Lab MST Worksheet

Student Name: Sahil Kaundal UID: 21BCS8197

Branch: CSE (Lateral Entry) Section/Group:20BCS-807\_B Semester: 4th Date of Performance: 21/03/2022

Subject Name: SE Lab Subject Code: 20CSP-255

1. Aim/Overview of the practical:

Development of SRS document.

1. Task to be done/ Objective:



To Develop an SRS document.

1. Requirement Analysis:

Software Requirement:

* Smart Draw
* Google Chrome

Hardware Requirement:

* Computer
* Windows 10
* Power Supply

1. Software Requirements Specification (SRS):

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for [software](https://www.techtarget.com/searchapparchitecture/definition/software) under development. The SRS fully describes what the software will do and how it will be expected to perform.

An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, [portability](https://www.techtarget.com/searchstorage/definition/portability), maintainability, [footprint](https://whatis.techtarget.com/definition/footprint), security and speed of recovery from adverse events are evaluated. Methods of defining an SRS are described by the [IEEE](https://whatis.techtarget.com/definition/IEEE-Institute-of-Electrical-and-Electronics-Engineers) (Institute of Electrical and Electronics Engineers) specification 830-1998.

Key components of an SRS

The main sections of a software requirements specification are:

* Business drivers – this section describes the reasons the customer is looking to build the system, including problems with the currently system and opportunities the new system will provide.
* [Business model](https://whatis.techtarget.com/definition/business-model) – this section describes the business model of the customer that the system has to support, including organizational, business context, main business functions and process flow diagrams.
* Business/functional and system requirements -- this section typically consists of requirements that are organized in a hierarchical structure. The business/[functional requirements](https://whatis.techtarget.com/definition/functional-requirements) are at the top level and the detailed system requirements are listed as child items.
* Business and system [use cases](https://www.techtarget.com/searchsoftwarequality/definition/use-case) -- this section consists of a Unified Modeling Language ([UML](https://www.techtarget.com/searchsoftwarequality/definition/Unified-Modeling-Language)) use case diagram depicting the key external entities that will be interacting with the system and the different use cases that they’ll have to perform.
* [Technical requirements](https://whatis.techtarget.com/definition/technical-requirements) -- this section lists the non-functional requirements that make up the technical environment where software needs to operate and the technical restrictions under which it needs to operate.
* System qualities -- this section is used to describe the non-functional requirements that define the quality attributes of the system, such as reliability, serviceability, security, scalability, availability and maintainability.
* Constraints and assumptions -- this section includes any constraints that the customer has imposed on the system design. It also includes the requirements engineering team’s assumptions about what is expected to happen during the project.
* Acceptance criteria -- this section details the conditions that must be met for the customer to accept the final system.

1. Purpose of an SRS:

An SRS forms the basis of an organization’s entire project. It sets out the framework that all the development teams will follow. It provides critical information to all the teams, including development, operations, quality assurance ([QA](https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance)) and maintenance, ensuring the teams are in agreement.

Using the SRS helps an enterprise confirm that the requirements are fulfilled and helps business leaders make decisions about the [lifecycle](https://whatis.techtarget.com/definition/product-lifecycle) of their product, such as when to retire a feature.

In addition, writing an SRS can help developers reduce the time and effort necessary to meet their goals as well as save money on the cost of development.

SRS Template

The following is a simple SRS template:

Table of Contents

1. Introduction

1.1 Purpose of this document

1.2 Scope of this document

1.3 Overview

1.4 Business Context

2. General Description

2.1 Product Functions

2.2 Similar System Information

2.3 User Characteristics

2.4 User Problem Statement

2.5 User Objectives

2.6 General Constraints

3. Functional Requirements

4. Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Communications Interfaces

4.4 Software Interfaces

5. Performance Requirements

6. Other non-functional attributes

6.1 Security

6.3 Reliability

6.4 Maintainability

6.5 Portability

6.6 Extensibility

6.7 Reusability

6.8 Application Affinity/Compatibility

7. Operational Scenarios

8. Preliminary Use Case Models and Sequence Diagrams

8.1 Use Case Model

8.2 Sequence Diagrams

9. Updated Schedule

10. Appendices

10.1 Definitions, Acronyms, Abbreviations

10.2 References

Features of an SRS

An SRS should have following characteristics:

