# Spring Fundamentals Exam

# Philately Application

Exam for the ["Spring Fundamentals" course @ SoftUni](https://judge.softuni.org/Contests/4821/Spring-Fundamentals-Retake-Exam-15-August-2024).

**Philately** is the collection and study of postage stamps and related postal items. It is a hobby enjoyed by people of all ages worldwide, and it can be both a leisure activity and an academic pursuit. Philately began shortly after introducing the first postage stamp, the Penny Black, in 1840 in Great Britain. As postal systems expanded and evolved, so did the hobby of collecting stamps.

There are several requirements you must follow in the implementation:

## Database Requirements

The **Database** of the **Philately** application needs to support **3 entities**:

### User

* Has an **Id – "UUID-String" or Long**
* Has a Username (unique, not null)
  + Username length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has a Password (not null)
  + Password length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has an Email (unique, not null)
  + Must contain '@'.
* Has Wished Stamps *(wished stamps are those that the user intends to purchase,* *from other users, by adding them to their wish list)*
  + A temporary collection of stamps to which the current user can add stamps by selecting them from the list of stamps offered by other users.
  + One user may have many stamps in his wishlist but if he logs out without purchasing them, the stamps must disappear from this collection. The user has the option to add them again at the next login.
  + The user can add a single stamp to his wish list only once!
* Has Purchased Stamps *(the stamps the user has purchased)*
  + Collection with stamps that the user has purchased from other users.
  + One user can purchase many stamps.
  + If a user buys a stamp, other users should not be able to buy the same stamp.
  + The user can buy a single stamp only once!

### Stamp

* Has an **Id – "UUID-String" or Long**
* Has a Name (not null)
  + Name length must be between 5 and 20 characters (inclusive of 5 and 20).
* Has a Description (not null) *(a brief description of what the stamp depicts)*
  + Description length must be between 5 and 25 characters (inclusive of 5 and 25)
* Has a Paper (not null) *(the type of paper on which the stamp is printed)*
  + A stamp has one paper and one paper can have many stamps.
* Has a **Price** (not null)
  + A positive integer.
* Has an **Image URL** (not null)
  + Valid image URL containing no more than 150 characters.
* Has an **Owner**
  + A stamp has one Owner, the User who added it.

### Paper

* Has an **Id** – **"UUID-String" or Long**
* Has a **Paper name** (unique, not null)
  + There are three common types of paper an option between (WOVE\_PAPER, LAID\_PAPER, GRANITE\_PAPER)
* Has a Description (not null)
  + For WOVE\_PAPER – "Has an even texture without any particular distinguishing features."
  + For LAID\_PAPER – "When held up to the light, shows parallel lines of greater or less width running across the stamp."
  + For GRANITE\_PAPER – "Has tiny specks of coloured fibre in it, which can usually be seen with the naked eye."

Implement the entities with the **correct data types** and implement **repositories** for them.

## Initialize paper

* Implement a method that checks (when the app started) if the database does not have paper and initialize them.
  + **You are free to do it in different ways.**

