Yaşar University

Spring, 2023 - 2024

SE 2224 - Software System Analysis

Final Project Report: Software Requirements Specifications Document (SRS)

Student Name:	**Defne Tekyiğit**
Student No:	**22070006021**
Department Name:	**Software Engineering**
Course Section No:	**Section-2**

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

1.1 Purpose

The purpose of this SRS document is to present a detailed description of the Favorite Sites Application. The SRS document will detail the objectives and capabilities of the Favorite Sites system. It will describe the system's functionalities, interfaces, and operational constraints. Additionally, it will outline how the system responds to external inputs and conditions.

1.2 Scope

Favorite Sites the software product goals to achieve an application that users can add their visits and their information to the system. The system will provide basic functionalities such as add, edit, delete for users. And some helpful functionalities such as displaying the most visited country and the countries with the best feature food. It will also allow users to share their favorite visits with their friends and view what visits have been shared with them.

1.3 Definitions, acronyms, and abbreviations

SRS: Software Requirements Specifications

GUI: Graphical User Interface

1.4 References

IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)

1.5 Overview

The next chapter, Specific Requirements section of this document describes in technical terms the details of the functionality of the product.

The fourth, fifth and sixth chapters include models of the system.

The seventh chapter, Graphical User Interface(s) (GUIs), shows the visual design of the system's user interfaces. This section includes screenshots of the GUIs, along with explanations of their purposes and functionalities.

The eighth and final chapter, Conclusion and Future Work, summarizes the work completed in this project and outlines potential enhancements and additional features that could be implemented in future iterations of the software.

2 Design and Implementation Constraints

IntelliJ IDEA: Version IU-223.8617.56

MySQL: Version 8.0.36

Java: Version 17.0.5+1-b653.25

Visual Paradigm: Version 17.1 (Build 20231001)

3 Specific Requirements

3.1 Functional Requirements

- The system shall allow users to login with username and password.
- The system shall authenticate users with their username and password from the database.
- The system shall allow users to add new visits with entering country name, city name, year visited, season visited, best feature, comment and rating of the visits.
- The system shall display all users visits on a table interface where user can see visitID and visit information.
- The system shall allow users to see and edit the details of an existing visit by entering visitID.
- The system shall allow users to delete their visits by entering visitID.
- The system shall display visits where the best feature is specified as food when the “Best Food” button is selected.
- The system shall display an informative message about the most visited country when the “Most Visited Country” button is selected.
- The system shall display visits that are visited only in spring when the “Spring Visits” button is selected.
- The system shall allow users to filter visits by entering the year visited and display the filtered results on the table.
- The system shall display an image of the location specified by the user.
- The system shall allow users to share visit details with a friend by entering the friend's username and the visitID.
- The system shall retrieve, and display visits shared with the selected user on a table interface when users selects “Shared with me” button.
- The system shall provide informative error messages to users in case of invalid inputs during visit addition, editing, or deletion.

3.2 Performance Requirements

1. The system shall respond to user interactions within 2 seconds under normal operating conditions.
2. The system shall be capable of storing up to 10,000 visit records in the database.
3. The system shall support a maximum of 100 terminals concurrently accessing the application.

3.3 Software System Attributes

- The system shall detect and handle errors by sending informative messages at all times. (reliability)
- The system shall be available 7/24 unless there is maintenance. (availability)
- The system shall implement user-specific access control, ensuring that each user can only access and modify their own visits, to prevent unauthorized access or modification to another user's visit data. (security)
- The system shall have regular updates, scheduled every three weeks, to ensure the incorporation of new features and bug fixes. (maintainability)
- The system shall prioritize user-friendliness by providing clear labeling of features and

3.4 Use Case Analysis

3.4.1 Actors

User: interacts with the system by adding new visits, editing and deleting existing visits and sharing visits with their friends, they have access to all the functionalities for their visits.

Friend: represents other users which the visits are being shared.

System: manages system-level tasks such as user authentication and database management.

3.4.2 Scenarios

Scenario name: viewSharedVisit

Participating actor instances: Defne: user , Lara: friend, System

Flow of events:

1. Lara, going through her visits she realizes she wants to share her visit to Japan with her friend defne and she enters the visitID of the visit and her friend Defne's username to the system and clicks "share visit with friend" button.
2. The system adds the visitID and Defne's username to the sharedVisits table along with the Lara's username and then displays a confirmation message to the Lara.
3. Lara receives the message and acknowledges that the visit has been shared.
4. Later, Defne logs into the system and she first checks if someone shared a visit with her by clicking "shared with me" button.
5. The system retrieves the shared visits info from the sharedVisits table and displays it on the table interface within the system.
6. Defne sees that Lara has shared the Japan visit with her and she can view the details of theshared visit.

Scenario name: editVisit

Participating actor instances: Defne: user, System

Flow of events:

1. Defne while reviewing her visits she realizes that her visit to Belgrade has wrong information about the year visited, she clicks the "Edit Visit" button and after that she enters the visitID of the Belgrade visit to see the information at the edit form.
2. The system after defne enters the VisitID fro the display selects the entered id and finds the information about the visit and prompts it to the edit form.
3. Defne sees the visits information on the form and she selects the year visited and changes it to the correct year after that she clicks to the "Edit" button.
4. The system updates the visits table at the database with the edited information and updates the table interface and then displays a confirmation message to Defne .
5. Defne sees the message and can view the edited version of the visit on the table interface.

3.4.3 Use Case Forms

1. **Use case name:** Login
   Participating actors: User, The system
   Description: This use case describes how a user logs into the application.
   Trigger: User decides to log in to the application.
   Preconditions:
   - The application is available and on-line.
   - User_info database is available and on-line.

   Normal Course:
   1. User opens the application.
   2. User enters their username and password.
   3. User clicks login button.
   4. System checks user's credentials from the user_info database.
   5. System displays "Login successful!" message.

   Post conditions:
   - The user is on the main page of the application.
   - The User can access application functions.

   Exceptions:

   2E1.

   a) User enters invalid credentials.
   b) The system displays an "Invalid username or password" message.
2. **Use case name:** Add visit
   Participating actors: User, The system
   Description: This use case describes how the user adds a new visit to the application.
   Trigger: User decides to add a visit to the application and clicks "Add Visit" button
   Preconditions:
   - The application is available and on-line.
   - Visits database is available and on-line.
   - The user is logged into the application.

   Normal Course:
   1. User clicks "Add Visit" button.
   2. User enters visits details; country name, city name, year visited, season visited, best feature, comment and rating.
   3. User clicks "Add" button.
   4. The system enters the visit details to the visits database.
   5. The system displays "Visit added successfully!" message to the user.
   6. The System updates the table interface with the newly added visits details.

   Post conditions:
   - User can see added new visit on the updated and displayed table interface.

   Exceptions:

   4E1.

   a) If there is a database error, the system displays an error message and the visit is not saved.

3. **Use case name:** Edit Visit
   Participating actors: User, The system
   Description: This use case describes how user edits an existing visits details.
   Trigger: User decides to make a change a visit details and clicks on the "Edit Visit" button
   Preconditions:
   - The application is available and on-line.
   - Visits database is available and on-line.
   - The user is logged into the application.

   Normal Course:
   1. User clicks "Edit Visit" button.
   2. The system displays a new screen where the user can enter the VisitID of the visit.
   3. User enters the VisitID of the visit which they want to change the details.
   4. User clicks "Dislay" button.
   5. System checks if that is a valid VisitID and if that visit is added by that user.
   6. System displays visit details in a form.
   7. User make changes for the desired details.
   8. User clicks to "Edit" button.
   9. The system updates the visits database with new details.
   10. System display "Visit updated successfully!" message

   Post conditions:

   - The system displays the edited version of the visit on the table interface.

   Exceptions:

   5E1.

   a) If VisitID is not valid or added by that user, system displays "Visit ID not found." Message.

   9E1.

   b) If there is a database error, the system displays an error message and the visit is not updated.

4. **Use case name:** Delete Visit
   Participating actors: User, The system
   Description: This use case describes how user deletes an existing visit.
   Trigger: User decides to delete a visit and clicks "Delete Visit" button.
   Preconditions:
   - The user is logged into the application.
   - The user must have an existing visit to delete.
   - The application is available and on-line.
   - Visits database is available and on-line.

5. **Use case name:** Display Visits by Year
   Participating actors: User, The system
   Description: This use case describes how a user can filter their visits by year detail and view them on the table interface.
   Trigger: The user enters a year and clicks "Display visits by Year" button.
   Preconditions:

- The user is logged into the application.
- The application is available and on-line.
- Visits database is available and on-line.

6. **Use case name:** Display location image
   Participating actors: User, The system
   Description: This use case describes how a user can view a specific visits location image by entering VisitID of the location.
   Trigger: The user enters a visitID and clicks “Display Image by VisitID”.
   Preconditions:
   - The user is logged into the application.
   - The application is available and on-line.
   - The location images are available and added to the application.
7. **Use case name:** Display best food visits
   Participating actors: User, The system
   Description: This use case describes how a user can view the visits with the best feature “food” sorted by their visit ratings on the table interface.
   Trigger: User clicks on the “Best food” button.
   Preconditions:
   - The user is logged into the application.
   - The application is available and on-line.
   - Visits database is available and on-line.
8. **Use case name:** Display Most Visited Country
   Participating actors: User, The system
   Description: This use case describes how a user can view an informative message about their most visited country.
   Trigger: User clicks on the “Most Visited Country” button.
   Preconditions:
   - The user is logged into the application.
   - The application is available and on-line.
   - Visits database is available and on-line.
9. **Use case name**: Display Spring Visits
   Participating actors: User, The system
   Description: This use case describes how a user can view the visits that are visited only in spring on the table interface.
   Trigger: User clicks on the “Spring Visits” button.
   Preconditions:
   - The user is logged into the application.
   - The application is available and on-line.
   - Visits database is available and on-line.

10. **Use case name:** Display All visits
    Participating actors: User, The system
    Description: This use case describes how a user can view all their visit on the table interface.
    Trigger: User clicks on the “All Visits” button.
    Preconditions:
    - The user is logged into the application.
    - The application is available and on-line.

- Visits database is available and on-line.

