

**Department of Computer Science and Engineering**

**SRS REPORT**

**Assignment No: 4**

**Assignment on: Software Requirements Specification (SRS)**

**Live Server Link:** <https://mahin5580.github.io/SoftTech.github.io/>

**GitHub Project Link:** [**https://github.com/Mahin5580/SoftTech.github.io**](https://github.com/Mahin5580/SoftTech.github.io)

**Submitted By:**

**Mmk Mahin**

**ID:221-15-5580**

**Section:61-U**

CSE236 | SOFTWARE PROJECT - 2

**Submitted to:** Md**.** Golam Rabbany

**Dept. of CSE, Daffodil International University**

**wertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

Niloy Sarker

Managing Director

Go Travel

Dhaka 1204

Subject: Application for agreement of Travel Management System Project.

Dear Mr sarker

I hope this letter finds you in good health and high spirits. I am writing to propose the development of a comprehensive Travel Management System that I believe will significantly enhance the efficiency and quality of your Travel planning services. I am the member of SoftTech company. I have identified the whole system of your Travel company and decided to create a unique Travel management system.

According to your business requirement we design GO TRAVEL Website and Web Application System for you, in this agreement paper we right each and every part and all possible feature which covered our application. So please read carefully and if it satisfies you then sign and confirm the agreement.

Sincerely

MMK Mahin

Senior developer

Application development Brach

SoftTech

**Table of Contents**

[Software Requirements Specification (SRS) 6](#_Toc152166195)

[1.Introduction 6](#_Toc152166196)

[1.1 Purpose 6](#_Toc152166197)

[1.2 Scope 6](#_Toc152166198)

[1.3 Definitions, Acronyms, and Abbreviations 6](#_Toc152166199)

[2. Project Goals 6](#_Toc152166200)

[2.1 Primary Goals 6](#_Toc152166201)

[2.2 Secondary Goals 6](#_Toc152166202)

[3.Use case design: 7](#_Toc152166203)

[4. Features 7](#_Toc152166204)

[4.1 User Registration and Authentication 7](#_Toc152166205)

[4.2 Trip Planning 7](#_Toc152166206)

[4.3 Destination Information 7](#_Toc152166207)

[4.4 Booking and Reservations 7](#_Toc152166208)

[4.5 User Profile Management 8](#_Toc152166209)

[4.6 Reviews and Recommendations 8](#_Toc152166210)

[4.7 Notifications 8](#_Toc152166211)

[4.8 Search and Filters 8](#_Toc152166212)

[5. UML diagram 9](#_Toc152166213)

[6.Functionality 9](#_Toc152166218)

[6.1 User Roles and Permissions 9](#_Toc152166219)

[6.2 User Interface 9](#_Toc152166220)

[6.3 Database Management 10](#_Toc152166221)

[6.4 Integration with External Services 10](#_Toc152166222)

[6.5 Security Measures 10](#_Toc152166223)

[6.6 Performance and Scalability 10](#_Toc152166224)

[7.Flow chart Diagram: 10](#_Toc152166225)

[8. Milestone & Reporting 11](#_Toc152166226)

[9. Software Development Life Cycle (SDLC) 11](#_Toc152166227)

[9.1 Purpose 11](#_Toc152166228)

[9.2 Phases 11](#_Toc152166229)

[9.3 Activities 12](#_Toc152166230)

[9.4 Documentation 12](#_Toc152166231)

[9.5 Roles and Responsibilities 12](#_Toc152166232)

[9.6 Iterations and Feedback 12](#_Toc152166233)

[9.7 Testing(SDLC) 12](#_Toc152166234)

[9.8 Change Management 12](#_Toc152166235)

[9.9 Risk Management 12](#_Toc152166236)

[9.10 Quality Assurance 12](#_Toc152166237)

[9.11 Deliverables 12](#_Toc152166238)

[9.12 Team Roles 12](#_Toc152166239)

[10.Types of Software Development Methodologies: 12](#_Toc152166240)

