APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 713-732
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Student E-Hailing Application with Multifactor Authentication

Fatnin Wafdi Auni[1], Zubaile Abdullah[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: *zubaile@uthm.edu.my*
DOI: https://doi.org/10.30880/aitcs.2025.06.02.039

Article Info

Received: 20 July 2025
Accepted: 19 November 2025
Available online: 30 November 2025

Keywords

Ride-hailing, Application, Multifactor Authentication

Abstract

Transportation is a vital aspect of university students' daily lives, but many face challenges due to the lack of personal vehicles. Existing e-hailing services often lack strong identity verification, posing cybersecurity risks such as impersonation and unauthorized access. The Student E-Hailing Application addresses these issues by ensuring only verified students can use the platform. Developed using Flutter and Firebase, it incorporates Multi-Factor Authentication (MFA) including passwords, student email verification and biometric authentication to enhance security. The application includes key modules such as registration, login, edit profile, book ride, confirm booking, rating and feedback, document verification, past bookings, and logout. Functional and security testing were conducted on all modules, with results confirming stable performance and resistance to common threats. Key findings indicate the system successfully creates a secure and trusted network for student transportation, mitigating risks associated with unverified users and enhancing safety through effective authentication mechanisms.

1. Introduction

Public transport systems, particularly buses, often fall short of meeting students' transportation needs due to limited routes, rigid schedules, and overcrowded conditions, making daily commuting inconvenient and inefficient. Additionally, existing e-hailing services such as Grab may not be budget-friendly for students and further limit access to affordable transportation options.

Security also remains a significant concern. Many universities rely on informal, student-run ride-sharing channels that lack a proper verification framework for drivers. This absence of identity validation introduces potential safety risks, as students cannot verify whether the individual's offering rides are affiliated with the university or trustworthy. Consequently, students may feel vulnerable when using these unregulated platforms.

To address these challenges, this project proposes the development of a Student E-Hailing Application specifically tailored for university students. The application aims to offer a cost-effective and secure transportation alternative by featuring verified student drivers, affordable fare calculations, and a user-friendly interface. A robust multi-factor authentication (MFA) system including student email verification, password and biometric authentication to ensure that all users are legitimate university members, thereby enhancing trust and safety within the campus community.

The project adopts the Agile development methodology to support iterative development, testing, and refinement. The application is developed using Flutter, a cross-platform framework, and Firebase, which handles backend services including authentication and real-time database management.

The study's objectives include designing a student-exclusive ride booking search system, developing a fully functional e-hailing platform, and evaluating it through iterative testing and user feedback. The project scope focuses on the university setting and university students, with Universiti Tun Hussein Onn Malaysia (UTHM) and its surrounding areas serving as the case study location.

The expected outcome of the project is a transportation solution that addresses both financial and safety challenges experienced by students. The system will deliver secure user verification, smooth ride-booking experience, and a trusted network for campus-based transportation. Furthermore, the application allows students to earn income as verified drivers. Beyond this initial implementation, the solution has the potential to be adapted for other educational institutions or community-based transport systems that require secure and affordable ride-hailing options.

2. Related Work

This section presents the literature review conducted for the Student E-Hailing Application, a mobile application platform designed to connect university students acting as passengers and drivers for ridesharing within campus areas and examines existing e-hailing platforms.

2.1 Authentication

Authentication is the process of verifying the identity of a user before granting access to a system. Traditional authentication methods rely on single-factor mechanisms such as passwords or personal identification numbers (PINs). However, these methods are increasingly vulnerable to security threats, including credential theft, brute-force attacks and phishing. As a result, the industry is shifting toward more secure, multi-factor authentication (MFA) approaches. MFA enhances security by requiring two or more independent credentials from different categories: something the user knows for example password, something the user has such as a email or phone number, or something the user is such as biometric fingerprint. Implementing MFA significantly reduces the risk of unauthorized access, even if one factor is compromised.

2.2 Password

Passwords are one of the most common methods of user authentication and remain a fundamental component of many digital security systems, including e-hailing applications. They are used to verify a user's identity by requiring the input of a secret known only to the user. Despite their widespread adoption, password-based authentication has notable limitations. Several studies have highlighted vulnerabilities associated with passwords, such as weak user-chosen passwords, reuse across multiple platforms and susceptibility to brute-force or phishing attacks. In response to these issues, modern applications often enforce password policies that include minimum length, complexity by implementing passwords must be more than 8 characters, has special character, number and capital letter.

2.3 Biometric Authentication

Biometric authentication is a security method that verifies a user's identity using unique characteristics such as fingerprints, facial features, voice patterns, or iris recognition. Unlike traditional methods that rely on knowledge or possessions, biometrics utilize unique user traits, making them difficult to forge or steal. In the context of e-hailing applications, biometric authentication enhances user security and convenience, allowing seamless login or verification without remembering credentials. Moreover, biometric systems can be integrated into multi-factor authentication frameworks to provide an extra layer of protection.

2.4 Comparing with Existing System

To evaluate the effectiveness of the proposed application, a comprehensive comparison was conducted with three existing similar applications. The analysis focused on security and authentication features to identify the strengths, limitations and unique aspects of each system. A summarized comparison of these features is presented in Table 1.

Table 1 *Comparison between three existing applications.*

Features/Application	Grab [1]	Airasia Move [2]	Bolt [3]

Email Verification	√	√	√
Password	×	√	√
Biometric Authentication	√	√	√
Student Verification	×	×	×
Android Based	√	√	√
IOS Based	√	√	√

A detailed comparison was conducted between similar existing applications and the proposed application to evaluate their security and authentication features. The findings indicate that all three applications support mobile number verification and email verification, which are essential for basic user authentication. In terms of password-based authentication, it is implemented by Airasia Move and Bolt, whereas Grab does not utilize this method. All three systems support both biometric authentication and multifactor authentication (MFA) verification. A key differentiator of the proposed application is its support for student verification, a feature not available in any of the existing systems compared. From a platform compatibility perspective, all applications are available on Android and iOS.

3. Methodology

The Agile development methodology will be used for the Student E-Hailing Application and explains the activities carried out in each phase of the process. Agile emphasizes collaboration, adaptability, continuous improvement and iterative development, enabling teams to deliver high-quality software more efficiently [5]. Fig. 1 shows the Agile model.

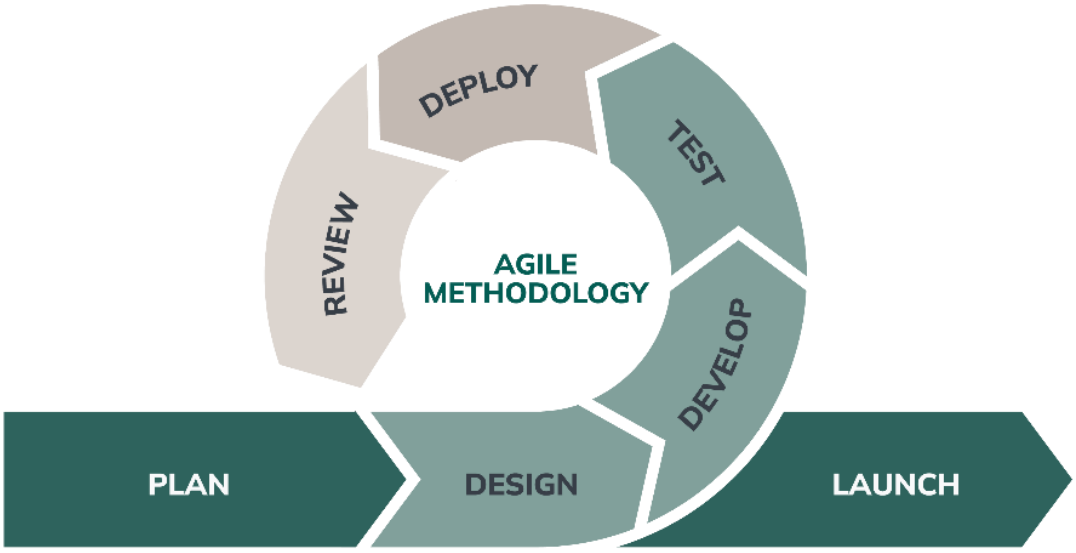


Fig.1 *Agile Methodology Model*

3.1 Planning Phase

In the planning phase, the foundation of the Student E-Hailing Application was established by clearly defining the scope, objectives, and expected deliverables. The project aimed to create a secure, student-focused ride-hailing platform, emphasizing affordability and safety. Key functionalities such as secure login, ride-booking, driver-passenger interaction, and multi-factor authentication were identified as priorities. A Gantt chart was developed to structure the timeline and manage milestones, and a comprehensive project proposal was written to guide the development strategy. During this stage, preliminary feedback was collected from university students to ensure that the application's features aligned with their actual needs and expectations for a safe and convenient transportation solution.

