Software Requirements Specification

for

Automatic Train Scheduling Simulator

Submitted to: Dr. Gaurav Harit, Asst. Professor, IITJ

Prepared by: Abhishek Kumar | Gaurav Shastri

Table of Contents

1.0. Introduction

1.1. Purpose

This Software Requirements Specification provides a complete description of all the functions and specifications of Automatic Train Scheduling Simulator.

The expected audience of this document is the faculty of CS, Dr. Gaurav Harit and the developers, Gaurav Shastri & Abhishek Kumar.

1.2. Scope

The Automatic Train Scheduling Simulator will automatically manage all the trains passing through any particular station & will display the real time graphical layout of all the processes. The management process will include most optimal scheduling & collision prevention. User can dynamically update the following software parameters:

- Number of Trains
- Number of Platforms & Number of outer lines
- Arrival & Departure info of any Train

A database will be deployed to store the required attributes of all the trains and platforms. This automated system can also be further used for upcoming Metro projects in India.

1.3. Glossary

Term	Definition
App	Application
ATSS	Automatic Train Scheduling Simulator
DS	Display Screen
DB	Database
GUI	Graphical User Interface
SQL	Structured Query Language

1.4. References

- [IEEE] The applicable IEEE standards are published in "IEEE Standards Collection," 2001 edition.
- [www.techwhirl.com] Writing Software Requirement Specifications
- [Sample SRS] Jacksonville State University Computing and Information Sciences (CIS) Web Accessible Alumni Database.

1.5. Document overview

The remainder of this document is two chapters, the first providing a full description of the project. It lists all the functions performed by the system. The final chapter concerns details of each of the system functions and actions in full for the software developers' assistance. These two sections are cross-referenced by topic; to increase understanding.

2.0. Overall description

The ATSS encompasses a main App file and some database files containing all information related to the App. It also uses a server for handling the database files. The software product is a new, self-contained independent product which is not a part of any other software product and does not act as an interface. It is an online App & will require an Internet connection for serving the purpose.

2.1. System environment

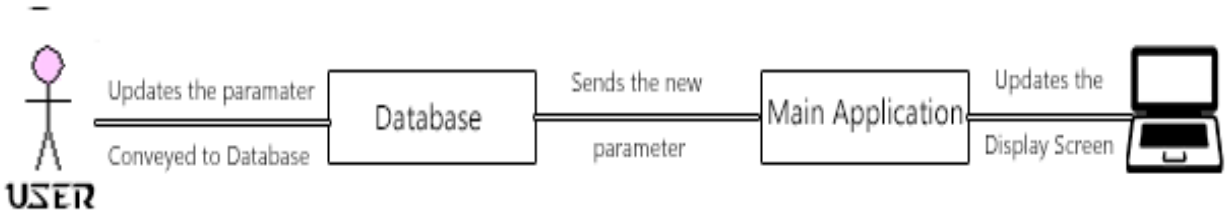

Fig 2.1 System Design and DS update on change of parameter

The ATSS App will simulate the real time motion of trains and their behavior on arrival & departure. On its own it will use a job scheduling algorithm to manage the trains. When the user tweaks with parameters, the corresponding values stored in the database will also be changed and then this will be communicated to main App via SQL. After getting the new data, main App will use it to update the current scenario. There will be plenty of parameters which can be altered by user & corresponding changes will be reflected in real time in the behaviour of all the components of the system.

2.2. Functional requirements definitions

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

2.3. Use cases

The App will consist of a main DS and a section of software parameters which can be updated by the user.

The first parameter which can be altered is the time factor which can be tweaked by the user using a slider. The position marker on the slider indicates the multiplication factor between real world time and the passage of time on main DS. The lowest position on the slider indicates passage of 10 minutes of real world time in 1 second on simulator which will increase subsequently when the marker is slid upwards.

