**CHAPTER 1: INTRODUCTION**

**1.1 Project Description**

The Bankify-Online Banking System is a project aimed at developing a user-friendly, secure, and convenient online banking application called Bankify. The project addresses the issues faced by users in existing online banking applications, such as complex interfaces, difficulties in navigation, and concerns about security and data privacy. Bankify aims to provide a simple and intuitive user interface, essential banking services, and robust security measures to ensure a seamless and reliable banking experience.

**1.2 Tools and Technology Used**

* **Frontend Development**: The user interface of the Bankify application was created using HTML, CSS, and JavaScript. These technologies were utilized to design and develop an intuitive and visually appealing frontend that offers a seamless user experience.
* **Backend Development**: The backend functionality of the Bankify application was implemented using PHP as the server-side scripting language. PHP enables the handling of server-side processing, data manipulation, and integration with the frontend components.
* **Database Management System**: The Bankify application employed MySQL, an open-source relational database management system (RDBMS), to store and manage the banking-related data. MySQL ensures efficient data storage, retrieval, and secure management of user account details and transaction information.
* **Version Control**: Git, a widely adopted version control system, was utilized to track changes, manage the codebase, and facilitate collaboration among developers working on the Bankify project. It enables effective code management, branching, and merging to ensure a smooth development process
* **Deployment and Hosting**: The Bankify application was deployed on a web server, such as Apache, and hosted on a PHP hosting server environment. This setup ensured that the application is accessible to users via the internet and maintained a stable and secure environment for hosting the banking system.

By leveraging these technologies, the Bankify-Online Banking System was developed with a reliable and scalable architecture, ensuring a robust and efficient banking application for users.

**2: SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 INTRODUCTION**

**2.1.1 Purpose**

The purpose of the Bankify application is to provide users with a convenient, secure, and user-friendly online banking experience. It aims to simplify banking tasks, such as money transfers, loan management, account creation, and closure, while ensuring the privacy and security of users' personal and financial information. The application strives to make online banking accessible to users of all technical backgrounds and age groups, offering essential banking services through a simplistic and modern user interface.

**2.1.2 Intended Audience**

The intended audience for the Bankify application is broad and includes individuals who are seeking a modern and convenient online banking solution. It targets users of various age groups and technical backgrounds who value simplicity, security, and ease of use in their banking experience. Bankify aims to cater to both tech-savvy individuals and those who may be less familiar with online banking, providing a user-friendly interface that is accessible to all.

**2.1.3 Project Scope**

The project scope for the Bankify application includes the development of a user-friendly online banking system that offers essential banking services such as money transfers, loan management, account creation, and closure. It involves designing and implementing a secure and robust application with features that cater to the needs of a broad audience. The scope also encompasses ensuring cross-platform compatibility, integrating necessary security measures, and providing reliable performance even during high traffic periods. Additionally, the project may explore integration with third-party services and include personal finance management tools to enhance the overall banking experience.

**2.1.4 Benefits**

* **Convenience:** Bankify provides users with the convenience of accessing their bank accounts and performing essential banking tasks anytime, anywhere, using their preferred devices. Users can manage their finances, transfer money, apply for loans, and perform other banking operations without the need to visit a physical branch.
* **User-Friendly Interface**: Bankify features a user-friendly interface that is intuitive and easy to navigate. Users of all technical backgrounds and age groups can quickly adapt to the application and carry out banking transactions seamlessly, making it accessible to a wide range of users.
* **Enhanced Security**: Bankify prioritizes the security and privacy of users' personal and financial information. Robust security measures, such as data encryption and two-factor authentication, ensure that transactions and data remain secure, instilling confidence in users to trust the application with their sensitive information.
* **Time and Cost Savings**: Bankify eliminates the need for users to physically visit a bank branch for routine banking tasks. This saves users time and transportation costs, allowing them to manage their finances efficiently from the comfort of their own homes or any location with an internet connection.
* **Time and Cost Savings**: Bankify eliminates the need for users to physically visit a bank branch for routine banking tasks. This saves users time and transportation costs, allowing them to manage their finances efficiently from the comfort of their own homes or any location with an internet connection.

**2.2 OVERALL DESCRIPTION**

**2.2.1 Identification of Existing Work**

The identification of existing work for the Bankify application involves reviewing and analyzing similar online banking applications currently available in the market. This includes studying their features, user interfaces, security measures, and overall user experience. By examining existing work, Bankify can gain insights into industry best practices, identify areas for improvement, and differentiate itself by offering unique features and a superior user experience. Additionally, it helps ensure that Bankify aligns with regulatory and compliance requirements and takes into account any industry standards or guidelines relevant to online banking systems.

**2.2.2 Product Perspective**

The Bankify application is developed as a standalone online banking system that operates independently from other products or systems. It provides a comprehensive and self-contained platform for users to manage their banking activities. While it may integrate with external services such as payment gateways or investment platforms, Bankify is designed to function as a distinct and self-sufficient product. It aims to offer a seamless and user-friendly online banking experience while ensuring the security and privacy of users' financial information.

**2.2.3 Product Features**

* **Account Management**: Users can create new bank accounts within minutes and manage their existing accounts, including checking balances, viewing transaction history, and updating personal information.
* **Money Transfer**: Users can securely transfer funds between their own accounts, make payments to other Bankify users, and initiate transfers to external bank accounts using standard transfer methods.
* **Loan Management**: Bankify provides a platform for users to apply for loans, view loan details, make loan payments, and track loan repayment schedules.
* **Account Closure**: Users have the option to close their bank accounts through the application, providing a streamlined process to terminate their account when needed.
* **Enhanced Security Measures**: Bankify incorporates robust security measures such as data encryption, two-factor authentication, and real-time fraud detection to safeguard user accounts and transactions.

**2.2.4 End User Characteristics**

* **Diverse Age Groups**: The Bankify application caters to users of different age groups, ranging from young adults to senior citizens, providing a user-friendly interface that accommodates users with varying levels of technical proficiency.
* **Varied Technical Backgrounds**: Bankify is designed to be accessible to users with different levels of technological knowledge, catering to both tech-savvy individuals and those who may have limited experience with online banking.
* **Financial Management Needs**: The application targets users who have various financial management needs, including individuals who seek basic banking services such as money transfers and account management, as well as those interested in advanced financial planning and goal tracking.
* **Security-Conscious Users**: Bankify appeals to users who prioritize the security and privacy of their personal and financial information. These users value robust security measures and are conscious of potential risks associated with online banking.
* **Convenience Seekers**: Bankify targets users who desire the convenience and flexibility of managing their banking activities from anywhere, at any time, without the constraints of physical branch visits.

**CHAPTER 3: PRODUCT FUNCTIONALITY**

**3.1 MODULES**

The Blends Wine Cellar Tracking System is organized into several modules that provide specific functionality. These modules work together to deliver a comprehensive and seamless user experience. The main modules of the system include:

**1. User Management Module:**

- This module handles user registration, login, and authentication.

- It manages user information such as username, password, and email.

- It ensures the security and privacy of user data.

**2. Wineries Module:**

- This module allows users to manage different wine storage places or wineries.

- It stores information about wineries, including their names, countries, regions, and websites.

- Users can associate their wine bottles with specific wineries.

**3. Wine Region Module:**

- This module provides information about the origin or region of wines.

- It stores details such as region names, countries, climates, and soil characteristics.

- Users can associate wines with specific regions for better organization and categorization.

**4. Grape Variety Module:**

- This module relates to the grape varieties used in winemaking.

- It stores information about different grape varieties, including their names, colors,

and percentages.

- Users can associate wines with specific grape varieties to track and categorize their collection.

**5. Wines Module:**

- This module represents the core functionality of managing wine bottles.

- It stores information about individual wines, including their names, wineries, regions, vintage years, prices, alcohol content, and descriptions.

**6. Reviews Module:**

- This module allows users to write reviews or comments about the wines they have stored.

- It stores review text, associated users, and the respective wine ID.

- Users can provide feedback, share their experiences, and rate the wines they have consumed.

**7. Wine Inventory Module:**

- This module enables users to manage their wine inventory.

- It stores information about the quantity of each wine bottle owned by a user, along with

the purchase date.

- Users can add or remove wines from their inventory and track their stock levels.

**8. Wine List Module:**

- This module facilitates the creation and management of custom lists or collections of wines.

- It allows users to create lists, provide descriptions, and associate wines with specific lists.

- Users can organize their wines based on personal preferences, occasions, or any other criteria.

These modules work together to provide users with a comprehensive system for managing and organizing their wine collections. Each module has specific functionalities and interacts with the database to store and retrieve relevant information.

**3.2 External Interface Requirements**

**User Interface:**

The system will have a user-friendly web-based interface that allows users to interact with the application. The user interface will be designed to provide an intuitive and seamless experience, making it easy for users to navigate, input data, and access various features.

**Hardware Interface:**

- Server: The system will require a server to host the Blends application and handle user requests and data processing.

- Client: Users will access the system through web browsers on their personal computers, laptops, or mobile devices.

**Software Interface:**

The system will interact with various software components, including:

- Web browsers: The system will be compatible with popular web browsers such as Chrome, Firefox, and Safari.

- PHP: The backend of the system will be implemented using PHP for server-side processing and data retrieval.

- MySQL: The system will utilize the MySQL database management system for efficient storage and retrieval of wine-related data.

**3.3 Other Non-Functional Requirements**

Performance Requirements:

- The system should provide fast response times to ensure a smooth user experience.

- The database queries and operations should be optimized to handle large volumes of data efficiently.

- The system should be able to handle multiple concurrent users without significant

performance degradation.

**Safety Requirements:**

- The system should implement appropriate security measures to protect user data and prevent unauthorized access.

- User authentication and authorization mechanisms should be in place to ensure secure access to the system.

**Software Quality Requirements:**

- The system should be reliable and robust, with minimal errors or failures.

- The codebase should follow good coding practices, be well-structured, and maintainable.

**Operating Environment:**

- Hardware: The system should be compatible with standard hardware configurations, including personal computers, laptops, and mobile devices with internet connectivity.

- Software: The system should be compatible with widely used operating systems (Windows, macOS, Linux) and web browsers (Chrome, Firefox, Safari).

