# **Chapter 2: Analysis**

# **2.1 Introduction to Analysis**

Analysis is the second phase of the SDLC. This is performed after the planning phase. This phase is where the actual project lifecycle begins. This phase acts as the bridge between Planning and Design. After completing the planning phase, analysis phase begins and this phase is driven according to the plan. In this phase, we break down the deliverables in Project into the more detailed business requirements. The Analysis Phase is also the part of the project where we will identify the overall direction that the project will take through the creation of the project strategy documents. In this phase, we will study and determine the system requirements.  This helps to determine and understand what the customer wants from the project. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements. This will be the main focus point of this phase.

Our objectives in the analysis phases can be listed as below:

* To determine and document how the current system works.
* To determine how the system can perform better.
* To develop a logical or business model of the new system/project.
* To make recommendations for improvement.

The three major steps to achieve the above objectives are as follows:

* Determine the system requirements
* Analyze the system requirements
* Evaluate the alternative solutions

At last, an optional document is created that helps transition from the Analysis Phase to the more technical and detailed Design Phase. This document, also known as Conceptual System Design, provides client feedback into many of the ways that the final solution will be implemented. This feedback includes much of the look-and-feel of the final solution.

# **2.2 Methodology**

Among the different methodologies like Hard approach, Soft approach, combine approach, People-oriented, process-oriented, etc. I have chosen Object-oriented methodology for analysis phase of this project.

In this methodology, the system requirements are determined, the classes are identified and the relationships among classes are identified.Object-oriented methodologies do not focus solely on the processes or data of a system but view a system as a collection of interacting ***objects*** that work together to complete tasks.

The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

* **Object Modelling**

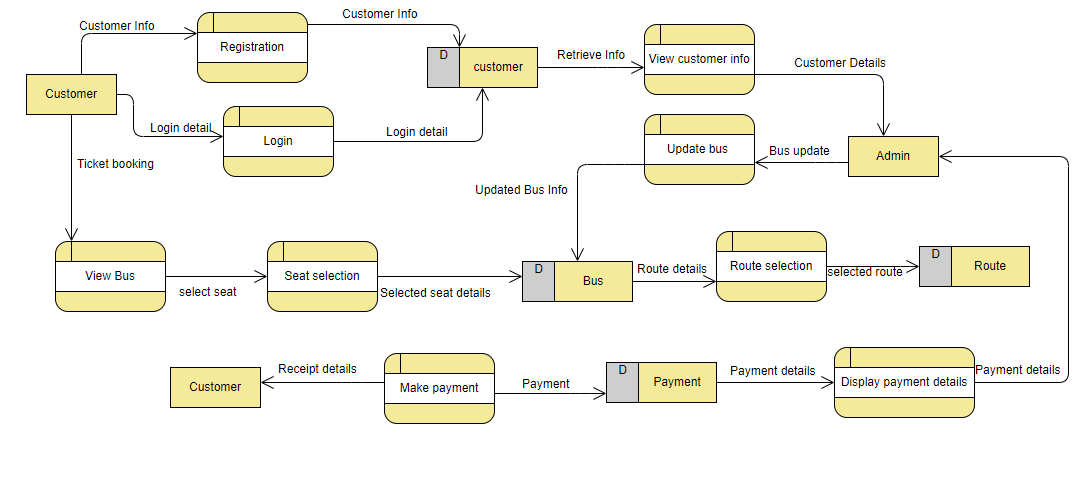
Object Modelling shows the object, classes, their relationships and attributes and operations as a *Class Diagram,* which represents the static structure of the system. The class diagram for this project is shown in another chapter.

* **Dynamic Modelling**

Dynamic modelling shows the behaviour of the system over time and the flow of control and events in *Event-Trace Diagrams* and *State Transition Diagrams (State Charts).* These diagrams will be shown in the design part of this project.

* **Functional Modelling**

Functional modelling is the final component of object-oriented analysis. It shows the internal processes independently from how these processes are performed with the help of Data Flow Diagram (DFD). I have created DFD diagram for this project as below:



# **2.3 Feasibility Study**

First of all, feasibility study refers to the analysis and evaluation of a proposed project to determine if it is technically possible, is possible within the estimated cost/budget, and will be profitable. It examines the practicability of any proposed projects or ideas. The main purpose of the feasibility study is to determine the project will continue or not.

The importance of the feasibility study can be shown as below:

* Shows how realistic your proposed project is.
* Helps to define goals.
* Determine the scope.
* Helps in plan development.
* Helps in plan execution.

