Gardening website

Project documentation

Student(s): Bodea Răzvan-Marius

**Group: 30432**

**Contents**

[I Project specification 2](#_Toc105472276)

[1.1 Domain Model Diagram 3](#_Toc644374682)

[II Use-Case model 3](#_Toc875761706)

[2.1 Users and stakeholders 4](#_Toc1322152187)

[2.2 Use-Case identification 4](#_Toc1784770617)

[2.3 UML Use-Case diagram 4](#_Toc1862277618)

[III Architectural design 4](#_Toc1348849003)

[3.1 Conceptual architecture 5](#_Toc1537720403)

[3.2 Package diagram 5](#_Toc414455459)

[3.3 Class diagram 5](#_Toc239513463)

[3.4 Database (E-R/Data model) diagram 5](#_Toc319679081)

[3.5 Sequence diagram 6](#_Toc1417122837)

[3.6 Activity diagram 6](#_Toc1113334912)

[IV Supplementary specifications 6](#_Toc347811227)

[4.1 Non-functional requirements 6](#_Toc1081796918)

[4.2 Design constraints 6](#_Toc636359167)

[V Testing 7](#_Toc1200093866)

[5.1 Testing methods/frameworks 7](#_Toc912424145)

[5.2 Future improvements 7](#_Toc1475142911)

[VI Bibliography 7](#_Toc1469804527)

# I Project specification

The main goal of this project is to create a web application for gardening enthusiasts. The main objectives of the project will be the following:

The app will let users log in with an existing account or create a new one, create one profile per account where they could post pictures of their own plants and a short description of themselves.

The app will include a shop feature where users could buy different plants and maybe list plants for sale. A bill will be created when the order is finished, and a worker will be notified.

Another feature will be a page with different gardening tips.

Moderators will have the role of verifying pictures and listings of items for sale to make sure they are appropriate and allow or deny them.

Admins will be able to create new moderators and perform CRUD operation on the data.

## 1.1 Domain Model Diagram

Diagram

Description automatically generated

# II Use-Case model

The most expected use case of the web app will be a user logging in or creating an account, then browsing the pages, editing its profile and browsing and ordering from the shop.

Moderators will have the role of allowing or removing different requests from the normal users.

The goal is to create use cases that have a natural flow and feel easy to follow.

## 2.1 Users and stakeholders

Normal user: will have access to the basic functionalities of the app: creating an account, logging in, browsing the shop, seeing tips, creating an order, editing their profile.

Moderator: will have the role of managing the text and images people will post and decide if they are appropriate for the app, on top of the normal user functions

Admin: will be able to promote users to moderators and perform CRUD operations on the data in the database.

Stakeholders: gardening enthusiasts all around the world (younger and older), plant stores.

## 2.2 Use-Case identification

1.

Use case name: Register

Level:

Main actor: user (new user, without an account)

Main success scenario:

Enter a correct username (not already taken),

Enter a correct password,

* A new account for the specific user is created and can be used on the login page.

Extension:

Failure: Username already taken/ wrong password

* Display error message, do not create account

2.

Use case name: Buy plant

Level:

Main actor: user

Main success scenario:

1.Go to store page,

2.Select a plant (go to plant page), add quantity,

3.Add plant to cart,

Repeat 2 and 3 as many times

Place order.

* A new order with the specific plants will be created for the user

Extension:

Failure: Cannot order an empty cart.

3.

Use case name: Manage post request

Level:

Main actor: moderator

Main success scenario:

