Applied Information Technology And Computer Science Vol. 4 No. 2 (2023) 1486-1502

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

The Development of UTHM Bus Tracking Mobile Application Using an Object-Oriented Approach

Muhammad Izryzza Nazrin Jafni[1], **Nur Ariffin Mohd Zin**[2]

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

DOI: https://doi.org/10.30880/aitcs.2023.04.02.084
Received 23 June 2023; Accepted 09 November 2023; Available online 30 November 2023

Abstract: This study focuses on the development of the UTHM Bus Tracking Mobile Application, a mobile-based system aimed at addressing the problem faced by Universiti Tun Hussein Onn Malaysia (UTHM) students in tracking the location of the bus. The objective of this research is to provide an effective solution to enhance the bus transportation experience for UTHM students. The system was developed using an object-oriented approach, employing the Waterfall model to ensure a systematic development process. The Flutter framework and Firebase server were utilized for application development and data storage, respectively. Key findings indicate that the UTHM Bus Tracking Mobile Application enables students to track the bus location, thereby improving the efficiency of public transportation. The discussion highlights the significance of this system in students' daily lives and suggests possibilities for future enhancements.

Keywords: Bus Tracking System, Mobile-Based System, Object-Oriented Approach

1. Introduction

This paper focuses on the development of the UTHM Bus Tracking Mobile Application, designed to address the challenges faced by students at Universiti Tun Hussein Onn Malaysia (UTHM) in tracking the campus bus. The objective of this study is to enhance the efficiency of the bus system and provide students with real-time bus information, ultimately improving their transportation experience. The motivation behind this research stems from the common problem of students struggling to estimate bus arrival times, resulting in prolonged waiting periods and inconvenience. By developing a mobile application that leverages real-time data and the Google Maps API, students can track bus locations, access bus information, and plan their journeys effectively. This paper presents the problem statement, research objectives, and the expected outcome of the UTHM Bus Tracking Mobile Application. The scope of the study focuses on the UTHM area, with a particular emphasis on improving bus system efficiency for students. The subsequent sections delve into the methodology, system design, and implementation details [1].

Corresponding author: ariffin@uthm.edu.my

2. Related Work

2.1 Bus Tracking System

Bus tracking systems utilize advanced GPS technology and software applications to precisely monitor and relay real-time information regarding the locations of buses to both operators and passengers. By providing accurate updates, these systems empower passengers with the ability to conveniently track the arrival times of buses. Simultaneously, they assist operators in effectively managing routes and optimizing resource allocation. Implementing a bus tracking system offers numerous benefits, particularly in ensuring punctuality and efficiency. Passengers can effortlessly monitor crucial details such as the bus's current location, speed, direction of travel, and other essential metrics. This wealth of information enables them to plan their journeys effectively, anticipate arrivals, and make informed decisions. Furthermore, operators can leverage this data to optimize route planning, establish reliable schedules, and even facilitate maintenance arrangements for the fleet [2].

2.2 Mobile Application

A mobile application, also known as a mobile app, is a sophisticated software program specifically designed to operate on mobile devices such as smartphones or tablets. Distinguished from desktop applications tailored for desktop computers and web applications that run within mobile web browsers, mobile apps offer a unique user experience optimized for handheld devices. Initially developed to enhance productivity through features like email, calendars, and contact databases, the demand for mobile applications swiftly expanded to encompass diverse domains. Today, these applications cater to a wide array of needs, including mobile gaming, factory automation, GPS and location-based services, order tracking, ticket purchases, and much more. With countless applications available, many can be accessed and downloaded through digital distribution platforms known as app stores, often requiring internet connectivity for seamless functionality [3].

2.3 Comparison with the Existing Systems

The study of the existing system is carried out based on the system that has already been developed. In this study, three systems have been selected to be used as reference in the development of the system and used as a guide in improving the quality of UTHM Bus Tracking Mobile Application. There are three bus tracking system that are selected which is Pulse, Moovit, and Katsana. Can be summarized in Table 1 below the existing systems and the proposed system.

Table 1: Comparison of the system

Feature / System	Pulse	Moovit	Katsana	UTHM Bus Tracking Mobile Application
Platform	Mobile-based	Mobile-based	Web-based	Mobile-based
Login	Yes	No	No	Yes
Real-time bus location	Yes	No	Yes	Yes
Nearby bus stop suggestion	Yes	No	No	Yes
List of all bus stops	Yes	Yes	Yes	Yes
Detect the user's current location	Yes	Yes	No	Yes

Feature / System	Pulse	Moovit	Katsana	UTHM Bus Tracking Mobile Application
View the number of passengers	No	No	No	Yes
View bus plate number	Yes	No	No	Yes
Estimated bus arrival time	Yes	Yes	No	Yes
View bus route	Yes	Yes	Yes	Yes

3. Methodology

Waterfall Model is widely utilized as a Software Development Life Cycle (SDLC) in various fields. It is also known as the linear-sequential life cycle model. This user-friendly model is highly appreciated for its ease of use and understanding. The Waterfall model consists of five distinct phases: analysis, design, implementation, testing, and maintenance. Each phase must be completed before progressing to the next, ensuring a sequential and non-overlapping flow. The subsequent phase cannot commence unless the current phase is successfully concluded. The phases in the Waterfall model encompass the crucial stages of analysis, design, implementation, testing, and maintenance [4]. The development process of the Waterfall model is illustrated below:

