Software Engineering

2023./2024.

CookBooked: Culinary Recipe Sharing Platform

Documentation, Rev. *1*

Group: *ByteTheRecipe*

Leader: *Arthur Maurer*

Date: *19. 10. 2023.*

Teacher: *Nikolina Frid*

Contents

1 Documentation change log **3**

2 Project Assignment Description **5**

3 Software specification **6**
3.1 Functional requirements 6
3.1.1 Use cases 8
3.1.2 Sequence diagrams 15

4 System architecture and design **17**
4.1 Database 17
4.1.1 Table description 18
4.1.2 Database diagram 18
4.2 Class diagram 19
4.3 State machine diagram 19
4.4 Activity diagram 19
4.5 Component diagram 19

5 Implementation and user interface **26**
5.1 Tools and technologies 26
5.2 Software testing 27
5.2.1 Component testing 27
5.2.2 System testing 30
5.3 Deployment diagram 31
5.4 Deployment instructions 32

6 Conclusion and future work **33**

References **34**

Index of figures **35**

Appendix: Group activity **36**

1. Documentation change log

Continuous updating

Rev.	Change description	Authors	Date
0.1	Changed basic texts (such as project name, header and footer settings)	Rufat Ismayilov	19.10.2023
0.2	Project Assignment Description	Arthur Maurer	20.09.2023
0.5	Functional Requirements	Rufat Ismayilov, Murad Ibrahimov	02.11.2023
0.6	Use Cases	Arthur Mauerer	05.11.2023
0.8			
0.10			
0.11			
0.12.1			
0.12.2			
1.0			
1.1			
1.2			
1.3			

Continued on next page

Continued from previous page

Rev.	Change description	Authors	Date
1.5			
1.5.1			
2.0			

There must be major revisions of documents 1.0 and 2.0 at the end of the first and second cycles. Between these revisions, there may be minor revisions depending on how the document is updated. It is expected that after each significant change (addition, modification, removal of parts of the text and accompanying graphic content) of the document, this will be recorded as a revision. For example, revisions within the first cycle will be labeled 0.1, 0.2,. . . , 0.9, 0.10, 0.11 .. until the final revision of the first cycle 1.0. In the second cycle, revisions 1.1, 1.2, etc. are continued.

2. Project Assignment Description

CookBooked is an innovative platform aimed at cooking enthusiasts, offering a space to share, discover, and discuss culinary recipes. The goal is to create a community where users can not only find recipes but also connect with their authors for more personalized exchanges.

To fully utilize the platform's features, users must register by providing a valid email address. Once registered, they can post their own recipes, interact with other users via messages, chats, and video calls, and follow their favorite recipe authors.

This project offers added value by creating a unique space for cooking enthusiasts to share and discover new recipes, as well as establish meaningful connections with others.

There are several similar apps and websites like Allrecipes or Food52. However, CookBooked sets itself apart with its real-time communication features and author-following system.

- Cooking enthusiasts
- Culinary bloggers
- Amateur and professional chefs

Users can customize their profiles, adjust their communication preferences, and choose the authors they wish to follow.

This project covers the development of either a web or mobile application, with features ranging from recipe publication to real-time communication among users.

- Integration with social media platforms
- Ingredient purchasing features
- Online cooking classes

The backend will be developed using object-oriented programming languages like Java or Python, and the frontend will be achieved using HTML, CSS, and JavaScript, all in compliance with GDPR regulations.

3. Software specification

3.1 Functional requirements

Stakeholders:

1. Recipe Creators
2. Developers and Designers
3. System Administrators
4. Content Moderators
5. Chat Services

Actors and their functional requirements:

1. Recipe Creators can:
   (a) Sign up using a valid email address to unlock all platform features.
   (b) Create and edit their own profile.
   (c) Create recipes, view other recipes, communicate with recipe authors when already logged in.
   (d) View recipes based on categories, cuisine types, or specific ingredients without registering.
   (e) Post cooking and baking recipes with details such as title, ingredients, preparation steps, cooking time, and tags.
   (f) Add images and videos related to the recipe.
   (g) Respond to messages, comments, and interactions from users interested in their recipes.
   (h) Manage personal information in private profiles, including communication preferences and notifications settings.
   (i) Like, comment on, and save recipes for future reference.
   (j) Follow favorite recipe authors to receive updates on new recipes.
2. Chat Services can:
   (a) Add and remove contacts.

(b) Users should be able to send text messages to individuals or groups in real-time.
(c) Users should be able to make voice calls and video calls to other users.

3. System Administrators can:

(a) Manage user accounts, including registration, account suspension, and account deletion.
(b) Handle user-related issues and inquiries.
(c) Delete inappropriate or violating recipes based on platform guidelines.

3.1.1 Use cases

Use case description

UC01 - User Registration

- **Main participant:** Recipe Creators
- **Goal:** To create a new user account
- **Participants:** Recipe Creators, System
- **Prerequisites:** The user must have access to the internet and an email address.
- **Description of the basic course:**
  1. Recipe Creator selects the option to create a new account.
  2. Recipe Creator provides required personal details.
  3. Recipe Creator sets a username and password.
  4. System sends a verification email to the Recipe Creator.
  5. Recipe Creator verifies the email address by clicking on the verification link.
- **Description of possible deviations:** 2.a If the user provides an email that is already in use:
  1. System prompts the user to provide a different email address.