[10.1 Waterfall Model: 13](#_Toc152166241)

[10.2 Agile Methodology (Scrum and Kanban): 13](#_Toc152166242)

[10.3 Iterative Model: 13](#_Toc152166243)

[10.4 Spiral Model: 13](#_Toc152166244)

[10.5 V-Model (Validation and Verification Model): 13](#_Toc152166245)

[10.6 Rapid Application Development (RAD): 13](#_Toc152166246)

[10.7 Incremental Model: 13](#_Toc152166247)

[10.8 DevOps: 13](#_Toc152166248)

[11.The methodology I would prefere to develop my application and why? 14](#_Toc152166249)

[12. Testing 15](#_Toc152166250)

[12.1 Testing approch: 15](#_Toc152166251)

[12.2 Testing levels: 16](#_Toc152166252)

[12.3 Testing types, techniques and tactics: 17](#_Toc152166253)

[12.4 Testing process: 18](#_Toc152166254)

[12.5 Test Plan: 19](#_Toc152166255)

[12.6 Test Case: 20](#_Toc152166256)

[12.7 Risks and Assumptions: 21](#_Toc152166257)

[13. Payment Terms & condition 21](#_Toc152166258)

[14. Responsibility 21](#_Toc152166259)

[15. Contact Us 21](#_Toc152166260)

[16. Agreement Signed By: 21](#_Toc152166261)

# Software Requirements Specification (SRS)

## 1.Introduction

This document includes specifications and features of the software in detail. It helps understand the target audience and user classes accordingly and internal and external interface requirements.

### 1.1 Purpose

The Software Requirements Specification (SRS) outlines the goals, features, functionality, and constraints of the "Go Travel" project. This document serves as a guide for the development team, stakeholders, and project managers to ensure a clear understanding of the project's requirements.

### 1.2 Scope

"Go Travel" is a web-based travel planning and booking platform that aims to simplify the travel experience for users by providing access to destination information, trip planning tools, and the ability to book various travel services.

### 1.3 Definitions, Acronyms, and Abbreviations

SRS: Software Requirements Specification

UI: User Interface

API: Application Programming Interface

## 2. Project Goals

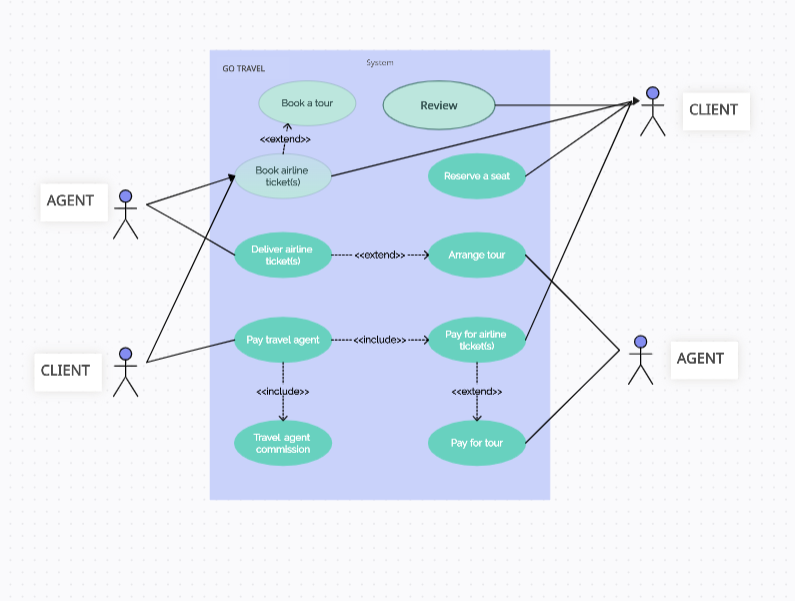
### 2.1 Primary Goals

* To provide users with a user-friendly platform for planning and booking trips.
* To offer comprehensive destination information and travel recommendations.
* To facilitate secure and convenient booking of flights, accommodations, and activities.
* To build a community where users can share travel experiences and reviews.

