Ride - Online Bus Ticket Booking System

SRS Document

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document should be used as a guideline by the students to design the Solution Architecture for the project. The document also describes the broad scope of the project and high level logical object model. But while developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation.

Online Bus Ticket Booking System - Ride

Introduction

The purpose of this document is to define scope and requirements of an Online Bus Ticket Booking System for a State Road Transport Corporation (SRTC). The proposed system will provide online bus ticket booking on the lines of existing popular Rail or Airline ticket booking systems.

This document should be used by the development team to architect the solution the project.

Management Summary

While bus is a popular mode of transport for short and medium distance travels, it is still not easy to plan a bus journey easily in advance like train or flights. A state government took an initiative to transform its State Road Transport Corporation (SRTC) bus ticket booking system on the lines of railways and airline systems. The proposed system - Ride will:

1. Allow travelers to search available services on the basis of trip details (i.e. To/From, One/Two way, and preferred timings). The traveler should be able to book a trip for the selected service.
2. Allow cancellation of existing bookings.
3. Enable an automated delivery of e-tickets via e-mail.

It was proposed to develop a web based online system with one key objectives in mind - “simplify booking process for the traveler”. Ride will be designed & developed to run on IBM WebSphere Application Server and IBM DB2 Universal Database in a 2-tier architecture.

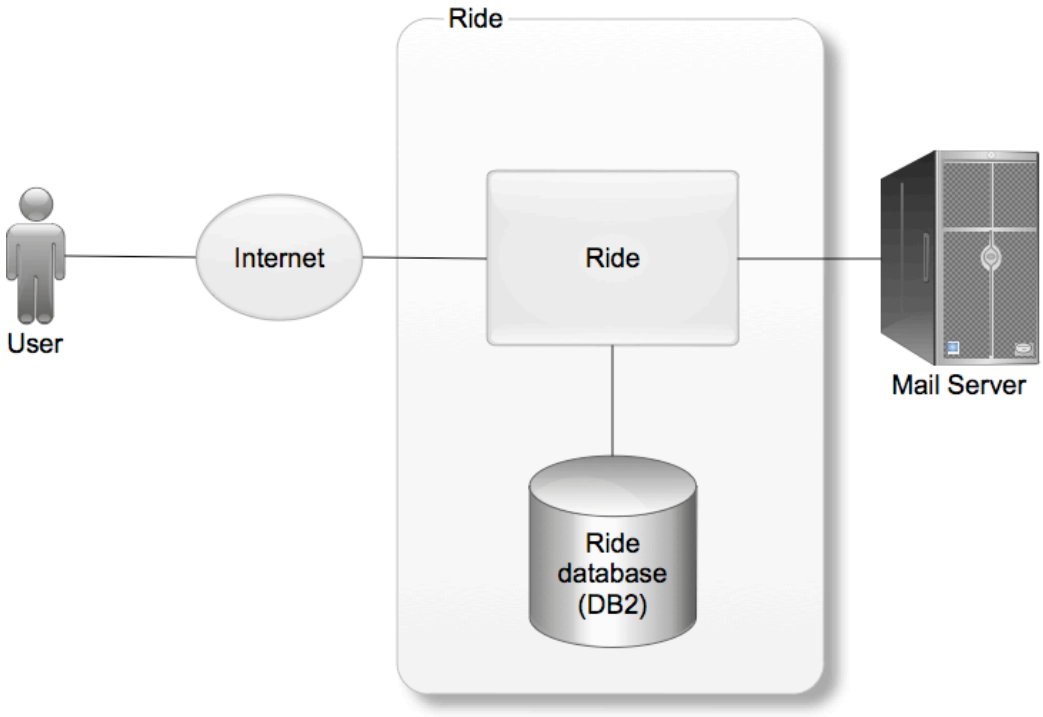
Key Assumptions

1. There is no integration with the payment gateway for this system; booking process will not involve any financial transaction.
2. While it is nice to have e-mail alerts & reminders to notify the passenger of change of schedule/delays or to remind of the journey timings a day before of the planned journey date, these features will not be incorporated in this system. However, adventurous developers are welcome to add such features.