UC02 - Login

- **Main participant:** Recipe Creators
- **Goal:** To login to an existing account on the platform
- **Participants:** Recipe Creators, System
- **Prerequisites:** The user has to have a registered account
- **Description of the basic course:**
  1. The user enters username and password

2. Login details are correct
3. Access to the account and application features are granted

- **Description of possible deviations:**
  - 1.a Entered username or password are invalid
    1. The user tries to login with the correct username and password
    2. The user cancels the login process

UC03 - Communication

- **Main participant:** Registered User
- **Goal:** To facilitate user interaction with recipe authors
- **Participants:** Registered User, Recipe Author
- **Prerequisites:** Recipe author has enabled communication options
- **Description of the basic course:**
  1. The user selects the communication option on a recipe page.
  2. The user chooses between messaging, chat, or video call.
  3. The platform checks the author's availability.
  4. Communication is established based on the author's set hours.
  5. The user interacts with the recipe author.
- **Description of possible deviations:**
  - 4.a Author not available for communication:
    1. The platform informs the user of the next available time.
    2. The user schedules a future interaction or leaves a message.

UC04 - Editing profile

- **Main participant:** Registered User
- **Goal:** To allow users to manage their personal and communication preferences

- **Participants:** Registered User
- **Prerequisites:** User is logged into their profile
- **Description of the basic course:**
  1. The user accesses their profile settings.
  2. The user updates their personal information.
  3. The user sets their communication preferences.
  4. The user configures their notification settings.
  5. The user saves the changes to their profile.
- **Description of possible deviations:**
  - 5.a User attempts to save invalid data:
    1. The platform displays a validation error message.
    2. The user corrects the data and saves again.

UC05 - Platform Administration

- **Main participant:** System Administrator
- **Goal:** To manage the platform's content and user activities
- **Participants:** System Administrator
- **Prerequisites:** System Administrator is logged into the administrative panel
- **Description of the basic course:**
  1. The administrator logs into the administrative panel.
  2. The administrator reviews recent activities and content submissions.
  3. The administrator manages user accounts and privileges.
  4. The administrator updates or deletes content that violates platform policies.
  5. The administrator logs out of the administrative panel.
- **Description of possible deviations:**
  - 4.a Content is reported for policy violation:

1. The platform notifies the administrator of the report.
2. The administrator reviews the reported content.
3. The administrator takes appropriate action according to platform policies.

UC06 - Recipe Submission

- **Main participant:** Registered User
- **Goal:** To publish a new cooking or baking recipe on the platform
- **Participants:** Registered User
- **Prerequisites:** User is logged into their account
- **Description of the basic course:**
  1. The user navigates to the recipe creation section.
  2. The user fills out the recipe details form (title, ingredients, steps, etc.).
  3. The user uploads images or videos of the recipe.
  4. The user submits the recipe for publication on the platform.
  5. The recipe becomes publicly available on the platform.
- **Description of possible deviations:**
  - 3.a Image or video upload fails:
    1. The platform shows an error message.
    2. The user is prompted to try uploading again.

UC07 - Search

- **Main participant:** Any users
- **Goal:** To search for profiles or recipes
- **Participants:** All users
- **Prerequisites:** The user must have access to internet connection
- **Description of the basic course:**

1. The user selects the search bar
2. The user searches for recipes or profiles
3. The user finds the desired recipe or profile

- **Description of possible deviations:**
  - 3.a The user does not find the recipe/profile they are looking for
    1. The user keeps searching
    2. The user stops searching

UC08 - Filtered recipe search

- **Main participant:** Any users
- **Goal:** To view recipes based on chosen filters
- **Participants:** All users
- **Prerequisites:** The user must choose at least one option given in the filter list
- **Description of the basic course:**
  1. The user clicks on the filter button
  2. The user chooses one or more options
  3. The user finds the desired recipe
- **Description of possible deviations:**
  - 3.a The user does not find the recipe with the filters they are looking for
    1. The user changes the filter options and keeps searching
    2. The user stops searching

UC09 - Recipe interaction

- **Main participant:** Registered users
- **Goal:** To be able to like, comment and save recipes
- **Participants:** Registered users
- **Prerequisites:** The user must be logged in to their account

- **Description of the basic course:**
    1. The user can save, leave a comment and like recipes of their choice
- **Description of possible deviations:**
    - 1.a The user is not logged in and can not interact with the recipe
        1. The user logs in to their account
        2. The user chooses not to interact with the recipe

UC10 - Following profiles

- **Main participant:** Registered users
- **Goal:** To follow profiles and receive updates from them
- **Participants:** Registered users
- **Prerequisites:** The user must be logged in to their account
- **Description of the basic course:**
    1. The user searches the profile they want to follow
    2. The user clicks on the follow button and follows the profile
- **Description of possible deviations:**
    - 1.a The user does not find the profile they searched for
        1. The user keeps searching for the profile
        2. The user stops searching

