JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

GeoSpatial: A Coordinate-Based Campus Ride Sharing System with Real-Time Route Matching, Face Recognition Login and Safety Alerts for University Students

Vennela Lavanya
M.Tech Student
Department of Information Technology and Computer Applications
Andhra University College of Engineering (A)
Visakhapatnam, Andhra Pradesh, India

Abstract : Urban traffic congestion creates a daily transportation challenge for students who travel to academic institutions from different parts of a city. Campus Ride is a campus-exclusive, web-based ride-sharing platform designed for verified students of the same institution. The system improves conventional campus ride-sharing by replacing destination-name matching with route-aware spatial matching and by integrating face-based login verification, route transparency, mutual profile visibility, emergency-contact notification, manual waypoint support, and flexible payment options. When a rider posts a ride, the selected route is stored as a LineString geometry in PostgreSQL/PostGIS. A passenger pickup location is evaluated using a 500-meter route-proximity threshold and an additional pickup-detour constraint of less than 2 km. Identity verification uses a student ID photograph and a live selfie, with 128-dimensional face encodings generated through the face_recognition library built on Dlib. Leaflet.js and OpenStreetMap provide map visualization, while OSRM supplies road routes. The prototype architecture uses Python, Django/DRF, React.js, PostgreSQL/PostGIS, Redis, Socket.io, emergency messaging services, and UPI deep links. The design deliberately avoids continuous live tracking; instead, an emergency notification is sent when the ride starts, providing essential ride information to the registered emergency contact. The proposed architecture aims to improve matching accuracy, student safety, route transparency, affordability, vehicle-seat utilization, and environmental sustainability while using open-source or free-tier technologies.

IndexTerms **- Campus Ride, Ride Sharing, Carpooling, Route Overlap Matching, PostGIS, Haversine Formula, Dijkstra Algorithm, OSRM, Face Recognition, Dlib, Student Security, UPI Payment, OpenStreetMap.**

I. INTRODUCTION

1.1 Background and Motivation

Students frequently commute to academic institutions at fixed or semi-fixed times from different areas of a city. In Visakhapatnam, inadequate public transport capacity, overcrowding, and increasing private vehicle use can make daily travel inconvenient and expensive. Private vehicles also commonly contain unused seats. The base-paper survey cited in the project reports that approximately 62.5% of seats in private vehicles remained empty, while 92% of students were willing to share rides with fellow students when suitable security and verification were provided.

These observations motivate a platform designed specifically for an academic community rather than a general commercial taxi service. Campus Ride focuses on students belonging to the same institution, making institutional verification, route visibility, identity confirmation, and safety communication central design requirements.

1.2 Problem Statement

- Lack of adequate intra-city public transportation relative to the growing student population.
- Traffic congestion is partly caused by underutilized private vehicles and unused seats.
- High daily cost of private taxis and autos for students.
- Safety concerns for students travelling with unfamiliar people, particularly for female students.
- Lack of a campus-specific platform with continuing student identity verification.
- General ride-sharing applications may not show the actual road route before passengers confirm a ride.
- Routing engines may not produce a student's preferred daily route, creating a need for manual route customization.

1.3 Objectives

- Design and implement a campus-exclusive ride-sharing web platform for verified students of the same institution.
- Implement route-aware passenger matching using PostGIS spatial queries and geographic distance calculations.
- Integrate face recognition identity verification using Dlib-based face encodings at every login.
- Provide route transparency through map visualization before ride confirmation.
- Enable mutual profile-photo visibility between the rider and passenger after acceptance and before the ride starts.
- Provide emergency contact SMS notifications without requiring continuous live tracking.

• Provide No Fare, Cash, and UPI payment options.
• Allow riders to add custom waypoints when OSRM does not produce their preferred daily route.