3.2 Design Phase

The design phase involved crafting the visual and structural design of the application. Wireframes and user interface layouts were created using Figma, including separate designs for driver and passenger workflows. System architecture diagrams were prepared to define how modules like confirm booking, rating and feedback, and driver document verification. The main focus is on incorporating biometric authentication and email verification for security. Flowcharts and data flow diagrams were drawn using Draw.io, and the database design was planned using Firebase's Realtime Database model.

3.3 Development Phase

In the development phase, the application was built using the Agile methodology, with development tasks divided into sprints. The frontend of the application was developed using Flutter, allowing the system to be compatible with both Android platforms. Firebase was chosen as the backend service, providing authentication, real-time database capabilities, and cloud storage. Each sprint focused on a key feature, such as setting up multi-factor authentication using password, email verification and biometric authentication, developing the user registration and login functions, enabling driver-specific features for ride acceptance, and implementing the ride booking system for passengers. The system was continuously tested and refined during development to ensure smooth integration between modules and adherence to functional and security requirements.

3.4 Testing Phase

The testing phase was important to determine the functionality, reliability, and security of the application. Unit testing was conducted for each module, including registration, login, and ride booking, to confirm they worked independently. Integration testing ensured that the modules communicated correctly, particularly between passengers and drivers during ride requests and updates. A group of university students participated in user acceptance testing to assess the system's usability, effectiveness, and overall user experience. Security testing focused on evaluating the robustness of the multi-factor authentication system by simulating wrong credentials during logins and unsuccessful biometric authentication attempts. The results of the testing phase led to necessary adjustments and improvements to enhance system performance and user satisfaction.

3.5 Deployment Phase

The deployment phase involved releasing the application for use by a selected group of university students in a controlled environment. The app was installed on Android devices and tested for compatibility across different screen sizes and devices. Firebase services were configured for production use, including real-time database access, storage, and authentication. This phase ensured the application was ready for use.

3.6 Review Phase

Following deployment, the review phase focused on collecting feedback from users to assess the application's effectiveness, usability, and areas for improvement. Surveys were used to gather insights from both passengers and drivers. Agile retrospectives were conducted to evaluate the development process and identify what worked well, what could be improved and overall performance of the application. These reflections were documented to inform future versions of the application, ensuring continuous improvement aligned with user expectations and technical best practices.

4. Analysis and Design

This section discusses the findings of the analysis and design phase of the development of Student E-Hailing Application.

4.1 System Requirement Analysis

The primary objective of the requirements analysis for the Student E-hailing Mobile Application is to ensure the system is designed to meet the specific needs of student users effectively. This involves understanding how users will interact with the application through its defined flow, from registration to booking rides. Table 2 and Table 3 shows the functional and non-functional requirements of the application respectively.

Table 2 *Functional Requirements of Application*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 713-732

Module	Functionalities	Entity
Registration	Allow users to create an account.	Driver, passenger, admin
Login	Allow users to log into existing account.	Driver, passenger, admin
Edit profile	Allow users to manage profile information.	Driver, passenger
Book ride	Allow users to book rides.	Passenger
Confirm booking	Allow users to confirm bookings.	Driver, passenger
Rating and feedback	Allow users to rate and give feedback.	Passenger
Document verification	Allow users to approve document.	Admin
Past bookings	Allow users to view past bookings.	Driver, passenger, admin
Logout	Allow users to log out of the system.	Driver, passenger, admin

Table 3 *Non-Functional Requirements of Application*

No.	Requirements	Description
1.	Operational	The application should be able to work on Android platform.
2.	Security	- Only users with valid student ID, student email and password can access own account. - The application could only be used by students of the university that has been verified during registration.
3.	Usability	User interface should be easy to navigate and understand.

User requirement analysis is the process of identifying and understanding the needs, expectations and goals of the end-users of a system. It focuses on determining what the users require from the system in terms of functionality, usability and performance to ensure the final product meets their needs. Table 4 shows the user requirement analysis of the application.

Table 4 *User Requirement Analysis*

User	User Requirements
Driver	- Register and login to the application. - Submit documents to be verified. - Turn on driver availability. - Accept or decline booking. - Track earnings. - Edit profile.
Passenger	- Register and login to the application. - Make or cancel bookings. - Track ride status. - View past trips. - Rate or complain. - Edit profile.
Admin	Able to access and review ratings, feedback and data related to passenger and driver.

Hardware and software requirement analysis involves identifying and defining the hardware and software resources necessary to support the development, deployment and operation of a system. Hardware requirement analysis focuses on determining the physical devices while software requirement analysis identifies the software tools required to build, run and maintain the system. The system's hardware and software requirements are displayed in Table 5 and Table 6, respectively.

Table 5 *Hardware Requirements*

No.	Hardware	Specification
1.	Computer	HP Pavillion Aero 13
2.	CPU	AMD Ryzen 5 5600U with Radeon Graphics 2.30 GHz
3.	RAM	8.00 GB

Table 6 *Software Requirements*

No.	Type	Software	Description
1.	Operating System	Windows 11 Home Single Language Version 23H2	The operating system that manages hardware and software resources.
2.	Programming Language	Dart	Language used to write code for the application.
3.	Programming Tools	Flutter, Android Studio	Tools to develop, write and edit code.
4.	Database	Firebase	Database to store and secure data within the app.
5.	Design Tools	Draw.io	Tool to create diagrams and flowcharts.

4.2 System Analysis

System analysis is a comprehensive process of studying, understanding, and evaluating how a system operates to ensure it effectively meets user needs and organizational objectives. Fig. 2 shows the driver use case where drivers can register and login for creating accounts and signing in, confirm booking to finalize reservations, upload document to submit required files, view past bookings to see booking history, check for ratings and driver availability to search for passengers and logout to end the session.

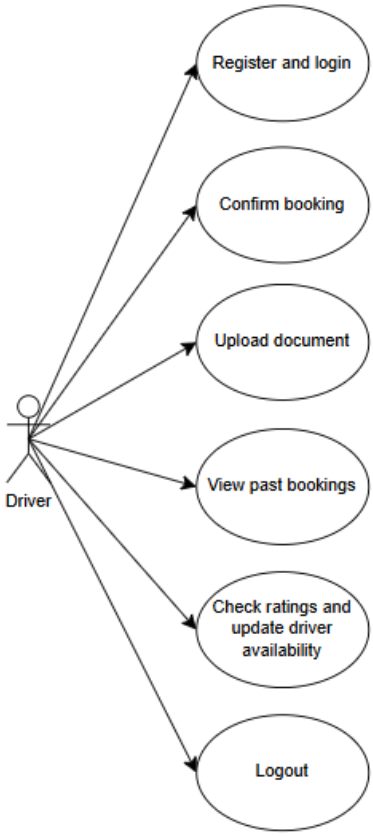


Fig.2 *Use Case Diagram for Driver*

Fig. 3 shows the Passenger use case diagram with six different modules. The system includes register and login for creating accounts and signing in, view past bookings to see booking history, book ride to make new

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 713-732

reservations, give rating and feedback to evaluate completed trips, edit profile to update personal information, and logout to end the session.