### 2.2 Secondary Goals

* To implement a responsive and aesthetically pleasing user interface.
* To support multiple languages and currencies.
* To integrate with external services for real-time pricing and availability.

## 3.Use case design:



## 4. Features

### 4.1 User Registration and Authentication

* User registration with email or social media accounts.
* User login and authentication mechanisms.
* Password reset functionality.

### 4.2 Trip Planning

* Create, edit, and save trip itineraries.
* Add and remove destinations and activities to itineraries.
* Calendar view for trip planning.

### 4.3 Destination Information

* Detailed information about destinations, including attractions, weather, and local tips.
* User-generated reviews and ratings for destinations.

### 4.4 Booking and Reservations

* Flight booking and reservation system.
* Hotel and accommodation booking.
* Activity and tour reservations.
* Payment processing and invoice generation.

### 4.5 User Profile Management

* User profile creation and editing.
* Profile picture upload.
* History of past trips and bookings.

### 4.6 Reviews and Recommendations

* User-generated reviews and ratings for accommodations, activities, and destinations.
* Personalized travel recommendations based on user preferences and history.

### 4.7 Notifications

Email and in-app notifications for booking confirmations, updates, and reminders.

### 4.8 Search and Filters

* Advanced search functionality for destinations, accommodations, and activities.
* Filters for price range, location, and user ratings.

5. UML diagram**:**

GO TRAVEL

**User**

**Reservation**

+user\_id:int

+user\_name: string

+user\_email: string

+searchflights ()

+bookflight ()

+Searchrentalcars ()

+bookcar ()

+travelhistory ()

+booking ()

+modification ()

+cancellation ()

+confirmation ()

**Management System**

+admindashboard ()

+usermanagement ()

+contentmanagement ()

+bookingmanagement ()

## 

+payment\_id:int

+payment\_amount: int

+payment\_date: date

+addpayment ()

+editpayment ()

+deletepayment ()

**User**

**Hotel**

## 

+hotel\_id:int

+hotel\_name: string

+hotel\_rent: string

+hotel\_type: string

+reservation ()

## 6.Functionality

### 6.1 User Roles and Permissions

User roles: guest, registered user, admin.

Permissions based on roles.

### 6.2 User Interface

* Intuitive and responsive design for web and mobile platforms.
* Consistent navigation and layout.
* Interactive trip planning interface.

### 6.3 Database Management

* Database to store user data, trip itineraries, bookings, and reviews.
* Regular backups and data security measures.

### 6.4 Integration with External Services

* Integration with payment gateways for secure transactions.
* Integration with third-party APIs for real-time data (flights, accommodations, activities).