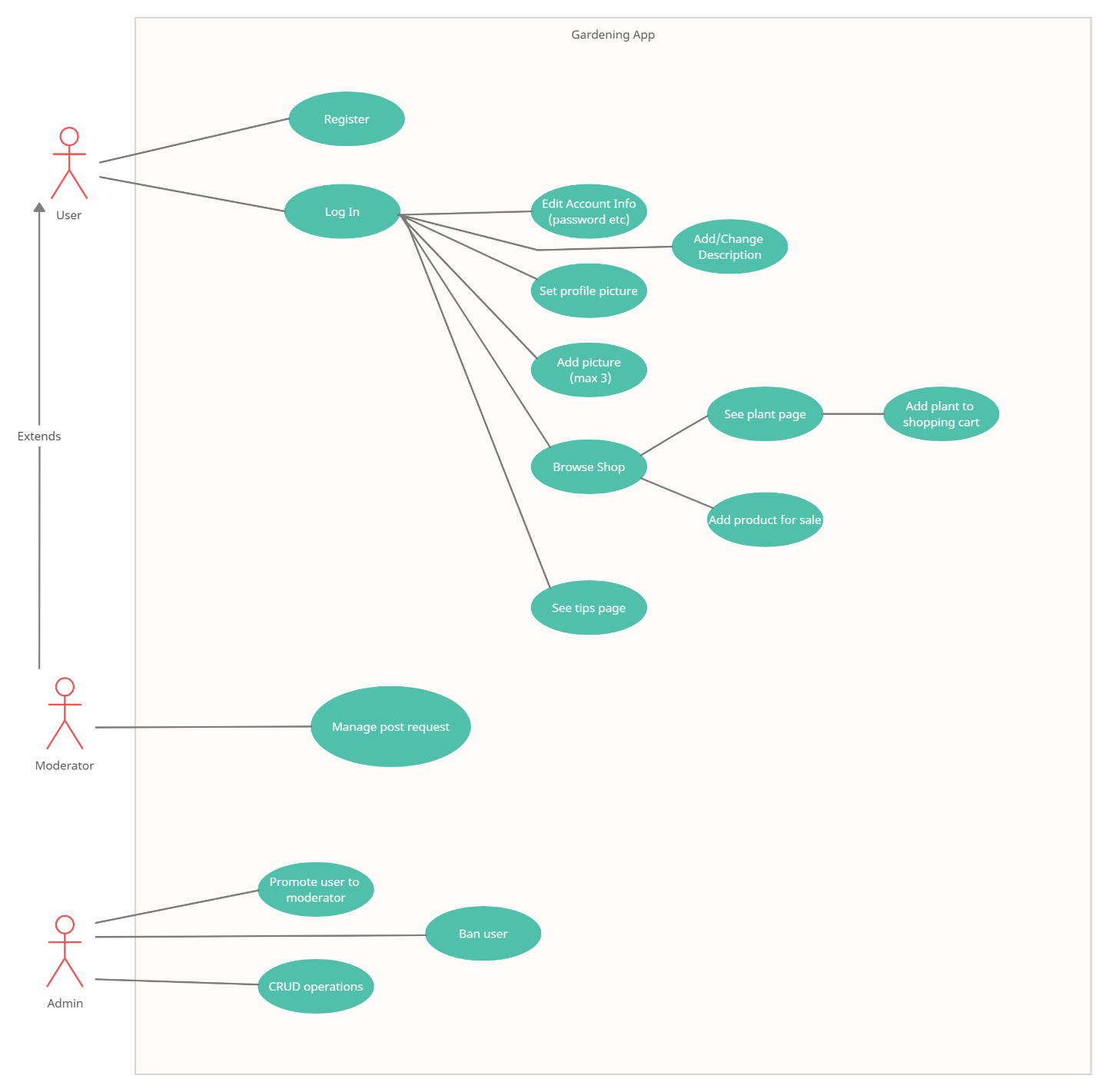
1.Go to request page

2.Accept/Deny post request

* The request will be saved and added or discarded, if denied

Extension:

## 2.3 UML Use-Case diagram

**

# III Architectural design

The architectural design should be a starting point for the development of the application and the final product should reflect the specifications. The architectural design is built by taking into consideration all the functional and non-functional requirements (price, need for real-time computation, speed etc.)

## 3.1 Conceptual architecture

Web app, with relational database (SQL), one server, multiple clients, using the Spring framework.