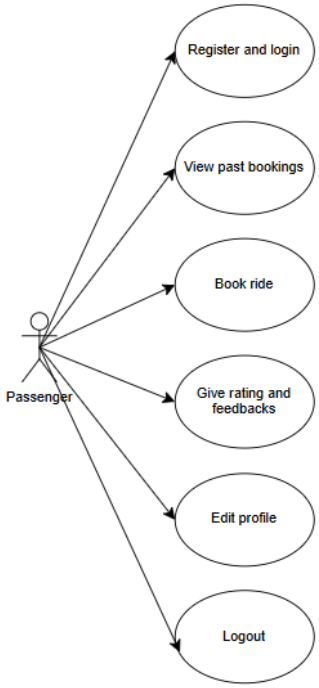


Fig.3 *Use Case Diagram for Passenger*

Fig. 4 shows Admin use case diagram that is connected to five different functions. The system includes login for accessing the admin account, view driver rating to check driver performance scores, view passenger ride feedback to see customer comments and reviews, approve or decline driver document to manage driver verification processes, and logout to end the admin session.

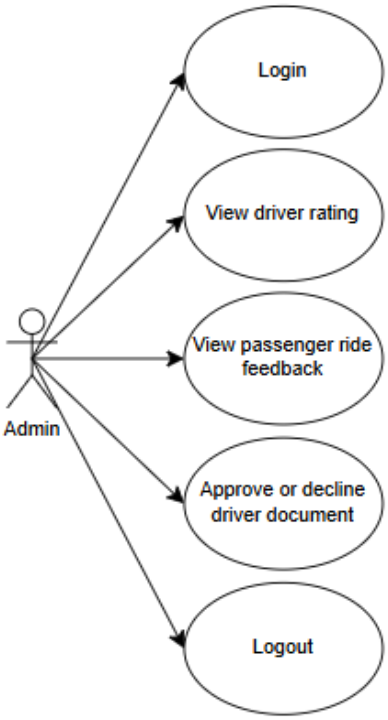


Fig.4 *Use Case Diagram for Admin*

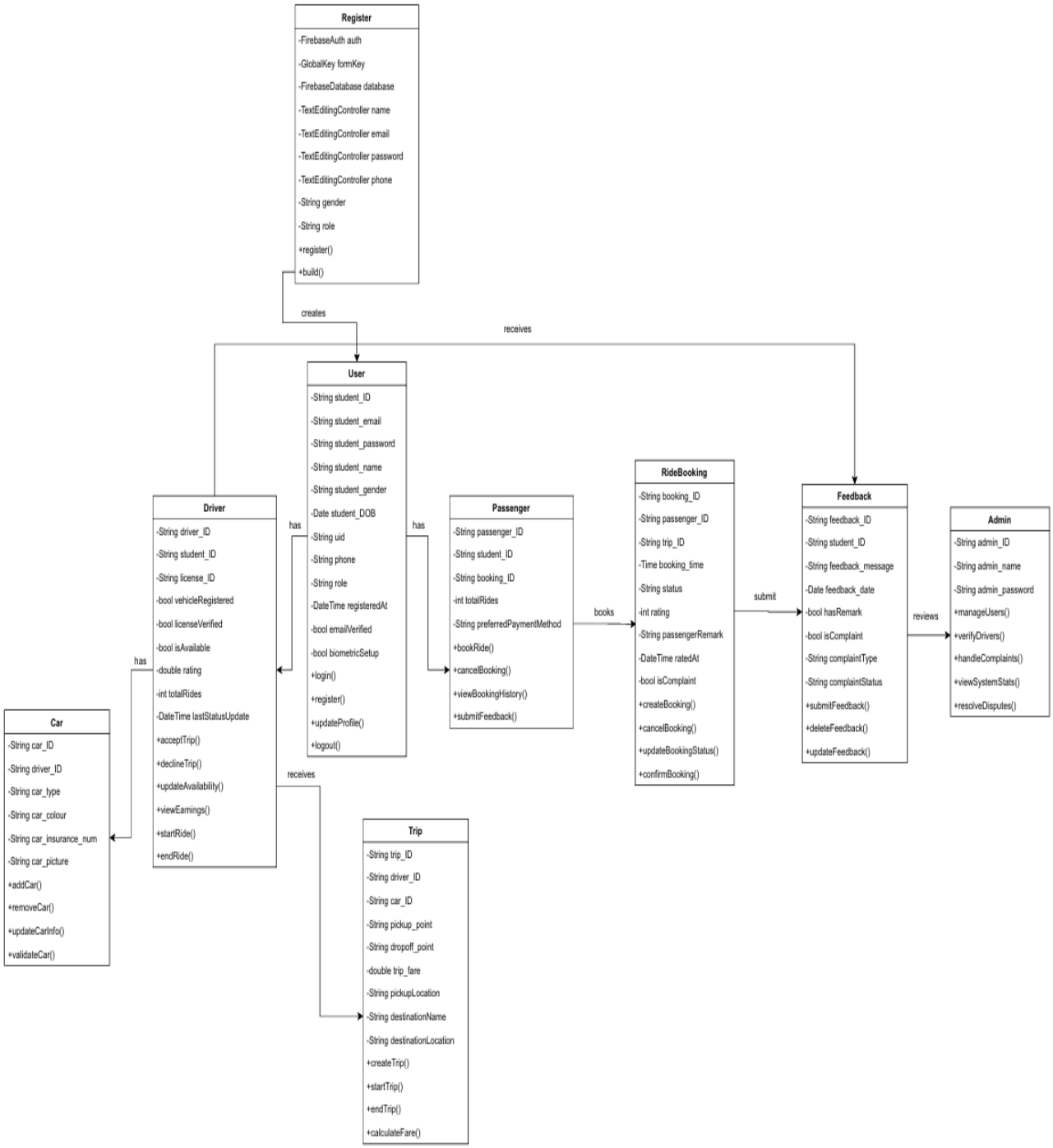


Fig.5 *Class Diagram of the Application*

The class diagram presented in Fig. 5 illustrates the comprehensive object-oriented structure of the Student E-Hailing Application, depicting the system's classes, their attributes, methods, and the relationships between different components. This diagram serves as a blueprint for the system's architecture, demonstrating how various entities interact within the application ecosystem. The foundational class that serves as the parent class for all system users. It contains essential attributes including student_ID (primary identifier), student_email, student_password, student_name, student_gender for basic user information. Additional attributes such as uid (unique identifier), phone, role, registeredAt, emailVerified, and biometricSetup support the application's advanced authentication and security features. The class provides fundamental methods for user management including login(), register(), updateProfile(), and logout().

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 713-732

The flowchart shown in Fig. 6 shows the full driver flow of the application including registration and verification process in the Student E-Hailing App. It starts with login, and once users are authenticated, they go to the homepage. A decision point lets users choose their next action and directs them accordingly. For driver registration, several processes happen at the same time to check the driver's qualifications. One process collects driver ratings and passenger feedback to help maintain service quality. Another process verifies driver documents, including licenses, vehicle registration, and insurance. The system also checks vehicle details and drivers can manage their profiles by updating personal information and account settings. Throughout the process, there are checkpoints to validate documents and confirm all required info is submitted. Admin approval is also part of the process to ensure driver is eligible to become driver.