* Correct -- should accurately reflect product functionality and specification at any point of time.
* Unambiguous -- should not be any confusion regarding interpretation of the requirements.
* Complete -- should contain all the features requested by a client.
* Consistent -- same abbreviation and conventions must be followed throughout the document.
* Ranked for importance and/or stability -- every requirement is important. But some are urgent and must be fulfilled before other requirements and some could be delayed. It’s better to classify each requirement according to its importance and stability.
* Verifiable -- an SRS is verifiable only if every stated requirement can be verified. A requirement is verifiable if there is some method to quantifiably measure whether the final software meets that requirement.
* Modifiable -- an SRS must clearly identify each and every requirement in a systematic manner. If there are any changes, the specific requirements and the dependent ones can be modified accordingly without impact the others.
* Traceable – an SRS is traceable if the origin of each of its requirements is clear and if it makes it easy to reference each requirement in future development.

The goals of an SRS

Some of the goals an SRS should achieve are to:

* Provide feedback to the customer, ensuring that the IT company understands the issues the software system should solve and how to address those issues.
* Help to break a problem down into smaller components just by writing down the requirements.
* Speed up [the testing](https://www.techtarget.com/searchsoftwarequality/tip/Pros-and-cons-of-requirements-based-software-testing) and validation processes.
* Facilitate reviews.

1. Example:

## INTRODUCTION

* 1. **PURPOSE**

The purpose of this document is to build an online system to manage flights and passengers to ease the flight management. <<*Include the purpose as applicable to your project*>>

* 1. **DOCUMENT CONVENTIONS**

This document uses the following conventions. <<*Include the conventions as per your application*>>

|  |  |
| --- | --- |
| DB | Database |
| DDB | Distributed Database |
| ER | Entity Relationship |

* 1. **INTENDED AUDIENCE AND READING SUGGESTIONS**

This project is a prototype for the flight management system and it is restricted within the college premises. This has been implemented under the guidance of college professors. This project is useful for the flight management team and as well as to the passengers.

* 1. **PROJECT SCOPE**

The purpose of the online flight management system is to ease flight management and to create a convenient and easy-to-use application for passengers, trying to buy airline tickets. The system is based on a relational database with its flight management and reservation functions. We will have a database server supporting hundreds of major cities around the world as well as thousands of flights by various airline companies. Above all, we hope to provide a comfortable user experience along with the best pricing available.

**1.5 REFERENCES**

* <https://krazytech.com/projects>
* Fundamentals of database systems by ramez elmarsi and shamkant b.navathe