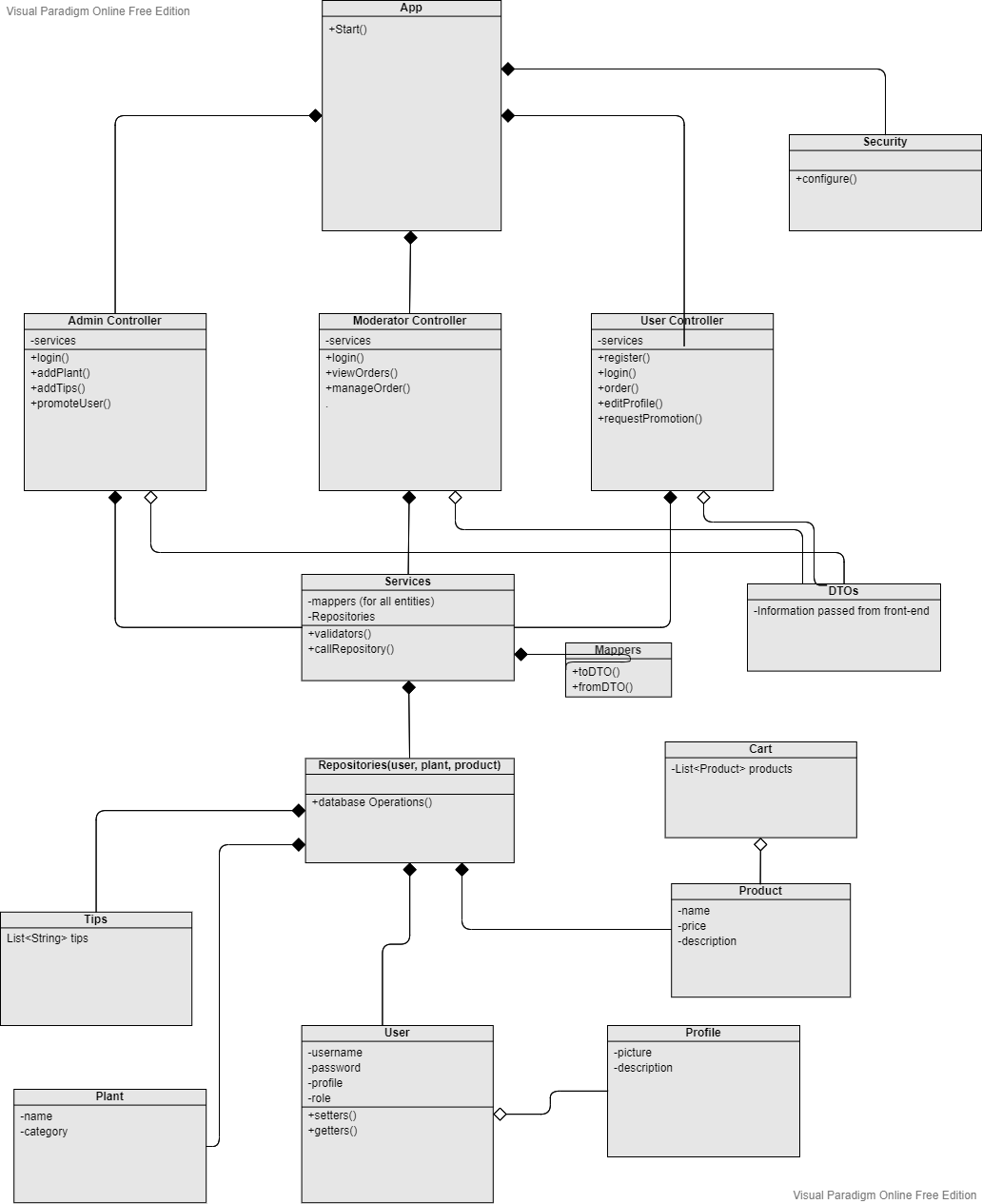
The architecture of the app will be a layered architecture, implementing the Model-View-Controller structural design pattern.

An observer design patter will be used for notifying the moderator when a user adds a new picture or tries to post a plant for sale.

The singleton design pattern could be used by defining services that are called for working with different products.

