UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Car Rental System

Hassan Mustafa Ibrahim
Faculty of Computing
Universiti Teknologi Malaysia
Johor Bahru, Johor, Malaysia
ibrahimhassan@graduate.utm.my

Dr Muhammad Khatibsyarbini
Faculty of Computing
Universiti Teknologi Malaysia
Johor Bahru, Johor, Malaysia
khatibsyarbini@utm.my

***Abstract*— The objective of this project is to create a mobile application for a car rental system, designed to streamline the car rental process for both car owners and renters. The traditional methods of car rental often involve local advertising and face-to-face interactions, which can be time-consuming and inefficient. This project aims to address these issues by providing a platform that allows car owners to list their cars for rent and renters to find and book available cars with ease. The application, developed using the Flutter framework and Firebase for the database, will feature functionalities such as car selection, booking, payment processing, and rental history management. It will also offer car owners tools for managing their car fleets and tracking revenue. By digitizing the car rental process, this system seeks to provide a convenient, cost-effective, and sustainable transportation solution.**

Keywords — Car Rental System, Software Engineering, Mobile ApplicationDevelopment, Flutter, Firebase, Car Fleet Management, Digital Transformation, Transportation Solution

I. Introduction

A. Problem Background

The advancements in technology have had a significant impact on several industries, including reservation systems. With Trivago, Expedia, and Agoda, just to name a few, recommending hotels to book, it's clear that the internet has revolutionized the way we make reservations [1]. However, finding applications that offer car rental services has been a challenge. Individuals searching for cars to rent often encounter difficulties due to the heavy reliance of car rental companies on local advertising. To address this issue, a platform that can assist both car owners and renters is needed. This platform will help owners promote their cars, while renters can quickly search for available cars.

A car rental agency is a firm that rents out cars to consumers for a price and a limited time. However, many rental car companies still operate traditionally by advertising their cars locally, which wastes both the owner's and renter's time and money. To address this issue, a better system is needed for supplying, renting, and managing cars. This proposed system will collect data from both renters and car owners, store personal information, and provide registered users with the option to rent a car. Car owners can also register their cars to market them to other users.

There are several cars for rent owned by various owners. Each car has benefits and cons of its own, and some cars are distinct from one another. Due to the wide variety of cars accessible, customers may find it difficult to select the one that best suits them and meets all of their needs. The conventional approach of renting a car by going to a renting agency also makes it difficult to track the history of rented cars, as well as those who do not have access to transit, such as students who need to leave campus for various activities. The car's owner, on the other hand, just informs the locals of the car's availability. A customer must connect with the car's rental owner and sign a contract to rent a car, which slows down the rental process.

To tackle this issue, a platform that can help both car owners and renters is needed to make car rental easier and more effective for everyone involved [2]. The system that is proposed will collect information from both renters and car owners, retain personal information, and allow registered users to rent a car. Car owners can also register their cars to market them to other users.

B. Project Objectives

The objectives of the project are:

- To elicit the requirements of the current car renting procedure by customers and car owners for the system.
- To design and develop a mobile application based on the requirements gathered.
- To test the developed mobile application for defects and weaknesses and to correct them until the product matches the original requirements.

II. LITERATURE REVIEW

The car rental industry has seen significant growth driven by the increasing demand for convenient and cost-effective transportation solutions, particularly for short-term travel needs [3]. Despite this expansion, traditional rental processes often remain cumbersome and inefficient, primarily relying on local operations that restrict availability and accessibility for both renters and owners. This limitation underscores the necessity for a modernized, user-friendly car rental system that can streamline the booking process and enhance overall service efficiency.

Existing literature highlights several challenges inherent in current car rental systems. These challenges include lengthy reservation procedures, limited vehicle availability, and a lack of transparent and standardized rental management practices. Moreover, traditional systems often fail to leverage advanced technologies to optimize operations, resulting in suboptimal user experiences and missed opportunities for business growth and customer satisfaction.

To address these shortcomings, the proposed system aims to integrate innovative features and functionalities essential for enhancing the rental experience for both owners and renters. Key requirements identified from the literature include robust reservation mechanisms, secure payment processing, real-time vehicle availability tracking, and user-friendly interfaces. These features not only aim to simplify the booking process but also to improve operational transparency and customer trust.

synthesizing insights from the reviewed literature emphasizes the critical need for an integrated and technologically advanced car rental system. By addressing current system limitations and leveraging best practices from successful platforms, the proposed system seeks to set a new standard in the industry, enhancing operational efficiency, user satisfaction, and business profitability.

