# Software Requirements Specification

For the Online Event Ticket Booking System

Authors: Lizzie Long and Michael Dye

Version: 1.0

# Table of Contents

**1. Introduction....2**
1.1 Purpose....3
1.2 Intended Audience....3
1.3 Intended Use....3
1.4 Product Scope....3
1.5 Definitions and Acronyms....4
**2. Overall Description....5**
2.1 User Needs....5
2.2 Assumptions and Dependencies....6
**3. System Features and Requirements....7**
3.1 Functional Requirements....7
3.2 Non-Functional Requirements....8
3.3 External Interface Requirements....8
3.3.1 User Interfaces....8
3.3.2 Software Interfaces....8
3.3.3 Communication Interfaces....9
3.4 System Features....9
**4. Other Requirements....10**
4.1 Database Requirements....10
4.2 Legal and Regulatory Requirements....10
4.3 Internationalization and Localization....10
4.4 Risk Management (FMEA Matrix)....10
**5. Appendices....11**
5.1 Glossary....11
5.2 Use Cases and Diagrams....11
5.3 To Be Determined (TBD) List....11

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Requirement Specification document is to detail the requirements, description, and functions of the Online Event Ticket Booking System. This system is being created as a part of an ENCE 420 Software Engineering course at Anderson University to create an efficient, secure, and user-friendly website for members of the Anderson University community to browse University events, book seats for events and cancel reservations. The administrators should also be able to manage events and view customer bookings. This SRS document will outline the design, implementation, and testing of the Event Ticket Booking System.

## 1.2 Intended Audience

The audience for this document includes the developers of the program, Michael Dye and Lizzie Long, as well as the Project Manager and professor, Dr. Vudumu. This document could be helpful for future developers in the Anderson University Computer Engineering program who might work on similar projects. End users may also reference this document for system features and capabilities.

## 1.3 Intended Use

This document will be used by the development team to maintain structure and consistency throughout the project. It will help in monitoring progress and ensuring that all team members stay aligned with the project goals. The SRS will also help developers identify and prioritize the functions that are most critical to the Online Event Ticket Booking System, reducing the risk of spending too much time on non-essential subtasks.

In addition, the SRS provides a clear reference point for instructors and evaluators to understand the planned functionality, and it can guide testers during the validation process. By defining intended use early, this document ensures that the project remains focused, efficient, and manageable within the given timeline.

## 1.4 Product Scope

This Software Requirements Specification pertains to the development of the Online Event Ticket Booking System. The scope of this project encompasses the creation of a web-based application that will serve as the primary platform for customers to discover events, reserve tickets, and manage their bookings, while providing administrators with the necessary tools to manage event inventory and analyze sales performance.

This SRS is intended to specify the requirements for the software to be developed. It defines the capabilities of the system that will allow for the online browsing, booking, and management of event tickets. The standard set forth in this document can be used to create the software directly or serve as a model for project-specific standards. It does not mandate a specific methodology, nomenclature, or tool for preparing the SRS or implementing the system.

## 1.5 Definitions and Acronyms

- Customer: A user who can log in, browse events, book tickets, cancel bookings, and view booking history. This could include students and faculty.
- Administrator: A user who can add, edit, or remove events, configure ticket details, and generate sales reports. This might include Anderson University staff members or event coordinators.
- Booking: Reserving one or more tickets for an event.
- SRS: Software Requirements Specification.
- UI/UX: User Interface / User Experience design.

## 2. Overall Description

This document has been created in response to the project requirements set forth for the Software Engineering course. It defines the problem statement of managing event ticketing manually and proposes a software solution as specified in the assignment guidelines. It identifies the intended user roles (Customer and Administrator) as defined by the professor and interprets the provided functional and non-functional requirements as the core needs of the system. This section also provides a high-level overview of the major features the system will implement to fulfill the project's objectives.

The following SRS contains the detailed product perspective as derived from the project assignment. It provides the product functions of the Online Event Ticket Booking System, along with user characteristics, constraints, assumptions, and dependencies as they relate to the academic scope of this project.

### 2.1 User Needs

The project requirements define the needs for two distinct user roles, which form the basis for all system functionality:

- **Customers**, as defined by the assignment, require the ability to browse a list of events, view available seating, book tickets for a chosen event, cancel reservations before the event date, and view their booking history.

- **Administrators**, as defined by the assignment, require the ability to add, update, and delete events, configure ticket prices and seating limits, and generate reports of ticket sales. Their need is for a system that ensures data integrity and provides accurate sales information for reporting purposes.