11. **Use case name:** Share Visit
    Participating actors: User, Friend, The system
    Description: This use case describes how a user can share their visits with a friend.
    Trigger: User enters the visitID and username of their friend and then clicks on the " Share visit with friend" button.
    Preconditions:
    - The user is logged into the application.
    - The user knows the friend username.
    - The application is available and on-line.
    - Visits database is available and on-line.
12. **Use case name:** View Shared Visits
    Participating actors: User, The system
    Description: This use case describes how a user can view the shared visit with them on the table interface.
    Trigger: User clicks on the "Shared with me" button.
    Preconditions:
    - The user is logged into the application.
    - The user has at least one shared visit with them.
    - The application is available and on-line.
    - Visits database is available and on-line.
13. **Use case name:** User Authentication
    Participating actors: The system, user
    Description: This use case describes how the system authenticates a user before allowing access to functions that require authentication.
    Trigger: User use a function that needs user authentication.
    Preconditions:
    - The application is available and on-line.
    - User_info database is available and on-line.

3.4.4 Relationships among Actors and Use Cases

1. User:

- Login: Users must authenticate themselves to access the system.
- Add Visit: Users can add details of their travel visits. This use case includes "User authentication" to ensure that only logged-in users can add visits.
- Edit Visit: Users can modify the details of their existing visits. This use case includes "User authentication" to ensure that users can only edit their visits.
- Delete Visit: Users can remove their visits from the system. This use case includes "User authentication" to ensure that users can only delete their visits.
- Display visits by year: Users can filter and display visits by the visited year of the visit. This use case includes "User authentication" to show only users visit on the table interface.
- Display visits by best food: Users can view visits with the best feature food. This use case includes "User authentication" to show only users visit on the table interface.
- Display visits visited in spring: Users can view visits, visited during spring season. This use case includes "User authentication" to show only users visit on the table interface.

- Display all visits: Users can view a list of all their visits. This use case includes "User authentication" to show only users visit on the table interface.
- Share visit: Users can share visit details with their friends by entering the friend's username and the visits ID to the application. This use case includes "Login" for user and friend which means friend must have a valid username for the application.
- Display shared visits with me: Users can see visits that has been shared with them by their friends. This use case includes "User authentication" to show only users shared visits on the table interface.
- Display location images: Users can view images of locations they have visited.

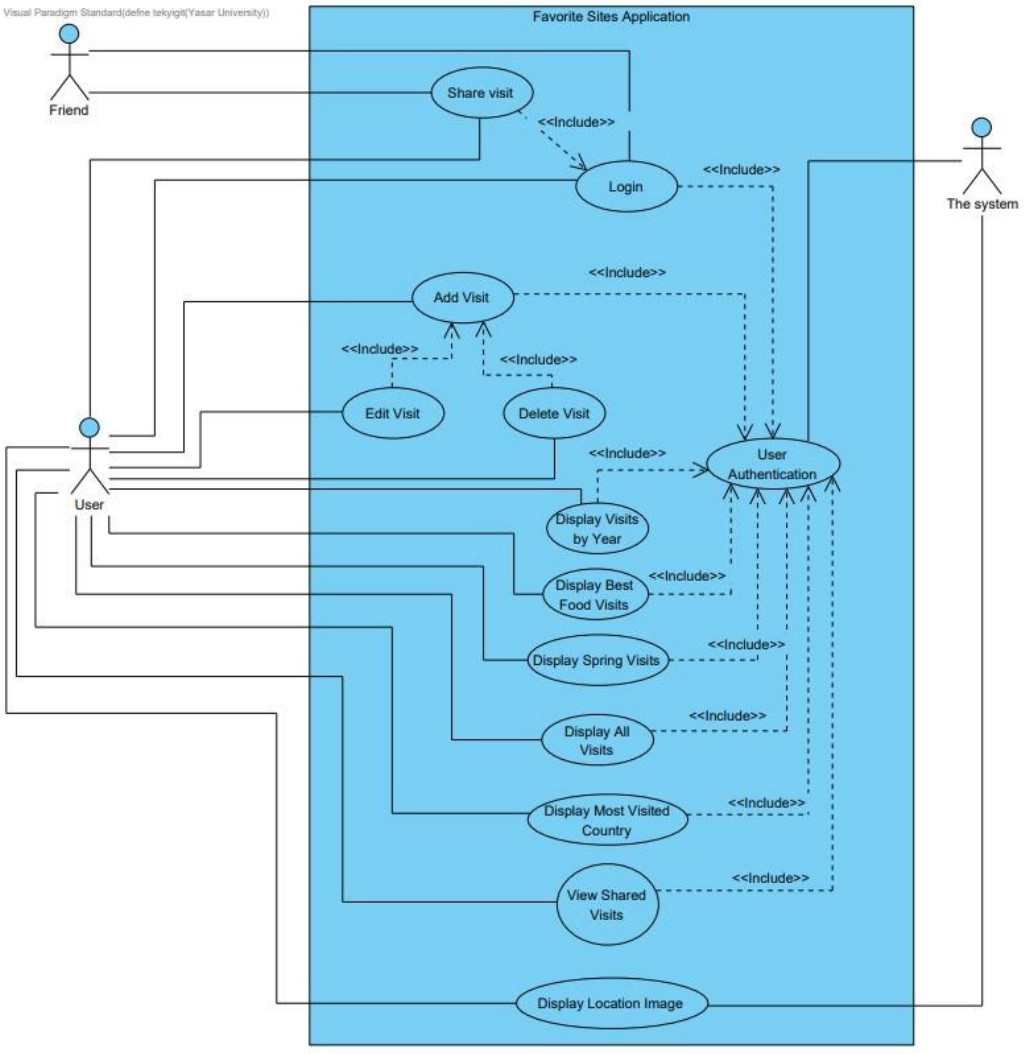
2. Friend:

- Share visits: Friends can be shared visits with them. This use case includes "User authentication" to ensure that only authorized users can be shared visits.

3. System:

- The system manages all user interactions and data operations, so that users can only perform actions they are authorized to do.

3.4.5 Use Case Diagram

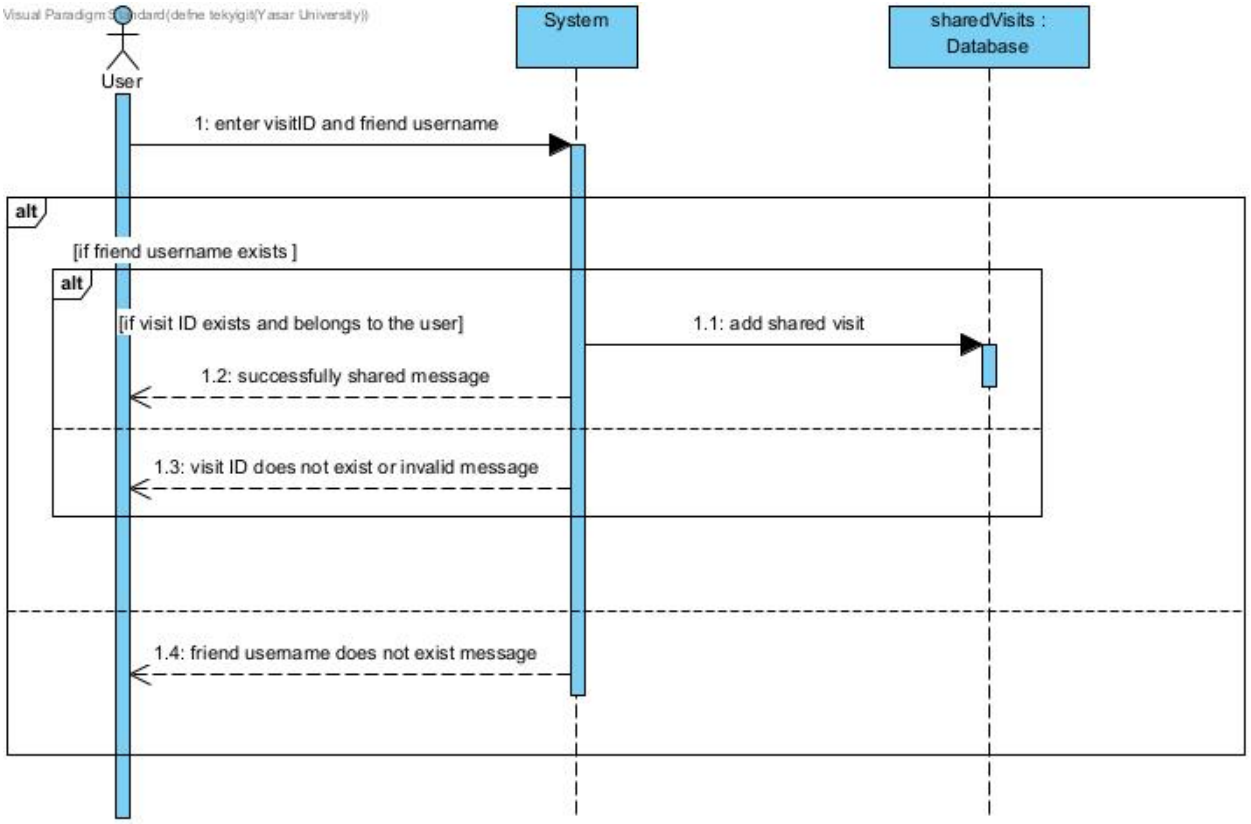


This diagram includes all the use cases described in detail in the "Use Case Forms" section. Each use case represents a specific functionality that users can perform in the system such as adding, editing, or deleting visits, displaying visits by specific features, sharing visits with friends, and viewing shared visits. The use case diagram provides a high-level overview of the system's functionalities and interactions for a better understanding of the system's functionalities.