III. SYSTEM DEVELOPMENT METHODOLOGY

System development methodologies are essential in software engineering, providing a structured approach to creating complex software systems. These frameworks guide the development process, helping teams manage complexity and deliver high-quality solutions. Key concepts like the Software Development Life Cycle (SDLC), requirements analysis, and stakeholder involvement are integral. By following these methodologies, development teams can optimize resources, improve communication, and increase project success rates.

The Waterfall methodology was chosen for developing the car rental system due to the industry's need for a thorough understanding of requirements from the beginning and the consistency of core functions. The Waterfall approach is linear and sequential, with each phase (requirements gathering, system design, implementation, testing, deployment, and maintenance) completed before moving to the next [4]. This ensures a solid foundation for planning and documentation.

The stable nature of the car rental industry makes the Waterfall methodology well-suited for delivering a reliable system. Its structured path facilitates effective resource allocation, timeline management, and stakeholder communication, providing a disciplined framework for successful system development.

The Waterfall methodology, a sequential and linear approach to software development, encompasses distinct phases, each completed before advancing to the next. Beginning with Requirements Gathering, stakeholders collaborate closely with the project team to define and document software requirements comprehensively. This phase involves interviews, workshops, and surveys to ensure clarity and feasibility, resulting in a Software Requirements Specification (SRS) document that outlines functional requirements and system constraints.

System Design transforms requirements into a detailed system architecture using Unified Modelling Language (UML). UML diagrams define component interfaces, data models, and user interface layouts, facilitated by tools like Enterprise Architect. The resulting Software Design Document (SDD) consolidates architectural diagrams, technical specifications, and design patterns, guiding development through meticulous documentation and enhanced collaboration.

Implementation follows, where developers write code and configure system components based on SDD guidelines. Coding standards and continuous integration ensure efficient, maintainable software. Configuration files and database schemas are established to build the system effectively.

Testing validates software functionality, performance, and reliability through unit testing, integration testing, and user acceptance testing. The Software Test Document (STD) outlines test strategies and scenarios, identifying and resolving defects to meet defined requirements.

Deployment prepares software for release, involving installation, configuration, and data migration to minimize disruptions and ensure a seamless transition. Finally, Maintenance provides ongoing support, addressing issues and enhancing software based on user feedback and evolving requirements to sustain optimal performance and usability.

The development of the car rental system harnessed a blend of technologies and tools essential for robust functionality and efficient development. Central to the project was the Flutter framework, enabling cross-platform app development with seamless deployment on Android devices. Firebase provided critical backend support, offering features like real-time database management, authentication, and cloud storage. Visual Studio Code served as the primary code editor, delivering a lightweight yet powerful environment for coding, debugging, and version control integration. Complementing these tools, Android Studio IDE facilitated comprehensive Android app development, supported by its emulator for testing across diverse device configurations. GitHub acted as the centralized version control system, ensuring collaborative code management and project backup capabilities.

IV. Requirement Analysis and Design

Requirement Analysis and Design is a critical phase in software development, offering essential guidance and practices for engineers and stakeholders alike. It begins with Requirement Analysis, where stakeholders' needs and constraints are meticulously identified and documented to define a clear project scope [5]. Project Design follows, enabling the creation of an architectural blueprint aligned with project goals to ensure scalability, maintainability, and system efficiency [6].

A. Requirement Analysis

1) Functional Requirements

Functional requirements show the common tasks that the users will conduct in a system. TABLE I shows the functional requirements extracted during the requirement elicitation process.

TABLE I. Functional Requirements

Functional Requirements	**Description**
FR01: User Registration	Users should be able to register and create an account, providing their personal information.
FR02: Car Listing	Owners should be able to list their cars on the platform, providing details such as make, model, year, fuel type, seating capacity, and photos.
FR03: Booking and Reservations	Renters should be able to search for available cars, book and make reservations for specific dates and times, and pay for the rental.
FR04: Geolocation	The system should display the nearest available cars to renters based on their current location.
FR05: Payment Processing	The system should be able to securely process payments from renters and distribute earnings to owners.
FR06: Ratings and reviews	Renters should be able to rate and review the cars they rented, helping to build trust and credibility on the platform.
FR07: Notifications	The system should send notifications to both owners and renters regarding bookings, and other important events.
FR08: User Verification	The admin should be able to verify the identity of both owners and renters to ensure the safety and security of the platform.
FR09: Integration	The system should be able to integrate with third-party services such as Google Maps and Google Sign in/up for improved functionality and user experience.
FR10: Search Option	Renters should be able to search for cars based on specific criteria to show only the cars that match their search.
FR11: Availability Status	Hosts should have the ability to change the status of their cars to make them available or unavailable for rent as needed.
FR12: Pricing	Owners should be able to set their own rental prices.
FR13: Rental Agreements	The system should provide rental agreements that outline the terms and conditions of the rental, including any liabilities or responsibilities of the owner or renter.
FR14: Reporting and analytics	The system should provide owners with reporting and analytics on the performance of their cars, including rental history, earnings, and feedback from renters.