Use case diagrams

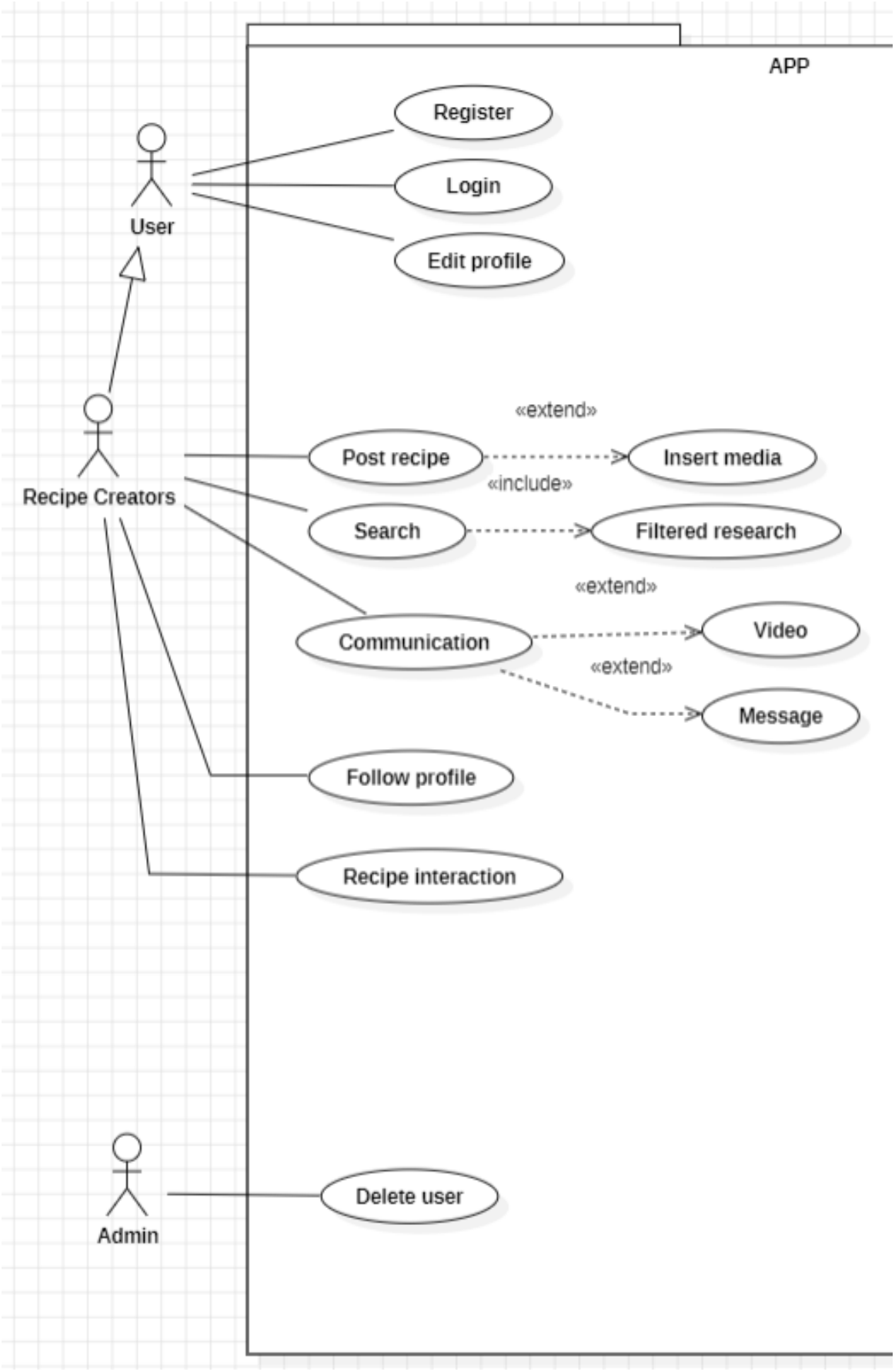

Figure 3.1: Enter Caption

3.1.2 Sequence diagrams

- Figure depicts the user registration process. Initially, the user is required to fill out a registration form, entering details such as name, surname, email address, profile picture, username, and password. The system then proceeds to verify whether the email or username is already in use and checks if the password adheres to the established criteria. If any of these checks fail, the user receives an error message and is prompted to provide a different email or username and/or select a new password. Once the registration is successfully completed, the user's profile is added to the database. Subsequently, a confirmation message is displayed to the user, and a confirmation email is sent to the provided email address.