**CHAPTER 4: SYSTEM DESIGN**

**4.1 ER Design**

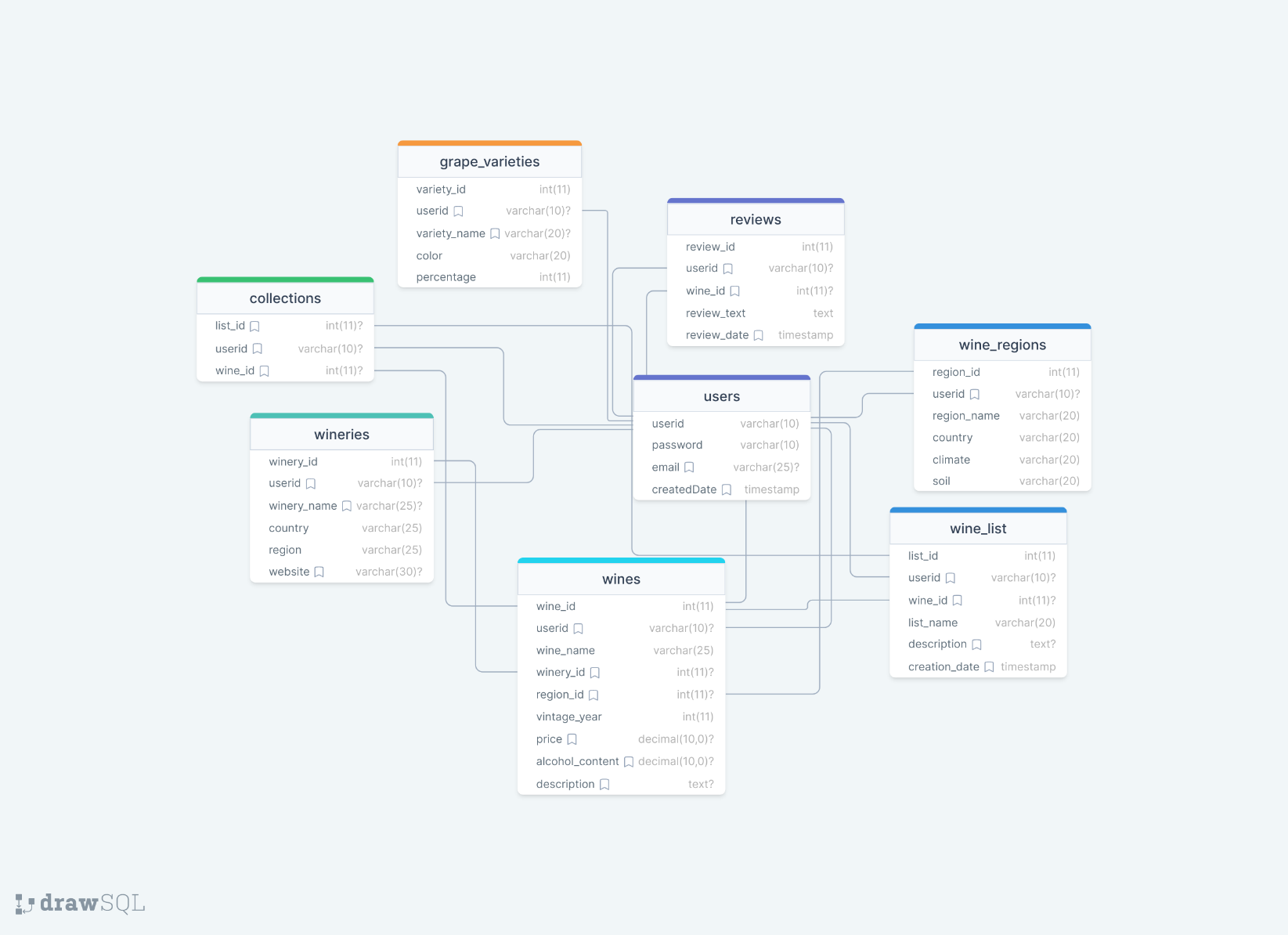
In the Blends system, the ER design will capture the key entities involved in wine collection management, such as users, wineries, wine regions, grape varieties, wines, reviews, wine inventory, and wine lists. Each entity will have its attributes that represent the specific information associated with it.

**CHAPTER 5: DETAILED DESIGN**

**5.1 Schema Design**

The schema design is a critical step in the development of the Blends Wine Cellar Tracking System. It involves translating the Entity-Relationship (ER) design into a detailed and structured database schema. The schema design defines the tables, columns, data types, and relationships that form the foundation of the database.

Here is the schema design for the Blends system, based on the entities and relationships identified in the ER design:



Schema Diagram for Blends Web Application

**CHAPTER 6: IMPLEMENTATION**

**6.1 Introduction**

The implementation of the Blends Wine Cellar Tracking System involves the creation and manipulation of the database using MySQL. This chapter provides an overview of the implementation process and covers the details of various operations such as CREATE, INSERT, UPDATE, ALTER, DROP, DELETE, and SELECT.

**6.2 Implementation Details**

**6.2.1 CREATE**

**Creating the user table:**

create table users(

userid varchar(10) primary key,

password varchar(10) not null,

email varchar(25) unique,

createdDate timestamp);

**Creating the wineries table:**

create table wineries(

winery\_id int AUTO\_INCREMENT Primary Key,

userid varchar(10) references users(userid),

winery\_name varchar(25) unique,

country varchar(25) not null,

region varchar(25) not null,

website varchar(30)

);

**Creating the regions table:**

table wine\_regions(

region\_id int AUTO\_INCREMENT primary key,

userid varchar(10) references users(userid),

region\_name varchar(20) not null,

country varchar(20) not null,

climate varchar(20) not null,

soil varchar(20) not null);

**Creating the grape varieties table:**

create table grape\_varieties(

variety\_id int AUTO\_INCREMENT primary key,

userid varchar(10) references users(userid),

variety\_name varchar(20) unique,

color varchar(20) not null,

percentage int not null);

**Creating the wines table:**

create table wines(

wine\_id int AUTO\_INCREMENT primary key,

userid varchar(10) references users(userid),

wine\_name varchar(25) not null,

winery\_id int references wineries(winery\_id),

region\_id int references wine\_regions(region\_id),

vintage\_year int not null,

price decimal,

alcohol\_content decimal,

description text);

**Creating the reviews table:**

create table reviews(

review\_id int AUTO\_INCREMENT primary key,

userid varchar(10) references users(userid),

wine\_id int references wines(wine\_id),

review\_text text not null,

review\_date timestamp not null);

**Creating a wine list table:**

create table wine\_list(

list\_id int AUTO\_INCREMENT primary key,

userid varchar(10) references users(userid),

wine\_id int references wines(wine\_id),

listname varchar(20) not null,

description text,

creation\_date timestamp);

**Creating a wine list table:**

create table collections(

list\_id int references wine\_list(list\_id),

userid varchar(10) references users(userid),

wine\_id int references wines(wine\_id));

**6.2.2 INSERT**

**1. Insert Query for the `users` table:**

INSERT INTO users (userid, password, email, createdDate)

VALUES ('user123', 'password123', 'user123@example.com', NOW());

**2. Insert Query for the `wineries` table:**

INSERT INTO wineries (userid, winery\_name, country, region, website)

VALUES ('user123', 'Winery A', 'Country A', 'Region A', 'www.winerya.com');

**3. Insert Query for the `wine\_regions` table:**

INSERT INTO wine\_regions (userid, region\_name, country, climate, soil)

VALUES ('user123', 'Region A', 'Country A', 'Climate A', 'Soil A');

**4. Insert Query for the `grape\_varieties` table:**

INSERT INTO grape\_varieties (userid, variety\_name, color, percentage)

VALUES ('user123', 'Variety A', 'Red', 80);

**5. Insert Query for the `wines` table:**

INSERT INTO wines (userid, wine\_name, winery\_id, region\_id, vintage\_year, price, alcohol\_content, description)

VALUES ('user123', 'Wine A', 1, 1, 2020, 29.99, 13.5, 'This is a great red wine.');

**6. Insert Query for the `reviews` table:**

INSERT INTO reviews (userid, wine\_id, review\_text, review\_date)

VALUES ('user123', 1, 'This wine has a rich flavor profile.', NOW());

**8. Insert Query for the `wine\_list` table:**

INSERT INTO wine\_list (userid, wine\_id, list\_name, description, creation\_date)

VALUES ('user123', 1, 'Favorites', 'My favorite wines', NOW());

**9. Insert Query for the `collections` table:**

INSERT INTO collections (list\_id, userid, wine\_id)

VALUES (1, 'user123', 1);

**6.2.3 UPDATE**

The UPDATE operation is used to modify existing data in the tables. Here is an example of the UPDATE statement for the 'wines' table:

```

UPDATE wines

SET price = 29.99

WHERE wine\_id = 1;

```

**6.2.4 ALTER**

The ALTER operation is used to modify the structure of the tables. Here is an example of the ALTER statement to add a new column 'rating' to the 'reviews' table:

```

ALTER TABLE reviews

ADD COLUMN rating int;

```

**6.2.5 DROP**

The DROP operation is used to remove tables or columns from the database. Here is an example of the DROP statement to delete the 'wine\_regions' table:

```

DROP TABLE wine\_regions;

```

**6.2.6 DELETE**

The DELETE operation is used to remove data from the tables. Here is an example of the DELETE statement to delete a review from the 'reviews' table:

```

DELETE FROM reviews

WHERE review\_id = 1;

```

**6.2.7 SELECT**

The SELECT operation is used to retrieve data from the tables. Here is an example of the SELECT statement to retrieve all wines from the 'wines' table:

```

SELECT \* FROM wines;

```

**6.2.9 KEY CONSTRAINTS**

Key constraints ensure data integrity and enforce uniqueness and relationships within the database. Here are examples of key constraints that can be implemented:

**Primary Key Constraint:**

**```**

CREATE TABLE wineries (

winery\_id int AUTO\_INCREMENT PRIMARY KEY,

winery\_name varchar(25) NOT NULL,

country varchar(25) NOT NULL

);

```

**Foreign Key Constraint:**

```

CREATE TABLE wines (

wine\_id int AUTO\_INCREMENT PRIMARY KEY,

winery\_id int,

region\_id int,

wine\_name varchar(25) NOT NULL,

FOREIGN KEY (winery\_id) REFERENCES wineries(winery\_id),

FOREIGN KEY (region\_id) REFERENCES wine\_regions(region\_id)

);

```

**6.2.10 ATTRIBUTE CONSTRAINTS**

Attribute constraints define rules and restrictions for individual attribute values. Here are examples of attribute constraints:

NOT NULL Constraint:

```

CREATE TABLE users (

userid varchar(10) PRIMARY KEY,

password varchar(10) NOT NULL,

email varchar(25) UNIQUE,

createdDate timestamp

);

**CHAPTER 7: SOFTWARE TESTING**

**7.1 Introduction**

Software testing is a crucial phase in the software development lifecycle. It ensures that the implemented system functions correctly, meets the requirements, and operates smoothly. This chapter focuses on the testing process for the Blends Wine Cellar Tracking System.