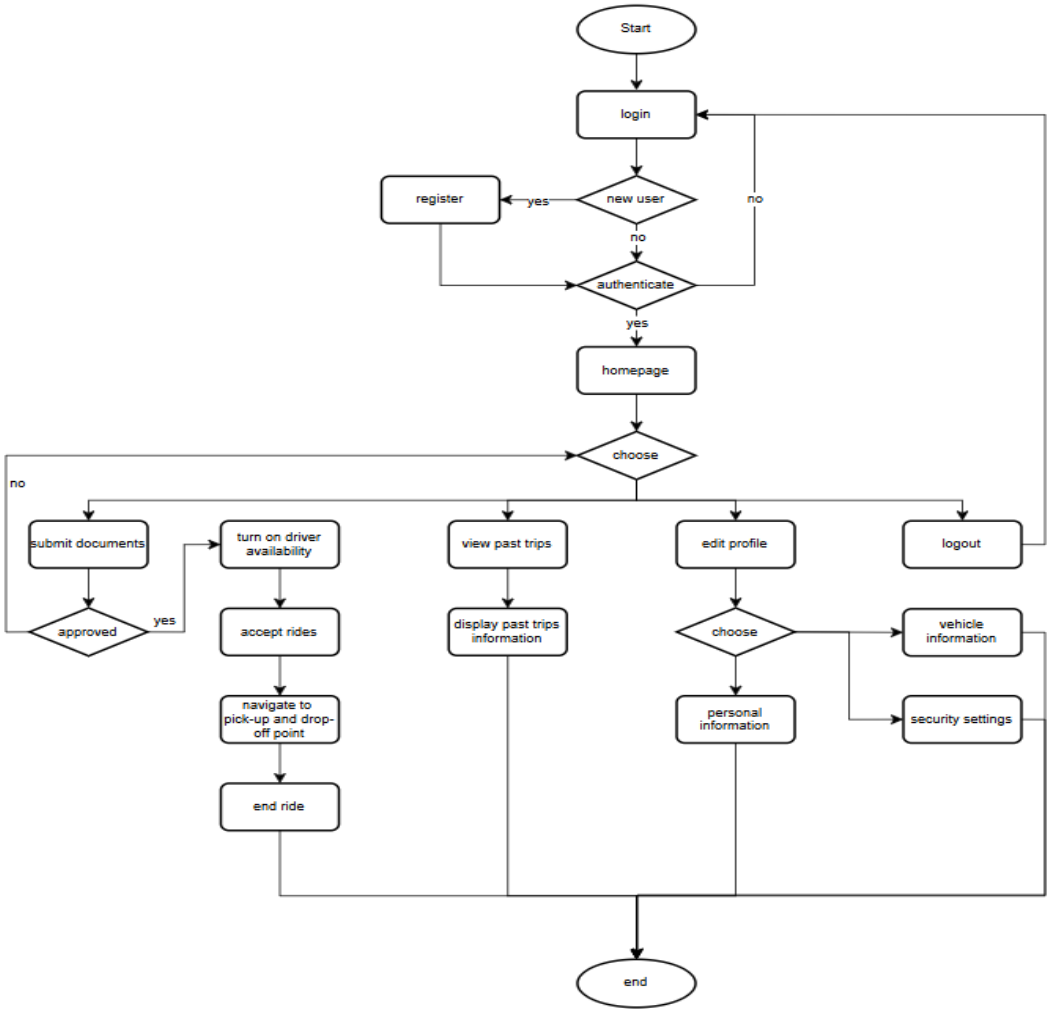


Fig.6 *Driver flowchart*

Fig. 7 shows the passenger flow in the Student E-Hailing App, from registration to booking a ride. It begins with login, which takes users to the homepage. A decision point lets passengers choose what they want to do next. During registration, users are needed to fill their information on the registration page such as student email verification to keep the platform exclusive to students. Security features include biometric login, email verification, and password protection to keep accounts safe. Passengers can also set up their profiles and edit information. For booking a ride, passengers can choose from available options based on their needs. The system confirms the booking and gives real-time updates to both the driver and passenger. Other features include booking history for reviewing past trips and a feedback system where passengers can rate drivers and share their experience to help improve the service.

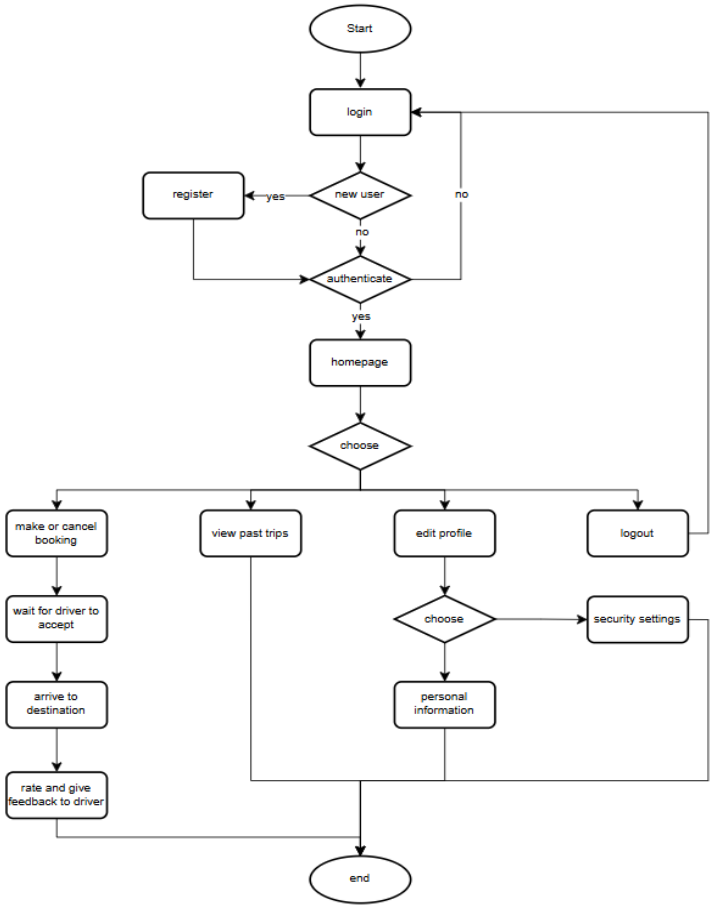


Fig.7 *Passenger flowchart*

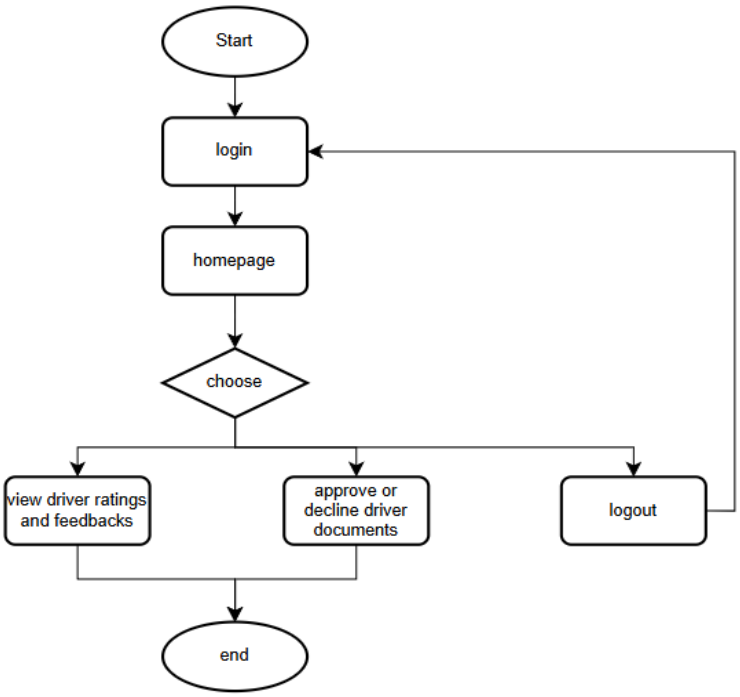


Fig.8 *Admin flowchart*

The flowchart in Fig. 8 outlines the admin interface and management functions in the Student E-Hailing App, focusing on safety, quality control, and efficient system oversight. The process begins with a secure login for admin users. After logging in, admins are taken to a dashboard that serves as the control center for all system management tasks. A decision point helps admins navigate to different functions based on what they need to do. One key function allows admins to view driver ratings and feedback. This helps monitor performance, respond to complaints, and identify areas for improvement or recognition. Admins can track feedback trends and take action when needed. Another major function handles driver verification. Admins review and approve new driver applications by checking documents like licenses, student credentials, vehicle registration, and insurance. This ensures all drivers meet safety and university standards. A secure logout feature ensures sessions end safely, protecting admin access from unauthorized use. Overall, this flowchart shows how the admin side of the app supports quality service through feedback monitoring and document verification.