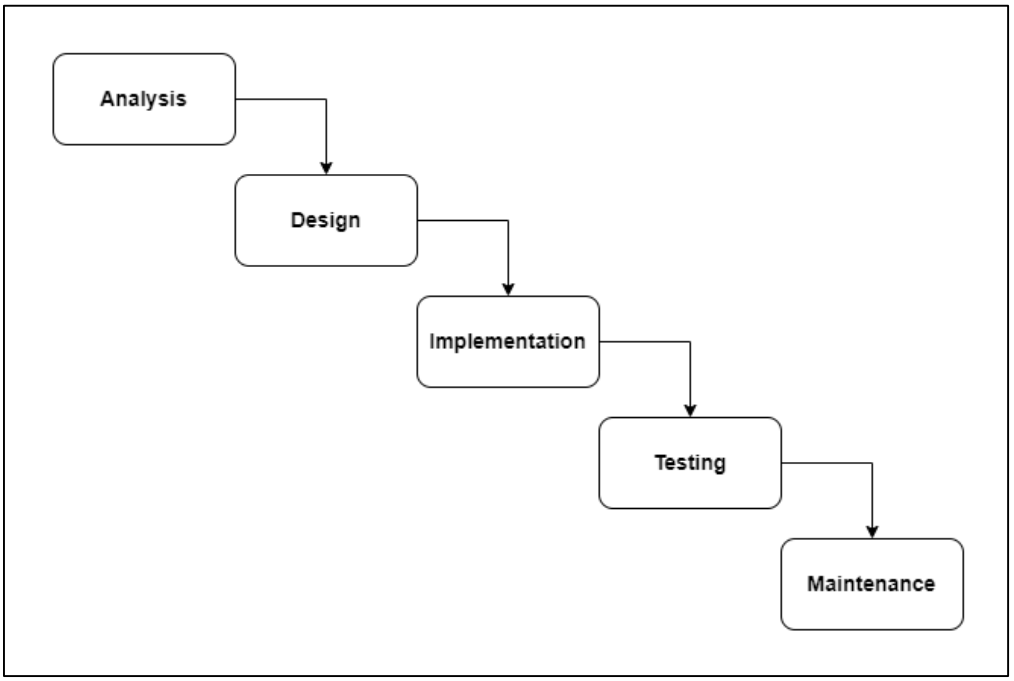


Figure 1: Waterfall development model

Table 2 shows the list of tasks performed at each phase in the Waterfall model for the UTHM Bus Tracking Mobile Application.

Table 2: Software development task and output

Phase	Task
Analysis	• Analyzed problem research. • Identify hardware and software requirements for the system development.
Design	• Design system prototype. • Design each module.
Implementation	• Developed interface for mobile application. • Developed the database.
Testing	• Test every function for each module. • Collect errors and defects.
Maintenance	• Check for any error not found during the testing phase.

3.1 Analysis Phase

Functional requirements are functions or parts of modules that developers need to implement to ensure that the system runs and completes its tasks. The functional requirement can be defined as how the system should be used and how it should work [5]. Table 3 presents the Functional Requirements that have been identified for the UTHM Bus Tracking Mobile Application. These requirements outline the specific functionalities expected from each module of the application.

Table 3: Functional Requirement

Module	Function
Login	• Allow the student and bus driver to log in using a UTHM email and password. • Display the contact us button which contains the detail of UTHM support for the user who forgot their password
Profile	• Display the details of the student such as name, matric number, level of study, year of study, faculty, phone number, gender, emergency contact, and account password. • Display the details of the bus driver such as name, staff id, phone number, emergency contact, and account password. • Allow the user to change their details such as emergency contact and account password.
Journey	• Display the suggestion of a bus stop near them according to the student's current coordinate. • Navigate the student to the nearest bus stop by using the google map application. • Allow the student to view the number of passengers and the bus estimated arrival time. • Allow the student to check in and check out of the bus.
Map	• Allow the student to view the real-time bus location.

Module	Function
	• Allow the student to view all the bus stops provided by the UTHM. • Allow the student to view the bus route used by the bus driver. • Display the UTHM area using google Maps.
Bus Status	• Allow the bus driver to view the GPS status. • Allow the bus driver to choose the plate number of the bus they going to drive. • Allow the bus driver to insert new bus plate numbers that are not recorded yet.
Passengers List	• Allow the bus driver to view the list of current students on the bus. • Allow the bus driver to view the student's name and their destination.

Non-functional requirements are different from the functional requirement because they specify how the system should work to meet user expectations. It describes the system's ability to improve user expectations when using the system [6]. Table 3 includes the Non-Functional Requirements that define the overall characteristics and quality of the application.

Table 4: Non-Functional Requirement

Requirement	Description
Performance	The reasonable operation and response time of the operating system should be expected.
Operational	The application should be able to work on the android platforms.
Usability	The general appearance and flow of the application are easy to understand by all types of users.
Security	The physical installation and from a cyber perspective are protected from an unauthorized party. The login module will verify the correct user account and if not verified the login information will be denied access to the application.
Integrity	The database of the application will be kept properly and secured by the system from any corruption and non-readable.

The use case diagram was developed as part of the analysis to show the overall functionality and component of the system [7]. It represents the methodology used in systems analysis to identify, clarify, and organize the system requirement of the UTHM Bus Tracking Mobile Application. The actor or users identified are students and bus driver, who implement various types of use cases. The major element of the use case diagram of the UTHM Bus tracking Mobile Application is shown in figure 2 below.

