**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**

This subsection should write the purpose of the SRS document.

## **Scope**

This subsection should

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

b) Explain what the software product will do.

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

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

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS document.

## **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**

Write and describe each actor that will interact with your system.

### **Scenarios**

Write 2 sample use case scenarios that can occur in your software application.

### **Use Case Forms**

Prepare the **complete use case forms** for 3 use cases that you will select. For the rest of your use cases, prepare only a **basic use case form**.

**Complete use case form** includes the following sections: Use case name, Participating actors, Description, Trigger, Preconditions, Normal course (Flow of events), Post conditions, and Exceptions.

**Basic use case form** includes the following sections: Use case name, Participating actors, Description, Trigger, Preconditions.

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

Explain the relationship between each actor and the use case in your use case diagram.

### **Use Case Diagram**

Put the image and explanation of your use case diagram.

# **Behavioral Models**

No explanation is needed here. Only complete the subsection.

## **Sequence Diagram**

Draw the sequence diagram of 3 use cases that you will select. Put their images and explanations in this subsection.

# **Structural Models**

No explanation is needed here. Only complete the subsection.

## **Class Diagram**

Put the image and explanation of your class diagram.

# **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.