4 Behavioral Models

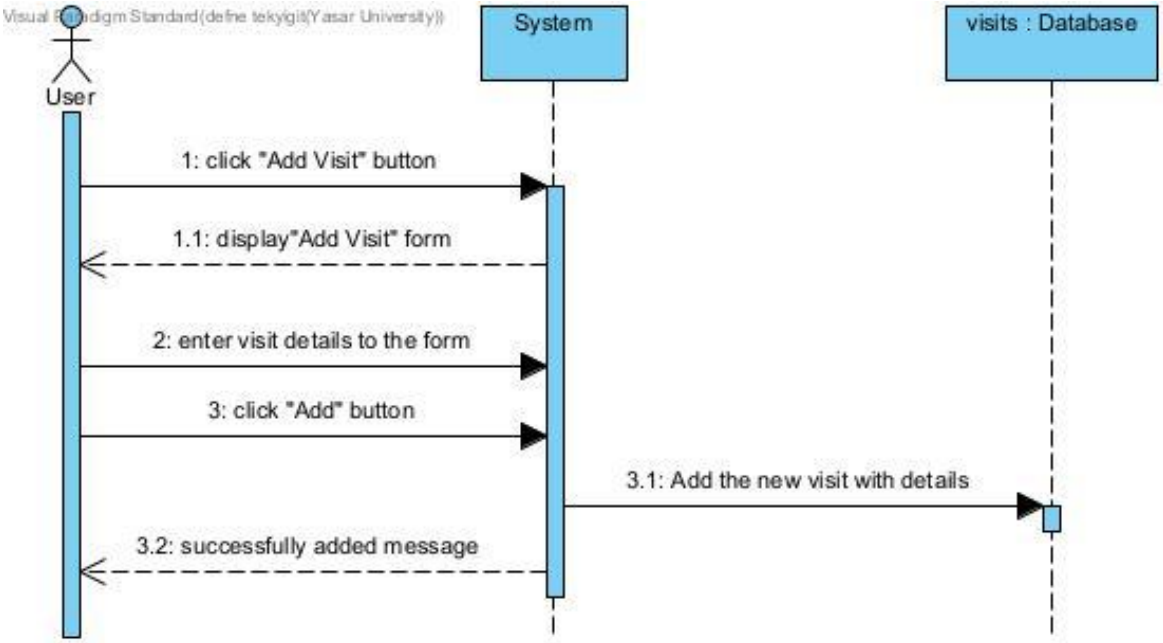
4.1 Sequence Diagram

1. **Share Visit**
   This sequence diagram describes how sharing visits work. First user enters the visit ID and friend's username to the system. And the system checks if username exists and if the visit ID exists and valid, system adds the shared visits details into the sharedVisits database. If username and visit ID are invalid, then returns informative messages to the user.



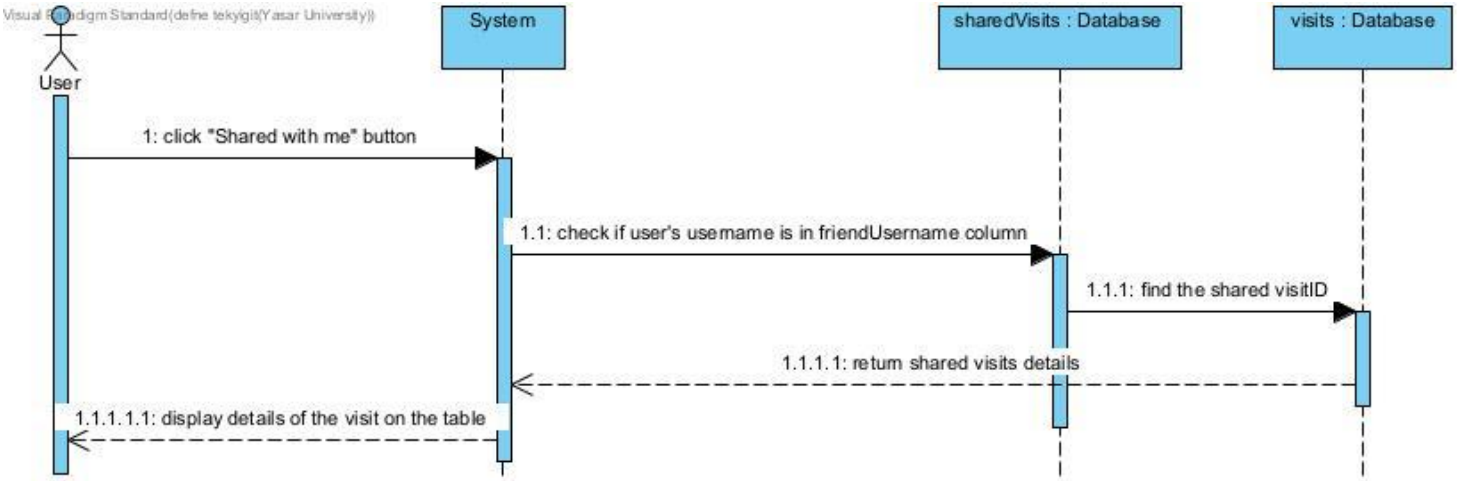
2. **Add Visit:**

This sequence diagram describes how user can add a new visit. First user clicks the “Add Visit” button, system displays “Add Visit” form and user enters their visits details. The system adds the new visit with correct details to the visits database then returns successfully added message to the user.



3. **Display shared visits with me**

This sequence diagram describes how user can view the visits that are shared with them. First user clicks the “shared with me” button then, system checks if are there any shared visits with the user by checking if user’s username is in sharedVisits table’s friend username column if system finds any, then system finds the shared visits id’s in the visits table and then displays the shared visits details on table interface so user can view the visit and its details.



5 Structural Models

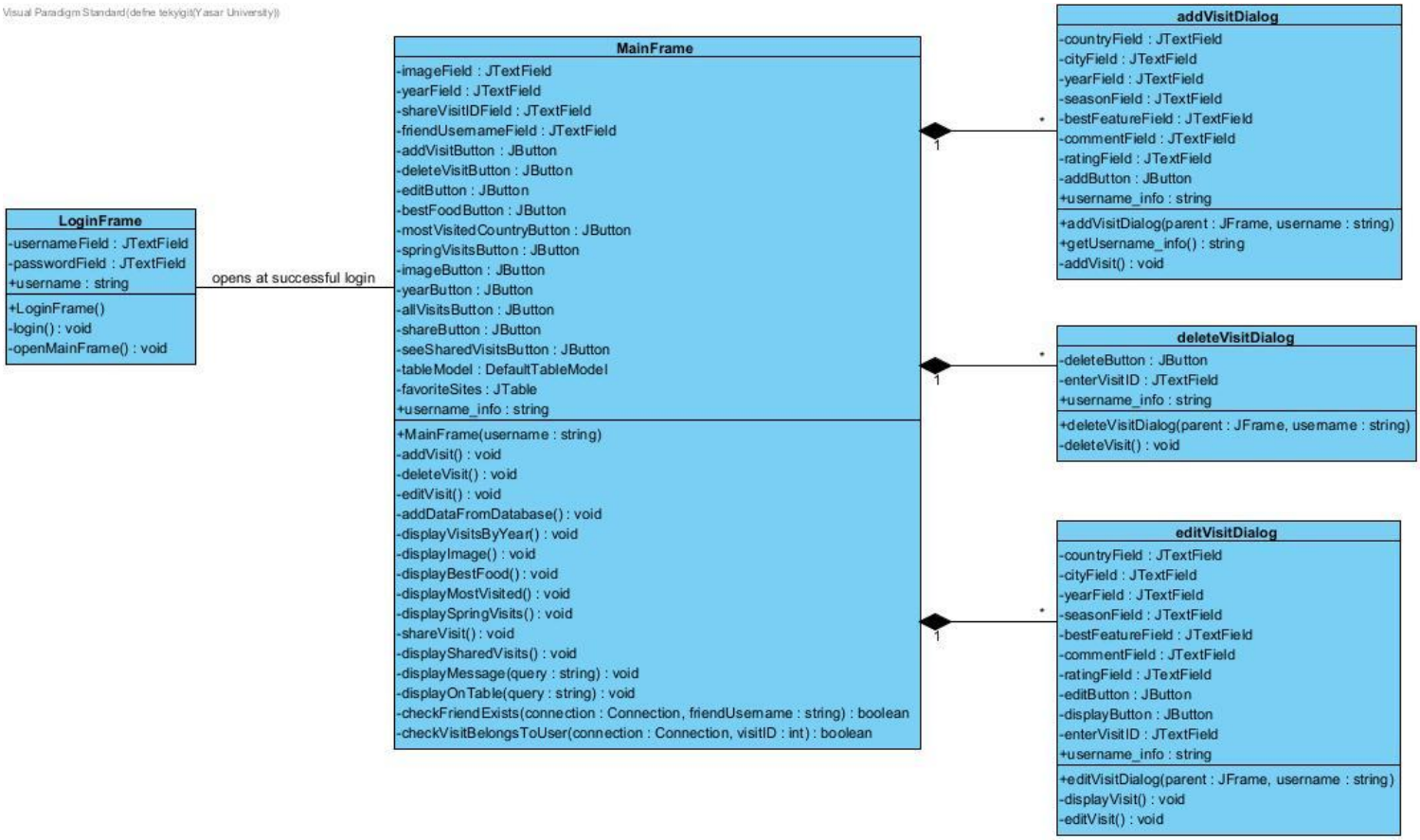
5.1 Class Diagram

This class diagram represents structure and relationship between different components of the function. The diagram includes five main classes: LoginFrame, MainFrame, addVisitDialog, deleteVisitDialog, editVisitDialog classes of the application. All classes have their attributes and operations representing the functionalities and interactions of the application.

Login Frame: This class handles user authentication and control access to the MainFrame, if the login is successful LoginFrame shows MainFrame.

MainFrame: This class is the central part of the application, and it contains addVisitDialog, deleteVisitDialog and editVisitDialog as part of functions and they have a composition relationship with MainFrame.

addVisitDialog, deleteVisitDialog and editVisitDialog: These classes manage specific functions related to visits. “addVisitDialog” is for adding of new visit, “deleteVisitDialog” is for deleting existing visits and “editVisitDialog” is for editing and updating existing visits.