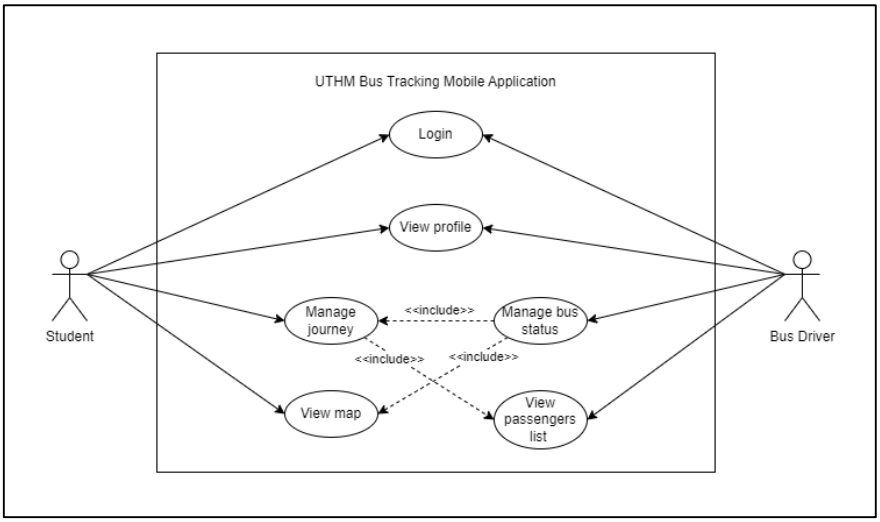


Figure 2: Use case diagram

Figure 3 presents a sequence diagram capturing the user's actions and interactions within the system. This diagram provides an overview of the chronological flow of communication and method calls between system components.

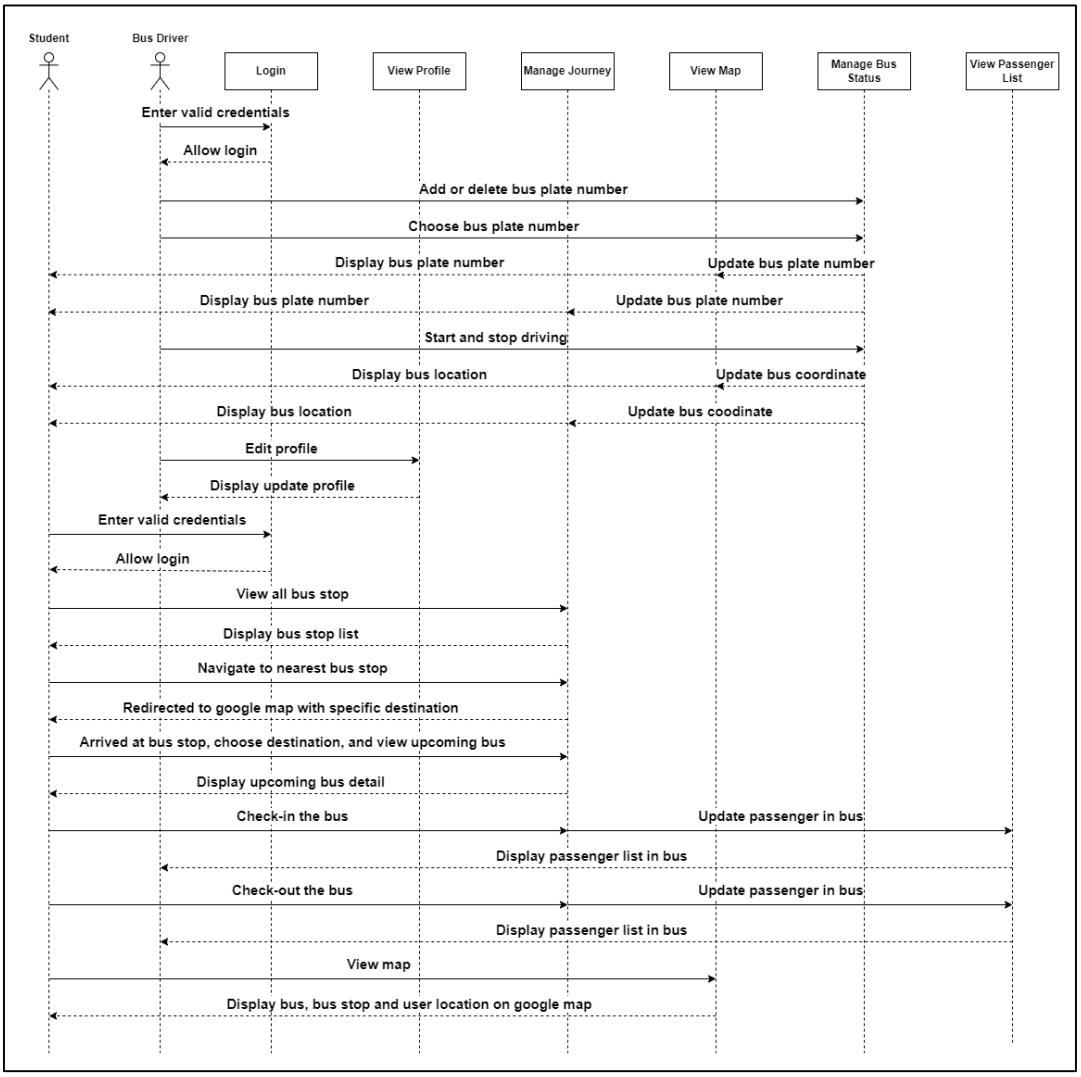


Figure 3: Sequence diagram

A class diagram needs to be created to determine the properties and methods to be used in the proposed application [8]. Figure 3 portrays the Class Diagram, which presents a structural view of the application by illustrating the different classes, their attributes, and their relationships. This diagram serves as a blueprint for the implementation and design of the application's objects and their interactions.