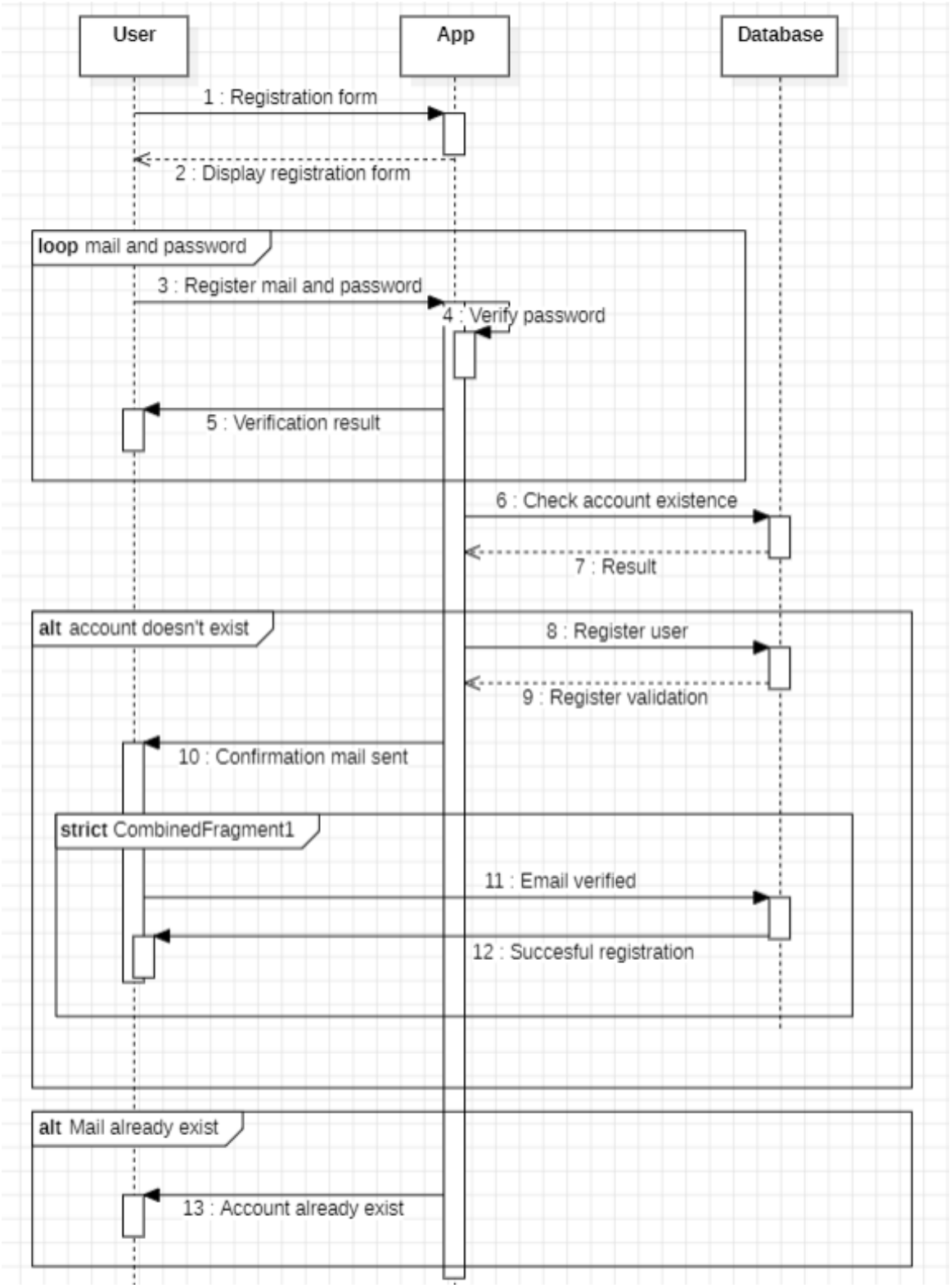

Figure 3.2: Enter Caption

4. System architecture and design

The app is set up with three main parts, making it easy to manage. We've organized it like a team where the frontend (what you see) and the backend (what handles data) work together.

The frontend, made with Java using Android Studio, takes care of how things look and talks to the backend using Firebase's Java tools. For storing and getting both the text details and images of recipes, we're using Firebase Storage.

The backend, built with Firebase Functions using Java, is the brain of the operation. It does the thinking for business rules, manages who's logging in, and directly talks to Firebase Storage for text and image info.

Think of this setup like having a central hub (Firebase Storage) that keeps all the recipe info and pictures. The part you interact with is user-friendly, designed for easy recipe browsing.

To keep things safe when talking between the user side and the brain, we're using HTTPS. When you do something on the app, like saving a new recipe, it sends a request to the brain (backend). Real-time updates make sure everything stays up to date.

For what's needed, the user side works on any Android device. On the brain side, it uses Firebase Functions with the Firebase Java toolkit.

4.1 Database

part of 1st revision

We're using Firebase Realtime Database for our recipe app. It's like a smart, real-time database in the cloud. Our data, like recipe details and user info, is stored neatly in a structure that's easy to manage. We keep recipe images in Firebase Storage and save

their web links in the database.

4.1.1 Table description

Each table must be described according to the template. On the left is the exact name of the variable in the database, in the middle is the data type, and on the right is the description of the variable. Highlight the primary key in light green and the foreign key in blue

Custom User		
ID	INT	id user
email	VARCHAR	user email
password	VARCHAR	user password

Recipe		
ID	INT	id recipe
dataTitle	VARCHAR	recipe title
dataImage	VARCHAR	recipe image
dataPerson	VARCHAR	number of person for the recipe
ingredientList	List(VARCHAR)	List of ingredient(s) use for the recipe
dataDescription	List(VARCHAR)	List of step(s) use for the recipe

4.1.2 Database diagram

A database diagram needs to be inserted in this subchapter. Primary and foreign keys must be highlighted and tables linked. The database needs to be normalized. Remind yourself of the "Databases" course.

