Imam Mohammed Ibn Saud

Islamic University

Computer and Information

Science

Department: Computer Science

Course: Software Engineering 1, Section 171, CS1350

Semester: First Semester, 1447, 2025

Project title: Ride-Sharing Application

Instructor's name: Prof. Gufran Ahmad Ansari

Team members

Name	ID
Turki Talal Alotaibi	445011243
Ahmad Suliman Algadheeb	445018669
Sultan Abdulraman Alanazi	445012456
Naif Ibrahim Almohaimeed	445010735
Yazeed Abdulrahman Alsalamah	444000868

Phase 1

TABLE OF CONTENTS

Introduction

It is hard to travel across campus for most students. Other days the bus schedules don't match with your clubs or classes, and ride-share apps cost too much from your pocket for daily rides. The majority of students end up waiting too long for a bus or have to walk excessively, which is a waste of time and energy. Students thus need an accessible, low-price, and effective way of traveling across campus quickly.1

The Student Ride-Sharing Application is here to bring that problem to an end. The application allows students to find and meet each other, traveling to the same place at the same time. For example, if two students are in the same building for class or if they live nearby, they can carpool instead of driving alone. It is easier, faster, and cheaper for everyone to get there.1

Security is also greatly prioritized in this application. All users must log in with their real university account, and therefore only actual students have access to it. Students can see the ratings and reviews of the driver's or rider's before accepting a ride so they can be sure that they are trustworthy. This makes sure that every user has a safe and comfortable experience.1

It is more than a transportation application; it's a tiny community builder. It allows students to meet fellow classmates, save money, and reduce traffic and parking congestion on campus. Carpooling also allows students to decrease their environmental impact by driving fewer vehicles.

In the long run, the Student Ride-Sharing Application is more than a way of getting from one place to another. It's an initiative that makes college life easier, safer, and more connected. It allows students to assist one another while saving them money and time.

Project Scope

This project is about making a ride sharing application for students in the university. The idea is to help students find rides and share them with others in a safe and easy way. The application will help them plan rides, split money costs, and talk to each other. It will also make sure every user is a real student, not from outside.2

We want to add safety things like an emergency button and rating system so students can feel more safe when they use it. We will use Agile (Scrum) and work in small sprints to finish step by step.2

The application is only for students, not for public or long rides. In the end we just want to make a simple and safe application that makes student life easier.

Objectives

1. Build a community you can trust.

- Help students connect with classmates for their regular rides.3
- Create a network where students feel comfortable sharing rides with each other.3
- Make it easy to pay your share of the ride automatically.

2. Make it cheap and easy.

- Cut the cost of a single trip in half compared to applications like Uber or Lyft.3
- Make it fast to find a ride, with most matches happening in under three minutes.
- Make the application work smoothly on both iPhone and Android.

3. Give students more control.

- Let students book a ride last-minute or schedule one for later in the week.3
- Allow riders to choose drivers based on preferences like gender or shared classes.
- Make sure the application's map knows all the best pick-up spots and shortcuts on campus.

4. Keep everyone safe.

- Make sure every user is a real student by checking their school email.3
- Let students see ratings for drivers and passengers before they ride.
- Include a panic button to quickly get help in an emergency.

5. Grow with the campus.

- Plan for big campus events like football games or finals week by adding extra ride options.
- Let students form groups for dorms to organize rides together.
- Keep the application updated based on real feedback from students who use it every day.

Target Audience

Who This Application Is For

This application is built for college students. It's for anyone on campus who needs a better way to get around. Here's who will use it most:

Students Who Need Rides:

- Freshmen and international students who don't have cars yet.3
- Students living off-campus who are tired of bus schedules
- Anyone studying late at the library who needs a safe ride home.3
- Students trying to save money on transportation

Students Who Can Drive:

- Commuters with empty seats in their car
- Students looking to make some extra gas money
- People who want to meet others from campus

Basically, if you're a student trying to save money, make new friends, or just get around campus easier - this application is for you. Everyone is verified as a real student, so you know you're riding with people from your own campus community.3

System’s features

Our app provides a wide set of features that are designed to allow this application to be practical and user-friendly, also affordable and convenient, these are the system’s features listed. 4

1. Authentication and verification the user

All users must sign in using the university emails only to assure that all users of this application are students.4

2. Profile Management

Every student can create a customizable profile, including data like name, photo, address, email and phone number, and data like car’s information and license plate.

3. Ride Matching

This feature automatically matches riders with nearby drivers and shared destination riders, but also sticking to let them filter their preferences such as gender, route and timing.4

4. Ride Scheduling

Rides can be requested by students instantly or scheduled.

5. Real-Time Tracking

Driver’s location will be tracked by students, also including the pick-up point, destination and the estimated time for arrival of the driver or the ride time.

6. Group and Event Rides

Group-rides can be created by students for transporting dorms, clubs and events, this helps organizing big shared rides when large occasions are hosted.5

System’s capabilities

These listed capabilities allow this app to be efficient and introduce the students to easy, safe and convenient rides. 4

1. Data Management Capability

Our application has the capability to store, manage and update students data, including history of rides and their feedback, having this data allows our application to perform better.4

2. Rating and Feedback Capability

Ratings and feedback are important to manage drivers, this overtime builds trust in the application and makes the future of rides continuously improved.6

3. Safety and Emergency Capability

Safety is a number one priority, so we include an emergency button in our application that informs the security from the campus and logging every student and ride for security measures.

4. Payment Processing Capability

Providing as many payment methods to ease transactions process for students, also calculate and split the trip cost automatically.

5. Scalability Capability

Every university’s needs can fit our design since the rides are scalable with more students when events are hosted, so as the university expands our application will.6

6. Communication Capability

This application allows messaging between riders and drivers, communication between them is important, for example agreeing on the exact pick up point.

Application Benefits

Our application offers multiple benefits that guarantee customer acquisition, including:

1. Low-cost rides

This application lets students share car trips inside or around campus, so everyone pays less. By splitting fuel costs between riders, it becomes an affordable way to move between classes or dorms.5

2. Saves time

Students can find a ride in just a few minutes instead of waiting for buses or walking long distances. The application connects people who are already heading the same way, making transportation quicker.5

3. Safe and trusted

Every user signs up with a university email, which keeps the community limited to real students. Features like driver ratings and an emergency button give extra peace of mind.5

4. Social interaction

While saving money, students also get a chance to meet classmates and make new friends during their rides.5

5. Simple to use

The application handles everything in one place — booking, tracking, and payment — so students don’t need to switch between different tools.

Application Characteristics

The app has some characteristics that we will list as follows:

1. For students only

Registration requires a valid university email to make sure all users belong to the same campus and not outside.7

2. Easy design

The layout is clear and simple, so users can find and book rides quickly without confusion and also without missing his time and effort.7

3. Data and payment protection

All personal information and transactions are safely handled to keep user privacy secure.6

4. Works on all devices

Whether you as a student use Android or iPhone, the application runs easily on both platforms. So no worries in that regard.6

5. Stable performance

The system can support heavy use during busy times like exams or big events without lagging.

6. Upgradable and flexible

The development follows short, planned stages so new ideas and feedback from students can be added easily.

Scrum Team Members Sheet

Column (A) has each member's first name — Ahmad, Turki, Naif, Yazeed, Sultan. The purpose of this sheet is to note who is in the team and the project organization is evenly spread among members.This sheet also provides a single point of reference for the entire project. By listing all team members in one place, it helps instructors and team members rapidly identify who is participating in the project. In addition, it supports proper documentation practices, critical in software engineering.

Each of the members listed under Column A is to contribute to various tasks spanning the product backlog and sprint cycles. Having their names clearly displayed ensures that work can be assigned equitably and responsibilities are clearly determined. This also allows for easy communication in the future, the planning of meetings, and the coordination of tasks among members. Document history created by Ahmad on 2025-10-06

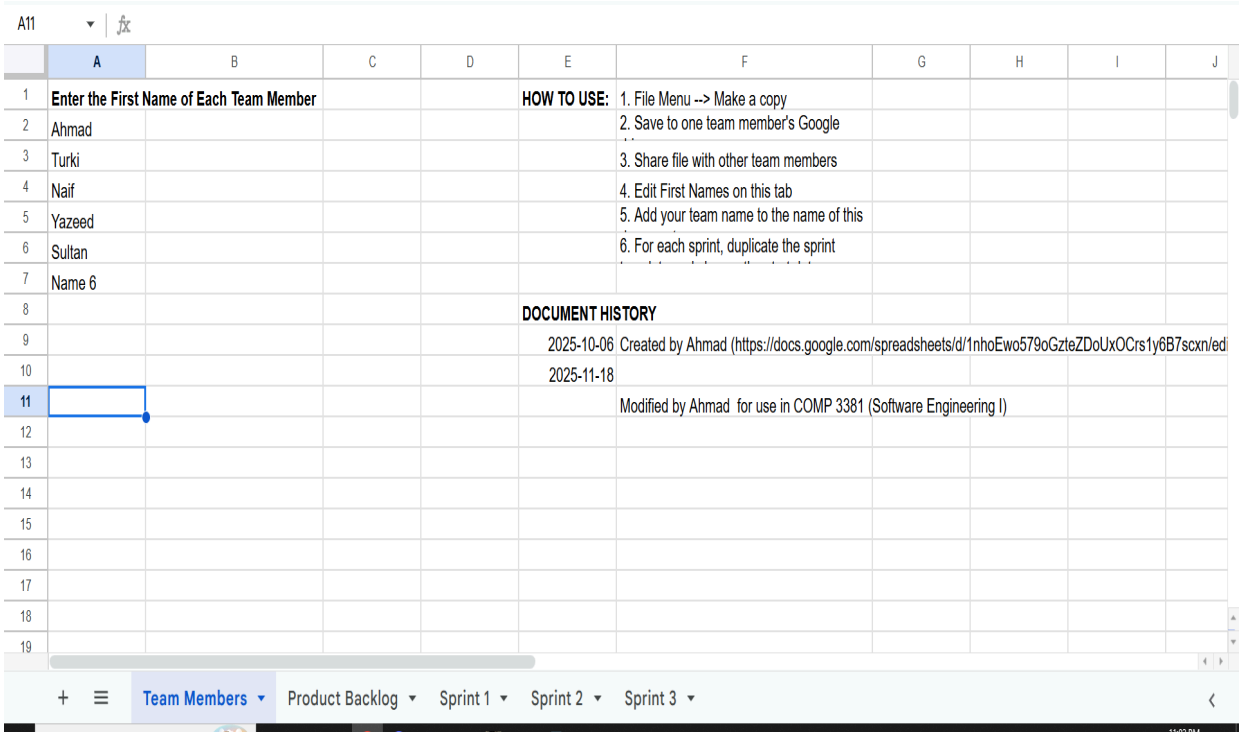


Figure 1 shows scrum members sheet.