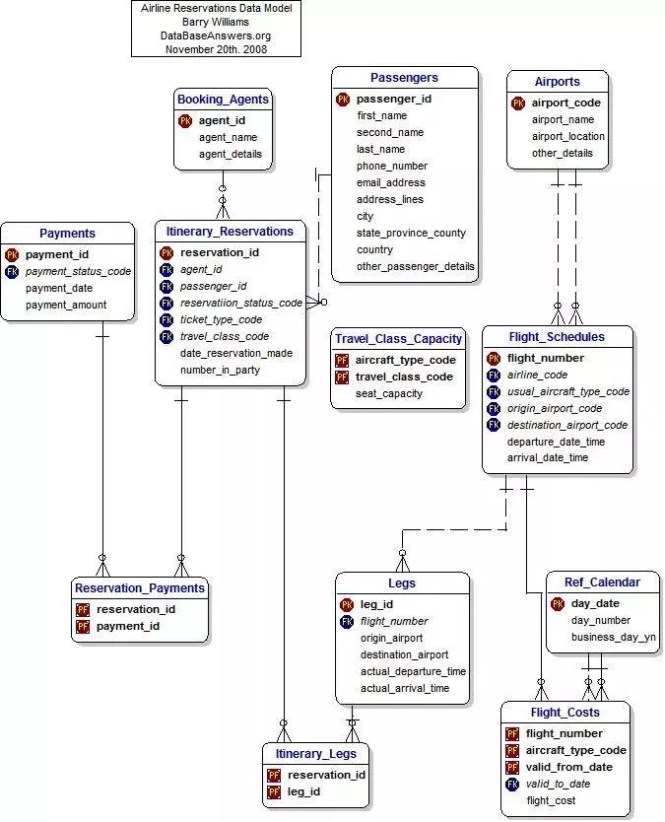
## OVERALL DESCRIPTION

* 1. **PRODUCT PERSPECTIVE**

A distributed airline database system stores the following information.

* Flight details:  
  It includes the originating flight terminal and destination terminal, along with the stops in between, the number of seats booked/available seats between two destinations etc.
* Customer description:  
  It includes customer code, name, address and phone number. This information may be used for keeping the records of the customer for any emergency or for any other kind of information.
* Reservation description:  
  It includes customer details, code number, flight number, date of booking, date of travel.
  1. **PRODUCT FEATURES**

The major features of airline database system as shown in below [**entity–relationship model**](https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model) (**ER model**)

[](https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database/attachment/ad-layout-of-airline-database-system)

* 1. **USER CLASS and CHARACTERISTICS**

Users of the system should be able to retrieve flight information between two given cities with the given date/time of travel from the database. A route from city A to city B is a sequence of connecting flights from A to B such that: a) there are at most two connecting stops, excluding the starting city and destination city of the trip, b) the connecting time is between one to two hours. The system will support two types of user privileges, Customer, and Employee. Customers will have access to customer functions, and the employees will have access to both customer and flight management functions. The customer should be able to do the following functions:

* Make a new reservation  
  • One-way  
  • Round-Trip  
  • Multi-city  
  • Flexible Date/time  
  • Confirmation
* Cancel an existing reservation
* View his itinerary

The Employee should have following management functionalities:

* CUSTOMER FUNCTIONS.  
  • Get all customers who have seats reserved on a given flight.  
  • Get all flights for a given airport.  
  • View flight schedule.  
  • Get all flights whose arrival and departure times are on time/delayed.  
  • Calculate total sales for a given flight.
* ADMINISTRATIVE  
  • Add/Delete a flight  
  • Add a new airport  
  • Update fare for flights.  
  • Add a new flight leg instance.  
  • Update departure/arrival times for flight leg instances.

Each flight has a limited number of available seats. There are a number of flights which depart from or arrive at different cities on different dates and time.

**2.4 OPERATING ENVIRONMENT**

Operating environment for the airline management system is as listed below.  <<*Include the details as per your application*>>

* distributed database
* client/server system
* Operating system: Windows.
* database: sql+ database
* platform: vb.net/Java/PHP

**2.5 DESIGN and IMPLEMENTATION CONSTRAINTS**

1. The global schema, fragmentation schema, and allocation schema.
2. SQL commands for above queries/applications
3. How the response for application 1 and 2 will be generated. Assuming these are global queries. Explain how various fragments will be combined to do so.
4. Implement the database at least using a centralized database management system.

**2.6 ASSUMPTION DEPENDENCIES**

Let us assume that this is a distributed airline management system and it is used in the following application:

* A request for booking/cancellation of a flight from any source to any destination, giving connected flights in case no direct flight between the specified Source-Destination pair exist.
* Calculation of high fliers (most frequent fliers) and calculating appropriate reward points for these fliers.

Assuming both the transactions are single transactions, we have designed a distributed database that is geographically dispersed at four cities Delhi, Mumbai, Chennai, and Kolkata as shown in fig. below.