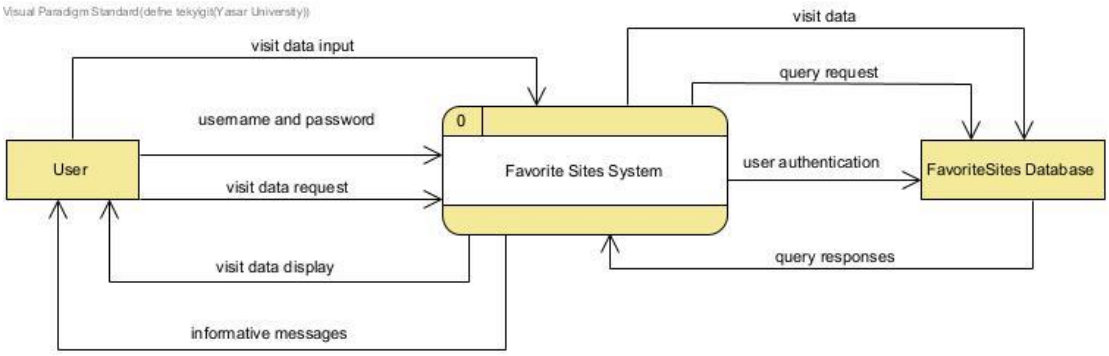


6 Process Modeling

6.1 Data Flow Diagram (DFD)

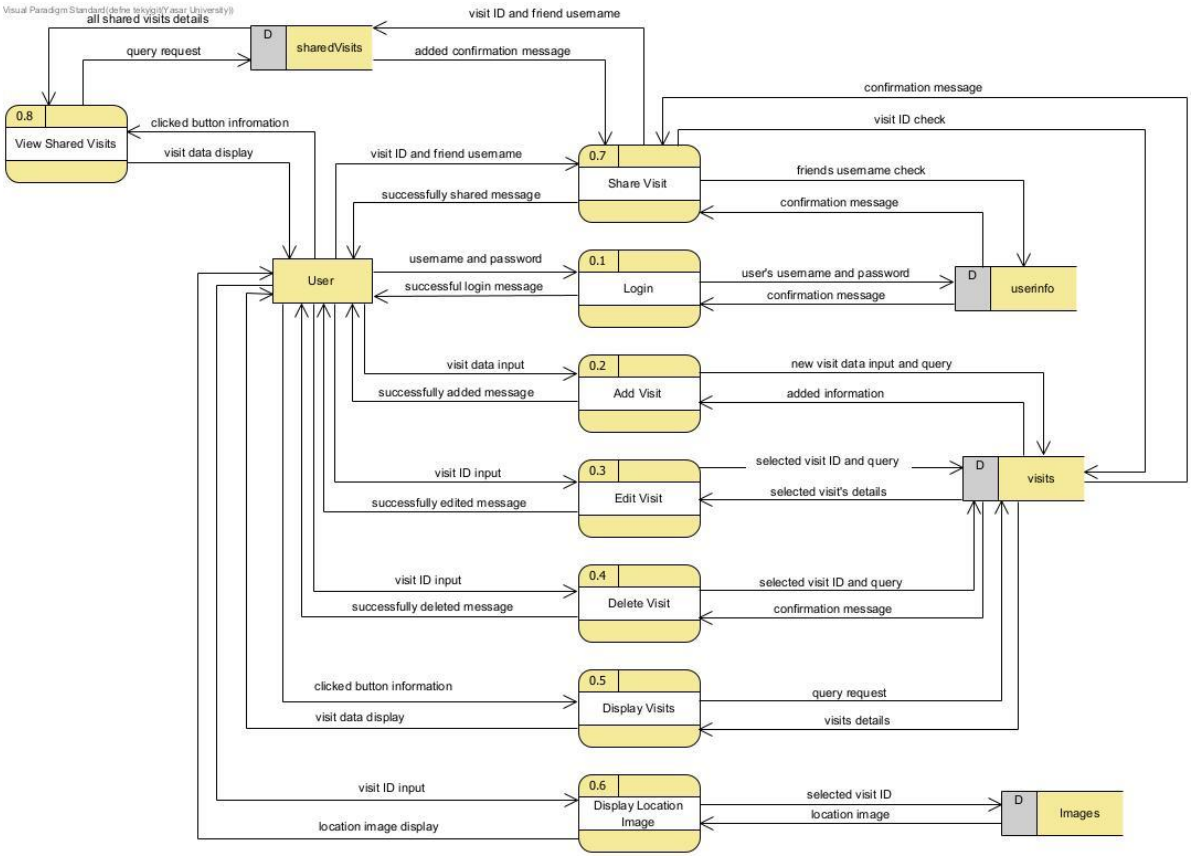
Context Diagram:

This context diagram gives an overview of the Favorite Sites System and includes important data flows for understanding user's interaction with the application and applications interaction with the database.



Level 0 Diagram:

This data flow diagram covers almost all functionalities of the application. And it includes specific data flows for the functions, 3 important data stores and user's interactions with the processes. Diagram shows how system processes various user requests, manages visit data, and interacts with the database to provide the desired information and functionalities.

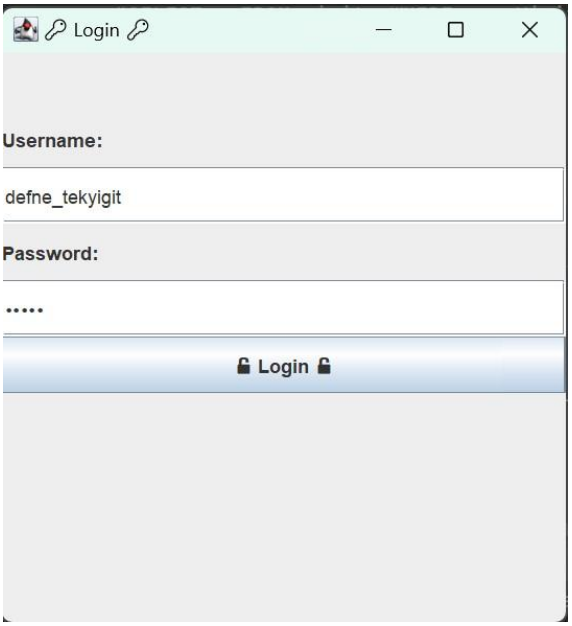


7 Graphical User Interface(s) (GUIs)

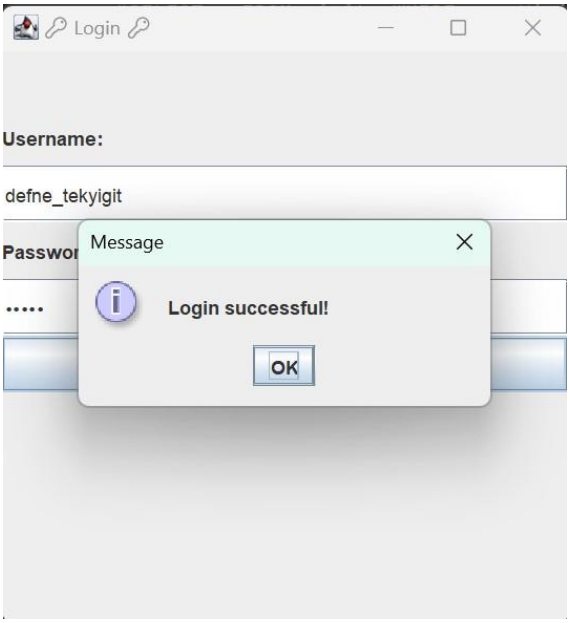
Put the images of your GUIs, also put the output screenshots of each function that is tested with sample data. Briefly explain the purpose of each GUI. Do not forget to add the form title for each image.

Login Form: This form allows users to access the system by entering their username and password.

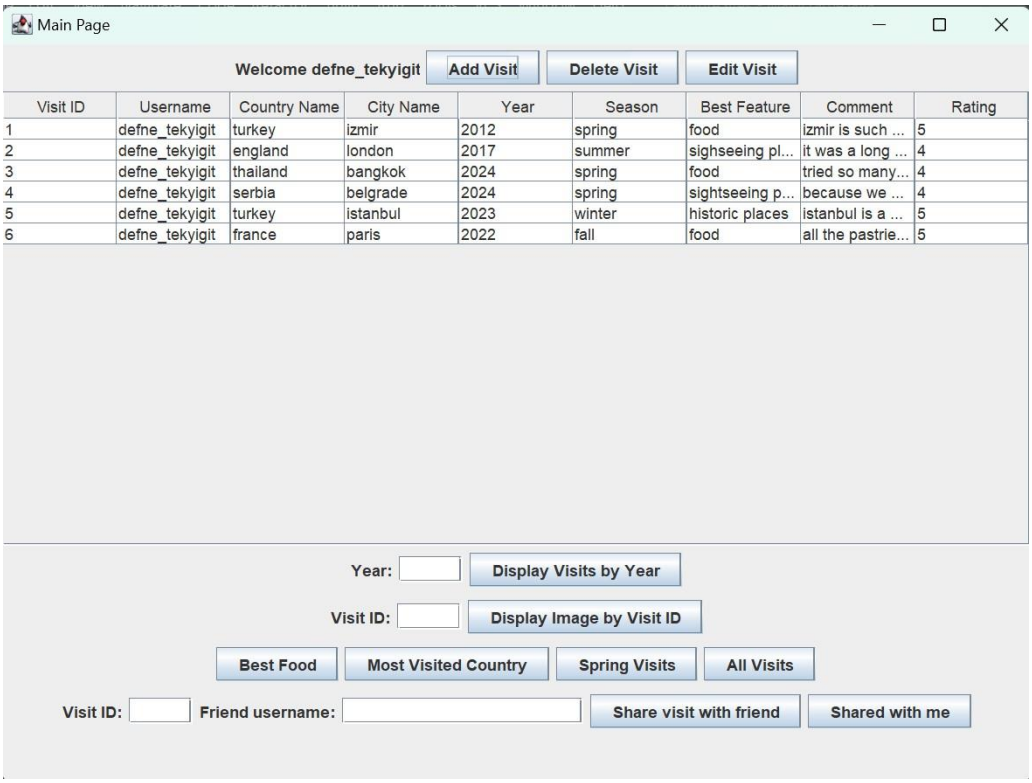
First user enters username and password:



After clicking login button:



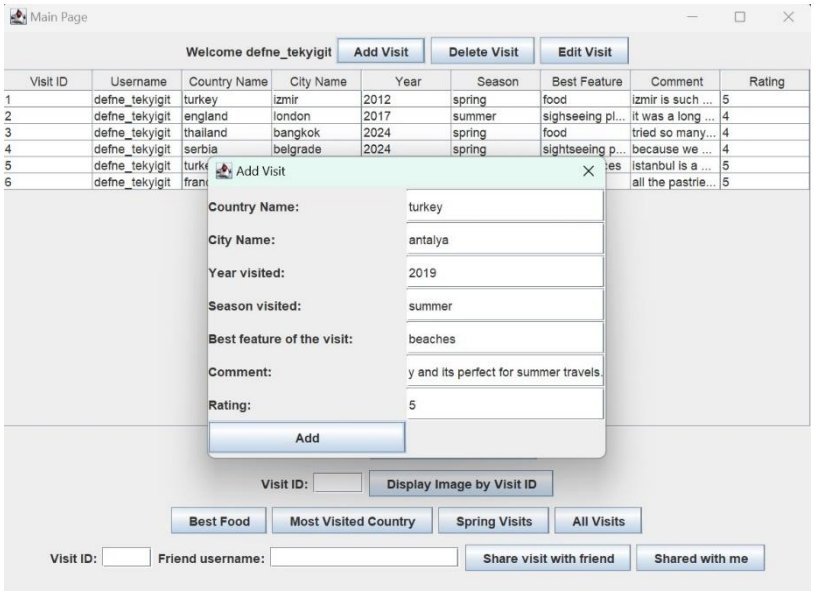
Main Page: This form is the main page of the application and allows users to view and manage their visits.



