**Yaşar University**

**Spring, 2023 - 2024**

**SE 2224 - Software System Analysis**

**Final Project Report: Software Requirements Specifications Document (SRS)**

|  |  |
| --- | --- |
| **Student Name:** | **Can Berk Soydan** |
| **Student No:** | **21070006034** |
| **Department Name:** | **Software Eng** |
| **Course Section No:** | **SE 2224-YAZILIM SİSTEMI ANALİZİ\_SB:1** |

This template is prepared based on the IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998).

**Table of Contents (Do not change the Section Names!)**

[**1** **Introduction** 3](#_Toc165879141)

[**1.1** **Purpose** 3](#_Toc165879142)

[**1.2** **Scope** 3](#_Toc165879143)

[**1.3** **Definitions, acronyms, and abbreviations** 3](#_Toc165879144)

[**1.4** **References** 3](#_Toc165879145)

[**1.5** **Overview** 3](#_Toc165879146)

[**2** **Design and Implementation Constraints** 3](#_Toc165879147)

[**3** **Specific Requirements** 3](#_Toc165879148)

[**3.1** **Functional Requirements** 4](#_Toc165879149)

[**3.2** **Performance Requirements** 4](#_Toc165879150)

[**3.3** **Software System Attributes** 4](#_Toc165879151)

[**3.4** **Use Case Analysis** 4](#_Toc165879152)

[**3.4.1** **Actors** 4](#_Toc165879153)

[**3.4.2** **Scenarios** 4](#_Toc165879154)

[**3.4.3** **Use Case Forms** 4](#_Toc165879155)

[**3.4.4** **Relationships among Actors and Use Cases** 5](#_Toc165879156)

[**3.4.5** **Use Case Diagram** 5](#_Toc165879157)

[**4** **Behavioral Models** 5](#_Toc165879158)

[**4.1** **Sequence Diagram** 5](#_Toc165879159)

[**5** **Structural Models** 5](#_Toc165879160)

[**5.1** **Class Diagram** 5](#_Toc165879161)

[**6** **Process Modeling** 5](#_Toc165879162)

[**6.1** **Data Flow Diagram (DFD)** 5](#_Toc165879163)

[**7** **Graphical User Interface(s) (GUIs)** 5](#_Toc165879164)

[**8** **Conclusion and Future Work** 5](#_Toc165879165)

# **Introduction**

No explanation is needed here. Only complete the subsections.

## **Purpose**

The purpose of this SRS document is to provide a detailed description of the "FavoriteSites" desktop application. This document outlines the functionalities, system requirements, design constraints, and user interactions that will guide the development and implementation of the software.

## **Scope**

a) Identify the software product to be produced by name.

* The software product to be produced is called "FavoriteSites".

b) Explain what the software product will do.

* "FavoriteSites" is a desktop application designed to help users keep a detailed record of the locations they have visited and liked. The application will provide functionalities for users to log in, add new visits, edit existing visits, delete visits, share visit information with friends, and view shared visits.

c) Describe goal of the application of the software being specified.

* The main goal of this application is to allow users to easily record their travel experiences in detail and share this recorded data with other friends.

## **Definitions, acronyms, and abbreviations**

Definitions

* Actor: An entity that interacts with the system. In this context, actors include Users, Friends, and System Administrators.
* Visit: A record of a user's travel experience, including details such as country name, city name, year visited, season visited, best feature, comments, and rating.
* GUI (Graphical User Interface)\*\*: The visual component of the application through which users interact with the software.
* JFrame: A class in Java Swing that represents a window with decorations such as a title, border, and buttons for closing, iconifying, and maximizing the window.
* JTextField: A class in Java Swing that allows users to enter and edit a single line of text.
* MySQL: An open-source relational database management system used to store and retrieve data for the application.
* Trigger: An event that initiates a use case.
* Precondition: A condition that must be true before a use case can be initiated.
* Postcondition: A condition that must be true after a use case has been completed.

Acronyms

* SRS: Software Requirements Specification
* GUI: Graphical User Interface
* IDE: Integrated Development Environment
* SQL: Structured Query Language
* -UML: Unified Modeling Language, used for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Abbreviations

* -ID: Identifier, a unique value used to identify an item in the database.

## **References**

This subsection should provide a complete list of all documents referenced elsewhere in the SRS document.

## **Overview**

This subsection should describe what the rest of the SRS contains and how the SRS is organized.

# **Design and Implementation Constraints**

Describe any items or issues that will limit the options available to the developers. These might include: hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; language requirements.

Things that can be mentioned under this section: NetBeans/IntelliJ including its version, MySql including its version, Java as a programming language, Visual Paradigm as a modeling tool.

# **Specific Requirements**

No explanation is needed here. Only complete the subsections.

