Planify

Plan your future

Name: Moga Eduard Mihai
Group: 4

Table of Contents

Deliverable 1

Project Specification

Planify is a Web App that lets you plan your feature to achieve it. You can notebooks that let's you track the path to your destinations. Each day you can bring your challenges to the table and the app you'll come in our help at the end of the week with statistics and improvements. For each dream you will have the notebook ask for a resolution you want to achieve, to motivate you to reach the heights. When needed you can ask the AI for more information about how you can work your way through.

At the end of the week, you will get all the results for your notebook in the archive so you can see when needed. All the work you put in will be rewarded with achievements you can see in the

Achievements tab.

Functional Requirements

1) Create user through register
2) Admin manages the users
3) Access the application through login
4) Create, modify, delete notebooks
5) Sort notebooks after its type
6) Get achievements
7) Join groups
8) Get recommendations from the AI

Use Case Model
Use Cases Identification:

Use case: Manage users

- **Level**: Admin Interaction
- **Primary Actor**: Super System User (Admin)
- **Main success scenarios**:
    - Admin should navigate through the users
    - Admin should be able to delete users, modify and add super users
    - Admin should be able to see all the achievements
    - Admin should be able to moderate the groups

Use case: Login / Register

- **Level:** User Interaction
- **Primary Actor:** User
- **Main success scenarios:**
    - Users should register with the required fields
    - Users should login with the email username and password

- **Failures success scenarios:**
    - Users should be notified of the right error and be redirected to the specified page

Use case: Manage notebook

- **Level:** User Interaction

- **Primary Actor:** User

- **Main success scenarios:**

    - Users should choose between the types of the notebook and complete with the required and related fields

    - Users should be able to sort the notebooks, search the notebooks, delete and update with relevant information

    - Users should be able to view the older notebook as well as the archives with the relevant analysis made by the AI

Use case: Group action

- **Level**: User Interaction

- **Primary Actor**: User

- **Main success scenarios**:

    - Users should be able to join and leave community groups for the same relevant hobbies and resolution

    - Users should be able to communicate between them in the dedicated servers

Use case: Achievements

- **Level:** System Interaction

- **Primary Actor:** User

- **Main success scenarios:**

    - Users should get achievements based on their performance and metrics set by the SUPER ADMIN / SYSTEM CONFIGURATION and view progress.

UML Use Case Diagrams
IN DRAW.IO FORMAT
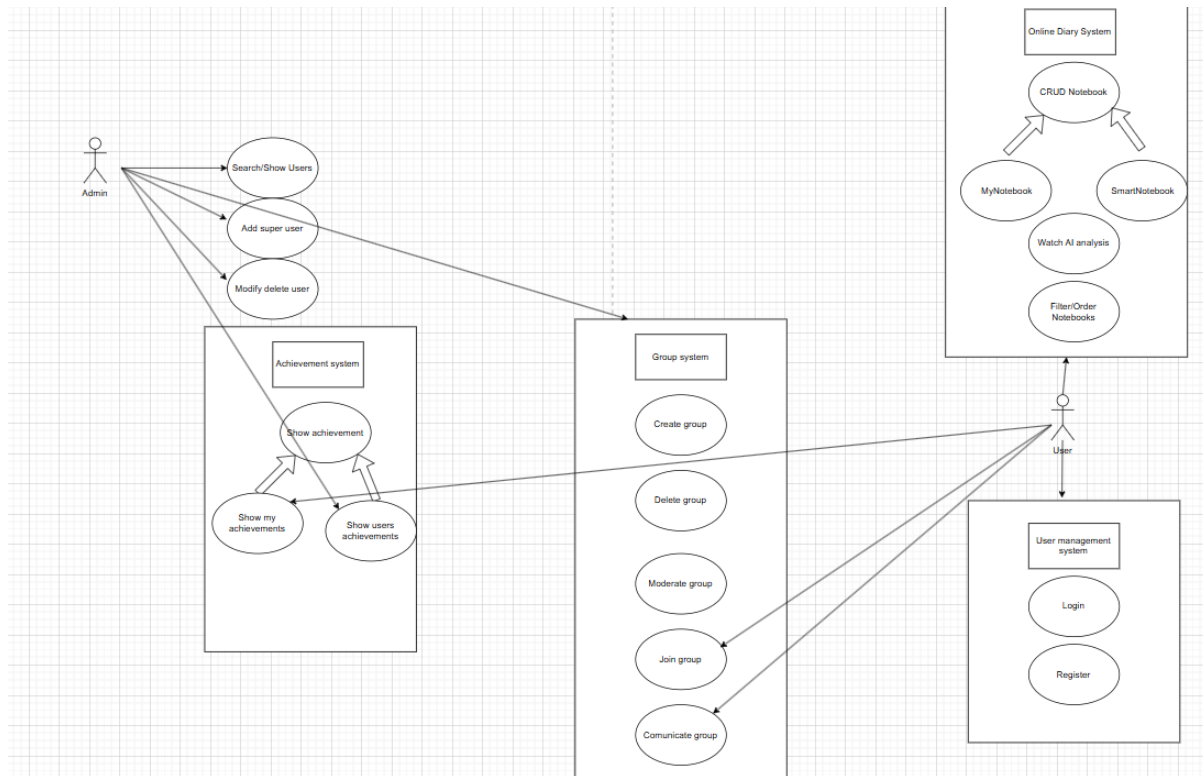
Supplementary Specification

Non-functional Requirements

- **Performance**: Ensures the fast response of the *AI*, *analyzes* the *group* communication without delay but also a seamless *login* and *register* experience
- **Scalability**: By following the right principles and *architectural designs* scalability of the system can be assured in the right context for multiple *users* doing *various tasks*.
- **Security** of the system: *Encryption* of the data is a necessity, also the prevention of *unauthorized* access.
- **Usability**: The ease of use should be a priority for a seamless *user experience*, achieved by simplifying various processes.