The requirements are then converted into use case diagram as shown in fig 1.

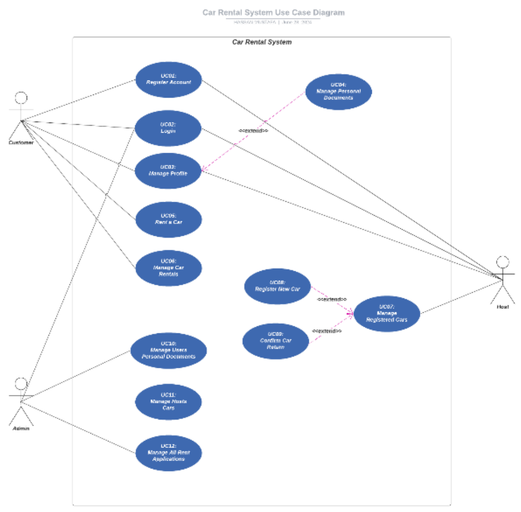


Fig. 1. Use Case Diagram Car Rental System

2) Non-Functional Requirements

The non-functional requirements of this system are described as table below:

TABLE II. Non-Functional Requirement

Non-Functional Requirements	**Description**
NFR01: Security	The system should be secure, protecting user data and financial transactions from potential threats.
NFR02: Usability	The system should be user-friendly, with clear navigation and intuitive interfaces for both owners and renters.
NFR03: Performance and Network	The system should remain responsive under high traffic, provide efficient data transfer, and function smoothly on standard mobile networks.
NFR04: Reliability	They system must be reliable, ensuring that it remains operational with minimal downtime and consistent availability.

B. System Design

The car rental system's overall design is based on the Model-View-Controller (MVC) pattern, as depicted in Fig 2. This pattern was chosen for its advantages in organizing and maintaining the codebase. In MVC, the Model manages data and business logic, the View handles the user interface, and the Controller orchestrates data flow and user interactions. This approach promotes modular development, encourages developer collaboration, and provides flexibility for system changes and upgrades. Additionally, the Class Diagram, shown in Fig 3, is integral to the system's architecture. It offers a comprehensive overview of the system's classes, attributes, relationships, and methods, such as car, customer, and car rental. This diagram serves as a visual blueprint, facilitating a thorough understanding of the system's structure and aiding in effective design, implementation, and maintenance. Together, the MVC pattern and Class Diagram ensure the development of a robust, scalable, and maintainable car rental system.

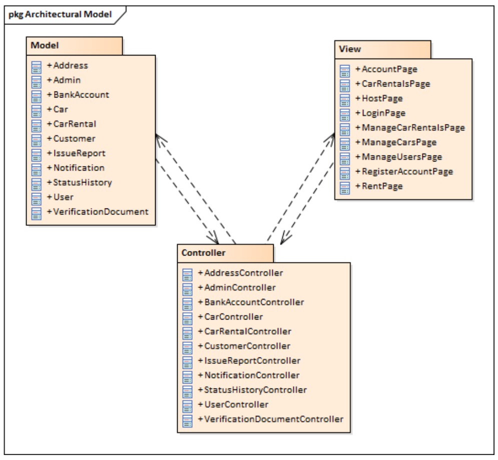


Fig. 2. Package Diagram of Car Rental System

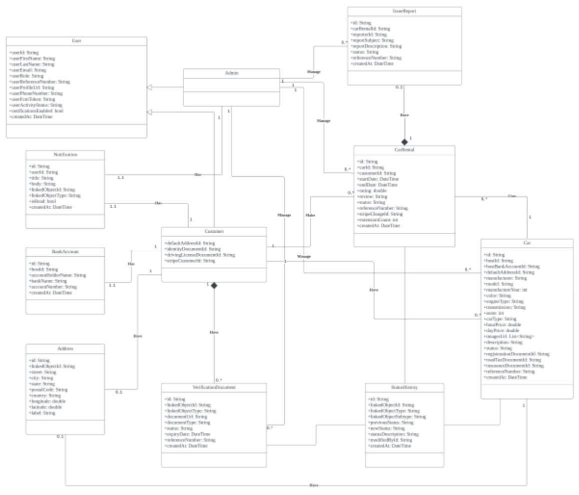


Fig. 3. Class Diagram of Car Rental System

C. Database Design

Figure below shows the database design for the car rental system.