### 6.5 Security Measures

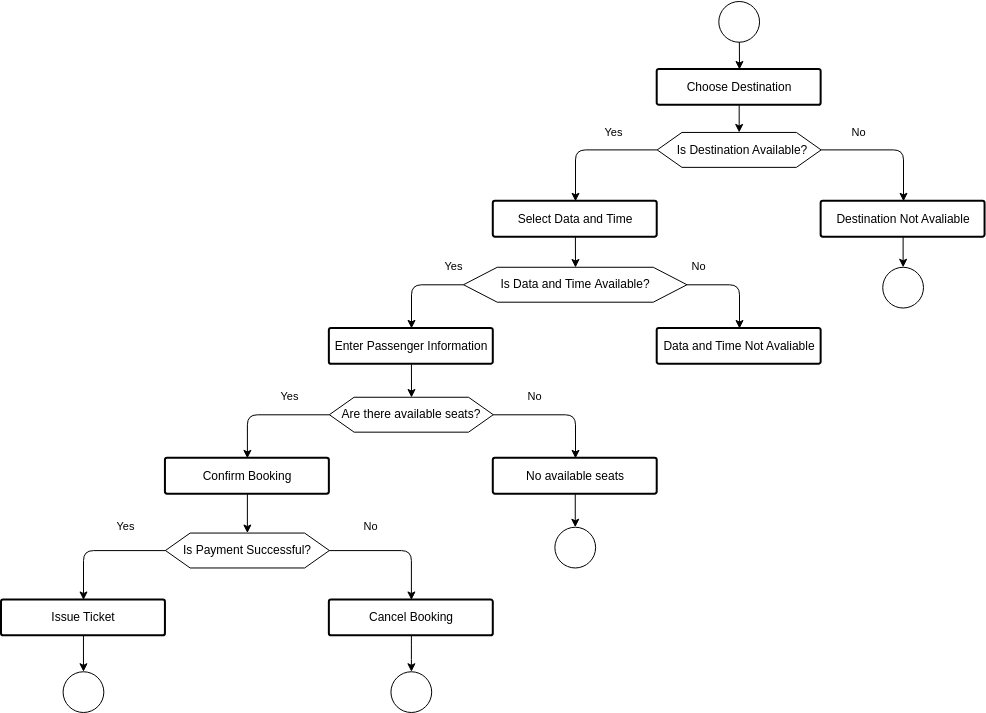
* Secure data transmission using HTTPS.
* Encryption of sensitive user data.
* User authentication and authorization.

### 6.6 Performance and Scalability

* Efficient coding practices for optimal performance.
* Scalability to accommodate a growing user base.

## 

## 7.Flow chart Diagram:



## 8. Milestone & Reporting

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone** | **Task** | **Reporting** | **Required Time** |
| Analyzing Project | Submit Idea & Design | Submit The Design | 3 Days |
| Requirements Collection | Gathering Data & submit |  | 5 Days |
| Development | Work with the project from the root | Review The Work | 21 Days |
| Testing | Testing the entire application system |  | 10 Days |
| Deployment | Fining and Review Final project | Review Final Work | 7 Days |
| Delivery Project | Available to Online platform | Live On Server | 5 Days |

## 

## 9. Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) methodologies are systematic processes and frameworks used by software development teams to plan, design, build, test, deploy, and maintain software applications. These methodologies provide a structured approach to managing the entire software development process. Each SDLC methodology follows a set of phases, activities, and best practices to ensure that software projects are completed efficiently and with high quality.

### 9.1 **Purpose**

The primary purpose of a software development methodology is to provide a set of best practices and guidelines to help development teams plan, execute, and complete software projects successfully. It aims to ensure that software is developed efficiently, meets quality standards, and satisfies customer requirements.

### 9.2 **Phases**

SDLC methodologies are typically divided into phases, which are the stages of development. Common phases include requirements analysis, design, coding, testing, deployment, and maintenance. The order and number of phases can vary between methodologies.

### **9.3 Activities**

Each phase involves a set of specific activities that must be performed. **For example**, the design phase may include activities such as architectural design, database design, and user interface design.

### 9.4 **Documentation**

SDLC methodologies emphasize the importance of documentation. Detailed documentation is created at each phase to capture requirements, design decisions, coding guidelines, and testing plans.

### **9.5 Roles and Responsibilities**

SDLC methodologies define roles and responsibilities for team members, including developers, testers, project managers, and stakeholders. This ensures that everyone understands their roles in the development process.

### 9.6 **Iterations and Feedback**

Some SDLC methodologies, such as Agile, emphasize iterative development with frequent feedback and collaboration between the development team and stakeholders.

### 9.7 **Testing (SDLC)**

Testing is a critical component of SDLC methodologies, ensuring that the software is reliable and free of defects. Different methodologies may have varying approaches to testing.

### 9.8 **Change Management**

SDLC methodologies often include change management processes to handle modifications to project requirements or scope.

### 9.9 **Risk Management**

Many methodologies incorporate risk assessment and mitigation strategies to identify and address potential project risks.