4.2 Class diagram

4.3 State machine diagram

4.4 Activity diagram

4.5 Component diagram

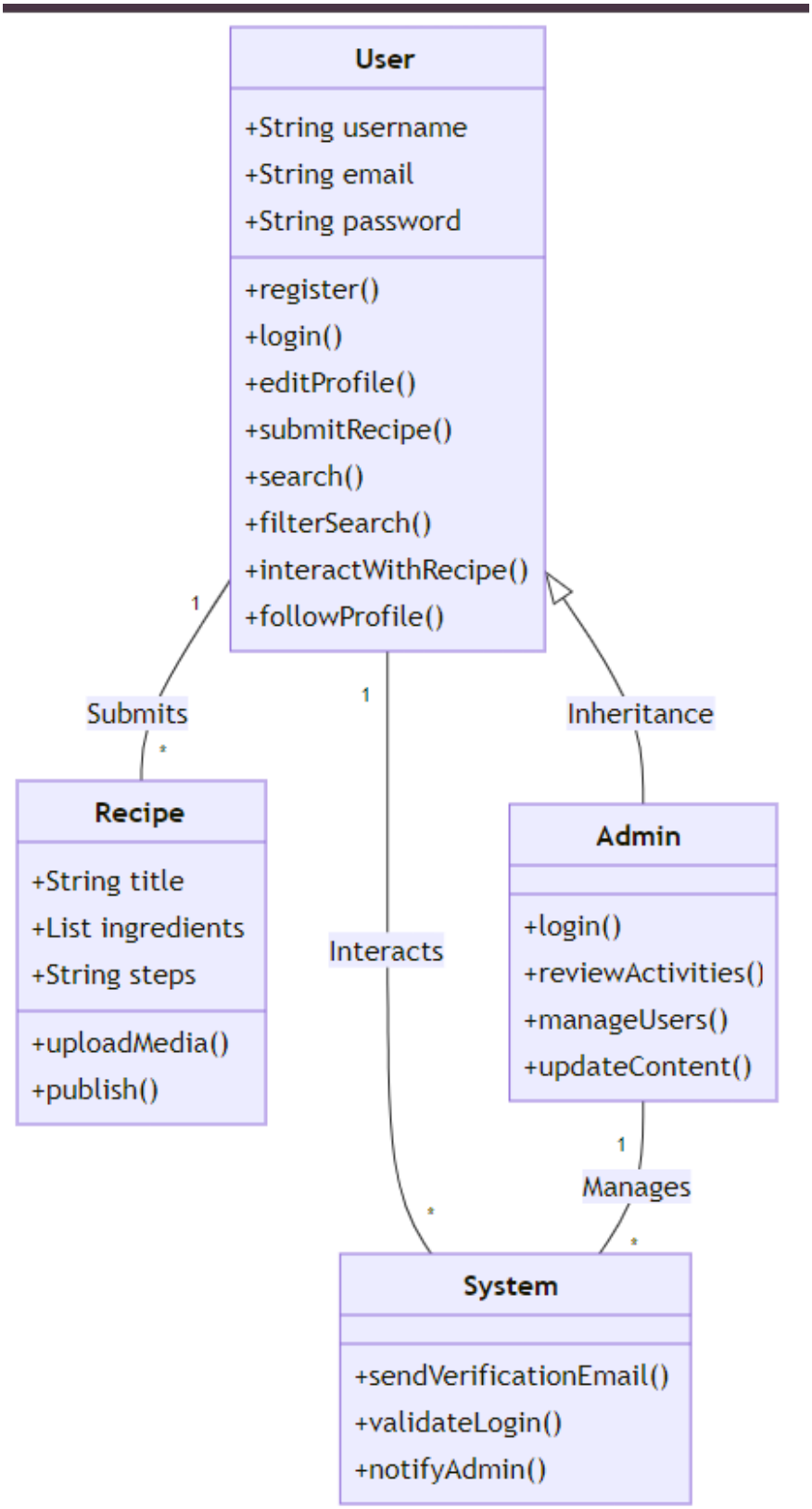

Figure 4.1: Class diagram

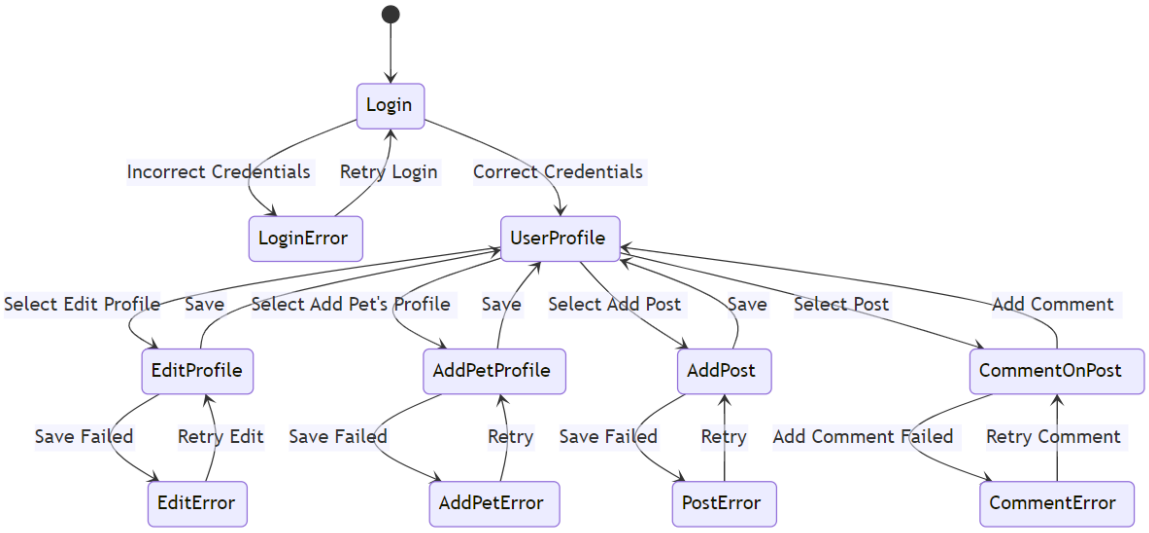

Figure 4.2: State machine diagram

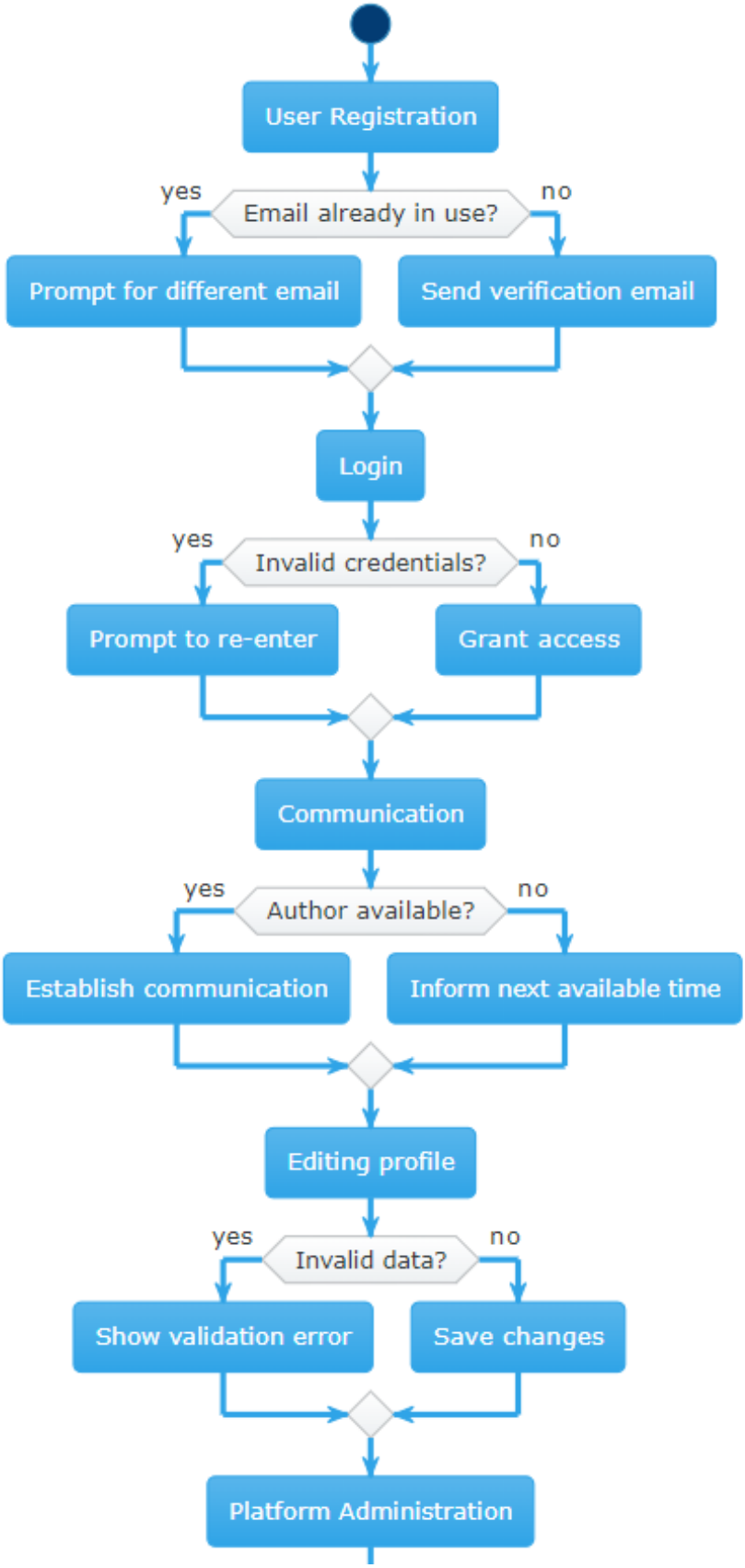

Figure 4.3: Activity diagram

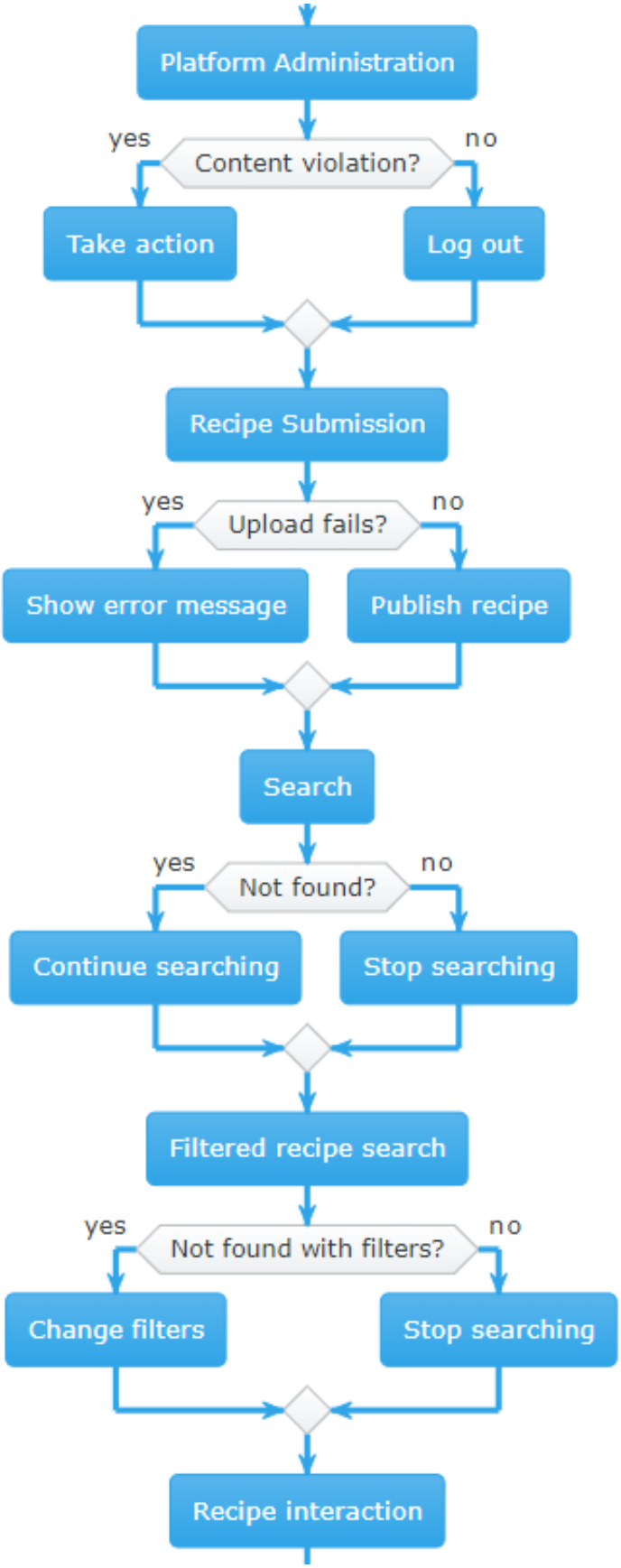

Figure 4.4: Activity diagram

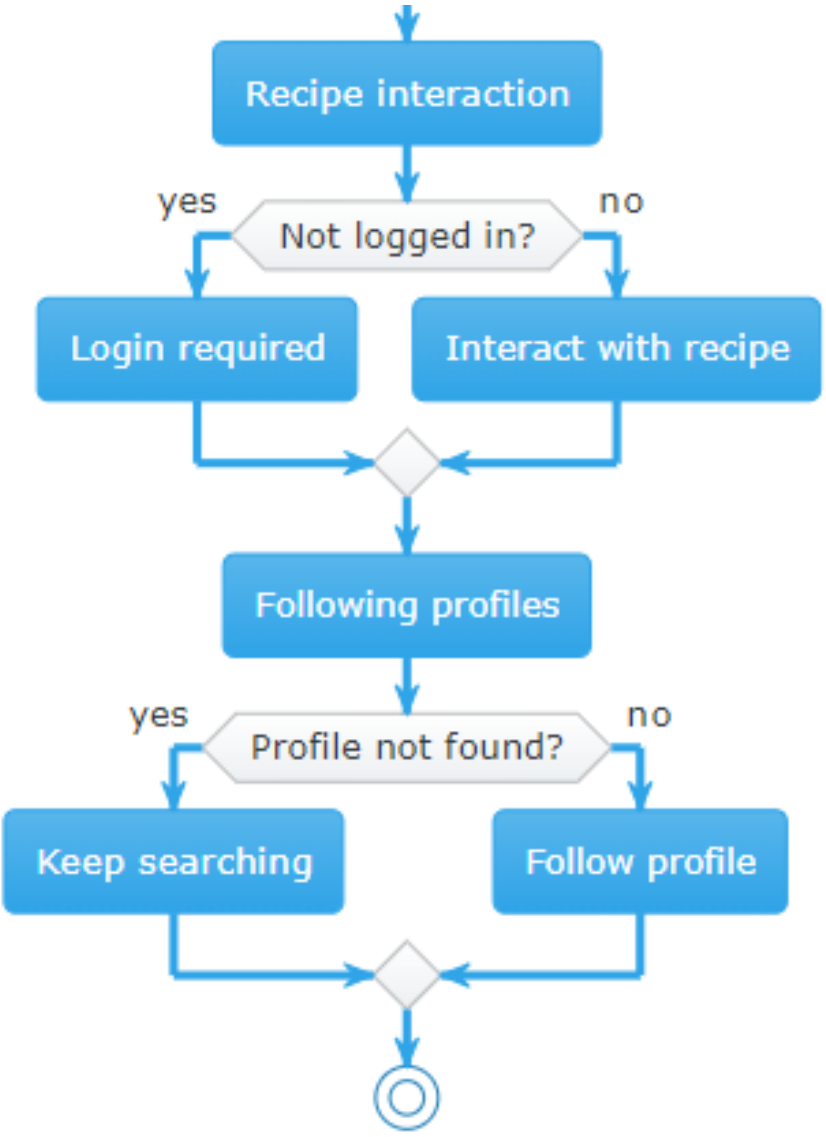

Figure 4.5: Activity diagram

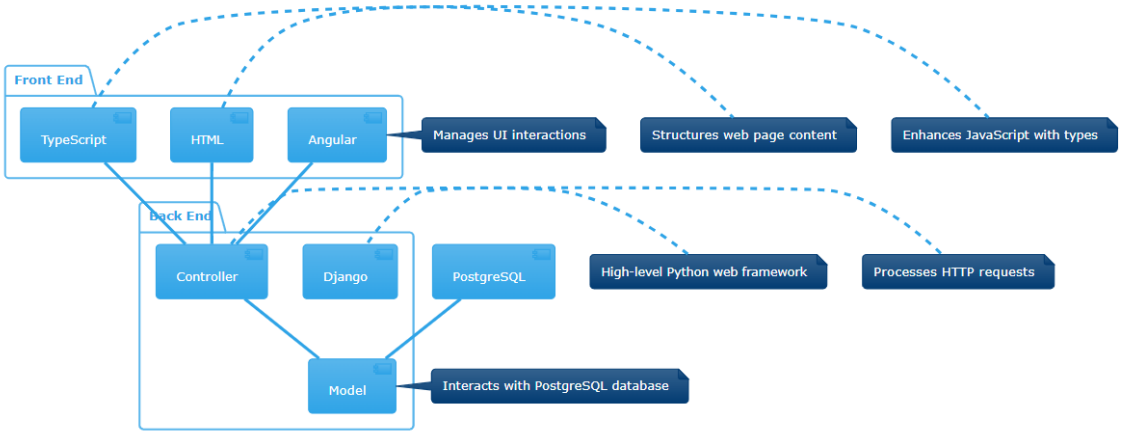

Figure 4.6: Component diagram

5. Implementation and user interface

5.1 Tools and technologies

part of the 2nd revision

List in detail all technologies and tools used in the development of documentation and applications. Briefly describe them, and state their meaning and place of application. For each of the listed tools and technology it is necessary to ***specify internet link*** *where you can download or learn more about them.*

5.2 Software testing

part of the 2nd revision