Design Constraints

- **Backend**: *Java* (Language)

  - Framework: Spring Boot

- **Frontend**: JavaScript, HTML, CSS (Language), Thymeleaf (Java – template engine)

  - Framework: NextJS

  - Library: ReactJS

- **Tool Manager:** Maven

- **Database:** MySQL (MySQL workbench)

- **IDE:** IntelliJ

- **Architectural Constraints:**

  o *MVC – model view control* architectural design with the Spring package separation in repository, model, controller, templates, service

Glossary

Deliverable 2
Domain Model
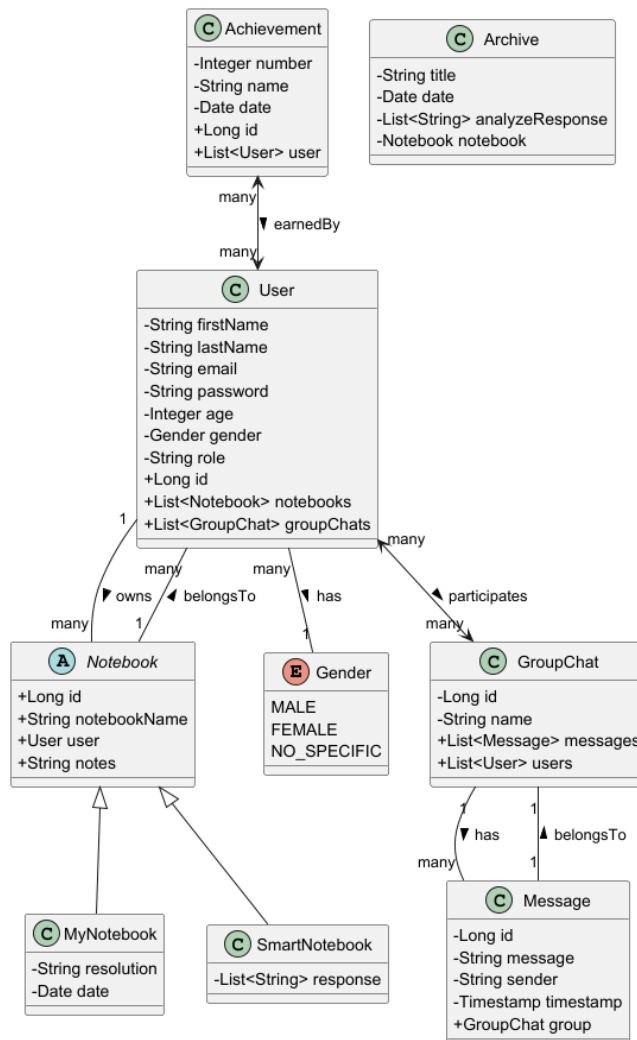
Architectural Design
Conceptual Architecture