### **9.10 Quality Assurance**

Ensuring the quality of the software is a central concern in SDLC methodologies. Various methodologies may have different quality assurance processes.

### **9.11 Deliverables**

Each phase of an SDLC produces specific deliverables, such as design documents, source code, test plans, and user manuals. These deliverables help track progress and ensure that project goals are met.

### **9.12 Team Roles**

Many methodologies define specific roles within a development team, such as product owners, scrum masters, and developers. These roles have specific responsibilities and contribute to the successful execution of the methodology.

## 10.Types of Software Development Methodologies:

There are several types of software development methodologies, each offering a distinct approach to managing and executing software projects. Here are some of the most commonly used software development methodologies:

### 10.1 Waterfall Model:

The Waterfall model is a linear and sequential approach to software development. It consists of distinct phases, such as requirements, design, implementation, testing, deployment, and maintenance. Each phase must be completed before moving on to the next. It's suitable for projects with well-defined and stable requirements.

### 10.2 Agile Methodology (Scrum and Kanban):

Agile methodologies are iterative and flexible, promoting collaboration and adaptability. They emphasize breaking the project into small increments and delivering working software in short cycles. Continuous feedback and user involvement are key principles. Scrum and Kanban are specific Agile frameworks with regular, time-boxed iterations (sprints).

### 10.3 Iterative Model:

The Iterative model involves developing a partial system or version and then refining it through repeated iterations. Each iteration can add new features or improve existing ones. It's suitable for projects where requirements may evolve or where prototyping is valuable.

### 10.4 Spiral Model:

The Spiral model combines elements of the Waterfall and Iterative models. It emphasizes risk analysis, with each cycle consisting of planning, risk analysis, engineering, and evaluation. It's beneficial for projects with high uncertainty and significant risks.

### 10.5 V-Model (Validation and Verification Model):

The V-Model is an extension of the Waterfall model that focuses on validation and verification. It emphasizes testing at each stage of development, corresponding to a verification phase and a validation phase. It's suitable for projects with strong testing requirements.

### 10.6 Rapid Application Development (RAD):

RAD emphasizes rapid prototyping and quick development. It involves building prototypes and iteratively refining them based on user feedback. It's ideal for projects where speed and user involvement are critical.

### 10.7 Incremental Model:

The Incremental model divides the software into smaller, manageable parts, and each part is developed separately. New increments build upon the previous ones. It's suitable for projects that can be divided into distinct and independently deliverable components.

### 10.8 DevOps:

DevOps isn't a traditional SDLC model but rather a cultural and technical approach that emphasizes collaboration between development and operations teams. It focuses on automation, integration, and continuous delivery (CI/CD). DevOps is beneficial for projects requiring rapid development, testing, and deployment.

## 11.The methodology I would prefere to develop my application and why?

In my case, developing a phonebook application, it's likely thatanAgile methodologylike Scrum or Kanban would be a good choice. These methodologies provide the flexibility to quick changes, gather user feedback, and incrementally build and refine theapplication. Also**,** the specific choice will depend on project size, and other contextual factors**.**

**Agile Methodology (Scrum):**

**Flexibility for Evolving Requirements:**

The project's primary and secondary goals include features that may evolve over time, such as user registration, trip planning, destination information, and booking. Agile allows for flexibility in accommodating changes and additions to requirements.

**Regular Collaboration with Stakeholders:**

Agile methodologies, including Scrum, emphasize close collaboration with stakeholders. This aligns well with your goal of building a community where users can share travel experiences and reviews. Frequent interactions and feedback are essential for this feature.

**Iterative Development**:

Agile promotes iterative development with short development cycles. Your project's milestone and reporting section includes tasks for development, testing, and reviewing work, which can fit well within Scrum's iterative approach.

**Continuous Improvement:**

Agile methodologies encourage continuous improvement and adaptation. For features like user-generated reviews and recommendations, this iterative approach can be valuable to refine algorithms and personalization.