## 2.2 Assumptions and Dependencies

**Assumptions:**

- The system is a prototype for a class project and will not be deployed in a production environment.
- All user and event data is simulated for demonstration and testing purposes.
- The "payment gateway simulation" and "email confirmation" (optional features) will be simulated processes without actual financial transactions or email delivery.
- The professor and teaching assistants will act as the validating stakeholders for all requirements.

**Dependencies:**

- The successful development of the project is dependent on the use of the suggested technology stack (e.g., Flask/Django for backend, SQLite/PostgreSQL for database) or an approved equivalent.
- The system's functionality is dependent on the accurate implementation of the professor-provided requirements as detailed in Section 3 of this document.
- The performance requirement to "support at least 200 concurrent bookings" is dependent on the capabilities of the local development and testing environment (e.g., personal computers, university servers).
- The project's grade is dependent on fulfilling all mandatory deliverables outlined in the assignment (SRS, Design, Implementation, Testing, Final Report).

# 3. System Features and Requirements

## 3.1 Functional Requirements

Customer Requirements:

- **Register/login:** The website should check customers' credentials to prevent outside sources from accessing the booking system. It should check for account type and load up the corresponding portal (customer or admin).
- **Browse list of events:** Customers should be able to view a list of up-to-date University events.
- **Choose an event and view the number of available seats:** The system should allow a customer to choose the desired event from the event list, see the number of seats that are available, and book an available seat. Customers should be able to go back to the event list if they clicked the wrong event.
- **Cancel booking before event date:** Customers should be able to cancel their event reservations before the date/time of the event. There should also be a message that warns them of their cancellation ("Are you sure you want to cancel?")
- **View booking history:** Customers should be able to see the events they have reserved seats for in the past.

Admin Requirements:

- **Add, update, and delete events:** Admins should be able to add new University events, update them, and cancel them. Updates and cancellations should generate alerts for those who signed up for those events.
- **Set ticket price and seat limits:** Admins should set the prices and seat limits for each event. These should be clearly displayed for customers to see.
- **Generate sales report:** Admins should be able to see the history of ticket bookings for past events.

## 3.2 Non-Functional Requirements

- **Security:** Password protection and safe data handling.
- **Usability:** Simple and intuitive booking interface.
- **Reliability:** Prevent overbooking and ensure accurate transactions.
- **Performance:** Support at least 200 concurrent bookings.

Optional features:

- **Email confirmation:** Customers who sign up for an event should be notified with a confirmation email.
- **Seat map visualization:** An admin should be able to add in a live seat map visualization so that specific seats could be reserved. Along with this, the specific seat number should be included in the customer's confirmation email.
- **Payment simulation:** Tickets should be paid for with apple pay, a card, etc using a simulation.
- **Admin Booking Functionality:** Admin accounts should be able to select a button to view the page as a customer and make bookings

## 3.3 External Interface Requirements

Due to this project's scope as a Software Engineering course assignment, at a functional level, there are very few external interface requirements as almost everything will be simulated internally. There are a few exceptions to this:

### 3.3.1 User Interfaces

The primary user interface shall be a web application accessible through modern web browsers, including but not limited to Chrome, Firefox, Safari, and Edge. The interface shall be built using HTML, CSS, and JavaScript and shall provide a consistent experience across these platforms.

### 3.3.2 Software Interfaces

- **Database:** The system shall interface with a relational SQL database management system (SQLite for development and testing, with the potential for PostgreSQL in production) for all persistent data storage, including user credentials, event details, and booking records.
- **Web Server:** The application backend, built using Python Flask/Django, shall interface with a WSGI-compatible web server (e.g., Gunicorn) for handling HTTP requests.
- **(Optional) Payment Simulation:** If the optional payment gateway simulation is implemented, the system shall interface with a simple internal module that mimics the API calls of a payment processor, returning successful or failed transaction statuses.

- **(Optional) Email Simulation:** If the optional email confirmation feature is implemented, the system shall interface with a simple internal module that logs confirmation messages to the console or a file, simulating the process of sending an email.

### 3.3.3 Communication Interfaces

The system shall communicate using the HTTP and HTTPS protocols over standard port 80 (HTTP) and port 443 (HTTPS). All client-server communication shall adhere to this standard.

## 3.4 System Features

# 4. Other Requirements

## 4.1 Database Requirements

## 4.2 Legal and Regulatory Requirements

## 4.3 Internationalization and Localization

## 4.4 Risk Management (FMEA Matrix)

# 5. Appendices

## 5.1 Glossary

## 5.2 Use Cases and Diagrams

## 5.3 To Be Determined (TBD) List