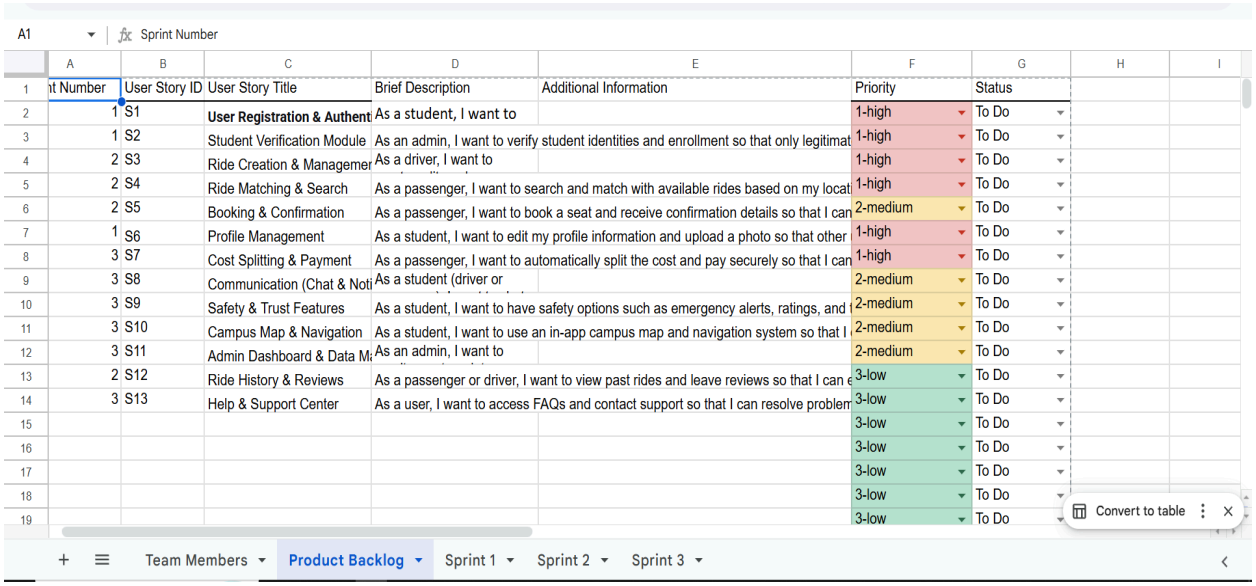
Scrum Product Backlog Sheet

This sheet displays the whole Product Backlog, including all the user stories of the Student Ride-Sharing Application project.

All the stories are numbered according to the sprint that they fit into, which makes planning easier and also helps to track progress.

This document serves as the master guide to all system requirements and helps the team plan and prioritize development efforts.This sheet also provides a clear breakdown of each User Story ID, title, and brief description, with anything additional that might be needed to support development. Organizing user stories in this structured manner helps the team understand which functionality needs to be created and for what purpose.

The priority column visually indicates which items are of high, medium, or low importance; this way, the team can put most of their concentration on the features that will provide more value to the users. The status column indicates the state each story is in at the moment and helps the team keep track of the progress throughout sprints, making it easier to identify pending tasks. Document history created by Ahmad on 2025-10-06



	A	B	C	D	E	F	G	H	I
1	ıt Number	User Story ID	User Story Title	Brief Description	Additional Information	Priority	Status		
2	1	S1	**User Registration & Authenti**	As a student, I want to		1-high	To Do		
3	1	S2	Student Verification Module	As an admin, I want to verify student identities and enrollment so that only legitimat		1-high	To Do		
4	2	S3	Ride Creation & Manageme	As a driver, I want to		1-high	To Do		
5	2	S4	Ride Matching & Search	As a passenger, I want to search and match with available rides based on my locati		1-high	To Do		
6	2	S5	Booking & Confirmation	As a passenger, I want to book a seat and receive confirmation details so that I can		2-medium	To Do		
7	1	S6	Profile Management	As a student, I want to edit my profile information and upload a photo so that other		1-high	To Do		
8	3	S7	Cost Splitting & Payment	As a passenger, I want to automatically split the cost and pay securely so that I can		1-high	To Do		
9	3	S8	Communication (Chat & Noti	As a student (driver or		2-medium	To Do		
10	3	S9	Safety & Trust Features	As a student, I want to have safety options such as emergency alerts, ratings, and		2-medium	To Do		
11	3	S10	Campus Map & Navigation	As a student, I want to use an in-app campus map and navigation system so that I		2-medium	To Do		
12	3	S11	Admin Dashboard & Data Mi	As an admin, I want to		2-medium	To Do		
13	2	S12	Ride History & Reviews	As a passenger or driver, I want to view past rides and leave reviews so that I can e		3-low	To Do		
14	3	S13	Help & Support Center	As a user, I want to access FAQs and contact support so that I can resolve problem		3-low	To Do		
15						3-low	To Do		
16						3-low	To Do		
17						3-low	To Do		
18						3-low	To Do		
19						3-low	To Do		

Figure 2 Shows scrum product backlog sheet.

Scrum Sprint 1 sheet

This sprint includes User Registration & Authentication and Student Verification Module.
Each task has anticipated hours, assigned team member (Ahamd), status, and due date.This sheet provides the team with a clear view of what work needs to be completed during Sprint 1 and how many hours each task is expected to take. By listing both expected hours and actual hours, the team can later evaluate the accuracy of their estimates and improve the planning of future sprints.

The "Status" column helps track the status of each task, whether it is pending, in progress, or finished. This helps add a measure of transparency towards milestone completion. The "Date Completed" column creates a record for when each task was completed and can be helpful for the review process at the end of the sprint. Document history created by Ahmad on 2025-10-06

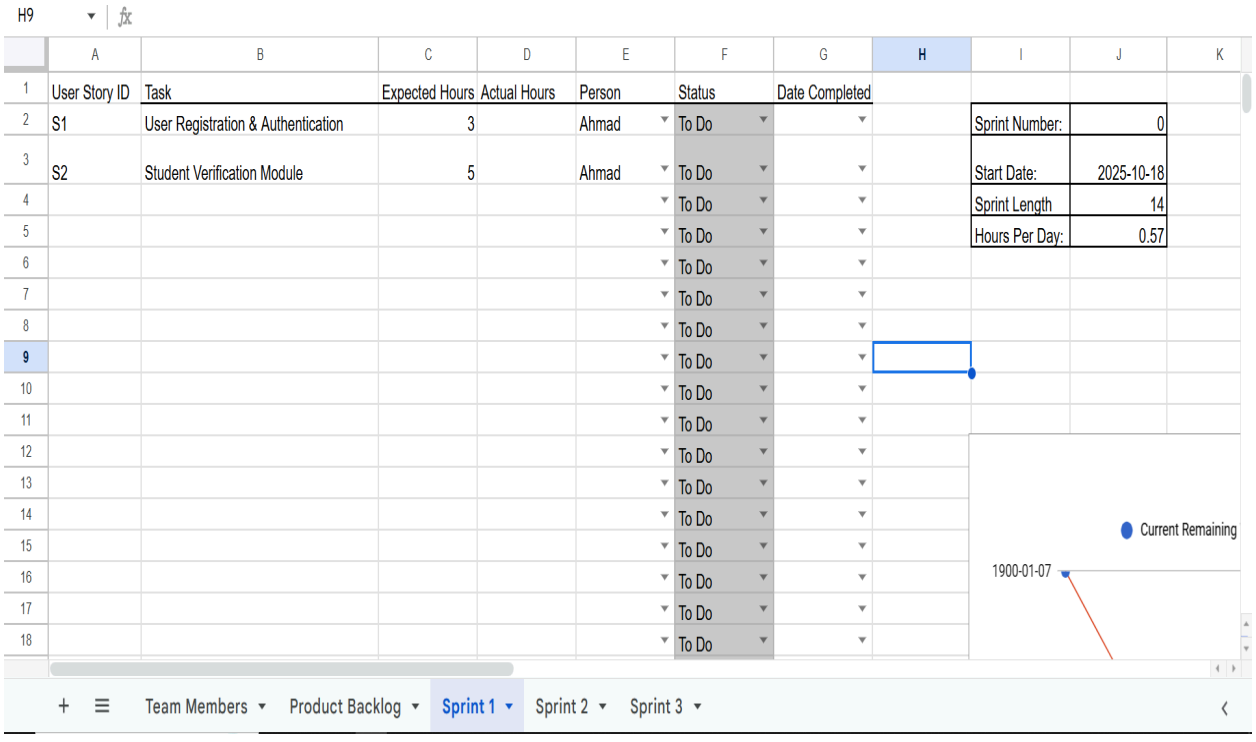
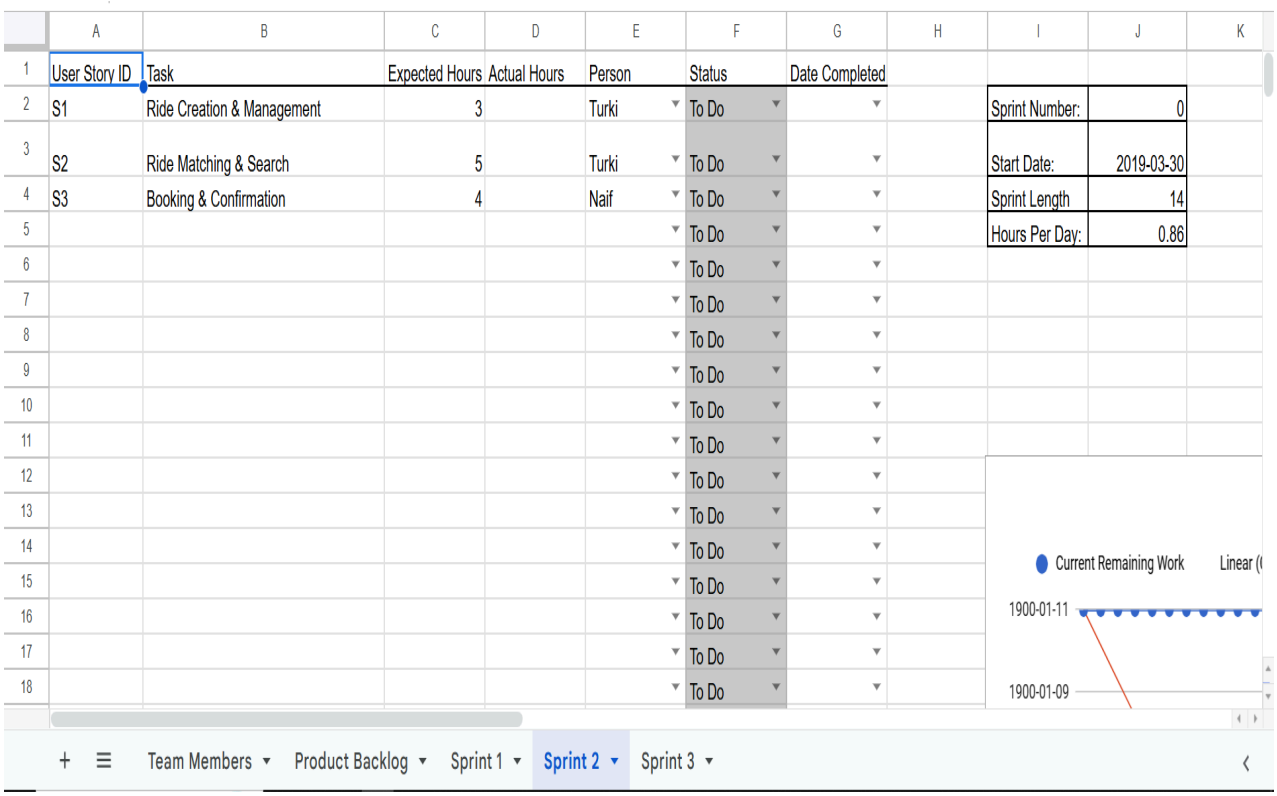


Figure 3 Shows scrum sprint 1 sheet.

