9/21/2023

Rifat Bin Saleh

IT Bangladesh

Software Requirements Specification (SRS)

DIU Transport

* Table of Contents

[Software Requirements Specification 4](#_Toc151865327)

[1.Introduction 5](#_Toc151865328)

[1.1. Overall Description 5](#_Toc151865329)

[2. Obstacles 5](#_Toc151865330)

[3. User Features in Details 5](#_Toc151865331)

[4. System Requirements 7](#_Toc151865332)

[4.1. The app must be compatible with the following devices: 7](#_Toc151865333)

[4.2. The app will require access to the following device features: 7](#_Toc151865334)

[5. Functional Requirements 7](#_Toc151865335)

[6. Non-Functional Requirements 7](#_Toc151865336)

[7. Technology Require (Application and Hardware) 8](#_Toc151865337)

[8. Use Cases 8](#_Toc151865338)

[9. Acceptance Criteria 8](#_Toc151865339)

[10. Schedule 8](#_Toc151865340)

[11. Resources 9](#_Toc151865341)

[12. Risks 9](#_Toc151865342)

[13. Preliminary Schedule 9](#_Toc151865343)

[14. Pricing 10](#_Toc151865344)

[15. Deployment Plan 10](#_Toc151865345)

[16. Maintenance Plan 10](#_Toc151865346)

[17. Approvals 10](#_Toc151865347)

[18. Changes 10](#_Toc151865348)

[19. Responsibility 11](#_Toc151865349)

[20. Contact 11](#_Toc151865350)

[Software Development Life Cycle 12](#_Toc151865351)

[1. SDLC 12](#_Toc151865352)

[2. Software development life cycle (SDLC) models 12](#_Toc151865353)

[3. Brief overview of each of the eight sdlc models: 12](#_Toc151865354)

[3.1. Waterfall Model 12](#_Toc151865355)

[3.2. Spiral Model 12](#_Toc151865356)

[3.3. V-model 13](#_Toc151865357)

[3.4. Incremental 13](#_Toc151865358)

[3.5. Rapid Application Development (RAD) 13](#_Toc151865359)

[3.6. Iterative 13](#_Toc151865360)

[3.7. Agile 13](#_Toc151865361)

[3.8. Big Bang 14](#_Toc151865362)

[4. Best SDLC Model for the University Transport System App 14](#_Toc151865363)

[4.1 The following are some of the reasons why I recommend Agile for this project 14](#_Toc151865364)

[4.2. Compare to other SDLC Models 14](#_Toc151865365)

[5. Agile Model 15](#_Toc151865366)

[5.1. Here is a summary of the different phases of the Agile SDLC model 16](#_Toc151865367)

[5.2. How to Implement Agile for the University Transport System App 16](#_Toc151865368)

[6. Conclusion 17](#_Toc151865369)

[Use Case Diagram 18](#_Toc151865370)

[UML Diagram 19](#_Toc151865371)

[Flow-Chart Diagram: 20](#_Toc151865372)

[Testing Approach 21](#_Toc151865373)

[1.Testing Approach 21](#_Toc151865374)

[1.1. Testing Approach: Black Box Testing 21](#_Toc151865375)

[1.2. Compare 22](#_Toc151865376)

[1.3. Choice of Black Box Testing 23](#_Toc151865377)

[2.Testing levels 23](#_Toc151865378)

[3.Testing types, techniques and tactics 24](#_Toc151865379)

[4.Testing process 26](#_Toc151865380)

[5.Measurement in software testing 27](#_Toc151865381)

[5.1. Hierarchy of Testing Difficulty 27](#_Toc151865382)

[5.2. Test Plan 28](#_Toc151865383)

[5.3. Test Cases 28](#_Toc151865384)

[1. Softwires 30](#_Toc151865385)

[1.1. Back-end Development 30](#_Toc151865386)

[1.2.Front-end Development for Website 31](#_Toc151865387)

[1.3.Front-end Development for Mobile App 31](#_Toc151865388)

[1.4.Additional Security Considerations 32](#_Toc151865389)

[2. Hardware 32](#_Toc151865390)

[2.1.Back-end (Server) 32](#_Toc151865391)

[2.2.Database 33](#_Toc151865392)

[2.3.Front-end (Web Server) 33](#_Toc151865393)

[2.4.Mobile App (Backend and Frontend) 33](#_Toc151865394)

[2.5.Additional Considerations 33](#_Toc151865395)

[Scheduling and Estimates 34](#_Toc151865396)

Software Requirements Specification

MD Arafat

Managing Director

DIU Transport System

Daffodil smart city, Ashulia,Savar,

Dhaka,Bangladesh.

Dear M D Arafat,

Introducing the Software Requirements Specification (SRS) for the University Transport System app gives me great pleasure. The features, specifications, and acceptability standards for the app are described in this document.

Students, professors, and staff have an easy and effective way to navigate around campus thanks to the University Transport System app, a smartphone application. Users of the app can plan their excursions, view projected arrival times, and track buses in real-time. The app also offers details on alternative modes of mobility on campus, such bike rentals and ride-sharing services.

In order to guarantee that the University Transport System app satisfies the demands of all users, the SRS is a crucial document. I urge you to evaluate the SRS and offer any comments or recommendations.