High Level Architecture

Ride's high level logical architecture is illustrated through the diagram shown below. It will have following categories of users:

1. Travelers
2. SRTC Employees



Ride Context Diagram

Ride	The proposed web-based application, Ride will be accessed by the SRTC employees to setup routes, fares, services, fleet details etc on an ongoing basis. Travelers will be required to register themselves in order to access the online booking services.
Ride Database	This will hold all the Ride data including the masters like routes, fares, fleet details and the transactions like bookings from travelers.
Mail Server	Mail server will be used for sending booking confirmations via email.

Functional Requirements

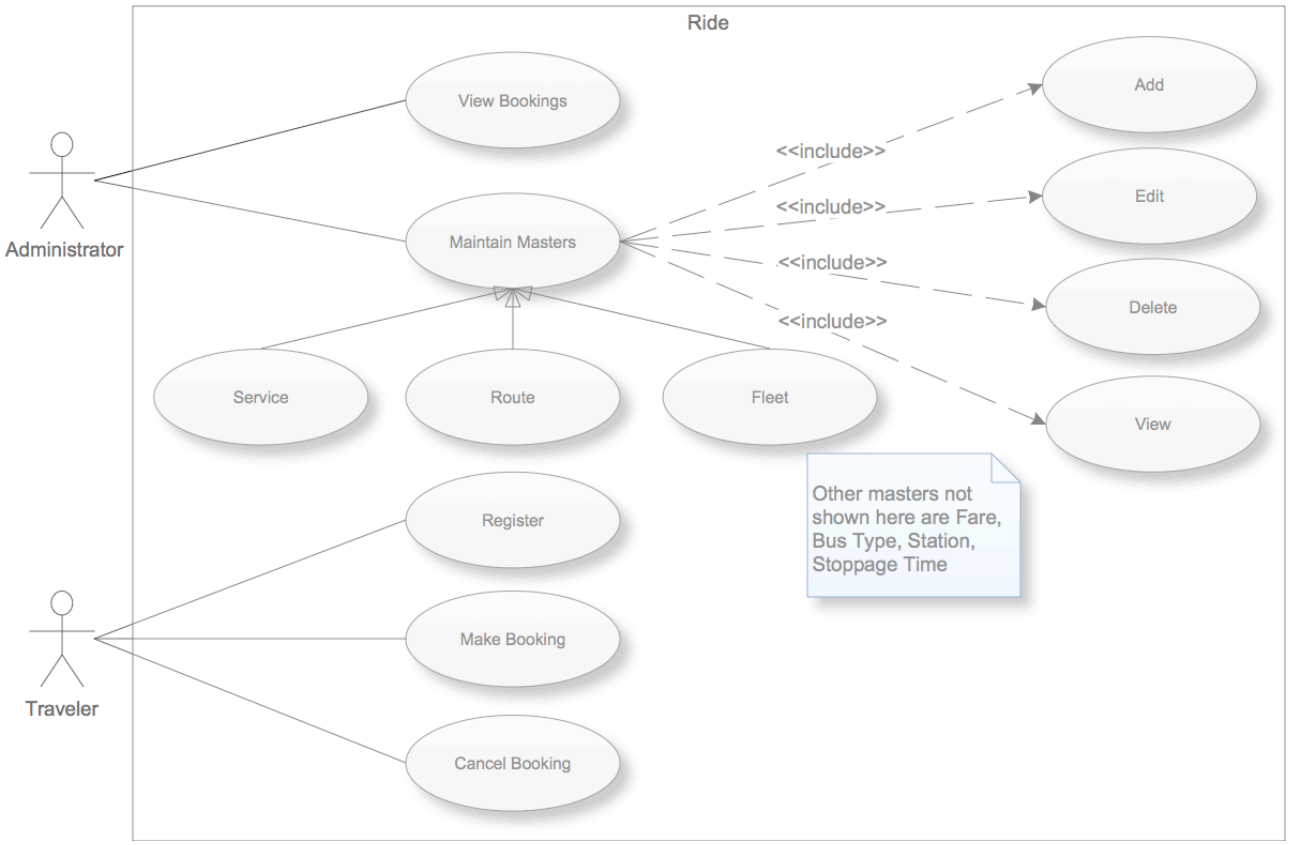
The high level functional requirements for the Ride system are outlined in the Use Case diagram described in this section.