Scrum Sprint 2 Sheet

This sheet contains the Sprint 2 information that includes activities such as Ride Creation & Management, Ride Matching & Search, and Booking & Confirmation. This sheet helps the team monitor (Turki and Naif) progress in Sprint 2 as well as work distribution among the developers.This sheet also gives a clear breakdown of estimated hours for each task, which allows the team to better understand the workload and plan their time effectively throughout Sprint 2. Later on, by comparing the expected hours with the actual hours taken, the team is able to gauge their accuracy in estimation and improve the planning at future sprints.

The "Person" column indicates the developer in charge of a specific activity. This adds accountability and transparency within the team. The "Status" and "Date Completed" columns provide details regarding the progress or completion date of tasks. The view makes it easy to see which work items remain pending and which are finished.Document history created by Ahmad on 2025-10-06



User Story ID	Task	Expected Hours	Actual Hours	Person	Status	Date Completed
S1	Ride Creation & Management	3		Turki	To Do	
S2	Ride Matching & Search	5		Turki	To Do	
S3	Booking & Confirmation	4		Naif	To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	
					To Do	

Sprint Number:	0
Start Date:	2019-03-30
Sprint Length	14
Hours Per Day:	0.86

Figure 4 Shows scrum sprint 2 sheet.

Scrum Sprint 3 Sheet

It caters to features such as Cost Splitting & Payment, Communication (Chat & Notifications), Safety & Trust Features, Campus Map & Navigation, and Admin Dashboard & Data Management.

This sheet allows the team to track the completion of the previous sprint and determine whether the project is still on schedule.This sheet also helps in delegating task responsibilities among the team members, as a clear indication of the person assigned to each user story is present. By listing the expected hours for every task, the team can estimate the total workload for Sprint 3 and make sure the remaining work can fit within the duration of the sprint. Once the actual hours are filled out later, the team would be able to assess how efficiently the tasks were completed with respect to the initial estimations. Document history created by Ahmad on 2025-10-06

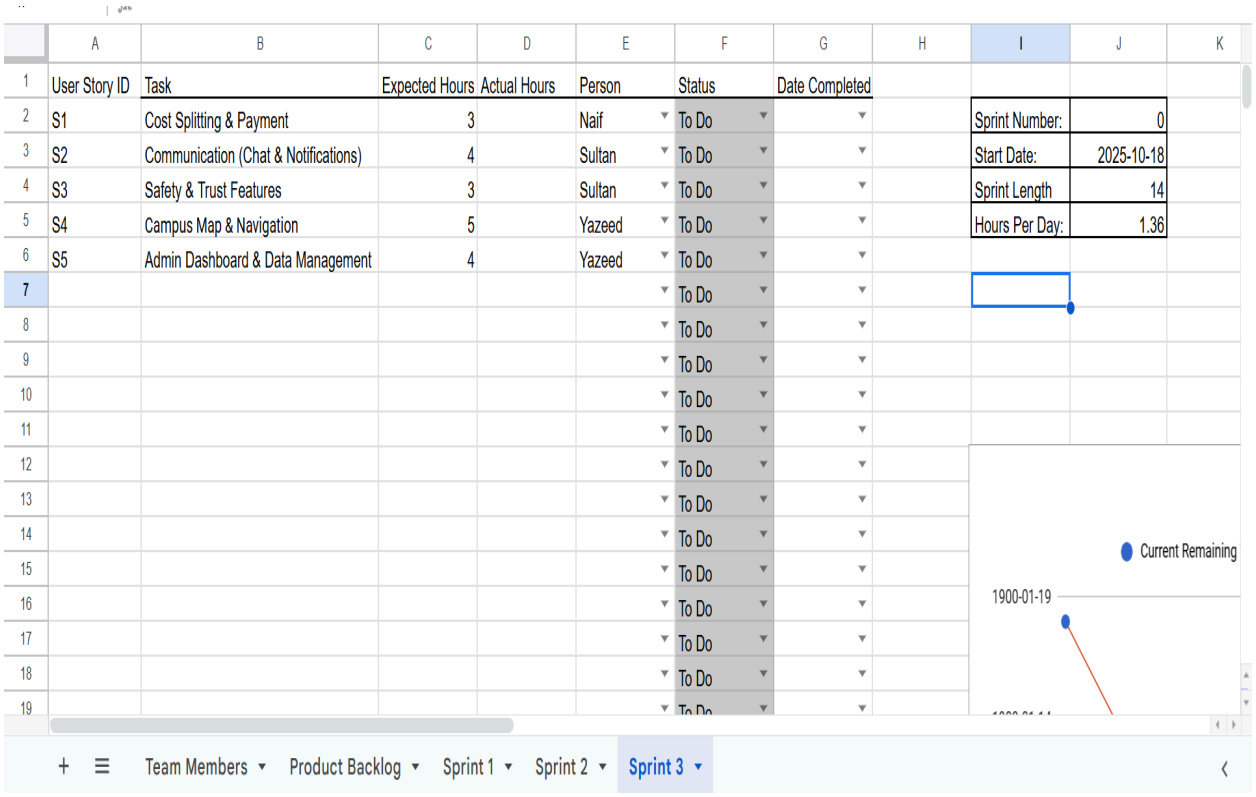


Figure 5 shows scrum sprint 3 sheet.

Phase 2

Architecture Design

This architecture was chosen to show how the student ride-sharing system works in a clear and simple way. The system is divided into four main modules: User Interface, Ride Matching, Database, and Notification. Each module has its own role, which makes the system easier to understand and manage. The user interface collects ride information from users, then the matching module processes this data and finds suitable matches. The database is used to store user and ride data, while the notification module informs users about the matching result. This design helps the system work efficiently and makes future updates easier.

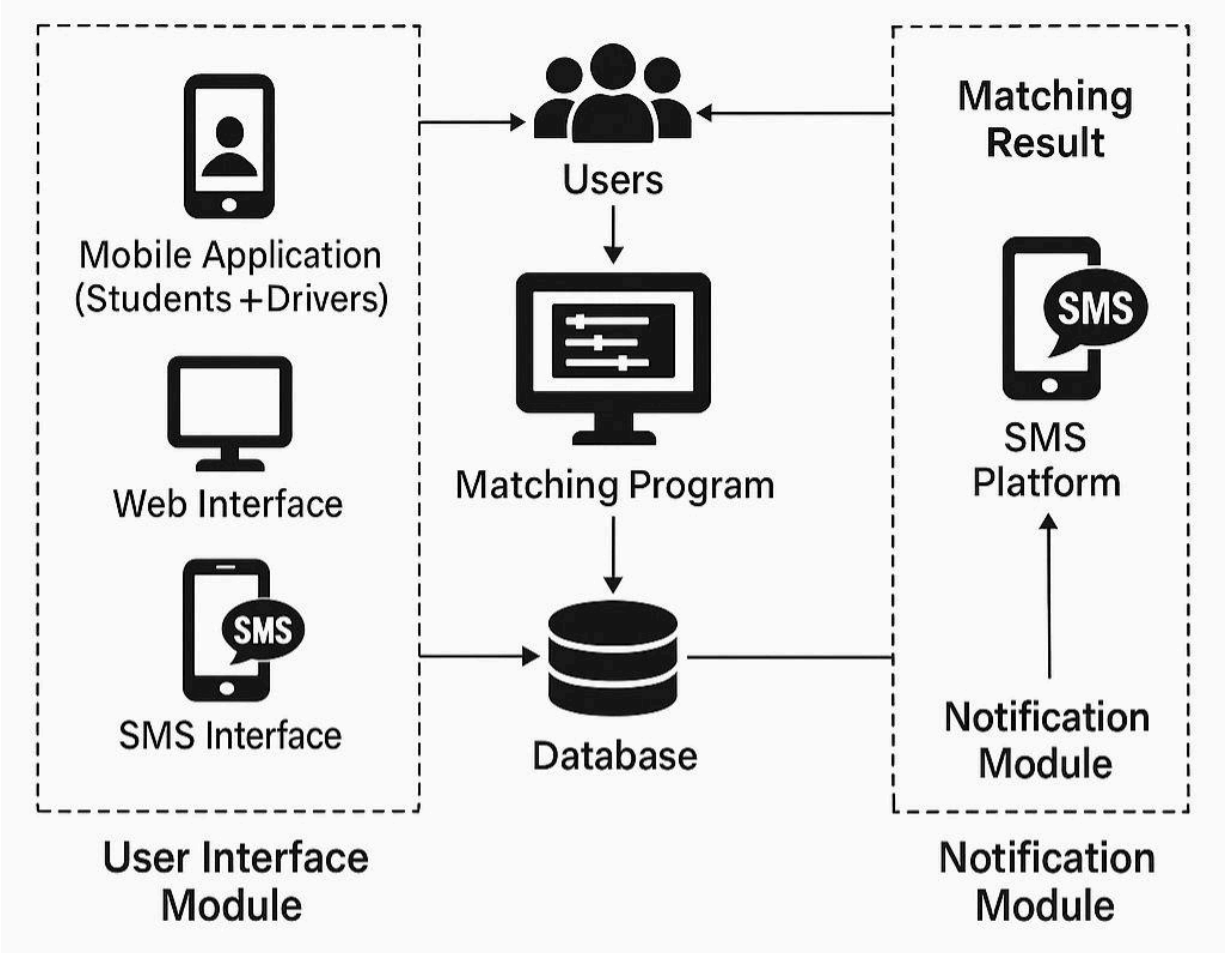


Figure 6 above shows Architecture Diagram for Student Ride-Sharing Application

Use Diagram

This use case diagram describes the main interactions that would be present within the Student Ride-Sharing Application and how a student will interact with the system. It identifies the main actors, starting with the student, who creates an account and logs into the app, before specializing into two roles: a passenger requesting a ride and the driver offering one. A passenger is allowed to view available rides, book a trip, make the intended payment, and provide a rating of their experience after the ride. For drivers, they can create new rides and update the status of their ride as necessary. Each use case described here represents a specific action to be performed by the user within the system. The result is that all essential functions involved in the system, such as booking, paying, and managing rides, are clearly stipulated. Generally, the diagram identifies how users will interact with the application, and the system will be organized, well-understood, and effective in facilitating transportation among students.