Please reply to this email with your signature in order to approve the SRS. Please get in touch with me if you have any inquiries or worries.

I appreciate you taking the time to read this.

Yours Truly,

Rifat Bin Saleh

Head

Application Development Brach

IT Bangladesh

**Product Name:** **DIU Transport**

**Version: 1.0**

**Date: 2023-09-21**

# 1.Introduction

This SRS document outlines the software specifications for a mobile application that will give Daffodil International University students, faculty, and staff access to the campus transportation system. The app will allow users to:

* View real-time bus arrival times and locations
* Map out their route and keep along with their bus in real time.
* Purchasing and controlling tickets
* Receive notifications about service changes and disruptions
* Express criticism on the transportation system.

## 1.1. Overall Description

The University Transport System App will allow users to:

* Check out current bus arrival and departure data
* Monitor the whereabouts of buses on a map.
* Determine routes and get travel times
* Buy bus tickets and passes
* File complaints about busses or bus stops

# 2. Obstacles

* Lack of Clear Requirements
* Data Security and Privacy
* Manage Storage File System

# 3. User Features in Details

Detailed user features of the DIU University Transport System App are provided here.

1. **User:**
* Registration and Login
* User Id and Password are provided by admin.
* View Routes and Schedules
* Tracking buses in real time
* Buy tickets
* Cancel tickets
* Booking History
* Payment Status
* Payment History
* Download Receipt
* Emergency Contact
* Notifications
* Comments/Issues
1. **Driver:**
* Sign-in and accessibility:
* User Id and Password are provided by admin.
* View Assigned Routes:
* Accept/Reject Ride Requests:
* Navigation in real-time:
* Status Update:
* View Ride History:
1. **Helper:**
* Sign-in and accessibility
* User Id and Password are provided by admin.
* View Assigned Routes:
* Verify payment receipt.
* Help Passengers:
* Status Update:
* View tasks History:
1. **Admin:**
* Authentication of Users
* Create User
* Manage Routes and Schedules
* Payment Management
* Assigning Drivers and Helpers
* Tracking System Activity
* Computerized Receipt
* Emergency Management
* Handle Complaints

# 4. System Requirements

## 4.1. The app must be compatible with the following devices:

* iOS 12 or later-running devices
* Android-powered gadgets with Android 6 or higher

## 4.2. The app will require access to the following device features:

* GPS location
* Push notifications
* Internet access

# 5. Functional Requirements

The University Transport System App's functional criteria are as follows:

* **User authentication:** Customers must be able to download the app, establish an account, and log in.
* **Real-time bus arrival and departure information:** Users must be able to view this information for all bus stops on campus.
* **Map-based bus tracking:** Users must be able to follow the whereabouts of buses.
* **Route planning:** Users must be able to determine travel times and routes between bus stops.
* **Ticket purchase:** The app must allow users to buy bus passes and tickets.
* **Problem reporting:** Users must be able to use the app to report issues with buses or bus stops.

# 6. Non-Functional Requirements

The University Transport System App doesn't need the following things in order to function:

* **Performance:** The application needs to be able to manage numerous simultaneous users and requests.
* **Security:** The application must prevent unwanted access to user data.
* **Reliability:** The app needs to be dependable and always accessible to users.
* **Usability:** The app's navigation and use must be simple.

# 7. Technology Require (Application and Hardware)

* **Framework:** React.js.
* **Database:** : MongoDB.
* **Design:** Standard.
* **Coding Architecture:** OOP.
* **Security:** Standard.

# 8. Use Cases

The University Transport System App can be used in the following situations:

* A student wants to know when the next bus to their class will arrive.
* A staff member needs to map out a path to an off-campus meeting.
* A faculty member has to buy a semester-long bus pass.
* A student wishes to voice concern over a bus stop.

# 9. Acceptance Criteria

The University Transport System App will be accepted when it meets the following criteria:

* All requirements, both functional and non-functional, are satisfied.
* All use cases have been successfully implemented and tested.
* The app works with all of the main mobile operating systems.
* Users with disabilities can use the app.

# 10. Schedule

In Q4 2023, the University Transport System app is expected to go live.

# 11. Resources

The following resources will be required to develop and maintain the University Transport System app:

* 3 developers
* 1 QA engineer
* 1 project manager

# 12. Risks

The following risks have been identified for the University Transport System project:

• The project may go over budget due to unanticipated expenses; it may be delayed because of unexpected technical difficulties.

• If the app does not satisfy users' needs, it might not be warmly received by them.

# 13. Preliminary Schedule

|  |  |  |
| --- | --- | --- |
| **Milestone**  | **Reporting**  | **Time**  |
| Analysis   | Submit The Design  | 4 days  |
| Requirements Collection  |   | 10 days  |
| Development  | Review The Work  | 40 days  |
| Testing  |   | 7 days  |
| Deployment  | Review Final Work | 5 days  |
| Delivery  | Live On Server  | 3 days  |

# 14. Pricing

The project will cost one lakh taka (100000Tk) to complete from start to finish.

# 15. Deployment Plan

The University Transport System App will be available on both the Google Play Store and the Apple App Store. The app will be available for free download.

# 16. Maintenance Plan

The University Transport System App will be maintained by the university's IT division. The IT department will have to update the program with new features and solve faults,bug fixes,security updates.