Ride will provide a secure user-id/password based secured login mechanism to access its services. The details of this are not outlined here. The development team is expected to create these keeping in mind the general practices followed by the web applications. Login will be a prerequisite to use Ride. Internal users will be provided user id/password pair separately; whereas travelers will be required to register to obtain their user-id/password pair.

Once traveler logs in, s/he can view past bookings or make new bookings from “My Travel Page”. Note, this page is only visible after a traveler logs in to the system.

Use Case Diagrams

The following figure illustrates the Use Case diagram for the system.



Use Case Diagram

Use Cases

View Bookings

Use Case Element	Description
Number	UC.01
Application	To view all bookings by different criteria like by route, by bus type, etc.
Use Case Name	View Bookings
Primary Actor	Administrator
Secondary Actor	None
Pre-condition	None
Trigger	Administrator clicks on the **View Bookings** link on the admin interface page
Basic Flow	• System displays the criteria selection list box • User selects the desired criteria and click go button • The required view is displayed.
Alternate Flow	None
Output	None

Maintain Masters

Use Case Element	Description
Number	UC.02
Application	Master maintenance in terms of basic operations viz. **add**, **edit**, **delete** and **view**. All master maintenance i.e. route, station, stoppage time, fare, service, fleet and bus type are child use cases of this Use Case.
Use Case Name	Maintain Masters
Primary Actor	Administrator
Secondary Actor	None
Pre-condition	None
Trigger	Administrator clicks on the **Maintain Masters** menu item on the admin interface page

Use Case Element	Description
Basic Flow	• System presents a list of masters that can be maintained. Administrator selects the desired master. • System displays a list *view* and links for *add*, *edit* and *delete*. 1. In case add, a new master record data entry form is presented. The master record is saved on clicking the save button provided form clears all the data validations (if any). The list view is updated accordingly. 2. In case of edit, from the list view user is prompted to select the desired record to edit, Selected record is opened for editing. The edited master record is updated on clicking the update button, provided form clears all the data validations (if any). 3. In case of delete, from the list view user selects the check box(s) against each record. Selected records are deleted up on clicking the delete button. However, user is presented a confirmation dialog before deleting the records.
Alternate Flow	• In event of any error, it is clearly displayed and user is asked to renter data or perform operation again.
Output	System displays the details of the successful operation.

Register

Use Case Element	Description
Number	UC.03
Application	Traveler registers in the Ride System to gain access to its online bus ticket booking services.
Use Case Name	Register
Primary Actor	Traveler
Secondary Actor	None
Pre-condition	None
Trigger	Traveler clicks on the **Register** menu item on the home page
Basic Flow	• System displays the new registration page and traveler is asked to fill personal information such as name, address, e-mail, and user-id/password as per the prevailing web registration standards. • On clicking the submit button, the travelers profile is saved; s/he is registered and a welcome e-mail is sent.
Alternate Flow	• In event of any error, traveler is asked to make correction. • If travelers clicks cancel button then the operation is cancelled.
Output	Error message, in event of error. Welcome e-mail on successful registration.