II. RELATED WORK

2.1 Base Paper

Chowdhury, Jamal, Alam, and Palit's "Campus Ride: An Environment-friendly Ride Sharing Platform for Academic Institutions," presented at IEEE CIT in 2016, is the base work identified in the dissertation. The original platform was intended for university students and included institutional email registration, advising-document upload, manual verification, ride posting, gender preference, public/private visibility, trusted circles, destination search, join requests, notifications, and star ratings. However, its matching was primarily based on destination names and proximity rather than actual road-path geometry, and it did not provide the proposed route-aware map experience or continuing face verification.

2.2 Related Research

The study reviews a 2024 IEEE campus shuttle ride-sharing application from Bells University, Nigeria. That work uses web/mobile technologies, GPS tracking, booking, and community features, but centers on fixed campus shuttle routes rather than flexible student-to-student carpooling. A 2025 IEEE campus carpooling study proposes a hybrid ridesharing algorithm using K-Means geographic grouping and a Genetic Algorithm for route optimization; however, it does not establish actual road-path overlap in the way the proposed PostGIS geometry approach does. PoliUniPool, a university carpooling system, emphasizes travel schedules and preferences but does not provide the proposed spatial route geometry, map transparency, or face-based verification.

2.3 Research Gap

Feature	Base Paper (2016)	Shuttle App (2024)	HRA Campus (2025)	Campus Ride
Route matching	Destination name	Fixed routes	Geographic proximity	PostGIS route geometry
Face verification	None	None	ID verification	Login + Dlib face comparison
Route transparency	None	Limited to fixed shuttle context	None	Map before confirmation
Mutual profile view	None	None	None	Both parties see verified profiles
Manual waypoints	None	None	None	Supported
Emergency safety	None	GPS tracking	Emergency contact	SMS + verified profile
Payment	Not implemented	Not central	Dynamic pricing	No Fare / Cash / UPI
Cost	Unknown	Technology dependent	Unknown	Open-source/free-tier prototype

III. METHODOLOGY

3.1 Overall Methodology

The methodology follows a complete ride lifecycle: verified registration, face-verified login, rider route creation, route locking and spatial storage, passenger search, route-overlap matching, passenger route inspection, join-request processing, mutual identity visibility, ride start with emergency notification, ride completion, payment confirmation, and optional rating. The central principle is that a passenger should be matched to a ride because the passenger's pickup location is near the actual road path selected by the rider, not merely because both users typed the same destination.

