Movie Booking Web Application

Version 1.2

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 26/03/2023 | 1.0 |  | Enache Mihai |
| 16/04/2023 | 1.0 | Changed project idea | Enache Mihai |
| 01/05/2023 | 1.1 | Back to original project idea | Enache Mihai |
| 20/05/2023 | 1.2 | Final touched on doc | Enache Mihai |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

# Introduction

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.

# Non-functional Requirements

## Availability

**Quality Attribute**: Availability

**Source of Stimulus**: External

**Stimulus**: High traffic, server failure, maintenance

**Environment**: Production environment

**Artifact**: Platform servers and infrastructure

**Response**: The platform should remain available and responsive to user requests, even during high traffic, server failure, or scheduled maintenance

**Response Measure**: The platform should aim to achieve a high uptime percentage, with minimal downtime for maintenance and updates

**Tactics**:

* Use load balancers, cloud infrastructure, and redundant servers to distribute traffic and ensure high availability
* Implement monitoring tools and alerts to detect and respond to server failures and other issues

## Performance

**Quality Attribute**: Efficiency

**Source of Stimulus**: External

**Stimulus**: User requests search results and booking

**Environment**: Production environment

**Artifact**: Platform software and database

**Response**: The system should respond to user requests within a reasonable amount of time.

**Response** **Measure**: The platform should aim to achieve fast load times for content, user data, and other metadata, and minimize the time it takes to render pages and search results

**Tactics**:

* Optimize code and database queries to improve performance
* Use caching to store frequently accessed data and reduce database requests

## Security

**Quality** **Attribute**: Security

**Source** **of** **Stimulus**: External

**Stimulus**: Hacking attempts, malware, data breaches

**Environment:** Production environment

**Artifact**: Platform servers, database, and user data

**Response**: The platform should protect user data and content from hacking attempts, malware, and other threats

**Response** **Measure**: The platform should aim to achieve a high level of security, with regular updates and audits of security measures

**Tactics**:

* Use encryption to protect sensitive user data
* Implement secure login and authentication processes
* Use firewalls and intrusion detection systems to protect against attacks
* Implement security best practices for code development and deployment.

## Testability

**Quality** **Attribute**: Maintainability

**Source** **of** **Stimulus**: Internal

**Stimulus**: Code changes and updates

**Environment**: Development and testing environment

**Artifact**: Platform software and infrastructure

**Response**: The platform should be easy to test and maintain, with clear code structure and documentation

**Response** **Measure**: The platform should be easy to update and maintain, with minimal downtime for updates and minimal risk of introducing new bugs or issues

**Tactics**:

* Use automated testing tools to test new code changes and updates
* Use version control tools to track changes and roll back changes if necessary
* implement coding standards and documentation guidelines to ensure maintainability

## Usability

**Quality** **Attribute**: Usability

**Source** **of** **Stimulus**: External

**Stimulus**: User requests for content, search results, and navigation

**Environment**: Production environment

**Artifact**: Platform user interface and user experience

**Response**: The platform should be easy to use and navigate, with clear and intuitive search and browsing functions

**Response** **Measure**: The platform should aim to achieve high user satisfaction and engagement, with low bounce rates and high time on site metrics

**Tactics**:

* Use user-centered design principles to create a clear and intuitive user interface
* Implement effective search and browsing functions
* Conduct user testing and feedback to improve the user experience

# Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

Design constraints for the movie booking web application:

* **Backend technology**: The system must be developed using Java and the Spring Boot framework.
* **Frontend technology**: The system must be developed using Angular for the user interface
* **Database management system**: The system must use MySQL Workbench database.
* **Development tools**: The development team must use Intellij IDEA or Eclipse for the backend, and Visual Studio Code for the frontend.
* **Security**: The system must implement specific security measures. This will be obtained with help of the Spring Security.
* **Performance**: The system must be designed to handle a high volume of concurrent user requests, with specific performance targets such as response time and throughput.