4.3 System Design

The system design section consists of system architecture of the application. It provides an overview of the application's structural components and their interactions, the database relationships and user interface elements, demonstrating how data flows and users interact with the system effectively.

4.3.1 System Architecture

This system architecture in Fig. 9 shows the application with three types of users: Passengers, Drivers, and Admins. All users start by registering through the User Interface and then logging in to access their specific features. Passengers can book rides, confirm bookings, view ride details, and rate drivers. Drivers can confirm bookings, view their past trips, and check their ratings. Admins can view all ratings and feedback, generate reports, and handle disputes.The system works through a request-response flow where users make requests that get processed and generate responses back to them. All the data is stored in a Database that keeps track of rides, users, ratings, and other information.

The architecture connects all these components so passengers can request rides, drivers can accept and complete them, and admins can oversee the entire operation. Everything flows through the central Login system that determines what each user type can access, and the request-response mechanism handles all the interactions between users and the system.

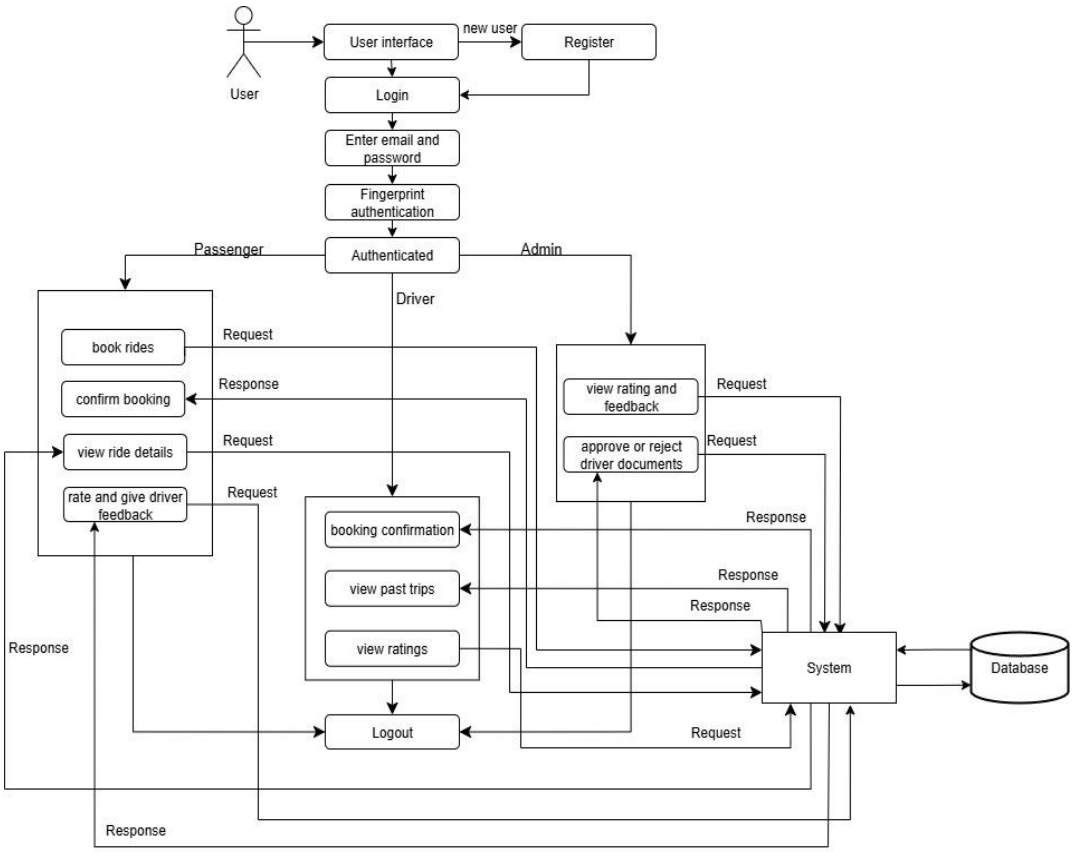


Fig. 9 *System Architecture of the Student E-Hailing Application*

5. Results and Discussion

5.1 Implementation

The Student E-Hailing Application is developed using Flutter and Firebase while the programming language is Dart. To access the application, a user must register an account by providing a name, student email, phone number, password, choose their gender and roles either passenger or driver. Fig. 10 shows the registration page for users while Fig. 11 shows the code segment of the registration page ensuring only emails from domain @student.uthm.edu.my is allowed to register.

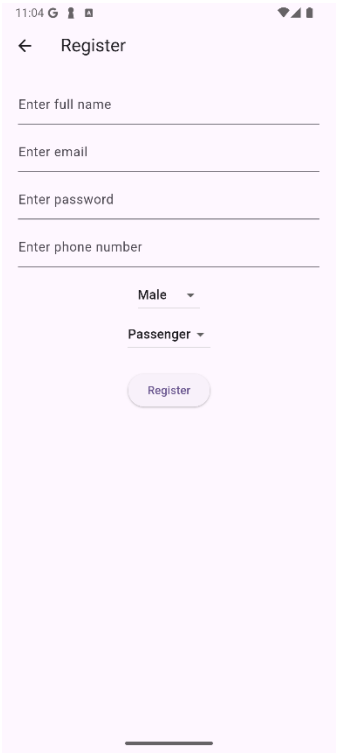


Fig. 10 *Registration Page*

```
                return 'Please enter an email';
              }
              if (!value.endsWith('@student.uthm.edu.my')) {
                return 'Email must be from @student.uthm.edu.my domain';
              }
              if (!RegExp(r'^[^@]+@student\.uthm\.edu\.my$').hasMatch(value)) {
                return 'Please enter a valid email format';
              }
              return null;
            },
          ),  // TextFormField
          SizedBox(height: 10),
          TextFormField(
            controller: password,
            obscureText: true,
            decoration: InputDecoration(hintText: 'Enter password'),
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter a password';
              }
              if (value.length < 8) {
                return 'Password must be at least 8 characters';
              }
              if (!RegExp(r'[A-Z]').hasMatch(value)) {
                return 'Password must contain at least one capital letter';
              }
              if (!RegExp(r'[0-9!@#$%^&*(),.?":{}|<>]').hasMatch(value)) {
                return 'Password must contain at least one number or special character';
              }
```

Fig. 11 *User Registration Code Segment*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 713-732

Fig. 12 shows the login interface for the application. The interface features a simple and clean design with two input fields for user credentials - an email address field and a password field for authentication.

