Software Requirements Specification

for

<Cake store management Website Application>

Version 1.0 approved

Prepared by Thai Binh Duong

<organization>

<07/06/2024>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 2

1.1 Purpose 2

1.2 Document Conventions 2

1.3 Intended Audience and Reading Suggestions 2

1.4 Project Scope 2

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 User Classes and Characteristics 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 Assumptions and Dependencies 2

3. System Features 2

3.1 System Feature 1 2

3.2 System Feature 2 (and so on) 2

4. Data Requirements 2

4.1 Logical Data Model 2

4.2 Data Dictionary 2

4.3 Reports 2

4.4 Data Acquisition, Integrity, Retention, and Disposal 2

5. External Interface Requirements 2

5.1 User Interfaces 2

5.2 Software Interfaces 2

5.3 Hardware Interfaces 2

5.4 Communications Interfaces 2

6. Quality Attributes 2

6.1 Usability 2

6.2 Performance 2

6.3 Security 2

6.4 Safety 2

6.5 [Others as relevant] 2

7. Internationalization and Localization Requirements 2

8. Other Requirements 2

Appendix A: Glossary 2

Appendix B: Analysis Models 2

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| First SRS | 29/05/2024 | Initial SRS document |  |
|  |  |  |  |

# Introduction

<The introduction presents an overview to help the reader understand how the SRS is organized and how to use it.>

## Purpose

This release focuses on enhancing the customer experience through online ordering, personalized recommendations, and seamless payment options. It also optimizes inventory management, automates order tracking, and provides comprehensive sales analytic to support data-driven business decisions. The project addresses challenges such as fluctuating demand, inventory wastage, and time-consuming administrative tasks, with the vision of future releases to further enhance functionality and customer satisfaction.

## Document Conventions

|  |  |
| --- | --- |
| **ER** | **Entity Relationship** |
| **DB** | **Database** |

**1.3 Intended Audience and Reading Suggestions**

This document is intended for:

❖ Development team: Responsible to develop detailed design, implement and perform unit

test, integration-test and system test for the migrated application

❖ Documentation Team: Responsible for writing User Guide for the application.

## 1.4 Project Scope

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Release 1 | Release 2 | Release 3 |
| FE-1 Feedback and Rating System | Users can provide feedback and rate their experience with the bakery's products and services through a basic feedback form.. | Implementation of a star-rating system and detailed review options. Admins can view and respond to customer feedback. | Advanced analytic on feedback and ratings to identify trends and areas for improvement. Integration with customer loyalty programs to reward users for providing feedback. |
| FE-2, Inventory Management | Admins can monitor and manage inventory levels in real-time. Basic alerts for low stock levels. | Enhanced reporting features for inventory management, including usage trends and order forecasts. Integration with supplier systems for automated reordering of supplies. | Advanced inventory optimization tools, such as predictive analytic to prevent overstocking or stock outs. Multi-location inventory management for larger operations with multiple bakery locations. |
| FE-3, Sales and Revenue Reporting | The system can generate basic sales and revenue reports, including daily, weekly, and monthly summaries. | Advanced reporting features with customization report generation options. Integration with financial software for seamless data transfer and analysis. | Enhanced account features and personalized recommendations. |

## 1.5 References

● COS Vision and Scope

# Overall Description

<This section presents a high-level overview of the product and the environment in which it will be used, the anticipated users, and known constraints, assumptions, and dependencies.>

## Product Perspective

<Describe the product's context and origin. Is it the next member of a growing product line, the next version of a mature system, a replacement for an existing application, or an entirely new product? If this SRS defines a component of a larger system, state how this software relates to the overall system and identify major interfaces between the two. Consider including visual models such as a context diagram or ecosystem map to show the product's relationship to other systems.>

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product and describe their pertinent characteristics. Some requirements might pertain only to certain user classes. Identify the favored user classes. User classes represent a subset of the stakeholders described in the vision and scope document. User class descriptions are a reusable resource. If available, you can incorporate user class descriptions by simply pointing to them in a master user class catalog instead of duplicating information here.>

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform; operating systems and versions; geographical locations of users, servers, and databases; and organizations that host the related databases, servers, and websites. List any other software components or applications with which the system must peacefully coexist. If extensive technical infrastructure work needs to be performed in conjunction with developing the new system, consider creating a separate infrastructure requirements specification to detail that work.>

## Design and Implementation Constraints

<Describe any factors that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing or memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; programming language requirements or restrictions.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, reuse expectations, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors outside its control.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, stimulus, response, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

### Description

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority.>

### Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

<Itemize the specific functional requirements associated with this feature. These are the software capabilities that must be implemented for the user to carry out the feature's services or to perform a use case. Describe how the product should respond to anticipated error conditions. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

## System Feature 2 (and so on)

# Data Requirements

<This section describes various aspects of the data that the system will consume as inputs, process in some fashion, or create as outputs.>

## Logical Data Model

<A data model is a visual representation of the data objects and collections the system will process and the relationships between them. Include a data model for the business operations being addressed by the system, or a logical representation for the data that the system itself will manipulate. Data models are most commonly created as an entity-relationship diagram.>

## Data Dictionary

<The data dictionary defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements that make up those structures. In many cases, you're better off storing the data dictionary as a separate artifact, rather than embedding it in the middle of an SRS. That also increases its reusability potential in other projects.>

## Reports

<If your application will generate any reports, identify them here and describe their characteristics. If a report must conform to a specific predefined layout you can specify that here as a constraint, perhaps with an example. Otherwise, focus on the logical descriptions of the report content, sort sequence, totaling levels, and so forth, deferring the detailed report layout to the design stage.>

## Data Acquisition, Integrity, Retention, and Disposal

<If relevant, describe how data is acquired and maintained. State any requirements regarding the need to protect the integrity of the system's data. Identify any specific techniques that are necessary, such as backups, check pointing, mirroring, or data accuracy verification. State policies the system must enforce for either retaining or disposing of data, including temporary data, metadata, residual data (such as deleted records), cached data, local copies, archives, and interim backups.>

# External Interface Requirements

<This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements.>

## User Interfaces

Admin Dashboard: The admin dashboard should be intuitive and user-friendly, offering a comprehensive overview of the cake store's operations. It should provide easy access to various modules such as order management, inventory tracking, staff management, and sales analytics.

Staff Portal: The staff portal should provide functionalities for staff to update inventory, manage baking schedules, process orders, and track their working hours and attendance.

Customer Interface: The customer interface should be engaging and easy to navigate. It should feature online cake ordering with customization options, a gallery of available cakes, detailed product descriptions, customer reviews, and a secure payment system.

Baker Portal: The baker portal should allow bakers to access and manage baking schedules, view order details, update cake status, and manage ingredient inventory.

Delivery Interface: The delivery interface should enable delivery personnel to view delivery schedules, access customer addresses and contact information, update delivery status, and track delivery routes.

Analytic Dashboard: The analytic dashboard should provide detailed reports and insights into sales performance, customer preferences, inventory levels, and staff productivity to support data-driven decision-making.

## Software Interfaces

The Cake Store Management Website Application interfaces with various software components to ensure seamless operation, data exchange, and integration with external systems. This section describes the connections between the application and other software components, including applications, databases, operating systems, tools, libraries, websites, and integrated commercial components.

|  |  |
| --- | --- |
| **Software Used** | **Description** |
| Operating System | **Windows/Linux:** The application is designed to run on both Windows and Linux operating systems for flexibility and robust support. |
| Database | **MySQL:** MySQL is used to store data related to customers, orders, inventory, products, and employees. It ensures reliable data management and retrieval. |
| Back-end Framework | **Spring Boot:** Spring Boot is chosen for its ability to create production-ready applications with minimal configuration, offering interactive and efficient support for back-end operations. |
| Web Server and Hosting Environment | **Apache/Tomcat:** The web server hosts the Cake Store Management website, handling HTTP/HTTPS requests and responses, ensuring efficient and secure communication. |
| Front-end Framework | **React.js:** React.js is utilized for building a responsive and dynamic user interface, providing a smooth user experience. |

## Hardware Interfaces

**Web Browsers:**

* Description: Devices used by customers and staff to access the Cake Store Management System through web browsers.
* Logical Characteristics: Must be compatible with popular web browsers such as Chrome, Firefox, and Safari.
* Physical Characteristics: Can be accessed from computers, smartphones, and tablets.

**Database Server:**

* Description: The server that stores and manages the website's database, including customer information, orders, and inventory.
* Logical Characteristics: Compatible with MySQL database management system.
* Physical Characteristics: High-performance server hardware with sufficient storage capacity for the database.

## Communications Interfaces

**Web Browsers:**

● Compatibility: Supports all major web browsers like Chrome, Firefox, Safari, and Edge.

Forms: Uses simple forms for cake orders, feedback, and contact inquiries.

**Email Communications:**

● Notifications: Sends emails for order confirmations, shipping updates, promotions, and password resets.

→ *These communication interfaces ensure the Cake Store Management Website Application is secure, efficient, and easy to use for customers and staff.*

# Quality Attributes

## Usability

The Cake Store Management Website Application is designed to be user-friendly, ensuring a positive and efficient experience for all users. The following requirements outline the key characteristics that contribute to the application's usability

**Ease of Use:**

* Simple navigation and clear instructions.
* Consistent design for easy familiarity.

**Ease of Learning:**

* Quick on-boarding with guides.
* Familiar design patterns for quick learning.

**Memorability:**

* Recognizable icons and patterns.
* Save user preferences and past orders.

**Error Handling:**

* Real-time form validation.
* Clear error messages with guidance.
* Option to undo actions.

**Efficiency:**

* Fast loading times.
* Streamlined, simple checkout process.

**Accessibility:**

* Screen reader compatibility.
* Keyboard navigation support.
* High contrast and adjustable text sizes.

**Ergonomics:**

* Mobile responsive design.
* Touch-friendly buttons and elements.

**Design Standards:**

* Consistent UI design.
* Immediate visual feedback for actions.

## Performance

The Cake Store Management Website Application has specific performance requirements to ensure a smooth and efficient user experience. These requirements are detailed below for various system operations:

**Page Load Time:**

* **Requirement:** All web pages should load within 2 seconds under normal conditions.

**Order Processing:**

* **Requirement:** Order submission and confirmation should be completed within 2 second.

**Search Functionality:**

* **Requirement:** Search results should be displayed within 1 second of query submission.

**Inventory Updates:**

* **Requirement:** Inventory updates should be reflected on the website within 5 seconds.

**Database Queries:**

* **Requirement:** Database queries should be executed within 1 second.

**Concurrent Users:**

* **Requirement:** The system should support up to 500 concurrent users without performance degradation.

**Scalability:**

* **Requirement:** The system should be scalable to accommodate increased user load and data volume.

## Security

<Specify any requirements regarding security or privacy issues that restrict access to or use of the product. These could refer to physical, data, or software security. Security requirements often originate in business rules, so identify any security or privacy policies or regulations to which the product must conform. If these are documented in a business rules repository, just refer to them.>

## Safety

<Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product. Define any safeguards or actions that must be taken, as well as potentially dangerous actions that must be prevented. Identify any safety certifications, policies, or regulations to which the product must conform.>

# Other Requirements

<Examples are: legal, regulatory or financial compliance, and standards requirements; requirements for product installation, configuration, startup, and shutdown; and logging, monitoring and audit trail requirements. Instead of just combining these all under "Other," add any new sections to the template that are pertinent to your project. Omit this section if all your requirements are accommodated in other sections. >

Appendix A: Glossary

<Define any specialized terms that a reader needs to know to understand the SRS, including acronyms and abbreviations. Spell out each acronym and provide its definition. Consider building a reusable enterprise-level glossary that spans multiple projects and incorporating by reference any terms that pertain to this project.>

Appendix B: Analysis Models

<This optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams. You might prefer to insert certain models into the relevant sections of the specification instead of collecting them at the end.>