**7.2 Testing Objective**

The primary objective of software testing is to identify defects, errors, and deviations from the expected behavior of the system. The testing process aims to ensure the reliability, functionality, and quality of the software. For the Blends Wine Cellar Tracking System, the testing objectives include:

-Validate the functionality of each module.

-Verify that the system meets the specified requirements.

-Identify and fix any defects or issues.

-Ensure that the system operates smoothly and efficiently.

**7.2.1 Test Cases for Each Module**

To ensure comprehensive testing, test cases need to be designed for each module of the Blends Wine Cellar Tracking System. Test cases are scenarios or situations that are executed to validate the behavior and performance of the system.

**1. Users Module:**

- Test case 1: Verify that a user can successfully register with valid credentials.

- Test case 2: Validate that the system prevents registration with an already existing email address.

- Test case 3: Ensure that the system enforces password strength requirements during registration.

- Test case 4: Validate that the system displays an error message for an incorrect login attempt.

**2. Wineries Module:**

- Test case 1: Validate that a user can add a new winery with all the required information.

- Test case 2: Verify that the system prevents the addition of a winery with a duplicate name.

- Test case 3: Ensure that the system properly associates the winery with the corresponding user.

- Test case 4: Validate that a user can edit the details of an existing winery.

**3. Wine Regions Module:**

- Test case 1: Validate that a user can add a new wine region with accurate details.

- Test case 2: Verify that the system prevents the addition of a wine region with duplicate name

and country.

- Test case 3: Ensure that the system properly associates the wine region with the corresponding user.

- Test case 4: Validate that a user can edit the details of an existing wine region.

- Test case 5: Verify that the system allows a user to delete a wine region.

**4. Grape Varieties Module:**

- Test case 1: Validate that a user can add a new grape variety with accurate details.

- Test case 2: Verify that the system prevents the addition of a duplicate grape variety.

- Test case 3: Ensure that the system properly associates the grape variety with the corresponding user.

- Test case 4: Validate that a user can edit the details of an existing grape variety.

- Test case 5: Verify that the system allows a user to delete a grape variety.

**5. Wines Module:**

- Test case 1: Verify that a user can add a new wine to their collection with accurate details.

- Test case 2: Validate that the system calculates and stores the alcohol content of the wine correctly.

- Test case 3: Ensure that the system properly associates the wine with the corresponding winery and region.

- Test case 4: Validate that a user can edit the details of an existing wine.

- Test case 5: Verify that the system allows a user to delete a wine from their collection.

**6. Reviews Module:**

- Test case 1: Verify that a user can write a review for a specific wine in their collection.

- Test case 2: Validate that the system stores the review text and associates it with the corresponding user and wine.

- Test case 3: Ensure that the system displays the reviews correctly for a wine.

**7. Wine Inventory Module:**

- Test case 1: Validate that a user can add a wine to their inventory with the quantity and purchase date.

- Test case 2: Verify that the system properly updates the inventory when a wine is added.

- Test case 3: Ensure that the system allows a user to update the quantity of a wine in their inventory.

- Test case 4: Validate that a user can remove a wine from their inventory.

- Test case 5: Verify that the system properly adjusts the inventory when a wine is removed.

**8. Wine List Module:**

- Test case 1: Verify that a user can create a new wine list with a unique name.

- Test case 2: Validate that the system prevents the creation of a wine list with a duplicate name.

- Test case 3: Ensure that the system associates the wine list with the corresponding user.

- Test case 4: Validate that a user can add wines to a wine list.

- Test case 5: Verify that the system allows a user to remove wines from a wine list.

**CHAPTER 8: CONCLUSION**

The Blends Wine Cellar Tracking System has been successfully implemented, providing wine enthusiasts with a centralized platform for managing and organizing their wine collections. Throughout the project, we have covered various aspects, including the project description, tools and technology used, software requirement specification, system design, implementation, and software testing.The project aimed to develop a digital platform that simplifies the management and organization of wine collections. By creating a user-friendly interface and implementing essential features such as wine storage management, wine region information, grape variety details, and wine reviews, the system provides users with a comprehensive tool for tracking and exploring their wine collections.

The implementation phase involved the utilization of HTML, CSS, JavaScript for the frontend, PHP for the backend, and MySQL for the database. These technologies were chosen for their compatibility, scalability, and ability to handle the required functionalities of the system.Throughout the project, we followed a systematic approach, including requirement gathering, system design, database schema design, implementation of modules, and rigorous testing. This ensured that the system meets the desired objectives, is user-friendly, and operates smoothly.

The testing phase involved the creation and execution of test cases for each module, covering various scenarios and functionalities. The goal was to identify and address any bugs, errors, or inconsistencies in the system, ensuring its reliability and performance.the Blends Wine Cellar Tracking System is a valuable tool for wine collectors to manage and organize their collections effectively. It provides an intuitive user interface, robust backend functionality, and a secure database for storing and retrieving wine-related information. The system enhances the wine collecting experience by enabling users to make informed decisions, discover new wine pairings, and enjoy their wine collections to the fullest.

Overall, the project has been successfully completed, meeting the project objectives and delivering a reliable and efficient Wine Cellar Tracking System. Further improvements and enhancements can be made based on user feedback and evolving requirements to ensure the system's continued success and usefulness in the wine collecting community.

**CHAPTER 9: FUTURE ENHANCEMENTS**

The Blends Wine Cellar Tracking System has provided wine collectors with a powerful tool for managing and organizing their wine collections. As technology and user needs continue to evolve, there are several potential future enhancements that can further improve the system's functionality and user experience. Here are some ideas for future enhancements:

**1. Mobile Application:** Develop a mobile application for iOS and Android devices to provide users with the convenience of accessing and managing their wine collections on the go. The mobile app can offer features such as barcode scanning for adding new wines, push notifications for wine consumption reminders, and a seamless user interface optimized for mobile devices.

**2. Social Features**: Introduce social features within the system to enhance user interaction and engagement. Users can connect with fellow wine enthusiasts, share their collections and reviews, and explore wines recommended by others. This social aspect can create a vibrant community within the platform, fostering knowledge exchange and promoting discovery.

**3. Integration with Wine Databases:** Collaborate with established wine databases to integrate their vast repositories of wine information into the Blends system. This integration would provide users with comprehensive and up-to-date wine details, including ratings, expert reviews, and additional tasting notes, enriching their understanding and appreciation of the wines in their collection.

**4. Advanced Search and Filtering:** Enhance the search and filtering capabilities of the system to allow users to easily find wines based on specific criteria, such as grape variety, region, vintage, or price range. Implementing advanced search options and filters can help users quickly locate specific wines within their collection or discover new wines based on their preferences.

**5. Wine Recommendations and Pairings:** Develop an intelligent recommendation engine that suggests wines and wine pairings based on user preferences, past purchases, and reviews. By leveraging machine learning algorithms and user data, the system can provide personalized recommendations, helping users explore new wines and discover perfect pairings for their meals or occasions.

**6. Integration with E-Commerce Platforms:** Enable seamless integration with e-commerce platforms to allow users to purchase wines directly from within the Blends system. This feature would streamline the wine buying process, providing users with easy access to their preferred wines and enhancing the overall convenience of managing their collection.

These future enhancements would further elevate the Blends Wine Cellar Tracking System, providing users with advanced features, personalized experiences, and a robust ecosystem to enhance their wine collecting journey. By staying at the forefront of technology and incorporating user feedback, the system can continue to evolve and meet the ever-changing needs of wine enthusiasts