# **2.4 Requirement Analysis**

## **2.4.1 Functional Requirement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| F1 | Database connection | Database connection means allowing system to interact with database. | To perform CRUD function | N/A |
| F2 | User Registration | Registration means registering the new user to the system to access the system. | To create new user | F1 |
| F3 | User login | Login is the entering of identifier info into system for accessing the system. | To access the system | F2 |
| F4 | Logout | Logout means to end the access to the system voluntarily. | To avoid security issues | F3 |
| F5 | Payment |  |  | F3 |
| F6 | Authentication | Authentication means identifying individual who is logging in. | To avoid unauthorized access to the system | F1 |
| F7 | Add buses | Only admin can add new buses in the system | To add new buses | F3 |
| F8 | Remove buses | Only admin can remove buses from the system | To delete buses | F3 |
| F9 | Update bus details | Only admin can update the bus details | To update bus details | F3 |
| F10 | Add route | Only admin can add route in the system | To add route | F3 |
| F11 | Remove routes | Only admin can remove the route from the system | To delete routes | F3 |
| F12 | Update route details | Only admin can update route details in the system | To update route details | F3 |
| F13 | Viewing bus details | Both admin and general users can view the bus details | To view the bus details | F3 |
| F14 | Searching routes | This function helps users to get their desired routes. | To search routes | F3 |
| F15 | Seat viewing | User can view whether seat is booked or available | To view seat details | F3 |
| F16 | Seat Booking | User can select available seats and book them. | To book seat | F3 |
| F17 | Cancel reservation (only available for admin) | Only admin can cancel the reservation as per user request | To cancel the reservation | F3 |

## **2.4.2 Non-functional Requirement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| N1 | Performance | System should have quick response time. | To make user more satisfied | N/A |
| N2 | Scalability | System should function well as user increases. | To handle the large number of the user effectively | N/A |
| N3 | Recoverability | System should be able to recover from a crash | To avoid effects of the possible threats of the system | N/A |
| N4 | Availability | System should be available anytime. | To provide better facility to the users | N/A |
| N5 | Reliability | System is expected to provide reliable data to user | To build trust within the users and good relation with them | N/A |
| N6 | Maintainability | System should be designed in such way that it can be maintained on a regular basis. | To ensure the condition of the system and improve the system | N/A |
| N7 | Security | System must have data security and have proper access control | To maintain data security and avoid unauthorized access | F2, F3 |
| N8 | Usability | System should be easy to learn and operate. | To make more user friendly | N/A |
| N9 | Interoperability | System should be designed in such way that it can work in integration with different systems. | To enhance the functionality of the system | N/A |
| N10 | Serviceability | Technical supports should be provided to install, configure and monitor system and other related facilities should be provided. | To make easy for the user to cope up with the system | N/A |
| N11 | Manageability | System should be easy to manage. | To ensure healthy monitoring, logging and alerting | N/A |
| N12 | Legal | System must be legal in the country. | To work in the market legally | N/A |
| N13 | Capacity | System must store data effectively and must anticipate the time remaining until all available storage is filled up. | To ensure that multiple users can use the system at the time | N/A |

## **2.4.3 MoSCoW Prioritization**

MoSCow prioritization is the popular prioritizing technique for understanding and managing requirements. It is also known as MoSCoW method or MoSCoW analysis.

This technique divides requirements into 4 categories:

* **Must-have**

This category includes those requirements has to be included in the product. They represents non-negotiable needs of the project.

* **Should-have**

Requirements under this category are important to the project but not vital as must-have requirements. The project still works without them but the project will be more effective if they are added.

* **Could-have**

These requirements are not necessary to the core function of the project. In other word, they are nice to have.

* **Would-have**

The requirements under this category are not included in a specific release but they are likely to be added in the future. They will get prioritized in the future.

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Title** | **MoSCoW** | **Remarks** |
| F1 | Database connection | Must have |  |
| F2 | User registration | Must have |  |
| F3 | User login | Must have |  |
| F4 | Logout | Must have |  |
| F5 | Payment | Must have |  |
| F6 | Authentication | Must have |  |
| F7 | Add buses | Must have |  |
| F8 | Remove buses | Must have |  |
| F9 | Update bus details | Must have |  |
| F10 | Add routes | Must have |  |
| F11 | Remove routes | Must have |  |
| F12 | Update route details | Must have |  |
| F13 | Viewing bus details | Must have |  |
| F14 | Searching routes | Must have |  |
| F15 | Seat viewing | Must have |  |
| F16 | Seat booking | Must have |  |
| F17 | Cancel Reservation | Must have |  |
| N1 | Performance | Should have |  |
| N2 | Scalability | Could have |  |
| N3 | Recoverability | Must have |  |
| N4 | Availability | Should have |  |
| N5 | Reliability | Should have |  |
| N6 | Maintainability | Should have |  |
| N7 | Security | Must have |  |
| N8 | Usability | Should have |  |
| N9 | Interoperability | Could have |  |
| N10 | Serviceability | Would have |  |
| N11 | Manageability | Would have |  |
| N12 | Legal | Must have |  |
| N13 | Capacity | Should have |  |