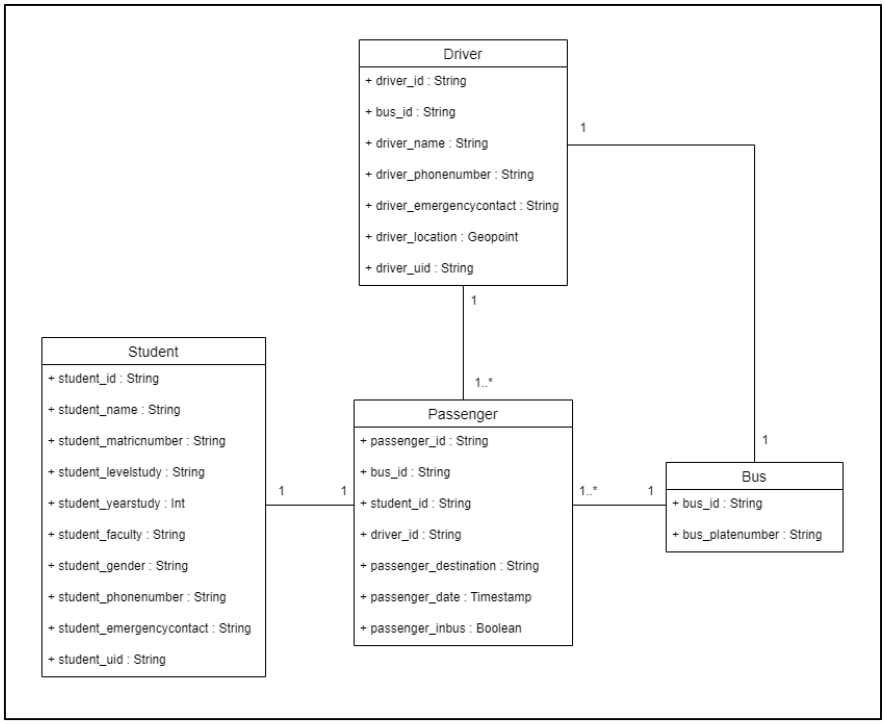


Figure 4: Class diagram

3.2 Design Phase

The design phase is a critical stage in software development, encompassing database design. Database design involves organizing and defining the system's data structure, ensuring efficient storage and retrieval. Tables 5 to 8 in this subtopic represent the data dictionary, defining student, driver, bus, and passenger information. These tables form the foundation for efficient data storage and retrieval within the UTHM Bus Tracking Mobile Application.

Table 5: Student database design

Attribute Name	Datatype	Size	Constraints		Description
			Null	Key	
student_id	String	50	No	PK	Student ID
student_name	String	50	No		Student full name
student_matricnumber	String	10	No		Student matric number
student_levelstudy	String	50	No		Student level of study
student_yearstudy	Int	10	No		Student year of study
student_faculty	String	50	No		Student faculty
student_gender	String	10	No		Student gender
student_phonenumber	String	50	No		Student phone number

Attribute Name	Datatype	Size	Constraints		Description
			Null	Key	
student_emergencycontact	String	50	No		Student emergency contact
student_uid	String	50	No		Student authentication code

Table 6: Driver database design

Attribute Name	Datatype	Size	Constraints		Description
			Null	Key	
driver_id	String	50	No	PK	Driver ID
bus_id	String	50	No	FK	Bus ID
driver_name	String	50	No		Driver full name
driver_phonenumber	String	50	No		Driver phone number
driver_emergencycontact	String	50	No		Driver emergency contact
driver_location	Geopoint	32	No		Driver location

Table 7: Bus database design

Attribute Name	Datatype	Size	Constraints		Description
			Null	Key	
bus_id	String	50	No	PK	Bus ID
bus_platenumber	String	10	No		Bus plate number

Table 8: Passenger database design

Attribute Name	Datatype	Size	Constraints		Description
			Null	Key	
passenger_id	String	50	No	PK	Passenger ID
bus_id	String	50	No	FK	Bus ID
student_id	String	50	No	FK	Student ID
driver_id	String	50	No	FK	Driver ID
passenger_destination	String	50	No		Passenger destination
passenger_date	Timestamp	10	No		Passenger date
passenger_inbus	Boolean	1	No		Passenger in the bus

3.3 Implementation Phase

The implementation phase marks the transformation of design concepts into tangible software components. This phase includes the development of source code and the creation of user interfaces. Figure 4 to Figure 19 provide snapshots of the source code and interface designs.

```dart
Future signIn() async {
    showDialog(
      context: context,
      builder: (context) {
        return Center(child: CircularProgressIndicator());
      },
    );
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _usernameController.text.trim(),
        password: _passwordController.text.trim(),
      );

      Navigator.of(context).pop();
    } catch (e) {
      Navigator.of(context).pop();
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            content: Text(
              'Invalid Credentials',
              textAlign: TextAlign.center,
            ),
          );
        },
      );
      setState(
        () {},
      );
    }
  }
```