Package Design

Component and Deployment Diagram
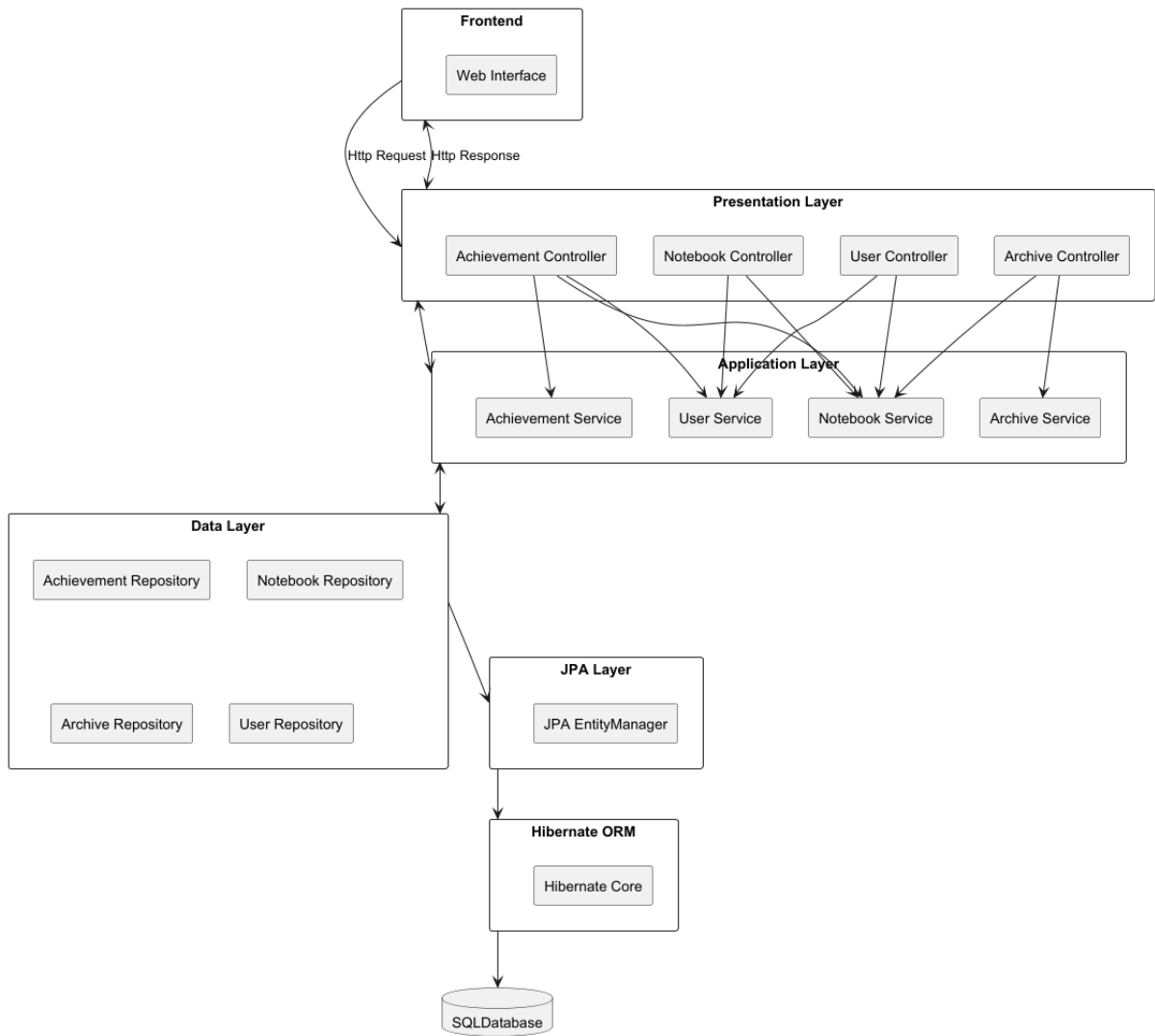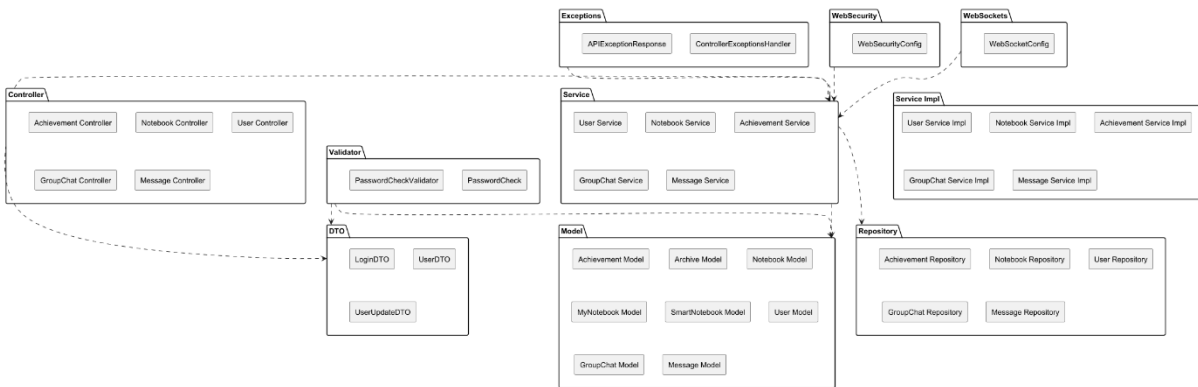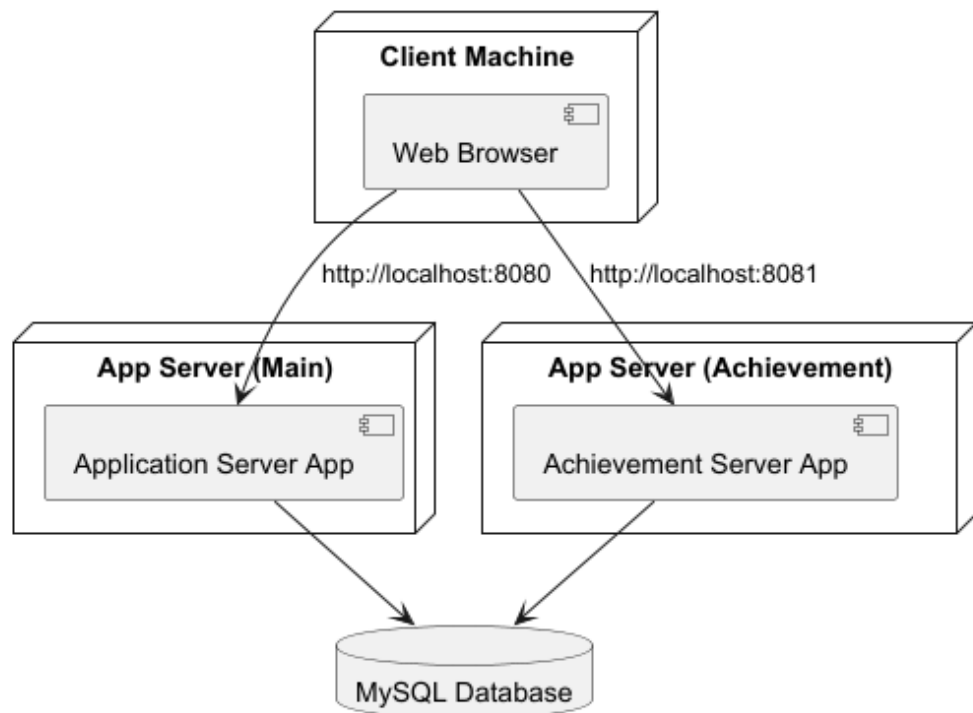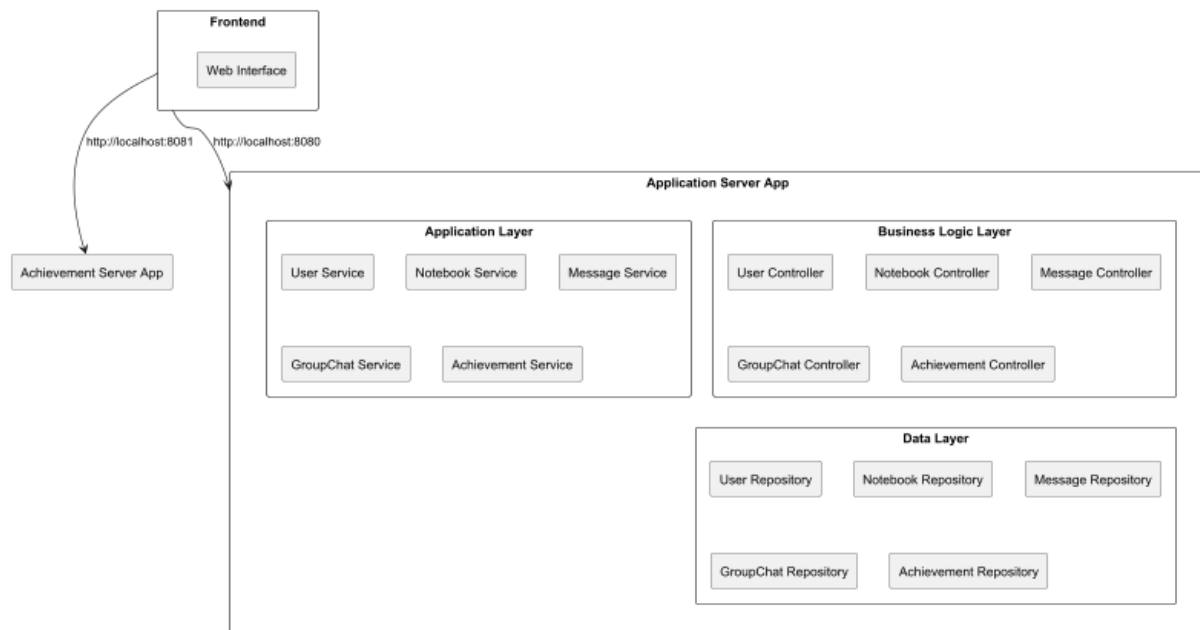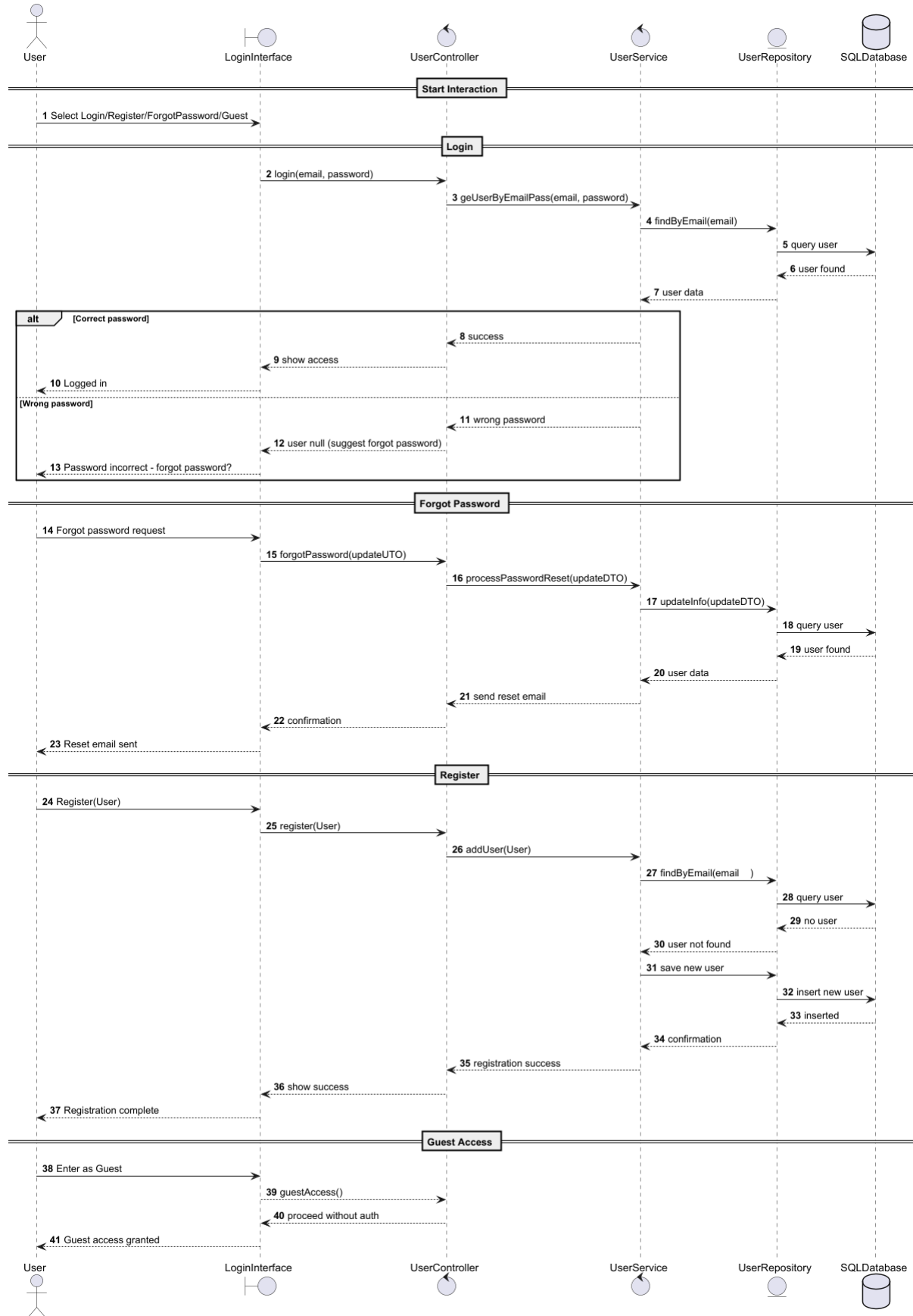
Deliverable 3
Design Model
Dynamic Behavior

