UNITED INTERNATIONAL UNIVERSITY

Computer Science and Engineering

SRS Report

Course Code: CSE 3411

Section: G

Participants:

Sayed Hasan Sami - 0112230939

Sheikh Sadi - 0112230646

Badhon Dalbot - 0112230660

Syed Rafidul Alam- 0112230319

Sunjana Binte Shameem - 011191246

1. Introduction

1.1. Introduction

This section briefly introduces the University Ride Sharing Management System. It explains that the system is designed to provide a platform for university students and staff to share rides in a secure, cost-effective, and efficient manner. The system connects riders and drivers within the university community, ensuring safety and reducing transportation costs and environmental impact.

1.2. Problem Statement

This system addresses several key problems:

- Lack of organized ride-sharing options among students and faculty.
- High transportation costs for individuals.
- Security concerns with using external ride-sharing platforms.
- Difficulty in finding rides at odd hours or to/from campus.

1.3. Motivation

- Reducing carbon emissions and traffic congestion.
- Encouraging a sustainable and collaborative campus culture.
- Leveraging technology to solve everyday campus commuting problems.
- Offering an alternative to services like Uber, Pathao, or inDrive, tailored for university use.

1.4. Overview/Goal

The goal is to develop a system that:

- Allows users to register as drivers or passengers.
- Matches ride requests with available drivers within the university.
- Ensures safety through verified university accounts.
- Tracks rides and maintains a feedback system for trustworthiness.

2. Information Gathering

2.1. Introduction

This subsection details the method which gathered complete data needed to build CampusRide as a university-exclusive ride-sharing system that matches students and employees for regular commuting. A comprehensive methodology was adopted which incorporated surveys for probable ridership and interviews with student advisors and faculty members and university officials to address the pressing requirement for economical and secure community-centric transportation in university grounds. Our research consisted of studying policies related to campus transportation at the university while analyzing the existing ride-sharing services available through a competitive analysis. The acquired information serves as a foundation to define precise functional and non-functional requirements for a user-centered solution which will include payment integration as well as route planning and safety features and verification processes.

2.2. Source of Information

The development of specifications and design for CampusRide relied on evaluation of academic work alongside user research and competitive evaluation. A list of sources includes:

2.2.1. Evaluation of Competition for Ride-Sharing Platforms:

Through extensive research, the current UI/UX features and feature sets of multiple ride-sharing systems operating in their respective comparable markets were studied. The study focused on analyzing three specific services.

- **Uber:** Uber received an evaluation because of its established brand, combined with advanced interface design and considerable service offering.
- **Pathao:** A comprehensive examination of Pathao included its handling of transportation connections and payment options as well as its position in the local market.
- **Shohoz:** The analysis of Shohoz intends to understand its focus on distant travel within Bangladesh while evaluating its connection to university commuting.
- **Obhai:** Obhai received scrutiny because of its unique aspects that address ride-sharing within communities and driver authentication systems.
- **InDrive:** Examined for user customization choices and its dynamic pricing structure.

This research aimed to discover best practices together with marketstanding chances while identifying potential risks, which would help maintain user-friendly features for CampusRide. All recorded data during this process consisted of screenshots along with extensive feature evaluations.

2.2.2. Research on Users- Survey:

For accurate data about university ride-sharing platform interests and commuter behavior, and essential user needs and expectations, the surveys targeted representatives from among staff and students. The survey addressed subjects like:

- Current modes of transportation and related expenses.
- A ride-sharing service must meet certain requirements, which include safety measures, multiple payment options, together with route planning functionality.
- University students who confirmed enrollment at the institution can access my offer to share rides.
- Staff and students at universities have both advantages and uncertainty regarding ride-sharing options.

2.2.3. Research Paper Analysis:

The author analyzed academic articles which focused on ride-sharing services in urban environments as their supporting research material. The papers that were analyzed:

- **"A Ride Sharing Service for University Community"**: This research addressed both positive and negative aspects of creating a transportation-sharing solution which addressed logistical barriers and student engagement.
- **"A Study on Motor Ride Sharing Service in Dhaka City, Present and Future Challenges":** This paper presented valuable knowledge regarding motor ride-sharing services in developing city like Dhaka through its discussion of safety and regulatory aspects and future outlook.

Our team gained knowledge about existing models and relevant laws, and distinct challenges for implementing CampusRide within the local community.

2.3. Information Gathering Tools

The development of CampusRide received data collection along with evaluation through various measurement and evaluation procedures. These techniques formed an effective process which allowed organizations to understand market trends and customer demands effectively.

2.3.1. Toolset for Competitive Analysis:

- **Snagit** and similar screen-capture solutions enable researchers to obtain detailed, high-resolution screenshots of competitive platforms' interfaces using annotation tools for complete user interface examination. Important features and design components require annotation along with developmental opportunities in order to gain attention.

2.3.2. Tools for User Research:

- We used an online survey platform such as **Google Forms** for distributing and creating the user survey to personnel and students. Users needed access to data collection as well as data analysis and reporting features through the platform.
- The survey data undergoes analysis through **Excel** software which identifies patterns while categorizing responses based on commuter behavior and demographic traits.

2.3.3. Resources for Academic Research:

- Academic research papers on ride-sharing services and urban transportation are located through online access to databases such as **ResearchGate**.

2.3.4. All-purpose Tools:

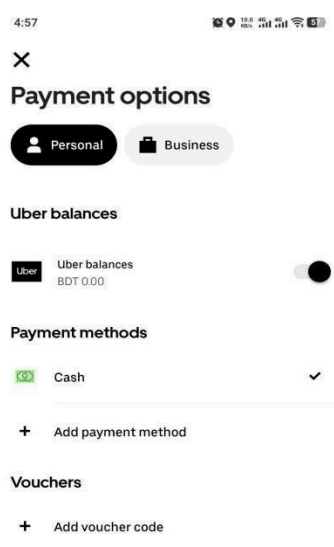
- During the whole information-gathering period the study team used **Google Workspace** as a collaboration tool to maintain communication and group work.
- The document management system consisting of **Google Drive** along with SharePoint enables researchers to store, arrange and distribute their research materials, screenshots and gathered data.

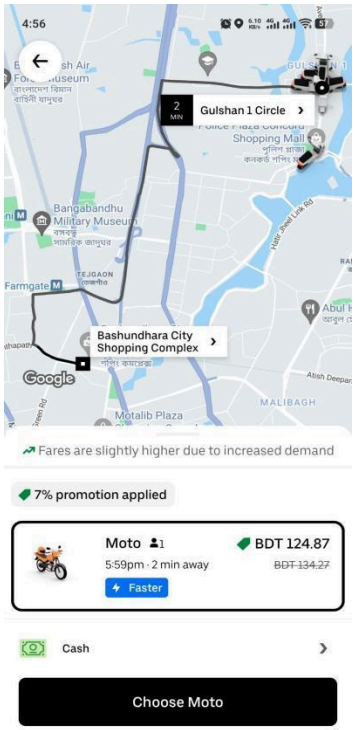
2.4. Detailed Information Gathering

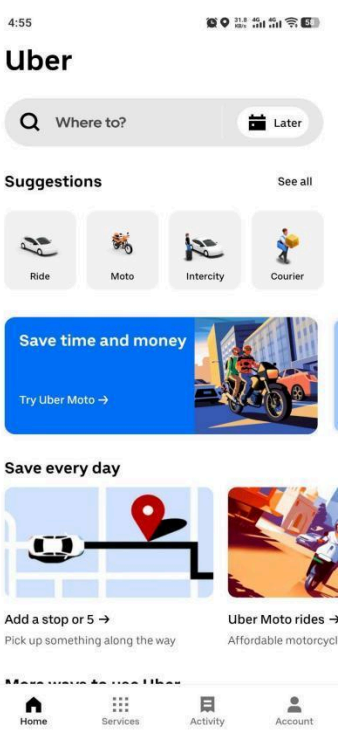
2.4.1. Competitor analysis:

Uber:

- **Ride Matching & Booking:** Real-time ride matching with nearby drivers.
- **Ride Scheduling & Recurring Rides:** Allows scheduling rides in advance.
- **Fare Estimation & Payment Integration:** Upfront fare estimation and seamless **payment integration.**
- **Driver Rating & Feedback System:** Two-way rating system for drivers and riders.
- **Customer Support:** 24/7 in-app support and help center.
- **Driver Features:** Earnings tracking, payout system, and ride history.
- **Safety Measures:** Real-time tracking, SOS button, and driver/rider verification.
- **Admin Features:** Manages driver onboarding, payments, and customer complaints.

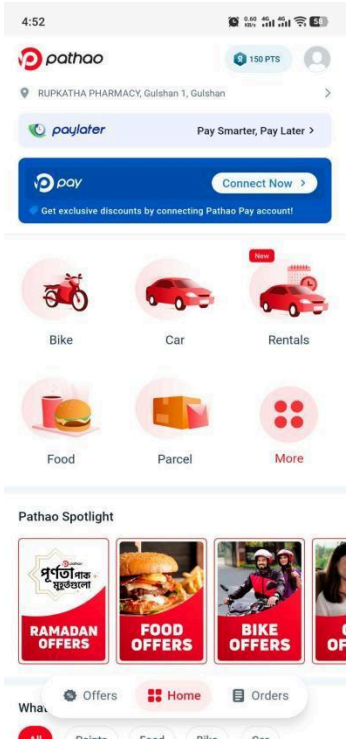


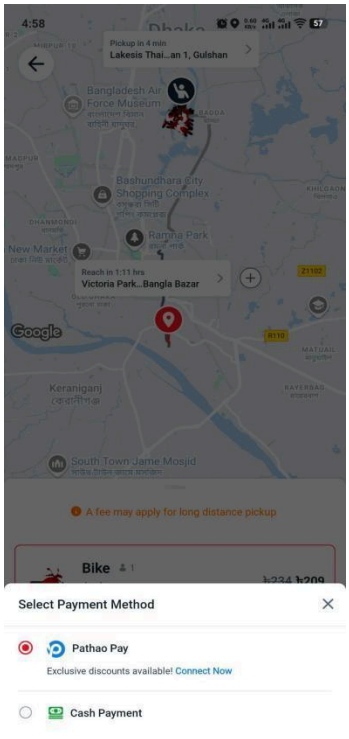


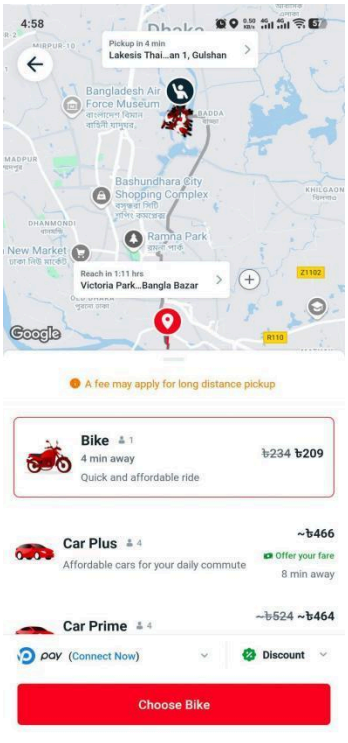


Pathao:

- **Ride Matching & Booking:** Real-time ride matching for bikes and cars.
- **Ride Scheduling & Recurring Rides:** Limited scheduling options.
- **Fare Estimation & Payment Integration:** Fare estimation and cashless payments.
- **Driver Rating & Feedback System:** Two-way rating system.
- **Customer Support:** In-app chat support.
- **Driver Features:** Earnings tracking and payout system.
- **Safety Measures:** Basic rider/driver verification.
- **Admin Features:** Manages driver onboarding and payments.