The second parameter which can be altered is the number of platforms displayed on the main DS. The change can be incorporated using a simple user input box where user can put a specific number of platform which should lie within a permissible limit. As soon as user enters the data, the data is sent to the database and the change will be conveyed to the main DS using SQL. In the response, the main program will increase or decrease the number of platforms displayed and in case the number of platforms is decreased, it will check for the possibility that whether the change is feasible or not considering the number of empty platforms present at the moment.

The third parameter which can be changed is the total number of trains present during the runtime of the App. The number of trains can be increased one by one using a simple button. As soon as user presses the button, a form will pop out in which user can input all the parameters of the new train. After this, the whole data is conveyed to the database which in turn is passed to the main App which updates the DS.

The fourth parameter which can be changed is the parameters of existing trains. The change can be incorporated using a dropdown list. Once a train is selected from this list of trains, all the existing parameters of the train are displayed in an editable form format. User can update any attribute of the train by updating the value inthis form. After this, the whole data is conveyed to the database which in turn is passed to the main App which updates the DS.

2.3.1. Use Case: Changing the time factor

Brief Description

User uses a slider to change the time factor ratio between real world and simulator.

Initial step-by-step description

1. The user moves the marker on the slider to desired position.
2. The slider sends the marker position to the database.
3. The marker position is stored in the database & is passed to the main App.
4. The time factor on DS is updated by main App using this information.

2.3.2. Use Case: Changing the number of platforms

Brief Description

User clicks on plus or minus button to increase or decrease the number of platforms.

Initial step-by-step description

1. The user clicks on any of the given buttons.
2. As the user clicks on the button, a messaged is passed to database depending on which button is pressed.
3. The value of number of platforms stored in the database is changed accordingly & is passed to the main App.
4. The main App uses this information to update the number of platforms on DS if feasible.

2.3.3. Use Case: Adding new trains

Brief Description

User clicks on plus button to add a new train & then fills up a form to provide the values of the attributes of the new train.

Initial step-by-step description

1. The user clicks on the plus button.
2. As the user clicks the plus button an input form is displayed with fields for the attributes of the new train.
3. The user fills the form & clicks submit.
4. The form sends the values entered to the database.
5. The value is stored in the database & is passed to the main App using SQL.
6. The main App uses this information to add a new train on DS if feasible.

2.3.4. Use Case: Changing the attributes of existing trains

Brief Description

User uses a dropdown list to select an existing train & then edits the existing attributes of the train to the desired value.

Initial step-by-step description

1. The user clicks on the drop down list and chooses one of the trains.
2. As he chooses one of the trains and presses enter an input form is displayed with fields containing existing attributes of the train.
3. The user edits the fields in the form to the desired value & clicks submit.
4. The form sends the values entered to the database.
5. The value is stored in the database & is passed to the main App using SQL.
6. The main App uses this information to reschedule the whole system and DS if required.

2.4. Non-functional requirements

There are requirements that are not functional in nature. Specifically, these are the constraints, the system must work within. Often these requirements must be achieved at a system-wide level rather than at a unit level.

As far as performance requirements are considered, the App must be compatible with both Linux and Windows. App must follow the constraints provided by the internet security of the location where it is deployed.

The delay between fetching data from database and scheduling in real time should not be very large (lesser response time)

2.5. Assumptions and Dependencies

These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. We assume that there is very less delay between updating parameters on DS and the actual values in DB. We also assume that the number of Outer Lines is 5 and the number of platforms is within a range. The PyQt has four basic dependencies namely, Python 2.7, Qt 4.8.6, SIP 4.16.1 and MySQL which can be resolved by installing them properly.