## **Functional Requirements**

**1. Login and Authentication:**

* The system shall display a login screen (LoginFrame) for user authentication.
* The system shall allow the user to enter a username and password to access the application.
* The system shall validate the entered credentials against the `userinfo` database table.
* The system shall display an error message if the login fails.
* The system shall display the main screen (MainFrame) upon successful login and hide the LoginFrame.

**2. Adding a Location:**

* The system shall provide input fields (JTextField components) for the user to enter the Country Name, City Name, Year, Season, Best Feature, Comments, and Rating.
* The system shall automatically generate and assign a unique visit ID to each new visit.
* The system shall insert the visit data, including the username, into the `visits` database table.
* The system shall display a confirmation message upon successful addition of a visit.

**3. Deleting a Visit:**

* The system shall provide an input field for the user to enter the visit ID to delete.
* The system shall delete the corresponding visit entry from the `visits` database table.
* The system shall display a message dialog box confirming the deletion of the visit.

**4. Display, Edit, and Update a Visit:**

* The system shall allow the user to enter a visit ID to display the visit information.
* The system shall retrieve and display the visit information from the `visits` database table.
* The system shall allow the user to edit the displayed visit information.
* The system shall update the `visits` database table with the edited information.
* The system shall display a confirmation message upon successful update of the visit.

**5. Display Informations:**

* The system shall retrieve and display country names whose best feature is food, sorted by their rating in descending order.
* The system shall provide an input field for the user to enter a visit ID.
* The system shall display the corresponding image file (Location1.jpg, Location2.jpg, etc.) from the project folder.
* The system shall provide an input field for the user to enter a year.
* The system shall retrieve and display all visits made in the specified year from the `visits` database table.
* The system shall retrieve and display the country name(s) visited the most by the user.
* The system shall retrieve and display the country name(s) that were only visited in the spring season by the user.

**6. Share Visit with a Friend:**

* The system shall provide input fields for the user to enter their friend’s username and the visit ID to share.
* The system shall insert the sharing information into the `sharedvisits` database table.

**7. Display Shared Visits:**

* The system shall retrieve and display visit information that has been shared with the user.
* The system shall match the user’s username in the `sharedvisits` database table and retrieve the corresponding friend’s username and visit ID.
* The system shall display the visit information (country name, city name, season visited, best feature) for all shared visits.

## **Performance Requirements**

* The system shall support up to 100 simultaneous users accessing the application without significant performance degradation.
* The system shall provide a response time of no more than 2 seconds for any database query, including login authentication, adding a location, deleting a visit, and retrieving visit information.
* The system shall handle up to 10,000 visit entries in the `visits` database table and 5,000 entries in the `sharedvisits` database table without affecting the performance or response time of the application.

## **Software System Attributes**

**1. Reliability:**

* The system shall have a 99.9% uptime over a one-year period, ensuring that it is consistently available for users to access and interact with their visit data and shared locations.

**2. Availability:**

* The system shall ensure that critical functionalities (such as login, adding a location, and viewing shared visits) are available 24/7, with scheduled maintenance windows not exceeding 2 hours per month.

**3. Security:**

* The system shall maintain detailed logs of all user activities, including login attempts, data modifications, and sharing actions.

**4. Maintainability:**

* The system shall be designed with modularity in mind, such that any module (e.g., login module, visit management module) can be updated or replaced independently, with a mean time to repair (MTTR) not exceeding 4 hours.

**5. Usability:**

* The system shall provide an intuitive user interface, with all major functionalities (e.g., login, adding a visit, sharing a visit) accessible within three clicks from the main menu, ensuring ease of use for users with minimal training.

## **Use Case Analysis**

No explanation is needed here. Only complete the subsections.

### **Actors**

**User:** A person who logs into the system to manage their favorite sites, add new visits, delete existing visits, share visits, and view shared visits. The user has access to all functionalities after logging in.

**Friend:** A person who receives shared visit information from other users. The friend can view the visits shared with them.

### **Scenarios**

**Scenario 1: Adding New Visit:**

1. The user logs into the application.
2. The user navigates to the section for adding a new visit.
3. The user fills in the required information (Country Name, City Name, Year, Season, Best Feature, Comments, Rating).
4. The user click the add button and saves the visit. The visit ID is automatically generated.

**Scenario 2: Sharing a Visit with a Friend:**

1. The user logs into the application
2. The user navigates to the section for sharing visit with friends.
3. The user selects a visit they want to share and enters their friend's username.
4. The user click the share button and visit information is saved in the shared visit database.

### **Use Case Forms**

**1. Use Case: Add a New Visit**

* **Participating Actors:** User
* **Description:** The user adds a new visit to the database.
* **Trigger:** The user decides to log a new visit.
* **Preconditions:** The user must be logged into the application.
* **Normal Course (Flow of Events):**

1. The user navigates to the 'Add Visit' section.
2. The user fills out the visit details in the provided text fields.
3. The user clicks the `Add New Location’ button.
4. The system generates a new visit ID and stores the visit information in the database.
5. A confirmation message is displayed to the user.