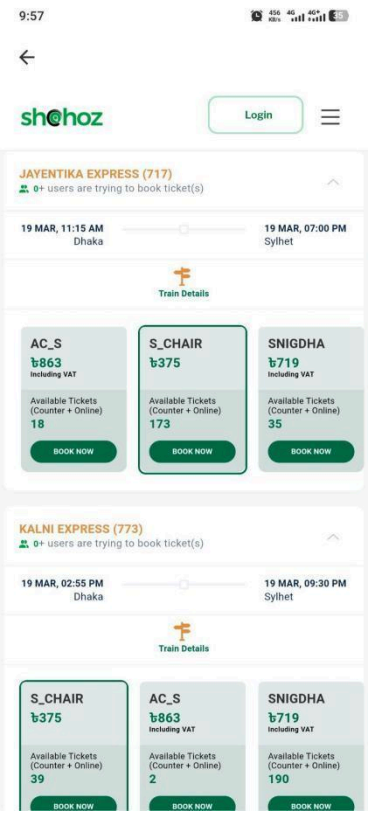


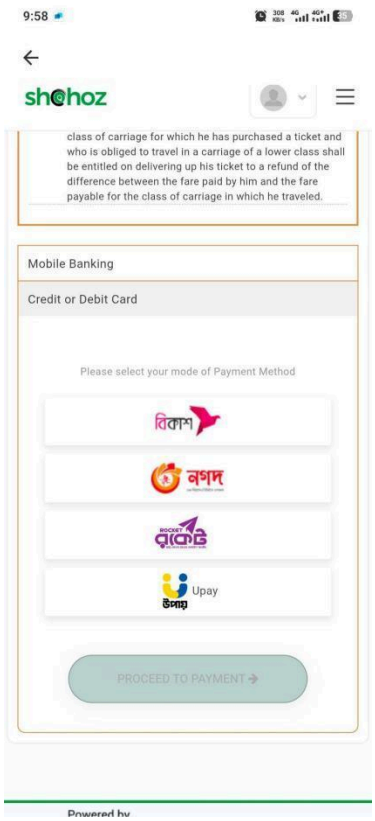


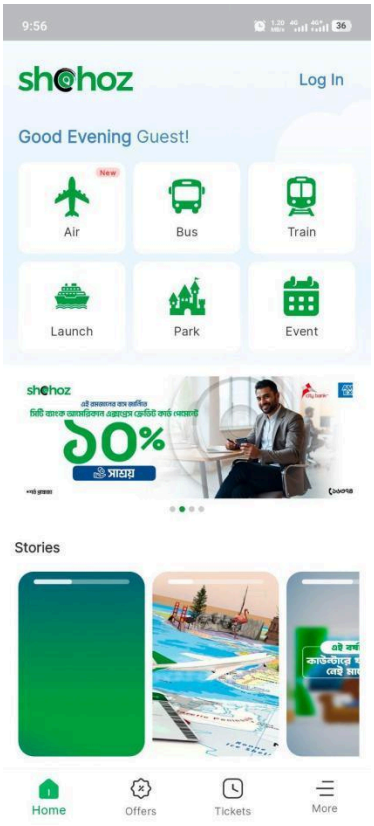


Shohoz:

- **Ride Matching & Booking:** Real-time ride matching for Bus, Train, Launch.
- **Ride Scheduling & Recurring Rides:** Limited scheduling options.
- **Fare Estimation & Payment Integration:** Fare estimation and cash payments.
- **Customer Support:** In-app chat support.
- **Company Features:** Earnings tracking and payout system.
- **Safety Measures:** Basic verification.
- **Admin Features:** Manages company's onboarding and payments.



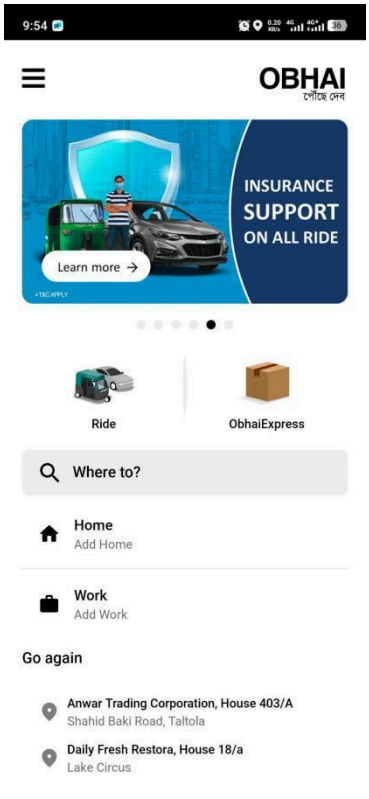


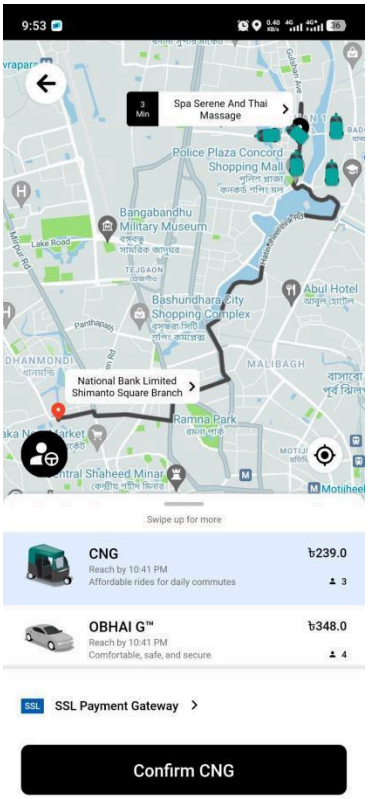


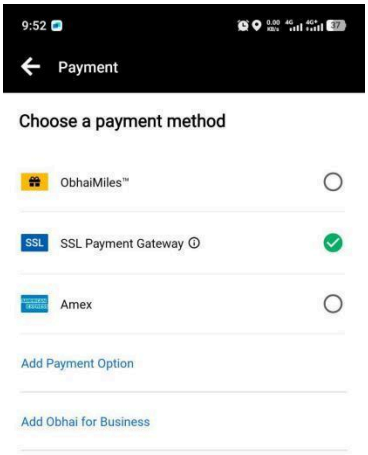
Obhai:

- **Ride Matching & Booking:** Real-time ride matching for bikes and cars.
- **Ride Scheduling & Recurring Rides:** Limited scheduling options.
- **Fare Estimation & Payment Integration:** Fare estimation and cash payments.

- **Driver Rating & Feedback System:** Two-way rating system.
- **Customer Support:** In-app chat support.
- **Driver Features:** Earnings tracking and payout system.
- **Safety Measures:** Basic rider/driver verification.
- **Admin Features:** Manages driver onboarding and payments.







inDrive:

- **Ride Matching & Booking:** Real-time ride matching with price negotiation.
- **Ride Scheduling & Recurring Rides:** Limited scheduling options.
- **Fare Estimation & Payment Integration:** Fare negotiation and cash payments.
- **Driver Rating & Feedback System:** Two-way rating system.
- **Customer Support:** In-app chat support.
- **Driver Features:** Earnings tracking and payout system.
- **Safety Measures:** Basic rider/driver verification.

- **Admin Features:** Manages driver accounts and complaints.

2.4.2. User survey:

The survey was conducted using a **Google Form**. Here is the summary of the information gathered:

Key Information Extracted from CampusRide Survey Responses:

1. Respondent Roles:

- Students: 24 responses.
- University Staff: 1 response.

2. Age Groups:

- 18–30: 24 responses.
- 31–45: 1 response (University Staff).

3. Commuting Methods:

- Public Transport (bus/train): 12 responses.
- University Buses/Shuttles: 9 responses.
- Personal Vehicle: 3 responses.
- Others: 3 responses (unspecified).

4. Biggest Commuting Challenges (Most Common):

- High cost (8 responses).
- Safety concerns (7 responses).
- Long wait times (10 responses).
- Unreliable availability (4 responses).
- Other: Generic "Other" (4 responses).

5. Monthly Transportation Spending:

- **Less than ৳1,000: 8 responses.**
- **৳1,000–৳3,000: 13 responses.**
- **৳3,000–৳5,000: 2 responses.**
- **More than ৳5,000: 1 response.**

6. Safety Perception of Existing Apps (Scale 1–5):

Average Rating: ~3.3/5.

Breakdown:

- 1 (Very unsafe): 2 responses.
- 2: 2 responses.
- 3: 9 responses.
- 4: 6 responses.
- 5 (Very safe): 5 responses.

7. Preference for University-Verified App:

- Yes: 17 responses.
- Maybe: 6 responses.
- No: 2 responses.

8. Most Important Safety Features:

Top Features:

- Real-time ride tracking (12 responses).
- SOS button (8 responses).
- University ID verification (10 responses).
- Driver/rider ratings (7 responses).
- Other: Data encryption, collaboration with campus security (mentioned in open comments).

9. Gaps in Existing Apps (e.g., Uber, Pathao):

Most Cited:

- Lack of student discounts (12 responses).
- No university-specific routes (8 responses).
- Poor customer support (6 responses). High pricing (11 responses).
- Unverified drivers/riders (7 responses).

10. Likelihood to Use Group Rides for Campus Events

- Very Likely: 6 responses.
- Likely: 4 responses.
- Neutral: 5 responses.
- Unlikely: 3 responses.
- Very Unlikely: 3 responses.

11. Importance of Student/Staff Discounts

Scale 1–5:

- 5 (Critical): 15 responses.
- 4: 2 responses.
- 3: 3 responses.
- 1 (Not important): 2 responses.

12. University Support for ID Verification:

- Yes: 7 responses.
- No: 3 responses.
- Needs Discussion: 10 responses.
- Blank: 1 response.

13. Security Measures to Prioritize - Top Measures:

- Two-factor authentication (9 responses).
- Collaboration with campus security (10 responses).
- Data encryption (6 responses).
- Other: Driver background checks (implied in comments).

14. Dislikes About Existing Apps - Top Complaints:

- High pricing (15 responses).
- Unverified drivers/riders (8 responses).
- Poor customer support (8 responses).
- Lack of safety features (6 responses).

15. Safety Issues Faced on Ride-Sharing Apps:

- Yes: 4 responses.
- No: 17 responses.
- Maybe: 2 responses.

16. Trust in University-Verified App vs. General Apps:

- Yes: 19 responses.
- Maybe: 4 responses.
- No: 1 response.

17. Preferred Payment Methods:

- Cash: 17 responses.
- Mobile Banking (bKash/Nagad): 15 responses.
- Credit/Debit Cards: 5 responses.
- In-App Wallet: 1 response.

18. Additional Suggestions - Key Feedback:

- Location-sharing feature for safety (1 response).

- "Build trust first, then focus on money."
- Simplify ID verification and ensure helmet safety.
- "Easy to use" interface.

This data can guide Campus Rides development to address gaps in existing apps and align with user needs.

2.4.3. Research Paper Analysis:

2.4.3.1 "A Ride Sharing Service for University Community"

TULTECH

International Journal of Innovative Technology and Interdisciplinary Sciences
www.IJITIS.org
ISSN:2613-7305
Volume 5, Issue 2, pp. 907-924, 2022
DOI: https://doi.org/10.15157/IJITIS.2022.5.2.907-924
Received February 14, 2022; Accepted April 1, 2022

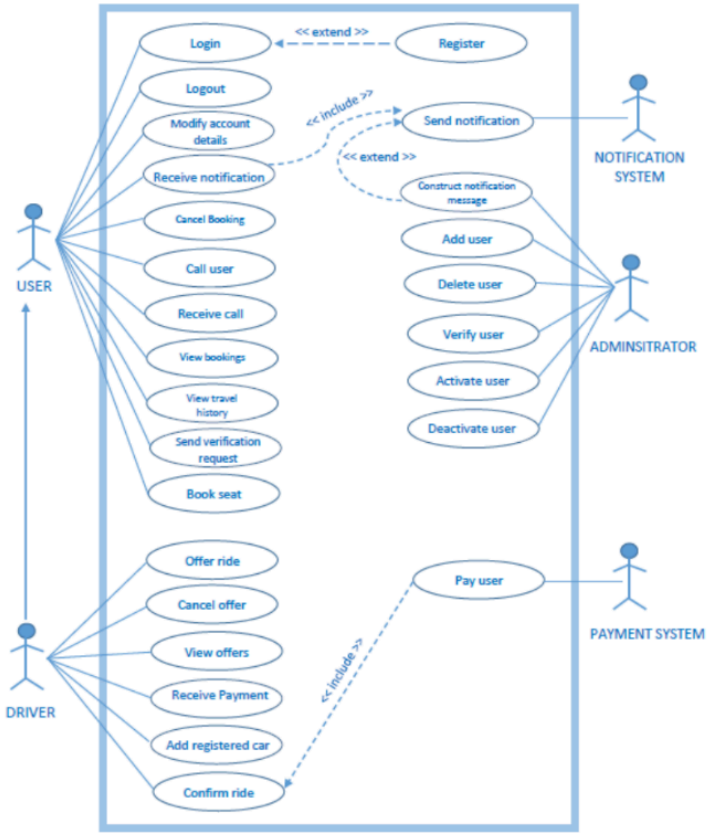
A Ride Sharing System for University Community

Benjamin Kommey[*a1], Benjamin Asane Asante[b1], Elvis Tamakloe[c1]

[1]Department of Computer Engineering, Kwame Nkrumah University of Science and Technology, 00233 Kumasi, Ghana

[*a]bkommey.coe@knust.edu.gh; [b]calculusaffairs@gmail.com; [c]tamakloe.elvis@gmail.com

- **Objective:** Develop an online ride-sharing system for university communities to address transportation challenges, reduce costs, and lower carbon emissions.
- **Key Features:**
  - Mobile and web-based platform for ride-sharing and booking.
  - User registration, travel advertisement, booking, geo-location, routing, and notifications.
  - Integration of GPS and navigation services**.**
- **Benefits:**
  - Reduces fuel costs and traffic congestion.
  - Minimizes carbon emissions and travel stress.
  - Enhances safety and convenience for students, especially at night.
- **System Architecture:**
  - **Users:** Drivers and passengers.
  - **Components:** Database, notification system, admin panel, and user devices.