**APPENDIX**

**APPENDIX A:- REFERENCES**

**Appendix B: USER MANUAL**

1. The bellow image1 shows the starting page or the home page interface of Blends web application.

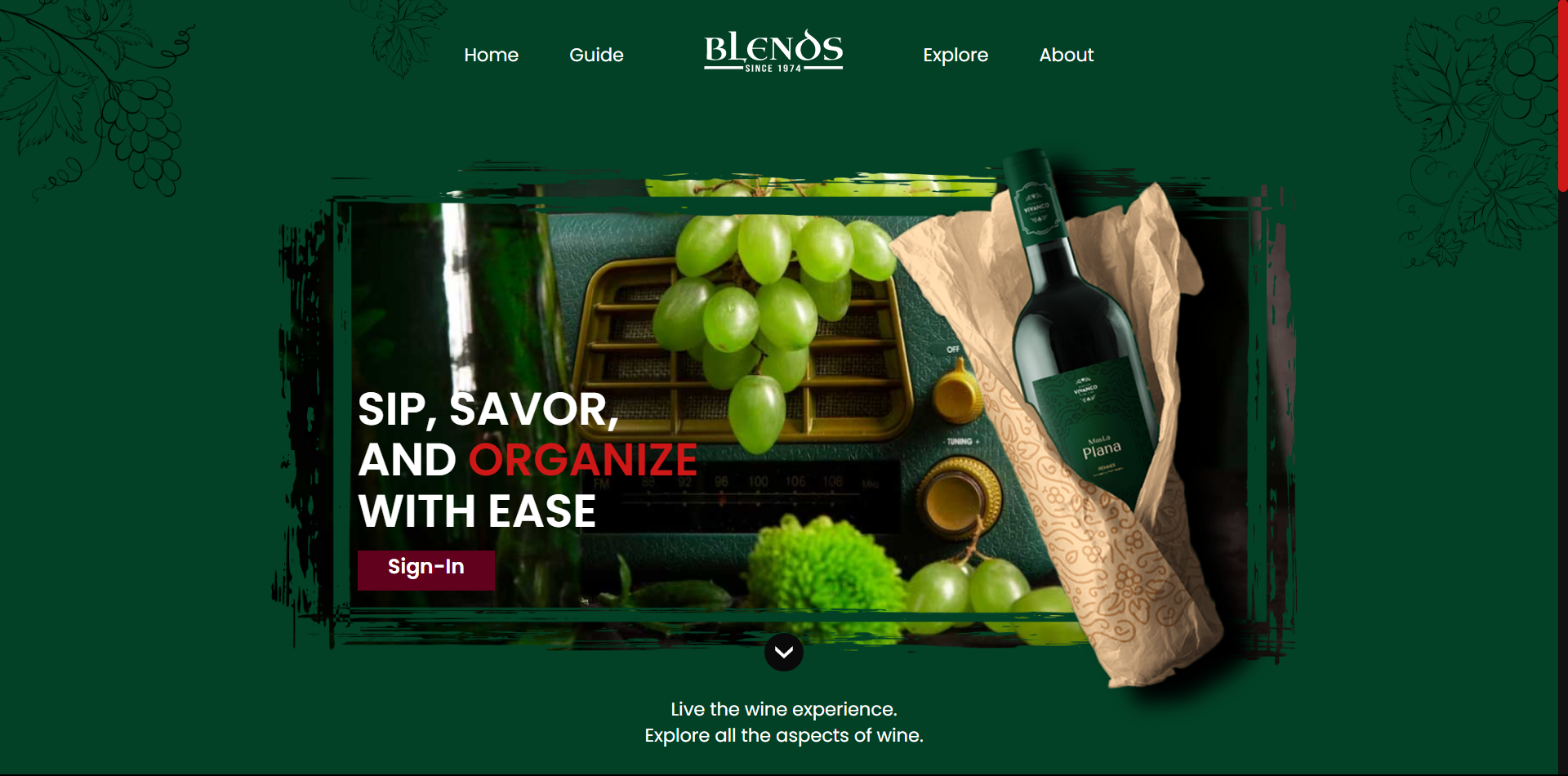


Image 1 :Home Page

2. The below image 2 shows the form, where you can register to the Blends application.



Image 2: Register page

3.After registering then user can enter the username and password to enter the application.

The below image 3 shows the sign in page.



Image 3: Sign-In page

4.After successful sign in, the user will get a dashboard page as shown in the Image 4. Where

user can directly see how many numbers of bottles he/she has. And what are the variety of

grapes, and number of wineries where the user stored his/her wines.

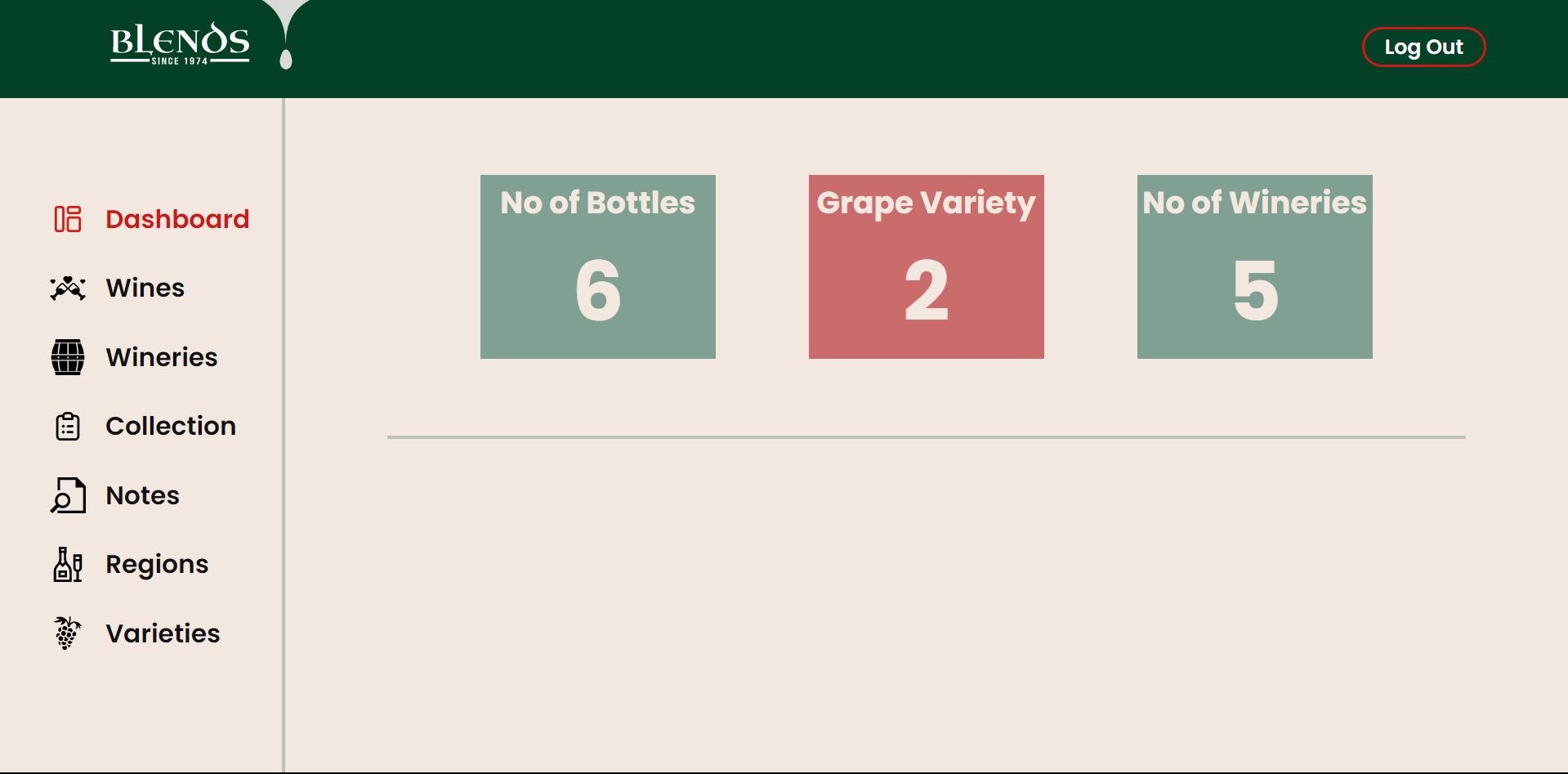


Image 4: Dashboard page

5. The Image 5 and Image 6 shows the Winery details and Adding wineries respectively. User

Can add the details of the winery by clicking on add winery button. The button is Shown in

the Image 5.