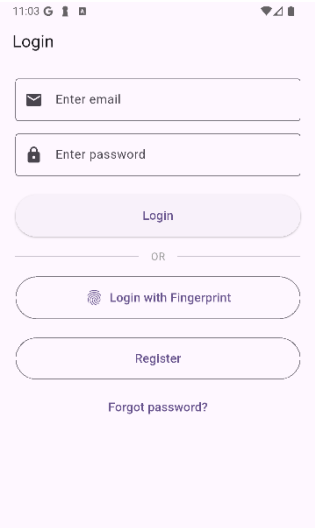
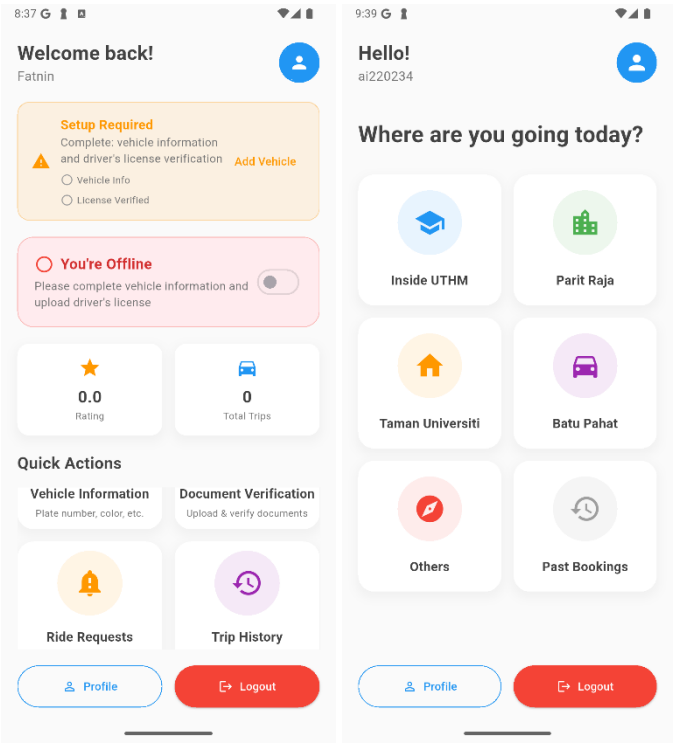


Fig. 12 *Login Page*

Fig. 13 (a) shows the driver homepage and passenger homepage in Fig. 13 (b). It displays the driver's verification status with a warning about incomplete vehicle information that needs attention before accepting rides. The driver can toggle between online and offline status to control ride availability. Quick Actions provide easy access to Vehicle Information for managing car details and Document Verification for maintaining required documentation. Additional features include Ride Requests to view incoming bookings and Trip History to track completed rides. The interface emphasizes status monitoring and ride management functionality. The passenger homepage on the right centers on booking and travel convenience. It greets the user and asks, "Where are you going today?" with six destination options for quick booking.

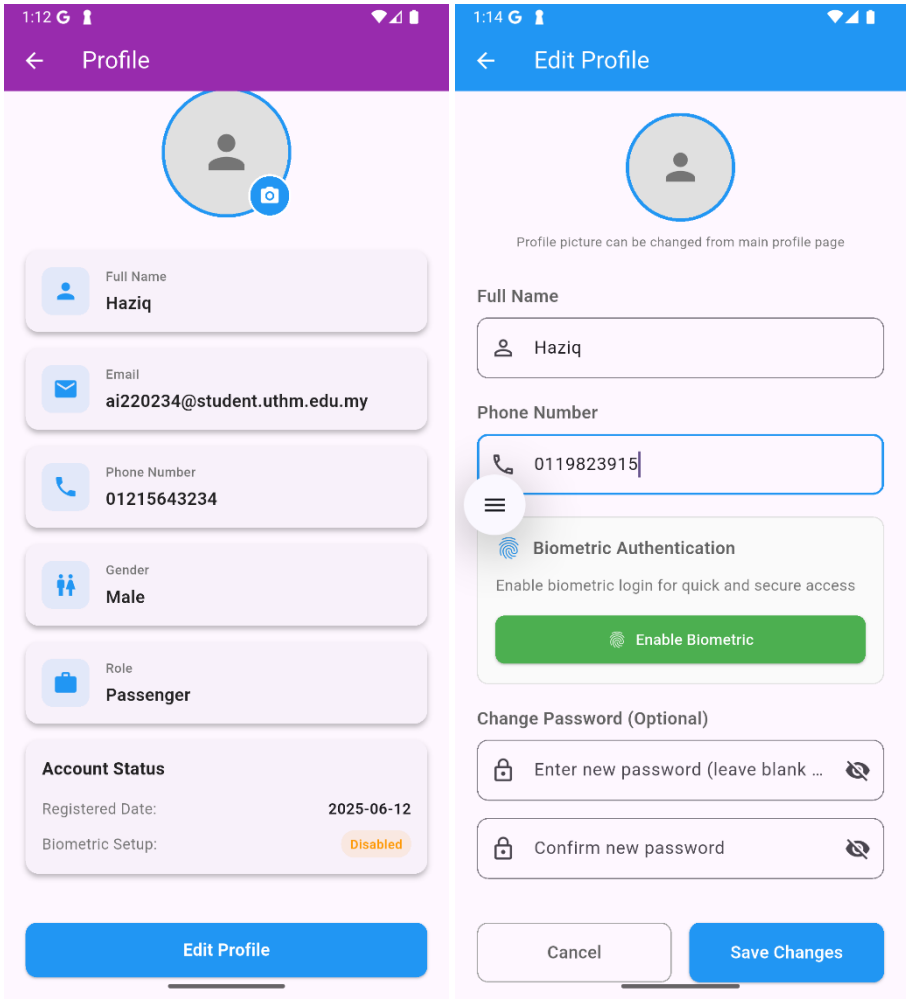


(a) **(b)**

Fig.13 *Homepage (a) Driver Homepage; (b) Passenger Homepage*

The screen display in Fig. 14 (a) shows the user profile interface. The interface shows a circular profile picture placeholder at the top, indicating where users can upload their profile photo. Below this, the screen presents several profile information fields in a card-based layout including sections for personal details such as name, contact information, and account status. The bottom of the screen features a prominent blue "Edit Profile" button that allows users to modify their information. The account status section shows a date indicator "2024-06-13", suggesting recent activity or account verification date. This screen represents the edit profile functionality, accessible from the main profile screen. The interface maintains the same design language with a blue header and circular profile image section at the top.

The screen in Fig. 14 (b) contains multiple input fields for users to update their personal information, including full name, contact details, and other relevant profile data. Notable features include password change functionality with fields for entering a new password and confirming the password change. The screen includes validation elements and secure input handling for sensitive information like passwords. A blue "Save Changes" button at the bottom allows users to commit their profile updates to the system. The interface also includes a "Cancel" option, providing users with the ability to discard changes if needed.



(a) **(b)**

Fig.14 *Users profile (a) Passenger Profile page (b) Edit profile*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 713-732

Fig. 15 shows the booking confirmation page for the application. The trip summary section displays the pickup and drop-off locations with green dot indicators as the destination along with fare information displaying. The pickup details section includes a text input field where users can add additional specifics about their pickup location. Below are the main action elements - a prominent green "Confirm Booking" button serves as the primary call-to-action, with a "Cancel" option positioned underneath it. The bottom of the screen features a typical mobile app navigation bar with various icons for different app functions.