**Quick Deployment:**

The project includes deployment as a milestone, indicating a need for timely releases. Agile, particularly Scrum, focuses on delivering potentially shippable increments at the end of each sprint.

**Transparency:**

Agile provides transparency into the project's progress through frequent meetings and status updates, ensuring that all stakeholders are informed and aligned.

## 

**Dynamic Scaling:**

Agile can handle dynamic scaling as your user base grows. The user roles and permissions, user profiles, and the community aspect of your application can benefit from Agile's adaptability.

**User-Centric Design:**

The project goals include providing a user-friendly platform and responsive user interface. Agile emphasizes user-centric design and incorporates user feedback into the development process.

## 12. Testing

### 12.1 Testing approch:

For Go travel, the testing typically involves a combination of frontend and backend functionalities along with a strong emphasis on user experience, a well-rounded testing strategy is crucial. So, there should use a combination of test approch:

The testing approch we use are:

* Black Box Testing
* Visual Testing
* Exploratory Testing
* Dynamic Testing

**Black Box Testing:** Black box testing focuses on testing the functionality of a software application without looking at its internal code or logic. Testers are concerned with the inputs and outputs of the software.

This is essential for testing the overall functionality of the website from an end-user perspective. It helps ensure that users can interact with the website as intended, including features like searching for flights, booking accommodations, and navigating through the site.

**Visual Testing:** Visual testing involves assessing the visual aspects of an application, such as user interface (UI) elements, layouts, and graphical elements, to ensure they meet design and usability standards.

Visual testing is important for the website Go Travel where the visual appeal and user interface play a significant role. Test across various devices and browsers to ensure a consistent and visually appealing experience for users.

**Exploratory Testing:** Given the dynamic nature of the website Go Travel, exploratory testing is beneficial for uncovering unexpected issues, assessing user journeys, and evaluating the overall user experience. Testers can simulate real-world scenarios and identify usability issues that may not be captured in scripted test cases.

**Dynamic Testing:** Dynamic testing, including functional, performance, and security testing, is crucial to ensure that the website operates correctly, handles traffic effectively, and is secure. This involves testing features like search functionality, booking processes, payment transactions, and assessing the website's response under varying.

And we don’t use these testing approch:

* White Box Testing
* Passive Testing
* Static Testing

Here's why some testing approaches might not be emphasized for Go Travel website:

**White Box Testing:** White box testing involves examining the internal logic and code of the application. For many web applications, including travel websites, the emphasis is often on user interactions, UI/UX, and overall functionality. White box testing may be less prioritized unless there are critical algorithms or backend processes that require in-depth examination.

**Passive Testing:** Passive testing involves observing the system without actively interacting with it. While it can be useful for monitoring and gathering data, it might not be a primary focus for functional testing of a travel website. The emphasis is typically on actively testing user interactions, functionalities, and performance.

**Static Testing:** Static testing involves reviewing documents, code, and other artifacts without executing the software. While it's beneficial in early stages to catch issues in requirements and design, it may not be as emphasized during later stages of testing when the focus shifts to dynamic testing to ensure that the application functions correctly.

### 12.2 Testing levels:

The testing levels for a website, including Go Travel, typically include various stages of testing to ensure comprehensive coverage. Here are common testing levels for a website.

**Unit Testing:** Unit testing focuses on testing individual components or units of the software in isolation. It verifies that each unit functions as intended. Unit testing is essential for backend components, algorithms, and functionalities that form the foundation of the travel website.

**Integration Testing:** Integration testing verifies the interaction and communication between different components or units to ensure they work together seamlessly.Essential for ensuring that different modules, services, or systems within the website integrate properly, such as the integration of payment gateways, booking systems, and user databases.

**System Testing:** System testing involves testing the entire system as a whole, ensuring that all components work together to meet specified requirements. Vital for assessing the overall functionality of the travel website, including end-to-end testing of user journeys, features, and interactions.