- **Implementation:**
  - Tested on Android and iOS platforms.
  - Future work includes expanding to other operating systems and adding cashless payment options.
- **Impact:**
  - Promotes carpooling, reduces traffic, and supports sustainable transportation.

2.4.3.2 "A Study on Motor Ride Sharing Service in Dhaka City, Present and Future Challenges"

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.9619

A Study On Motor Ride Sharing Service In Dhaka City, Present And Future Challenges

Md. Rasidul Islam[1], Md. Hasnul Habib[2], Hasan Ahmed Joy[3],

A. B. M. Shamsul Haque Nayem[4]

Department of Civil Engineering, Taiyuan University of Technology -TYUT. Shanxi, China[1]

Dept. of Civil Engg, Bangladesh University of Engineering and Technology-BUET. Dhaka-1219, Bangladesh[2]

Department of Civil Engineering, Stamford University of Bangladesh, Dhaka-1219, Bangladesh[3]

Department of Civil Engineering, Stamford University of Bangladesh, Dhaka-1219, Bangladesh[4]

- **Objective:** The study aims to analyze the current state and future challenges of motor ride-sharing services in Dhaka City. It explores user experiences, market growth, and obstacles faced by the industry.
- **Key Features:**
  - It highlights key issues such as funding constraints, lack of logistical support, poor government regulations, and intense competition.
  - The research emphasizes the role of ride-sharing in reducing traffic congestion and improving transportation efficiency.
- **Benefits:**
  - Provides real-time, on-demand transportation, making travel more convenient.
  - Appeals especially to younger generations and businesses due to its flexibility and ease of use.
  - Reduces the number of private vehicles on the road, contributing to better traffic management and environmental benefits.
- **System Architecture:**
  - **User Interface (UI):** Mobile application for users and drivers with features for booking rides, tracking trips, and making payments.
  - **Backend System:** Handles data processing, trip allocation, user authentication, and route optimization.
- **Implementation:**
  - **User Onboarding:** Both riders and drivers register via the app, submitting required documents for verification.
  - **Booking Process:** Users request rides by entering pick-up and drop-off locations.
- **Impact:**
  - Reduces commute time with on-demand service.

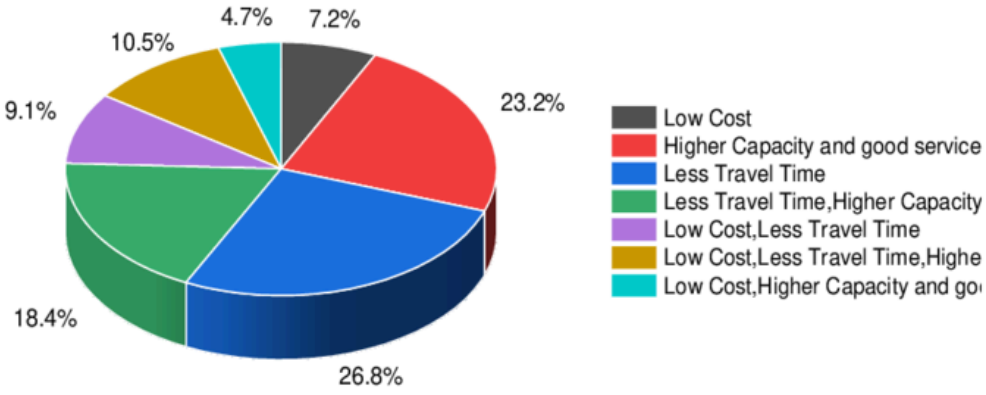


Figure 7: Factors affecting to use Ride-sharing

Challenges:

Badly needed government support.

Increasing smartphone users and teach them about this technology then customer use too

Rent must be kept within limits So that the customer is interested for ride sharing.

The information gathering phase allowed us to establish a complete understanding of CampusRide's target market along with its competitive situation and industry trends. User research with competitor analysis and academic research enables the development of specific and useful requirements. User data collected through surveys and platform assessments and academic research documents will directly guide the design of CampusRide to provide secure competitive transportation service for university students. The gathered data will guide the following requirement elicitation and system design phases with the purpose of building a user-focused ride-sharing platform.

3. System Analysis

3.1. Introduction

The **CampusRide** project is designed to address the transportation needs of students, faculty, and staff within and around a university campus. As urban campuses continue to grow and expand, the demand for a reliable, efficient, and cost-effective transportation solution has become increasingly important. The aim of this system analysis is to explore the functional and non-functional requirements, evaluate current transportation issues, and identify opportunities for process improvement through the development of a ride-sharing platform tailored specifically to the campus environment.

This system analysis will focus on understanding the existing problems students face, such as irregular transport availability, high travel costs, and lack of coordination among commuters. It will also assess user requirements, system feasibility, and technical constraints to ensure the proposed CampusRide system meets the expectations of its end users. By analyzing the system's operational environment and engaging with potential users, the analysis phase will lay a strong foundation for designing a robust, user-friendly, and scalable ride-sharing solution.

3.2. Comparison Table

Feature	Campusride	Uber	Pathao	Shohoz	Obhai	inDrive
University-Exclusive Carpooling	✔	✘	✘	✘	✘	✘
Ride Scheduling & Recurring	✔	✔	✘	✘	✘	✘
Fare Estimation & Payment	✔	✔	✔	✔	✔	✔
Driver Rating & Feedback	✔	✔	✔	✔	✔	✔
Customer Support	✔	✔	✔	✔	✔	✔
Ride Listing & Availability	✔	✔	✔	✔	✔	✔
Earnings & Payout System	✔	✔	✔	✔	✔	✔
User Verification & Safety	✔ (University verified)	✔	✘	✘	✘	✘
Customer Care Support	✔	✔	✔	✔	✔	✔
Sustainability & Traffic Reduction	✔ (Unique)	✘	✘	✘	✘	✘

3.3. Gap Analysis

Identified Gaps in Competitors:

1. No university-exclusive user base → security risks.
2. High pricing for students.
3. Limited carpooling options for campus events.
4. No integration with university systems (e.g., ID verification).

How CampusRide Fills Gaps:

1. Mandatory university email/ID verification.
2. Discounted fares for students.
3. Dedicated “Event Rides” for campus activities.
4. Seamless integration with university systems for authentication and access control.
5. Sharing Rides with your fellow mates.

3.4. Feature List Finalization

Features for users:

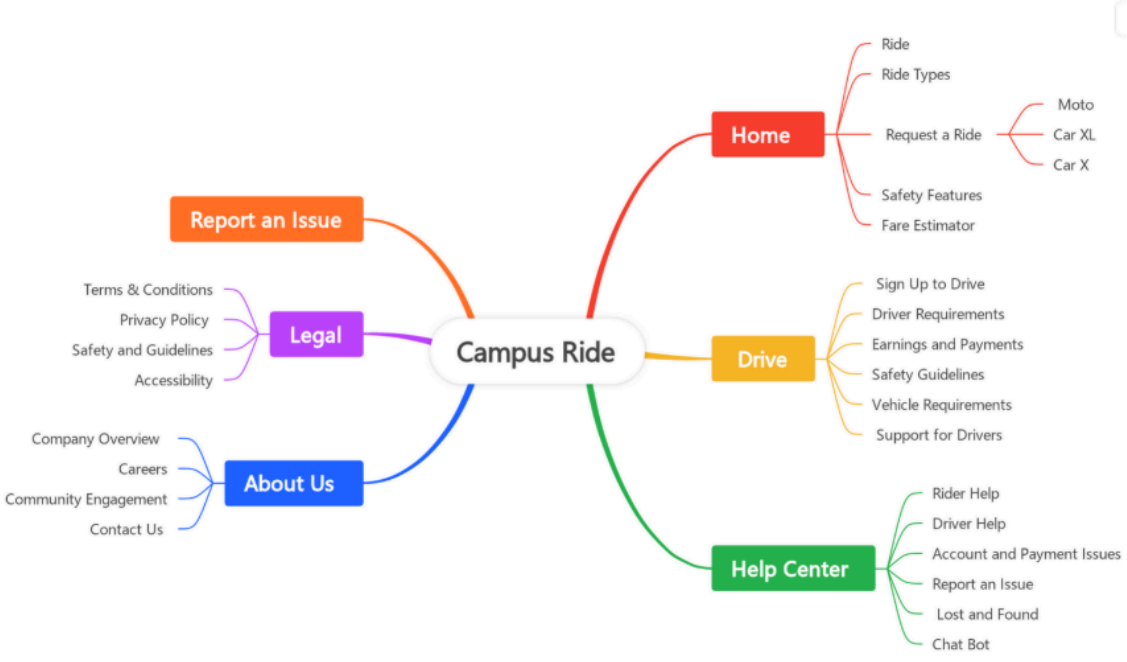
1. Ride Matching & Booking.
2. Ride Scheduling & Recurring Rides.
3. Fare Estimation & Payment Integration.
4. Driver Rating & Feedback System.
5. Customer Support.
6. Make Complaints.

Features for Drivers and Admins:

1. Ride Listing & Availability Setting.
2. Earnings & Payout System.
3. User Verification & Safety Measures.
4. Customer Care Support.
5. Manages.

3.5. SiteMap

Sitemap (Proposed Structure)



4. Feasibility Analysis

4.1. Introduction

CampusRide is a ride-sharing service for university students, staff, and faculty only. The system allows confirmed users from a particular university to give or get rides on or around the campus, promoting affordable, safe, and green transportation.

Types of Ride Sharing on CampusRide
Carpooling (Peer-to-Peer): Students or staff provide a ride in their personal car.

Motorbike Sharing: Widespread use in Bangladesh for traveling short distances.

Campus Shuttles (Optional, Future Feature): University-managed routes and shared costs.

On-Demand Matching: Dynamic ride offers and requests based on proximity and time.

4.2. SWOT Analysis

Strengths

- **Exclusive to University Members –** Provides security and trust.
- **Cost-Effective –** Reduces daily travel expenses.
- **Eco-Friendly Approach –** Promotes ride-sharing, reducing carbon footprint.
- **Targeted Niche Market –** Clear audience focus (students & staff).
- **Convenient Payment System –** Integration with mobile wallets.

Weakness

- **Limited User Base –** Only university students and staff can use it.
- **Dependence on Active Drivers –** If not enough drivers, service becomes unreliable.
- **Operational & Regulatory Challenges –** Need university cooperation for verification.
- **Initial Adoption Challenge –** Students may be hesitant to switch from existing options.

Opportunities

- **Expansion to Multiple Universities –** Scaling beyond one campus.
- **Strategic Partnerships –** Collaborating with universities for official endorsement.
- **Integration with Public Transport –** Hybrid model with buses or local transit.
- **AI-based Matching & Pricing –** Smart algorithms to optimize rides and costs.

