is UberPool?" *Uber Help*, https://help.uber.com/driving-and-delivering/article/what-is-uberpool?nodeId=b1d57ab0-2bb0-4db7-9d72-5c7a43c5a2ce (accessed Nov. 11, 2025).

[7] "Explore the Uber Platform," *Uber*, https://www.uber.com/ (accessed Nov. 11, 2025).

[8] "Carpooling Software | Scoop Commute," *Scoop*, https://www.scoopcommute.com/ (accessed Nov. 11, 2025).