Figure 5: Authenticate user login code segment

```dart
void updateMarkerPosition(Position position) {
  LatLng newPosition = LatLng(position.latitude, position.longitude);
  Marker updatedMarker = _markers[markerId]!;
  updatedMarker = updatedMarker.copyWith(positionParam: newPosition);
  _markers[markerId] = updatedMarker;
  setState(() {});
}
```

Figure 6: Update bus location on map code segment

```dart
void determineNearestBusStop() async {
  Position position = await getCurrentLocation();
  final userLatitude = position.latitude;
  final userLongitude = position.longitude;

  double minDistance = double.infinity;
  String nearestStopName = '';

  for (var busStop in busStops) {
    final distance = calculateDistance(
        userLatitude, userLongitude, busStop.latitude, busStop.longitude);
    if (distance < minDistance) {
      minDistance = distance;
      nearestStopName = busStop.name;
    }
  }

  setState(() {
    nearestBusStop = nearestStopName;
    selectedBusStopIndex =
        busStops.indexWhere((busStop) => busStop.name == nearestStopName);
  });
}
```

Figure 7: Determine the nearby bus stop code segment

```dart
Future<Position> getCurrentLocation() async {
  LocationPermission permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied ||
      permission == LocationPermission.deniedForever) {
    permission = await Geolocator.requestPermission();
    if (permission != LocationPermission.whileInUse &&
        permission != LocationPermission.always) {
      // Handle location permission denied
      return Future.error('Location permission denied');
    }
  }

  Position position = await Geolocator.getCurrentPosition(
    desiredAccuracy: LocationAccuracy.high,
  );

  return position;
}
```

Figure 8: Detect user current location code segment

```
Stream<int> getPassengerCountStream() {
  return FirebaseFirestore.instance
      .collection('Passenger')
      .where('passenger_inbus', isEqualTo: true)
      .snapshots()
      .map((QuerySnapshot snapshot) => snapshot.size);
}
```

Figure 9: Get number of passengers in bus code segment

```
String busPlateNumber = '';
Future<String> getBusPlateNumber() async {
  DocumentSnapshot snapshot = await FirebaseFirestore.instance
      .collection('Driver')
      .doc('driver1')
      .get();
  if (snapshot.exists) {
    busPlateNumber = snapshot['bus_id'] ??
        ''; // Assign the value directly to the outer variable
    return busPlateNumber;
  } else {
    return '';
  }
}
```

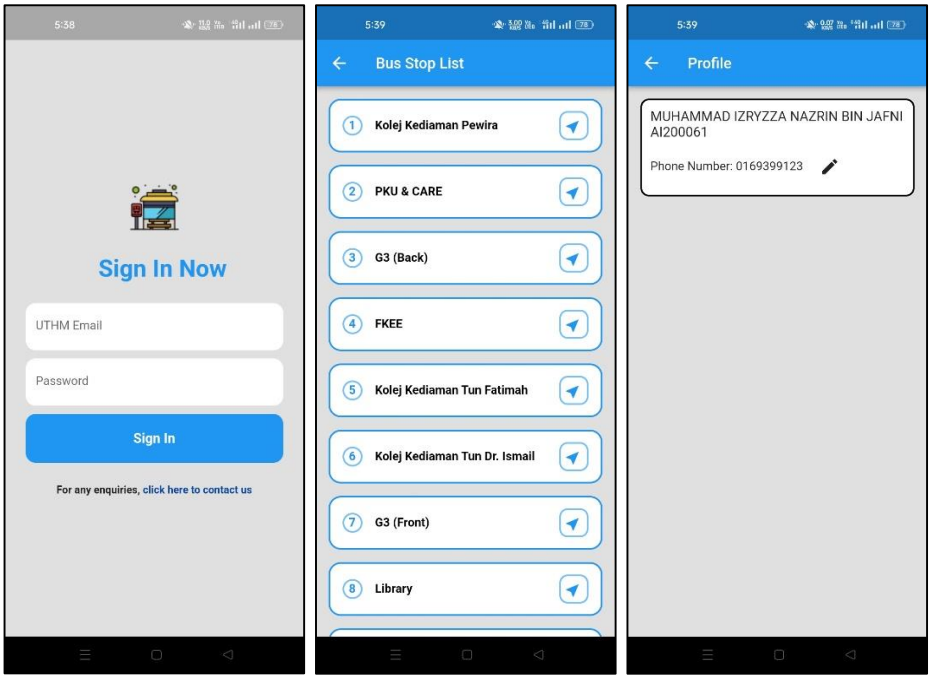
Figure 10: Get bus plate number code segment

```
double calculatePolylineDistance(List<LatLng> polylinePoints) {
  double totalDistance = 0.0;
  LatLng previousPoint = polylinePoints[0];

  for (int i = 1; i < polylinePoints.length; i++) {
    LatLng currentPoint = polylinePoints[i];
    double segmentDistance = distanceBetween(previousPoint, currentPoint);
    totalDistance += segmentDistance;
    previousPoint = currentPoint;
  }

  return totalDistance;
}

double calculateRemainingTime(double distance) {
  double busSpeed = 10.0;
  return distance / busSpeed;
}
```