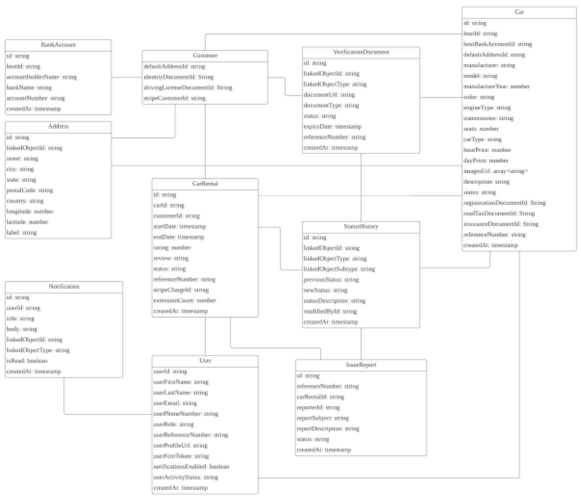


Fig. 4. Database Design for the Car Rental System

D. User Interface Design

The Interface Design section illustrates the system's visual layout and user experience, showcasing example interfaces in fig 5 to 7.

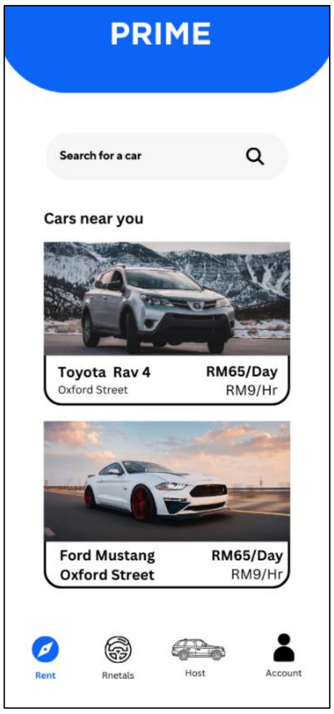


Fig. 5. Rent Page

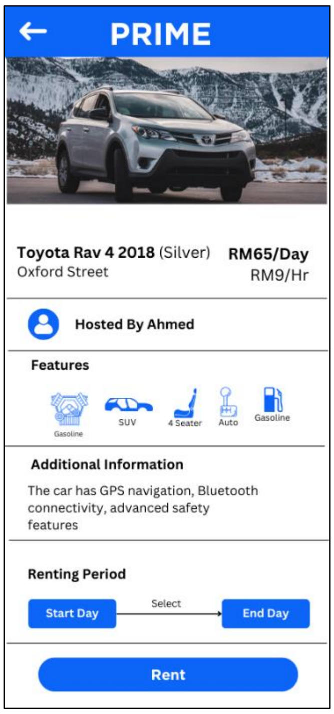


Fig. 6. Rent Car Details Page

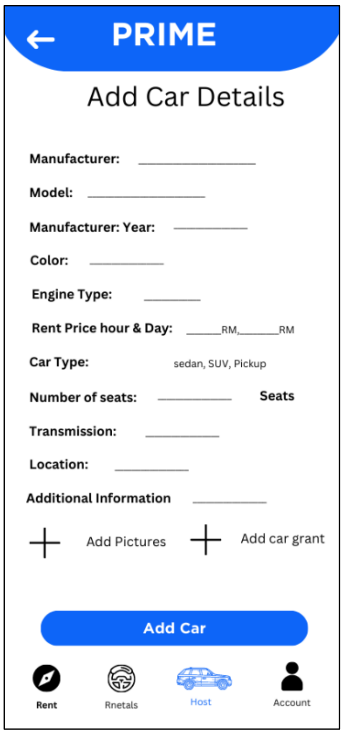
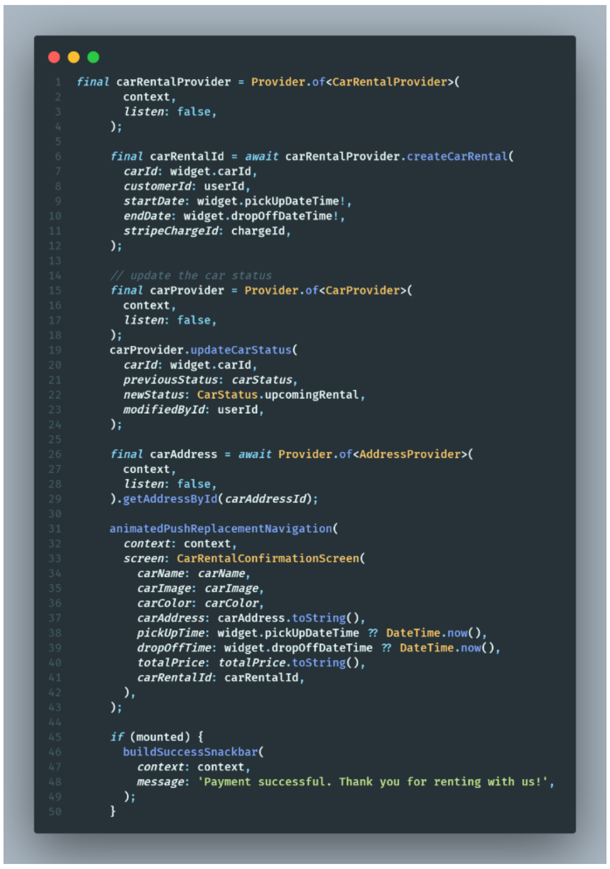