Threats

- **Competition from Established Players –** Uber, Pathao, Shohoz, etc.
- **Regulatory Issues –** Government or university restrictions on ride-sharing.
- **User Trust & Safety Concerns –** Any incident can damage reputation.
- **Sustainability of Business Model –** If commission-based, profit margins could be thin.

4.3. Cashflow (1-Year Projection in BDT)

Campus Ride (cash flow)

		Month 1	Month 2	Month 3	Month 4	Month 5	Month 6	Month 7	Month 8	Month 9	Month 10	Month 11	Month 12
Reveneu	Ride Commisions	৳0.00	৳0.00	৳0.00	৳15,000.00	৳20,000.00	৳35,000.00	৳55,000.00	৳60,000.00	৳75,000.00	৳90,000.00	৳110,000.00	৳125,000.00
	Partnership(University Ads, sponsorship)	৳10,000.00	৳10,500.00	৳11,025.00	৳11,576.25	৳12,155.06	৳12,762.82	৳13,400.96	৳14,071.00	৳14,774.55	৳15,513.28	৳16,288.95	৳17,103.39
	Other(Donation, promotion)		৳15,000.00	৳15,750.00	৳16,537.50	৳17,364.38	৳18,232.59	৳19,144.22	৳20,101.43	৳21,106.51	৳22,161.83	৳23,269.92	৳24,433.42
	Total Reveneu	৳10,000.00	৳25,500.00	৳26,775.00	৳43,113.75	৳49,519.44	৳65,995.41	৳87,545.18	৳94,172.44	৳110,881.06	৳127,675.11	৳149,558.87	৳166,536.81
Expenses	Server & Hoisting costs	৳20,000.00	৳0.00	৳0.00	৳0.00	৳0.00	৳0.00	৳5,000.00	৳5,000.00	৳5,000.00	৳5,000.00	৳5,000.00	৳5,000.00
	Maintainance & Update	৳15,000.00	৳15,375.00	৳15,759.38	৳16,153.36	৳16,557.19	৳16,971.12	৳17,395.40	৳17,830.29	৳18,276.04	৳18,732.94	৳19,201.27	৳19,681.30
	Marketing & Promotion	৳30,000.00	৳30,750.00	৳31,518.75	৳32,306.72	৳33,114.39	৳33,942.25	৳34,790.80	৳35,660.57	৳36,552.09	৳37,465.89	৳38,402.54	৳39,362.60
	Payment Processing fees(BKash, Nagad etc)	৳15,000.00	৳15,375.00	৳15,759.38	৳16,153.36	৳16,557.19	৳16,971.12	৳17,395.40	৳17,830.29	৳18,276.04	৳18,732.94	৳19,201.27	৳19,681.30
	Others Cost	৳5,000.00	৳5,125.00	৳5,253.13	৳5,384.45	৳5,519.06	৳5,657.04	৳5,798.47	৳5,943.43	৳6,092.01	৳6,244.31	৳6,400.42	৳6,560.43
	Total Expenses	৳85,000.00	৳66,625.00	৳68,290.63	৳69,997.89	৳71,747.84	৳73,541.53	৳80,380.07	৳82,264.57	৳84,196.19	৳86,176.09	৳88,205.50	৳90,285.63
Cash Flow(TR-TE)		-৳75,000.00	-৳41,125.00	-৳41,515.63	-৳26,884.14	-৳22,228.40	-৳7,546.12	৳7,165.11	৳11,907.86	৳26,684.87	৳41,499.02	৳61,353.37	৳76,251.18
Cummulative Cash Flow		-৳75,000.00	-৳116,125.00	-৳157,640.63	-৳184,524.77	-৳206,753.17	-৳214,299.29	-৳207,134.18	-৳195,226.32	-৳168,541.45	-৳127,042.42	-৳65,689.05	৳10,562.13

Assuming local transaction costs, limited marketing in early months, and scaling up user adoption gradually.

4.4. Behavioral & Technical Feasibility

Behavioral Feasibility

Target Users: Students are price-conscious but appreciate convenience and community.

Trust & Safety: Needs verification (university email/ID), rating system, and public profiles.

Incentives: Early users may need discounts or incentives to drive adoption.

Adoption Potential: High, especially in universities with poor public transportation or large campuses.

Technical Feasibility
Tech Stack: React, Firebase (auth + DB), Mapbox/OpenStreetMap API (free), Node.js backend.

Location Services: Accurate GPS in urban campuses.

Hosting & Dev Tools: Affordable hosting (e.g., Hostinger) and free-tier cloud services available.

Integrations: SSLCommerz or bKash for payment processing is feasible and documented.

4.5. Feasibility Recommendation

Cash Flow Insights (BDT)
Loss of Initial Investment: First 6 months of operations are in a net negative cash flow, as is typical with early-stage technology startups.

Break-even Trend: Since Month 7, cash flow gets better with reduced losses and then reverses to positive net flow by Month 9.

Revenue Growth: Strong and consistent growth in ride commissions, partnerships, and other sources.

Cost Stability: Baseline costs of server, maintenance, and level of promotion stabilize, suggesting peak cost control.

Overall Recovery: Even though the app loses -৳60,000 to -৳70,000 overall at the end of the period, the monthly margin in Month 12 is +৳74,000, which reflects profitable momentum.

5. System Design

5.1. Introduction (UML Concept)

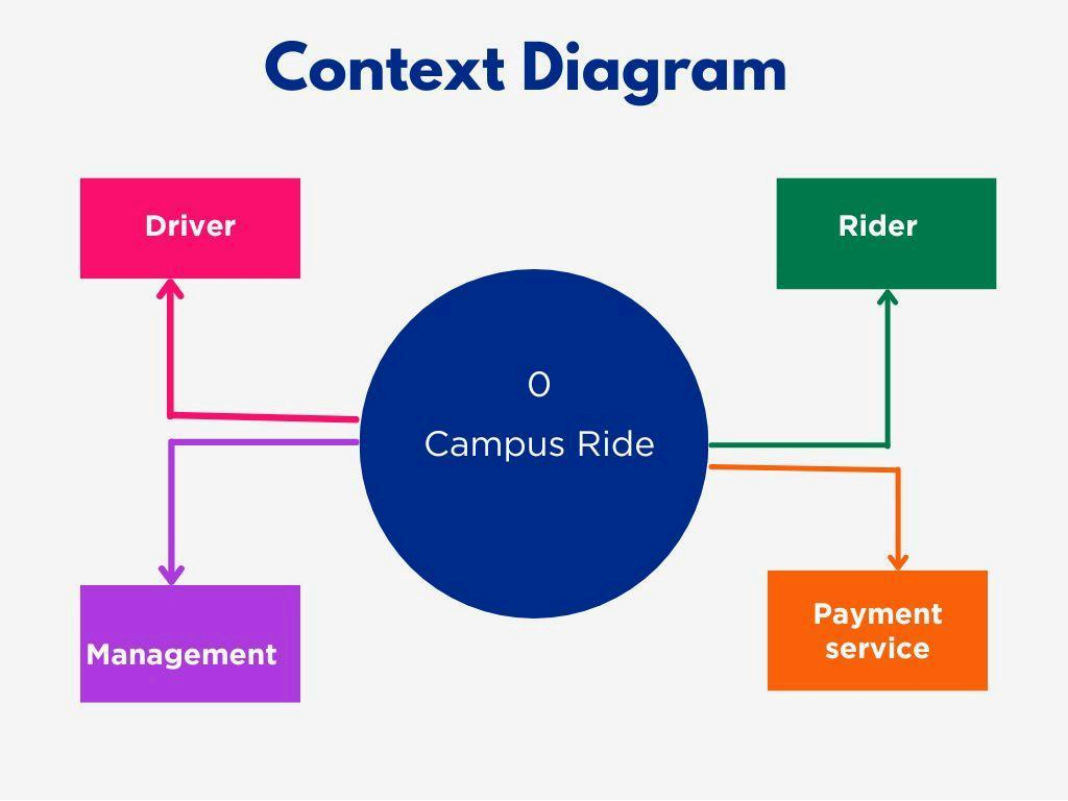
Unified Modeling Language (UML) is a standardized visual modeling language used in software engineering to represent the design and structure of a system. It provides a set of diagrams and notations to help developers, designers, and stakeholders visualize how the system will function, how components interact, and how data flows.

In the context of our community-based ride-sharing system, UML diagrams play a crucial role in communicating system requirements, use cases, and internal architecture. The use of UML ensures a clear and consistent understanding of system behavior across all team members.

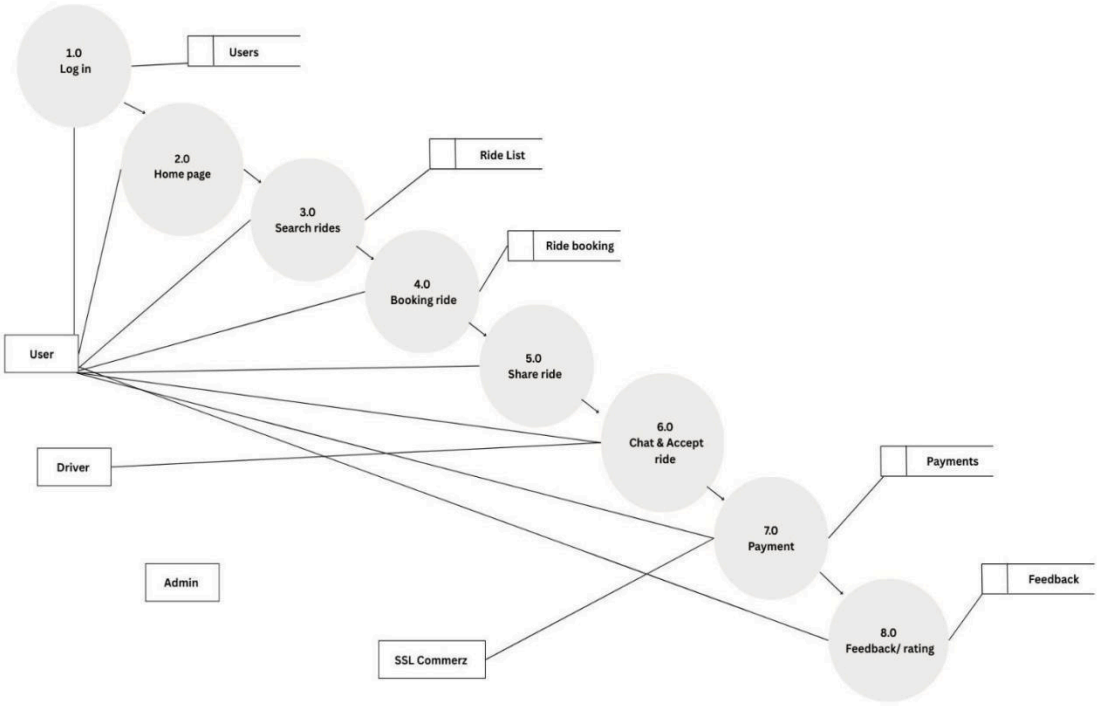
We will use the following UML diagrams to design and explain our system:

- **Use Case Diagram** – To show the major functions of the system and the interactions between users (passengers and drivers) and the system.
- **Class Diagram** – To represent the data model and the relationships between different entities such as User, Ride, Payment, etc.
- **Sequence Diagram** – To describe the flow of operations in scenarios like ride booking or payment processing.
- **Activity Diagram** – To visualize the workflow or business logic for processes like ride matching or fare calculation.
- **Component Diagram** – To illustrate the high-level structure of software components and their dependencies.

5.2. Context Diagram



5.3. Data Flow Diagram



5.4 Use Case Diagram

Use Case Diagram:

A **Use Case Diagram** provides a high-level overview of how users (actors) interact with a system, highlighting key functionalities. It serves to summarize the relationships between actors, use cases, and the system, offering a broad yet simplified understanding. The diagram does not display the sequence of actions or the specific flow of steps required to achieve the objectives of a use case but rather focuses on key interactions. Ideally, a use case diagram should be kept simple and concise, incorporating only essential shapes for clarity and effectiveness.