Figure 11: Get estimated bus arrival time code segment

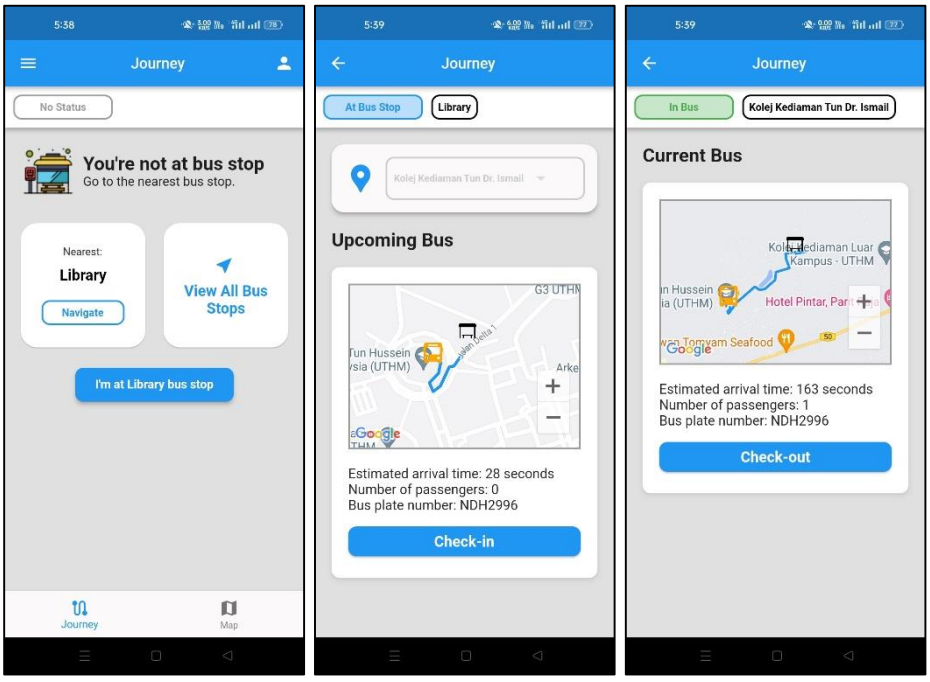
```
PolylineResult result17 = await polylinePoints.getRouteBetweenCoordinates(
  "API_KEY",
  PointLatLng(bustop17.latitude, bustop17.longitude),
  PointLatLng(bustop18.latitude, bustop18.longitude),
);

if (result17.points.isNotEmpty) {
  result17.points.forEach(
    (PointLatLng point) => polylineCoordinate.add(
      LatLng(point.latitude, point.longitude),
    ),
  );
  setState(() {});
}
```

Figure 12: Display bus route code segment



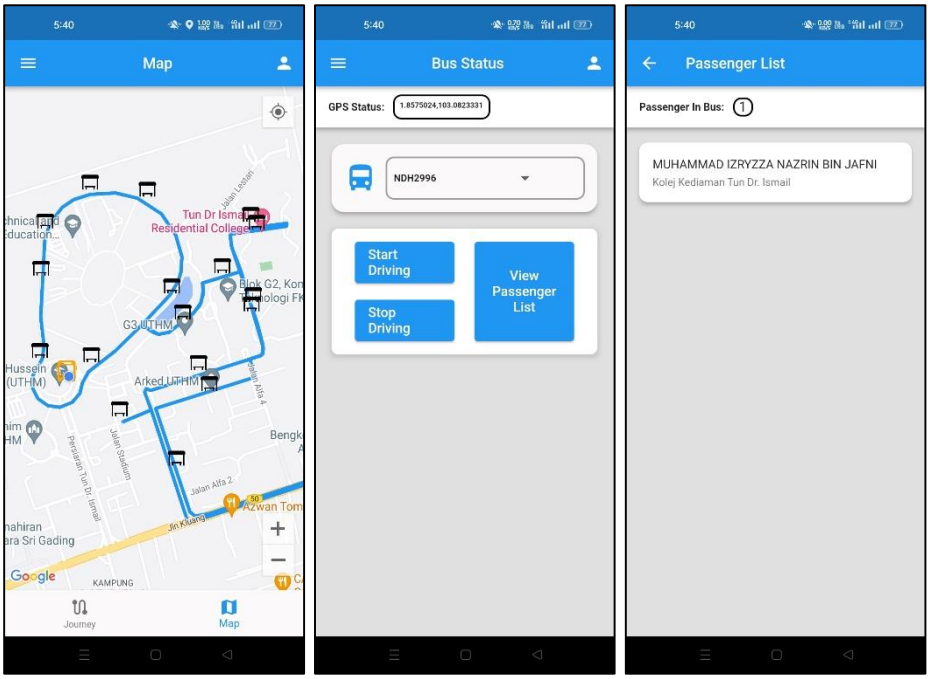
(a) **(b)** **(c)**

Figure 13(a): Login interface, Figure 13(b): Bus stop list interface, Figure 13(c): Profile interface



(a) **(b)** **(c)**

Figure 14(a): First journey interface, Figure 14(b): Second journey interface, Figure 14(c): Third journey interface



(a) **(b)** **(c)**

Figure 15(a): Map interface, Figure 15(b): Bus status interface, Figure 15(c): Passenger list interface

4. Results and Discussion

The system testing phase encompasses the functional modules of the system, involving both student and bus drivers. This crucial testing procedure ensures the achievement of the project's objectives and validates the fulfillment of all system requirements. Through rigorous evaluation, we ascertain that the system operates seamlessly, meeting the desired goals and delivering a satisfactory user experience.