# 17. Approvals

This SRS must be approved by the following stakeholders:

* University CIO
* University Transportation Director
* Student Government Association President
* Faculty Senate President

# 18. Changes

The above-mentioned stakeholders must consent to any changes to this SRS.

# 19. Responsibility

All responsibility for this application rests with **Rifat Bin Saleh,** Head of IT Bangladesh.

# 20. Contact

You can get in touch with us in any of the below ways:

By Phone:

+8801627248153

By Email:

aywon15-4846@diu.edu.bd

**Agreement Signed By:**

|  |  |  |
| --- | --- | --- |
| **……………………………** **Client Signature** **Rifat Bin Saleh****Head****Application Development Brach****IT Bangladesh** | **…………………………..** **Order Provider Signature****Akber Ahmed****Officer****IT Bangladesh** | **………………………****Authority Signature****MD Arafat****Managing Director****DIU Transport System****Daffodil smart city, Ashulia,Savar,****Dhaka,Bangladesh.** |

Software Development Life Cycle

# 1. SDLC

The Software Development Life Cycle (SDLC) is a conceptual framework for project management that outlines the steps of an information system development project, from the first phase of a feasibility study to the ongoing support of the finished application.

# 2. Software development life cycle (SDLC) models

1. Waterfall
2. Spiral
3. V-model
4. Incremental
5. Rapid Application Development (RAD)
6. Iterative
7. Agile
8. Big bang

# 3. Brief overview of each of the eight sdlc models:

## **3.1. Waterfall Model**

A method that is linear and sequential and requires that each stage be finished before the next one can start. Despite being simple to manage, it isn't flexible. A conventional and sequential method of software development is the waterfall model. The development process is broken down into separate phases in this approach, and each phase must be finished before going on to the next. From the beginning of the project to its completion, it takes a straight and organized path.

## 3.2. ****Spiral Model****

combines iterative development with Waterfall model components. It places a strong emphasis on risk analysis and permits adjustments at different stages. The Spiral Model is a software development methodology that combines aspects of waterfall and iterative models with a significant emphasis on project risk management. It was created by Barry Boehm in the 1980s and is especially beneficial for complicated, large-scale undertakings.

## 3.3. V-model

A linear paradigm that links every stage of development to testing. It makes certain that testing is incorporated into every stage of the development process. The Verification Model, often known as the V-Model, is a software development and testing strategy that places an emphasis on the connection between the development and testing phases. The Waterfall model is frequently seen as being extended, but with a significant emphasis on validation and verification tasks.

## 3.4. Incremental

Software development uses an iterative process called the incremental model. The job is broken down into manageable, little pieces or increments using this methodology. A fraction of the functionality of the entire system is represented by each increment. Each incremental development and delivery builds on the functionality of the preceding ones. Increments are produced and provided one at a time.

## 3.5. Rapid Application Development (RAD)

Rapid prototyping and prompt end-user response are key components of RAD. It emphasizes on creating software prototypes or components that are eventually combined to create the finished solution. A specialized method within the Software Development Life Cycle (SDLC) called rapid application development (RAD) is designed to build and deploy software applications quickly. Rapid development and iterative design are made possible by RAD's emphasis on speed and flexibility.

## 3.6. Iterative

The project is divided into more manageable pieces using the iterative technique. Each iteration passes through the development cycle and involves a smaller set of criteria. Up until the entire system is built, iterations are performed repeatedly. The term "iterative" describes a method of project development in which a larger project is divided into smaller cycles or iterations, each of which focuses on a different aspect of the project. Iterative approaches are frequently employed in a variety of disciplines, including product design, project management, and software development.

## 3.7. Agile

Agile is an incremental and iterative method of software development. It places a strong emphasis on customer input, teamwork, and flexibility. Projects are broken down into manageable iterations called sprints by agile methodologies like Scrum and Kanban. Agile is a project management and software development methodology that places a high value on adaptability, teamwork, customer satisfaction, and quick iterations. It was developed as an alternative to conventional, plan-driven project management approaches and is distinguished by a number of ideas expressed in the Agile Manifesto.

## 3.8. Big Bang

There is no defined or structured process in the Big Bang theory. Without clear requirements or strategy, development starts. Until the project is deemed finished, developers continue to work on it. When discussing software testing and development, the term "Big Bang" does not refer to the cosmological hypothesis on the universe's creation. Instead, it alludes to the "Big Bang Testing" method, a particular kind of software testing strategy.

# 4. Best SDLC Model for the University Transport System App

The Agile SDLC is a suitable Software Development Life Cycle (SDLC) model for the " University Transport System App " mobile application project because it is flexible and adaptable, which is necessary for a project with complex and changing needs. Agile also enables the early and frequent release of functional software, which will enable the university to receive input from both staff and students. Agile is a team-based methodology that encourages interaction and input among stakeholders, project managers, QA engineers, and developers.