Use Case:

A Use Case refers to a sequence of actions or events that define how a system interacts with an actor (which can be a user or another system) to accomplish a specific goal. The purpose of a use case is to describe a scenario of interactions between the actor and the system to achieve the

desired outcome. These interactions are usually outlined step by step, making it clear how a user or external system engages with the software in different situations.

Actor:

In the context of a use case diagram, an Actor is any entity (person, device, or system) that interacts with the system to perform tasks. An actor represents a particular role, not necessarily a specific physical entity, but an aspect or function that the entity serves in the context of the system. An actor is external to the system but plays a significant role in its operation.

Identifying Actors:

To identify actors in a system, consider the following critical questions (Schneider and Winters, 1998):

> Who interacts with the system?

> Who installs or sets up the system?

> Who is responsible for starting up the system?

> Who maintains and supports the system?

> Who shuts down or disables the system?

>Which other systems interact with or rely on the system?

> Who receives or retrieves data from the system?

> Who provides data or input to the system?

> Are there any automated processes or actions that happen independently at specific times?

System Boundary:

A **System Boundary** is a visual element represented by a rectangle in a use case diagram. It serves to distinguish between the internal components (use cases) of a system and external

entities (actors) that interact with the system. The boundary acts as a visual aid, helping to clarify which use cases are part of the system's functionalities and which actors exist outside of it. However, it is important to note that the system boundary does not carry any semantic meaning or influence the model's structure; its role is purely for clarity and organization.

Association:

An **Association** represents the relationship between an **actor** and a **use case** in a system. It indicates that an actor has access to or utilizes a particular functionality of the system. An association is depicted as a line connecting an actor to a use case.

Despite its simplicity, the association does not specify the details of how the interaction occurs. For example, it does not clarify whether multiple actors can perform a use case individually or must collaborate to execute it. If a use case is associated with several actors, the diagram won't reveal whether each actor can execute the use case independently or if they must work together. In essence, an association simply signifies that an actor participates in the use case, but it leaves out the specifics of the interaction process.

References:

- "OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, pp. 586–588" (Archived from the original on 2010-09-23, Retrieved November 7, 2010).

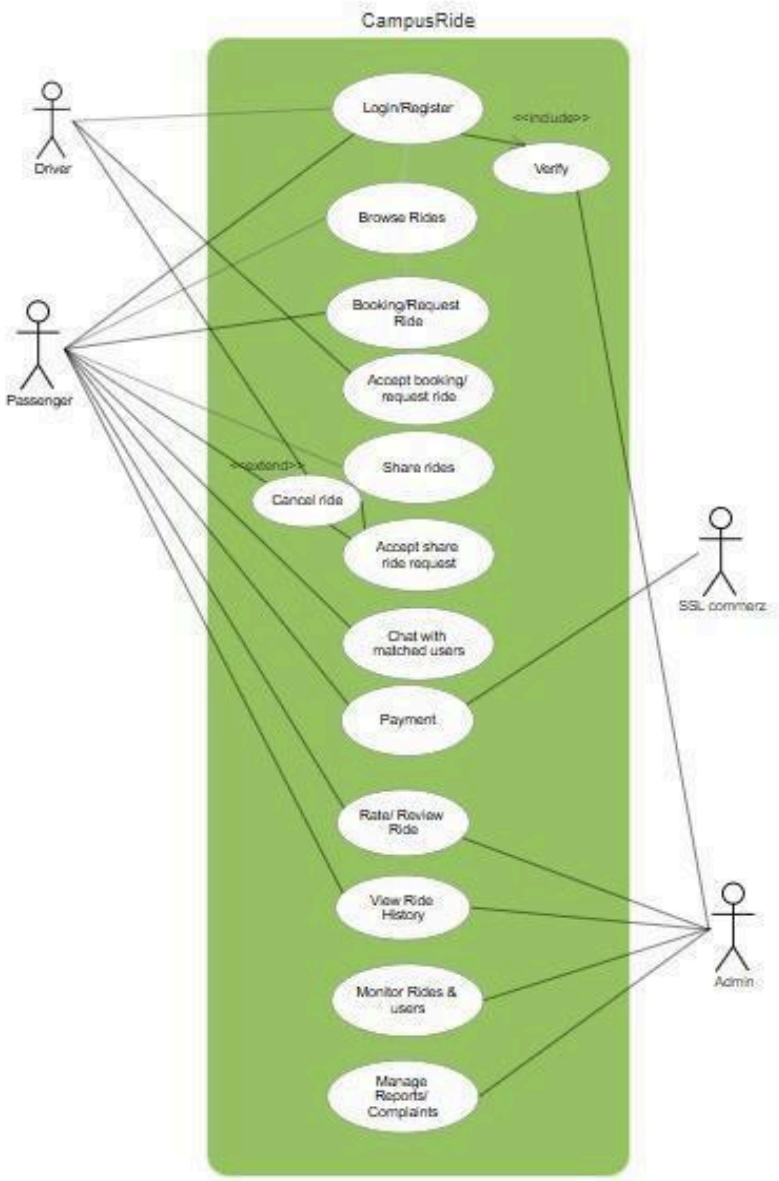
- "Problems and Deficiencies of UML as a Requirements Specification, s.3.2" (Archived (PDF) from the original on 17 October 2010, Retrieved November 7, 2010).

- "UML 2 Specification" (Retrieved July 4, 2012).

- *System Analysis and Design*, 5th Edition by Alan Dennis, Barbara Haley Wixom, Roberta M. Roth.

Actors of our Project:

• Passenger
• Driver
• ADMIN
• ADMIN
• SSL commerce



5.5 Use Case Descriptive form (Only for major use cases)

5.5.1 Use Case 1: Ride Booking

Use Case Name: Ride Booking
Actor(s): Passenger (User)

Description: The passenger requests a ride by entering the pickup and destination locations. The system finds available drivers and sends the request to them.

Precondition(s):

- The user is logged into the system.
- The user has GPS/location enabled.

Basic Flow (Main Success Scenario):

- The passenger opens the app and chooses the “Book Ride” option.
- Enter pickup and drop-off locations.
- Confirms ride request.
- The system broadcasts the request to available drivers nearby.
- Displays "Searching for Driver" status.

Postcondition(s):

- The ride request is sent and visible to nearby drivers.

Alternative Flows:

- **Location not found:** The system prompts the user to manually input the address.
- **No drivers available:** The system notifies the user and suggests trying again later.

5.5.2 Use Case 2: Ride Sharing (Driver Accepts Ride)

Use Case Name: Ride Sharing / Ride Acceptance
Actor(s): Driver

Description: A driver receives a ride request and chooses to accept it. Once accepted, the ride is assigned to that driver.

Precondition(s):

- Driver is logged in and marked as “Available”.
- Driver has location access enabled.
-

Basic Flow (Main Success Scenario):

- Driver receives a ride request notification.
- Views passenger’s pickup and destination locations.
- Accepts the request

- The system assigns the ride to the driver and notifies the passenger.
- Driver navigates to the pickup location.

Postcondition(s):
- Ride is assigned and active for both driver and passenger.

Alternative Flows:
- **Driver declines:** The request is sent to the next available driver.
- **Driver accepts but cancels later:** System informs the passenger and restarts driver search.

5.5.3 Use Case 3: Payment

Use Case Name: Payment
Actor(s): Passenger, Driver

Description: After the ride ends, the system calculates the fare based on distance/time. The passenger pays using cash or mobile banking, and the driver confirms the payment.

Precondition(s):
- A ride has been completed.

Basic Flow (Main Success Scenario):
- The system calculates total fare after reaching the destination.
- Passenger chooses payment method (cash or mobile banking).
- Passenger pays the fare.
- Driver confirms payment received.
- System logs the payment and completes the ride session.

Postcondition(s):
- Payment is marked complete and the ride session is closed.

Alternative Flows:
Payment failure (mobile): System prompts the passenger to retry or use cash.
Driver disputes non-payment: The system flags the ride for admin review.

5.6 Activity Diagram

Overview:

An **Activity Diagram** is a type of UML (Unified Modeling Language) diagram used to model the dynamic aspects of a system. It shows the flow of control or data from one activity to another, representing the sequence of actions, decisions, and events in a system. It is particularly useful for describing workflows, business processes, and the flow of control in system operations.

Activity diagrams can be seen as a flowchart that visualizes the execution of processes and operations. They are essential in understanding the behavior of a system in response to various events and user actions.

Key Components of Activity Diagram:

1. **Initial Node**:
    - Represented by a filled black circle. This marks the starting point of the activity diagram, where the process begins.
2. **Activity/Action**:
    - Represented by a rounded rectangle. It represents a task or action that occurs in the system, which might involve data processing, decision-making, or other tasks.
3. **Decision Node**:
    - Represented by a diamond shape. A decision node indicates a branching point where the flow of control can go in different directions based on a condition or event.
4. **Merge Node**:
    - Also represented by a diamond shape. It merges multiple flow paths into one. It doesn't have a condition attached to it, and it simply combines different paths after they have diverged.
5. **Fork Node**:

    - Represented by a solid horizontal or vertical bar. A fork splits a flow into multiple concurrent activities, allowing parallel execution of actions.

6. **Join Node**:

    - Represented by a solid horizontal or vertical bar. It joins multiple flows that were split at a fork, ensuring that all parallel tasks are completed before proceeding.

7. **Final Node**:

    - Represented by a filled circle with a surrounding border. The final node indicates the end of the activity diagram, signifying the completion of the process.

8. **Control Flow**:

    - Represented by arrows. These arrows show the direction of flow from one activity to another, indicating the sequence of operations or tasks in the process.

9. **Object Flow**:

    - Represented by dashed arrows. These indicate the flow of objects (data, files, or other entities) between actions or activities, showing how information moves through the system.

10. **Swimlanes**:

- Represented by vertical or horizontal partitions. Swimlanes are used to categorize activities based on who or what is responsible for them (e.g., user, system, external entity).

Steps to Draw an Activity Diagram:

1. **Identify the Start and End Points**: Clearly define the initial and final nodes of the process. This is the first step in mapping out the flow of actions.

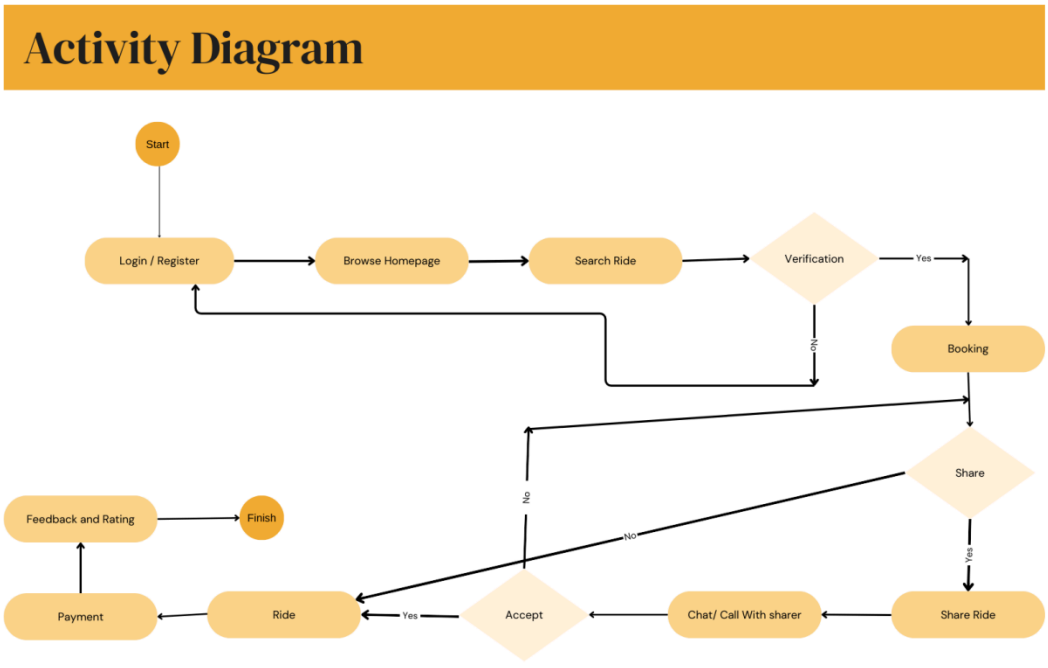
2. **Identify the Activities**: Break down the process into individual actions or tasks that need to be performed.