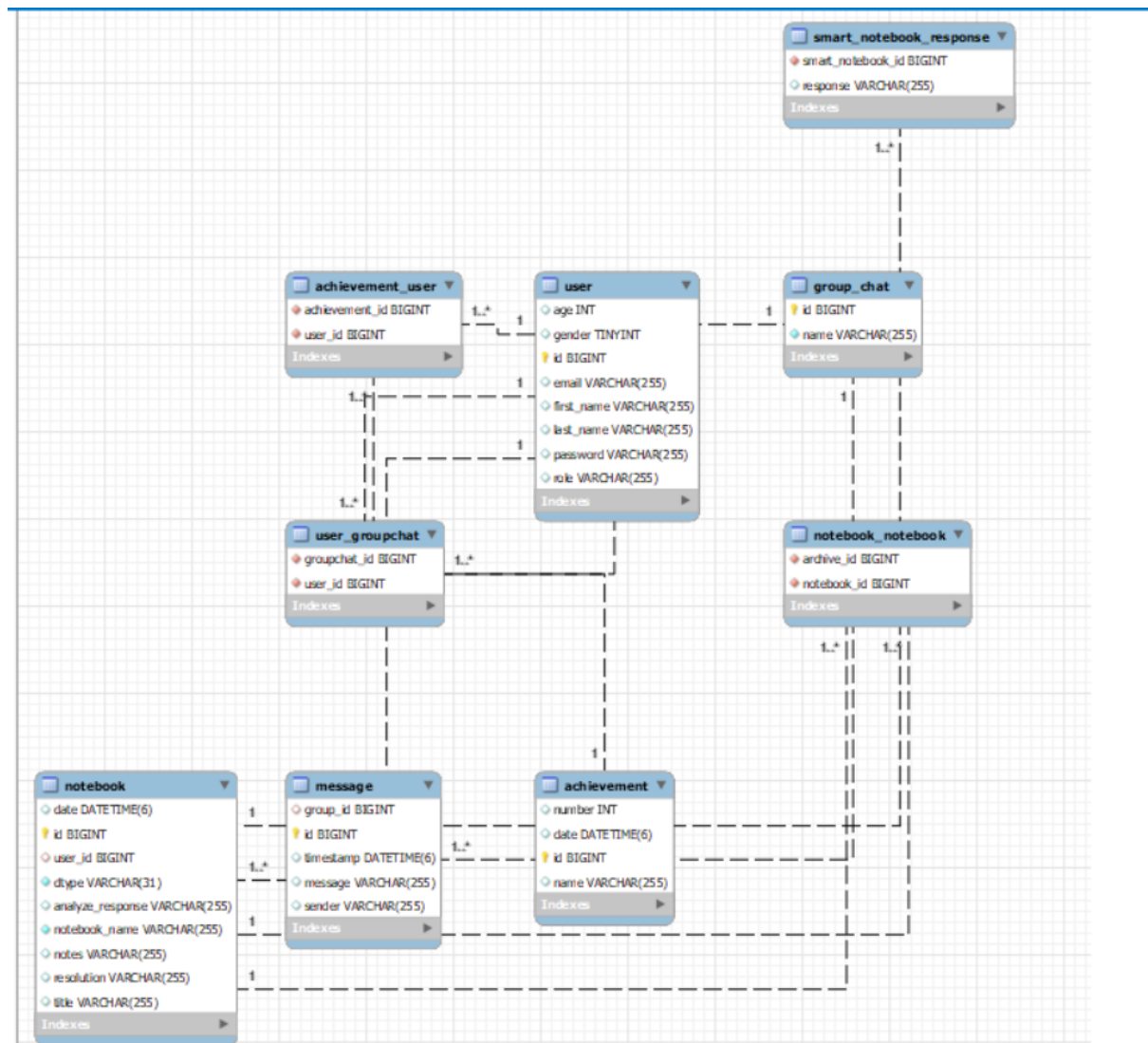Sequence diagram for login, register, forgot password and enter as a guest.