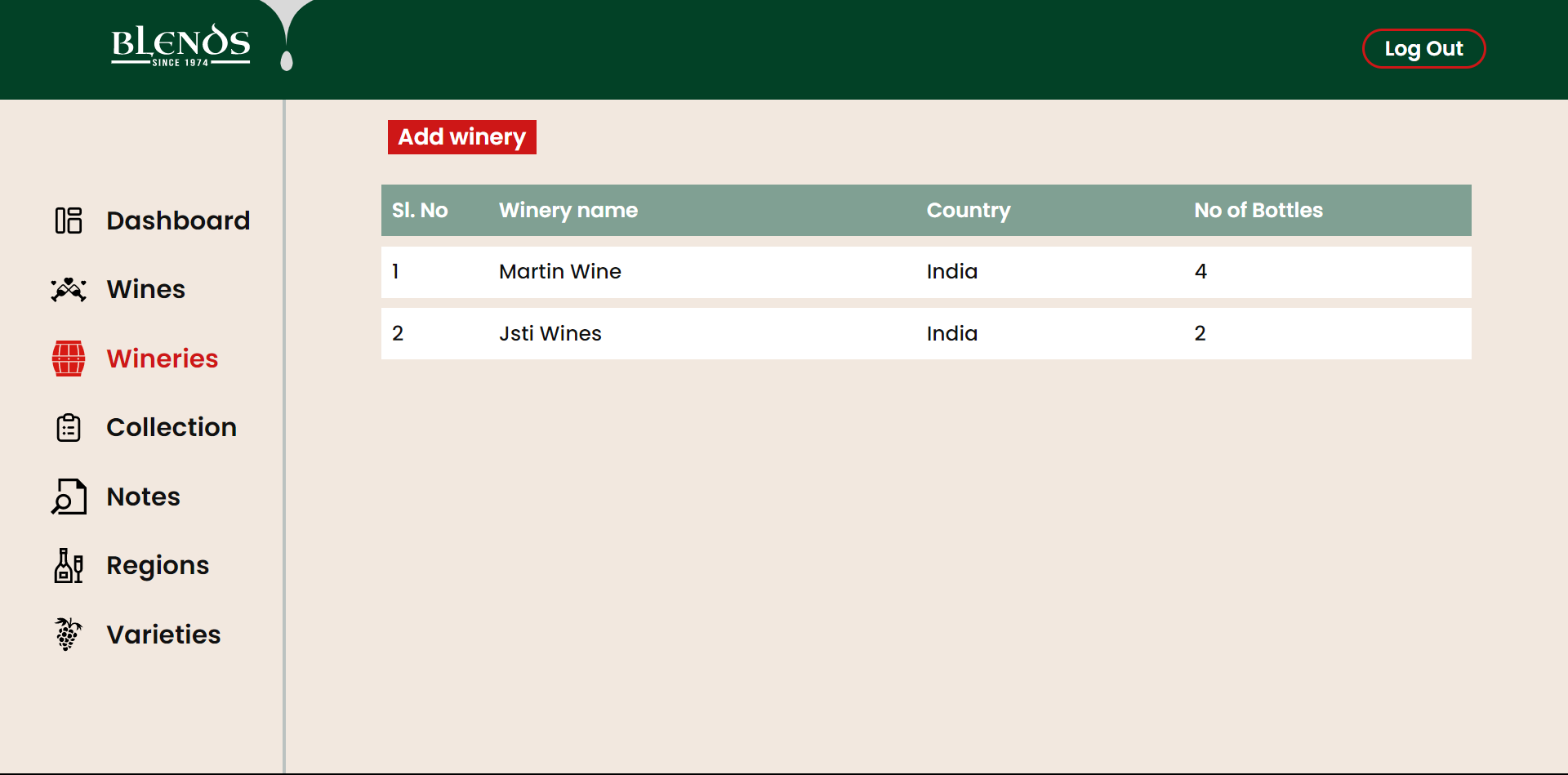


Image 5: Wineries page

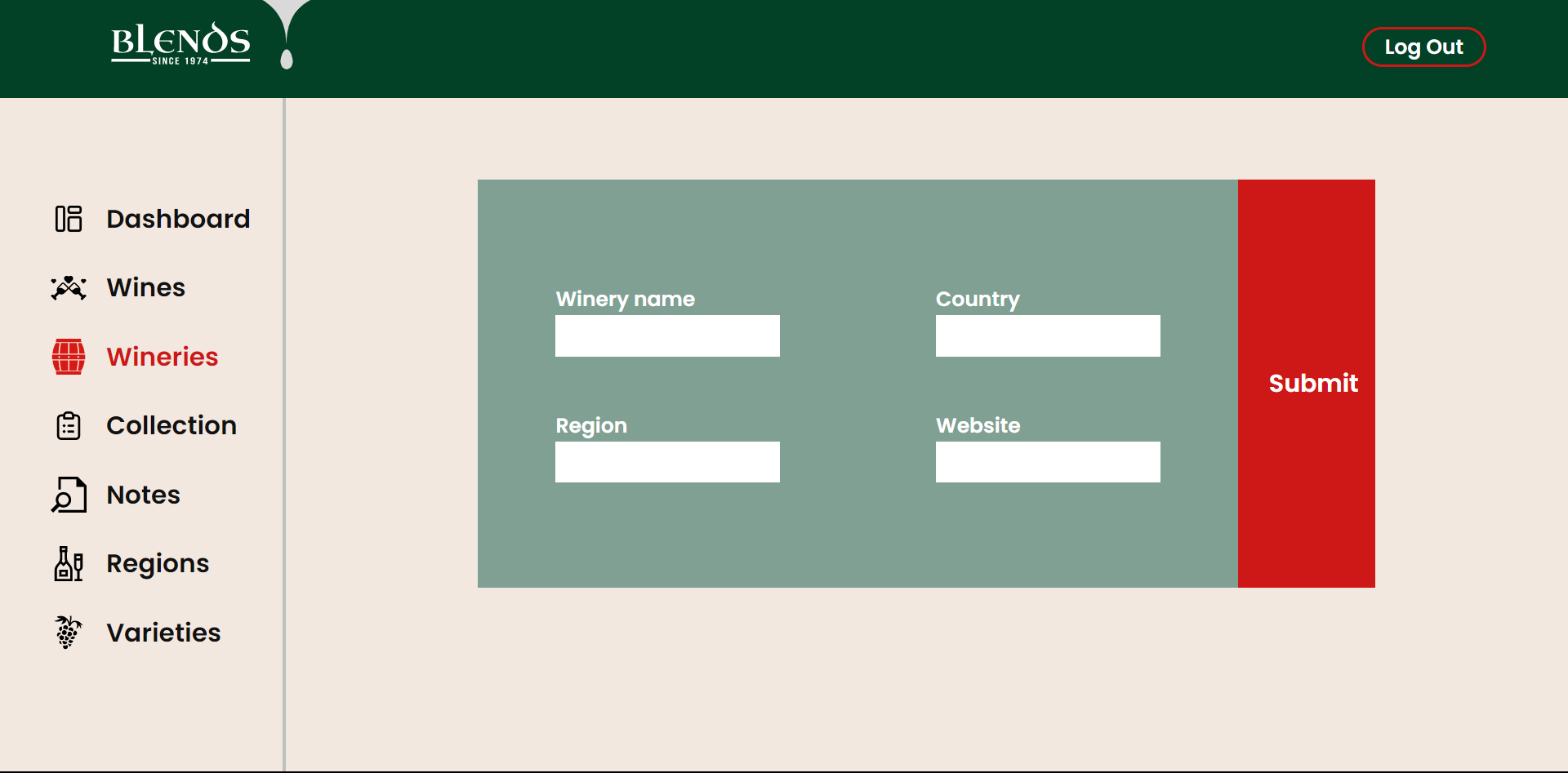


Image 6: Adding Wineries

6. The Image 7 shows where the user can see his or her collections. And the Image 8 shows the

web page where the user can add his/her collections. When user clicks on the create collections

button the user will be redirected to the adding collection page as shown in the image 8.

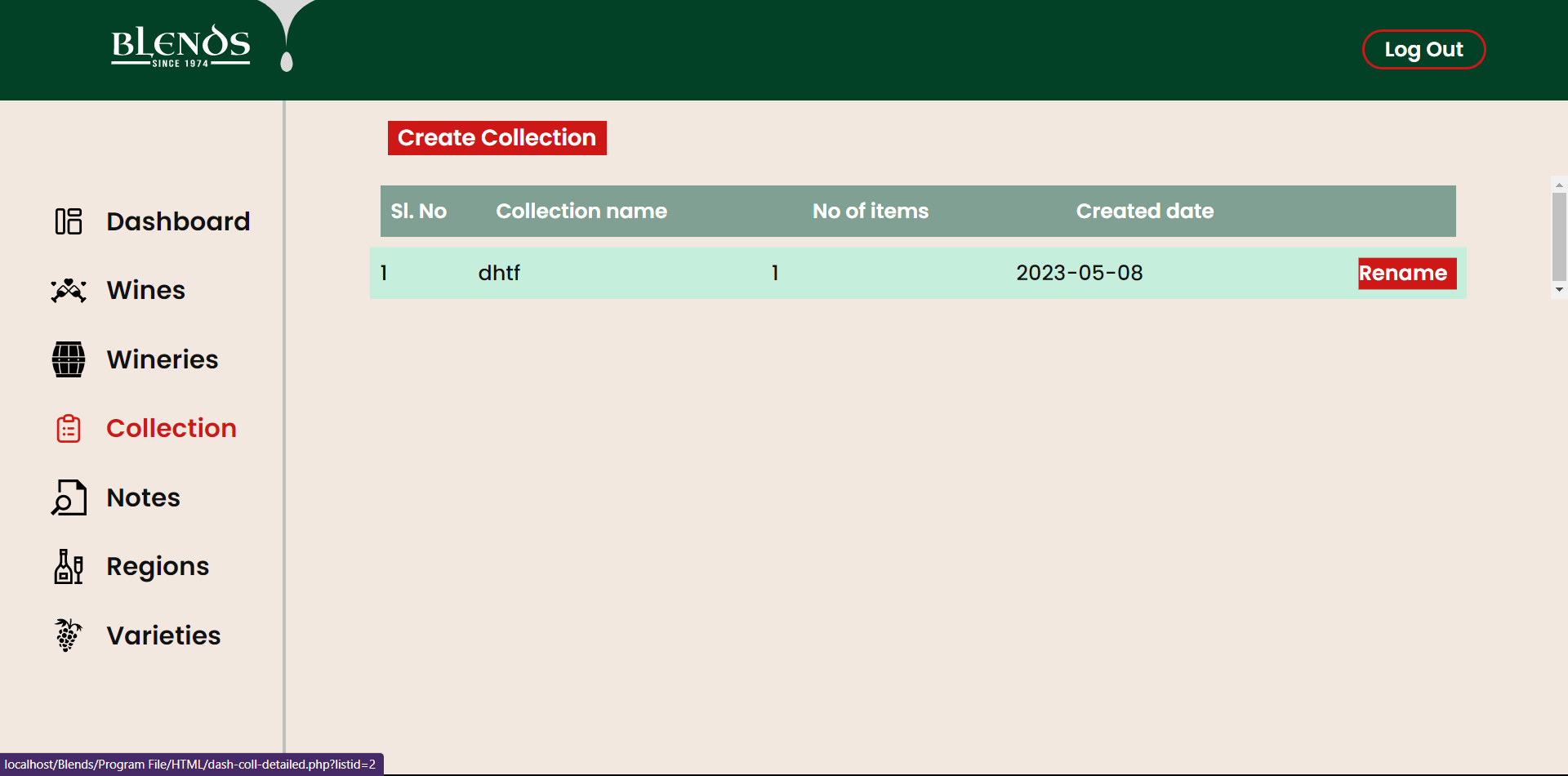


Image 7 : Collection page



Image 8: Adding Collections

7. when the user clicks on the region, the user will be redirected to the region page where the use

can see the regions. The Image 9 shows the interface of the region page. And If the user wants to add

region to the list, the user can click on the add region button and it will be redirected to the add region

page where the user can add the region. The image 10 shows the add region page.