## 4.1 The following are some of the reasons why I recommend Agile for this project

1. **Customer-Centric:** Agile puts a big emphasis on customer satisfaction by including them in every step of the development process, making sure the finished product satisfies their wants and expectations.
2. **Flexibility:** Agile is very adaptive to changing requirements, making it appropriate for projects where requirements are hazy or likely to change, like the creation of mobile apps.
3. **Continuous Improvement:** Agile encourages a culture of ongoing improvement through frequent retrospectives, which results in improvements to the end product and the development process.
4. **Faster Time-to-Market:** Agile's incremental and iterative approach enables shorter time-to-market for the product by enabling speedier delivery of functional increments.
5. **Risk Mitigation:** Agile supports early risk identification and mitigation through iterative development and testing, lowering the likelihood that significant problems will surface later in the project.
6. **Higher Quality:** Agile practices encourage frequent testing and feedback loops, which result in better software and fewer errors.
7. **Transparency and Collaboration:** Agile encourages open dialogue and teamwork among stakeholders, customers, and team members, increasing project openness and team cohesiveness.

## 4.2. Compare to other SDLC Models

1. **Waterfall Model:**
	* **Rigidity:** Waterfall's sequential nature makes it inflexible to changes, which can be problematic for projects with evolving requirements.
	* **Limited User Involvement:** It lacks ongoing customer involvement, potentially resulting in a product that doesn't meet user expectations.
2. **Spiral Model:**
	* **Complexity:** The Spiral model can be complex to manage, and its formal risk analysis may not be necessary for all projects.
	* **Higher Costs:** Managing risk through multiple iterations can increase project costs.
3. **V-Model:**
	* **Rigidity:** Similar to Waterfall, V-Model can be inflexible to changes in requirements.
	* **Limited Adaptability:** It may not handle evolving requirements or late-stage changes effectively.
4. **Big Bang Model:**
	* **Lack of Structure:** Big Bang lacks a formalized development process, which can lead to chaotic and uncontrolled development.
5. **Incremental Model:**
	* **Integration Challenges:** Integration of increments can be challenging and may lead to compatibility issues.
6. **RAD Model (Rapid Application Development):**
	* **Resource Intensive:** RAD may require a significant amount of resources and expertise.
	* **Not Suitable for All Projects:** It's not ideal for projects with very tight budgets or strict time constraints.
7. **Iterative Model:**
	* **Limited User Involvement:** Like some other models, it may not prioritize ongoing customer collaboration.

# 5. Agile Model




## 5.1. Here is a summary of the different phases of the Agile SDLC model

1. **Planning:** During the planning phase, the team develops a project roadmap. The project's aims, objectives, and schedule are described in this road plan.
2. **Requirements gathering:** During the analysis stage, the team compiles information on the requirements of the client. The features and functionality of the software are decided by this data.
3. **Design:** The group develops a software prototype during the design phase. This prototype evaluates the viability of the concept and elicits user input.
4. **Development or coding**: The developers examine the requirements and divide them into manageable, quick-and-efficient activities. The group creates the software's code throughout the development phase.
5. **Testing:** The software is tested during the testing phase to make sure it satisfies the needs of the client. The group fixes bugs when they are discovered.
6. **Implementation:** After passing the testing phase, the software is prepared for implementation. At this point, the team assists the client with installing and using the program.
7. **Maintenance:** Following the installation and activation of the program, the team offers maintenance and assistance. Additionally, it guarantees that the software is updated as required.

## 5.2. How to Implement Agile for the University Transport System App

1. **Define the project scope and vision:** Establish the project's scope and vision by consulting with stakeholders to ascertain the needs of the app and their desired outcomes.
2. **Prioritize the requirements:** Once the requirements have been established, they need to be ranked in order of importance so that the most crucial features may be created first.
3. **Create a backlog:**The backlog is a list of all the app's requirements, ranked in priority.
4. **Break down the requirements into smaller tasks:** The requirements must be divided into manageable activities that can be finished in a sprint.
5. **Plan the sprint:** A sprint is a short, usually two-week period of time during which a team works to complete a set of tasks.
6. **Execute the sprint:**During the sprint, the team strives to finish the planned tasks.
7. **Review the sprint work:** The team evaluates the work that was finished and receives input from stakeholders at the conclusion of the sprint.
8. **Repeat steps 5-7 until the project is complete:** When all of the requirements have been implemented and tested, the project is finished.

# 6. Conclusion

While the Agile model has numerous benefits, like flexibility, customer-centricity, and continuous improvement, many other conventional SDLC models could have drawbacks, including rigidity, lack of adaptation, and minimal user participation. The requirements and characteristics of the project should guide the selection of the SDLC model.

Use Case Diagram



UML Diagram



Flow-Chart Diagram:



Testing Approach

# 1.Testing Approach

I tested my university transportation apps using black box techniques. These are the specifics of it.

## Testing Approach: Black Box Testing

1. **Functional Testing:**

* **User Module:** - Check the functionality of user registration and login. Examine the functionality of viewing schedules, routes, and real-time bus tracking. Verify that ticket purchases and cancellations go according to plan. Examine the payment status, booking history, and features related to the payment history. - Check the emergency contact and download receipt features. - Check the features for notifications and issues/comments.
* **Driver Module:** Verify accessibility and sign-in using the given credentials. - Confirm that you can see the designated routes and approve or reject ride requests. - Assure status updates and real-time navigation functionality. Examine the function to view ride history.
* **Helper Module:** Verify accessibility and sign-in using the given credentials. - Confirm that you can see the designated routes, check payment receipts, and help passengers. - View the history of tasks and check status updates.
* **Admin Module:** - Manage routes and schedules, test user authentication, and create users. - Check the ability to assign drivers and helpers, as well as payment management. - Examine the computerized receipt system, emergency response, complaint handling, and activity tracking.