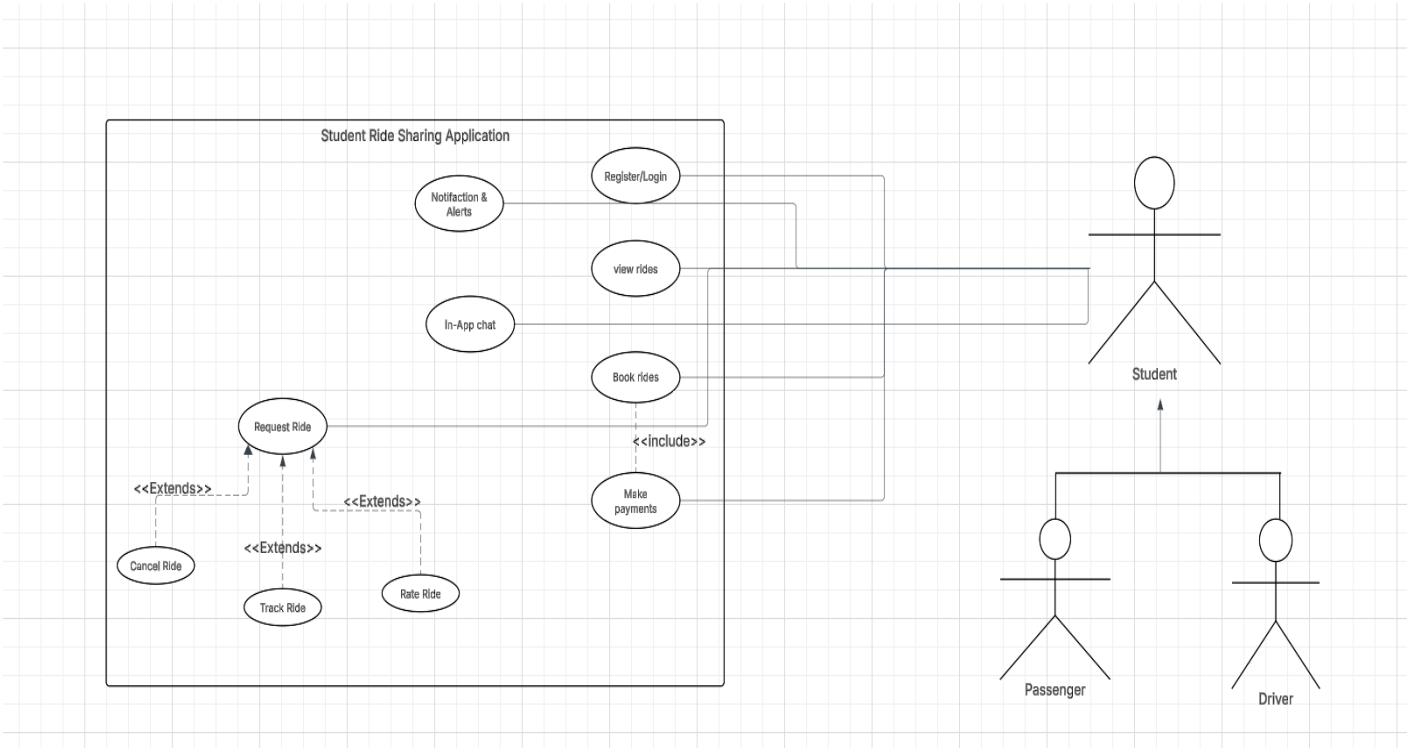


Figure 7 above shows Use Diagram for Student Ride-Sharing Application

Sequence Diagram

This sequence diagram illustrates how the Ride-Sharing Application works. A student first registers with their university email and selects whether they need an immediate ride or prefer to schedule one in advance. The app then matches them with nearby drivers or riders based on their location and preferences. Once the driver's details are verified, the student can choose to share the ride with trusted contacts. Payment is processed smoothly, and if there's ever an emergency, the student can quickly alert authorities and contacts for help. Once the ride is completed, the student has the opportunity to rate the driver and leave feedback. This helps improve the overall experience for future users, making sure the system stays safe and reliable. The application also gives the student easy access to their ride history, payment records, and account settings so they can manage everything in one place. Throughout the whole process, the app focuses on making things as safe and convenient as possible, offering features like emergency alerts, ride sharing with friends, and the flexibility to book rides whenever it suits the student.

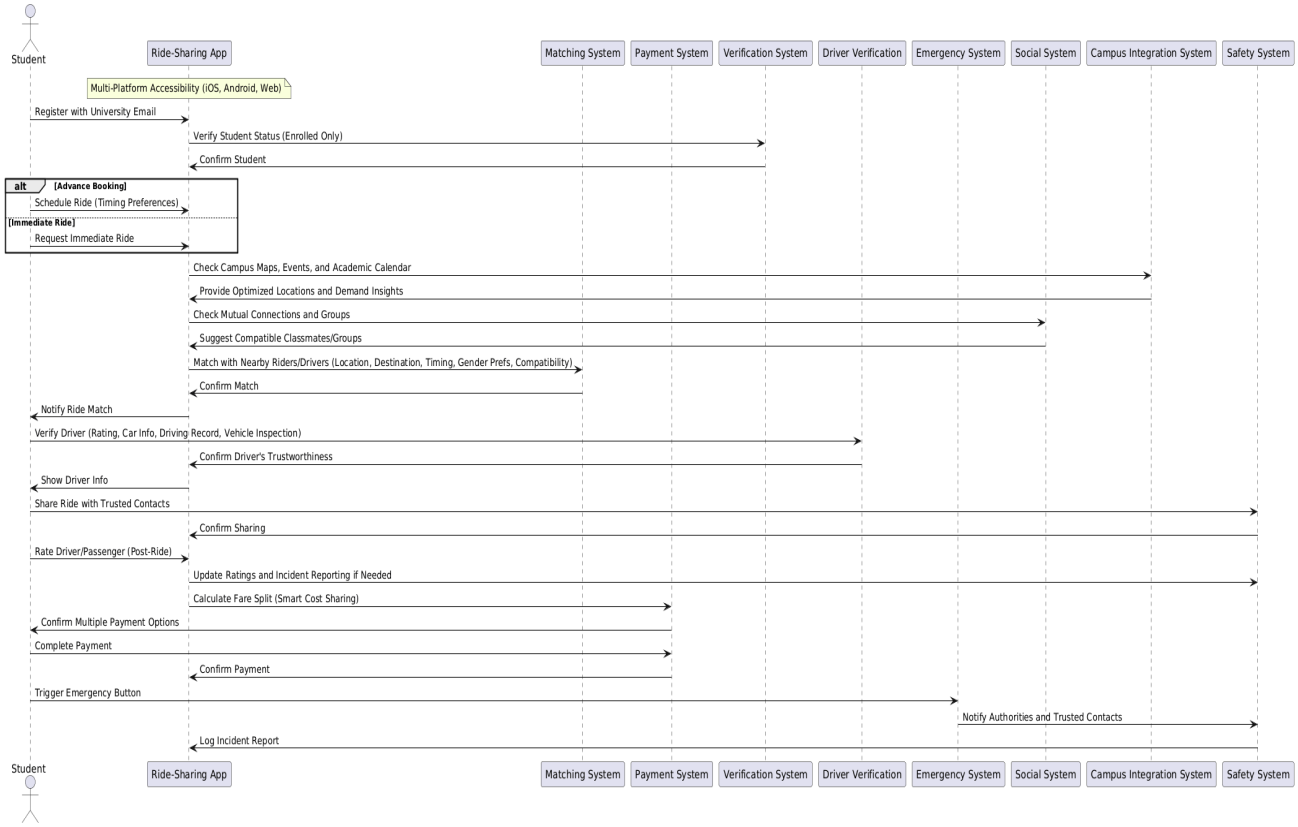


Figure 8 above shows Sequence Diagram for Student Ride-Sharing Application

Class Diagram

This class diagram shows the main elements of the Student Ride-Sharing App and their relationships. It includes users who will register and then log into the application, students who will book rides, and drivers who will offer transportation using their vehicles. Drivers will be able to create and manage rides, while bookings will connect students with available trips. Each booking will involve a payment to process the transaction and a rating to deliver feedback after the ride. Vehicles store the driver's car details, while rides organize trip information like origin, destination, and time. Each class has a distinctive role, making the system well-structured, easy to administer, and efficient in facilitating student transportation.