3. **Define the Flow of Control**: Identify the sequence in which activities occur and the conditions that may alter the flow of control (e.g., decisions or loops).

4. **Identify Parallel Activities**: If the process involves tasks that can be done concurrently, use fork and join nodes to represent parallel flows.

5. **Handle Decision and Merge Points**: Use decision nodes where conditional branching occurs, and merge nodes to bring together different flow paths.

6. **Use Swimlanes (Optional)**: Organize the diagram into swimlanes to indicate which entities (e.g., users or systems) are responsible for each.



5.7 Swimlane Diagram

Overview:

A **Swimlane Diagram** is a type of flowchart that visually distinguishes different responsibilities within a process by dividing the diagram into "swimlanes." Each lane represents an actor, entity, system, or department responsible for a set of actions or tasks. Swimlane diagrams are helpful in clarifying roles and responsibilities in a process, especially in complex workflows where multiple entities are involved.

A **Swimlane Diagram** is commonly used to model business processes, system workflows, and interactions between different components of a system. By organizing actions into distinct lanes, the diagram provides clarity on who or what is responsible for each step, thus reducing ambiguity.

Key Components of a Swimlane Diagram:

1. **Swimlanes**:
    - Represented by vertical or horizontal divisions in the diagram. Each swimlane represents a distinct entity (person, department, system, or role) responsible for specific tasks in the process. The lanes are named according to the entities they represent (e.g., "Customer," "Order System," "Shipping Department").
2. **Activities/Actions**:
    - Represented by rectangles within the swimlanes. Each activity or action shows a task or decision point that takes place in the workflow. These tasks are associated with the entities or systems that perform them.
3. **Flow Arrows**:
    - Represented by arrows connecting the activities. These arrows show the sequence of tasks and the direction in which the process flows.
4. **Start and End Points**:
    - The diagram begins with a start symbol (usually a filled circle) and ends with an end symbol (a circle with a border). These points mark the beginning and

conclusion of the process.

5. **Decision Points**:
    - Represented by diamonds. Decision points indicate where a process might diverge based on conditions or criteria, leading to different paths in the workflow.
6. **Transitions/Control Flows**:
    - Indicated by arrows between activities or decision points, these flows show the progression of tasks through the system.

Types of Swimlane Diagrams:

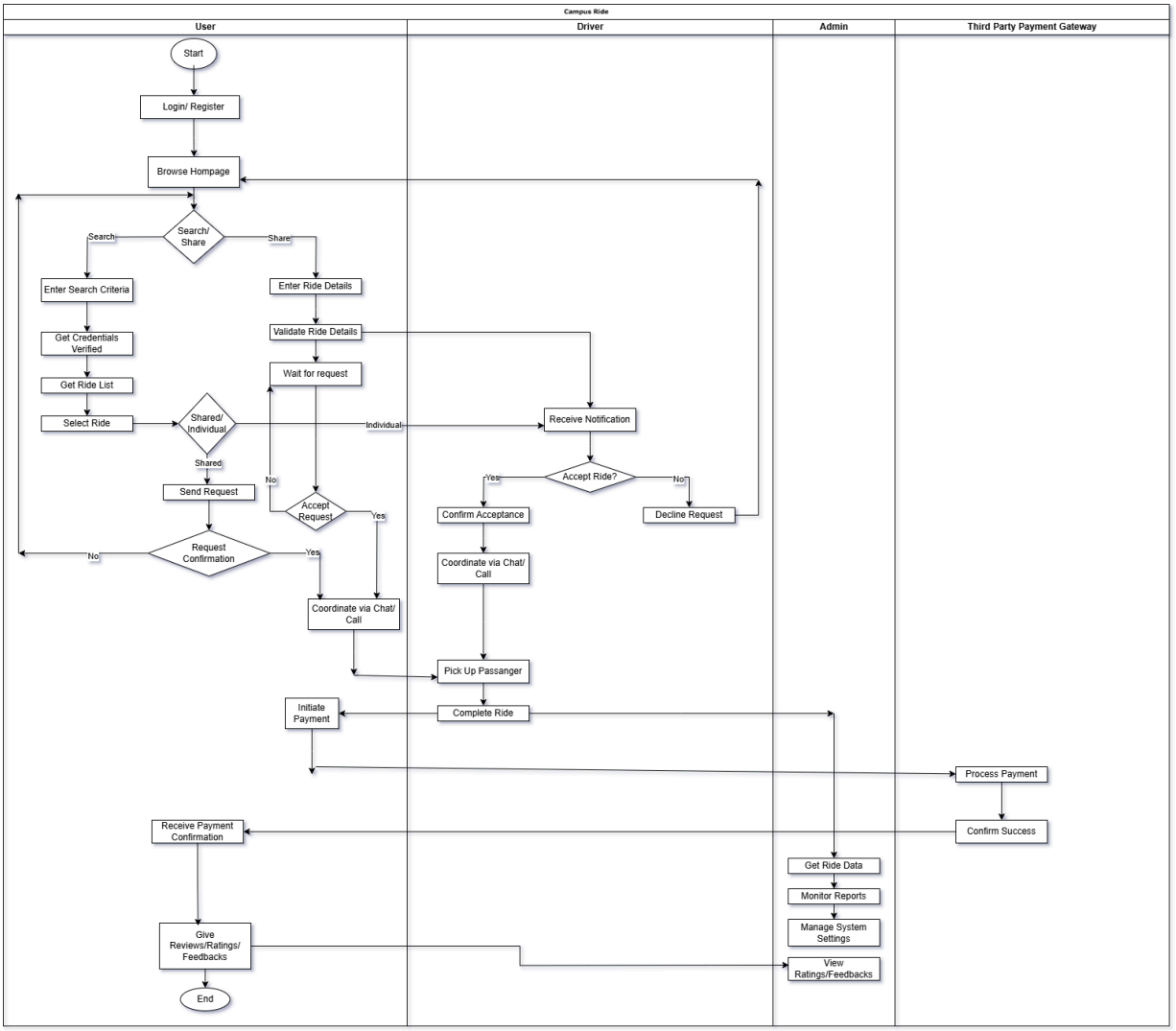
1. **Vertical Swimlane Diagram**:
    - In this type, swimlanes are arranged vertically, and each lane represents an actor or entity responsible for different steps in the process.
2. **Horizontal Swimlane Diagram**:
    - Here, the swimlanes are arranged horizontally. The entities or systems responsible for each action are represented in horizontal bands.

Steps to Draw a Swimlane Diagram:

1. **Identify the Process**: Understand the process or workflow you need to represent. Identify the different entities or actors involved in the process.
2. **Define the Swimlanes**: Label each swimlane with the appropriate entity or role that will be responsible for the activities within that lane.
3. **Map the Activities**: Identify the individual tasks or actions in the process and place them within the appropriate swimlanes based on the actor or entity performing them.
4. **Determine Flow**: Determine the flow of activities between the entities. Connect the tasks using arrows to show the sequence of steps.

5. **Add Decision Points**: Identify decision points in the workflow where conditions change the course of action, and represent them with diamonds.

6. **Finalize the Diagram**: Ensure the diagram clearly shows the process flow, who is responsible for each task, and the relationship between tasks.



5.8 CRC Diagrams :

A **CRC Diagram** stands for **Class–Responsibility–Collaborator Diagram**, and it is a fundamental tool in **object-oriented design**. It helps designers identify and organize the components (classes) of a system by clearly outlining:

1. **Class** – //needs to improve The object or entity in the system (e.g., `User`, `Ride`, `Admin`).

2. **Responsibilities** – What the class is responsible for doing (its behavior or functionality).

3. **Collaborators** – Other classes it interacts with to fulfill its responsibilities.

Purposes:

- **Object-Oriented Thinking**: Encourages thinking in terms of real-world objects and their interactions.

- **Simplified Design**: Focuses on behavior and interaction, not just data.

- **Collaboration-Friendly**: Easy for teams to brainstorm, especially during early-stage design.

- **Avoids Overloaded Classes**: Promotes the Single Responsibility Principle (SRP).

- **Foundation for UML**: Helps prepare for more formal diagrams like Class or Sequence Diagrams.

CRC Diagram: CampusRide

User

Responsibilities:

- Register/Login
- Manage Profile
- Request Ride
- Share Ride'ĸ Reduce Fare
- Rate Driver/Pessenger
- Chat with Driver/Pas-
- Submit Complaint

Collaborators:

- Ride
- Chat System
- Dashboard (Der)

Ride

Responsibilities

- Create/Request Ride
- Set Source, Destination. Time, Seats
- Match Ride for Sharing
- Estimate Fare

Collaborators

- User
- Driver

Rating System

Responsibilities

- Allow Ratings/Reviews for Rides
- Display Average Ratings

Collaborators

- User

Driver

Responsibilities

- Offer Ride
- Manage Vehicle Info
- Accep1/ Reject Ride Requests
- View Ride History
- Estimate Fare

Collaborators

- User
- Driver
- Fare-Estimation

Ride Sharing System

Responsibilities

- Enable Multiple Users for Same Route

Collaborators

- Ride
- User

Fare Estimation

Responsibilities

- Allow Ratings/Reiews for Rides
- Display Average Ratings

Collaborators

- Ride

Admin

Responsibilities

- Monifor Complaints
- Manage Users/Drivers
- Resolve Disputes
- Block(Allow Users
- View Reports

Collaborators

- Complaint System
- Dashboard (Admin)
- User

Chat System

Responsibilities

- Enable Chat Between Drivers and Passengers
- Save Conversation Logs

Collaborators

- User
- Driver

Dashboard (User / Driver / Admin)

Responsibilities

- Display Parsonalized Ride Infoj
- Access Notifications
- Manage Profile and Settings

Collaborators

- User
- Driver
- Admin

5.9 Sequence Diagram:

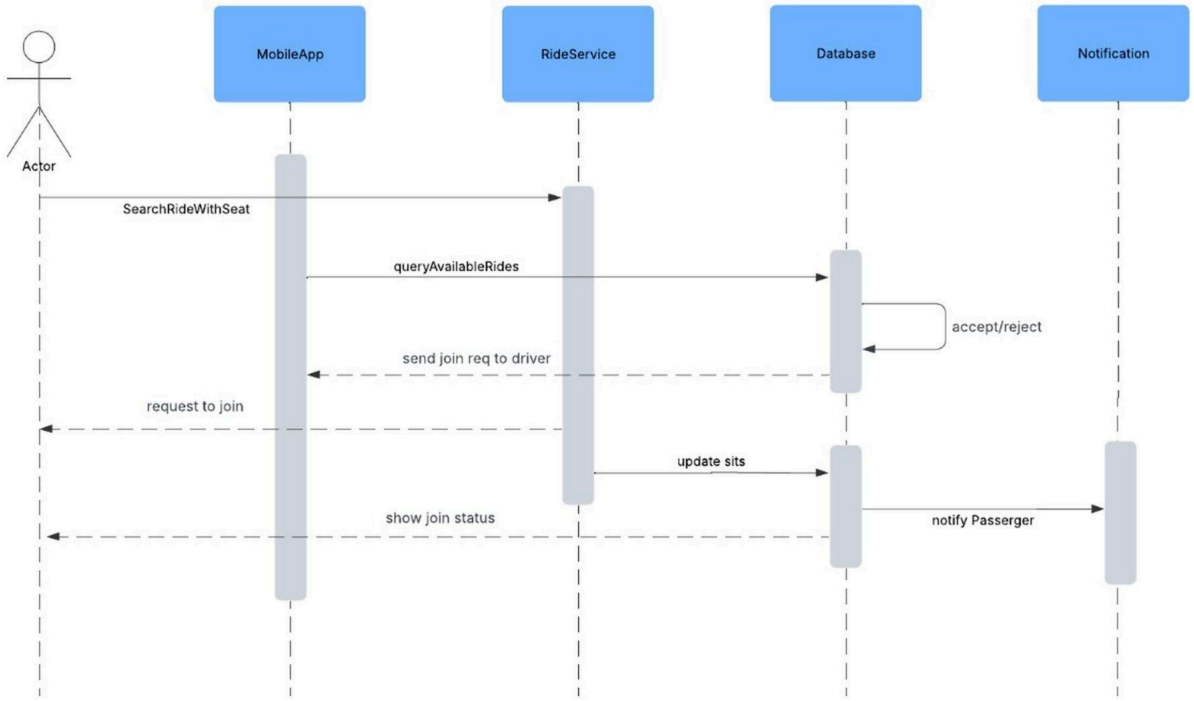
A **Sequence Diagram** is a type of **UML (Unified Modeling Language)** diagram used to model the **interaction between objects in a time sequence**. It visually represents **how and in what order** different parts of a system interact with each other to accomplish a specific functionality.