2. **Non-Functional Testing:**

* **Performance Testing:** - Check the program's capacity to manage several users at once. Verify the turnaround times for different tasks.
* **Security Testing:** - Confirm that there is no unauthorized access to and that user data is secure. Examine the authentication procedure for weaknesses.
* **Reliability Testing:** Examine the availability and dependability of the application in various scenarios. - Look for any outages or crashes in the system.
* **Usability Testing:** - Assess the user interface and navigation in general. Make sure that users with disabilities can utilize the app.

3. **Compatibility Testing:**

* Try the app on devices running iOS 12 or later. b. Use Android 6 or higher-powered devices to test the app.

4. **Integration Testing:**

* Check that the app's features for internet access, push notifications, and GPS location are integrated.

5. **Regression Testing:**

* Regression testing should be done following every update or modification to make sure that all features continue to function as intended.

6. **User Acceptance Testing (UAT):**

* Engage stakeholders in UAT, such as the president of the Faculty Senate, the director of transportation, the university's CIO, and the SGA president. b. Verify that every use case is successfully tested and put into practice.

7. **Deployment Testing:**

* Check the app's release on the Apple App Store and Google Play Store. b. Confirm that downloading the app is free of charge.

8. **Maintenance Testing:**

* Test updates that include bug fixes and new features. b. Check for security updates for the application.

9. **Performance and Load Testing:**

* To make sure the app can withstand scenarios of high user traffic, simulate heavy user loads.

## 1.2. Compare

Compare the University Transport System App's Black Box Testing method with a couple of other testing strategies, particularly White Box Testing and Grey Box Testing.

**1. Black Box Testing:**

* **Advantages:**
* **Independence from Internal Structure:** Black Box Testing is appropriate for testing without a thorough understanding of the internal code because it doesn't require it.
* **User-Centric:** Concentrates on evaluating user interactions and the functionality of the application.
* **Emulates Actual User Experience:** Replicating end users' interactions with the program.
* **Disadvantages:**

• **Limited Coverage of Internal Logic:** It may overlook problems with the code paths, internal logic, or structure.

**2. White Box Testing:**

* **Advantages:**

**• Comprehensive Code Coverage:** Verifies thorough testing by analyzing internal code paths, logic, and structures.

**• Early Code-Level Issue Detection:** able to recognize problems early on, such as logical errors and security vulnerabilities.

**• Sufficient for Unit Examination:** Ideal for testing single components, functions, or procedures.

* **Disadvantages:**

**• Dependent on Completeness and Quality of Code:** The efficacy is largely dependent on the completeness and quality of the code.

**• Time-consuming:** May take a lot of time and requires in-depth understanding of the internal code.

**3. Grey Box Testing:**

* **Advantages:**

**• Combines the Benefits of White Box and Black Box Testing:** Uses a combination of partial internal code knowledge and external functionality testing.

**• Determines Integration Problems:** can reveal problems with the way various components interact with one another.

* **Disadvantages:**

**• Partial Coverage:** For internal code paths, partial coverage might not be attained to the same extent as with White Box Testing.

**• Complexity:** necessitates striking a balance between upholding an external viewpoint and comprehending the internal logic.

## 1.3. Choice of Black Box Testing

* **Reasoning:**

• The University Transport System App is largely user-focused, and a seamless user experience is essential to its success.

• Black Box Testing fits in well with testing these external behaviors, and the SRS document places a strong emphasis on user functionalities.

• Black Box Testing is a good option for mobile applications since user interaction is the main concern.

* **Considerations:**

• Code reviews and security assessments are examples of additional testing that can address aspects that Black Box Testing may miss, such as internal logic.

# 2.Testing levels

This is the University Transport System App's testing-

**Testing Levels:**

1. **Unit Testing:**

• Perform independent unit tests for every module.

• Unit tests can be automated by using testing frameworks.

Unit tests should be carried out by developers while they are working on a project.

1. **Integration Testing:**

• Integrate the various modules and run integration tests.

• Evaluate module-to-module communication.

• Check that information moves between components correctly.

1. **System Testing:**

• Test the entire system from beginning to end.

• Run realistic scenarios to find any problems with the way the system works.

• Verify that all of the components are integrated.

1. **Acceptance Testing:**

• Incorporate stakeholders in the testing of user acceptance.

• Examine the application in light of the acceptance standards.

• Verify that the program meets the needs of stakeholders and end users.

1. **Regression Testing:**

Regression testing should be done following every update or modification.

• Verify that there are no adverse effects on currently available functionalities.

•Regression tests that are automated can help expedite the procedure.

# 3.Testing types, techniques and tactics

Here, testing approaches, methodologies, and strategies are incorporated into the University Transport System App testing strategy.