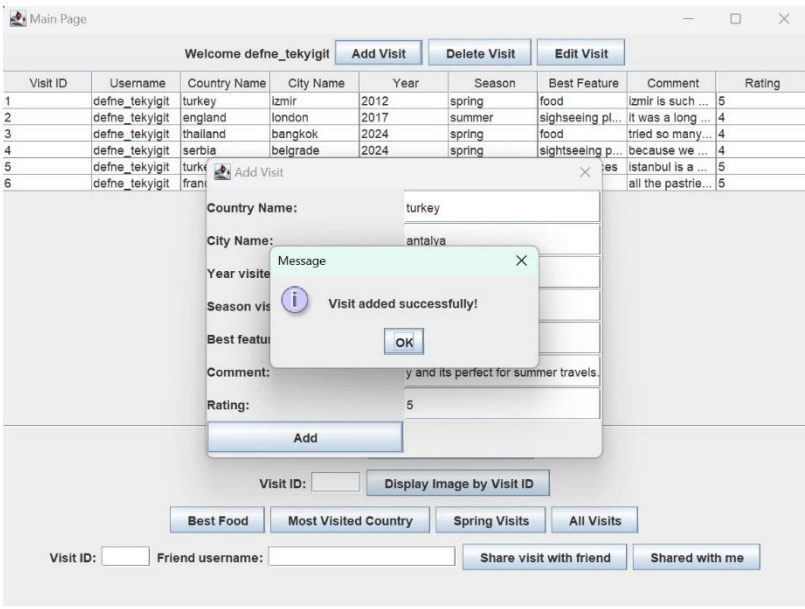
Visit ID	Username	Country Name	City Name	Year	Season	Best Feature	Comment	Rating
1	defne_tekyigit	turkey	izmir	2012	spring	food	izmir is such ...	5
2	defne_tekyigit	england	london	2017	summer	sighseeing pl...	it was a long ...	4
3	defne_tekyigit	thailand	bangkok	2024	spring	food	tried so many...	4
4	defne_tekyigit	serbia	belgrade	2024	spring	sightseeing p...	because we ...	4
5	defne_tekyigit	turkey	istanbul	2023	winter	historic places	istanbul is a ...	5
6	defne_tekyigit	france	paris	2022	fall	food	all the pastrie...	5

Add Visit Dialog: This form allows users to add new visits by entering visits details such as country name, city name, year visited, season visited, best feature, comments and ratings.

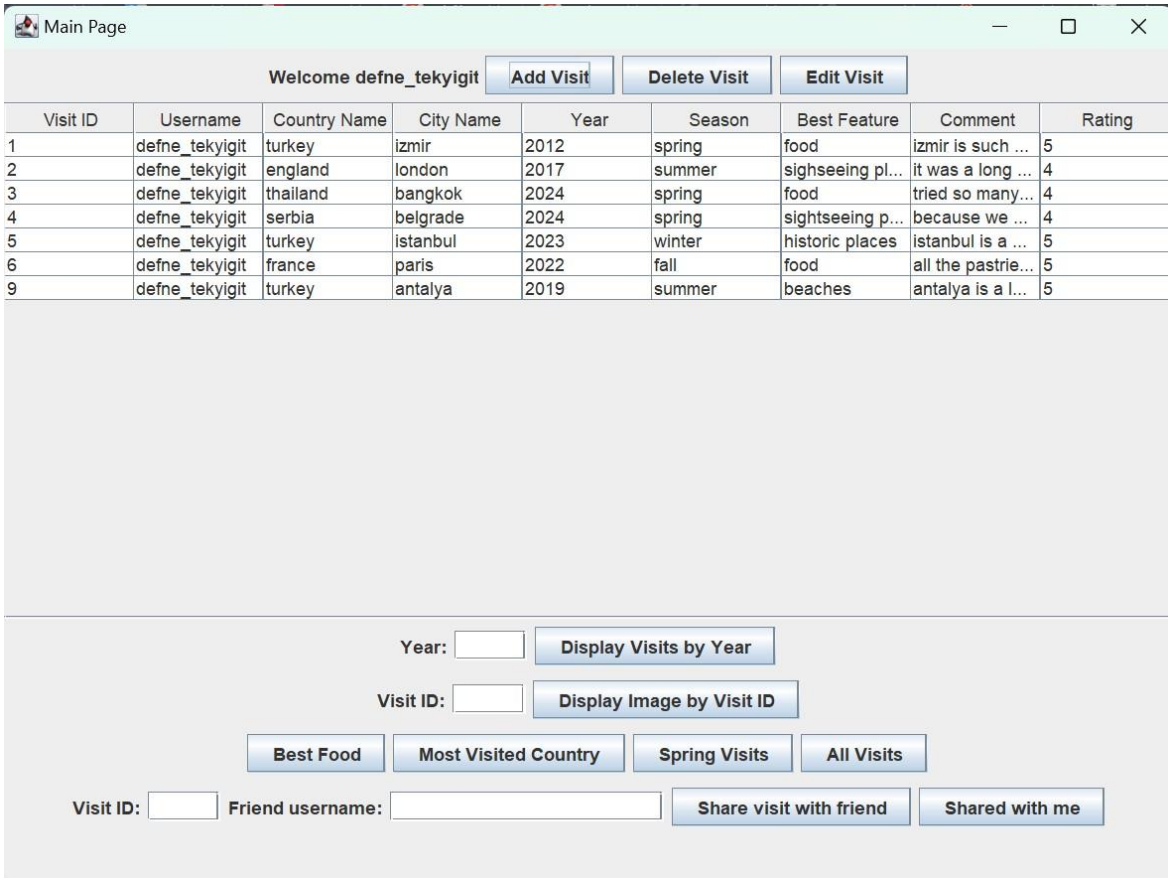
First user enters visit details:



After clicking add button:

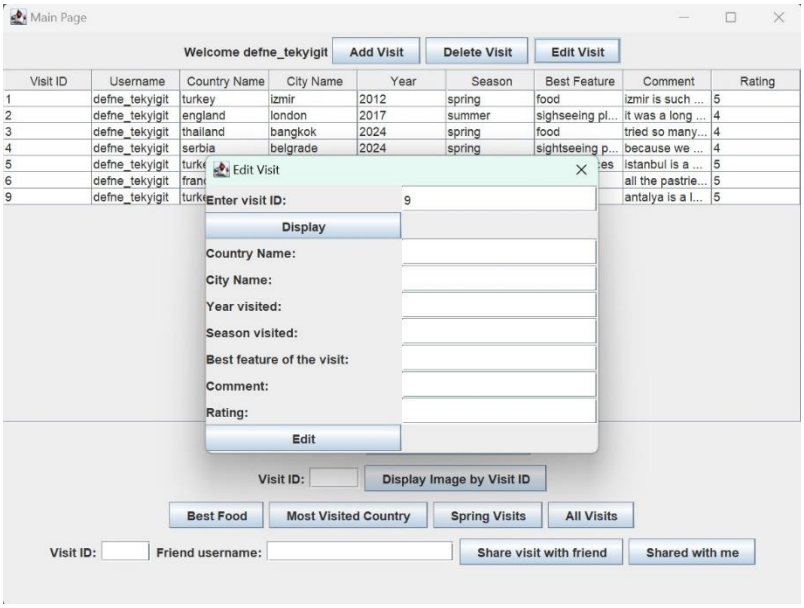


After that, system updates the new visit to the table :

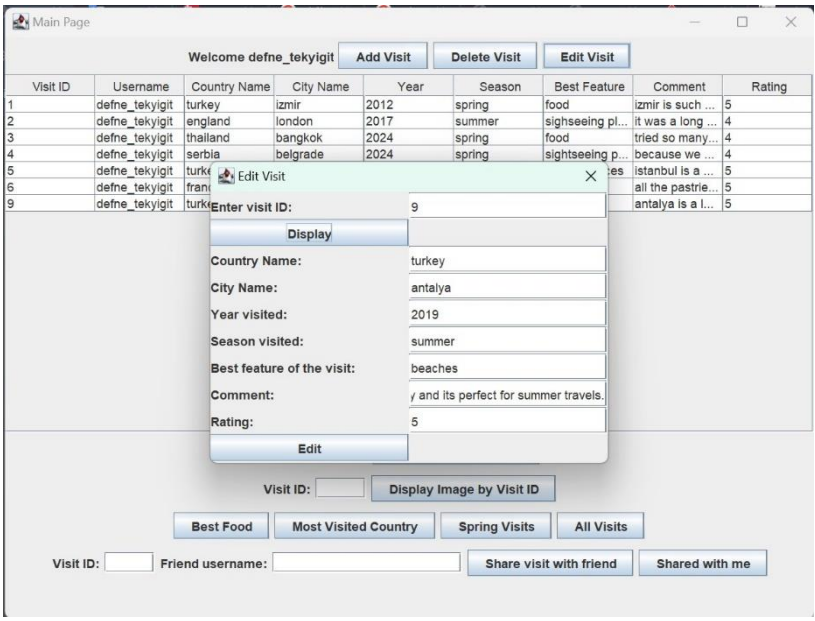


Edit Visit Dialog: This form allows users to make changes on their visit's details, user needs to enter the visit ID first so that system can display the visit details on the form then updates the visit.