## 3.2 Package diagram

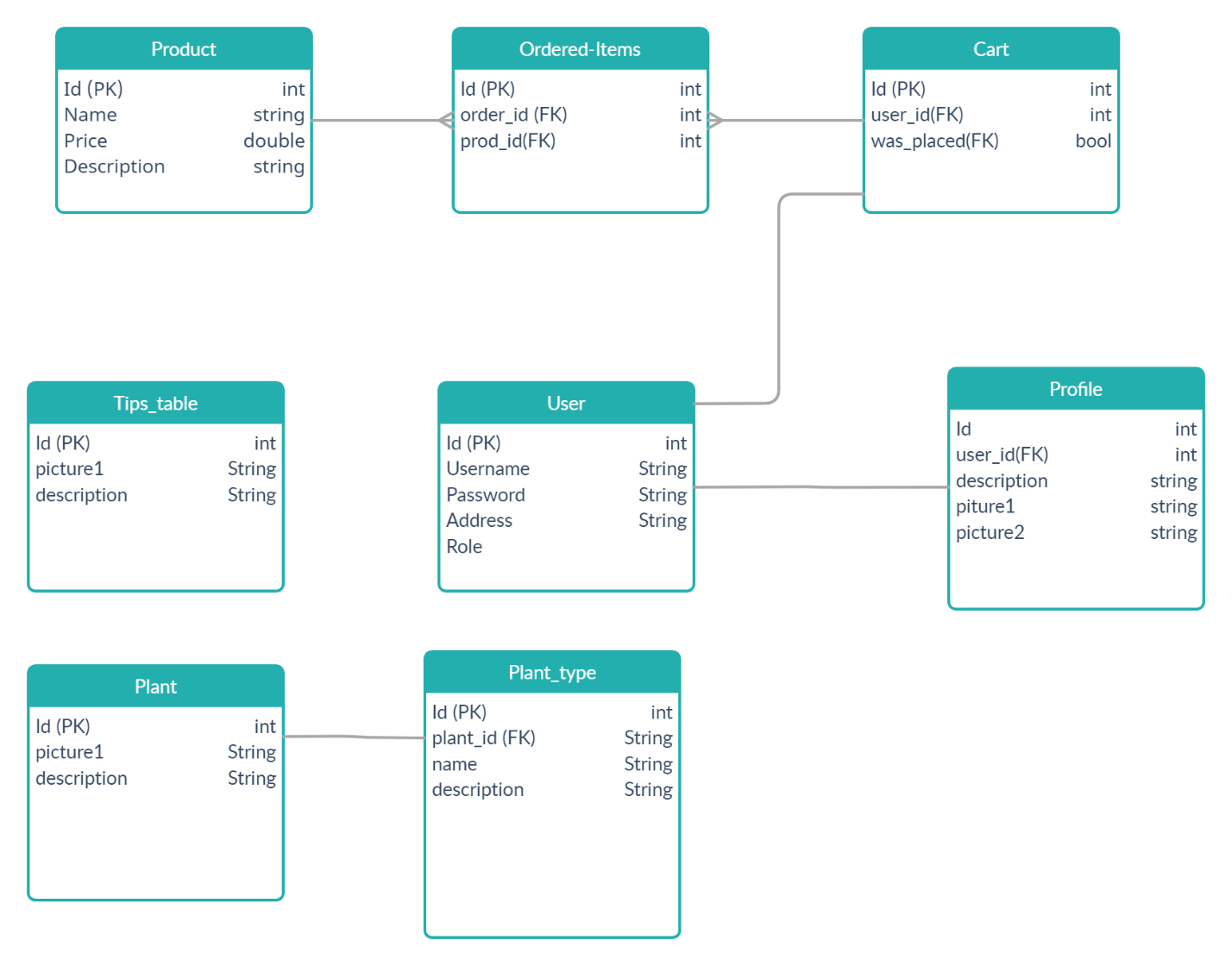
## 3.3 Class diagram (simplified)