Fig. 7. Register Car Page

V. IMPLEMENTATION AND TESTING

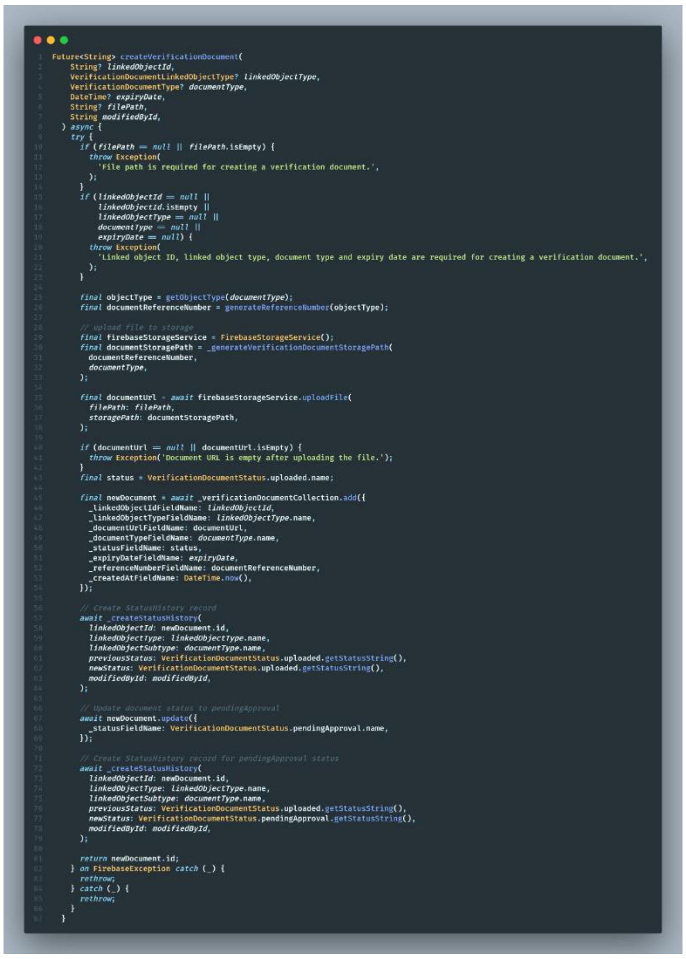
A. Coding of system main functions

The front-end application of this car rental system is built using the Flutter framework. The back-end utilizes Firebase, providing robust infrastructure with real-time database capabilities, authentication, and cloud storage. Figures below showcase some of the coding aspects of the system.



```
final carRentalProvider = Provider.of<CarRentalProvider>(
        context,
        listen: false,
      );

      final carRentalId = await carRentalProvider.createCarRental(
        carId: widget.carId,
        customerId: userId,
        startDate: widget.pickUpDateTime!,
        endDate: widget.dropOffDateTime!,
        stripeChargeId: chargeId,
      );

      // update the car status
      final carProvider = Provider.of<CarProvider>(
        context,
        listen: false,
      );
      carProvider.updateCarStatus(
        carId: widget.carId,
        previousStatus: carStatus,
        newStatus: CarStatus.upcomingRental,
        modifiedById: userId,
      );

      final carAddress = await Provider.of<AddressProvider>(
        context,
        listen: false,
      ).getAddressById(carAddressId);

      animatedPushReplacementNavigation(
        context: context,
        screen: CarRentalConfirmationScreen(
          carName: carName,
          carImage: carImage,
          carColor: carColor,
          carAddress: carAddress.toString(),
          pickUpTime: widget.pickUpDateTime ?? DateTime.now(),
          dropOffTime: widget.dropOffDateTime ?? DateTime.now(),
          totalPrice: totalPrice.toString(),
          carRentalId: carRentalId,
        ),
      );

      if (mounted) {
        buildSuccessSnackbar(
          context: context,
          message: 'Payment successful. Thank you for renting with us!',
        );
      }
```