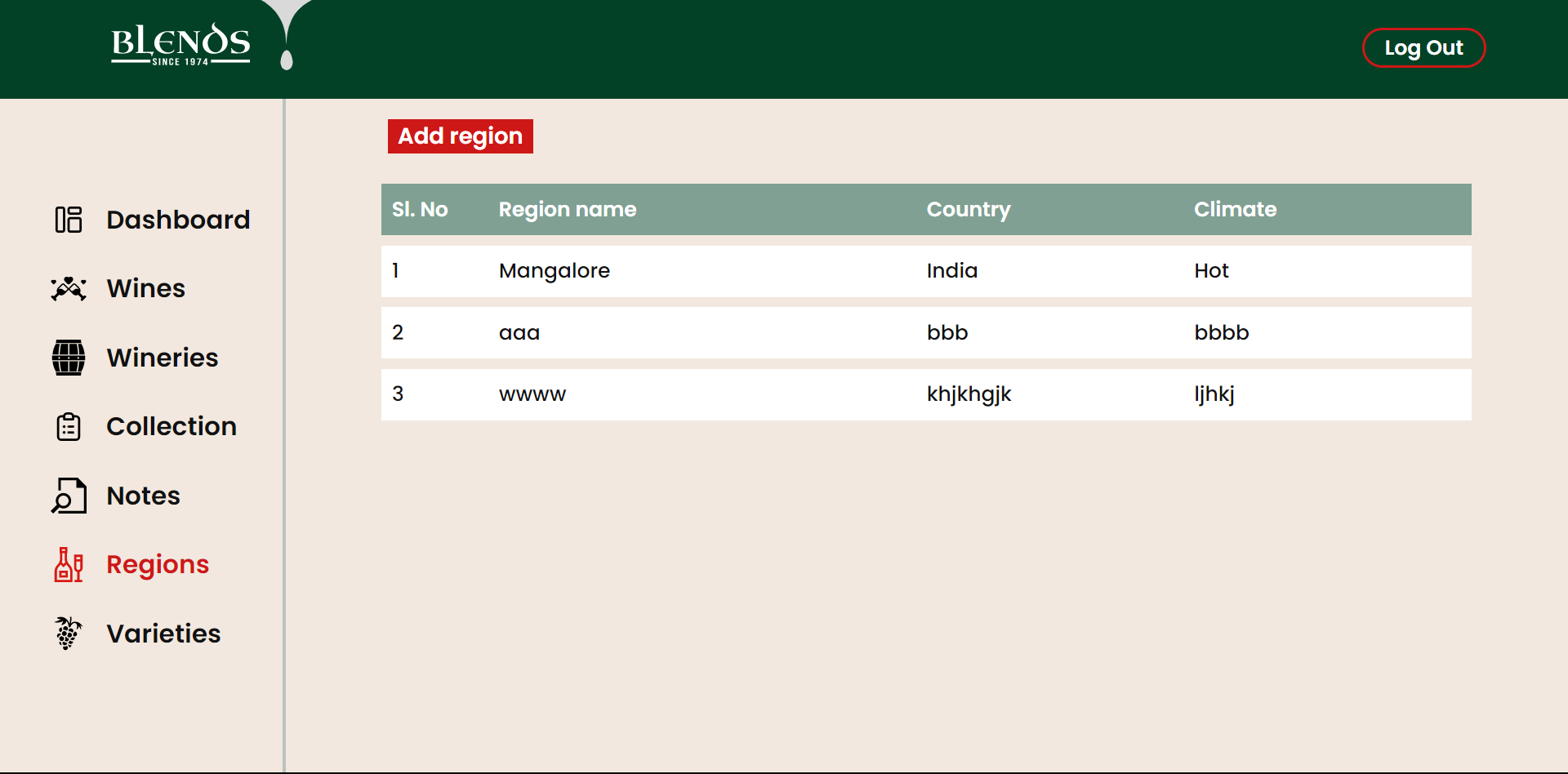


Image 9: region page

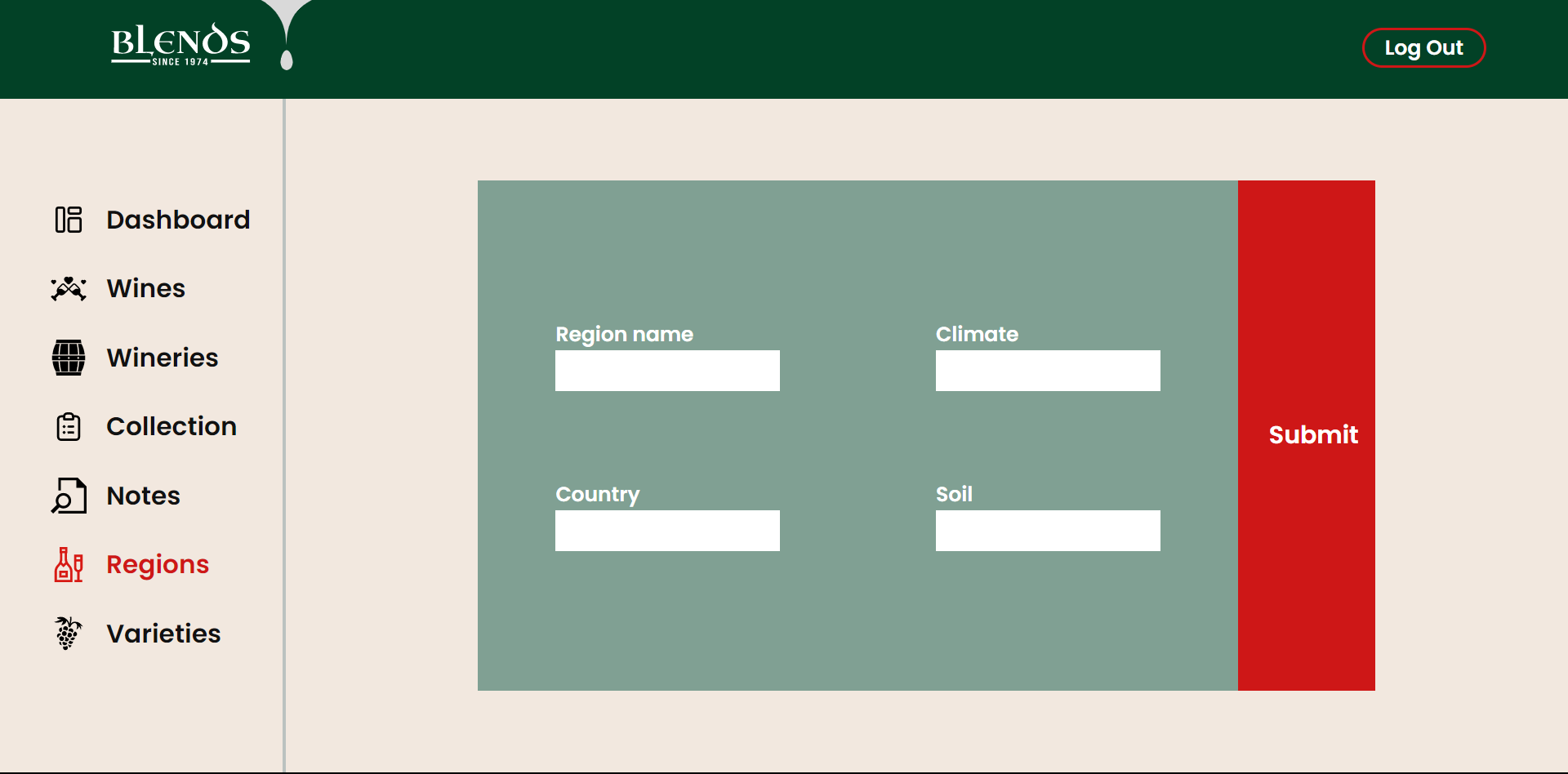


Image 10: Add region page

8. When user clicks in the notes, the user will see the notes which added. The Image 11 shows the

notes page where all the notes about the wines are displayed. And If the user wants to add notes

about a wine the user can click on the add notes button which will be in the same page of notes list,

and then the user can add the notes about a particular wine. The image 12 shows the page of adding

the notes.

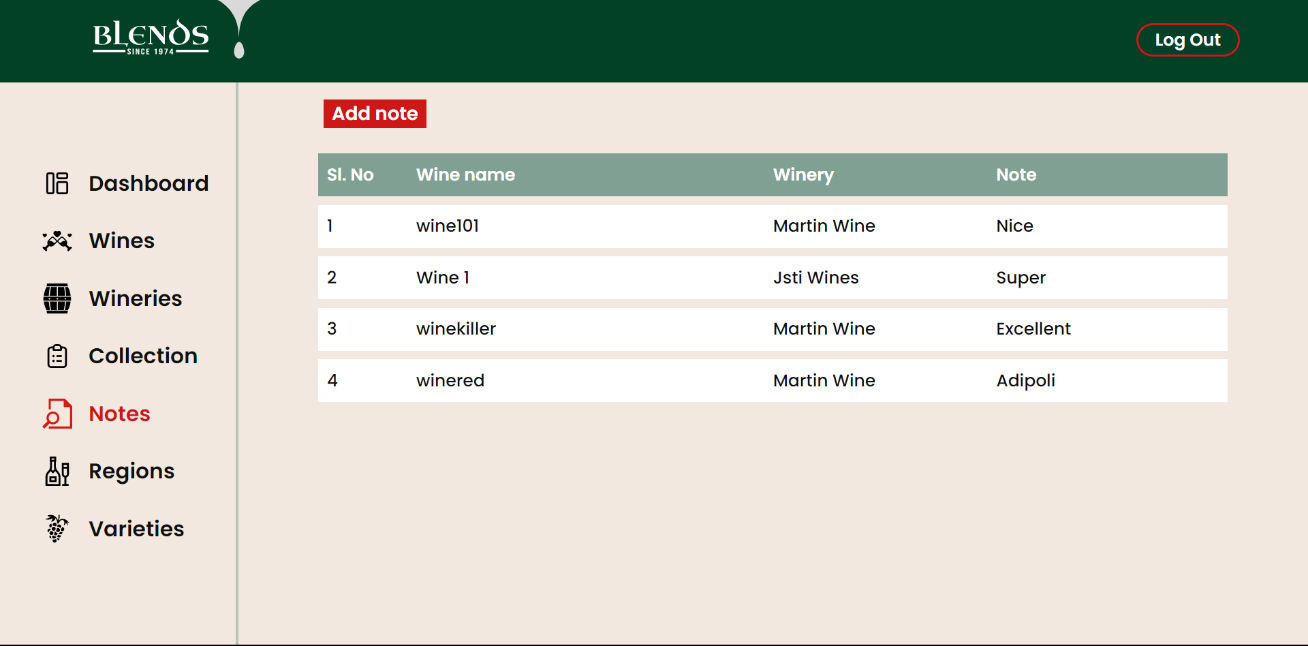


Image 11: Notes page

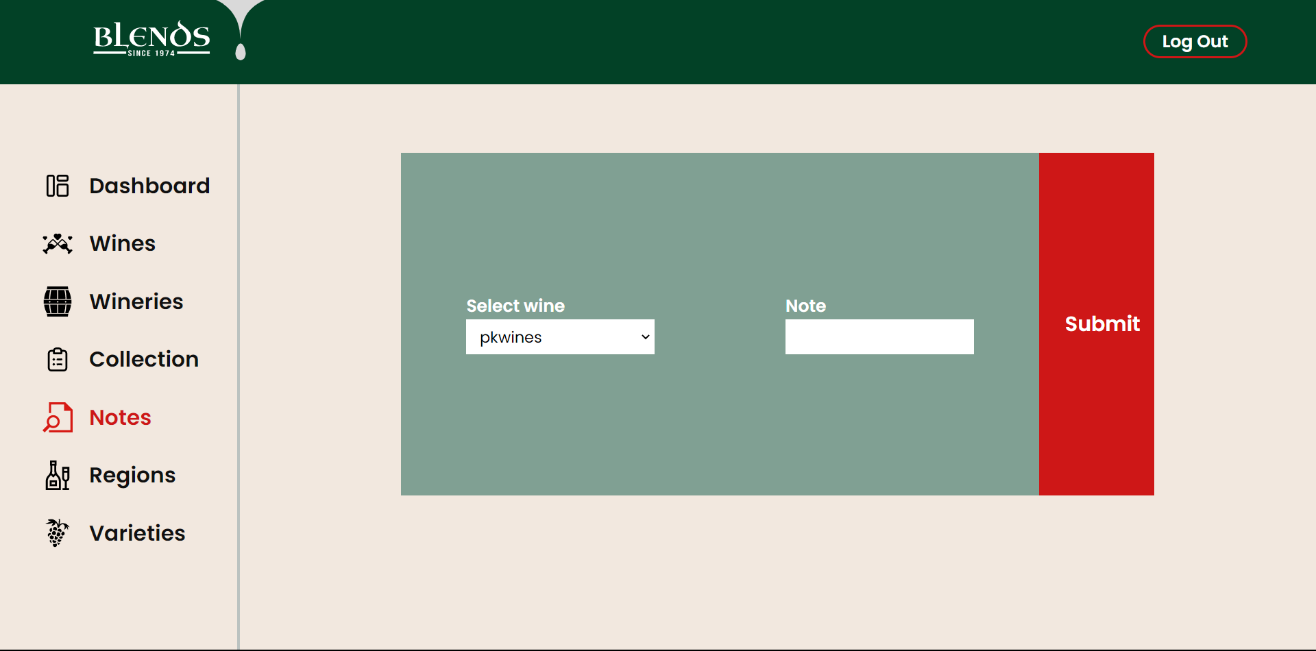


Image 12: Add notes page

9.If the user wants to add the varieties, the user can click on the varieties and then the user can see the variety that the user added. the image 13 shows the page where the list of varieties. If the user wants to add, then just click on the create variety and fill the form and click on submit. Then that list will be added to the list of varieties. The 14 shows the add variety page.

