Střední průmyslová škola elektrotechnická

Informační technologie

Ječná 30, 120 00, Praha 2

Garden of tasks

(to-do list)

Martina Ilko

IT

2025

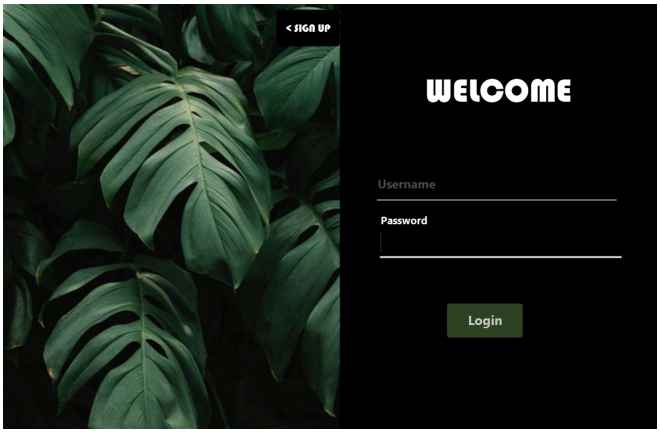
Content

Project Documentation

1. Goal of the project

The goal of the project 'Garden of Tasks' is to create a to-do list where users can manage their daily tasks and earn coins as a reward for completing them. These coins can be spent to buy flowers and fill a virtual garden. The application aims to make productivity more fun and motivating.

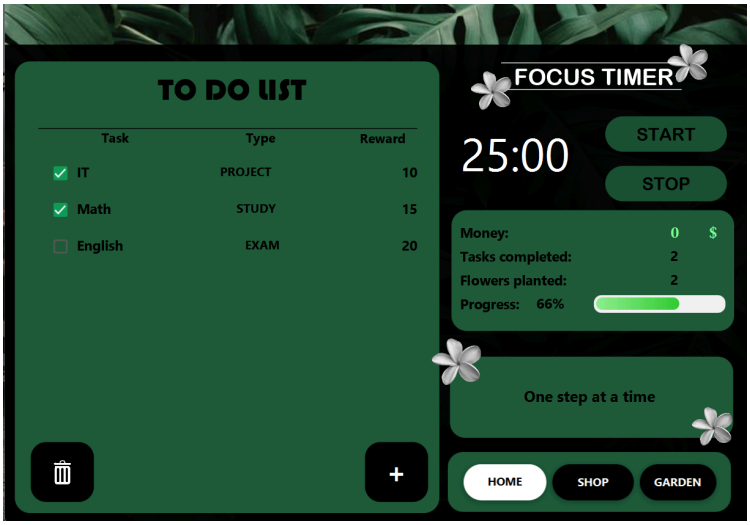
2. Project description



The app is made with JavaFX and uses the JFoenix library to make the design look better. Each task has a type, and when the task is finished, the user gets some coins based on that type. That application also has a timer to help with focus and shows motivational quotes to keep users inspired.

The application allows users to sign up and log in. Tasks can be added, marked as done, and removed. Completing a task rewards the user with coins.

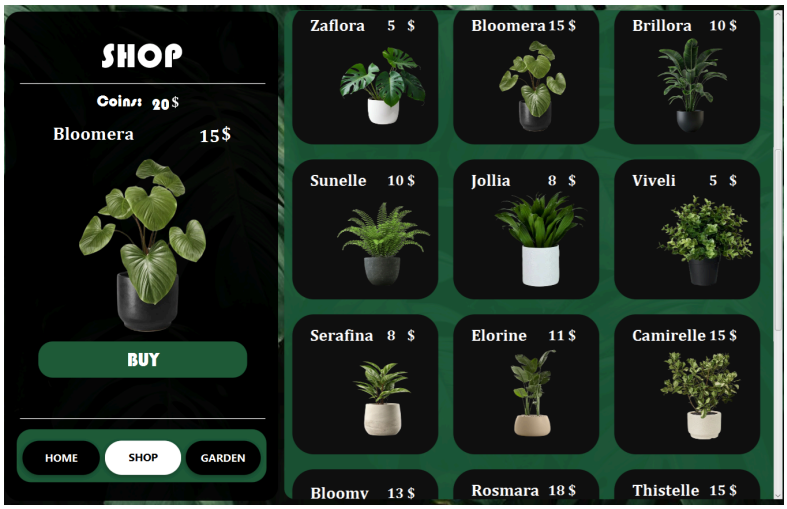
Main page:



When the user clicks the "+" button to **add** a new task, a new window opens where they can enter the task's name, a description, and choose the task type. The task is immediately added to the user's **task list**. If the user hovers over a task, they can see the task **description** as a tooltip. The task type and the reward are also visible. Once the user marks the task as completed, the ProgressBar updates to show how much progress they've made and what percentage of all tasks they have finished. From this page, the user can go to the shop or visit their garden.