* **Postconditions:** The new visit is saved in the database with a unique visit ID.
* **Exceptions:** If any required field is empty, the system displays an error message and prompts the user to fill in the missing information.

**2. Use Case: Share Visit With a Friend**

* **Participating Actors:** User, Friend
* **Description:** The user shares a visit with a friend.
* **Trigger:** The user decides to share visit.
* **Preconditions:** The user must be logged into the application and have at least one visit logged.
* **Normal Course (Flow of Events):**

1. The user navigates to the ‘Share With Friends’ section.
2. The user enters the visit ID and friend’s username.
3. The user clicks the `Share With Friends’ button.
4. The system saves the shared visit information in the sharedvisits database.
5. A confirmation message is displayed to the user.

* **Postconditions:** The shared visit information is stored in the databse.
* **Exceptions:** If the friend’s username or visit ID are invalid or does not exist, the system displays an error message.

**3. Use Case: Display Image of a Visit**

* **Participating Actors:** User
* **Description:** The user views an image of a location associated with a visit. The user enters the visit ID, and the system retrieves and displays the corresponding image.
* **Trigger:** The user wants to see an image of a specific visit by entering the visit ID.
* **Preconditions:** The user must be logged into the application and there must be visits logged with associated images.
* **Normal Course (Flow of Events):**

1. The user navigates to the ‘Display Image’ section.
2. The user enters the visit ID.
3. The user clicks the `Display Image’ button.
4. The system locates the image file associated with the visit ID.
5. The system displays the image in the GUI.

* **Postconditions:** The image corresponding to the specified visit ID is displayed to the user.
* **Exceptions:** If there is no image associated with the visit ID, the system displays a question mark image.

**1. Use Case: Log In**

* **Participating Actors:** User
* **Description:** The user logs into the system using their username and password.
* **Trigger:** The user wants to access the application's functionalities.
* **Preconditions:** The user's information must be registered in the database.

**2. Use Case: Delete Visit**

* **Participating Actors:** User
* **Description:** The user deletes a visit from the databse.
* **Trigger:** The user decides to remove visit.
* **Preconditions:** The user's must be logged in the application and have at least one visit logged.

**3. Use Case: View Visits by Year**

* **Participating Actors:** User
* **Description:** The user views all visits made in a spesific year.
* **Trigger:** The user wants to see visits from a spesific year.
* **Preconditions:** The user's must be logged in the application and have at least one visit logged in the spesific year.

**4. Use Case: View Visits by Spring Season**

* **Participating Actors:** User
* **Description:** The user views visits made only in spring as a list on the GUI.
* **Trigger:** Login to the system..
* **Preconditions:** The user's must be logged in the application and have at least one visit logged in the spring season.

**5. Use Case: View Visits by Food Best Feature**

* **Participating Actors:** User
* **Description:** The user views country names whose best feature is food, sorted by rating as a list on the GUI.
* **Trigger:** Login to the system..
* **Preconditions**: The user must be logged into the application and have visits logged with food as the best feature.

**6. Use Case: View Country Visited the Most**

* **Participating Actors:** User
* **Description:** The user views the country they have visited the most as a list on the GUI.
* **Trigger:** Login to the system.
* **Preconditions:** The user must be logged into the application and have multiple visits logged.

**7. Use Case: View Shared Visits**

* **Participating Actors:** User
* **Description:** The user views visits shared with them by friends.
* **Trigger:** The user wants to see visits shared with them.
* **Preconditions:** The user must be logged into the application and have visits shared with them.

**8. Use Case: Edit Visit Table**

* **Participating Actors:** User
* **Description:** The user edits the visit table and update on database.
* **Trigger:** The user wants to update spesific visit data.
* **Preconditions:** The user's must be logged in the application and have at least one visit logged in.

### **Relationships among Actors and Use Cases**

User - Add a New Visit: The user interacts with the system to log a new visit, entering the necessary details and saving them to the database.

User - Share Visit with a Friend: The user selects and shares a visit with a friend, requiring interaction with the sharedvisits table.

User - Delete Visit: The user removes a visit from the database.