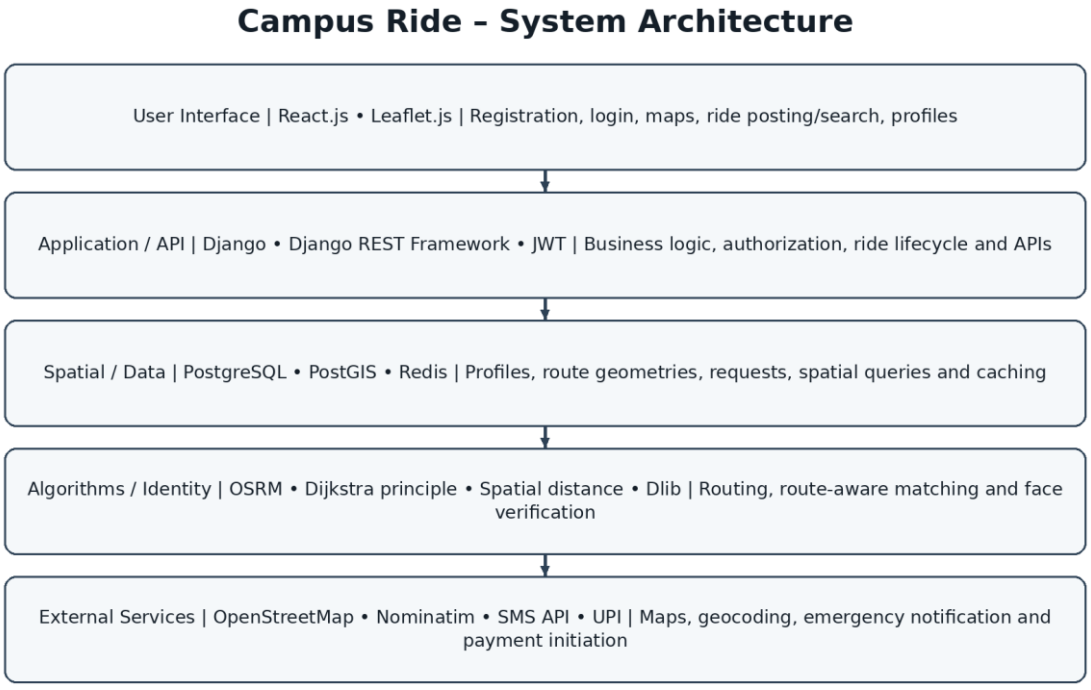

Fig. 1. Overall system architecture of the proposed Campus Ride platform.

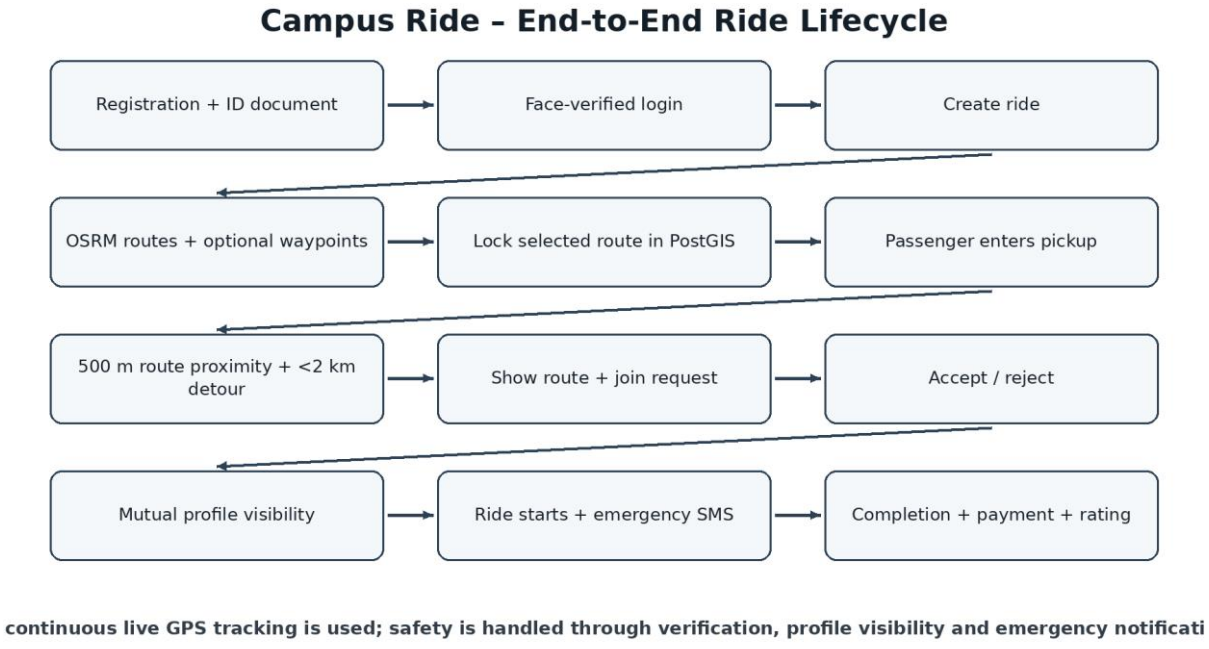

Fig. 2. End-to-end Campus Ride workflow from student registration to ride completion.

3.2 Dataset / Data Description

Campus Ride is an application and spatial-systems project rather than a machine-learning model trained on a benchmark text dataset. Therefore, an ArXiv/PubMed-style training dataset is not defined. The operational data consist of student profiles, verified identity documents, face encodings, home locations, ride records, route geometries, pickup coordinates, requests, ratings, notifications, circles, messages, and verification records. Geographic coordinates are obtained from map/address services, whereas road paths are produced by OSRM and stored as spatial LineString geometries in PostGIS.