4.1 Test Case

Test case is a step or action that is performed on the system to determine if the system satisfies the functional requirement and user need. The document is mainly focused on separating all the different modules and features in the system. The goal is to verify all the function of the system and code behavior by optimize the effort and time to test the system.

Table 8: Login module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_100_001	Enter valid UTHM email and password.	The system allow user to login to the system.	The system allow user to login to the system.	Pass
TC_100_002	Enter invalid UTHM email and password.	The system display invalid credential.	The system display invalid credential.	Pass
TC_100_003	Press contact us.	The system display contact information pop-up.	The system display contact information pop-up.	Pass

Table 9: View profile module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_200_001	Press profile button.	The system display user data according to the database.	The system display user data according to the database.	Pass
TC_200_002	Press edit button.	The system allow user to edit selected data.	The system allow user to edit selected data.	Pass

Table 10: Manage journey module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_300_001	Press navigate button.	The system redirected to the google map with a specific destination.	The system redirected to the google map with a specific destination.	Pass
TC_300_002	Press view all bus stop.	The system display list of all bus stops available.	The system display list of all bus stops available.	Pass
TC_300_003	Select destination.	The system display the upcoming bus card with its detail.	The system display the upcoming bus card with its detail.	Pass
TC_300_004	Press check-in button.	The system display current bus card with its detail.	The system display current bus card with its detail.	Pass
TC_300_005	Press check-out button.	The system should redirected user to the homepage.	The system should redirected user to the homepage.	Pass

Table 11: View map module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_400_001	Press on the bus marker.	The system display bus plate number on the marker.	The system display bus plate number on the marker.	Pass
TC_400_002	Press on the certain bus stop marker.	The system display the bus stop name for	The system display the bus stop name for	Pass

Test Case	Description	Expected Result	Actual	Pass/Fail
		the specific bus stop pressed.	the specific bus stop pressed.	

Table 12: Manage bus status module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_500_001	Press add button.	The system allow the user to insert new bus plate number to the database.	The system allow the user to insert new bus plate number to the database.	Pass
TC_500_002	Press delete button.	The system allow the user to delete selected bus plate number from the database.	The system allow the user to delete selected bus plate number from the database.	Pass
TC_500_003	Select bus plate number.	The system display the start and stop driving button.	The system display the start and stop driving button.	Pass

Table 13: View passenger list module test plan

Test Case	Description	Expected Result	Actual	Pass/Fail
TC_600_001	Press passenger list button.	The system display the number and list of passenger with its destination.	The system display the number and list of passenger with its destination.	Pass

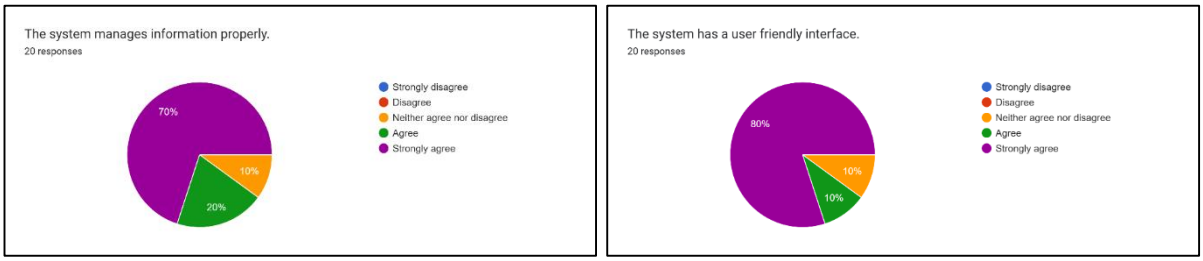
Table 14 shows the overall result of the test cases. The overall result will detail the test cases modules, number of test cases, and total success and fail test cases.

Table 14: Overall test case result

Test Case Modules	**Number of Test Cases**	**Total Success Test Case**	**Total Fail Test Case**
TC_100	3	3	0
TC_200	2	2	0
TC_300	5	5	0
TC_400	2	2	0
TC_500	3	3	0
TC_600	1	1	0

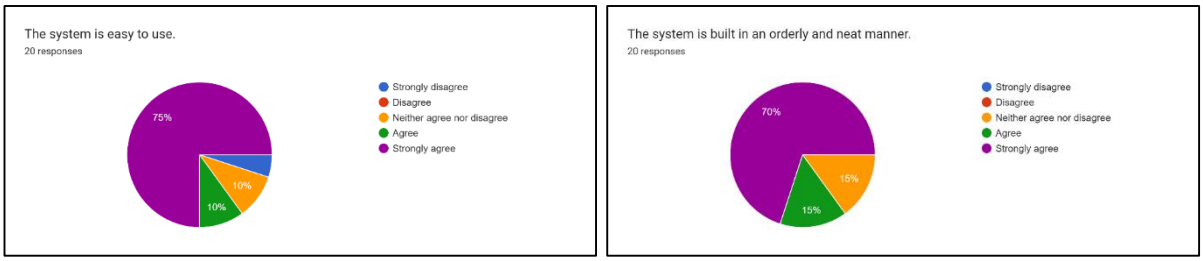
4.2 User Acceptance Testing

User acceptance testing is a phase to determine if the system is applicable in the real world. The requirements for the system must be accepted by the user who wants to use the system. It is mostly performed before the system is deployed to the end user or the market. User acceptance testing allows the user to interact with the system even before the final release of the system. To gather user feedback, a questionnaire consisting of 10 questions was conducted through a Google Form, with a total of 20 users providing their responses. Figures 16 to 20 were included in the questionnaire, and the feedback received from the users indicated a positive response towards the mobile application.