## SYSTEM FEATURES

* **DESCRIPTION and PRIORITY**

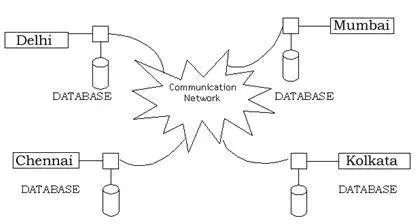
The airline reservation system maintains information on flights, classes of seats, personal preferences, prices, and bookings. Of course, this project has a high priority because it is very difficult to travel across countries without prior reservations.

* **STIMULUS/RESPONSE SEQUENCES**
  + Search for Airline Flights for two Travel cities
  + Displays a detailed list of available flights and make a “Reservation” or Book a ticket on a particular flight.
  + Cancel an existing Reservation.
* **FUNCTIONAL REQUIREMENTS**

Other system features include:

**DISTRIBUTED DATABASE:**

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network as shown in below figure.

[](https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database/attachment/ad-distributed-database-located-at-four-different-cities)

**CLIENT/SERVER SYSTEM**

The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).

A client/server system is a distributed system in which,

* Some sites are client sites and others are server sites.
* All the data resides at the server sites.
* All applications execute at the client sites.

## EXTERNAL INTERFACE REQUIREMENTS

* 1. **USER INTERFACES**
* Front-end software: Vb.net version
* Back-end software: SQL+

**4.2 HARDWARE INTERFACES**

* Windows.
* A browser which supports CGI, HTML & Javascript.

**4.3 SOFTWARE INTERFACES**

Following are the software used for the flight management online application. <<*Include the software details as per your project*>>

|  |  |
| --- | --- |
| **Software used** | **Description** |
| Operating system | We have chosen Windows operating system for its best support and user-friendliness. |
| Database | To save the flight records, passengers record we have chosen SQL+ database. |
| VB.Net | To implement the project, we have chosen Vb.Net language for its more interactive support. |

**4.4 COMMUNICATION INTERFACES**

This project supports all types of web browsers. We are using simple electronic forms for the reservation forms, ticket booking etc.

## NONFUNCTIONAL REQUIREMENTS

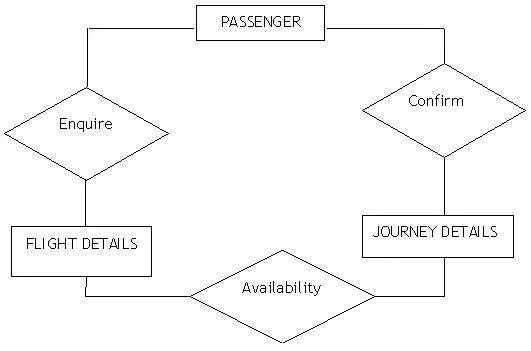
* 1. **PERFORMANCE REQUIREMENTS**

The steps involved to perform the implementation of airline database are as listed below.

1. **E-R DIAGRAM**

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

* ENTITIES: Which specify distinct real-world items in an application.
* PROPERTIES/ATTRIBUTES: Which specify properties of an entity and relationships.
* RELATIONSHIPS: Which connect entities and represent meaningful dependencies between them.

[](https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database/attachment/ad-er-diagram-of-airline-database)

**B) NORMALIZATION:**

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

* 1. **SAFETY REQUIREMENTS**

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

* 1. **SECURITY REQUIREMENTS**

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

**5.4 SOFTWARE QUALITY ATTRIBUTES**

* AVAILABILITY: The flight should be available on the specified date and specified time as many customers are doing advance reservations.
* CORRECTNESS: The flight should reach start from correct start terminal and should reach the correct destination.
* MAINTAINABILITY: The administrators and flight in chargers should maintain correct schedules of flights.
* USABILITY: The flight schedules should satisfy a maximum number of customer’s needs.

1. Result/Output/Writing Summary:

I have successfully done this practical.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):



|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |