Real-Time Food Recipe & Budgeting System
Requirements Document

UVIC CSC

2024-04-27

Table of Contents

List of Tables

List of Figures

Revision History

Name	Date	Reason for Changes	Version
All	2024-04-27	Requirements Document 1.0	0.0

1 Introduction

1.1 Purpose

Product: "Real-Time Food Recipe Making & Budgeting System"

The purpose of the Real-Time Food Recipe Making & Budgeting System is to create software that can manage recipes and their costs in real-time. This system will continuously update ingredient prices from suppliers. Additionally, the software might provide users with the option to adjust recipes to the portion size they require.

1.2 Project Scope

The Real-Time Food Recipe Making & Budgeting System project aims to develop a sophisticated software solution enabling real-time management of recipes and associated costs. The system will feature seamless integration with supplier databases, ensuring continuous updates of ingredient prices to reflect real-time market fluctuations. Key functionalities include a user-friendly interface for recipe creation, modification, and cost estimation. Additionally, the system will potentially incorporate a dynamic portion adjustment feature, allowing users to customize recipes to their desired serving sizes. The project scope encompasses the development of robust backend algorithms for data integration and processing, along with an intuitive frontend interface for user interaction. Quality assurance measures will be implemented to ensure accuracy and reliability in cost calculations and ingredient updates. Throughout the development process, emphasis will be placed on scalability, security, and usability to deliver a comprehensive solution that meets the evolving needs of users in the food industry.

1.3 Glossary of Terms

1. **Real-Time Food Recipe Making & Budgeting System:** The overarching software solution designed to manage recipes and associated costs in real-time.
2. **Ingredients:** Any individual component used in the preparation of a recipe.
3. **Supplier:** A business or entity that provides ingredients to the system. Suppliers' prices are continuously updated within the system.
4. **Recipe:** A set of instructions detailing the ingredients and steps required to prepare a dish.
5. **Cost:** The monetary value associated with the procurement of ingredients for a recipe.
6. **Portion Size:** The quantity or serving size of a recipe, which users may adjust to suit their needs.
7. **User:** An individual who interacts with the Real-Time Food Recipe Making & Budgeting System, including administrators and regular users.
8. **Backend:** The server-side component of the system responsible for data processing, integration with supplier databases, and business logic implementation.
9. **Frontend:** The client-side component of the system, providing the user interface for recipe management, adjustment, and cost estimation.
10. **API:** Application Programming Interface, facilitating communication between the system and external supplier databases for real-time ingredient price updates.
11. **Dynamic Portion Adjustment:** The feature allows users to modify recipe serving sizes, with corresponding adjustments in ingredient quantities and costs.
12. **Budgeting:** The process of setting spending limits or budgets for recipes or ingredient categories to manage

expenses effectively.

13 **Authentication:** The process of verifying the identity of users accessing the system, ensuring data security and user privacy.

14 **Access Control:** The mechanism for defining and enforcing user permissions and privileges within the system, regulating access to features and functionalities.

15 **Quality Assurance:** The process of ensuring the reliability, accuracy, and usability of the system through testing and validation procedures.

16 **Scalability:** The system's ability to handle increased workload and user demand without sacrificing performance or functionality.

17 **Usability:** The ease with which users can navigate and interact with the system to accomplish their tasks effectively and efficiently.

18 **Security:** Measures implemented to protect system data, prevent unauthorized access, and ensure the confidentiality and integrity of user information.

1.4 References

1.5 Overview

The Real-Time Food Recipe Making & Budgeting System is a user-friendly software solution designed to streamline recipe management and enhance cost control capabilities. Its primary objective is to enable users to manage recipes and their associated costs in real-time, ensuring accuracy and efficiency. Through seamless integration with supplier databases, the system provides continuous updates of ingredient prices, reflecting market fluctuations instantly. Key features include an intuitive interface for recipe creation, modification, and cost estimation, with potential dynamic portion adjustment options for enhanced customization. The project encompasses the development of robust backend algorithms for efficient data integration and processing, complemented by an intuitive frontend interface for seamless user interaction.

This comprehensive solution not only empowers users with precise cost control but also addresses key challenges faced by the food industry. By enabling real-time management of recipes and associated costs, the system offers valuable insights and flexibility to businesses. Quality assurance measures are integral to ensuring accuracy and reliability throughout the system, with emphasis placed on scalability, security, and usability. Ultimately, the Real-Time Food Recipe Making & Budgeting System aims to deliver a comprehensive software solution that meets the evolving needs of users in the food industry, providing them with the tools they need to succeed in a competitive market landscape.

2 Overall Description

2.1 Product Perspective

This software is a powerful tool for recipe production, offering a user-friendly interface for editing recipe-making methods in table format. By leveraging public data from major supermarkets, it retrieves the

prices of each ingredient, allowing users to calculate the production cost reference for different types of foods. Additionally, users can easily share their recipes across various social media platforms, fostering a community of recipe enthusiasts and professionals exchanging ideas and techniques. This comprehensive approach not only aids in cost-effective production but also encourages collaboration and creativity within the catering community.

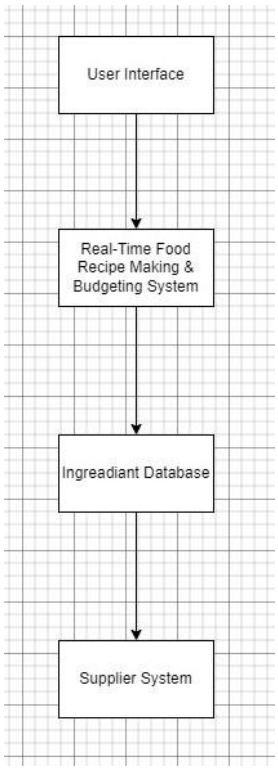

Figure 2.1 Simplified representation of how RTFRBS interfaces with other components on the Internet.

2.2 Product Features

The Real-Time Food Recipe Making & Budgeting System (RTFRBS) consists of three major components:

User Interface Application:

- The Application the user friendly Interface to help the user to create or modify the recipe in the way they needed.
- The Application also provides a one-click function to help users to share their recipe to the social media platform (X, Facebook, etc…).
- This customized interface allows users to link the RTFRBS, which will allow users to access their data by using different platforms such as Windows, MacOS, Linux, etc…

Recipe management system:

- Allow users to custom recipes,users can make different recipes by using various ingredients.
- Allow users to modify the created recipe to the portion of size they wanted.
- Price calculate, Obtain real-time price information of various raw materials through the database and calculate the required reference prices.

Background database:

- The database stores various of ingredients and retrieve the real-time price from the supplier
- The database also store the customize ingredients for each user, but the user have to input the price manually
- The database will allow users to manually input the quantities of the ingredients storage by themselves. This function might be automatically done in the future system.

The 3 systems will support each other to update ingredients and recipes prices in real time

2.3 User Classes and Characteristics

The Real-Time Food Recipe Making & Budgeting System (RTFRBS) is anticipated to be used by several distinct user classes, each with their unique characteristics and roles:

User:

Who uses this software and they are able modify the existing recipes to the portion size they need. Also the user is able to create the new recipes and share them to the social media platform (X, Facebook, etc…).

Role: user

Developer:

Who develops and maintains RTFRBS.

Role: developer

Supplier:

Databases from major supermarkets provide real-time ingredient information.

Role: For data reference only.

2.4 Operating Environment

The running environment of this software is most likely daily life and kitchen, so this software should focus more on the development of web and mobile terminals.

2.5 Design and Implementation Constraints

The design and implementation of the Real-Time Food Recipe Making & Budgeting System (RTFRBS) are subject to various constraints that shape the development process.

Hardware Restrictions:

Cross-platform development is one of the constraints, because our software needs to run on both the computer and mobile terminals, this thing will be difficult to develop. Not only because the systems are different, but also the various performances of the mobile terminal are different from the computer terminal, and the code needs to

be rewritten on a large scale.

Data Retrieve:

Retrieving and calculating data from external websites is also a problem, because the data we need in each website may be stored differently, and the structure of each web page is different. So we need different crawlers for each different situation to get the information we need. Also some websites might have some security measures to prevent their data from being read.

Security Protocols:

The most common thing is that information should not be leaked during transmission. At the same time, since this software has the function of sharing to social media, the information of private social accounts and user data should also be protected from leaks.

2.6 Assumptions and dependencies

Availability of Third-Party Components:

Assumption: The availability and functionality of third-party components or software libraries for specific functionalities, such as Data Retrieve integration.

Impact: If these components are unavailable or do not meet requirements, it could affect the project's development and capabilities..

Software Component Reuse:

Assumption: The reusability and compatibility of certain software components from other projects to expedite development.

Impact: Incompatibility or unforeseen issues with reused components could impact development timelines.

Regulatory Changes:

Assumption: Regulatory requirements and standards remain consistent during the project's development.

Impact: Changes in regulations may necessitate revisions to the system's features and functionalities.

3 System Features

For the Real-Time Food Recipe Making & Budgeting System to be a successful project, several features are required in the design. In this section, features and functional requirements will be described in detail. Below is a use-case diagram outlining the features available to the users of the system.

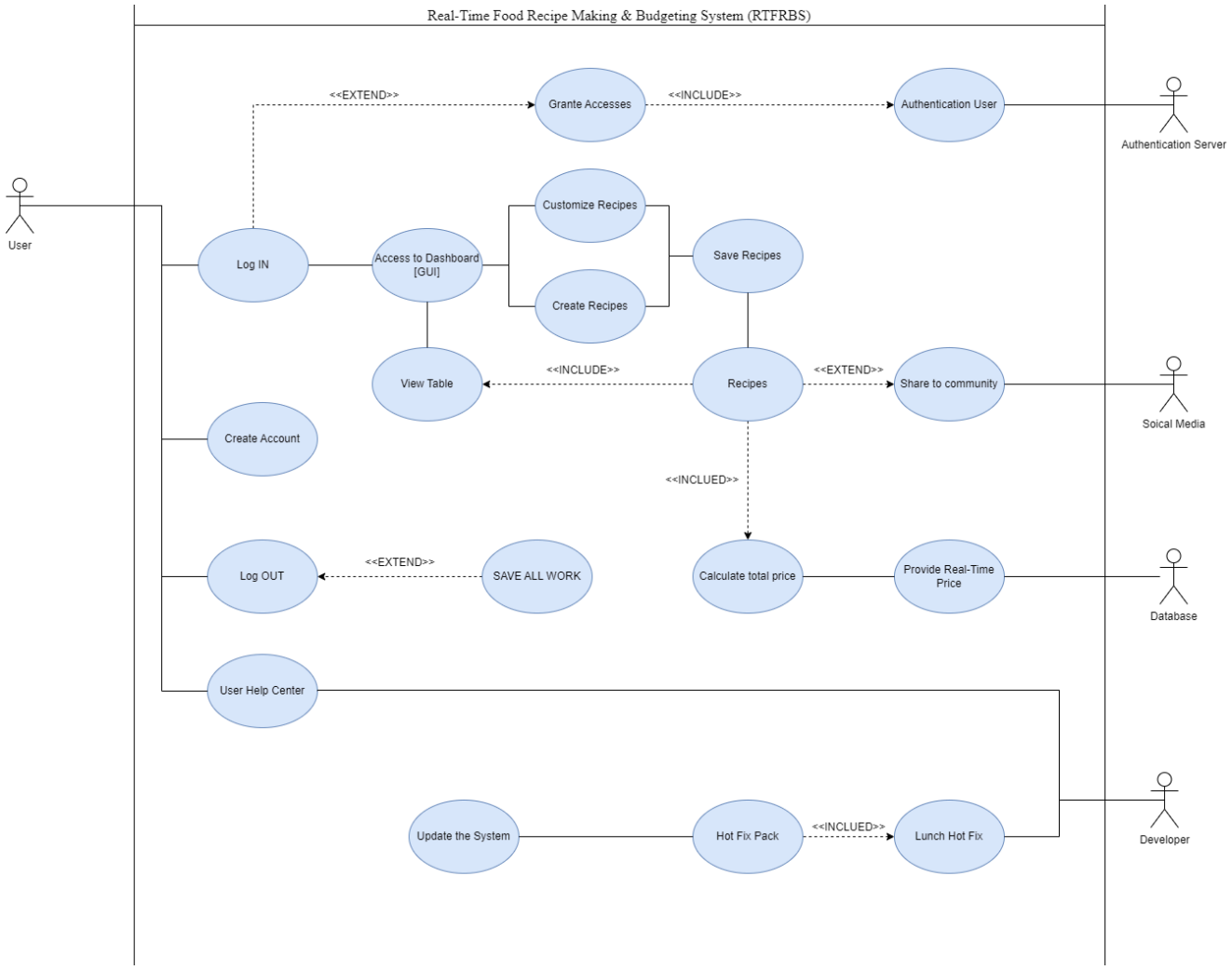

Figure 3.1 RTSHMS Use-Case Diagram

3.1 User Authentication

3.1.1 Description and Priority

Before a user can be granted access into the system, it is important that the system is able to determine the user's identity. User's will have credentials, a unique user id and password, which they can input into the system's login form. Upon successful authentication, the user will be granted access to features appropriate to their role. Given the level of security required by the Canadian navy, this feature has high priority.

3.1.2 Functional Requirements

REQ1-1: Each user's login user ID must be unique.
Rationale: The system must know who is currently using the terminal to determine access based on the user's role as well as log the user's activity when creating or customizing the recipes within the system.

REQ-1-2: The user must "sign-in" by submitting their credentials before being granted access.
Rationale: The user must be able to initiate authentication with their provided credentials, no functionality other than authentication shall be granted until successful authentication.

REQ-1-3: Upon authentication, the user shall be directed to the system dashboard(GUI).
Rationale: The dashboard is the main interface from which the user can navigate to other system features.

REQ-1-4: Users shall be able to contact their Developer by Leaving Message for issues regarding their credentials.
Rationale: If a user has forgotten their password, they need a method to change or retrieve it.

REQ-1-6: The terminal must “lock-out” user access for five minutes upon 5 consecutive failed login attempts.
Rationale: To protect the authentication process from brute-force attacks.

Logging In
ID: UC-1-1
Description: A user attempts to log into the system.
Actors: Maintenance User, RTFRBS, Develope
Secondary Use Cases: N/A
Preconditions: The user has login credentials and is situated at a terminal.
Main Flow: 1. The User enters their user id and password into the login form. 2. The login form sends these credentials to the authentication server. 3. The authentication server attempts to authenticate the user id and password. 4. Upon successful authentication the User is granted access privileges based on their role. 5. The User is directed to the system dashboard(GUI).
Postconditions: 1. The User has been granted access appropriate for their role and has access to the features displayed on the dashboard.
Alternative Flow(s): 1. If the user selects the forgotten password feature, a notification is sent to their System and it will send password reset mail to the user. 2. If the credentials are incorrect the user is notified and returns to step 1. 3. If the authentication has failed 5 times, the system will make a log and send notification to the Developer. 4. If the authentication has failed 5 times, the terminal locks users out for five minutes.

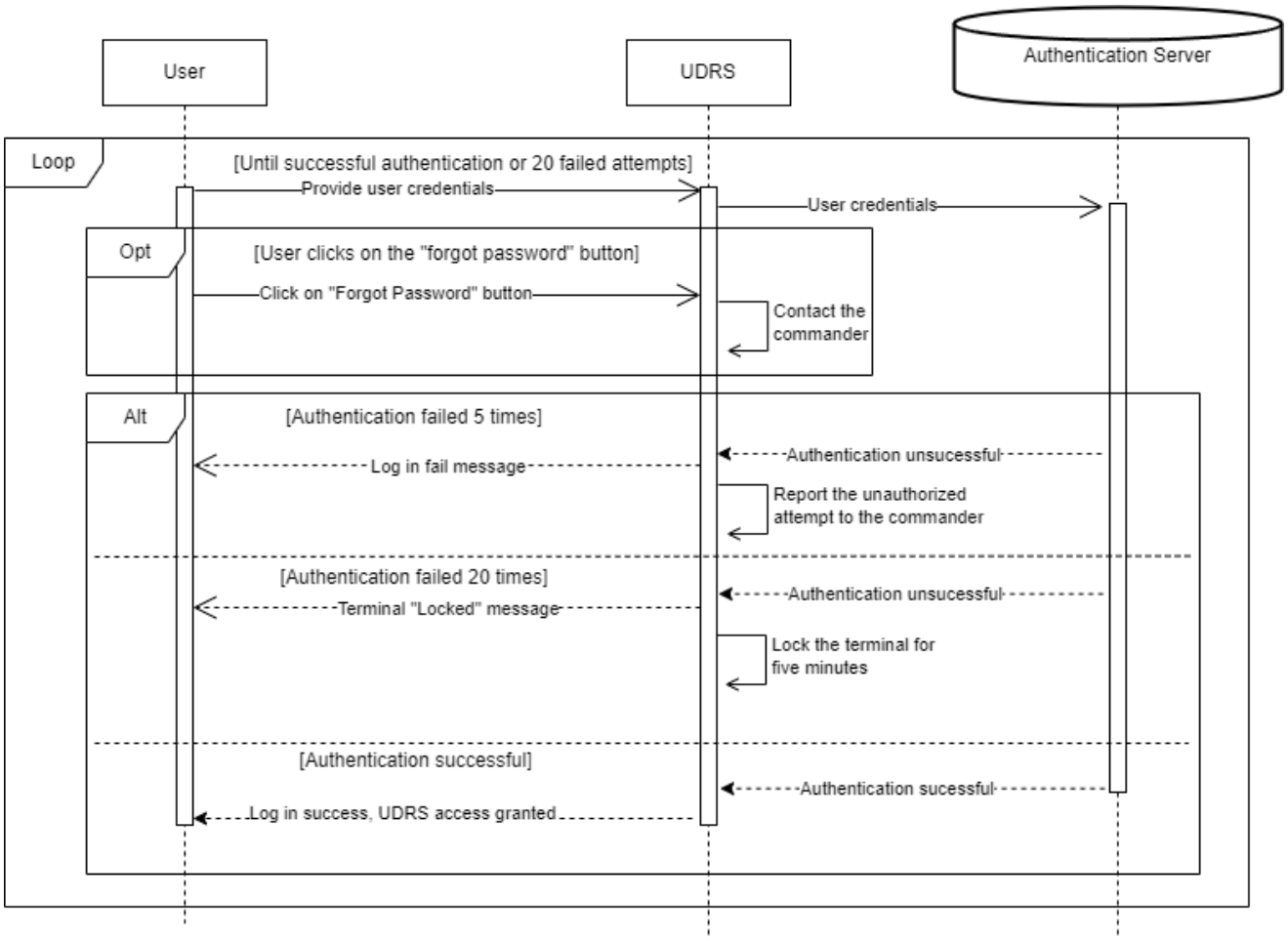

Figure 3.2 Logging In Sequence Diagram

Logging Out
ID: UC-1-2
Description: A user logs out of the system.
Actors: User, RTFRBS
Secondary Use Cases: Logging in
Preconditions: The user has already logged into the terminal.
Main Flow: 1. The User navigates back to the main dashboard. 2. The User clicks on the “Log Out” button. 3. The terminal relinquishes access privileges for that session and returns to the login page.
Postconditions: 1. The terminal relinquishes access and returns to the login page.
Alternative Flow(s): N/A

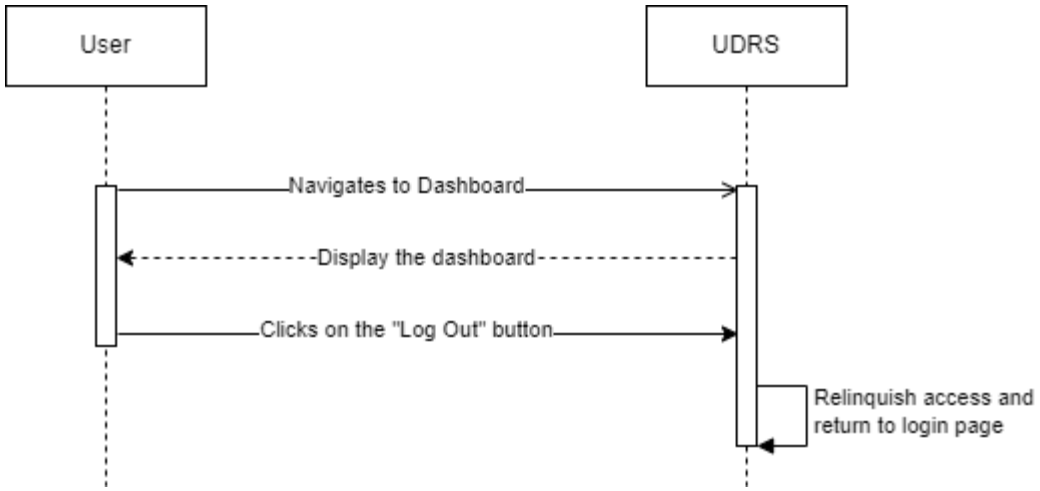

Figure 3.3 Logging Out Sequence Diagram

3.2 Recipe Create & Customize

3.2.1 Description and Priority

This is the main function of the whole system, the goal is to edit a Recipe. Via GUI, display and edit recipes in table form.

3.2.2 Functional Requirements

REQ2-1: When creating tables, each table should be able to select the type of recipe, and users can enter the share of each ingredient themselves.

REQ2-2: When Customizing the recipes, each table should be able to recalculate the share of ingredients based on the number of people that recipes share.

REQ2-3: Recipe files can be saved locally or online, and the system should have the ability to read the saved file and share it on social media.

REQ2-4: Also, the function has the ability to customize the price of each ingredient or quote prices from external websites. The total cost of the recipe can be calculated based on this.

3.2.3 Use Cases

Create the recipe
ID: UC-2-1
Description: The user customizes and creates the recipe
Actors: User, RTFRBS, Database, Social Media
Secondary Use Cases: N/A
Preconditions: The user must log in to the system.
Main Flow: 1. After the user logs in to the system, the system shows the recipes the user has created. 2. The user creates recipes and writes the number of the people that recipe can feed.

3. When the user selects ingredients, the system links to the database for the real-time price or the user can type a price for each ingredient. 4. After the user finishes the recipe, the system will show the total price according to the database. 5. The user saves the edited recipes. 6. The user can select if they want to share the recipes via Social Media.
Postconditions: N/A
Alternative Flow(s): 1. If there are no existing recipes. 2. If there exists a recipe with the same name , the system will ask the user if they want to replace the one that already existed.

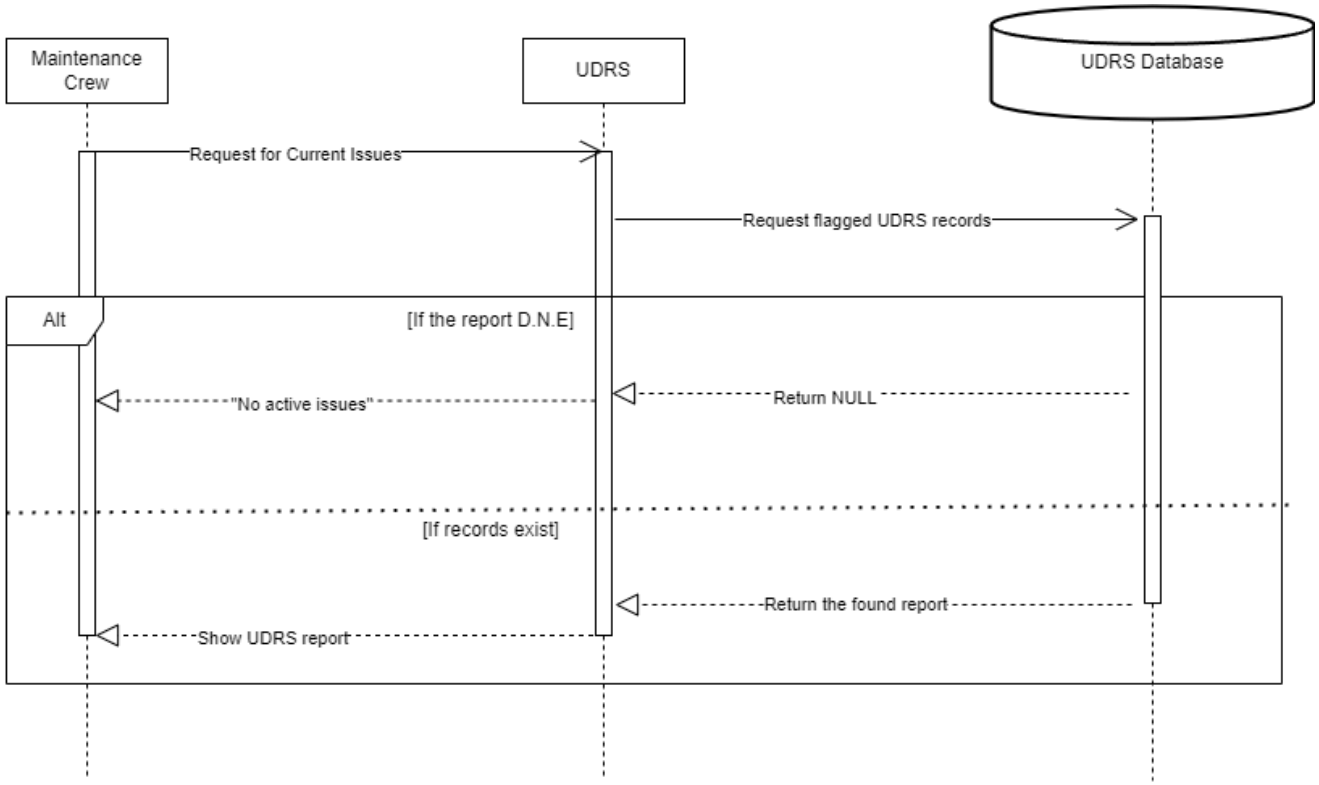

Figure 3.4 View Current Issues Sequence Diagram

Figure 3.1.2 View Current Issues Sequence Diagram

Customizing the recipes
ID: UC-2-2
Description: Edit the existing recipes
Actors: User, RTFRBS, Database, Social Media
Secondary Use Cases: Logging In
Preconditions: The user has logged into the system and has at least one created recipe.
Main Flow: 1. User selects an existing recipe 2. Users edit the ingredients or delete the ingredients. 3. The User enter the numbers of people that they want the recipe to feed

4. The system will recalculate the total price and the amount of the ingredients that the user needs. 5. The system updates and saves the recipes. 6. The user can select if they want to share the recipes via Social Media.
Postconditions: N/A
Alternative Flow(s): 1. a. If there are no records it will return an error message, UDRS system will keep updating every minute.

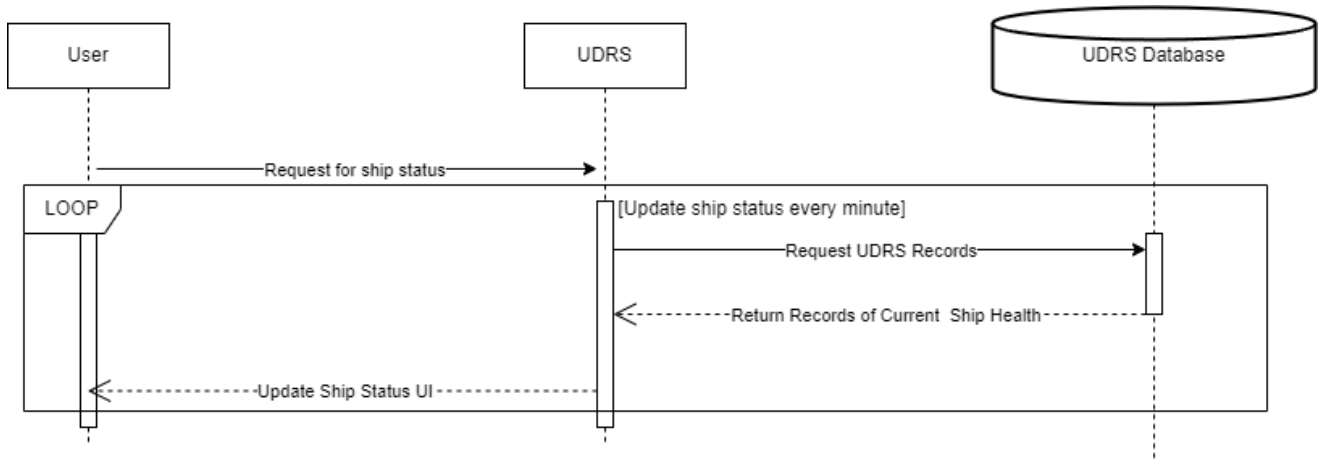

Figure 3.5 View Ship Status Sequence Diagram

Figure 3.1.3 View Ship Status Sequence Diagram

4 External Requirements

Note: Choose those that apply to your project

4.1 User Interfaces

Operator Interface:

User Type: Housewife or someone who needs to cook.

Sample Screen: The operator interface may include a dashboard displaying all the created recipes and the option to create or share the recipes.

GUI Standards: Follow intuitive and user-friendly graphical user interface (GUI) standards with a clear layout.

Screen Layout: Emphasize a clean, uncluttered and customizable layout with easily readable data.

Standard Buttons/Functions: Include a list of created recipes, the button to create the new recipes, also the button to share the recipes.

Keyboard Shortcuts: Incorporate keyboard shortcuts for efficient access to different health status views.

Error Messages: Clear and user-friendly error messages that help users take appropriate actions in case of system issues.

Maintenance Personnel Interface:

User Type: Maintenance and repair professionals responsible for implementing scheduled maintenance and addressing anomalies.

Sample Screen: The interface must provide a maintenance calendar, task lists, and real-time diagnostics.

GUI Standards: A well-organized, task-centric GUI with efficient navigation.

Screen Layout: Prioritize task lists, maintenance instructions, and diagnostics tools.

Standard Buttons/Functions: Include task creation and tracking, access to equipment manuals, and alerts.

Keyboard Shortcuts: Incorporate keyboard shortcuts for efficient task management and data entry.

Error Messages: Detailed error messages that guide maintenance personnel in diagnosing and resolving issues.

Common Characteristics:

Help Function: All interfaces should provide access to a help function or documentation to assist users.

Consistent Design: Maintain Systems' consistent look and feel and colour scheme across interfaces to create a unified user experience.

Security: Ensure strong security features, including user authentication, access control, and data encryption, to protect sensitive ship data.

4.2 Software Interfaces

Operating System:

Name and Version: RTFRBS is compatible with Windows 10+ and Mac OS systems.

Purpose: The operating system provides the platform on which RTFRBS runs, handling low-level system tasks, resource management, and hardware interaction.

Database Management System (DBMS):

Name and Version: Examples include MySQL, PostgreSQL, or Microsoft SQL Server.

Purpose: RTFRBS relies on a DBMS to store and manage data related to ship health, maintenance schedules, and historical records.

External Communication Systems:

Tools and Protocols: Internet protocols (e.g., HTTP, HTTPS), email servers, and communication protocols.

Purpose: RTFRBS may need to communicate with external systems for remote monitoring, alert notifications, and data exchange with onshore maintenance and command centers.

User Interfaces:

Software Components: These include the operator interface, and maintenance personnel interface, within RTFRBS.

Purpose: These interfaces allow different users to interact with RTFRBS for monitoring, maintenance, and

system configuration.

API and Data Exchange with External Systems:

The system provides APIs (Application Programming Interfaces) to facilitate data exchange with external systems, such as enterprise resource planning (ERP) systems, allowing for the integration of maintenance data with broader organizational processes.

5 Other Non Functional Requirements

5.1 Performance Requirements

1. Monitoring has to be in real-time, with a maximum delay of 1 second. This will help to ensure immediate data availability for users using the system.
2. It is crucial for the system to have a minimum uptime of 99.99%, to maintain the continuity of data for the crew.
3. Given the clients need of having multiple user groups work with it with their own needs highlighted in the system, the system needs to have multi-user support.
4. The system will need to make maintenance predictions and recommendations, therefore the data processing must be efficient.

5.2 Software Quality Attributes

1. The system must operate with at least 99.99% availability to continuously monitor the operational parameters of the ship. This only allows for 52 minutes of downtime per year.
2. The system should be able to accommodate additional sensors and updates in the future without significant changes or disruptions.
3. Target a user error rate below 5% and aim for at least 90% positive user feedback regarding UI/UX in bi-annual surveys.
4. Ensure the system can be easily modified to cater to evolving needs and technologies, while maintaining a standard where system updates or maintenance do not result in more than 0.01% of total annual downtime.

6 Other Requirements

6.1 Training

1. **Training Programs:** Develop and deliver training programs for different user roles.
2. **Knowledge Base:** Create a knowledge base or forum for users to share experiences and solutions.
3. **User Manuals and Documentation:** Provide comprehensive user manuals that adhere to technical documentation standards, ensuring clarity and ease of understanding for end-users.

6.2 Testing

1. **Unit Testing:** Verify the functionality of individual components, such as data retrieval from a single sensor or the generation of a maintenance ticket.
2. **Integration Testing:** Ensure that the UDRS effectively communicates with the sensor arrays and Maintenance Ticket Generator.
3. **System Testing:** Validate the end-to-end functionality of the entire Real-Time Ship Health and Maintenance System.

6.3 Support and Maintenance

1. **Support Desk:** Establish a support desk to assist users and resolve issues.
2. **Maintenance Window:** Define regular maintenance windows for performing updates and preventive actions.
3. **Notification System:** Notify users of upcoming maintenance or system changes.

6.4 Audit and accountability

1. **Audit Trails:** Implement detailed audit trails capturing user actions, data modifications, and access.

2. **Review Process:** Establish a process for periodic review of audit logs.
3. **Alerts:** Implement alerts for anomalous or unauthorized activities detected in audit logs.

6.5 Interoperability and Integration

1. **Middleware:** Use middleware solutions to enable communication among heterogeneous systems.
2. **Adapters:** Develop adapters to translate data and commands between systems with different communication protocols.
3. **Third-Party System Integration:** Ensure compatibility and integration capability with external or third-party systems, such as port management systems, logistics systems, or maritime traffic monitoring systems.
4. **Remote Accessibility:** Enable remote access and control of the system, ensuring secure and efficient interoperability with onshore facilities and external entities.

6.6 Compliance and Standards

1. **Transport Canada:** Adhere to regulations set by Transport Canada regarding ship operations, safety, and maintenance.
2. **International Maritime Regulations:** Adhere to laws and regulations set by the International Maritime Organization.
3. **PIPEDA:** Ensure compliance with the Personal Information Protection and Electronic Documents Act (PIPEDA) for handling and protecting personal information.
4. **Environmental Regulations:** Comply with Canadian environmental laws regarding emissions, waste management, and energy usage in maritime operations.

7 Entity Relationship Diagram

7.1 Entity Relationship Diagram

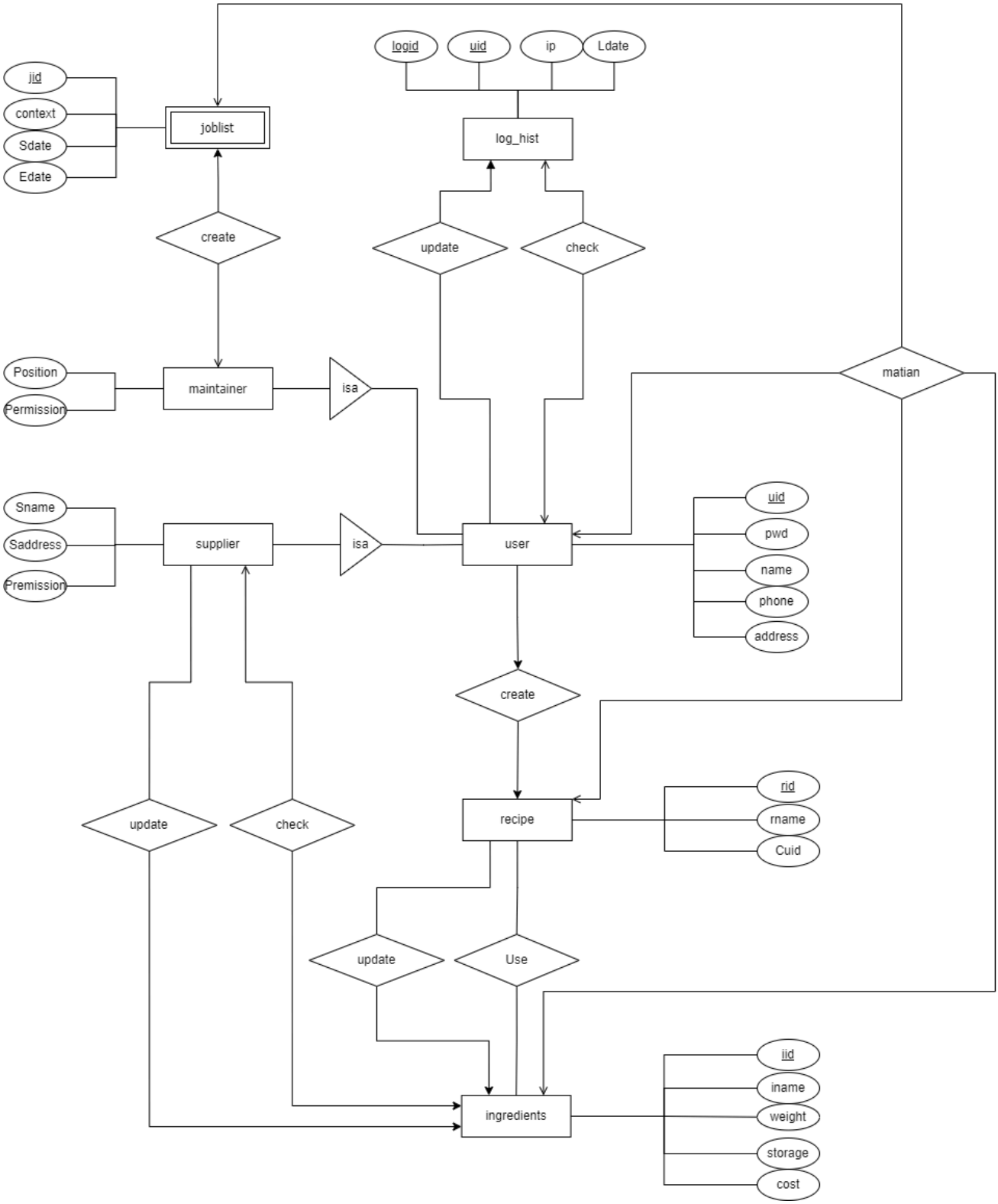

Figure 7.1 Entity Relationship Diagram

7.2 Database Schema

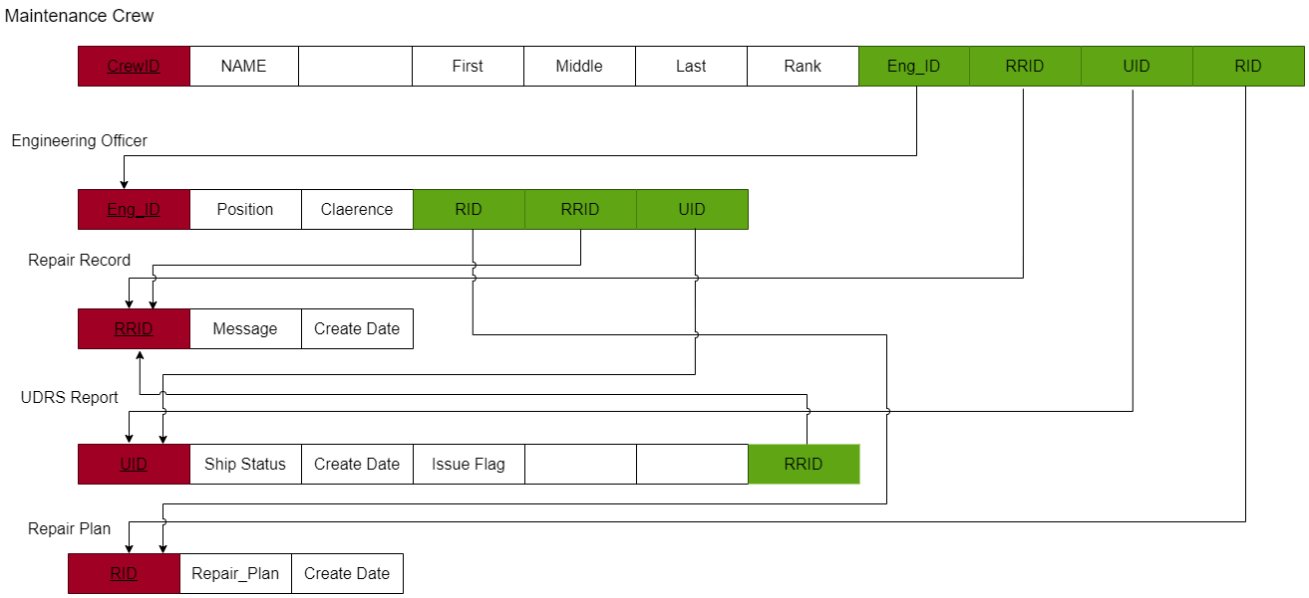

Figure 7.2 ERD Database Schema

7.3 Data Dictionary

Table 7.1 Data Dictionary

Object Class	Attributes
User	uid(int), pwd(int), name(varchar), address(varchar), phone(int)
Maintainer	uid(int), mid(int), Position(varchar), Permission(varchar)
Supplier	uid(int), sid(int), sname(varchar), address(varchar), Permission(varchar)
Log_hist	logid(int), uid(int), ip(int), Ldate(DATETIME)
Joblist	mid(int), jid(jid), context(varchar), Sdate(DATETIME), Edate(DATETIME)
Recipe	rid(int), rmane(varchar), Cuid(int)
Ingredients	iid(int), iname(varchar), weight(float), storage(float), cost(float)

7.4 Unique Keys

Table 7.2 Unique Keys Table

Type	Table	Column
Primary	User	uid
Primary	Log_hist	uid

Primary	Joblist	jid
Primary	Recipe	rid
Primary	Ingredients	iid

7.5 Foreign Keys

Table 7.3 Foreign Keys Table

Table	Column	Referenced Table	Referenced Column
Log_hist	uid	user	uid

8 Prototype User Interface

Appendix: Issues List

No issues for this iteration.