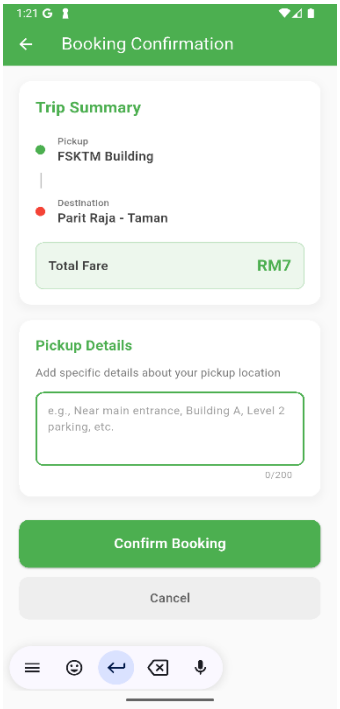
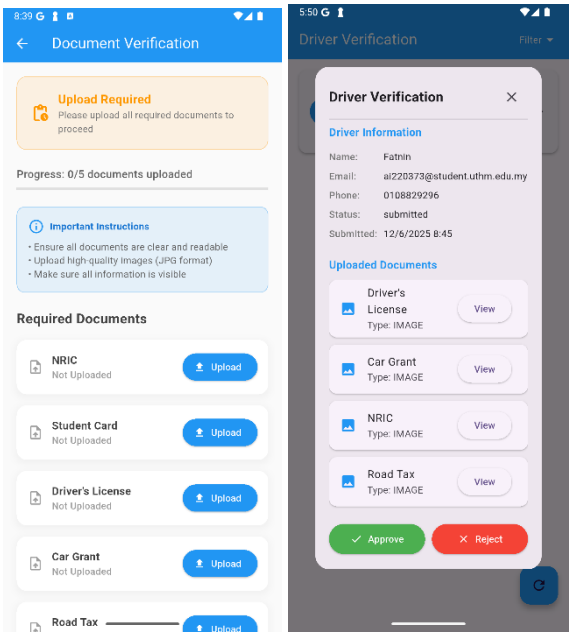


Fig.15 *Confirm Booking Page*

Fig. 16 is the Driver Document Verification Interface. The first image shows the driver's view where they can see what documents are required and upload them, while the second image shows the admin's view where they can review the submitted documents and approve or reject them.



(a) **(b)**

Fig.16 *Driver Document Verification Interface (a) Document Submission; (b) Document Verification Page*

Fig. 17 shows the logout confirmation dialog active. Once a user presses the 'Yes' button it will log out user and redirect users to the login page.

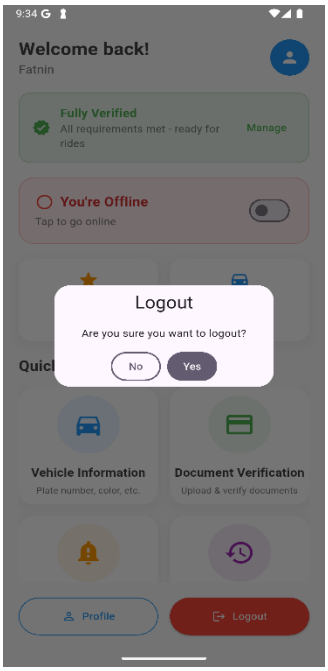


Fig.17 *Logout Page*

5.2 Testing

The test plan results show that the proposed system achieved all the expected results for all modules, such as registration, login, edit profile, book ride, confirm booking, rating and feedback, document verification, past bookings and logout functionality.

Table 7 *Testing result of proposed application*

No	Module	Test Case	Expected Result	Actual Result
1	Registration	Users attempt to register with incomplete personal data	Alert message displays if required field (name, email, phone, student ID) is empty.	Pass
		Users register with invalid email format	Alert if email format is invalid or not a university domain.	Pass
		Users register with weak password	Alert if password doesn't meet policy (8+ chars, upper/lowercase, number, special char).	Pass
		Users register with existing email/student ID	Alert if email or student ID already exists in database.	Pass
		Users register without completing email verification	Registration is blocked until email verification is completed.	Pass
		Users upload fake/invalid student card	Document verification fails, admin approval required before account activation.	Pass
2	Login	Users login with incorrect credentials	Alert message for invalid email/password.	Pass
		Users log in without completing MFA (e.g., no password, no fingerprint authentication)	Login fails without completing MFA.	Pass

Table 7 *(cont)*

No	Module	Test Case	Expected Result	Actual Result
		Users exceed login attempts	Accounts are temporarily locked after defined failed attempts.	Pass
3	Edit Profile	Users input invalid data formats (e.g., wrong phone/email)	Validation errors displayed.	Pass
		Unauthorized profile access	Access is denied by role and session checks.	Pass
4	Book Ride	Unauthenticated users try to book rides	Booking is denied until user is authenticated.	Pass
		Users try manipulating fare calculation	Fare logic is handled by the server-side and can't be altered.	Pass
5	Confirm Booking	Unauthorized users try confirming others' bookings	Only creator or assigned driver can confirm.	Pass
		Users try confirming expired/canceled bookings	System prevents confirmation of invalid booking status.	Pass
		Users try to manipulate booking details during confirmation	Booking data is read-only during confirmation.	Pass
6	Rating & Feedback	Users try submitting multiple ratings for the same ride	Only one rating per ride per user is allowed.	Pass
		Users try submitting rating before ride completion	Rating is disabled until ride is marked complete.	Pass
		Unauthorized users try to rate rides they weren't in	Only users involved in the ride can rate.	Pass
7	Document Verification	Non-admins try accessing document approval UI	Only admin role can access verification interface.	Pass
		Admin approves document without proper checks	Document review before approval.	Pass
		Unauthorized users try bypassing verification	Access is blocked; backend prevents manipulation or skipping.	Pass
8	Past Bookings	Users try accessing others' booking history	Only user's own booking history is accessible.	Pass
		Users try modifying past bookings	Past bookings are read-only.	Pass
9	Logout	Users try accessing features post-logout	Sessions terminate properly; access denied.	Pass

5.3 User Acceptance Testing

User acceptance testing form was created using Google Form and distributed to 18 android phone users to get their feedback. All respondents were university students, with an even distribution between those testing the application as drivers and those testing it as passengers. The user acceptance testing aims to evaluate the proposed application regarding interface, features and security. The Google Form consists of 11 questions with a satisfactory scale. The result of the user acceptance testing is shown in Fig. 18, 19 and 20.