**1. Functional Testing:**

* **Testing Type:**
	+ **Unit Testing:**
		- **Technique:** Verify individual functions, methods, or modules independently.
		- **Tactic:** Use frameworks like JUnit for Java or XCTest for Swift to perform unit tests on specific code units.
	+ **Integration Testing:**
		- **Technique:** Validate the interaction between different modules or components.
		- **Tactic:** Use mock data to simulate interactions and ensure seamless integration.
	+ **System Testing:**
		- **Technique:** Assess the entire system's compliance with specified requirements.
		- **Tactic:** Perform end-to-end testing of user scenarios, including user registration, ticket purchase, and real-time tracking.

**2. Non-Functional Testing:**

* **Testing Type:**
	+ **Performance Testing:**
		- **Technique:** Evaluate the app's responsiveness and stability under varying loads.
		- **Tactic:** Use tools like JMeter or Gatling to simulate multiple users accessing the app simultaneously.
	+ **Security Testing:**
		- **Technique:** Identify vulnerabilities and ensure data protection.
		- **Tactic:** Conduct penetration testing to discover potential security risks and implement secure coding practices.
	+ **Usability Testing:**
		- **Technique:** Assess the app's user interface and overall user experience.
		- **Tactic:** Involve real users to perform usability tests, considering accessibility for users with disabilities.

**3. Compatibility Testing:**

* **Testing Type:**
	+ **Device Compatibility Testing:**
		- **Technique:** Ensure the app works seamlessly across various devices.
		- **Tactic:** Test the app on different smartphones and tablets running iOS 12 or later and Android 6 or higher.
	+ **Browser Compatibility Testing:**
		- **Technique:** Confirm compatibility with different web browsers if applicable.
		- **Tactic:** Test web-based components on popular browsers like Chrome, Firefox, and Safari.

**4. Regression Testing:**

* **Testing Type:**
	+ **Functional Regression Testing:**
		- **Technique:** Ensure new updates don't break existing functionalities.
		- **Tactic:** Implement automated regression tests using tools like Selenium for web-based features.
	+ **Performance Regression Testing:**
		- **Technique:** Verify that performance remains consistent after updates.
		- **Tactic:** Re-run performance tests after each update to identify any degradation.

**5. User Acceptance Testing (UAT):**

* **Testing Type:**
	+ **Alpha and Beta Testing:**
		- **Technique:** Test the app with a limited user group before the full release.
		- **Tactic:** Release the app to a small group of users (alpha) and later to a broader audience (beta) to gather feedback.
	+ **Exploratory Testing:**
		- **Technique:** Allow users to explore the app freely to identify any unexpected issues.
		- **Tactic:** Encourage users to interact with the app naturally and report any issues they encounter.

**6. Deployment Testing:**

* **Testing Type:**
	+ **Smoke Testing:**
		- **Technique:** Confirm basic functionalities work after deployment.
		- **Tactic:** Perform quick tests on critical features like user login and real-time tracking.
	+ **Rollback Testing:**
		- **Technique:** Ensure the ability to revert to the previous version if issues arise.
		- **Tactic:** Simulate a rollback scenario to confirm the process is smooth.

#  4.Testing process

Agile development approaches that place a high value on teamwork, customer feedback, and flexibility in response to changing needs. procedure for testing The University Transport System App's agile models.

**Agile Development:**

**Overview:**

• Agile is a software development methodology that is incremental and iterative.

• Places a focus on adaptability, teamwork, and client satisfaction.

• Gives priority to promptly delivering usable software and adjusting to evolving requirements.

**Considerations:**

• Iterative Development: Regular feedback and ongoing enhancement are made possible by frequent iterations.

• Adaptability: Throughout the development process, Agile takes into account shifting requirements and priorities.

• Cross-functional Teams: Promotes cooperation amongst various roles for effective growth.

**Testing in Agile:**

• Continuous Testing: Throughout the entire development lifecycle, testing is integrated.

• Automated testing: This guarantees more accurate and timely feedback.

• Regular Review and Adaptation: Regular evaluations and retrospectives improve the testing procedure.

# 5.Measurement in software testing

The University Transport System App is examined in the context of software testing, taking into account the test plan, test cases, and testing difficulty hierarchy.

## 5.1. Hierarchy of Testing Difficulty

1. **Unit Testing:**
	* **Objective:** Ensure that individual components and functions work as expected.
	* **Example Test Cases:**

• Check that user login and registration work as intended.

•Test the ability to retrieve real-time bus arrival and departure information.

1. **Integration Testing:**
	* **Objective:** Confirm that different modules of the app work seamlessly together.
	* **Example Test Cases:**

• Verify how well the bus tracking user interface integrates with the back-end data retrieval system.

• Confirm that route planning and user authentication are compatible.

1. **System Testing:**
	* **Objective:** Validate the entire system as a whole, considering all integrated components.
	* **Example Test Cases:**

• Test complete scenarios, like when a user registers, makes travel plans, buys a ticket, and gets alerts..

1. **Acceptance Testing:**
	* **Objective:** Ensure the app meets specified requirements and is ready for deployment.
	* **Example Test Cases:**

• Verify whether the app offers precise real-time bus information.

• Check that the functions of each user role—User, Driver, Helper, and Admin—function as intended.