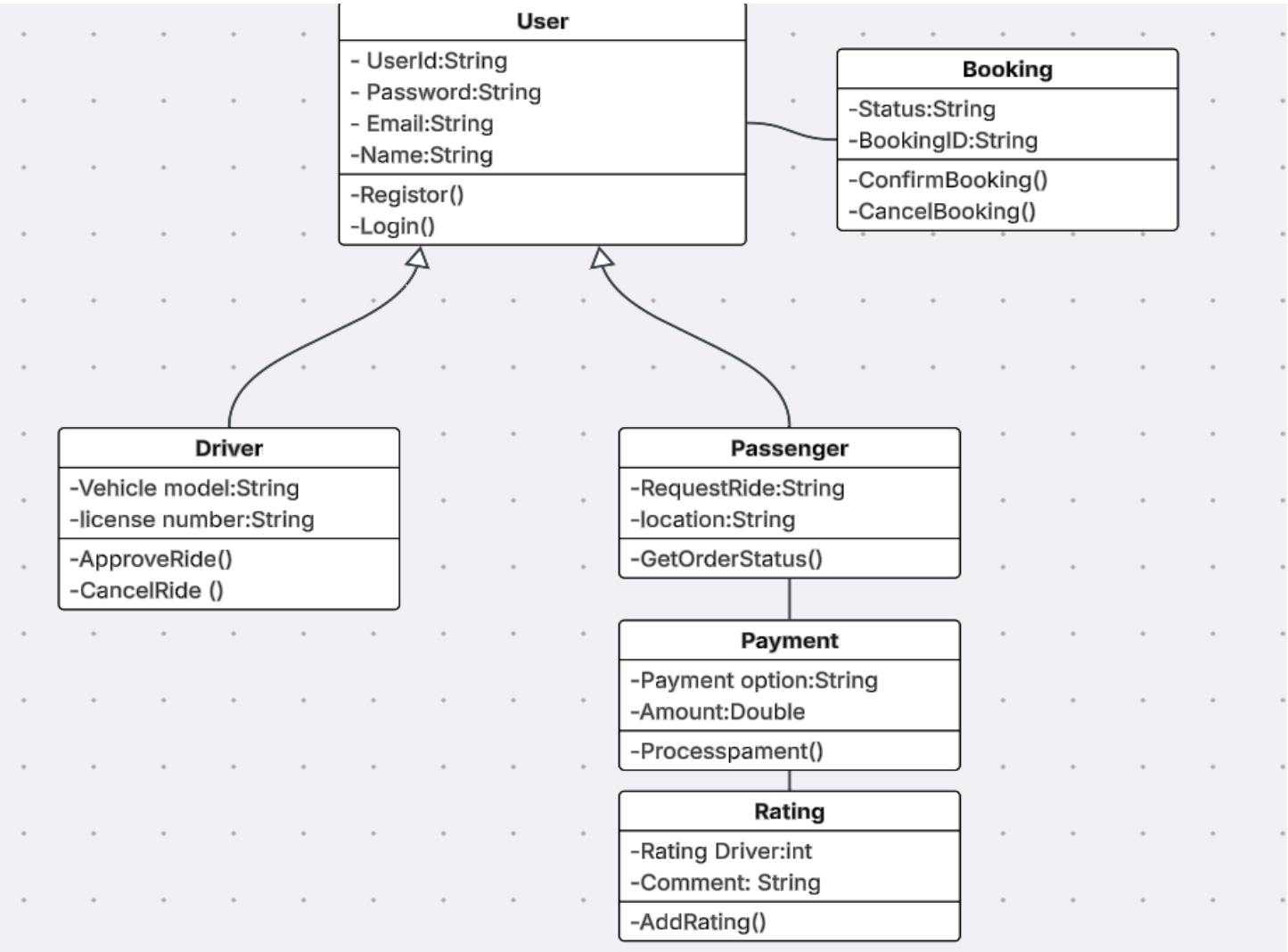


Figure 9 above shows Class Diagram for Student Ride-Sharing Application

Activity Diagram

This activity diagram shows the work-flow of a user in our Ride-Sharing Application in a university. The first activity after opening the application is logging into the account, or registering if the user is new. Then new users must validate that they are students, then the system verifies them. Now the user can search available rides to desired destinations, book it then confirm it. After that the system will handle both normal payment methods or cost-splitting if requested. Then passengers until a driver accepts. Finally after the ride they can show ride history to leave a feedback and contact support

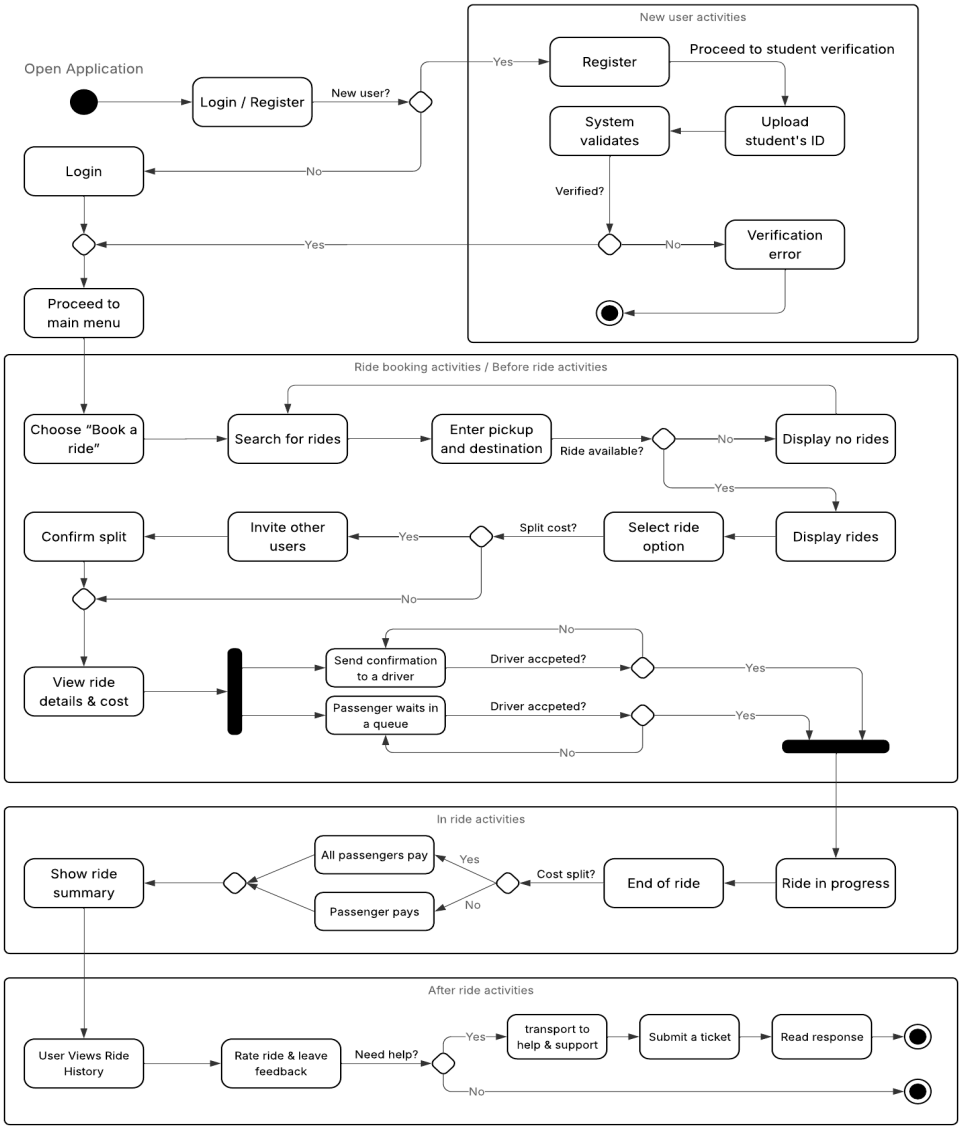
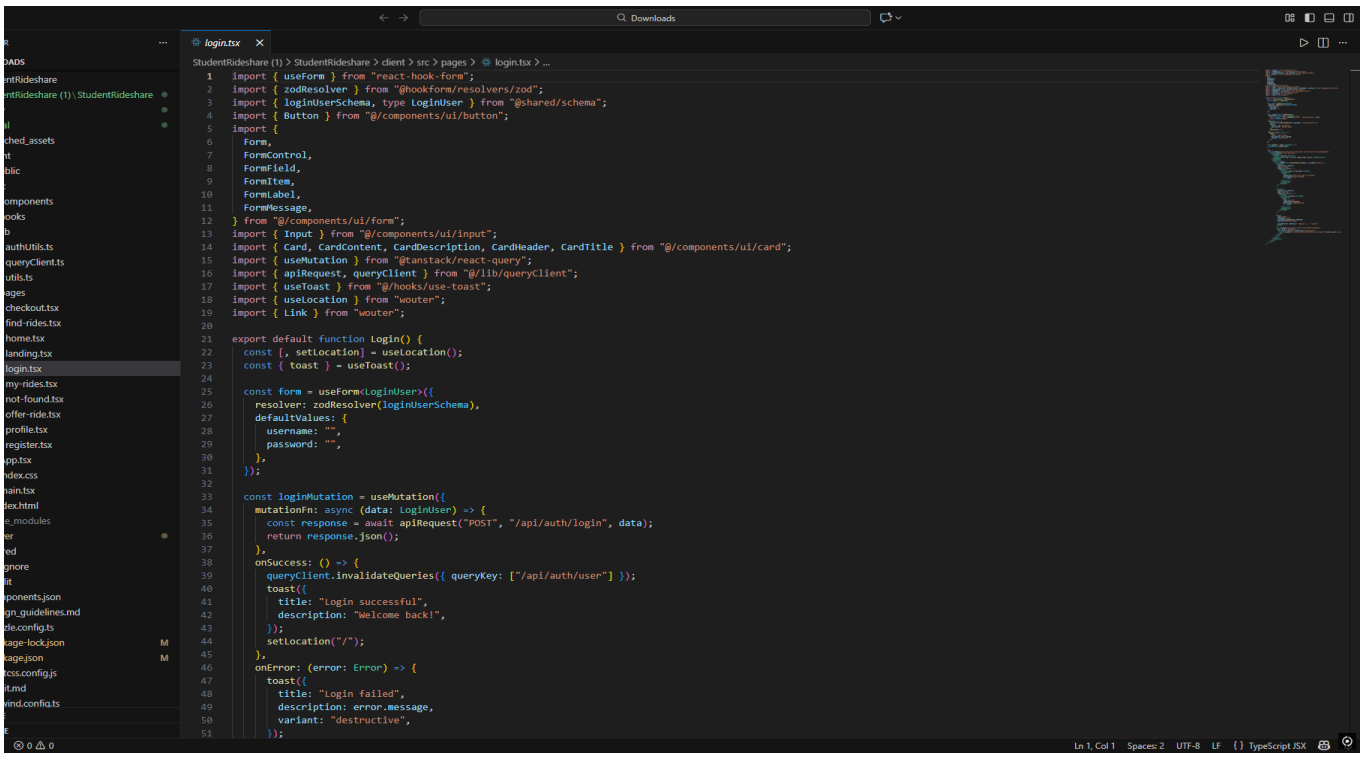


Figure 10 above shows Activity Diagram for Student Ride-Sharing Application

Login



```tsx
import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import { loginUserSchema, type LoginUser } from "@shared/schema";
import { Button } from "@/components/ui/button";
import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "@/components/ui/form";
import { Input } from "@/components/ui/input";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";
import { useMutation } from "@tanstack/react-query";
import { apiRequest, queryClient } from "@/lib/queryClient";
import { useToast } from "@/hooks/use-toast";
import { useLocation } from "wouter";
import { Link } from "wouter";

export default function Login() {
  const [, setLocation] = useLocation();
  const { toast } = useToast();

  const form = useForm<LoginUser>({
    resolver: zodResolver(loginUserSchema),
    defaultValues: {
      username: "",
      password: "",
    },
  });

  const loginMutation = useMutation({
    mutationFn: async (data: LoginUser) => {
      const response = await apiRequest("POST", "/api/auth/login", data);
      return response.json();
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/auth/user"] });
      toast({
        title: "Login successful",
        description: "Welcome back!",
      });
      setLocation("/");
    },
    onError: (error: Error) => {
      toast({
        title: "Login failed",
        description: error.message,
        variant: "destructive",
      });
```