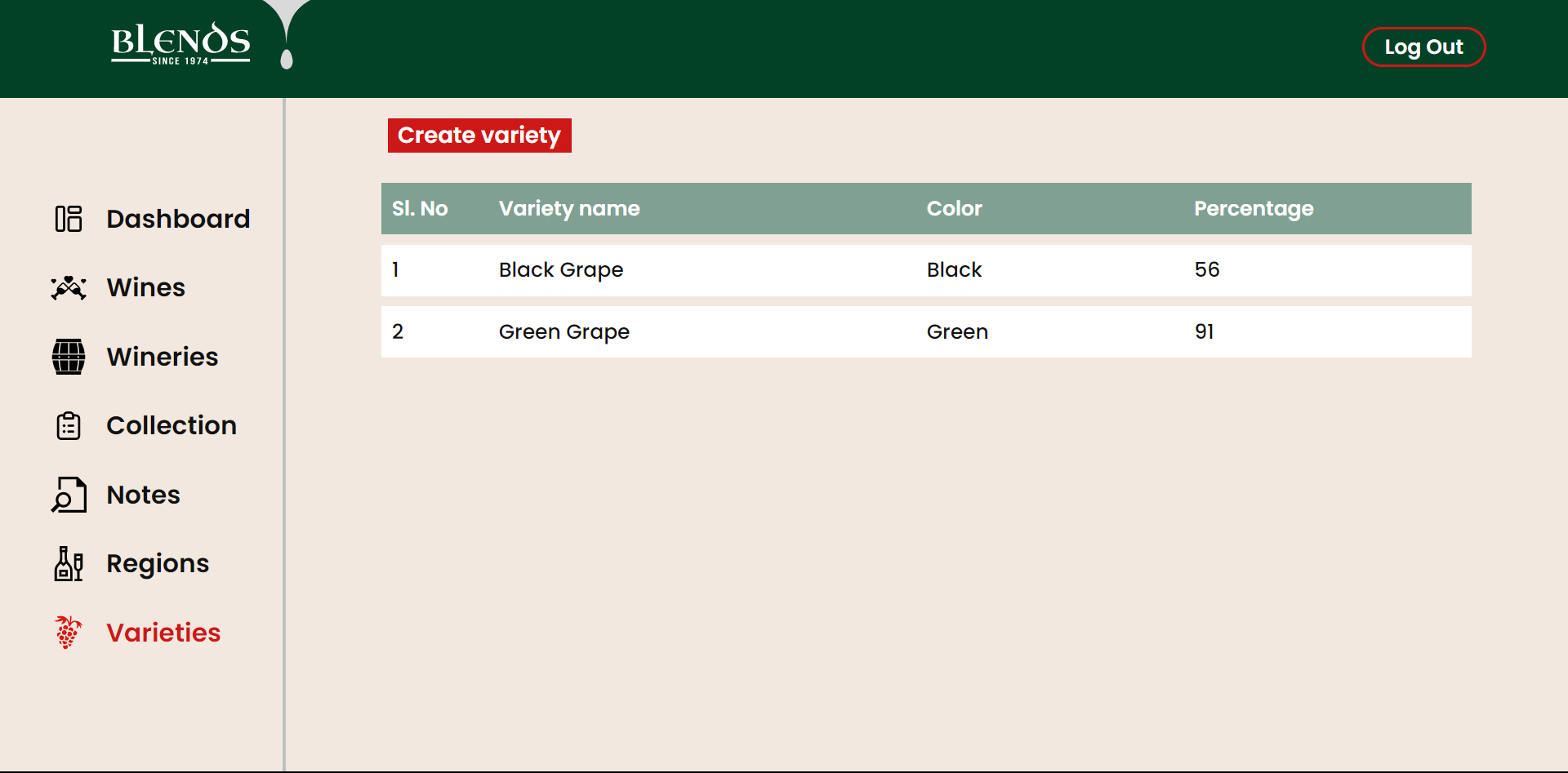


Image 13: variety page

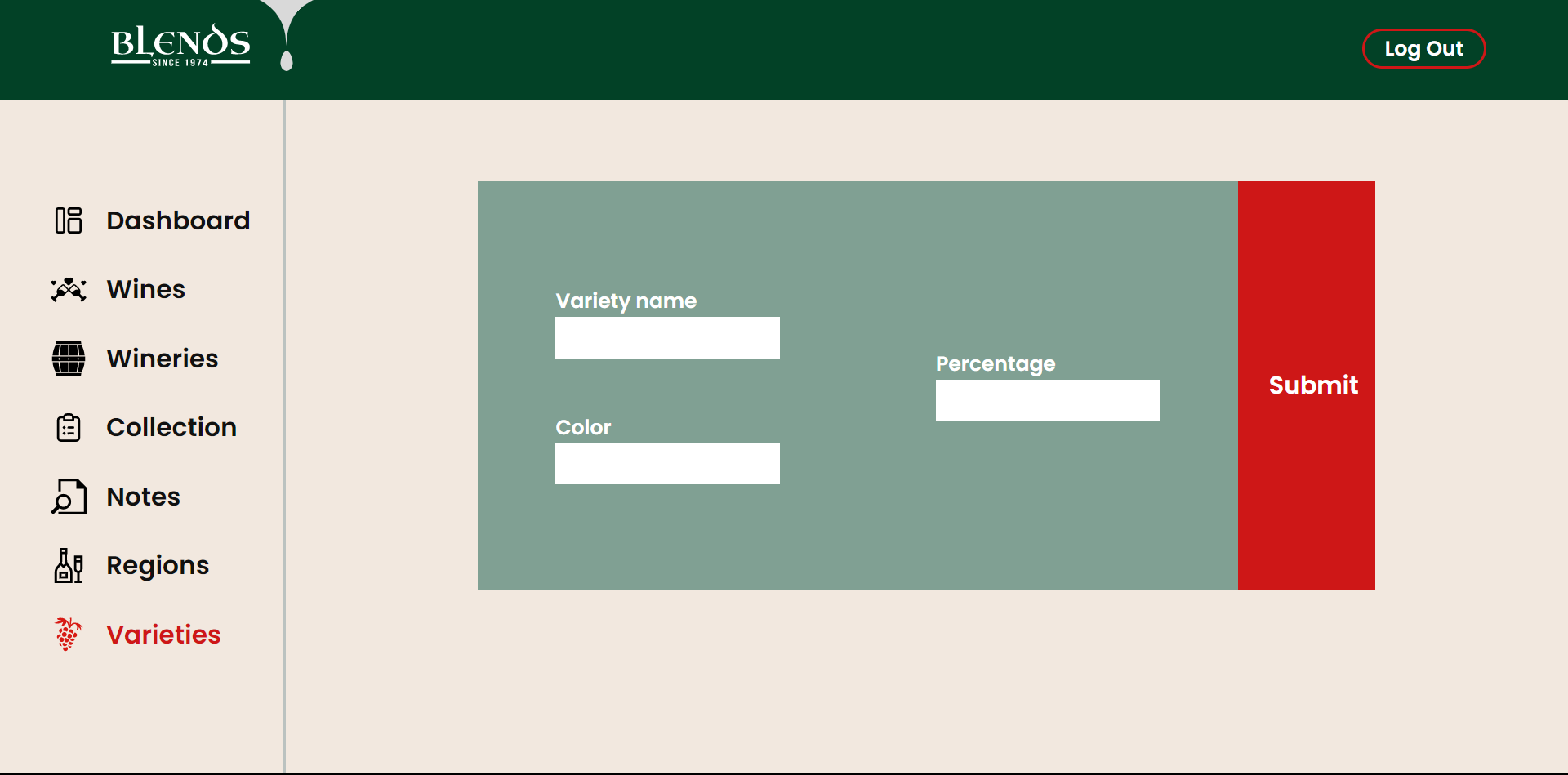


Image 14: add variety page

10. And the main thing about wines, After all these steps then user will add the wines to the list. The

user needs to press on the wines option. Then the user needs the press on the add wines button. Then

the form will be displayed, fill the form and press on submit. Next the wine information will be

displayed. The image 15 shows the page where the list of wines will be displayed. And the image 16

shows the form where you can add the wine and press the submit button.