## 5.2. Test Plan

1. **Introduction:**
	* Brief overview of the University Transport System App and its purpose.
2. **Test Scope:**
	* Identify the areas and features to be tested.
		+ e.g., User registration, bus tracking, ticket purchase, etc.
3. **Test Objectives:**
	* Define what the testing aims to accomplish.
		+ e.g., Ensure real-time bus information is accurate and reliable.
4. **Test Environment:**
	* Specify the devices, operating systems, and network conditions for testing.
		+ e.g., iOS 12 and later, Android 6 or higher, with GPS and internet access.
5. **Test Cases:**
	* Detail individual test cases for each feature and functionality.
		+ e.g., Test case for user registration, test case for bus tracking accuracy.
6. **Testing Schedule:**
	* Outline the timeline for different testing phases.
		+ e.g., Unit testing completed by [Date], Integration testing by [Date].
7. **Testing Resources:**
	* List the personnel involved in testing and their roles.
		+ e.g., QA engineer responsible for system testing.

## 5.3. Test Cases

1. **User Registration:**
	* **Objective:** Verify that users can successfully register and log in.
	* **Test Cases:**

• Confirm that a user is able to register using legitimate credentials.

• Verify that a user who has registered can log in properly.

1. **Real-time Bus Information:**
	* **Objective:** Confirm the accuracy of real-time bus arrival and departure information.
	* **Test Cases:**

• Verify that the bus timings shown correspond to the published schedules.

• Confirm that real-time updates to bus information are made by the app.

1. **Bus Tracking:**
	* **Objective:** Ensure the map-based bus tracking feature functions correctly.
	* **Test Cases:**

• Check that the buses are shown in the correct locations on the map.

• Check the map's responsiveness while tracking in real time.

1. **Ticket Purchase:**
	* **Objective:** Confirm that users can buy tickets seamlessly.
	* **Test Cases:**

• Verify the security and dependability of the payment procedure.

• Verify that the user's booking history shows the tickets they have purchased.

1. **User Feedback (Comments/Issues):**
	* **Objective:** Validate the functionality for users to express concerns.
	* **Test Cases:**

• Verify that users can accurately submit feedback.

• Verify whether the administrator can see and respond to user comments.

Technological Requirement

# 1. Softwires

Here, the necessary softwires are:

## 1.1. Back-end Development

1. **Server-Side Language: Node.js (JavaScript)**
	* The server-side JavaScript runtime is called Node.js. It is renowned for having an event-driven architecture, which makes it effective at managing several connections at once. It's a popular option in web development and works well for real-time applications.
2. **Database: MongoDB**
	* MongoDB is a NoSQL database that holds information in BSON documents that resemble JSON. Due to its lack of schema, it offers flexibility in managing different kinds of data. In situations where the data structure may change over time, this can be helpful.
3. **APIs: RESTful or GraphQL**
	* RESTful APIs are straightforward and scalable because they are stateless and communicate via standard HTTP methods. A more adaptable and effective substitute is offered by GraphQL, which enables users to request just the data they require.
4. **Authentication: JSON Web Tokens (JWT)**
	* JWT is a small, secure URL-based way to represent transferable claims between two parties. JWTs are frequently used in web development for information sharing and authentication between the client and the server.
5. **Web Server: Nginx or Apache**
	* Two well-known web servers are Nginx and Apache. While Apache is highly configurable and has long been a popular choice for web hosting, Nginx is renowned for its high performance and effective resource utilization.
6. **Hosting: AWS or Heroku**
	* Scalable and dependable hosting is possible with Amazon Web Services (AWS), a cloud computing platform that offers a variety of services. Heroku is a cloud platform that frees developers from the burden of managing infrastructure so they can concentrate more on the application.
7. **Security:**
	* Use secure coding techniques to guard against common online vulnerabilities like cross-site request forgery (CSRF), SQL injection, and cross-site scripting (XSS). Use parameterized queries, make sure the data is validated properly, and clean up user input.
8. **Model-View-Controller (MVC):**
	* Although Node.js doesn't mandate an exact MVC architecture, you can still organize your code using MVC best practices. For instance, you can use standalone files for view rendering, Mongoose for data modeling (Model), and Express.js for routing (Controller).

## 1.2.Front-end Development for Website

1. **HTML/CSS: Standard HTML5/CSS3**
	* The fundamental technologies for organizing and styling web content are Standard HTML5 and CSS3. New elements and APIs are introduced by HTML5, and improved styling capabilities are offered by CSS3.
2. **JavaScript Framework: React.js**
	* React.js is a JavaScript package for UI development. It is effective for developing dynamic and interactive web applications because it enables developers to create reusable user interface components.
3. **CSS Framework: Bootstrap for responsive design**
	* Bootstrap is a well-known CSS framework that offers pre-styled components and a responsive grid system. It makes the process of creating and guaranteeing a unified appearance and feel across various devices simpler.
4. **Web Server: Nginx or Apache**
	* Nginx or Apache can be used to serve the frontend assets, just like in the backend.
5. **Security:**
	* Additionally, use secure coding techniques on the client end. By validating and sanitizing user inputs, you can prevent client-side vulnerabilities such as XSS. To encrypt data in transit, use HTTPS. Put Content Security Policy (CSP) headers into practice to reduce the risk of code injection.
6. **Model-View-Controller (MVC):**
	* An architecture based on components is a natural fit for React.js. You can arrange your React components in a way that follows MVC guidelines even if it's not strictly MVC. State management can be accomplished by using libraries such as Redux, which separate the data (Model) from the user interface (View).