**Acceptance Testing:** Acceptance testing commonly includes the following four types:

* User acceptance testing (UAT)
* Operational acceptance testing (OAT)
* Contractual and regulatory acceptance testing
* Alpha and beta testing

Acceptance testing evaluates whether the system meets the acceptance criteria and requirements set by stakeholders**.** Ensures that the travel website satisfies user expectations and business requirements. This can include user acceptance testing (UAT) performed by end-users or stakeholders.

### 12.3 Testing types, techniques and tactics:

For a travel website, we can employ a variety of testing types, techniques, and tactics to ensure thorough coverage of different aspects of the application**.** Here are some testing types:

#### 12.3.1 Testing types:

**Functional Testing:** Validates that the features and functionalities of the travel website work as expected. Use test cases to verify functionalities such as flight search, hotel booking, user registration, and payment processing.

**Non-Functional Testing:** Focuses on non-functional aspects like performance, security, and usability. Conduct performance testing, security testing, and usability testing to ensure the website's speed, security, and user experience meet expectations.

**Compatibility Testing:** Ensures the website functions correctly across different browsers, devices, and operating systems. Test the website on various browsers (Chrome, Firefox, Safari, etc.) and devices (desktop, mobile, tablet) to identify and resolve compatibility issues.

**Usability Testing:** Assesses how user-friendly the website is and identifies areas for improvement in design and user experience. Conduct user surveys, usability testing sessions, and heuristic evaluations to gather feedback on the website's usability.

**Performance Testing:** Evaluates the website's speed, responsiveness, and scalability under different conditions. Use tools to simulate various user loads, measure response times, and identify bottlenecks in the system.

**Security Testing:** Identifies vulnerabilities and weaknesses in the website's security measures. Perform penetration testing, vulnerability assessments, and security scans to ensure the website is resilient against potential threats.

#### 12.3.2 Testing techniques:

**Automated Testing:** Involves using testing tools and scripts to automate repetitive and time-consuming test scenarios. Implement test automation for regression testing, performance testing, and repetitive functional tests to enhance efficiency.

**Exploratory Testing:** Testers explore the application without predefined test cases to discover defects and issues. Encourage testers to explore the Go travel website organically, mimicking real user interactions, to uncover unexpected issues.

**Boundary Value Analysis:** Tests values at the boundaries of allowed input ranges to ensure robustness. Validate input fields with boundary values, such as minimum and maximum dates for booking, to identify potential issues.

#### 12.3.3 Testing Tactics:

**Risk-Based Testing:** Prioritizes testing efforts based on the perceived risk of features or functionalities. Identify critical areas of the Go travel website, such as payment processing, and allocate more testing resources to ensure thorough coverage.

**Continuous Testing:** Integrates testing throughout the development lifecycle to catch defects early. Implement continuous integration and continuous testing practices to provide rapid feedback on changes and ensure a stable website.

**Session-Based Testing:** Organizes testing efforts into focused sessions with specific goals. Schedule testing sessions dedicated to specific aspects of the Go travel website, such as user registration or booking workflows, to thoroughly test each area.

### 12.4 Testing process:

Implementing an agile testing process for the Go travel website involves incorporating testing seamlessly into the development lifecycle, ensuring rapid and iterative testing cycles. We choose agile testing process, for the website Go Travel.

The simulation includes key elements of an Agile testing process, aligning testing activities with the various stages of Agile development. It incorporates concepts such as continuous integration, daily stand-ups, sprint planning, regression testing, and ongoing monitoring, which are commonly associated with Agile methodologies. Here is the simulation for the testing:

* **Sprint Planning:** Identify the testing levels relevant for each sprint, such as unit testing for new features and regression testing for existing functionalities. Determine the types of testing needed, such as functional, non-functional, and compatibility testing. Use automated testing techniques for regression testing and incorporate risk-based testing to prioritize critical features.
* **Continuous Integration**: Integrate unit testing into the continuous integration process to catch code issues early. Leverage automated unit tests to ensure that individual components function as expected. Implement continuous testing to provide rapid feedback on code changes.
* **Sprint Execution**: Conduct functional testing, integration testing, and usability testing during the sprint. Use automated functional tests to verify new features. Perform exploratory testing to uncover issues that may not be covered by scripted test cases. Implement usability testing to assess the user experience.
* **Daily Stand-ups**: Address testing progress, challenges, and coordination in daily stand-up meetings. Discuss testing goals for the day, potential roadblocks, and collaborate with developers to address issues promptly**.**
* **Regression Testing:** Conduct regression testing as part of each sprint to ensure that new features do not impact existing functionalities. Use automated regression tests to validate critical workflows. Implement session-based testing to focus on specific areas of the application during regression cycles.
* **Sprint Review and Retrospective:** Gather feedback on the implemented features and identify areas for improvement. Incorporate feedback from end-users and stakeholders during the review. Use retrospective meetings to discuss testing process improvements and address any challenges encountered.
* **Continuous Deployment:** Integrate testing into the continuous deployment process to ensure that production releases are reliable. Implement automated deployment testing to verify the functionality of the travel website in the production environment. Utilize canary releases and feature flags to mitigate risks associated with new releases.
* **Ongoing Monitoring:** Implement ongoing monitoring to identify and address issues in the live environment. Use tools for performance monitoring, security monitoring, and error tracking to detect and address issues promptly.

By aligning testing activities with the agile development process, teams can achieve a balance between rapid development and quality assurance. Regular collaboration between developers and testers, automated testing practices, and continuous improvement efforts are key elements of a successful agile testing process for the Go travel website.

### 12.5 Test Plan:

Below is a simplified the test plan focusing on testing the flight booking functionality of Go travel website.

Test Plan for Flight Booking Functionality.

**Introduction:**

Project Name: Go Travel

Testing Team: QA Avengers

Version: 1.0

Date: [11/28/2023]

**Resources:**

Testing Team Members: QA Avengers

Development Team Contact

Testing Environment

Testing Tools: Selenium, Jira, BrowserStack

**Scope:**

In Scope:

* Functional testing of the flight booking process.
* Compatibility testing across major browsers (Chrome, Firefox, Safari).
* Usability testing to assess the user interface and overall user experience.

Out of Scope:

* Testing of functionalities other than flight booking.
* In-depth security testing (handled separately).

### 12.6 Test Case:

Test case for Flight Booking Functionality:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| TC001 | Perform a Flight Search | Departure: New York, Destination: London, Date: 2023-12-01 | Display list of available flights | List of flights displayed as expected | Passed |
| TC002 | Select a Flight | Choose the first available flight | Display selected flight details | Flight details displayed correctly | Passed |
| TC003 | Complete Booking | Passenger Name: John Doe, Payment: Visa Card, Additional Info: None | Confirm booking and display confirmation message | Booking confirmed; confirmation message displayed | Passed |

### 12.7 Risks and Assumptions:

Risks:

* Limited testing time may result in incomplete test coverage.
* Changes in the flight booking API may impact testing.

Assumptions:

* The development team provides timely builds for testing.
* Test environment is stable and reflects the production environment accurately.

## 13. Payment Terms & condition

* 15% payment will be accepted for the Project proposal and design Submission.
* 45% payment will be accepted for the Application Development
* 70% payment will be accepted after application review and Testing
* 100% payment will be accepted after handover the fully completed Application

## 14. Responsibility

The entire Application has been done by MMK Mahin and all the responsibility including terms and condition will goes to him.

## 15. Contact Us

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

Golam Rabbany

By Phone: +8801787774996

By Email

grabbany1234@gmail.com

## 16. Agreement Signed By:

|  |  |  |
| --- | --- | --- |
| ……………………………    Client Signature  Niloy sarker  Managing Director | …………………………..    Order Provider Signature  MMK Mahin  Officer  SoftTech | ……………………………    Authority Signature  Golam Rabbany  Managing Director (MD)  SoftTech |