Data Entity	Important Fields	Purpose
Users	user_id, email, face_encoding, home_location	Student profile and identity
Rides	ride_id, route_path, start, end, departure time, seats	Ride and locked road geometry
Ride Requests	request_id, pickup_location, status	Passenger requests and matching
Ratings	rating_id, score, comment	Post-ride feedback
Notifications	notif_id, type, message, is_read	Ride alerts
Circles	circle_id, user_id, friend_id, status	Trusted connections
Messages	message_id, ride_id, content	Ride communication
Verifications	verify_id, document_url, face_status	Student/document verification

3.3 Algorithms and Core Technical Components

The core technical components are PostGIS spatial distance matching, geographic distance calculation, OSRM road routing based on the Dijkstra shortest-path principle, and Dlib-based face encoding comparison. These components solve different problems: routing determines plausible road paths; spatial matching determines whether a passenger is close enough to a rider's selected path; face verification establishes identity continuity; and the application workflow controls when information becomes visible.

3.4 Route Overlap Matching

Suppose a rider travels from Gajuwaka to the CMR Complex. Multiple road paths can connect the same endpoints, for example through NAD Junction, Steel Plant Road, or Kommadi. A destination-only system can incorrectly treat all students heading to the CMR Complex as suitable matches. Campus Ride instead stores the rider's selected path as a spatial LineString and tests the passenger pickup point against that geometry.

The project specifies a 500-meter proximity condition and a maximum additional pickup detour of less than 2 kilometers. A simplified spatial query is: SELECT ride_id FROM rides WHERE ST_Distance(route_path, passenger_location) < 500 AND departure_time BETWEEN now() AND now() + interval '1 hour' AND available_seats > 0; A ride is considered only when the route passes sufficiently close to the passenger, the departure time is relevant, and a seat is available. The additional deviation check prevents a geometrically close point from creating an impractical pickup.

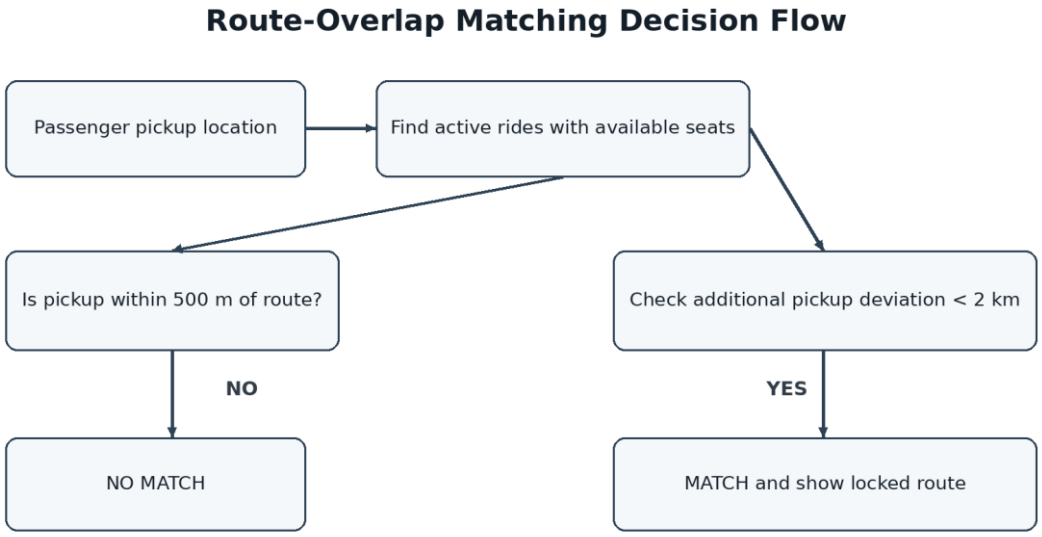

Fig. 3. Route-overlap matching decision flow using the spatial proximity and detour constraints.