## 1.3.Front-end Development for Mobile App

1. **Mobile Framework: React Native**
	* React Native lets you use JavaScript and React to create mobile applications. It provides an economical and expedient solution for cross-platform development by allowing you to write code only once and have it run on both the iOS and Android platforms.
2. **JavaScript: JavaScript (React Native uses JavaScript)**
	* JavaScript is the programming language used by React Native. Developers can create mobile applications that look native by utilizing their current JavaScript knowledge.
3. **UI Design: Use React Native components for cross-platform consistency**
	* Pre-built components that closely resemble native user interface elements are included with React Native. This keeps native apps' performance intact while enabling a unified look and feel across platforms.
4. **Security:**
	* There are similar security concerns with the React Native mobile application. Use secure communication protocols, validate and sanitize inputs, secure sensitive data storage on the device, and implement secure coding practices.
5. **Model-View-Controller (MVC):**
	* Apps built with React Native also use a component-based architecture. You can arrange your components using concepts akin to those found in React.js. You can use state management libraries to manage the state of your application, such as Redux or the integrated React Context API.

## 1.4.Additional Security Considerations

1. **Authentication and Authorization:**
	* Use JWT to implement a secure authentication system for the mobile app and website. Make sure the server-side authorization checks are done correctly to manage access to various features and data.
2. **Data Validation:**
	* Use extensive data validation on the client and server sides to guard against malicious inputs and improve your application's overall security.
3. **Logging and Monitoring:**
	* Configure monitoring and logging systems to keep tabs on and examine possible security events. This may entail keeping track of unsuccessful login attempts, keeping an eye out for odd user behavior, and being on the lookout for security risks
4. **Regular Security Audits:**
	* Regularly audit the security of your infrastructure and codebase to find and fix any vulnerabilities. Keep up with the most recent security updates and best practices for the devices you use.

# 2. Hardware

It's crucial to remember that specific hardware specifications, like memory capacity and bandwidth, are dependent on a number of variables, including the anticipated user load, application complexity, and data processing volume.

## 2.1.Back-end (Server)

1. **Memory Size:**
	* The server's memory (RAM) capacity is essential for processing multiple requests at once and effectively managing data in memory. The precise amount will vary depending on things like how big your dataset is and how complicated the calculations are. A server with at least 8GB to 16GB of RAM might be appropriate for moderately sized applications, though this can vary.
2. **Bandwidth:**
	* The speed at which data can be transferred to and from your server depends on its bandwidth. This is essential for managing assets, responding to queries, and communicating with databases. The amount of data being transferred and the anticipated volume of traffic determine how much bandwidth is needed. It's crucial to have a server with enough bandwidth to accommodate the anticipated volume of users.

## 2.2.Database

1. **Memory Size:**
	* To improve query performance and cache frequently accessed data, the memory (RAM) of the database server is essential. The amount of memory should be enough to store data that is accessed often. An appropriate place to start for databases such as MongoDB or MySQL could be a server with 16GB to 32GB of RAM.
2. **Bandwidth:**
	* Data transfers and query processing between the application server and the database depend on the database server's bandwidth. Make sure the bandwidth is adequate for the anticipated amount of data transfers and database queries.

## 2.3.Front-end (Web Server)

1. **Memory Size:**
	* The web server's memory capacity affects how well it can serve static assets and manage concurrent user requests. A web server with 4GB to 8GB of RAM may be adequate for a moderately-sized React.js application hosted on it.
2. **Bandwidth:**
	* The bandwidth of the web server is essential for responding to incoming requests, handling asset serving, and facilitating backend communication. Make sure the web server has sufficient bandwidth to manage the anticipated amount of data coming in and going out.

## 2.4.Mobile App (Backend and Frontend)

1. **Memory Size:**
	* The anticipated workload should be used to determine the memory size of the server hosting the mobile app backend. Memory needs for the mobile app's frontend will depend on how complex the user interface is and how big the assets that are loading are.
2. **Bandwidth:**
	* The bandwidth of the backend server is essential for processing API requests made by the mobile application. The mobile app's frontend also needs enough bandwidth in order to interact with the backend and load assets. Make sure there is enough bandwidth available for the frontend and backend.

## 2.5.Additional Considerations

1. **Scaling:**
	* When planning for scalability, take into account the possibility of scaling vertically—improving server specifications—or horizontally—adding more servers in response to rising demand.
2. **Load Balancing:**
	* Use load balancing to split up incoming traffic among several servers, which will increase dependability and performance.
3. **Monitoring and Optimization:**
	* Continually track hardware performance and adjust configurations in response to usage trends and expansion.

# Scheduling and Estimates

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone** | **Description** | **Release Date** | **Iteration** |
| M1 | Application view and Design (Front-end development) | October 5, 2023 | R1 |
| M2 | Database for my application (Back-end) | October 17, 2023 | R1 |
| M3 | Integrating views and designs (Integrating front-end and back-end) | November 12, 2023 | R1 |
| M4 | Testing for initial release | November 20, 2023 | R2 |
| M5 | Issue tracker, user reviews, web design integration | December 1, 2023 | R2 |
| M6 | Final release | December 23, 2023 | R2 |