Figure 11 above shows login code for Student Ride-Sharing Application

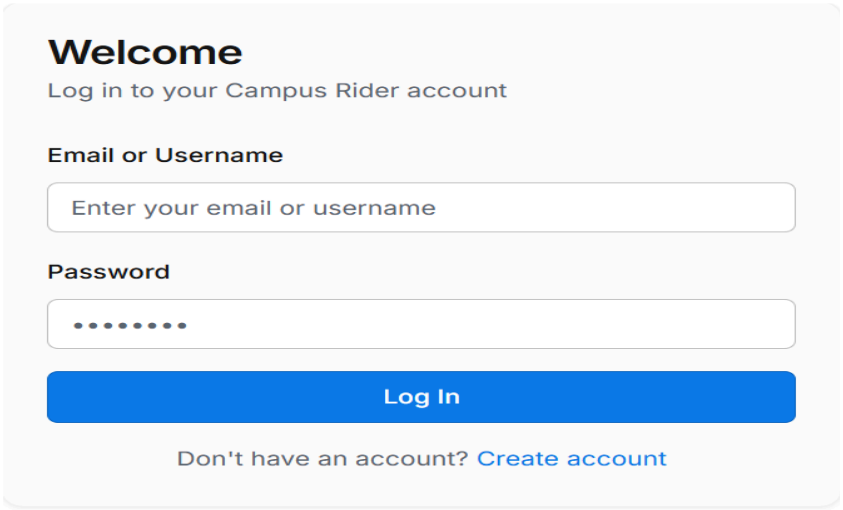
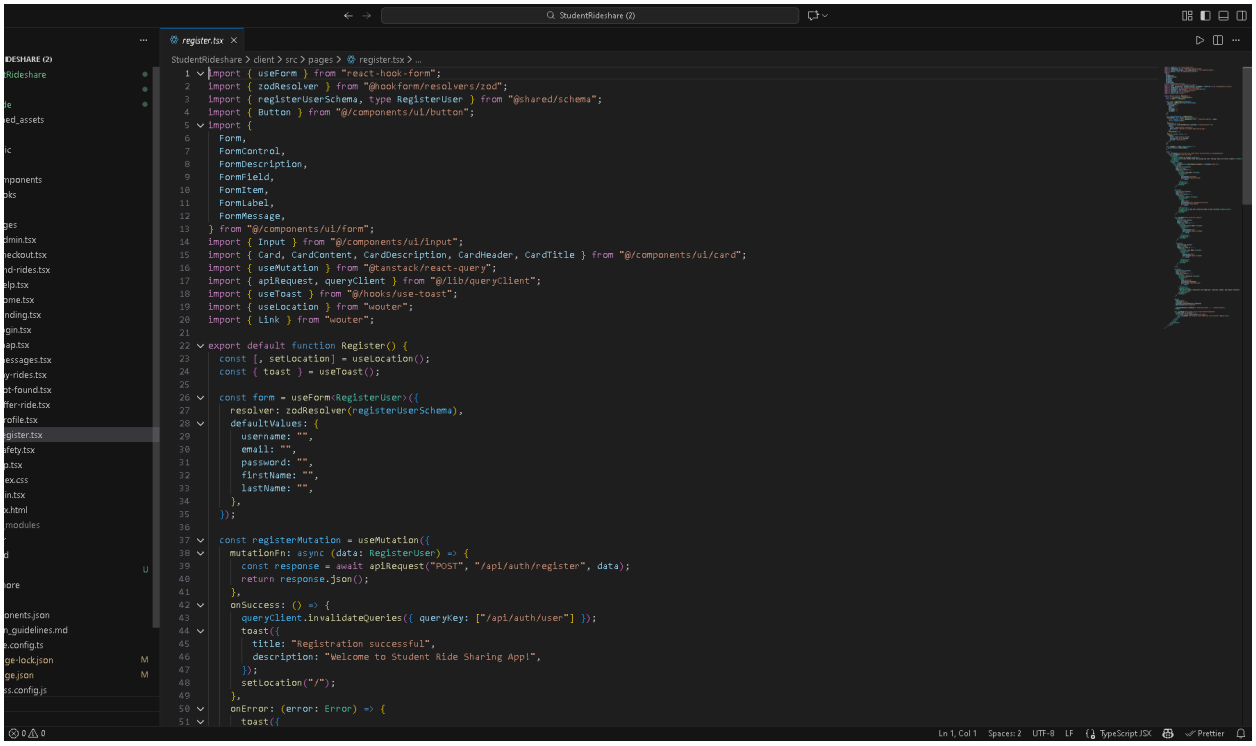


Figure 12 above shows login interface for Student Ride-Sharing Application

Sign in



```tsx
import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import { registerUserSchema, type RegisterUser } from "@shared/schema";
import { Button } from "@/components/ui/button";
import {
  Form,
  FormControl,
  FormDescription,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "@/components/ui/form";
import { Input } from "@/components/ui/input";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";
import { useMutation } from "@tanstack/react-query";
import { apiRequest, queryClient } from "@/lib/queryClient";
import { useToast } from "@/hooks/use-toast";
import { useLocation } from "wouter";
import { Link } from "wouter";

export default function Register() {
  const [, setLocation] = useLocation();
  const { toast } = useToast();

  const form = useForm<RegisterUser>({
    resolver: zodResolver(registerUserSchema),
    defaultValues: {
      username: "",
      email: "",
      password: "",
      firstName: "",
      lastName: "",
    },
  });

  const registerMutation = useMutation({
    mutationFn: async (data: RegisterUser) => {
      const response = await apiRequest("POST", "/api/auth/register", data);
      return response.json();
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/auth/user"] });
      toast({
        title: "Registration successful",
        description: "Welcome to Student Ride Sharing App!",
      });
      setLocation("/");
    },
    onError: (error: Error) => {
      toast({
```

Figure 13 above shows sign in code for Student Ride-Sharing Application

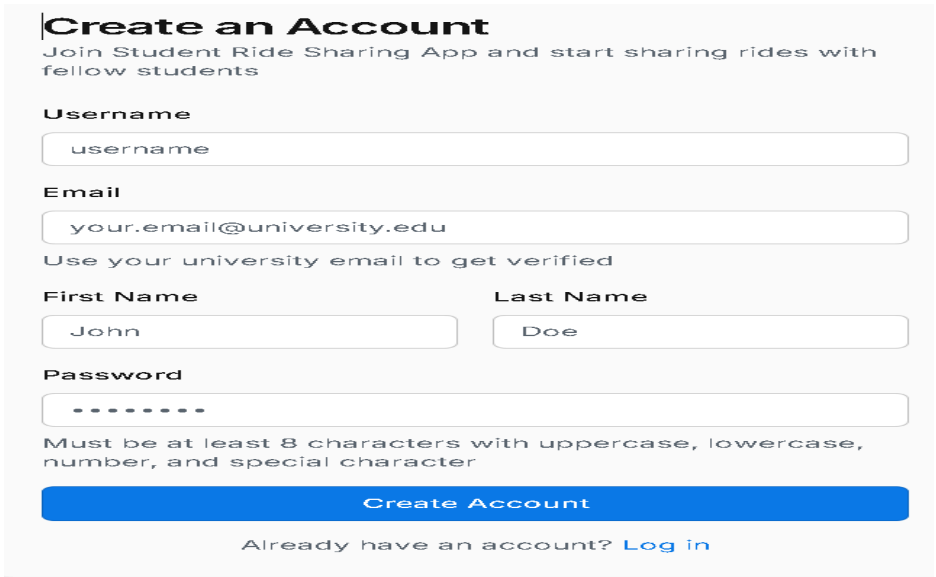
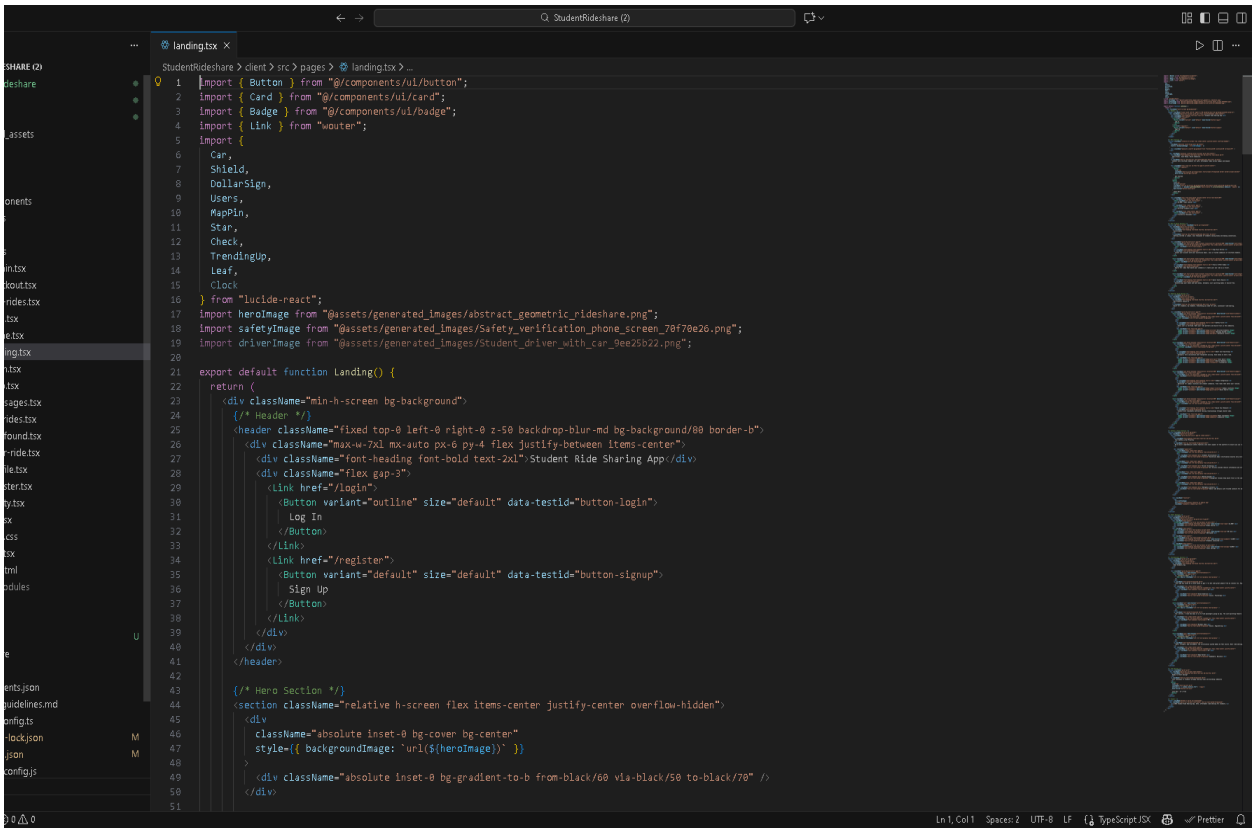