**

## 

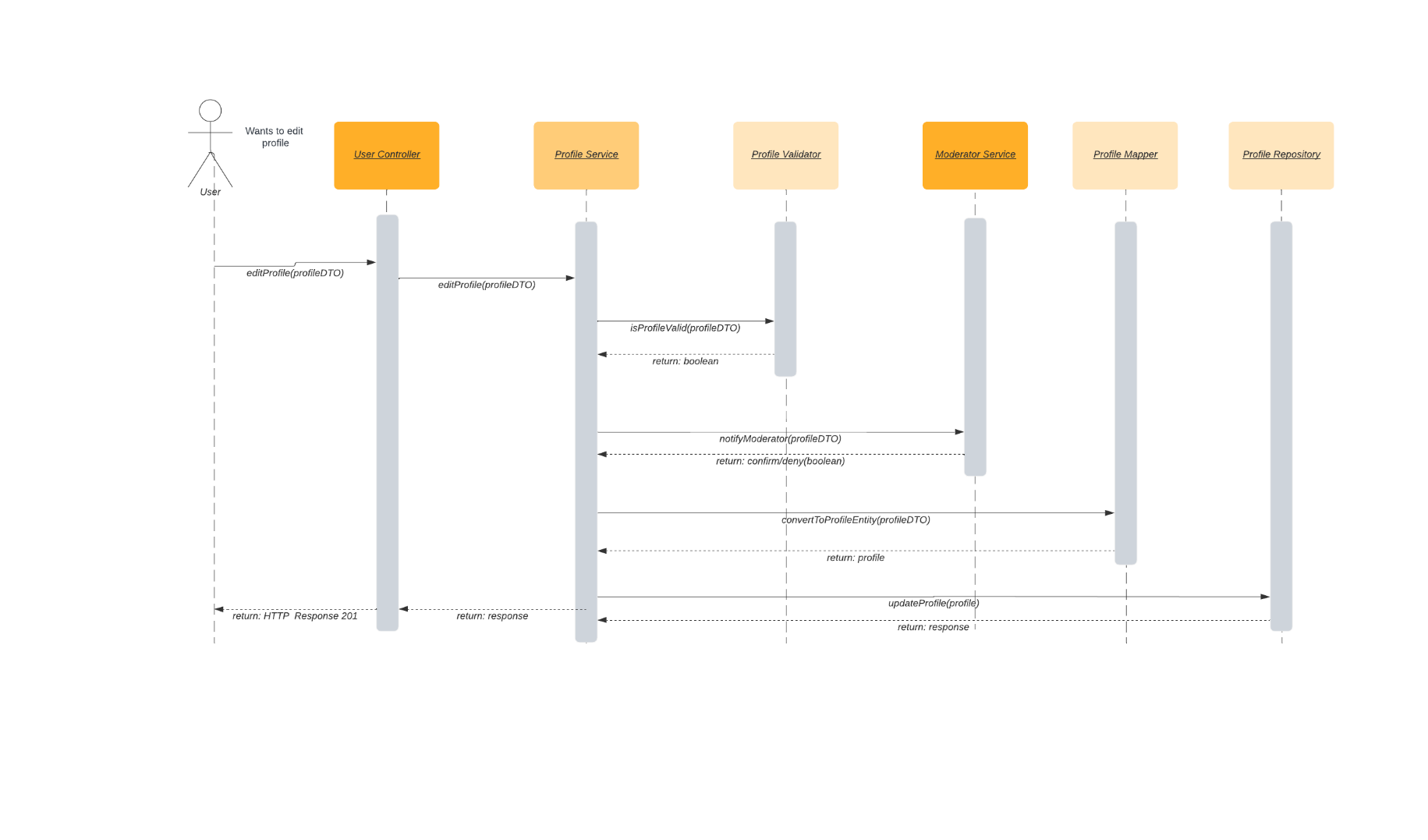
## 3.4 Database (E-R/Data model) diagram

(Subject to change)