Fig. 8.Create Car Rental Function



```
Future<String> createVerificationDocument(
    String? linkedObjectId,
    VerificationDocumentLinkedObjectType? linkedObjectType,
    VerificationDocumentType? documentType,
    DateTime? expiryDate,
    String? filePath,
    String modifiedById,
  ) async {
    try {
      if (filePath == null || filePath.isEmpty) {
        throw Exception(
          'File path is required for creating a verification document.',
        );
      }
      if (linkedObjectId == null ||
          linkedObjectId.isEmpty ||
          linkedObjectType == null ||
          documentType == null ||
          expiryDate == null) {
        throw Exception(
          'Linked object ID, linked object type, document type and expiry date are required for creating a verification document.',
        );
      }

      final objectType = getObjectType(documentType);
      final documentReferenceNumber = generateReferenceNumber(objectType);

      // upload file to storage
      final firebaseStorageService = FirebaseStorageService();
      final documentStoragePath = _generateVerificationDocumentStoragePath(
        documentReferenceNumber,
        documentType,
      );

      final documentUrl = await firebaseStorageService.uploadFile(
        filePath: filePath,
        storagePath: documentStoragePath,
      );

      if (documentUrl == null || documentUrl.isEmpty) {
        throw Exception('Document URL is empty after uploading the file.');
      }
      final status = VerificationDocumentStatus.uploaded.name;

      final newDocument = await _verificationDocumentCollection.add({
        _linkedObjectIdFieldName: linkedObjectId,
        _linkedObjectTypeFieldName: linkedObjectType.name,
        _documentUrlFieldName: documentUrl,
        _documentTypeFieldName: documentType.name,
        _statusFieldName: status,
        _expiryDateFieldName: expiryDate,
        _referenceNumberFieldName: documentReferenceNumber,
        _createdAtFieldName: DateTime.now(),
      });

      // Create StatusHistory record
      await _createStatusHistory(
        linkedObjectId: newDocument.id,
        linkedObjectType: linkedObjectType.name,
        linkedObjectSubtype: documentType.name,
        previousStatus: VerificationDocumentStatus.uploaded.getStatusString(),
        newStatus: VerificationDocumentStatus.uploaded.getStatusString(),
        modifiedById: modifiedById,
      );

      // Update document status to pendingApproval
      await newDocument.update({
        _statusFieldName: VerificationDocumentStatus.pendingApproval.name,
      });

      // Create StatusHistory record for pendingApproval status
      await _createStatusHistory(
        linkedObjectId: newDocument.id,
        linkedObjectType: linkedObjectType.name,
        linkedObjectSubtype: documentType.name,
        previousStatus: VerificationDocumentStatus.uploaded.getStatusString(),
        newStatus: VerificationDocumentStatus.pendingApproval.getStatusString(),
        modifiedById: modifiedById,
      );

      return newDocument.id;
    } on FirebaseException catch (_) {
      rethrow;
    } catch (_) {
      rethrow;
    }
  }
```

Fig. 9. Create Verification Document Function

B. Interfaces of system main functions

The next section will present samples of the system's main functions interfaces, providing a detailed look at the user interactions and functionalities of the car rental application.