## **2.4.4 SRS (System Requirements Specification)**

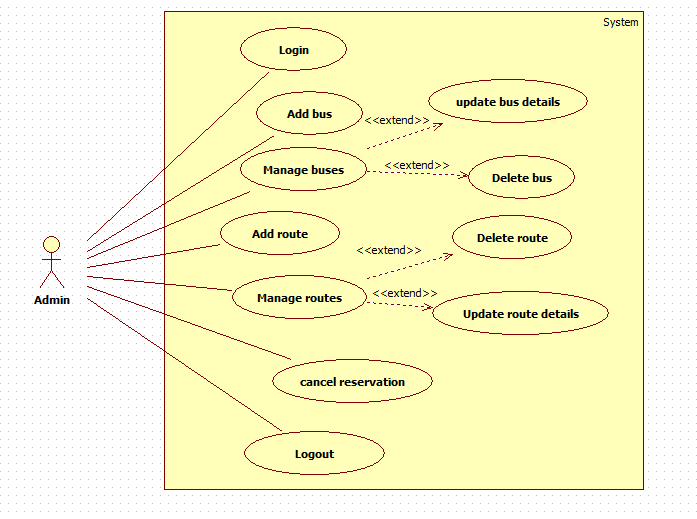
# **2.5 Use-case diagram**

Use case is the graphical representation of the interaction between users (actors) and the system. It shows what a user can do in the system. It summarizes the details of the users and their interaction with the system. The notations used in the use-case diagram are as follows:

* **Use cases:** It is horizontally shaped ovals that represent the different uses that a user might have.
* **Actors:** Stick figures that represent the people actually employing the use cases.
* **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* **System boundary:** It is shown as a box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system.

**The reasons behind using the use-case diagram are as follows:**

* Define the interaction between users and the system.
* Define the roles of the users in the system.
* Define and organize the functional requirements.
* Model the basic work flow in the system.

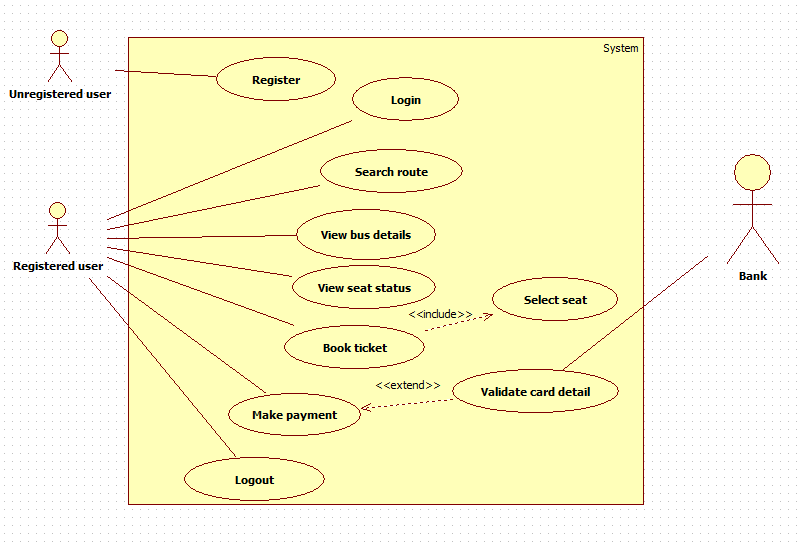


**Description:**

Actor: Admin

Admin is the super user of the system who can perform the following activities:

* Admin can login into the system.
* Admin can add buses in the system.
* Admin can manage the buses which includes update bus details and delete buses from the system.
* Admin can add routes in the system
* Admin can manage the routes which includes update route details and delete routes from the system.
* Admin can cancel the reservation on the request of the customer/passenger.
* Admin can logout from the system at his/her convenience.



**Description:**

Actor: customer (general user)

Customer is the general user of the system who can perform the following actions in the system:

* Unregistered user can register themselves in the system.
* Registered user can login into the system.
* Customer can view bus details for booking.
* Customer can view the seat status that is booked or available.
* Customer can book the ticket.
* Customer can make the payment for the booking.
* Customer can logout from the system as per their wish.

# **2.6 NLA and Initial class diagram**

Travel By Bus is the online bus ticket booking system. This is a web-based application where people can book ticket for bus seats for travel easily with less effort and time. This system is built for managing and computerizing the traditional database, ticket booking and travel made. The main aim of this system is to automate the manual booking reducing the paper works. It records and maintains all the customer details, bus details and booking (reservation) details. It also facilitates the online payment system.

There are two types of users in the system: General User and Admin. If user is not registered, then he/she has to register first. He/she has to register with some data like email, name, password, phone, etc. After registration, General user can book the ticket, search the routes, view seat status that is available or booked, and view the bus details. On the other side, Admin can add and manage buses (update and delete function), add and manage routes (update and delete function), and cancel the reservation. General user cannot cancel the reservation as they have to inform the ticket counter to do so.

|  |  |  |
| --- | --- | --- |
| **Candidate Classes** | **Candidate Method** | **Candidate attribute** |
| seat, payment, users, route, bus | Book, register, search, view, add, delete, update, cancel | email, name, password, phone |