Start Interaction
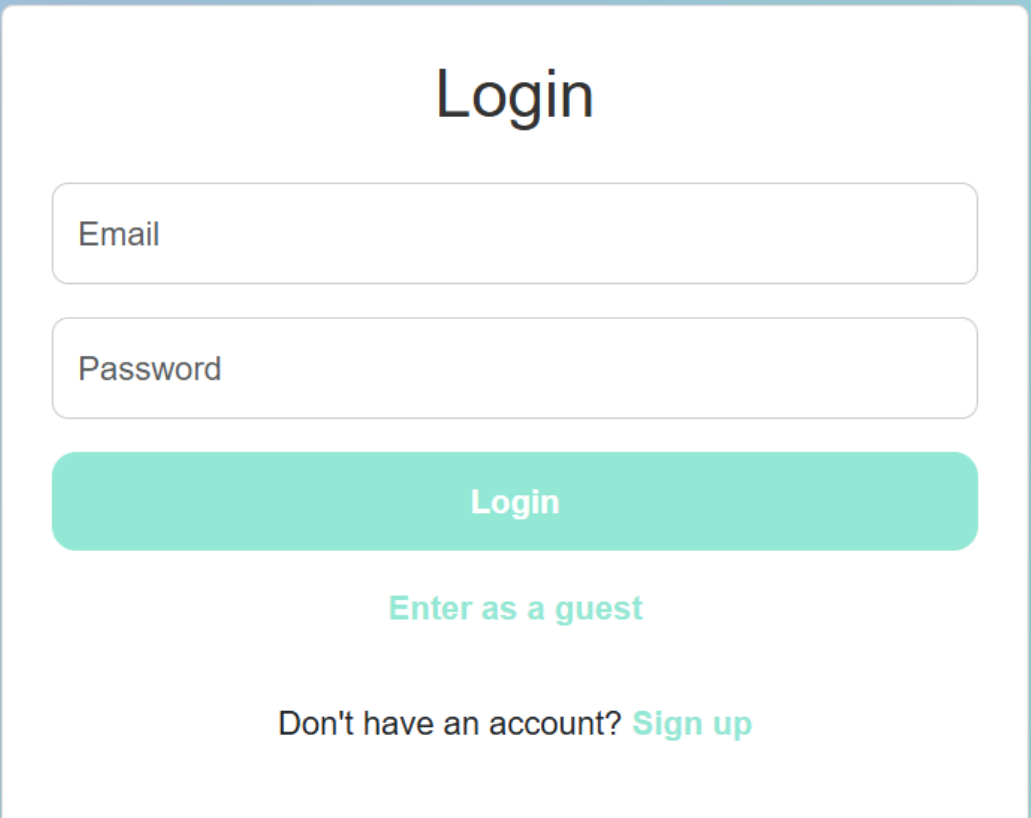
User → LoginInterface: **1** Select Login/Register/ForgotPassword/Guest