Figure 14 above shows sign in interface for Student Ride-Sharing Application

Homepage



```tsx
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Link } from "wouter";
import {
  Car,
  Shield,
  DollarSign,
  Users,
  MapPin,
  Star,
  Check,
  TrendingUp,
  Leaf,
  Clock
} from "lucide-react";
import heroImage from "@assets/generated_images/abstract_geometric_rideshare.png";
import safetyImage from "@assets/generated_images/Safety_verification_phone_screen_70f70e26.png";
import driverImage from "@assets/generated_images/Student_driver_with_car_9ee25b22.png";

export default function Landing() {
  return (
    <div className="min-h-screen bg-background">
      {/* Header */}
      <header className="fixed top-0 left-0 right-0 z-50 backdrop-blur-md bg-background/80 border-b">
        <div className="max-w-7xl mx-auto px-6 py-4 flex justify-between items-center">
          <div className="font-heading font-bold text-2xl">Student Ride Sharing App</div>
          <div className="flex gap-3">
            <Link href="/login">
              <Button variant="outline" size="default" data-testid="button-login">
                Log In
              </Button>
            </Link>
            <Link href="/register">
              <Button variant="default" size="default" data-testid="button-signup">
                Sign Up
              </Button>
            </Link>
          </div>
        </div>
      </header>

      {/* Hero Section */}
      <section className="relative h-screen flex items-center justify-center overflow-hidden">
        <div
          className="absolute inset-0 bg-cover bg-center"
          style={{ backgroundImage: `url(${heroImage})` }}
        >
          <div className="absolute inset-0 bg-gradient-to-b from-black/60 via-black/50 to-black/70" />
        </div>
```

Figure 15 above shows homepage code for Student Ride-Sharing Application

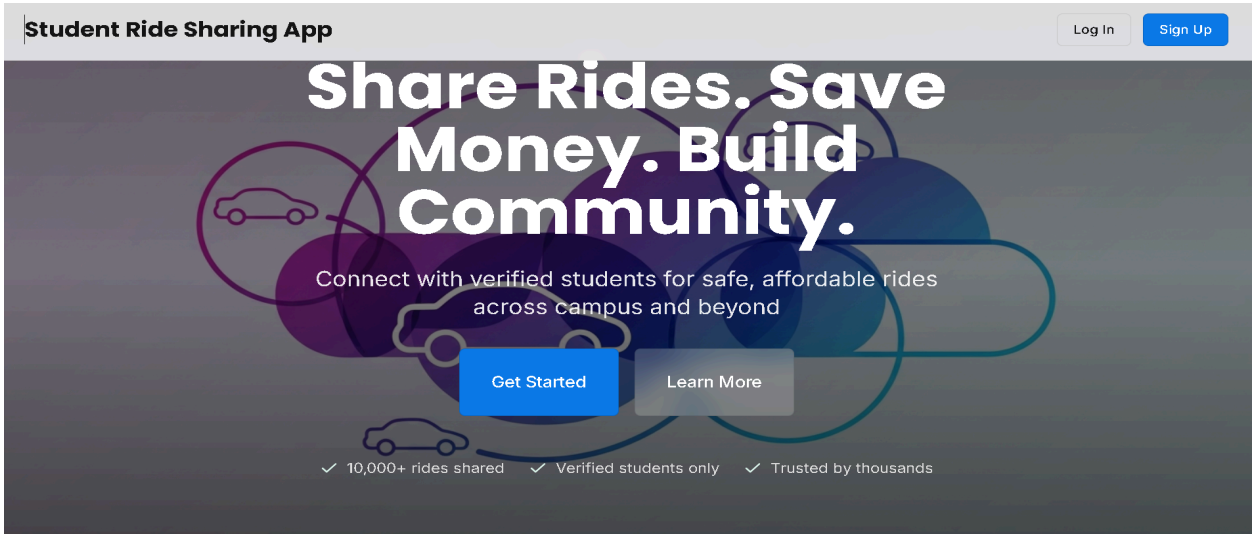


Figure 16 above shows homepage interface for Student Ride-Sharing Application

Phase 3

Test plan

This document outlines the test plan for the Student Ride-Sharing application, developed using React with TypeScript. The goal is to ensure that the system functions correctly and meets all requirements specified in the project scope.

1. Objectives

- Ensure the application is free from critical bugs and provides a decent user experience.
- Identify and document any defects found during testing.
- Validate the functionality of key features such as user authentication, route planning, map navigation, ride creation.

2. Scope

The test plan covers the following modules:

- Authentication (Login/Signup)
- Ride Creation & Management
- Map Navigation
- Ride Requesting
- Profile Management
- Admin Dashboard
- In-app Feedback

3. Testing Approach

- **Functional Testing:** Verify each feature against the expected behaviour.
- **Usability Testing:** Ensure ease of navigation through the application.
- **Performance Testing:** Measure response time and app stability.
- **Security Testing:** Secure user's info, log in data, and safe user's communication.
- **Integration Testing:** Validating interactions between system parts like maps, notifications, and backend.

4. Design Choices

• **Technology Used:** The app is developed using JavaScript, allowing flexible and efficient implementation of core features

● **User Interface:** Simple and clean design to make booking fast and easy .

● **Data Management:** Stores student profiles, ride history, ratings, and chat messages securely

● **Performance Considerations:** Optimized to work smoothly during peak hours such as morning classes or finals week.

● **Security Measures:** Protects personal data and payment information with strict privacy handling.

● **Scalability:** Modular structure allows adding new features easily (group rides, extra payment methods, etc.)

5. Strategies for Ongoing Maintenance and Improvement

- **Regular Updates:** We'll keep the app fresh with new features, bug fixes, and security updates.
- **Listening to Users:** We'll act on your feedback to improve the app.
- **Performance Monitoring:** We'll monitor and fix any issues quickly.
- **Security Enhancements:** We'll ensure your data stays safe with up-to-date security.
- **Continuous Improvement:** The app will keep improving with new features.
- **Quick Fixes:** Problems will be fixed fast to minimize disruption.
- **Staying Flexible:** We'll adapt the app as technology and needs evolve.

Test cases

1: Login test case IDs

Login_001 Login_002

	A	B	C	D	E
1	Test case:	Login		Test case description	Testing the log in functionality
2					
3	Tester name:	Ahmad		Date tested	12/6/2025
4				Test case(Pass/Fail)	Pass
5					
6	S#	Precondition			
7	1	The user has entered the application.			
8	2	The user has registered with valid inputs(email and password).			
9	3	The application is accessible and the login screen is functional.			
10					
11	Test ID	Test case	Data entered by the tester	Expected result	Actual result
12	Login_001	Successful login	email: whitbeardznb@gmail.com password: 0554543322zz5	Login success	Pass
13	Login_002	Unsuccessful login: invalid inputs	email: whitbeardznb@gmail.com password: 1342	Login unsuccessful	Pass

2: Ride request test case IDs

RR_001 RR_002

	A	B	C	D	E	F	G	H
1	Test case:	Ride Request		Test case description	Testing the ability to request a ride with valid and invalid inpu			
2								
3	Tester name:	Ahmad		Date tested	16/6/2025			
4				Test case(Pass/Fail)	Pass			
5								
6	S#	Precondition						
7	1	User is logged into the Student Ride-Sharing App.						
8	2	Find a ride page is accessible from dashboard.						
9	3	Ride component loads correctly.						
10	4	Choose a ride and Book it.						
11								
12	Test ID	Test case	Data entered by tester	Expected result	Actual result			
13	RR_001	Successful Ride Request	Pickup: main campus Destination: sport complex Time: 10:30 AM	Ride request created with status	Pass			
14	RR_002	Unsuccessful Request (Missing fie	Pickup: EMPTY Destination: sport complex Time: 11:00 AM	System shows: 'Pickup location is	Pass			

3: Profile page management test case IDs

PM_01 PM_02 PM_03 PM_04 PM_05 PM_06

	A	B	C	D	E
1	Test case:	Profile page management		Test case description	Managing the profile page
2					
3	Tester name:	Turki		Date tested	6/12/2025
4				Test case(Pass/Fail)	Pass
5					
6	S #	Precondition			
7	1	The user has logged into the system			
8	2	The user has entered the profile page			
9					
10					
11	Test ID	Test case	Data entered by the tester	Expected result	Actual result
12	PM_01	Log out	None	User logs out	Pass
13	PM_02	View rides taken and user's rating	None	User sees rides and ratings	Pass
14	PM_03	View and edit user's information	Name, phone number, universtiy email	User sees data and can edit it	Pass
15	PM_04	View and edit user's emergency contact	Emergency contact name and phone	User sees data and can edit it	Pass
16	PM_05	User can become a driver	Vehicle make, model, year, color, plate info	User becomes a driver	Pass
17	PM_06	View and edit driver's information	Vehicle make, model, year, color, plate info	User sees data and can edit it	Pass

4: Admin dashboard test case IDs

AD_01 AD_02 AD_03 AD_04 AD_05

	A	B	C	D	E
1	Test case:	Admin Dashboard		Test case description	Testing admin dashboard modules inclu
2					
3	Tester name:	yazeed		Date tested	12/5/2025
4				Test case(Pass/Fail)	Pass
5					
6	S #	Precondition			
7	1	Admin user is logged in.			
8	2	Dashboard backend API is reachable.			
9	3	System contains users, alerts, and tickets.			
10					
11	Test ID	Test case	Data entered by tester	Expected result	Actual result
12	AD_001	Load Dashboard Overview	Open Admin Dashboard	Dashboard shows: Total Users, Verified	Pass
13	AD_002	View Users Tab	Click Users tab	Users list loads with names, emails, rol	Pass
14	AD_003	View Alerts	Click Alerts tab	Alerts list loads, at least 1 active alert d	Pass
15	AD_004	View Tickets	Navigate to Tickets tab	Open tickets displayed, count matches	Pass
16	AD_005	Verify User Info	Open Users tab, inspect a verified user	Verified badge shown, Driver/Student r	Pass

5: Map test case IDs

Map_01 Map_02 Map_03

Test case:	Map test		**Test case description**	Interacting with a map
Tester name:	Sultan		**Date tested**	07/12/25
			Test case(Pass/fail)	pass
S #	**Precondition**			
	1 The user has loged into the system			
	2 The user has entered the map page			
Test ID	**Test case**	**Data entered by the tester**	**Expected result**	**Actual result**
Map_01	Detecting the location of the college on the map	User taps the location-detection button on the map	The user can detect the location successfully	Pass
Map_02	Pressing on wanted college	User selects the college marker on the map	The user should be able to press on wanted college	Pass
Map_03	Favorite place	User presses the favorite icon to add the college to the favorites list	The user is able to add a college to the favorite places list	Pass