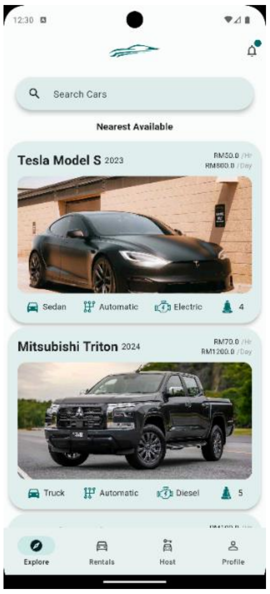
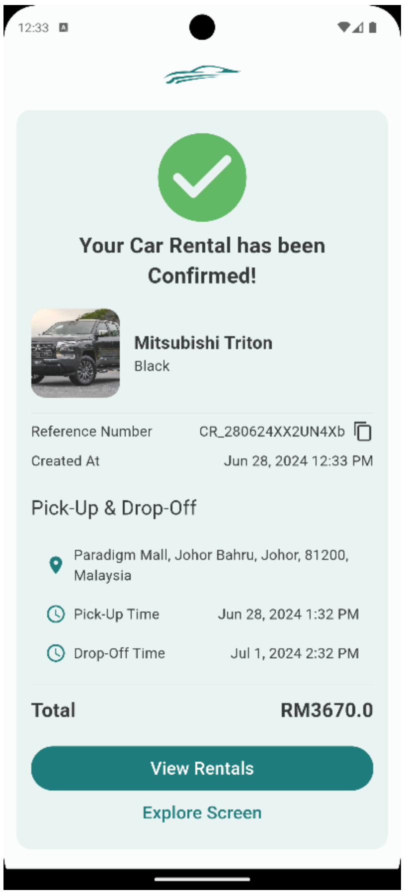


Fig. 10. Explore Screen


12:33

Your Car Rental has been Confirmed!

Mitsubishi Triton
Black

Reference Number CR_280624XX2UN4Xb
Created At Jun 28, 2024 12:33 PM

Pick-Up & Drop-Off

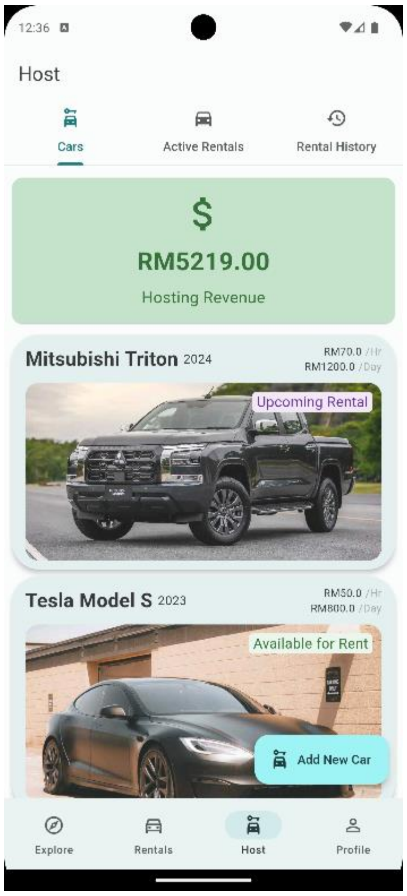
Paradigm Mall, Johor Bahru, Johor, 81200, Malaysia

Pick-Up Time Jun 28, 2024 1:32 PM

Drop-Off Time Jul 1, 2024 2:32 PM

Total RM3670.0

Fig. 11. Car Rental Confirmation Screen


12:36

Host

$

RM5219.00

Hosting Revenue

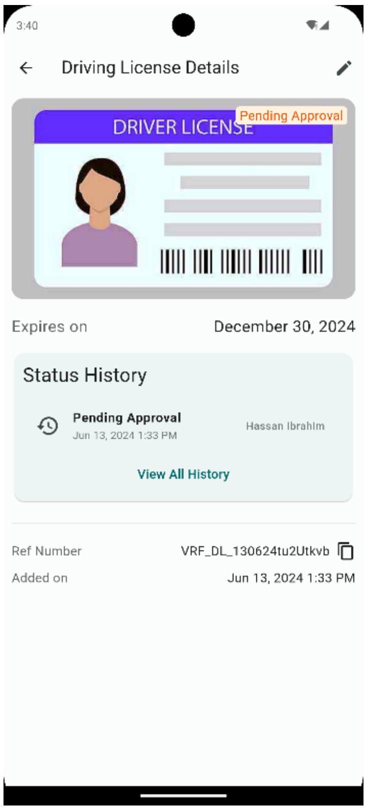
Mitsubishi Triton 2024 RM70.0 /Hr RM1200.0 /Day

Upcoming Rental

Tesla Model S 2023 RM50.0 /Hr RM800.0 /Day

Available for Rent

Fig. 12. Host Cars Screen


3:40

Driving License Details

Expires on December 30, 2024

Status History

Pending Approval
Jun 13, 2024 1:33 PM
Hassan Ibrahim

Ref Number VRF_DL_130624tu2Utkvb
Added on Jun 13, 2024 1:33 PM

Fig. 13. Verification Document Details

C. Testing

Testing is a crucial phase in the software development lifecycle, ensuring that the system functions as intended and meets the specified requirements. Effective testing identifies defects, enhances system performance, and improves overall user satisfaction by delivering a robust and reliable application. During the development of the Car Rental System, extensive testing was conducted to verify the system's functionality and usability. This section focuses on two primary types of testing that were performed: Black Box Testing, User Acceptance Testing (UAT) and Non-Function Requirements (NFR) Testing.