Make Booking

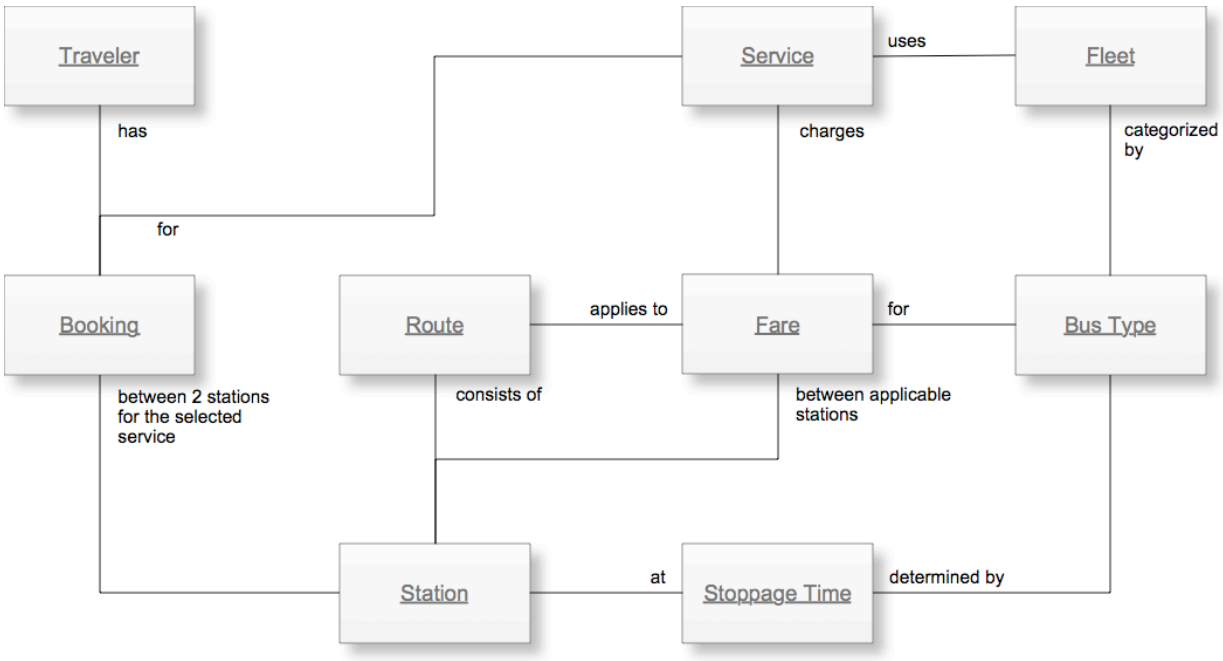
Use Case Element	Description
Number	UC.04
Application	To make booking; booking is carried out for one segment of the journey at a time, i.e. one source-destination pair.
Use Case Name	Make Booking
Primary Actor	Traveler
Secondary Actor	None
Pre-condition	None
Trigger	Traveler clicks on the **Book Ticket** link on “My Travel Page”
Basic Flow	• System displays search page that allows traveler to enter source, destination, date of travel and preferred timings. Upon clicking search, all possible list of services are listed in a view. [Please refer to Services as defined in Object Model] • Traveler is prompted to select the desired service and clicks on Make my Bookings link. The desired service is booked and appropriate record(s) are created in the system including blocking the requested number of seats in the desired service. • An e-mail containing e-ticket is sent to the traveler. This e-mail contains ticket number along with the journey details. • The details of this booking are now visible on the “My Travel Page”.
Alternate Flow	If no service is found then the traveler is prompted to revise his/her search criteria.
Output	E-Mail containing e-ticket.

Cancel Booking

Use Case Element	Description
Number	UC.05
Application	To cancel an existing booking made by the traveler.
Use Case Name	Cancel Booking
Primary Actor	Traveler
Secondary Actor	None
Pre-condition	None
Trigger	Traveler clicks on the **Cancel Ticket** link on “My Travel Page”
Basic Flow	• System displays all the future bookings. • Traveler is allowed to select a booking and click on cancel button. • System reconfirms cancellation. • On cancellation, the booking is released and seats are marked available in the service booked. An e-mail confirming cancellation is also sent to the traveler.
Alternate Flow	None
Output	E-Mail confirming cancellation details.

Logical Object Model

A high level logical object model of the system is shown below. During technical design it will be transformed into a physical model covering all system entities. Such a diagram will include their relationship and its cardinality.