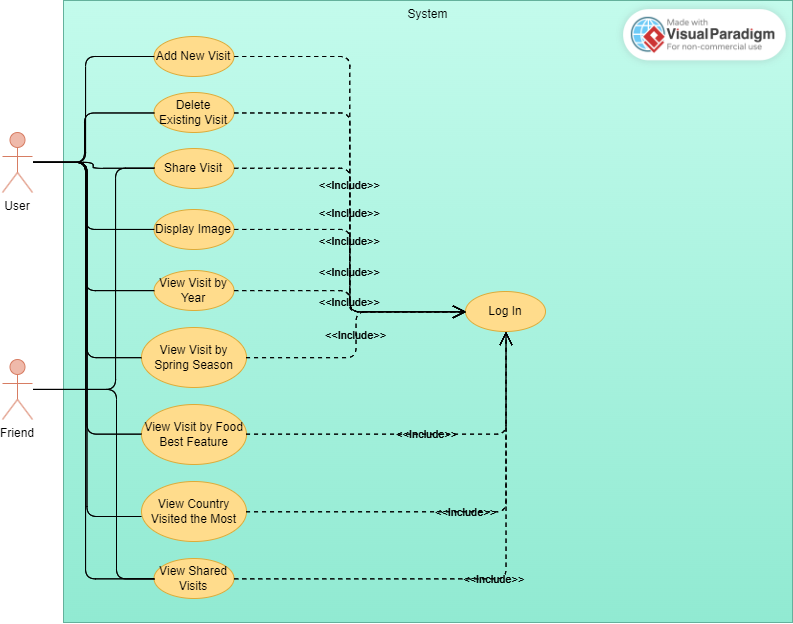
User – Edit Visit: The user edit a visit data from the table.

User - View Visits by Year: The user retrieves and views visits based on the year they were made.

User - View Shared Visits: The user views visit information shared by friends.

Friend - View Shared Visits: The friend receives and views visit information shared by another user.

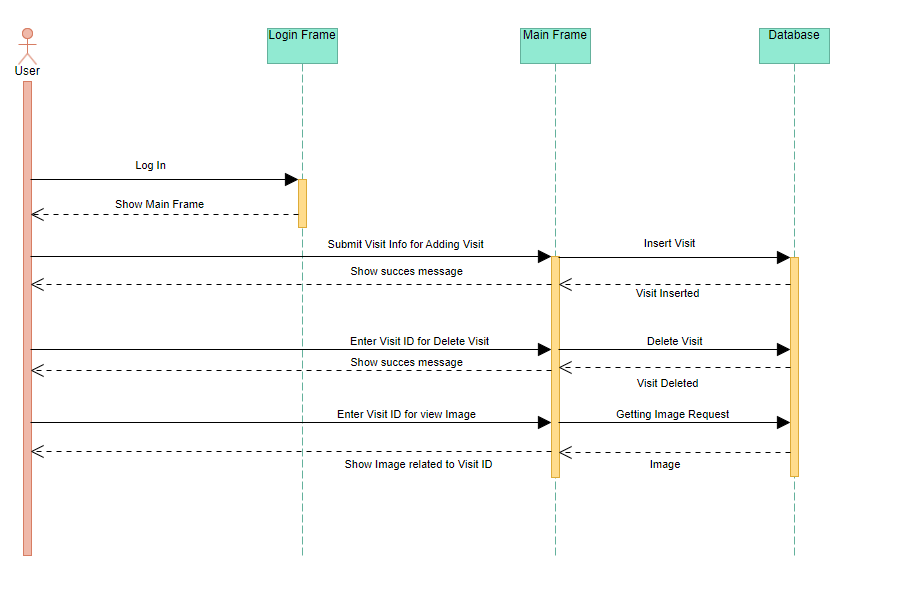
### **Use Case Diagram**



# **Behavioral Models**

No explanation is needed here. Only complete the subsection.

## **Sequence Diagram**



**1 – Log In :** The user enters the login information, the login frame shows the main frame to the user in line with the information.

**2 – Add New Visit:** The user adds a new visit. MainFrame verifies this information and adds it to the database. The user is informed that he/she is successful.

**3 – Delete Visit**: The user enters the visit ID they want to delete. MainFrame verifies this ID and deletes it from the database. The user is informed that he/she is successful.

**4 – Show Image by Visit ID:** The user enters the visit ID and requests to see the image. MainFrame retrieves the relevant visit information and image according to the ID in the database and displays it to the user.

# **Structural Models**

No explanation is needed here. Only complete the subsection.

## **Class Diagram**



Database Manager: Database Manager is a class created to perform different operations on data and tables in MySQL.

Demo: Demo class is the class that contains all GUI elements and fills the necessary lists and tables in the GUI through the database manager.

Login Form: Login form is the class that transports registered users in the database to Demo.

Database Singleton: Database Singleton is a global class that holds the data that connects MySQL and java.

# **Process Modeling**

No explanation is needed here. Only complete the subsection.

## **Data Flow Diagram (DFD)**

Put the images and the explanations of your Context Diagram and Level 0 DFD.

# **Graphical User Interface(s) (GUIs)**

Put the images of your GUIs, also put the output screenshots of each function that is tested with sample data. Briefly explain the purpose of each GUI. Do not forget to add the form title for each image.

# **Conclusion and Future Work**

Write a conclusion for your project and summarize your work. Also write what kind of features/functions can be added to your project as a future work.