Login

LoginInterface → UserController: **2** login(email, password)
UserController → UserService: **3** geUserByEmailPass(email, password)
UserService → UserRepository: **4** findByEmail(email)
UserRepository → SQLDatabase: **5** query user
SQLDatabase → UserRepository: **6** user found
UserRepository → UserService: **7** user data

alt [Correct password]

UserService → UserController: **8** success
UserController → LoginInterface: **9** show access
LoginInterface → User: **10** Logged in

[Wrong password]

UserService → UserController: **11** wrong password
UserController → LoginInterface: **12** user null (suggest forgot password)
LoginInterface → User: **13** Password incorrect - forgot password?

Forgot Password

User → LoginInterface: **14** Forgot password request
LoginInterface → UserController: **15** forgotPassword(updateUTO)
UserController → UserService: **16** processPasswordReset(updateDTO)
UserService → UserRepository: **17** updateInfo(updateDTO)
UserRepository → SQLDatabase: **18** query user
SQLDatabase → UserRepository: **19** user found
UserRepository → UserService: **20** user data
UserService → UserController: **21** send reset email
UserController → LoginInterface: **22** confirmation
LoginInterface → User: **23** Reset email sent

Register

User → LoginInterface: **24** Register(User)
LoginInterface → UserController: **25** register(User)
UserController → UserService: **26** addUser(User)
UserService → UserRepository: **27** findByEmail(email   )
UserRepository → SQLDatabase: **28** query user
SQLDatabase → UserRepository: **29** no user
UserRepository → UserService: **30** user not found
UserService → UserRepository: **31** save new user
UserRepository → SQLDatabase: **32** insert new user
SQLDatabase → UserRepository: **33** inserted
UserRepository → UserService: **34** confirmation
UserService → UserController: **35** registration success
UserController → LoginInterface: **36** show success
LoginInterface → User: **37** Registration complete

Guest Access

User → LoginInterface: **38** Enter as Guest
LoginInterface → UserController: **39** guestAccess()
UserController → LoginInterface: **40** proceed without auth
LoginInterface → User: **41** Guest access granted

Class Diagram

Data Model

System Testing

Login feature in which you can login with email and password

You need valid email and password, or you will get validation error.

Login

test@gmail.com

••••••••

Login

Invalid credentials

Forgot password

Enter as a guest

Don't have an account? **Sign up**

Login

admin@gmail.com

••••••••

Login

Enter as a guest

Don't have an account? **Sign up**

Planify   Logout   Notebooks   Group   Dashboard   Welcome, **admin test**

Welcome to home!