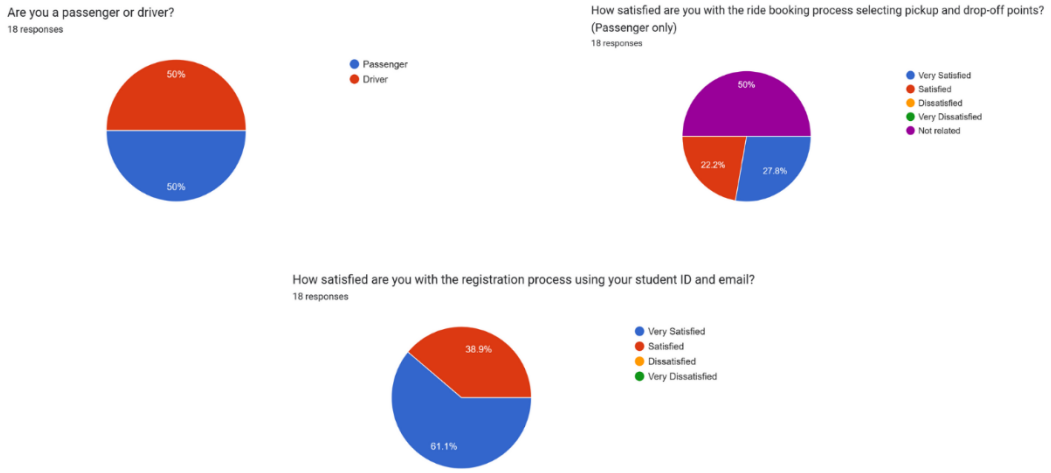


Fig.18 *User's satisfaction level*

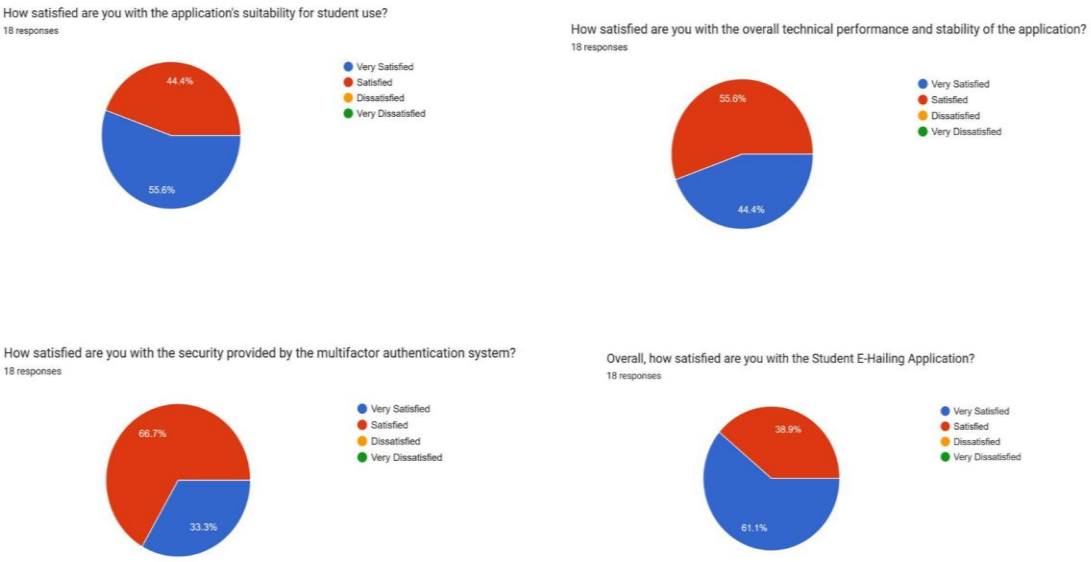


Fig.19 *User's satisfaction level*

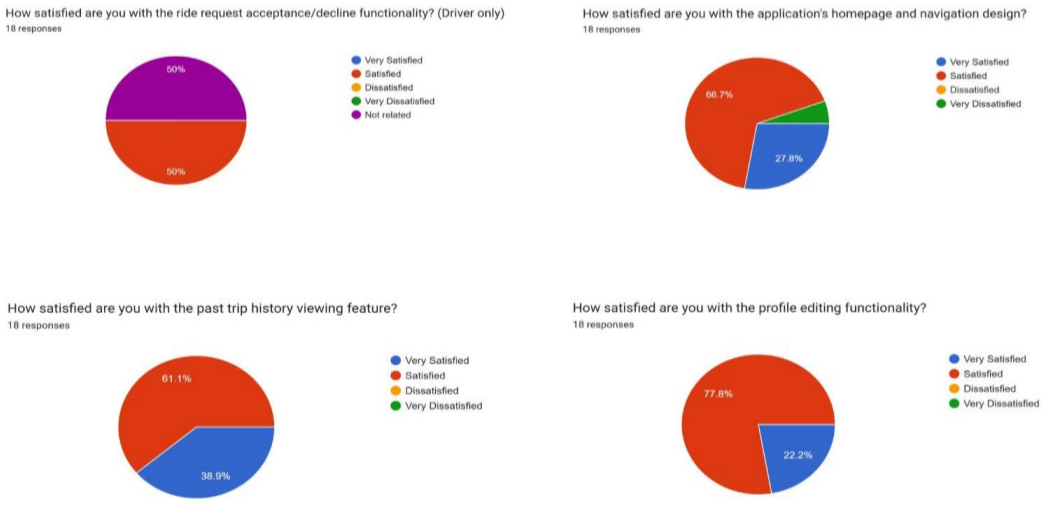


Fig.20 *Users' satisfaction level*

The feedback collected from users of the Student E-Hailing Application indicates a strong level of satisfaction with the system's functionality, user interface, and security features. Both passengers and drivers consistently rated the registration as "Very Satisfied," suggesting that the identity verification process using student ID and email was smooth and reliable. The Multi-Factor Authentication (MFA) system, a key security feature of the app, received overwhelmingly positive responses, with users feeling confident about its ability to protect their accounts.

Passengers expressed high satisfaction with the ride booking process, especially in selecting pickup and drop-off locations, which highlights the app's usability and convenience. Drivers responded positively to the ride request acceptance and earnings tracking features, showing that the system effectively supports their role within the platform. Additionally, most users found the homepage navigation, profile editing, and past trip viewing features to be intuitive and helpful, though a few responses indicated room for improvement in interface design consistency and stability.

6. Conclusion

The project has successfully achieved all its objectives, which were to design, develop, and evaluate a Student E-Hailing Application with Multi-Factor Authentication (MFA) tailored for university students. The application was developed using Flutter and Firebase, with the primary aim of enhancing transportation safety and convenience within a university environment. The system provides secure user authentication through the integration of student email verification, OTP (One-Time Password), and biometric facial recognition.

All functional and non-functional requirements were fulfilled to ensure system usability, performance, and security. Sensitive user information such as passwords is securely stored using hashing, and the authentication process includes multiple layers to prevent unauthorized access. Only verified students can register as passengers or drivers, and security measures like biometric authentication and OTP-based login significantly reduce the risk of impersonation or misuse.

After the development phase, the system underwent a comprehensive test plan, including unit testing, integration testing, and user acceptance testing involving real student users. The results demonstrated high levels of user satisfaction, particularly regarding the application's ease of use, booking process, and security features. Positive feedback confirmed that the multi-factor authentication system added a layer of trust and reliability to the platform.

However, there are several limitations to the current version of the application. First, the system is only available for Android devices. iOS support has not been implemented, which limits accessibility for users with Apple devices. Additionally, the application currently does not support advanced features such as trip-sharing between passengers, real-time driver location tracking, or fare payment integration. The application also only allows user to register as only one role.

To overcome these limitations, future improvements may include extending support to iOS platforms and offering alternative biometric options such as fingerprint recognition. Additional features like in-app payment, real-time GPS tracking, ride history export, and in-app messaging between drivers and passengers could be added to further enhance user experience. These enhancements will help transform the application into a more robust, full-featured e-hailing solution that meets the evolving needs of university students.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design**: *F. Wafdi Auni, Z. Abdullah*; **data collection**: *F. Wafdi Auni, Z. Abdullah*; **analysis and interpretation of results**: *F. Wafdi Auni, Z. Abdullah*; **draft manuscript preparation**: *F. Wafdi Auni, Z. Abdullah*. All authors reviewed the results and approved the final version of the manuscript

References

[1] Grab Holdings, *Grab - Taxi & Food Delivery* [Mobile application software], Google Play Store, 2012. Retrieved Nov. 12, 2024, from https://play.google.com/store/search?q=grab&c=apps.

[2] AirAsia Superapp Sdn Bhd, *AirAsia MOVE: Flights & Hotels* [Mobile application software], Google Play Store, 2020. Retrieved Nov. 12, 2024, from https://play.google.com/store/search?q=airasia&c=apps.

[3] Bolt Technology OÜ, *Bolt: Request a Ride* [Mobile application software], Google Play Store, 2019. Retrieved Nov. 12, 2024, from https://play.google.com/store/search?q=bolt&c=apps.
[4] Sabar, Sarinah, "Students Satisfaction of Using E-Hailing in UiTM Puncak Alam," *Journal of International Business, Economics and Entrepreneurship*, vol. 8, pp. 79-88, 2023. doi: 10.24191/jibe.v8i2.24739.
[5] Natarajan, T., & Pichai, S., "Transition From Waterfall to Agile Methodology: An Action Research Study," *IEEE Access*, vol. 12, pp. 49341-49362, 2024, doi: 10.1109/ACCESS.2024.3384097.
[6] Yan, Eng, Md. Jusoh, Zuroni, Zainudin, Norzalina, & Osman, Syuhaily, "Determinants of E-Hailing Service Adoption among University Students in Peninsular Malaysia," *International Journal of Academic Research in Business and Social Sciences*, vol. 14, 2024, doi: 10.6007/IJARBSS/v14-i10/23281.
[7] Nexapp, "Agile Software Development: Understanding the Cycle," *Nexapp*, https://www.nexapp.ca/en/blog/agile-software-development