3.5 Haversine Distance

The Haversine formula estimates the great-circle distance between two latitude/longitude coordinates on the Earth's curved surface. In Campus Ride, geographic distance is used as part of spatial matching logic, while the principal application-level rule is the 500-meter route-proximity threshold followed by the detour constraint. The standard form is $d = 2R \arcsin(\sqrt{(\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cos\varphi_2 \sin^2(\Delta\lambda/2)})$, where R is the Earth's radius and $\Delta\varphi$ and $\Delta\lambda$ are the latitude and longitude differences.

3.6 Dijkstra Algorithm through OSRM

OSRM represents the road network as a weighted graph in which road junctions behave as nodes and road segments behave as weighted edges. Dijkstra's shortest-path principle can be used to determine a minimum-cost route between locations. When the rider enters the start and destination locations, OSRM generates route choices. The rider can then select one route. If the desired daily route is missing, manual waypoints can be added and the route is recalculated through those points.

3.7 Face Verification with Dlib

At registration, the student's identity-document photograph is processed to obtain a 128-dimensional face encoding. During every login, the student captures a live selfie. The live face encoding is compared with the stored encoding. A successful comparison allows access, whereas a failed comparison rejects the login. This design is intended to reduce account sharing and impersonation while keeping the verification mechanism integrated with the student's campus account.

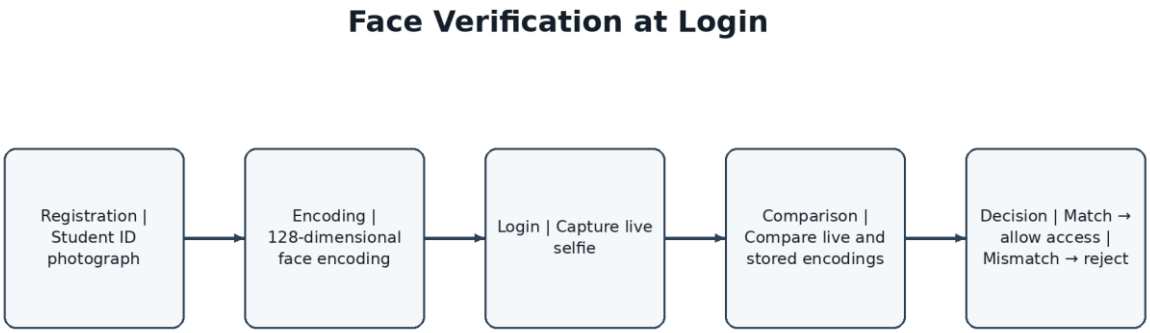

Fig. 4. Face-verification process using the stored student-ID photograph and live login selfie.

3.8 Route Selection and Locking

The conference template supplied with the project contains PEGASUS, LoRA, and Beam Search because its source paper is a scientific-text summarization project. These components are not part of Campus Ride and must not be falsely inserted into its methodology. Campus Ride has no text-generation model, no sequence-decoding stage, and no training-time beam-search requirement. The corresponding application-level decision stage is route selection: OSRM generates route candidates, the rider chooses one, optional manual waypoints are incorporated, and the selected route is locked for subsequent matching and transparency.

3.9 Proposed Framework

- Student registration and identity-document submission.
- Face-verified login.
- Ride creation and route generation.
- Manual waypoint customization when required.
- Route locking and PostGIS geometry storage.
- Passenger pickup search and spatial route-overlap filtering.
- Route transparency and join requests.
- Rider acceptance/rejection with instant notifications.
- Mutual verified-profile visibility.
- Ride start and emergency-contact SMS.
- Ride completion, payment confirmation, ratings, and history storage.

3.10 Evaluation Metrics

Because Campus Ride is not a text-summarization model, ROUGE and BERTScore from the supplied template are not appropriate primary metrics. The project should instead evaluate functional and system-level outcomes supported by its objectives. The following metrics are suitable for a completed prototype evaluation.