Key Components:

1. **Actors/Objects**: Represented horizontally at the top (e.g., User, Ride System, Driver).
2. **Lifelines**: Vertical dashed lines that show the object's existence over time.
3. **Messages**: Horizontal arrows between lifelines representing the flow of communication (e.g., method calls, responses).
4. **Activation Bars**: Thin rectangles on lifelines showing when an object is performing an action.

Purpose:

- To describe **dynamic behavior** of the system.
- To detail **how functionalities are executed step by step**.
- To illustrate **real-time interactions** between objects.



5.10 Class Diagram:

A **class diagram** is a type of **static structure diagram** in the Unified Modeling Language (UML) that describes the **structure of a system** by showing its:

- **Classes**
- **Attributes**
- **Methods (Operations)**

- **Relationships** (associations, inheritance, etc.)

It serves as a **blueprint** for how software is built — similar to architectural drawings in construction.

Purpose of Class Diagram in CampuRide:

Purpose	How It Applies to Your Project
✅ **System Structure Visualization**	Shows how core components like `User`, `Driver`, `Passenger`, `Ride`, `RideRequest`, `Payment`, etc., are related and interact.
✅ **Defines Roles and Responsibilities**	Clarifies what each class (e.g., `Admin`, `Driver`, `Passenger`) is responsible for — like `acceptRide()` or `giveFeedback()`.
✅ **Supports Database Design**	Helps map out how data should be stored (e.g., one-to-many relations between `Passenger` and `RideRequest`).
✅ **Enables Code Generation**	Can serve as a basis for generating boilerplate code in object-oriented programming languages.
✅ **Improves Communication**	Makes it easier for developers, designers, and stakeholders to understand the system quickly.↑
✅ **Assists in Maintenance**	When updating the system, this diagram helps ensure changes don't break existing functionality.

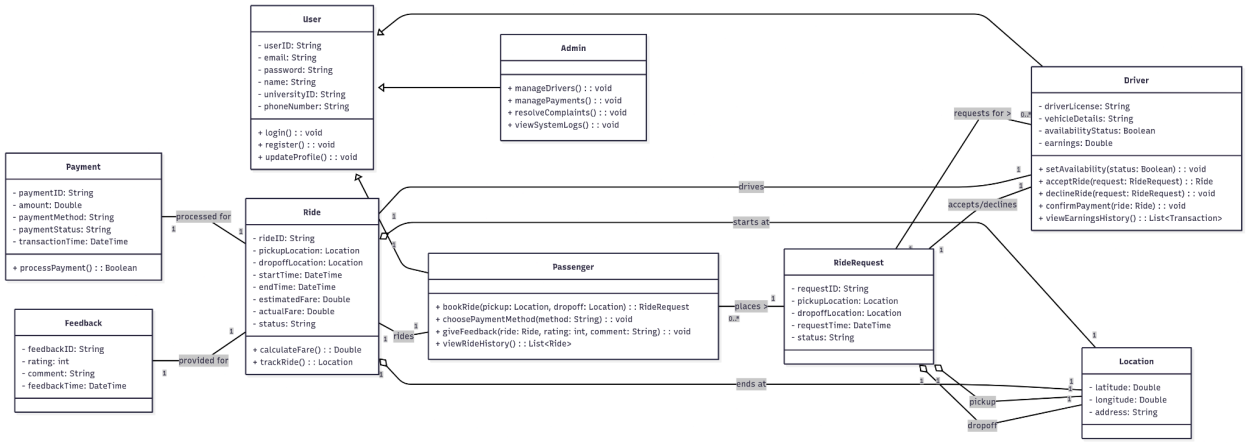
Why It's Important for CampusRide

1. **Modularity**: Classes like `Ride`, `Location`, and `Payment` encapsulate specific data and behavior, promoting reusability and scalability.

2. **Security**: Separates user roles (Admin, Driver, Passenger), enforcing different permissions and access levels.

3. **Extendibility**: Easy to add new features (e.g., promo codes, ride-sharing between multiple passengers) without redesigning the entire system.

4. **Error Prevention**: Prevents design flaws early in development by clearly laying out dependencies and relationships.



5.11 Deployment Diagram:

A **Deployment Diagram** in UML (Unified Modeling Language) models the **physical deployment** of artifacts (like software components) on **hardware nodes** (like servers, devices). It shows how and where the software runs in the real world.

Key Elements:

- **Nodes**: Physical hardware or devices (e.g., mobile phone, server)
- **Artifacts**: Software components (e.g., databases, web apps, APIs)
- **Communication Paths**: How nodes are connected and communicate

---

Purpose of Deployment Diagram

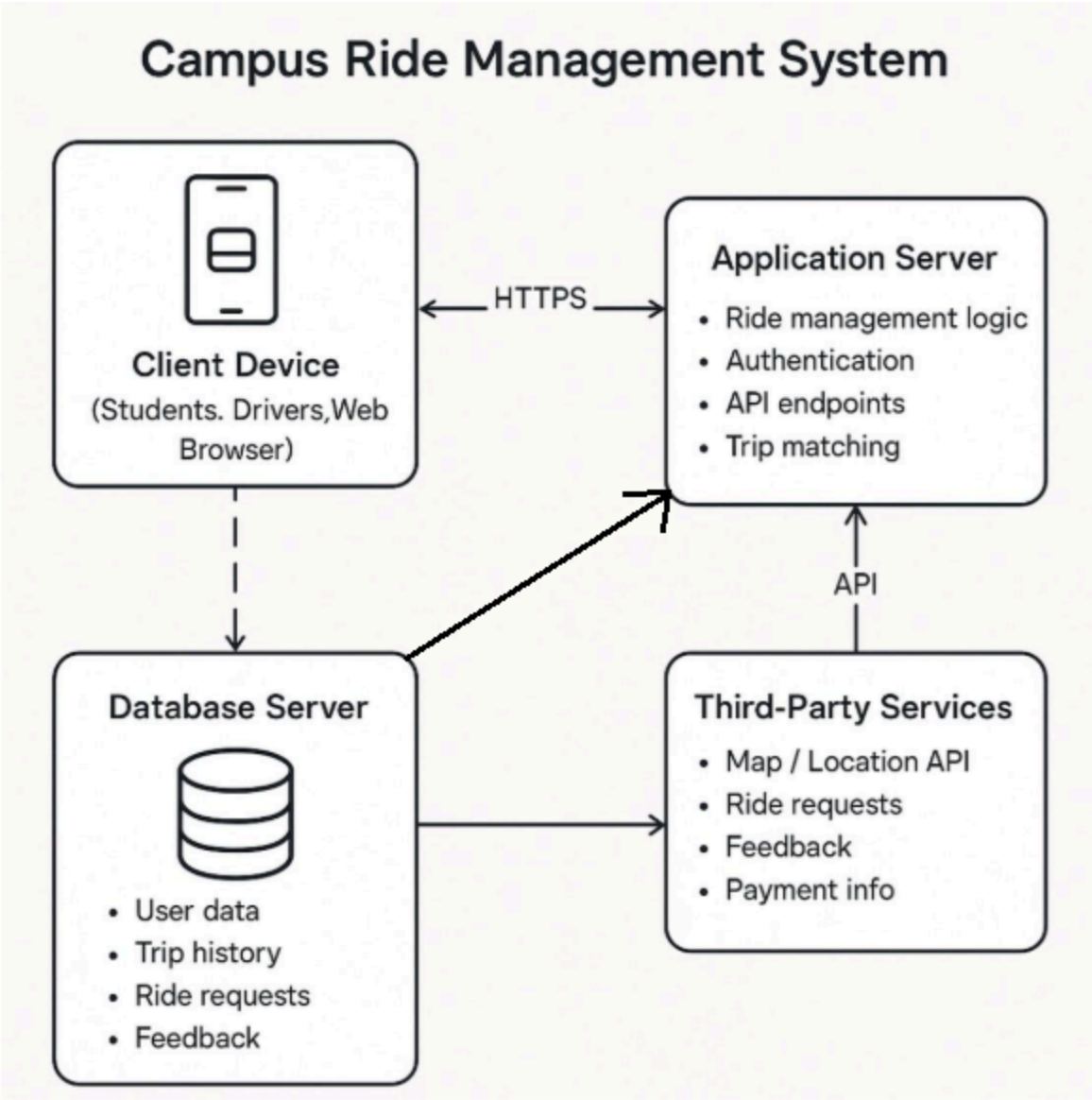
Purpose	Description
✅ Visualizes System Architecture	Helps stakeholders see where software components are deployed.
✅ Defines Hardware Requirements	Shows what infrastructure (servers, devices) is needed.
✅ Aids in Network Planning	Illustrates communication paths and bandwidth needs.
✅ Supports Security Planning	Identifies data flow and potential security vulnerabilities.
✅ Helps Troubleshooting	Simplifies diagnosing system issues by showing component distribution.

Deployment Diagram Purpose in the CampusRide Project

Project Context:

CampusRide is a ride-sharing platform for university students, likely involving:

- A **mobile app** for students/drivers
- A **web-based admin panel**
- A **backend server**
- A **database**
- Possibly integration with **payment gateways** and **map services**



(Graph of Deployment Diagram)

5.12 State Diagram:

A **State Diagram** is a type of UML (Unified Modeling Language) diagram that describes the dynamic behavior of a single object within a system in response to various events. It is also referred to as a Harel State Chart or State Machine Diagram. This diagram illustrates the transitions between different states of an object as it interacts with events and responses during its lifecycle. It models how an object behaves at different points in time in relation to the occurrences within the system, making it an essential tool for capturing the behavior of objects.

A **Behavioral State Machine Diagram** focuses on the states that a specific instance of a class passes through, showing how the object reacts to events, performs actions, and responds to changes during its life.

Key Definitions:

- **State**: A state represents the condition of an object at a specific point in time. It describes the situation where an object either satisfies a particular condition, performs an action, or waits for an event to occur.

- **Event**: An event is an occurrence that takes place at a specific moment, causing a change in the object's state. This change reflects the impact of the event on the object's values or attributes.

Basic Components of a State Diagram:

1. **Initial State**: Represented by a filled black circle. It indicates the starting point of the state machine, where the object begins its life cycle.

2. **Transition**: Represented by a solid arrow. A transition signifies the change of control from one state to another. The arrow is labeled with the event that triggers this transition.

3. **State**: Depicted as a rounded rectangle. A state represents an object's condition at a specific time, capturing its values or situation during that point in its lifecycle.

4. **Fork**: Represented by a solid rounded rectangular bar. It indicates the splitting of a state into two or more concurrent states, allowing multiple activities to occur in parallel.

5. **Join**: Represented by a rounded solid rectangular bar. It signifies the convergence of two or more states into one. This notation occurs when multiple parallel states combine into a single state upon the occurrence of one or more events.

6. **Self Transition**: Depicted as a solid arrow pointing back to the state itself. This notation is used when an event does not cause a change in the state of the object but keeps it in the same state.

7. **Composite State**: Depicted as a rounded rectangle. It represents a state that contains internal activities or sub-states, showing that a state consists of multiple smaller, nested states.

8. **Final State**: Represented by a filled circle within another circle. It marks the termination point in a state machine diagram, indicating that the object has reached its final state.

Steps to Draw a State Diagram:

1. **Identify the Initial and Final States**: Clearly define where the object begins (initial state) and where it ends (final state).

2. **Identify the Possible States**: List all potential states the object may enter. These are determined by the attributes and conditions of the object at different points in its lifecycle.