Shop page:

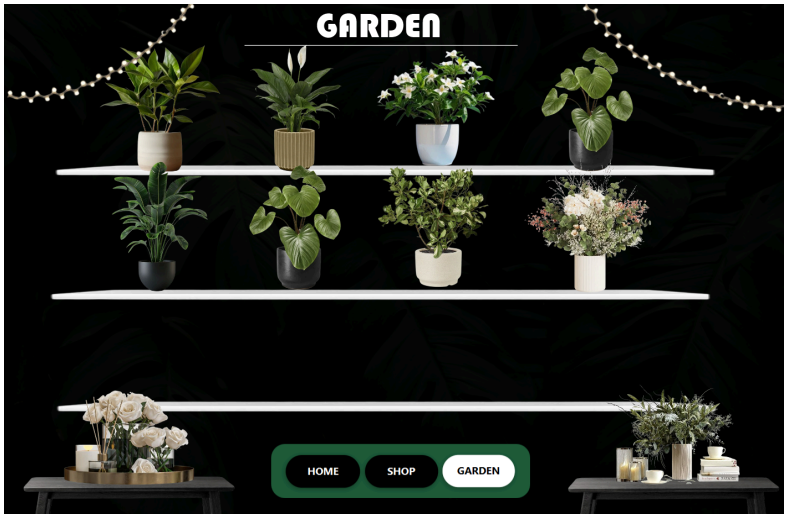
In the shop, all available plants for purchase are displayed. When the user clicks on a flower, it appears on the side as the selected one, and it can be bought. After buying, the flower is added to the garden, and a congratulatory message is shown. If the user doesn't have enough coins, the program will display a warning message. From the shop, the user can go to the garden or return to the main page.



Garden page:

The garden is simple — it has shelves where the images of the purchased flowers are displayed.

All data is saved using Java serialization so progress is not lost when the application is closed.



3. System requirements

The project was developed in Java, specifically using Java SE **21**. To run the application, the corresponding **JDK 21** must be installed on the system. The application uses **JavaFX** for the graphical user interface, and the **JFoenix 9.0.1** library to improve the visual design with UI components. Therefore, both **JavaFX SDK** and **JFoenix** are required to run and build the application successfully. The project was created and tested in **IntelliJ IDEA.**

The application can be launched in IDE or from the command line with the correct module path. (JavaFX modules used: javafx.controls, javafx.fxml)

4. Project structure

The application is designed using **OOP.** MainApp is the entry point that initializes the primary JavaFX stage**.**

`Package model` – contains classes such as `Task`, `User`, `Flower` and `TaskType` enum that represent the objects.

`Package controller` – handles the interaction between the user interface and the application logic. It processes actions such as task creation, task completion, and flower purchases.

The shopping functionality is implemented in the `ShopController`, which displays available flowers, handles the selection and purchase process, updates the user's coin balance, and adds bought flowers to the user's garden. The `GardenController` then visualizes the user's collection of flowers in a simple layout.

Scene transitions in the application are handled by the `ViewSwitcher` class located in the `util` package. It is a utility class designed to simplify navigation between different parts of the JavaFX application.

The `switchToScene()` method takes three arguments – the current stage, the path to the FXML file, and the currently logged-in user. It loads the FXML scene and sets it as the active view. If the associated controller belongs to one of the main screens (`TaskController`, `ShopController`, `GardenController`, or `AddTaskController`), the method automatically passes the `User` object to it.

This way, the user's data (like tasks, coins, or flowers) stays available in every scene without repeating the same code.All data like tasks, coins, and flowers are saved to a file and loaded from it using serialization, so the user's progress is kept even after closing the program.

5. Testing

I tried to write tests , but I kept getting an `IllegalAccessError`. I looked for solutions, but I couldn't fix it. I think it's because testing JavaFX controllers can be difficult and maybe they need special setup or libraries. Still, I managed to test a few things that don't depend directly on the JavaFX UI. I did my best, so I hope the tests I managed are enough :3

User test: I tested adding tasks, plants, and coins, because these are the main ways a user makes progress in the app. I also tested removing coins to make sure the balance updates correctly.

UserManager test: I tested registration and login, because they are critical for user access. I also tested logging in with a wrong password to check error handling.

TaskManager test: I tested adding tasks, completing and uncompleting tasks, and counting completed tasks.

6. User guide

1. Run the application using the correct Java and JavaFX version.
2. On the welcome screen, either log in with your existing account or sign up for a new one.
3. You can create new tasks and use the focus timer to stay productive
4. Complete tasks to earn coins as rewards.
5. Open the shop and buy beautiful plants..
6. Check out your plants in the garden.
7. Your progress is saved automatically.
8. Finish tasks and let your garden grow! 🌿

7. Conclusion

While working on this project, I had some problems. I couldn't get the unit tests to work, even though I tried different ways to fix them. After I had issues with creating the **.jar** file — mostly because of different Java and JavaFX versions. After creating the **.jar**, the program couldn't read files correctly, which made things more difficult.

Even with these problems, I learned a lot. Now I understand better how to build a JavaFX app, save user data with serialization, and use design patterns like Singleton.

8. Some references

- **JAR:**

  https://youtu.be/_XQjs1xGtaU?si=_T7F4IZkV-Rvv8hz

  https://youtu.be/d02PK8C5EaA?si=io_4FAAWlCJondDz

- **ListCell:**

  Display Custom Items in JavaFX ListView | Baeldung

- **Singleton:**

  Singleton Method Design Pattern in Java | GeeksforGeeks

- **ChatGPT**