Metric	What It Measures	Campus Ride Interpretation
Route-match accuracy	Correct vs. incorrect route matches	Whether the spatial rule identifies genuinely usable rides
False-match rate	Incorrect matches returned	Whether destination-only errors are reduced
Spatial query time	Time to find candidate rides	Responsiveness of PostGIS matching
Route generation time	OSRM response time	Speed of producing route options
Face verification rate	Successful/failed identity checks	Reliability of login verification
SMS delivery success	Emergency message delivery	Safety notification reliability
Payment completion rate	Confirmed payment records	Reliability of payment workflow
User feedback	Perceived safety/usability	Practical acceptance of the platform

IV. IMPLEMENTATION DETAILS

4.1 Technologies and Environment Used

Category	Technology	Role in Campus Ride
Programming	Python 3.x	Backend integration and face-verification logic
Backend	Django + Django REST Framework	APIs, authentication support, ORM and administration
Frontend	React.js	Interactive pages and ride workflow
Map	Leaflet.js + OpenStreetMap	Map display and route visualization
Routing	OSRM	Road-route generation
Address search	Nominatim	Place-name to coordinate conversion
Database	PostgreSQL + PostGIS	Spatial and application data
Caching	Redis	Sessions/caching of frequent operations
Face recognition	face_recognition + Dlib	128-dimensional face encoding and comparison
Authentication	JWT	Stateless API authentication
Notifications	Socket.io	Instant ride-request/status notifications; not live tracking
Emergency	Twilio / WhatsApp API	Emergency-contact notification
Payment	UPI deep link	Open GPay/PhonePe with pre-filled payment information
Hosting	Railway / Render	Prototype deployment
Version control	GitHub	Source control and deployment workflow
API testing	Postman	Backend API testing

The prototype is described using open-source or free-tier technologies with zero infrastructure cost for the prototype configuration.

4.2 System Architecture

The architecture can be viewed as five cooperating layers. The presentation layer contains React.js and Leaflet.js. The application/API layer contains Django and Django REST Framework. The spatial/data layer contains PostgreSQL and PostGIS, with Redis supporting caching. External service integration includes OSRM, Nominatim, OpenStreetMap, SMS/WhatsApp services, and UPI deep links. The identity layer uses the face_recognition library built on Dlib.

4.3 Training Setup and Hyperparameters

The supplied JETIR template includes training epochs, learning rates, batch sizes, sequence lengths, and LoRA hyperparameters because PEGASUS is a machine-learning summarization model. Campus Ride does not contain a model-training pipeline in the dissertation. Accordingly, no learning rate, batch size, epoch count, LoRA rank, or GPU training result is invented or reported here. The relevant implementation parameters are system thresholds and configuration values.

Configuration	Value / Rule	Purpose
Route proximity threshold	500 m	Candidate route must pass close to passenger
Maximum pickup deviation	< 2 km	Avoid impractical detours
Face encoding	128 values	Represent registered/live face
Time filter	Relevant active departure window	Avoid stale rides
Seat filter	available_seats > 0	Return only rides with capacity
Routing customization	Manual waypoints supported	Represent actual daily routes

4.4 Application Workflow

Initially, a student registers with institutional details and submits the required identity document. At login, a live selfie is compared with the stored face representation. A rider then enters source and destination locations; Nominatim converts place names to coordinates and OSRM produces route choices. The rider can add waypoints when necessary and selects the route that represents the intended daily path. The selected path is stored as a LineString in PostGIS.

A passenger searches using a pickup location. The backend uses spatial filtering to identify rides whose locked routes fall within 500 meters of the pickup point, while also applying the less-than-2-kilometer pickup-detour rule, departure-time condition, and seat availability. The passenger can inspect the route before submitting a request. The rider accepts or rejects the request. After acceptance, mutual verified

profile information becomes visible. When the ride begins, an emergency notification is sent to the registered contact. After completion, the selected payment mode and confirmation are recorded, followed by optional rating.