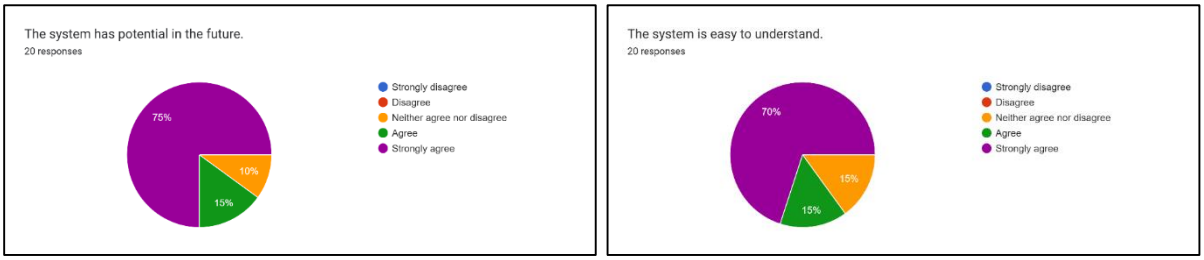
(a) **(b)**

Figure 16(a): Result for question 1, Figure 16(a): Result for question 2



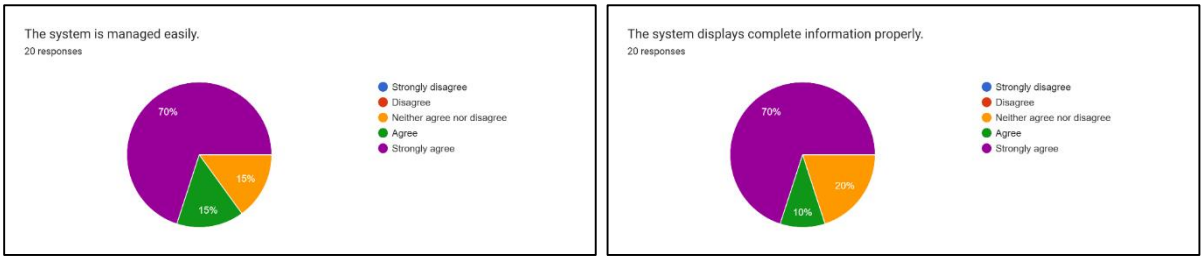
(a) **(b)**

Figure 17(a): Result for question 3, Figure 17(a): Result for question 4



(a) **(b)**

Figure 18(a): Result for question 5, Figure 18(a): Result for question 6



(a) **(b)**

Figure 19(a): Result for question 7, Figure 19(a): Result for question 8

(a) **(b)**

Figure 20(a): Result for question 9, Figure 20(a): Result for question 10

5. Conclusion

In conclusion, the development of the UTHM Bus Tracking Mobile Application using an object-oriented approach has successfully addressed the challenges faced by UTHM students in tracking the location of the campus bus. By leveraging the Flutter framework and Firebase server, the application enables students to track bus locations, access real-time information, and plan their journeys effectively. The system's implementation follows the systematic Waterfall model, ensuring a sequential development process. The application's key findings demonstrate improved efficiency in public transportation, enhancing students' overall bus transportation experience. The significance of this mobile-based system in students' daily lives is highlighted, along with potential avenues for future enhancements. With its successful implementation and positive impact, the UTHM Bus Tracking Mobile Application serves as a valuable tool for optimizing bus system efficiency and improving the overall transportation experience for UTHM students.

Acknowledgment

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

References

[1] J. Prasetijo, W. Z. Musa, Z. F. Zainal, K. Ambak, and M. E. Sanik, "Level of Bus Performance Based on the Relationship between Distance and Travel Time of Universiti Tun Hussein Onn Malaysia (UTHM) Bus Services," in MATEC Web of Conferences, vol. 87, p. 02006, EDP Sciences, 2017.

[2] L. Singla and P. Bhatia, "GPS based bus tracking system," in 2015 International Conference on Computer, Communication and Control (IC4), pp. 1-6, IEEE, September 2015.

[3] F. Nayebi, J. M. Desharnais, and A. Abran, "The state of the art of mobile application usability evaluation," in 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1-4, April 2012.

[4] A. A. Adenowo and B. A. Adenowo, "Software engineering methodologies: a review of the waterfall model and object-oriented approach," International Journal of Scientific & Engineering Research, vol. 4, no. 7, pp. 427-434, 2013.

[5] R. Malan and D. Bredemeyer, "Functional requirements and use cases," Bredemeyer Consulting, 2001.

[6] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, "Non-functional requirements in software engineering," vol. 5, Springer Science & Business Media, 2012.

[7] A. Y. Aleryani, "Comparative study between data flow diagram and use case diagram," International Journal of Scientific and Research Publications, vol. 6, no. 3, pp. 124-126, 2016.

[8] H. Herchi and W. B. Abdessalem, "From user requirements to UML class diagram," arXiv preprint arXiv:1211.0713.