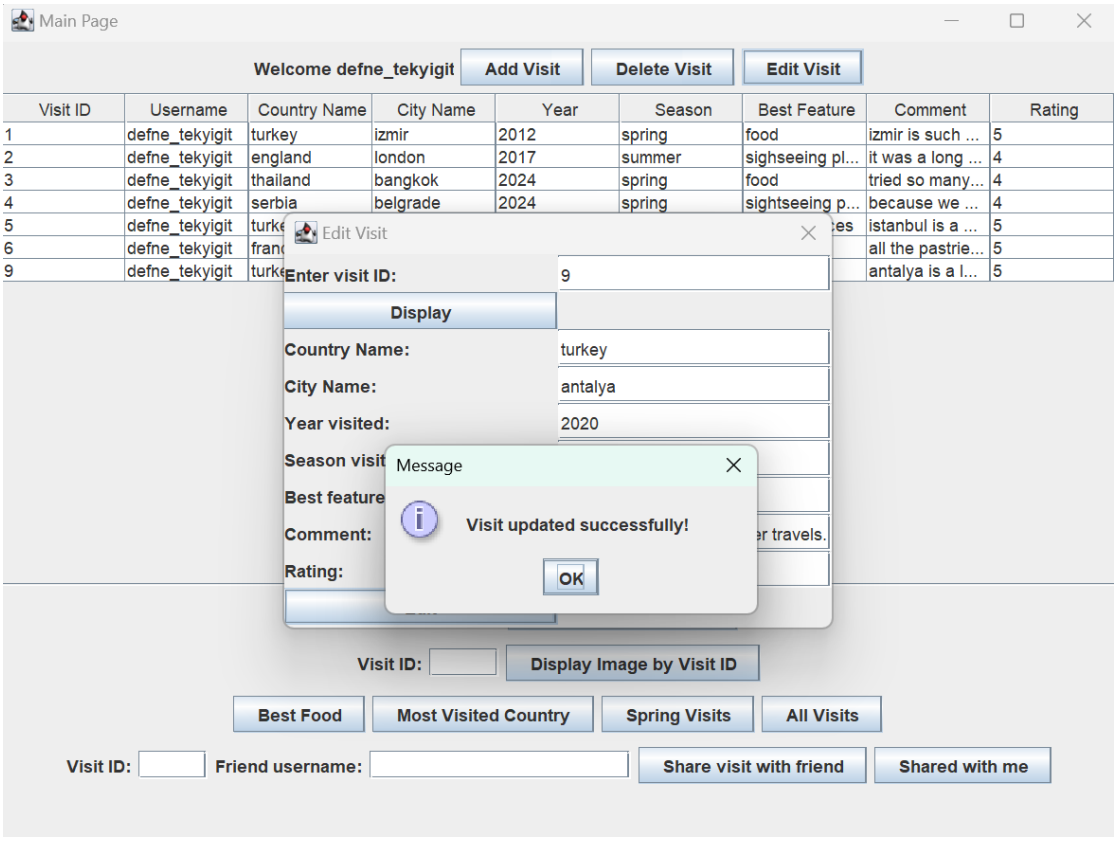
First user enters visit ID:



Then system displays visit's details:

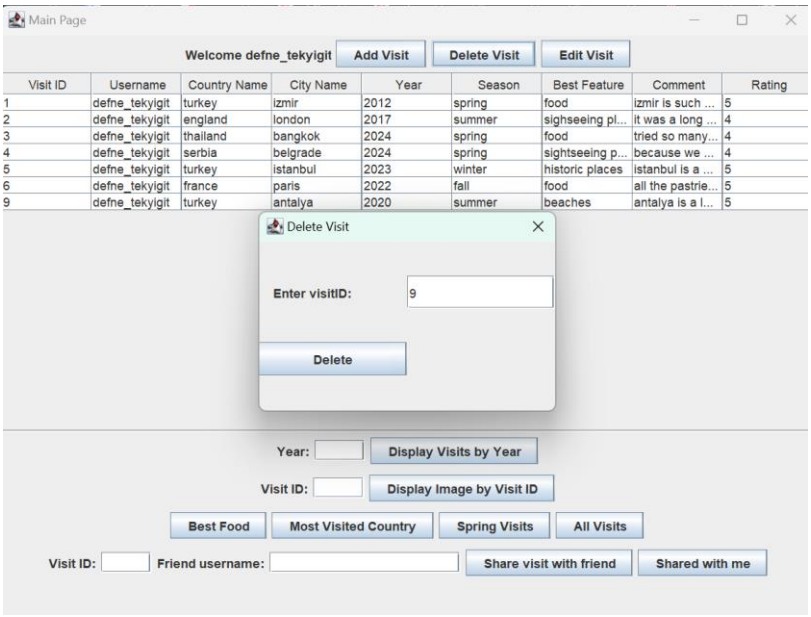


After making changes and clicking edit button:

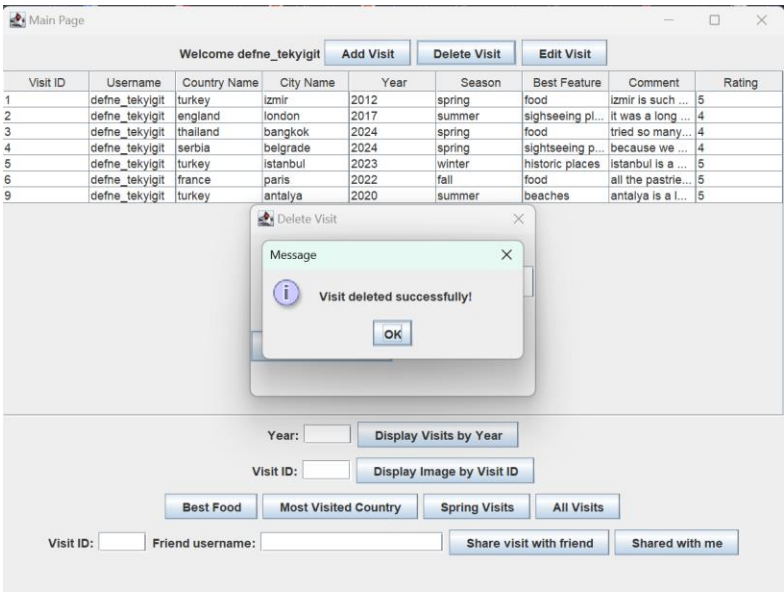


Delete Visit Dialog: This form allows users to edit an existing visit by entering visit ID.

First user enters visit ID:

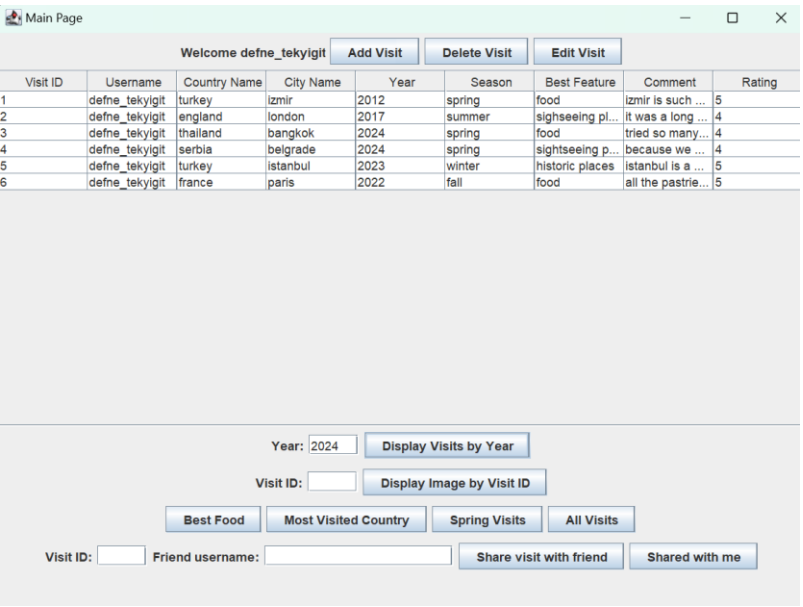


Then clicks delete button:

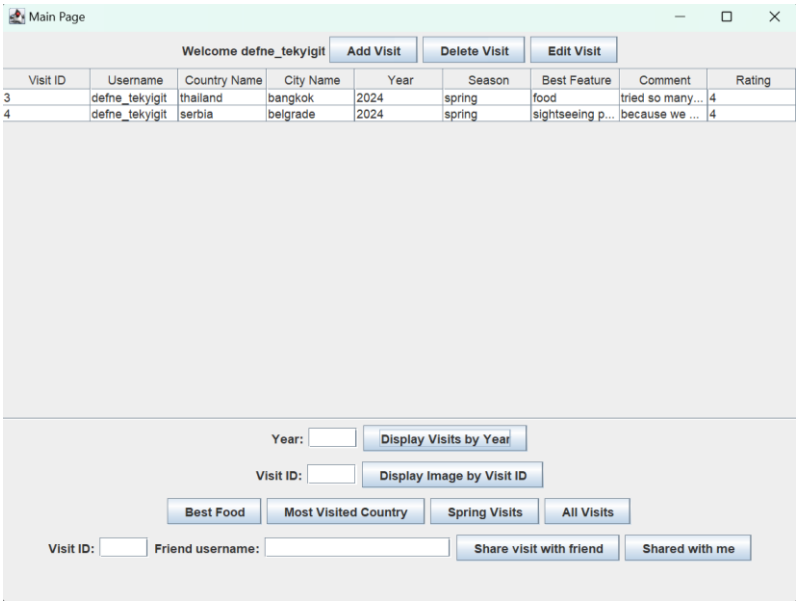


Display Visits by Year: This form allows users to view visits by year, which was entered by user.