4.5 Data and API Interaction

The React.js frontend communicates with Django REST APIs for registration, authentication, ride creation, ride search, request processing, notifications, profile access, and payment confirmation. PostgreSQL stores application records, while PostGIS stores and queries route geometries. Redis can support sessions or frequently accessed operations. Socket.io provides instant ride-request and status notifications; it is not used for continuous live location tracking.

V. RESULTS AND DISCUSSION

5.1 Quantitative Results

The supplied Campus Ride project document is a project design/prototype document and does not report a completed experimental results table with measured accuracy, latency, face-verification statistics, or user-study values. Therefore, numerical prototype results are not fabricated in this paper. The strongest quantitative values directly supported by the project are the 500-meter spatial matching threshold, the less-than-2-kilometer detour rule, the 128-value face representation, and the base-paper survey figures of 62.5% unused vehicle seats and 92% willingness to share when suitable security and verification are provided.

Supported Quantitative Item	Value	Meaning
Unused seats cited from base-paper survey	62.5%	Motivates better utilization of private vehicles
Students willing to share with security/verification	92%	Motivates campus-specific trusted ride sharing
Route proximity threshold	500 m	Spatial matching condition
Maximum additional pickup deviation	< 2 km	Practical detour constraint
Face representation	128-dimensional encoding	Identity comparison representation
Payment choices	3	No Fare, Cash, UPI

5.2 Graphs and Visualizations

A defensible final evaluation should visualize measured prototype results rather than reuse the PEGASUS paper's training-loss and ROUGE charts. Suitable Campus Ride visualizations include route-match accuracy comparing destination-name matching with PostGIS route matching; false-match cases before and after route-aware filtering; average spatial-query response time as ride records increase; OSRM route-generation response time; face-verification success/failure counts; emergency-SMS delivery success; and payment-option usage distribution. These graphs require actual measurements from the implemented prototype and should be added only after those measurements are collected.

5.3 Route-Matching Case Analysis

The project provides a Visakhapatnam route example in which a rider travels from Gajuwaka to the CMR Complex through NAD. NAD Junction and Gopalapatnam lie on or near the selected path and are therefore suitable matching candidates, whereas Steel Plant Area and Kommadi are several kilometres away from that locked path and should not be returned by the route-aware filter.

Passenger Pickup	Rider's Selected Route	Route Proximity	Expected Result
NAD Junction	Gajuwaka → NAD → CMR Complex	0 m / on path	MATCH
Gopalapatnam	Gajuwaka → NAD → CMR Complex	On/near path	MATCH
Steel Plant Area	Gajuwaka → NAD → CMR Complex	~4 km away	NO MATCH
Kommadi	Gajuwaka → NAD → CMR Complex	Off route	NO MATCH

5.4 Qualitative Results

Qualitatively, the proposed workflow improves passenger decision-making because the route is visible before confirmation. Mutual profile visibility adds another identity-check opportunity after a request has been accepted. Face verification strengthens the registration/login boundary, whereas emergency notification provides a safety communication mechanism without requiring continuous tracking.

The system also addresses a practical routing problem: a rider may know a daily route that a routing engine does not choose automatically. Manual waypoint addition allows user knowledge to be incorporated into the route that is ultimately locked and displayed to passengers.

5.5 Functional Test Cases

Functional testing is defined around valid and invalid input conditions. The following test categories are supported by the project document and should be executed on the deployed prototype.

Test Category	Test Input / Condition	Purpose
Positive	Clear valid application input	Verify intended workflow under normal conditions
Positive	Multiple supported functions/components	Verify combined workflow behavior
Positive	Valid detection / matching information	Verify successful processing
Positive	Complete valid workflow	Verify end-to-end operation
Negative	Unsupported input or condition	Verify graceful rejection/handling
Negative	Blank or invalid input	Verify validation
Negative	Low-visibility / unclear input	Examine behavior under difficult conditions

5.6 Discussion

The central improvement over the base system is the transition from name-based matching to geometry-aware matching. Destination-name matching can generate false positives because a common destination does not imply a common road path. PostGIS allows the application to reason about the actual stored route geometry and the passenger's pickup location.