Black Box Testing was employed to examine the functionality of the Car Rental System without delving into its internal code structure. This approach focused on verifying that the input provided by the user resulted in the expected output and that the system behaved correctly under various scenarios. By treating the system as a "black box," it simulated real-world usage and identified discrepancies between the actual and expected outcomes. Specific test cases were designed based on the system's specifications and requirements, and these cases were executed to ensure that all functionalities performed as expected. The results of Black Box Testing confirmed that the Car Rental System adhered to the specified requirements and performed its functions accurately.

User Acceptance Testing (UAT) is a crucial phase in the software development lifecycle where the end users test the application to ensure it meets their requirements and is ready for production. The purpose of this UAT is to ensure that the car rental system meets the business requirements and provides a satisfactory user experience.

The scope includes testing all major functionalities, user interfaces, and performance aspects of the app. The main objectives are to validate that the app functions as expected, identify any defects, and confirm that the app is ready for production.

on-Functional Requirements (NFR) Testing ensures the robustness and efficiency of the Car Rental System beyond its functional features. This testing focused on critical non-functional aspects such as security, usability, performance, and reliability. Security testing leveraged Firebase's robust infrastructure, ensuring secure data storage and transmission. Usability testing involved verifying the application’s responsiveness and user-friendliness across various devices. Performance and network testing ensured the app's scalability and responsiveness under high traffic conditions, while reliability testing confirmed the system’s consistent uptime and availability. NFR Testing demonstrated that the Car Rental System met and exceeded industry standards for non-functional requirements, providing a secure, efficient, and reliable user experience.

VI. CONCLUSION

The project successfully achieved its stated objectives. It began with gathering requirements from customers and car owners through a comprehensive literature review and stakeholder analysis, identifying the essential features for an effective car rental system. The team then designed and developed a Flutter-based mobile application incorporating these features, including user registration, car listing, booking management, and customer reviews, along with tools for car owners to manage their listings and access performance analytics. Rigorous testing was conducted to ensure functionality, performance, and reliability, addressing any defects or weaknesses to match the original requirements. The result was a seamless, user-friendly mobile application that significantly enhances the car rental process and customer satisfaction.

Despite achieving its objectives, there are opportunities for further improvement. Potential enhancements include integrating dynamic pricing based on demand and market factors, expanding compatibility across multiple platforms, and adding features like rental extensions, date-specific car filtering, map views, and car favoriting. For verification documents, implementing automatic status updates and admin notifications can streamline processes. Additional improvements involve generating user reports, alerting users about poor connectivity, integrating third-party services, and incorporating AI-driven personalized recommendations and loyalty programs. Adding a chatbot for customer support, an internal chat system, and features to manage user activities for security can also enhance the system. These improvements can solidify the app's position in the industry, offering exceptional user experiences and staying ahead of market demands.

ACKNOWLEDGMENT

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed to my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Dr. Muhammad Khatibsyarbini Abd Rahim, for his encouragement, guidance, critics, and friendship.

My fellow students should also be recognized for their support. My sincere appreciation also extends to all my colleagues and others who have assisted on various occasions. Their views and tips are useful indeed. grateful to all my family members.

REFERENCES

[1] Cobanoglu, C., Dogan, S., Berezina, K., & Collins, G., 2021. Hospitality and Tourism Information Technology. 17th ed. University of South Florida M3 Center Publishing.

[2] Sundararajan, A., 2014. Peer-to-peer businesses and the sharing (collaborative) economy: Overview, economic effects and regulatory issues. Written testimony for the hearing titled The Power of Connection: Peer to Peer Businesses, pp. 1-7.

[3] Soares Machado, C. A., de Salles Hue, N. P. M., Berssaneti, F. T., & Quintanilha, J. A. (2018). An overview of shared mobility. *Sustainability*, 10(12), 4342.

[4] Royce, W. W., 1970. Managing the development of large software systems: concepts and techniques. Proceedings of IEEE WESCON, Volume 26, pp. 1-9.

[5] Sommerville & I., 2016. Software engineering. 10th ed. s.l. Pearson.

[6] Bass, et al., 2015. Software architecture in practice. 3rd ed. s.l. Addison-W