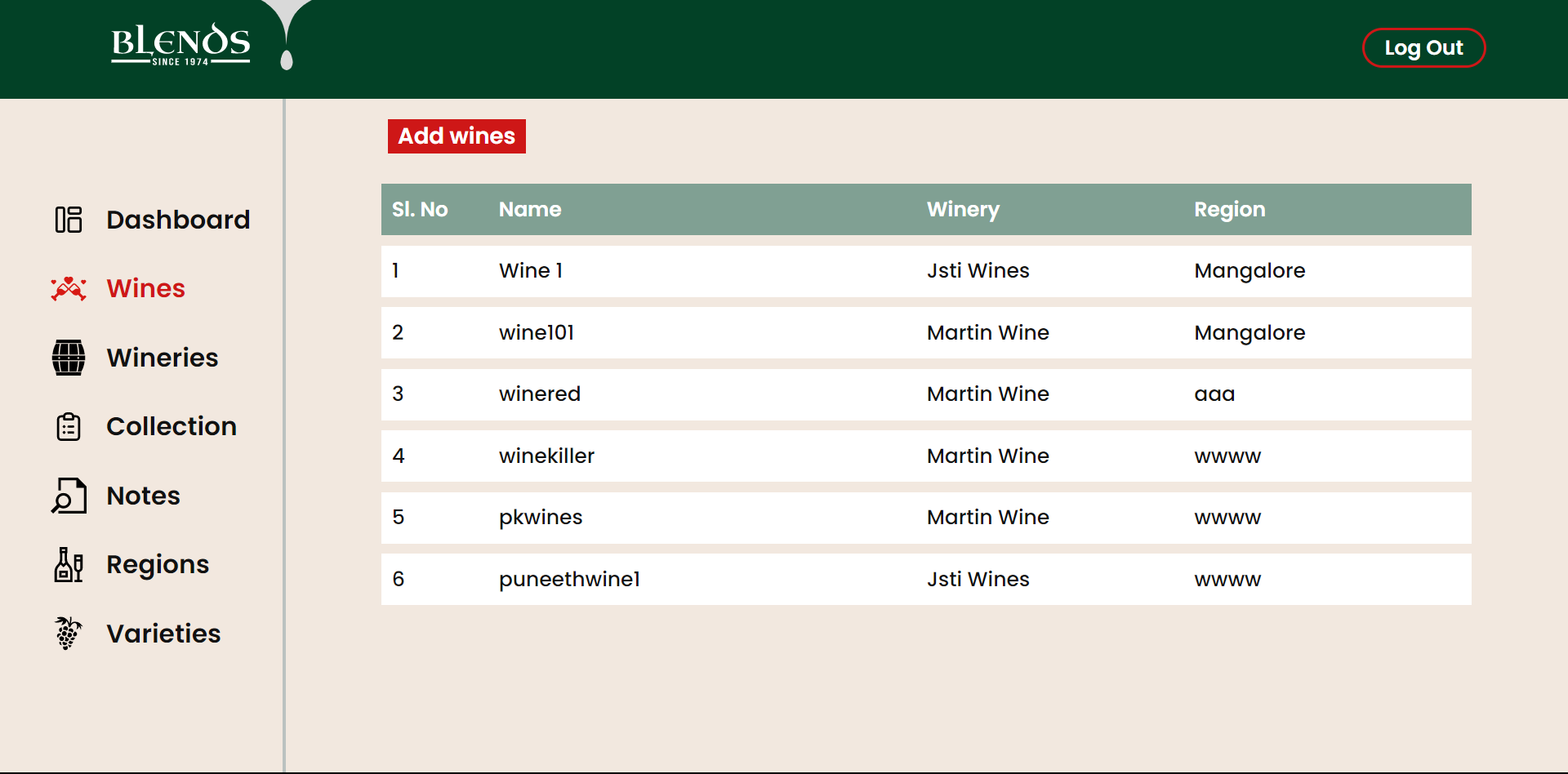


Image 15: Wine list page

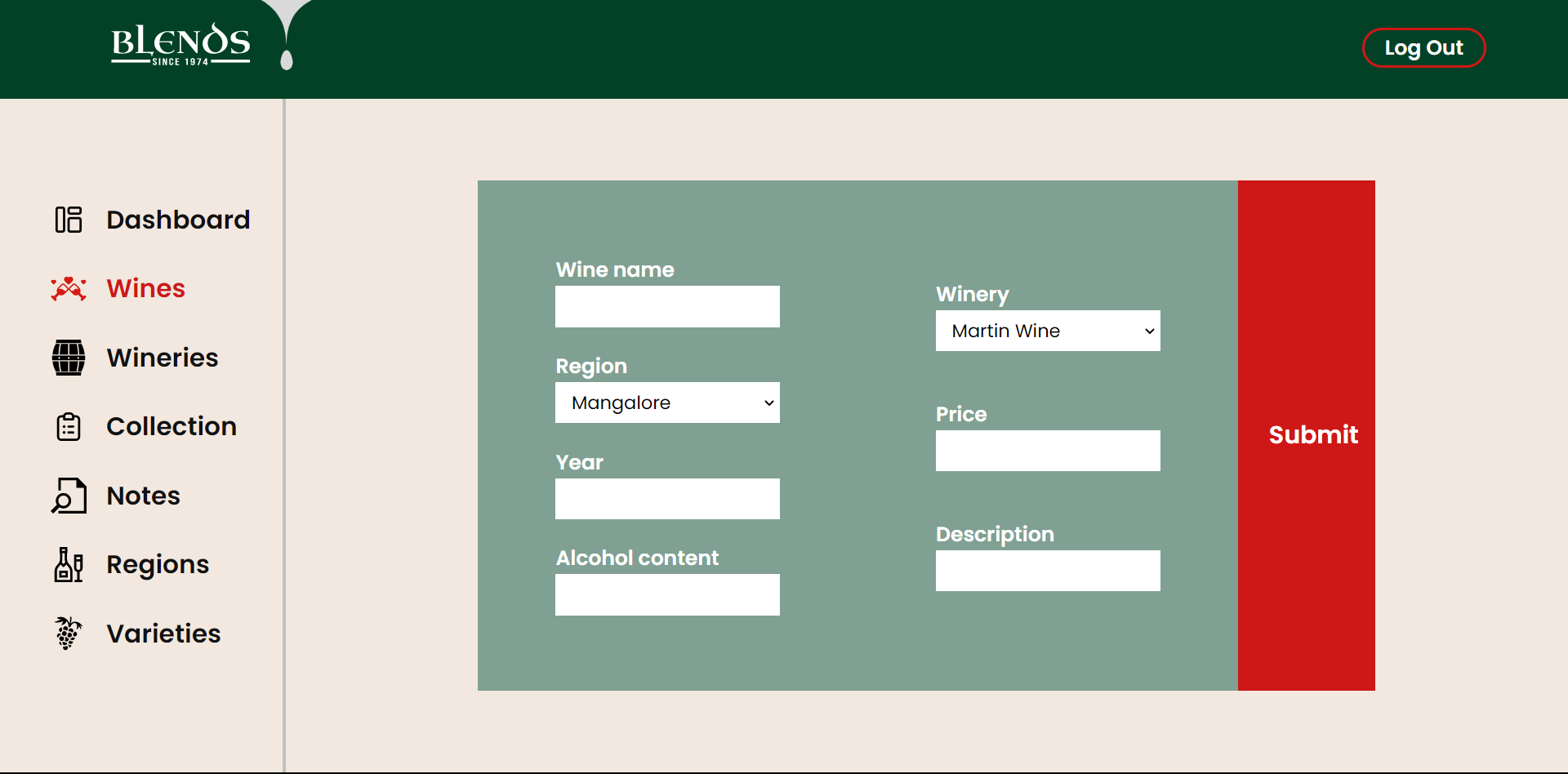


Image 16: add Wine page

11. finally the user can log out, by clicking on the Logout button which is placed in the top right of the page. And it will redirect the page to the starting page of the Blends web application.

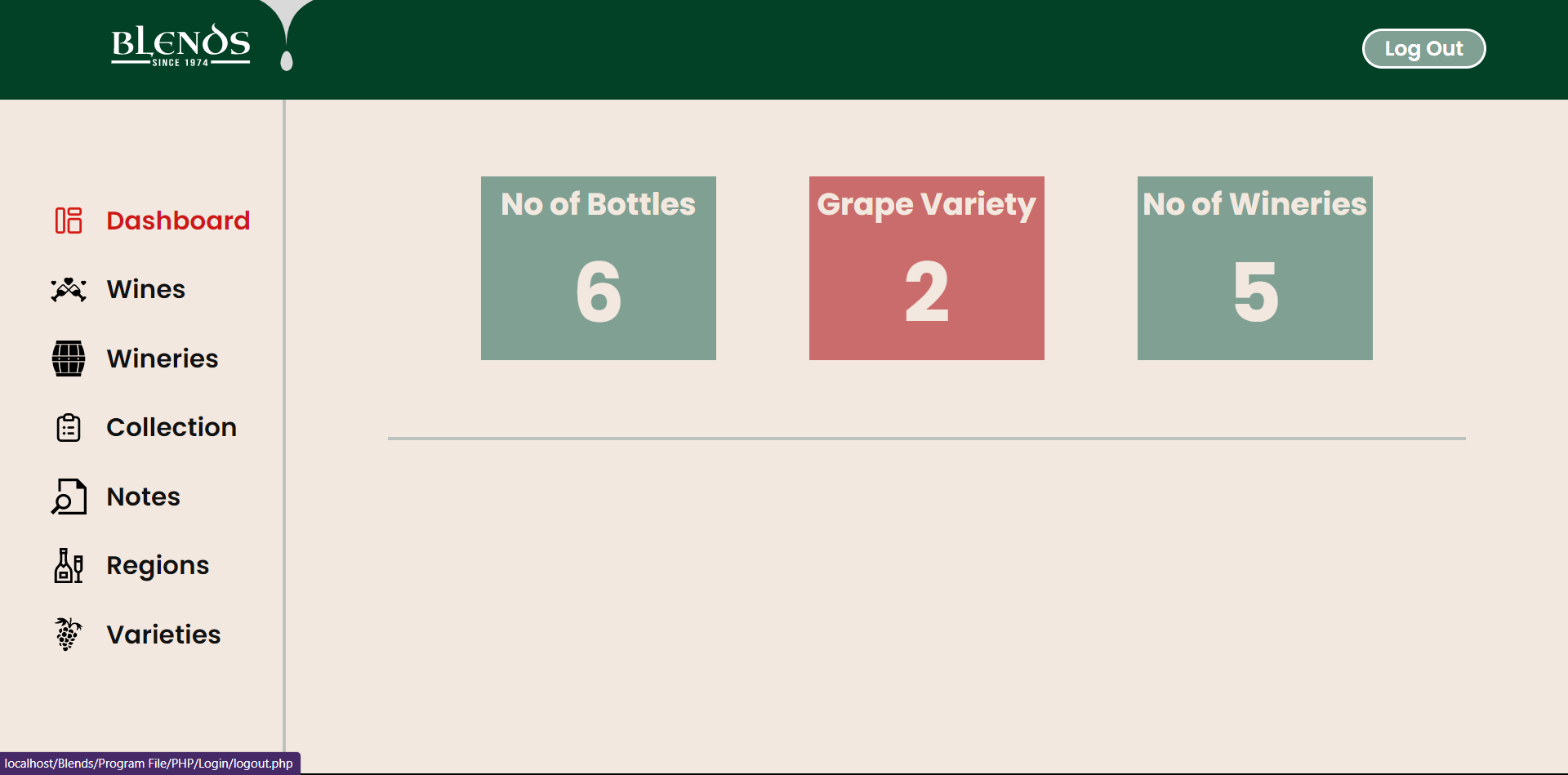


Image 17: Showing the Logout Button at the top right