First user enters year:

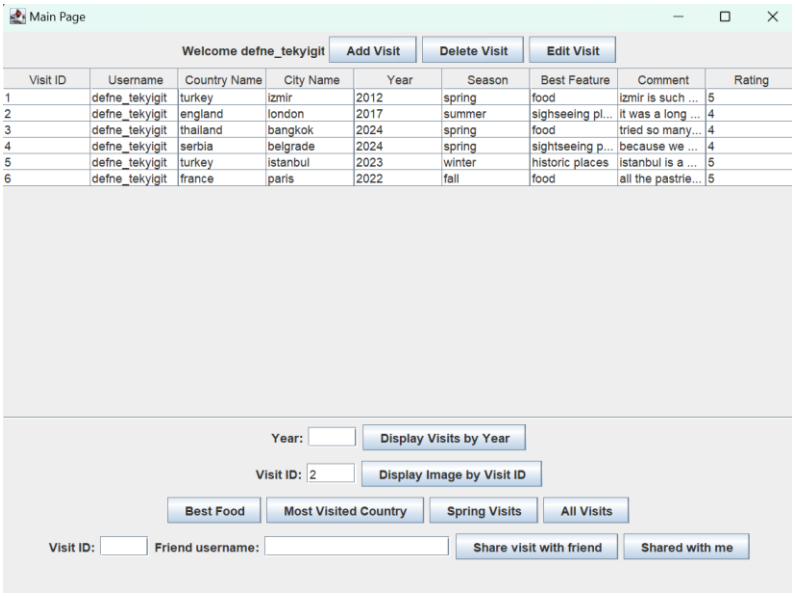


Then system displays visits by that year on the table interface:

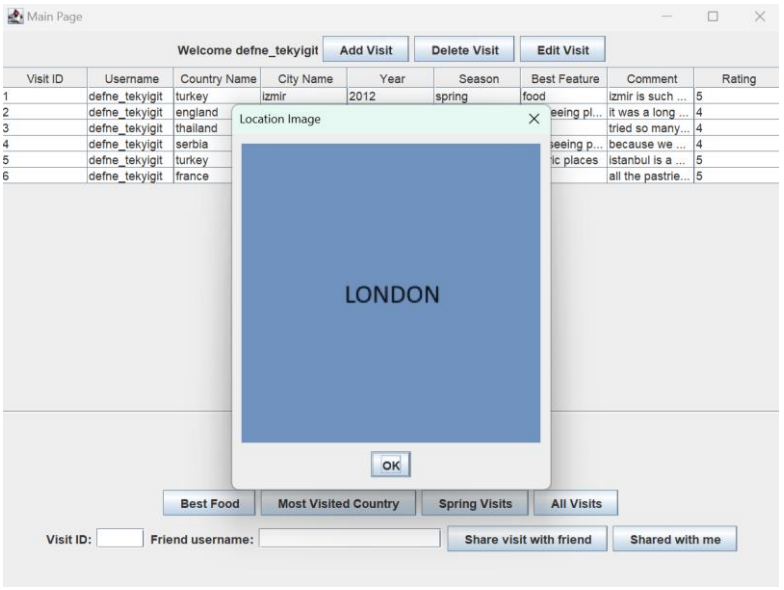


Display image by visit ID: This form allows users to view a location image by entering desired locations visit ID.

First user enters visit ID and clicks display button:

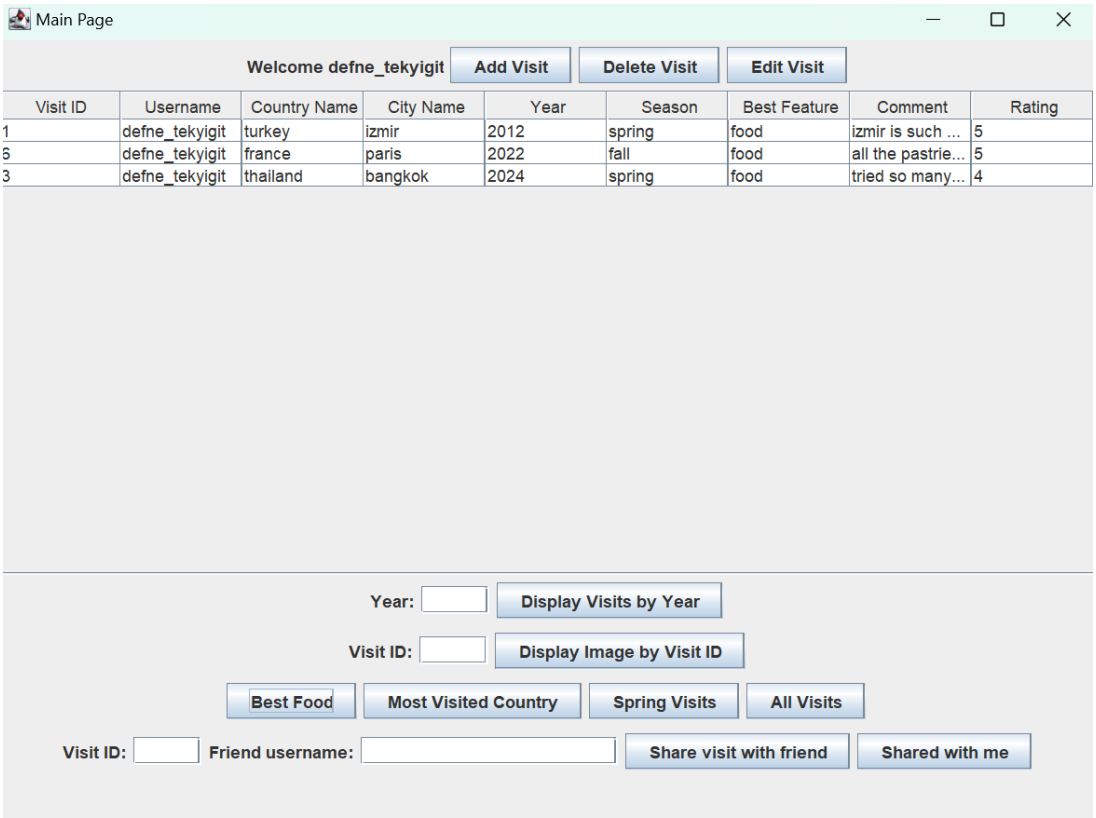


Then system displays location image:



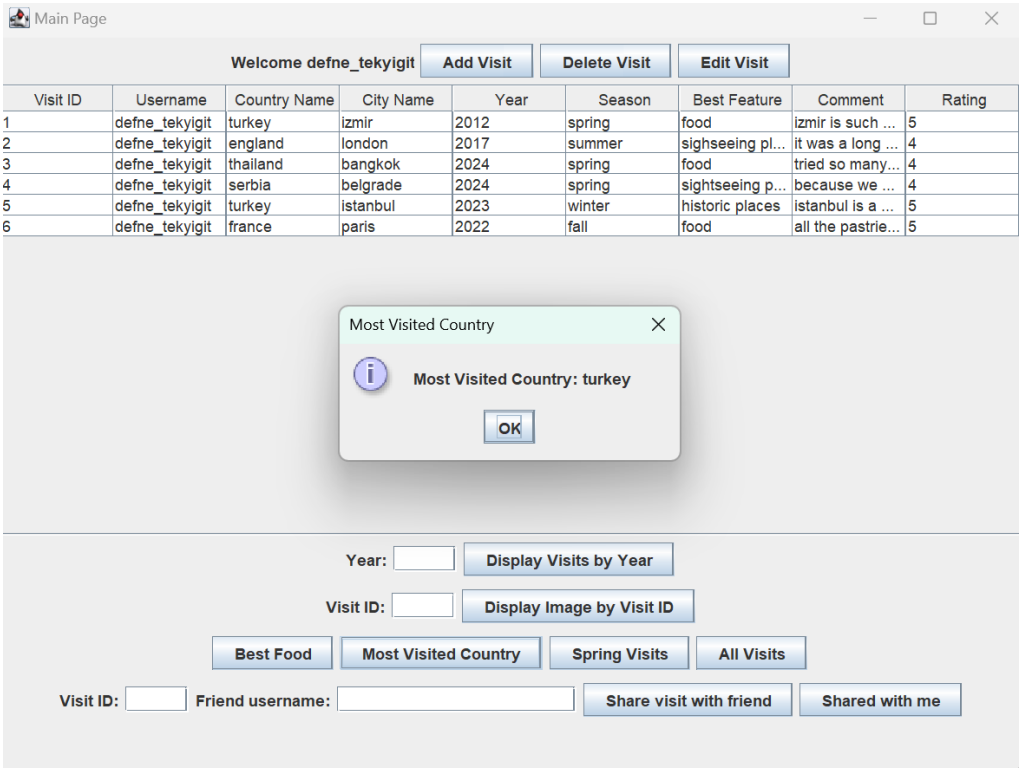
Display Best Food Visits: This form allows users to view the visits where their best features are food and displays them with their sorted rankings.

First user clicks to the best food button then system displays them on the table interface:



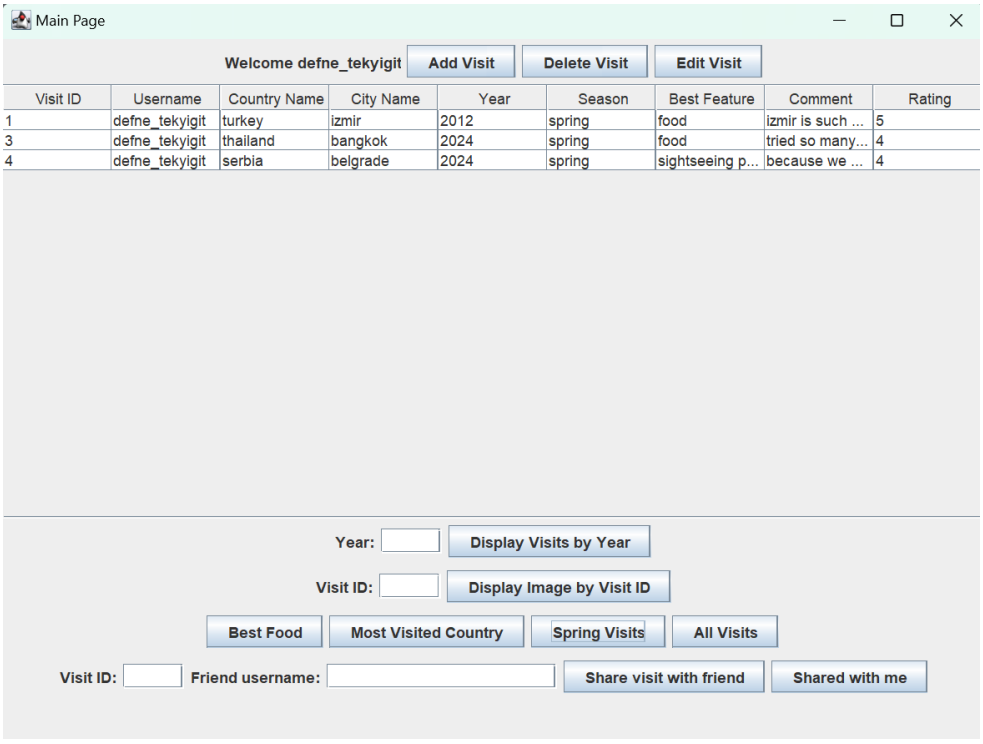
Display Most Visited Country: This form allows users to see their most visited country in a message dialog.