You can add notebooks.

Your Notebooks

You have no notebooks yet.

+ Create Notebook

Create Notebook   ✕

Title

[ ]

Notes

[ ]

Resolution

[ ]

Cancel   **Create Notebook**

Name and resolution must be not blank or a pop up will occur:

localhost:3000 afișează mesajul

Notebook name and resolution are required.

OK
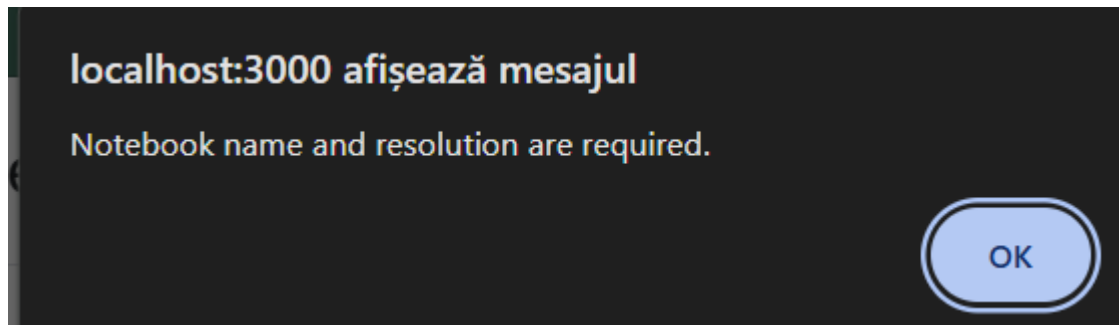
If completed the notebook will successfully be added.
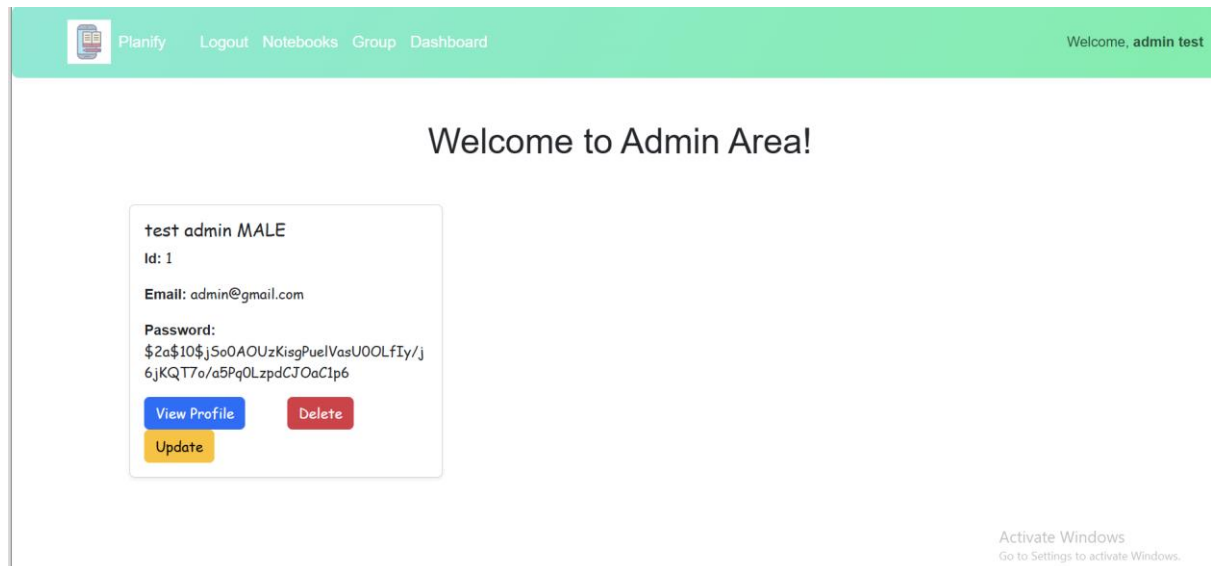
Your Notebooks

Notebook #1

Title: Test

Notes: Hellou
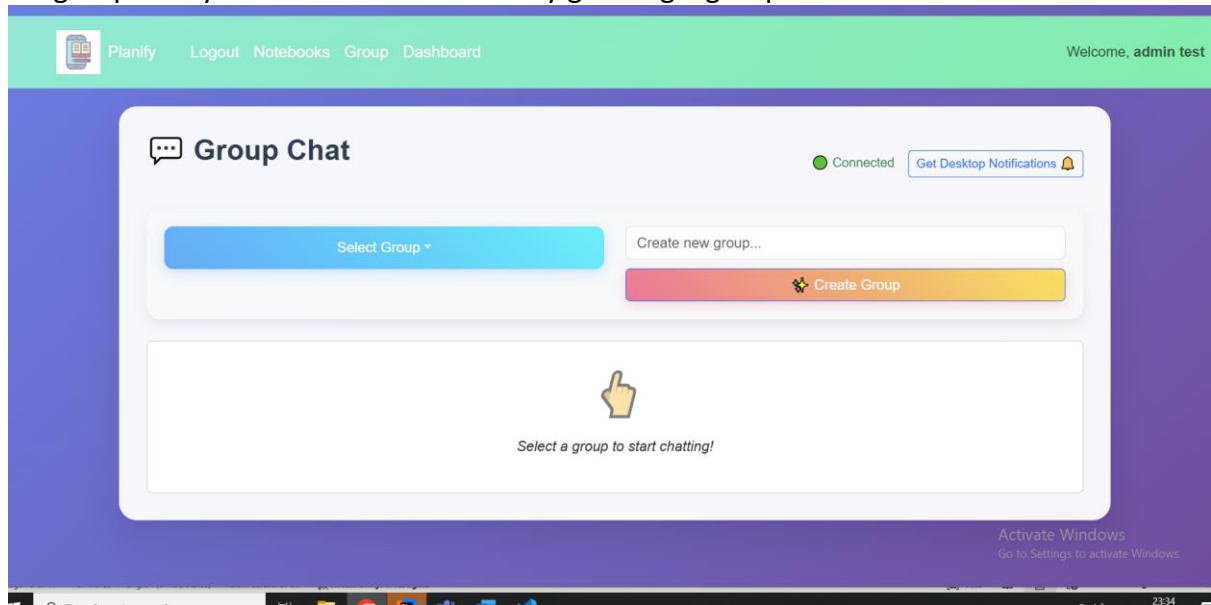
Resolution: This will be great!