The security design is layered. Institutional registration establishes the student context, face comparison checks identity at login, mutual profile visibility allows the two ride participants to inspect verified identity information, and emergency notification communicates essential ride information to the registered contact. The system intentionally avoids live tracking, which is consistent with the privacy-oriented design of the project.

The architecture also separates routing from matching. OSRM is responsible for producing road paths, while PostGIS is responsible for storing and querying the selected path. This separation makes it possible to replace or improve the routing service without redesigning the complete ride-matching database.

However, the current project document does not provide complete experimental measurements. Consequently, claims regarding measured accuracy, latency, user satisfaction, carbon reduction, or security effectiveness should be presented as expected outcomes until they are measured on the implemented prototype.

VI. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

Campus Ride proposes an enhanced campus-exclusive ride-sharing platform that combines institutional access, route-aware spatial matching, face verification, route transparency, mutual profile visibility, emergency notification, custom waypoints, and flexible payments. The design directly addresses the limitations identified in the base campus ride-sharing system. PostGIS route geometry provides a stronger basis for matching than destination-name comparison, while the 500-meter proximity and less-than-2-kilometer detour rules aim to keep matches practical.

The prototype architecture uses Python, Django/DRF, React.js, PostgreSQL/PostGIS, Leaflet/OpenStreetMap, OSRM, Nominatim, Redis, Dlib-based face recognition, Socket.io, emergency messaging, and UPI deep links. Therefore, the project is positioned as an integrated smart-campus transportation system rather than simply a ride-posting application.

6.2 Future Scope

- Conduct a real user study with students from the target institution and measure usability, perceived safety, and willingness to use the platform.
- Build a ground-truth route dataset to quantitatively compare destination-only matching with geometry-based matching.
- Measure spatial-query latency and scalability as the number of active rides increases.
- Improve route selection with more advanced multi-route and detour optimization.
- Add stronger liveness detection and privacy-preserving storage for face verification.
- Evaluate the reliability of emergency notification across different communication providers.
- Extend the architecture to multiple institutions with institution-specific verified student pools.
- Add richer analytics for environmental impact, vehicle utilization, and commuting cost savings.
- Study privacy-preserving alternatives to continuous location tracking while retaining safety benefits.
- Deploy the prototype in a controlled campus environment and collect long-term operational feedback.

REFERENCES

[1] A. Chowdhury, A. Jamal, R. Alam and R. Palit, “Campus Ride: An Environment-friendly Ride Sharing Platform for Academic Institutions,” IEEE International Conference on Computer and Information Technology (CIT), 2016, pp. 120–124.

[2] Q. B. Sodipo, A. W. Elegbede, C. D. Ajibola, K. O. Orunsolu and B. J. Adegboyega, “Development of Campus Ride-Sharing Application for Shuttle Rides,” IEEE 5th International Conference on Electro-Computing Technologies for Humanity (NIGERCON), 2024.

[3] Dharshini B. S., Logeshwaran E., Mohamed Aadhil A., “Campus Carpooling: Optimized Ridesharing for Students using Hybrid Ridesharing Algorithm HRA,” IEEE International Conference on Emerging Systems and Intelligent Computing (ESIC), 2025, pp. 449–454.

[4] M. Bruglieri et al., “PoliUniPool: A Carpooling System for Universities,” Procedia Social and Behavioral Sciences, vol. 20, pp. 558–567, 2011.

[5] PostGIS Official Documentation — Spatial and Geographic Objects for PostgreSQL.

[6] A. Geitgey, face_recognition Python Library built on Dlib.

[7] OpenStreetMap Foundation — OpenStreetMap.

[8] OSRM — Open Source Routing Machine.

[9] Google Maps Directions API Documentation.

[10] Django REST Framework Documentation.

[11] Leaflet.js — Interactive Maps for the Web.

[12] Nominatim — OpenStreetMap Geocoding Service.

[13] Redis Official Documentation.

[14] UPI Deep Link Specification — National Payments Corporation of India (NPCI).