3. **Label the Events**: Identify and label the events that trigger state transitions. These events define when and why an object moves from one state to another.

References:

- SmartDraw: State Diagram Overview
- *System Analysis and Design*, 5th Edition by Alan Dennis, Barbara Haley Wixom, Roberta M. Roth
- GeeksforGeeks: UML State Diagrams
- Wikipedia: State Diagram

5.13 UI Design:

A **User Interface (UI)** is the part of a system that enables users to interact with it. The UI design focuses on how external entities (users, other systems, or devices) engage with the system, including how inputs are provided and how outputs are received. It ensures that users can interact with the system in a way that is efficient, effective, and intuitive.

Three Golden Rules of User Interface Design:

1. **Place Users in Control:**

- **Modeless**: The system should not impose unnecessary modes that force users to work in restricted ways.
- **Flexibility**: Allow users to customize and adapt the system to their needs.
- **Interruptible**: Let users stop or change tasks without penalty.
- **Helpful**: Provide useful guidance or feedback to assist users.
- **Forgiving**: Errors should be easily corrected, and users should not feel penalized for mistakes.
- **Navigable**: Ensure that users can find their way easily through the system.
- **Accessible**: Ensure that all users, including those with disabilities, can interact with the system.
- **Facilitative**: Make the interface facilitate smooth interaction and decision-making.
- **Preferences**: Allow users to personalize the system according to their preferences.
- **Interactive**: Encourage user interaction in a clear and engaging manner.

2. **Reduce Users' Memory Load:**
    - **Relieve Short-Term Memory**: Avoid requiring users to remember complex sequences of actions.
    - **Rely on Recognition, Not Recall**: Provide options and visual cues that users can recognize rather than relying on their memory.
    - **Provide Visual Cues**: Help users understand what actions are possible with visual hints.
    - **Forgiving**: Let users undo mistakes easily.
    - **Frequency**: Frequently used actions should be easy to access.
    - **Promote Object-Action Syntax**: Use clear and consistent labels for actions and objects.

- **Use Real-World Metaphors**: Use familiar symbols and concepts to make interactions more intuitive.

- **User Progressive Disclosure**: Show only relevant information at each stage to avoid overwhelming the user.

- **Organize**: Group similar items together to facilitate easier navigation and decision-making.

3. **Make the Interface Consistent:**

    - **Continuity**: Maintain consistency across different parts of the system.

    - **Consistency Across Products**: Use the same terminologies, symbols, and workflows in different systems or versions.

    - **Interaction Results**: Ensure that the system responds predictably to actions.

    - **Aesthetic Appeal**: The interface should be visually pleasing, contributing to an overall sense of quality.

    - **Encourage Exploration**: The interface should encourage users to explore features without fear of making mistakes.

Shneiderman's Eight Golden Rules (1987):

1. **Strive for Consistency**: Consistent actions and terminology should be used throughout the system to avoid confusion.

2. **Seek Universal Usability**: Design for a wide range of users, including both novices and experts, and allow for customization.

3. **Offer Informative Feedback**: Provide users with appropriate feedback for both minor and major actions.

4. **Design Dialogs to Yield Closure**: Provide feedback when a group of actions is completed, giving users a sense of accomplishment.

5. **Prevent Errors**: Design the system to minimize the risk of serious errors and provide easy recovery options.

6. **Permit Easy Reversal of Actions**: Allow users to easily undo actions to alleviate stress and encourage experimentation.

7. **Keep Users in Control**: Avoid surprising users with unexpected changes or behavior and provide them with options to control the system.

8. **Reduce Short-Term Memory Load**: Minimize the need for users to remember information across multiple displays.

Norman's Design Principles (1988):

1. **Visibility**: Ensure that all available options are clearly visible to users so they know what actions are possible.

2. **Feedback**: Provide feedback on what actions have been completed and the current state of the system, ensuring users know the system's response.

3. **Affordance**: Objects in the interface should visually indicate how they can be used (e.g., buttons should look clickable).

4. **Mapping**: The relationship between controls and their effects should be intuitive (e.g., a volume slider should increase or decrease volume).

5. **Constraints**: Restrict certain actions to prevent errors or confusion (e.g., graying out unavailable options).

6. **Consistency**: The interface should behave in a consistent manner, using familiar conventions and patterns.

References:

- Golden Rules of User Interface Design
- Mandel Golden Rules
- Shneiderman's Eight Golden Rules
- Norman's Design Principles
- Norman's Design Principles on Educative.io

CampusRide

Home Ride Drive About Us Log in Sign up

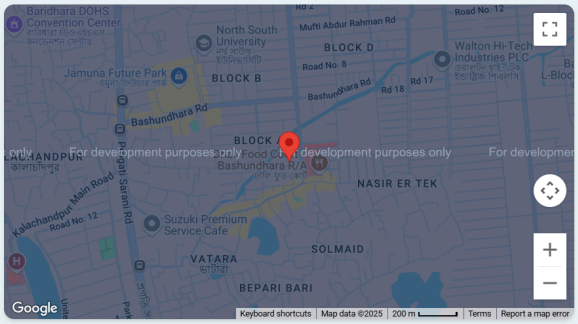
Go anywhere with CampusRide

Pick-up Location

Drop-off Location

Ride

See more options



Rentals

Reserve

Share your Ride

Rentals

Book rentals for your journey

See more

Reserve

Schedule your ride in advance

Book Now

Share your Ride

Save cost by carpooling

Share Now

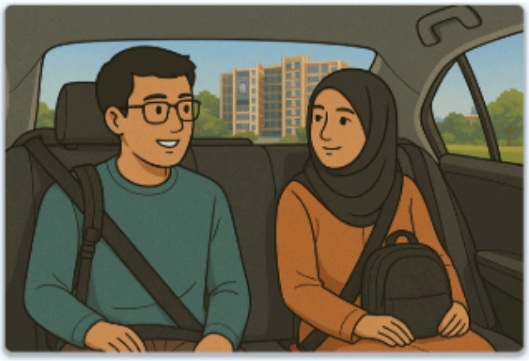
Log in to see your recent activity

View your past rides, saved locations, and booking history.

Login to CampusRide

We reimagine the way the world moves for the better

Movement is what we are. It's our lifeblood. It runs through our veins. It's what gets us out of bed each morning. It pushes us to constantly reimagine how we can move better. For all the times you want to go from here to there. All the times you want to earn. Across the entire world. In real time. At the incredible speed of now.

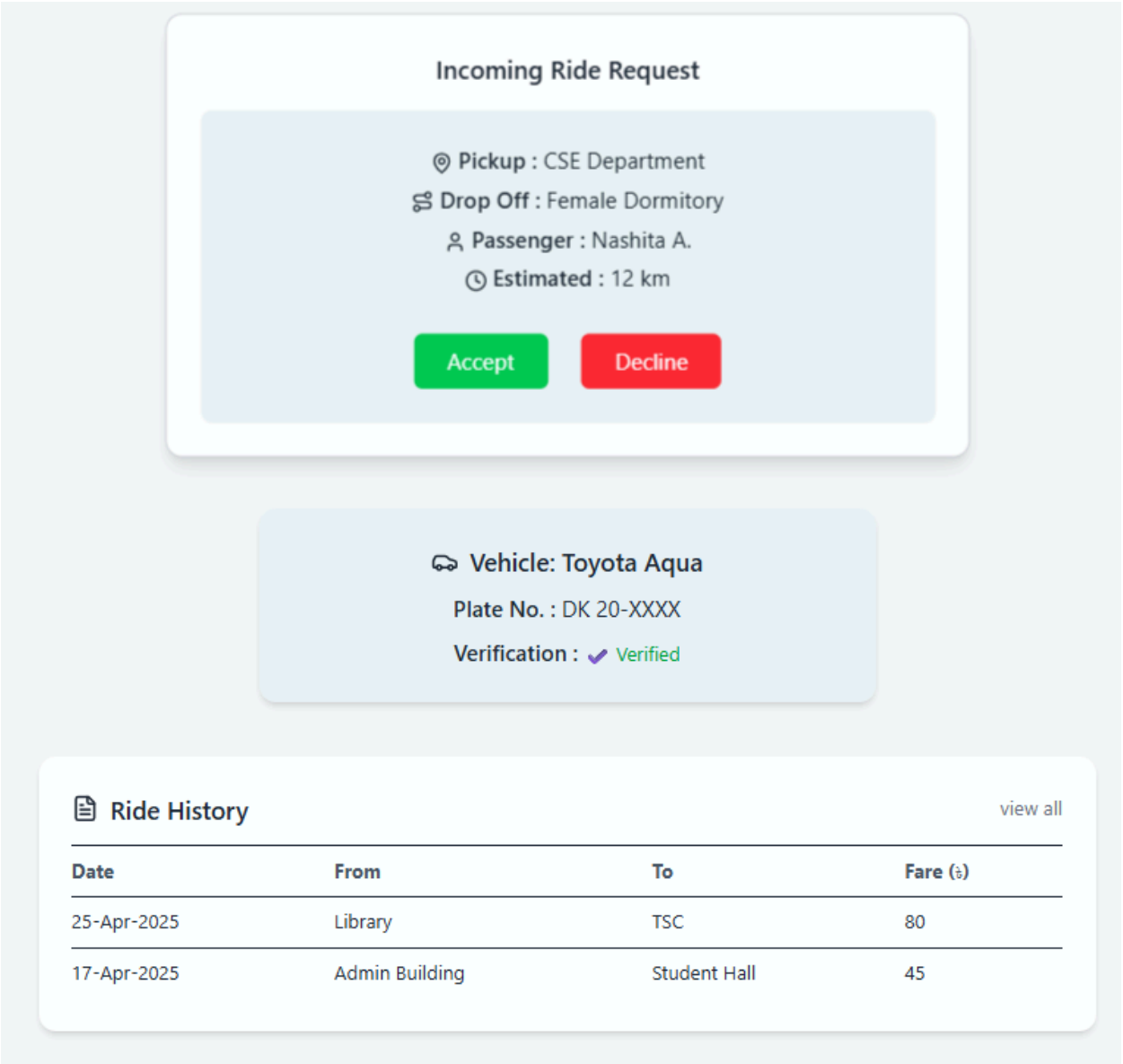


Share your Ride

Connecting millions of riders, drivers, and couriers. At anytime, anywhere, with just a few taps. Our technology and make transportation more accessible. We share your ride, and everyone wins from sharing together.

Your safety drives us

Whether you're in the back seat or behind the wheel, your safety is essential. We are committed to doing our part, and technology is at the heart of our approach. We partner with safety advocates and develop new technologies and policies to help improve safety and help make it easier for everyone to get around.


Incoming Ride Request
Pickup : CSE Department
Drop Off : Female Dormitory
Passenger : Nashita A.
Estimated : 12 km
Accept
Decline
Vehicle: Toyota Aqua
Plate No. : DK 20-XXXX
Verification : Verified
Ride History
view all
Date
From
To
Fare (৳)
25-Apr-2025
Library
TSC
80
17-Apr-2025
Admin Building
Student Hall
45

References

1. Uber Technologies Inc. – Platform features and safety measures. Retrieved from: https://www.uber.com
2. Pathao Limited – Local ride-sharing service with mobile integration. Retrieved from: https://www.pathao.com
3. Shohoz Limited – Transportation and ticketing platform in Bangladesh. Retrieved from: https://www.shohoz.com
4. Obhai Solutions Ltd. – Ride-sharing service with community safety focus. Retrieved from: https://www.obhai.com
5. inDrive – Ride-hailing app with price negotiation features. Retrieved from: https://www.indrive.com
6. Google Forms – Used for conducting user surveys. Retrieved from: https://forms.google.com
7. Snagit by TechSmith – Tool for competitive UI analysis. Retrieved from: https://www.techsmith.com/screen-capture.html
8. ResearchGate – Platform for accessing academic articles. Retrieved from: https://www.researchgate.net
9. "A Ride Sharing Service for University Community" – Research article on university-focused ride-sharing platforms.
10. "A Study on Motor Ride Sharing Service in Dhaka City, Present and Future Challenges" – Research paper analyzing ride-sharing challenges in Dhaka.
11. Google Workspace – Used for team collaboration and document sharing. Retrieved from: https://workspace.google.com