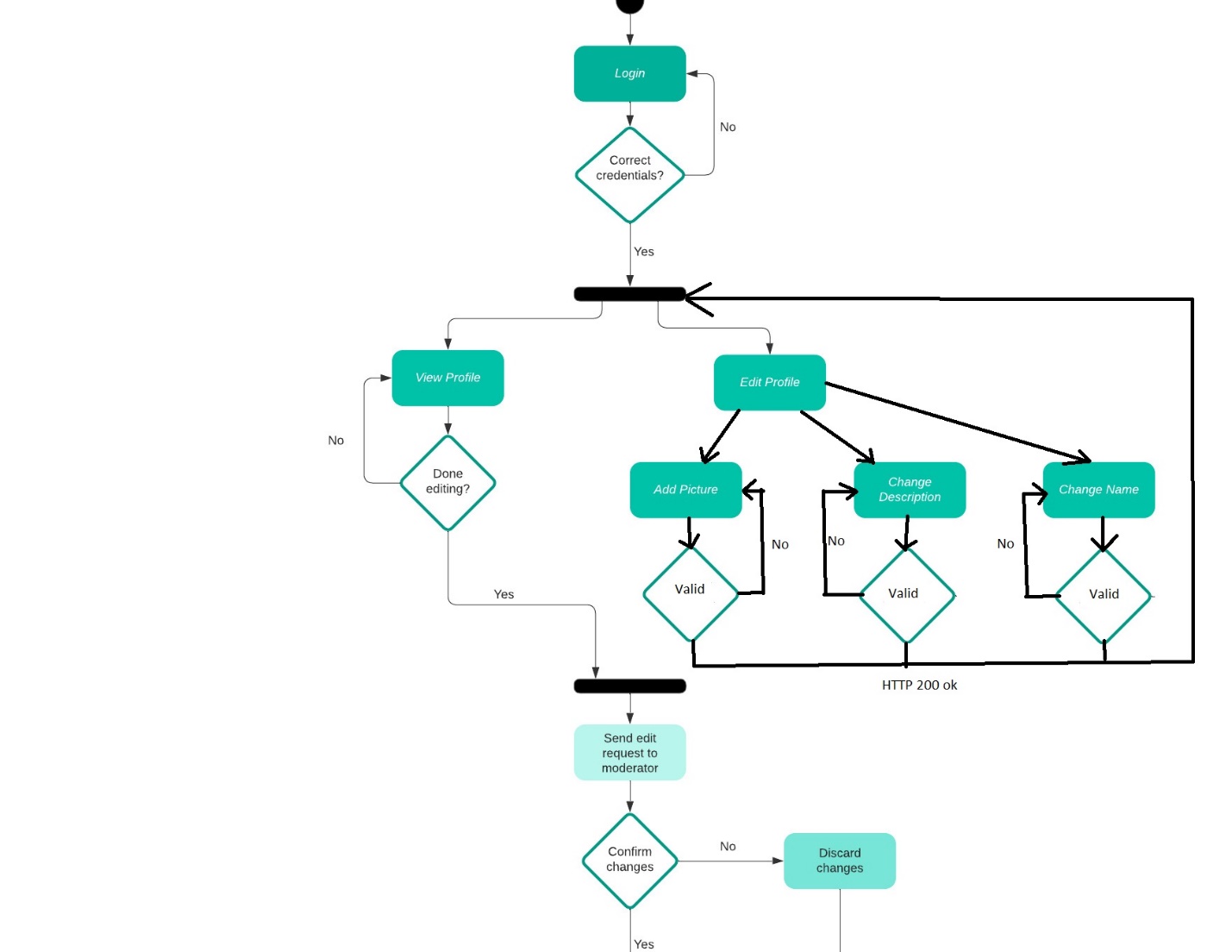


## 

## 3.5 Sequence diagram



## 3.6 Activity diagram



# IV Supplementary specifications

## 4.1 Non-functional requirements

Security: use Spring Security for determining the logged-in users at a given time and have access to their data.

A user should have access to the web page only if he is logged in.

The user should not be able to access the pages that are dedicated to admins.

Scalability: The web page should be able to have multiple users logged in at the same time and an admin

Cross-platform: The web page should be accessible to phone users.

Performance: The loading time should be as small as possible (<1s).

The time between an order being placed and the admin/moderator being notified should be as small as possible.

## 4.2 Design constraints

Technical constraints: Have at least 1 pre-defined admin account

Have at least 5 plants added in the database

Have at least 2 gardening tips

Security: Use a hashing algorithm for storing a hashed version of the password so they are hard to find if the database is accessed by a malicious user.

Keep the user’s personal data safe.

Languages: Java, JavaScript, CSS, HTML, SQL.

The database should be a relational one.

# V Testing

For testing I used Mockito and Junit

## 5.2 Future improvements

The project turned out to be a bit different from the original idea.

Some improvements could be adding a report button so that users are able to report posts that they consider inappropriate, so that a moderator could review them, or improving the way the balance works (right now for adding money to your account you just enter a number).

# VI Bibliography