Logical Object Model

1. A route is defined between the origin and the destination bus stations including all the in between bus stations where the bus has a schedule stop. It is identified against an unique route id.
2. Bus type categorizes a bus in the fleet. Examples are express, luxury, premium, etc.
3. The stoppage time on an en-route station depends on the bus type plying on that route.
4. Fare applies to a route for every applicable en-route bus station for a bus type.
5. Fleet defined by parameters such as bus registration number, its bus type and its seating capacity.
6. A service runs on a route using fleet and charges applicable fare.
7. Traveler is a user of bus services who has already registered on the Ride System.
8. Traveler has bookings for between 2 stations for the chosen service.

Database Design Guidelines

This involves the transformation of the use cases, state diagrams, and logical object model into detailed and optimized physical database table designs.

Typically persistent classes will map to table(s) with their attributes as columns of the table. In some cases a high level object may map in to a master-child table. Invoice is one such example where it maps in to "invoice_header" and "invoice_line_item" table.

Associations between two persistent objects are realized as foreign keys to the associated objects. A foreign key is a column in one table that contains the primary key value of the associated object.

Similarly, a standard technique in relational modeling is to use an intersection entity to represent many-to-many associations. Following is a broad checklist for physical database database design:

1. Database must be properly normalized except those instances where de-normalization help improves performance. This option must be used with special care.
2. All persistent classes that use the database for persistency must map to database structures.
3. Many-to-many relationships must have an intersecting table.
4. Primary keys should be defined for each table, unless there is a performance reason not to define a primary key.
5. Indexes should be defined to optimize access.
6. Data and referential integrity constraints should be defined.

Testing Approach

Quality of the software can be achieved with basic hygiene and consistency followed during design and development of User Interface(UI), Navigation, Validations as per the business process requirement.

To ensure the project delivers acceptable quality to the customer, its important to create a checklist of the conventions to be followed across. Common checks as below are for your reference during design and development:

Common Checks	Validation Type
Page Title is valid for the feature being provided on the page	UI
Order of the Data Entry Fields is logical as per the functionality being provided by the feature	UI
Order of the Display only Fields makes viewing and understanding easy for the user	UI
Spellings and Correctness of Label for the Data Entry and Display fields	UI
The labels are not wrapping onto another row thereby adding a blank row on the page	UI
The fields with drop down are displayed in single row instead of drop down coming on the next row	UI
Data Entry field basic validations are working i.e Text field /Numbers / Dates allow data for their type only	Functional
The dates are following a standard format dd/mmm/yyyy on all forms	UI
The color scheme of all forms i.e headers labels , alerts, entry fields are uniform throughout the application	UI

Common Checks	Validation Type
The action buttons for a New Data Entry Form are uniform for all forms that is allowing data entry	UI
The action buttons are performing the desired action e.g. "submit" is creating a new record if there are no errors and recording all the input fields, whereas 'cancel' is not creating a new record in the database	Functional
The links provided on the forms are opening correctly.	Functional
The data feed mechanism for Read and Write files is generating a log with count of entries.	Navigation

Suggested Technical Reading

The project is aimed at making the student understand concepts of Design and Development using IBM Rational tools, Web Sphere Application Server and DB2 Database. The following reading reference is easy to understand and should be read to get a clear understanding of capabilities of the tools and how you would leverage them to execute a project.

Technical Reference	URL to access
RAD - Tackling challenges of software development with Rational Application Developer for WebSphere Software	http://www.ibm.com/developerworks/rational/library/08/0926_ackerman-mahate/index.html
IBM Education Assistant - Rational Application Developer 7.5	http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp?topic=/com.ibm.iea.rad_v7/rad/rad75.html
RSA-Overview of Rational Software Architect for WebSphere Software Version 7.5	http://www.ibm.com/developerworks/rational/library/08/0926_arnold/index.html
Using the new features of UML Modeler in IBM Rational Software Architect Version 7.5	http://www.ibm.com/developerworks/rational/library/08/0926_diu/index.html
Rational Technical Library	http://www.ibm.com/developerworks/rational/library/