6: In-app feedback test case IDs

FB_01 FB_02 FB_03

Test case:	In-app feedback		Test case description	submit feedback in ride sharing app	
Tester name:	Naif		Date tested	12/6/2025	
			Test case(Pass/Fail)	pass	
S #	Precondition				
	1 ride sharing app is entred				
	2 the used is logged in				
	3 the device has an active internet connection				
Test ID	Test case	Data entered by the tester	Expected result		Actual result
FB_01	Write feedback	Feeedback message	The user should be able to write and send feedback		Pass
FB_02	Feedback functionality	None	The feedback should load without errors		Pass
FB_03	Verfying feedback	None	Upon clicking submit, a message should appear to confirm it		Pass

Scrum Sheets

Scrum Product Backlog Sheet

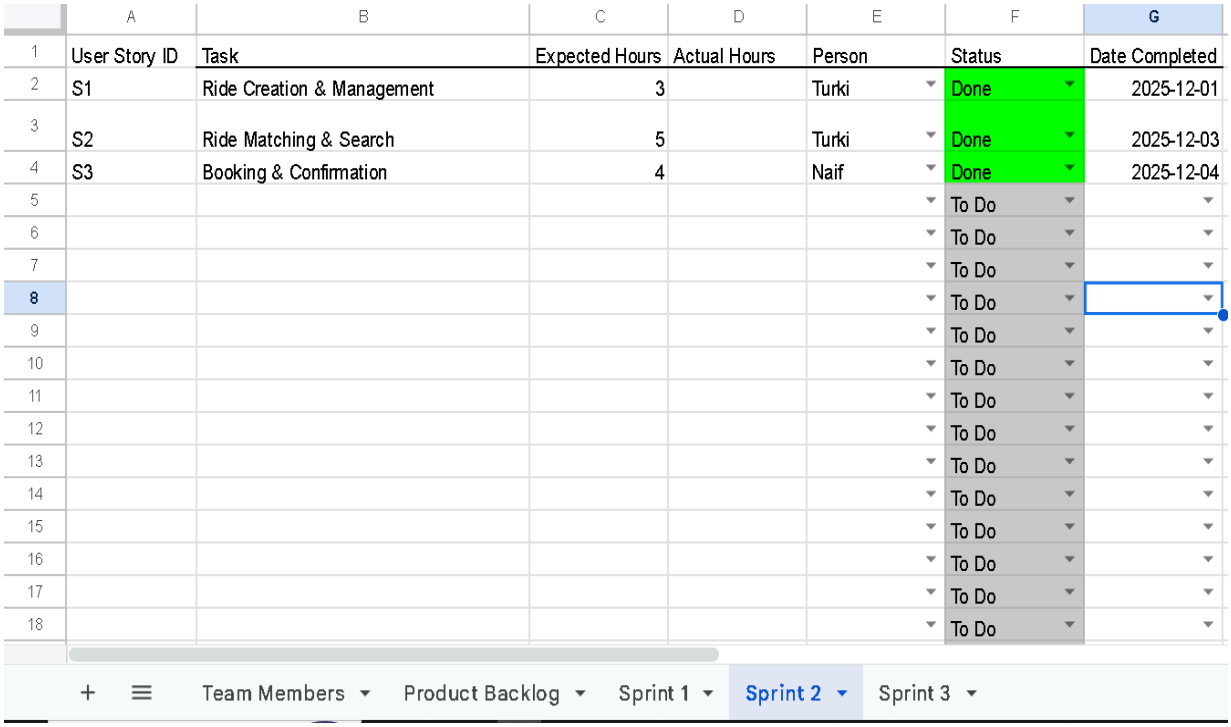
	A	B	C	D	E	F	G
1	Sprint Number	User Story ID	User Story Title	Brief Description	Additional Information	Priority	Status
2	1	S1	User Registration & Authentic	As a student, I want to		1-high	Done
3	1	S2	Student Verification Module	As an admin, I want to verify student identities and enrollment so that only legitimate		1-high	Done
4	2	S3	Ride Creation & Management	As a driver, I want to create,		1-high	Done
5	2	S4	Ride Matching & Search	As a passenger, I want to search and match with available rides based on my locatio		1-high	Done
6	2	S5	Booking & Confirmation	As a passenger, I want to book a seat and receive confirmation details so that I can b		2-medium	Done
7	1	S6	Profile Management	As a student, I want to edit my profile information and upload a photo so that other us		1-high	Done
8	3	S7	Payment	As a passenger, I want to automatically split the cost and pay securely so that I can		1-high	Done
9	3	S8	Communication (Chat & Notif	As a student (driver or		2-medium	Done
10	3	S9	Safety & Trust Features	As a student, I want to have safety options such as emergency alerts, ratings, and tr		2-medium	Done
11	3	S10	Campus Map & Navigation	As a student, I want to use an in-app campus map and navigation system so that I c		2-medium	Done
12	3	S11	Admin Dashboard & Data Ma	As an admin, I want to		2-medium	Done
13	2	S12	Ride History & Reviews	As a passenger or driver, I want to view past rides and leave reviews so that I can eva		3-low	Done
14	3	S13	Help & Support Center	As a user, I want to access FAQs and contact support so that I can resolve problems		3-low	Done

Scrum Sprint 1 sheet

	A	B	C	D	E	F	G	H
1	User Story ID	Task	Expected Hours	Actual Hours	Person	Status	Date Completed	
2	S1	User Registration & Authentication	3		Ahmad	Done	2025-11-20	
3	S2	Student Verification Module	5		Ahmad	Done	2025-11-20	
4						To Do		
5						To Do		
6						To Do		
7						To Do		
8						To Do		
9						To Do		
10						To Do		
11						To Do		
12						To Do		
13						To Do		
14						To Do		
15						To Do		
16						To Do		
17						To Do		
18						To Do		

Team Members | Product Backlog | Sprint 1 | Sprint 2 | Sprint 3

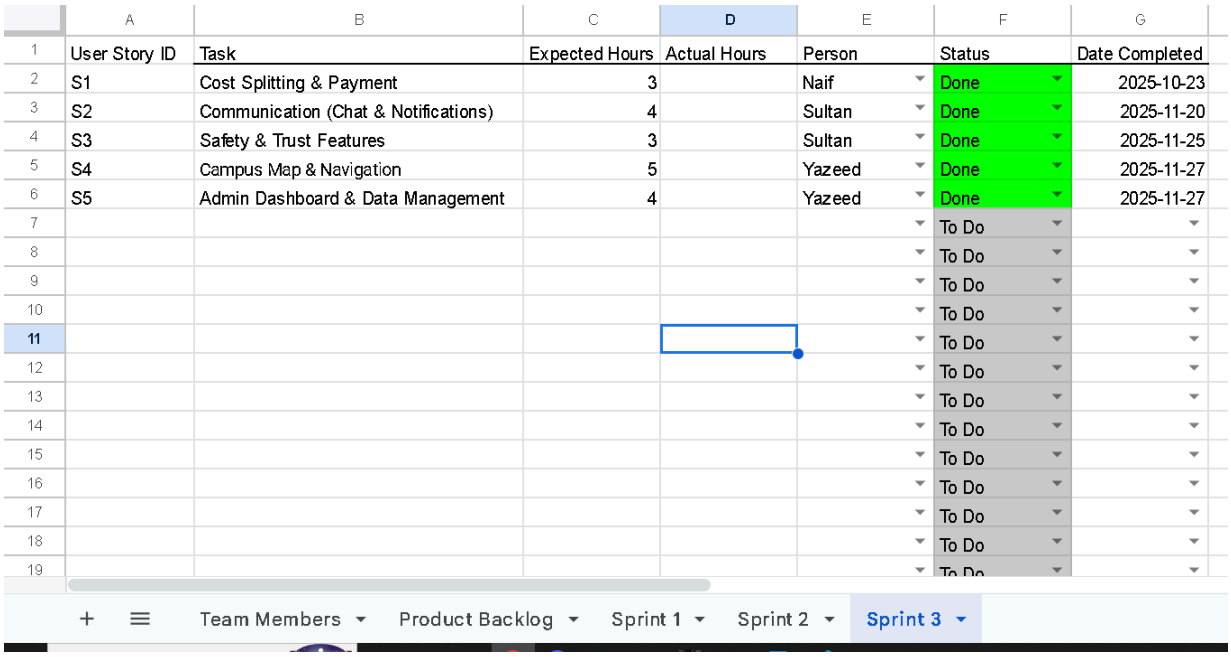
Scrum Sprint 2 Sheet



	A	B	C	D	E	F	G
1	User Story ID	Task	Expected Hours	Actual Hours	Person	Status	Date Completed
2	S1	Ride Creation & Management	3		Turki	Done	2025-12-01
3	S2	Ride Matching & Search	5		Turki	Done	2025-12-03
4	S3	Booking & Confirmation	4		Naif	Done	2025-12-04
5						To Do	
6						To Do	
7						To Do	
8						To Do	
9						To Do	
10						To Do	
11						To Do	
12						To Do	
13						To Do	
14						To Do	
15						To Do	
16						To Do	
17						To Do	
18						To Do	

Team Members | Product Backlog | Sprint 1 | Sprint 2 | Sprint 3

Scrum Sprint 3 Sheet



	A	B	C	D	E	F	G
1	User Story ID	Task	Expected Hours	Actual Hours	Person	Status	Date Completed
2	S1	Cost Splitting & Payment	3		Naif	Done	2025-10-23
3	S2	Communication (Chat & Notifications)	4		Sultan	Done	2025-11-20
4	S3	Safety & Trust Features	3		Sultan	Done	2025-11-25
5	S4	Campus Map & Navigation	5		Yazeed	Done	2025-11-27
6	S5	Admin Dashboard & Data Management	4		Yazeed	Done	2025-11-27
7						To Do	
8						To Do	
9						To Do	
10						To Do	
11						To Do	
12						To Do	
13						To Do	
14						To Do	
15						To Do	
16						To Do	
17						To Do	
18						To Do	
19						To Do	

Team Members | Product Backlog | Sprint 1 | Sprint 2 | Sprint 3

References :

1. https://www.scribbr.com/research-paper date on (08 / 11 / 2025)
2. https://www.techtarget.com/ date on (08 / 11 / 2025)
3. https://www.insivia.com/determining-target-audience date on (08 / 11 / 2025)
4. https://productive.io/blog/project-objectives date on (08 / 11 / 2025)
5. https://plaky.com/blog/project-management-software-features date on (08 / 11 / 2025)
6. https://www.perforce.com/ date on (08 / 11 / 2025)
7. https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf date on (08 / 11 / 2025)
8. https://docs.google.com/spreadsheets/d/1nhoEwo579oGzteZDoUxOCrs1y6B7scxn/edit?gid=1727305754#gid=1727305754 date on (08 / 11 / 2025)