3.0. Requirement specifications

3.1. External interface specifications

None

3.2. Functional Requirements

3.2.1. Changing the time factor

Use Case Name:	Changing the time factor
Priority	Essential
Trigger	Slider with a marker on it.
Precondition	App is connected to internet & the database.
Basic Path	1. The user moves the marker on the slider to desired position. 2. The slider sends the marker position to the database. 3. The marker position is stored in the database & is passed to the main App using SQL. 4. The main App uses this information to update the time factor on DS.
Alternate Path	N/A
Postcondition	Time factor is updated on the DS
Exception Path	1. If there is a connection failure the time factor remains as it is and is not updated according to user's choice with a warning popup displayed about connection failure.
Other	
Reference	SRS 2.3.1

3.2.2. Changing the number of platforms

Use Case Name	Changing the number of platforms
Priority	Essential
Trigger	1. A plus signed button. 2. A minus signed button.
Precondition	App is connected to internet & the database.
Basic Path	1. The user clicks on any of the given buttons. 2. As the user clicks on the button, a messaged is passed to database depending on which button is pressed. 3. The value of number of platforms stored in the database is changed accordingly & is passed to the main App using SQL. 4. The main App uses this information to update the number of platforms on DS if feasible.
Alternate Path	N/A
Postcondition	Number of platforms displayed is updated on the DS.
Exception Path	1. If there is a connection failure, the screen is not updated & a warning popup is displayed about connection failure.

	2. If the negative sign button is pressed and no platform is empty at the moment, the DS is not updated and a pop up containing a warning is displayed.
Other	
Reference	SRS 2.3.2

3.2.3. Adding new trains

Use Case Name:	Adding new trains
Priority	Essential
Trigger	1. A plus signed button. 2. A simple user input form with required fields.
Precondition	App is connected to internet & the database.
Basic Path	1. The user clicks on the plus button. 2. As the user clicks the plus button an input form is displayed with fields for the attributes of the new train. 3. The user fills the form & clicks submit. 4. The form sends the values entered to the database. 5. The value is stored in the database & is passed to the main App using SQL.

	6. The main App uses this information to add a new train on DS if feasible.
Alternate Path	N/A
Postcondition	Number of trains displayed is updated on the DS.
Exception Path	1. If there is a connection failure the screen is not updated & a warning popup is displayed about connection failure. 2. If all the platforms and the lines at outer signal are already occupied the DS is not updated and train is simply added in the waiting queue.
Other	
Reference	SRS 2.3.3

3.2.4. Changing the attributes of existing trains

Use Case Name	Changing the attributes of existing trains
Priority	Essential
Trigger	1. A dropdown list. 2. An editable form with required fields.
Precondition	App is connected to internet & the database.
Basic Path	1. The user clicks on the drop

	down list and chooses one of the trains. 2. As he chooses one of the trains and presses enter an input form is displayed with fields containing existing attributes of the train. 3. The user edits the fields in the form to the desired value & clicks submit. 4. The form sends the values entered to the database. 5. The value is stored in the database & is passed to the main App using SQL. 6. The main App uses this information to reschedule the whole system and DS if required.
Alternate Path	N/A
Postcondition	The whole scenario displayed is updated on the DS.
Exception Path	1. If there is a connection failure the screen is not updated & a warning popup is displayed about connection failure.
Other	
Reference	SRS 2.3.4

3.3. Detailed non-functional requirements

Attribute Name	Attribute Type	Attribute Size
Arrival Time* (in 24-hr format)	Int	4
Departure Time* (in 24-hr format)	Int	4
Source Station*	String	1
Destination Station*	String	1

Fields marked with an '*' are required fields.

Server	Local Server (Xampp)/ Web Server
Operation System	Window XP or above/Linux Distribution(debian)
Internet Connection	Any internet connection
Code Standard	The main App will be developed using Py-Qt (Python with Qt) framework. The SQL queries will be achieved using both Py-Qt & PHP. The Database will of MySQL format.
Performance	The App should update the simulator DS 100% of time when the query is feasible.

3.4. System Evolution

In the future this App can be expanded for use in real life situations like upcoming metro projects in India. It can also be expanded in the way where a user can view any of the stations in real time out of all stations where any train can travel.