First user clicks to the most visited country button then system displays the message.



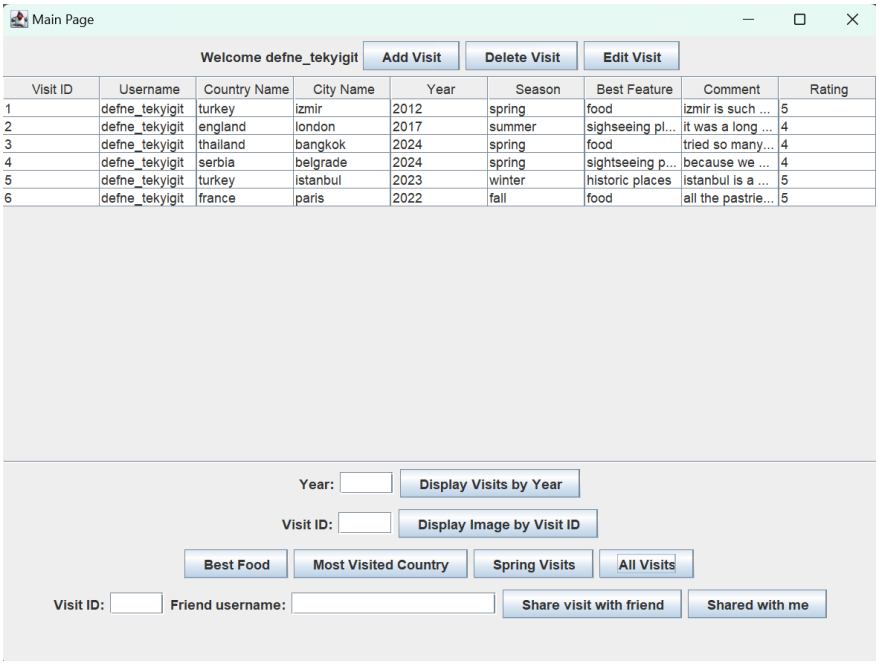
Display Spring Visits: This form allows user to view their visits that are visited in the spring season.

First user clicks to the spring visits button then system displays spring visit on the table interface.



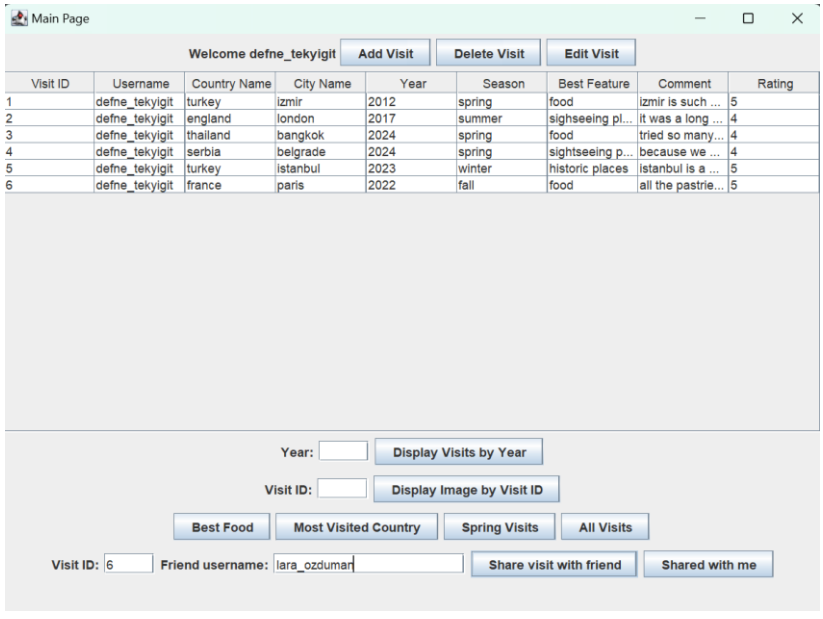
Display All Visits: This form allows users to view all of their visits on the table interface.

First user clicks to all visits button then system displays all visits on the table interface.

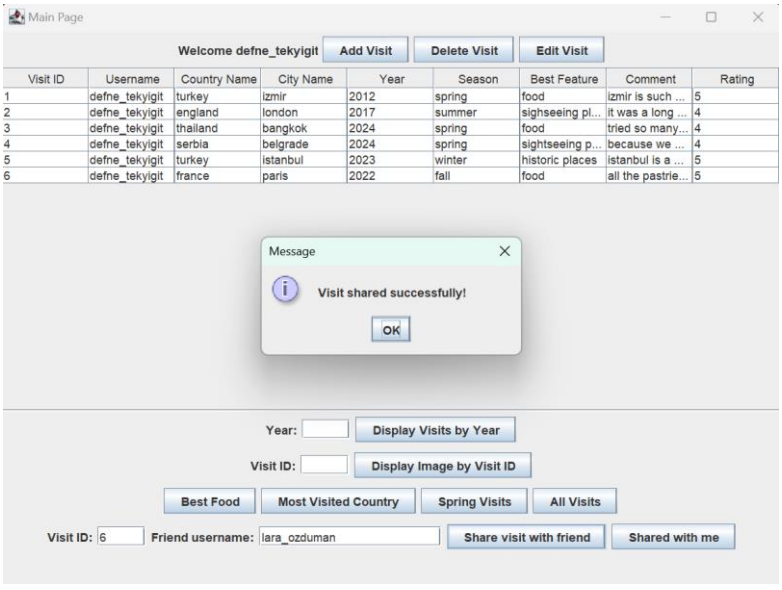


Share Visit: This form allows users to share their visits with their friends by entering desired visit ID and their friend's username.

First user enters Visit ID and friend's username:

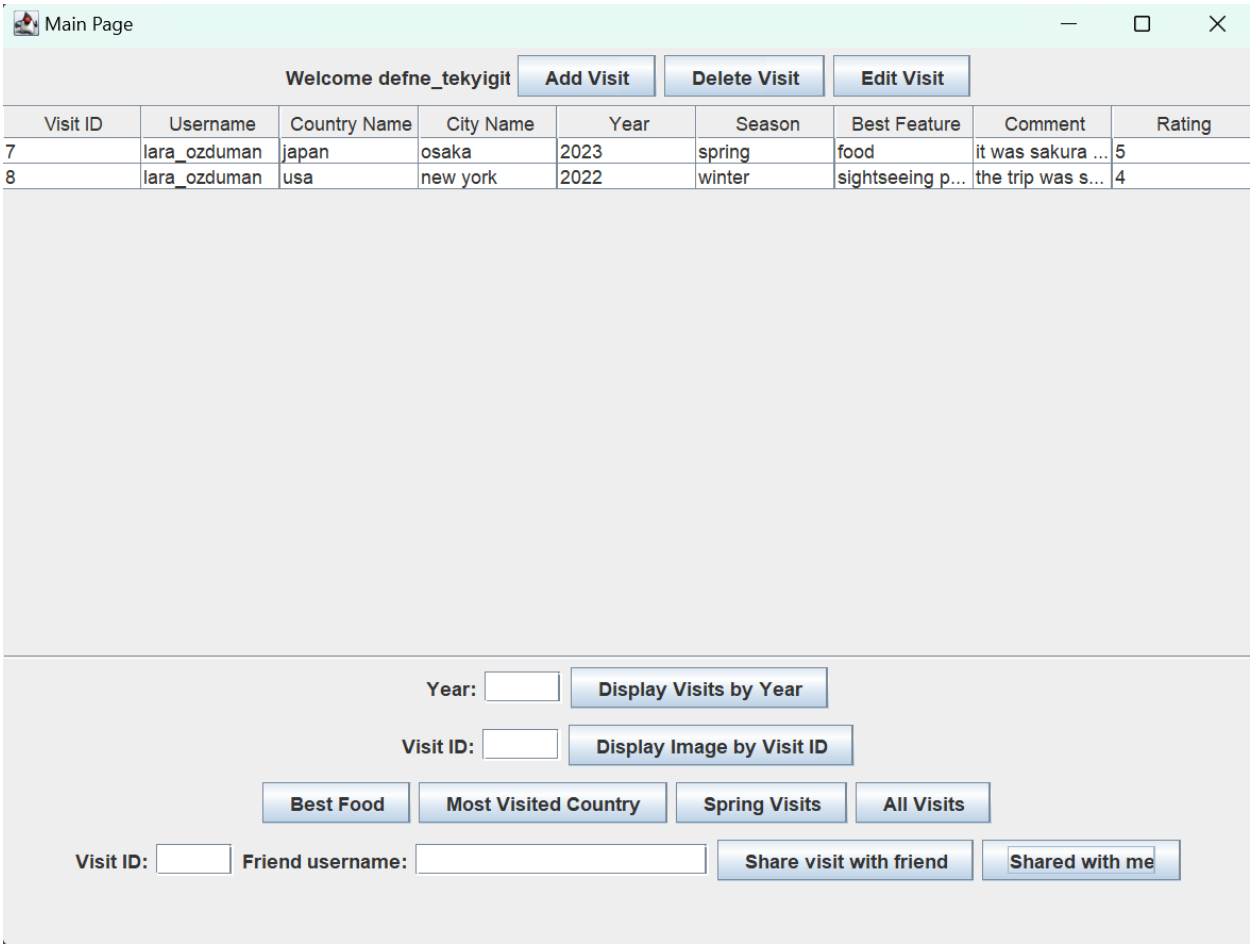


After clicking share visit with friend button:



View Shared Visits: This form allows users to view visits shared with them.

First user clicks to the shared with me button then system displays all the shared visits with user:



8 Conclusion and Future Work

In this project I implemented an application where users can manage and share their visits. The Favorite Sites Application keeps users visits, allows them to share visits with their friends. The implementation included creating databases, designing clear and simple GUI and adding various functionalities with Java Swing and MySQL. The application meets the requirements outlined in the project specifications and handles errors well. Throughout this project I prepared functional, nonfunctional requirements, use case forms, use case diagram, sequence diagrams, a class diagram and data flow diagrams by using this course's teachings. For additional features and functions:

Sharing location images with friends can be added to the system, so that users can share their favorite locations images with their friends too.

Adding friends comment to the shared visits can be added for increasing users' interaction with other users, with this function users can add their comment about friends visit to their friends shared visit so that friend later can view their comment.

Gamification elements can be added to the application, where users can get rewards for adding new visits or sharing more visits with their friends.