Update

Dashboard for admins only.

Planify   Logout Notebooks Group Dashboard   Welcome, **admin test**

Welcome to Admin Area!

test admin MALE

Id: 1

Email: admin@gmail.com

Password:
$2a$10$jSo0AOUzKisgPuelVasU0OLfIy/j
6jKQT7o/a5Pq0LzpdCJOaC1p6

View Profile   Delete

Update

Activate Windows
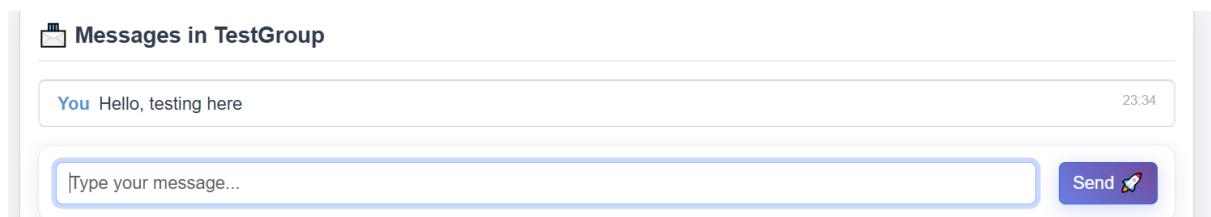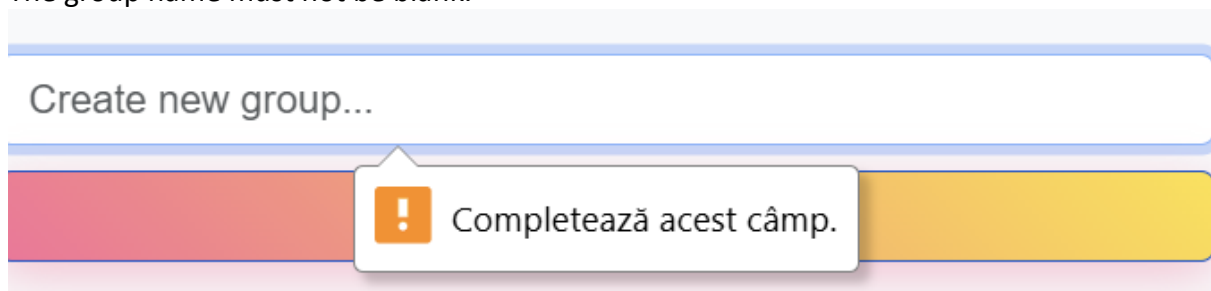Go to Settings to activate Windows.

On group chat you can chat with others by greating a group!

The group name must not be blank.

Future Improvements

        The Planify app will be improved in several practical ways.

First, an AI tool will be added to help analyze content. For example, it can summarize notes, detect the tone of messages, or suggest better names for notebooks. This will make the app smarter and more useful when handling content.

        Security will be upgraded by using Spring Security. This means users will log in more safely, and only authorized users will be able to access certain parts of the app. The system will use secure tokens and roles to manage who can do what.

        The chat feature will also be expanded. Users will be able to chat one-on-one in addition to group chats. There will also be dedicated groups people can join, making communication more organized and private.

        The user interface will be improved to make the app easier and more pleasant to use. The homepage will become more useful by showing relevant content like tips, updates, or data pulled from online APIs.

        Achievements will become more meaningful. The app will track what users accomplish and rank them based on their activity. This adds a competitive element and encourages users to stay active.

        Finally, the login system will be expanded with new options, including OCR (Optical Character Recognition). This means users might be able to scan something like an ID or a QR code to log in more easily.

These updates aim to make Planify more secure, smarter, easier to use, and more fun.

Bibliography

https://spring.io/
https://www.baeldung.com/spring-pathvariable
https://www.geeksforgeeks.org/spring/
https://chatgpt.com/
https://react-bootstrap.netlify.app/
https://refactoring.guru/design-patterns
https://www.geeksforgeeks.org/software-design-patterns/
https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS
https://www.techtarget.com/whatis/definition/monolithic-architecture
https://www.geeksforgeeks.org/microservices/
https://react.dev/reference/react/createContext
https://legacy.reactjs.org/
https://www.w3schools.com/react/react_useeffect.asp
https://www.w3schools.com/css/