## Page Requirements

### Index Page (logged out user)

A screenshot of a cell phone

Description automatically generated

### Login Page (logged out user)

A screenshot of a login page

Description automatically generated

### Login Page validations

A screenshot of a login screen

Description automatically generated A screenshot of a login screen

Description automatically generated

### Register Page (logged out user)

A screenshot of a computer

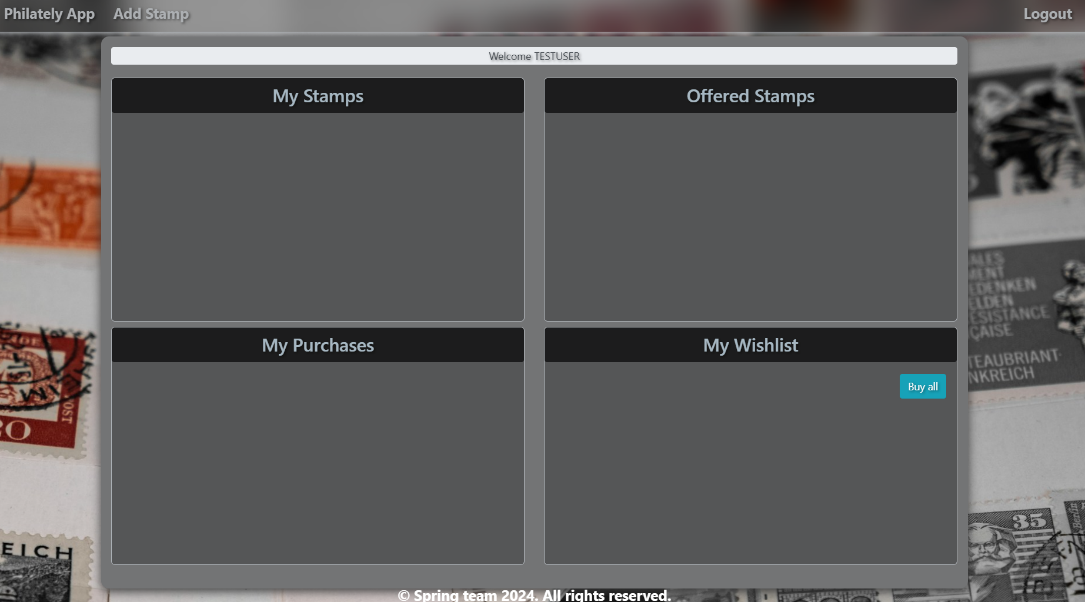
Description automatically generated

### Register Page validations

A screenshot of a phone

Description automatically generated

### Home Page (without having any stamps when a logged in user)



### Add stamp (when a logged-in user)

A screenshot of a computer screen

Description automatically generated

### Add stamp validation

A screenshot of a phone

Description automatically generated

### Home Page (with stamps, when a logged-in user)

* Note: The left top section (My Stamps) of the page should display **all the current logged user stamps** from the database which he added, if there are any.
* Note: To the left bottom section (My Purchases) of the page should display the **currently logged-in user's purchased stamps** that can be bought fromthe **"My Wishlist"** section.
* Note: The right top section (Offered Stamps) of the page should display **all offered stamps added by the rest users** (except the current logged one).
* Note: The bottom right section (My Wishlist) of the page should display **wished stamps ready for purchase by the current logged-in user**.

A screenshot of a computer

Description automatically generated

The templates have been given to you in the application skeleton, so make sure you implement the pages correctly.

**NOTE**: The templates should look as shown above.

**NOTE**: The templates do **NOT** **require** **additional** **CSS** for you to write. Only **bootstrap** and the **given CSS** are enough.

## Functional Requirements

The Functionality Requirements describe the functionality that the application must support.

The application should provide **Guest** (not logged in) users with the functionality to **log in**, **register** and view the **Index** page.

The application should provide **User** (logged in) with the functionality to **log out, add** a new stamp **(Add stamp page), view** all stamps **(Home page)** and **add** a stamp from **"Offered Stamps"** to his **Wishlist** as well as to **purchase** already added stamps.

Philately Application in the navbar should **redirect** to the appropriate URL depending on that if the **user is logged in**.

The application should provide functionality for **adding stamps** with the following paper types - WOVE\_PAPER**,** LAID\_PAPER**, or** GRANITE\_PAPER.

The stamps should be separated into four different sections according to their ownership.

**At the top**, above all sections in a banner, the **username** of the logged-in user should be displayed:   
**"Welcome current logged-in User"**.

### My Stamps section

This section displays added stamps by the **currently** logged-in **User**. When the **User** adds a **new** stamp, it should appear here.

### Offered Stamps section

This section shows stamps added and offered by **all other users** who use the app.

When the user clicks on the "**To Wishlist" button of some stamp** in the **"Offered Stamps"** section, he adds the stamp to **his "My Wishlist"** section. You should **not delete** this stamp from **DB** and you should **not** remove it from the **"Offered Stamps"** section, yet. The stamp should be added to the **"My Wishlist"** collection with wished stamps of the currently logged-in **User**. The added stamp should appear in the **"My Wishlist"** section of the **User.**

### My Wishlist section

In this section, the wished stamps of **currently** logged-in **User** are shown, and they **have not been** purchased but have **only been added temporarily**. The user should be able to delete stamps from his Wishlist one by one using the "**Remove”** button. If it does, the relevant stamp **should be removed** from the **"Wishlist"** section. If the user wants to **purchase** wished stamps, he must click the **"Buy all"** button. If he does, the stamps from the **"My Wishlist"** section **should** move to the **"My Purchases"** section, then you must **remove** the purchased stamps from the **"Offered Stamps"** section **without** **deleting** them from the database. The Wishlist of the current **User** **goes empty**. The rest of the users **should no longer be able to see** these (purchased) stamps in **any section**.

### My Purchases section

If the currently logged-in user has **successfully purchased** any stamps - they are displayed in this section

The application should store its data in a **MySQL** database.

## Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access the Index page.
* Guest (not logged in) users can access the Login page.
* Guest (not logged in) users can access the Register page.
* Users (logged in) can access the Home page.
* Users (logged in) can access Add Stamp page.
* Users (logged in) can access Logout functionality.

When uploading a solution it is necessary to make an archived file of **src** + **pom.xml**. Go to **SoftUni Learning System** select the archived file from your device and press the "**Upload Homework**" button.

**Only the LAST uploaded file is saved.**

## Scoring

### Database – 10 points.

### Pages – 25 points.

### Functionality – 40 points.

### Security – 5 points.

### Validations – 15 points.

### Code Quality – 5 points.