In this chapter it is necessary to describe the implementation of testing of implemented functionalities at the level of components and at the level of the whole system with the presentation of selected test cases. Students should examine core functionality and boundary conditions.

5.2.1 Component testing

Test Case: Register User : Success

Input:

```
Username: "test@example.com"
Password: "password123"
```

Expected Result:

```
Registration succeed.
```

Actual Result:

```
Registration succeeded.
```

Status: Passed

```
public void testRegisterUserSuccess() {
        when(mockAuth.createUserWithEmailAndPassword(any(), any()))
                .thenReturn(mockTask(true));

        registerActivity.registerUser("test@example.com", "password123");
        assertTrue("User registration should succeed", registerActivity.isSuccessful());
    }
```

Test Case: Login User with Email Verification : Success

Input:

Email: "test@example.com"
Password: "password123"

Expected Result:

Login should succeed after email verification.

Actual Result:

Login succeeded after email verification.

Status: Passed

```
public void testLoginUserWithEmailVerification() {
        FirebaseUser mockUser = mock(FirebaseUser.class);
        when(mockAuth.getCurrentUser()).thenReturn(mockUser);

        when(mockAuth.signInWithEmailAndPassword(any(), any()))
                .thenReturn(mockTask(true));
        when(mockUser.isEmailVerified()).thenReturn(true);

        loginActivity.loginUser("test@example.com", "password123");

        assertTrue(loginActivity.isSuccessful());
    }
    private Task<AuthResult> mockTask(boolean success) {
        Task<AuthResult> task = mock(Task.class);
        when(task.isSuccessful()).thenReturn(success);
        return task;
    }
```

Test Case: Login Failure : Incorrect Password
Input:

Email: "test@example.com"
Password: "incorrect_password"

Expected Result:

Login should fail with an error message.

Actual Result:

Login failed with the expected error message.

Status: Passed

```
public void testLoginFailureIncorrectPassword() {

    when(mockAuth.signInWithEmailAndPassword(any(), any()))
            .thenReturn(mockTask(false, new FirebaseAuthInvalidCredentialsException("message", "errorCode")));

    loginActivity.loginUser("test@example.com", "incorrect_password");

    assertFalse(loginActivity.isSuccessful());
}

private Task<AuthResult> mockTask(boolean success, Exception exception) {
    Task<AuthResult> task = mock(Task.class);
    when(task.isSuccessful()).thenReturn(success);
    when(task.getException()).thenReturn(exception);
    return task;
}
```

Test Case: Login Failure : Weak Password

Input:

Email: "test@example.com"

Password: "weak"

Expected Result:

Login should fail with an error message indicating that the passwor

Actual Result:

Login failed with the expected error message.

Status: Passed

```
public void testLoginFailureWeakPassword() {

    when(mockAuth.signInWithEmailAndPassword(any(), any()))
            .thenReturn(mockTask(false, new FirebaseAuthWeakPasswordException("errorCode", "message")));

    loginActivity.loginUser("test@example.com", "weak");

    assertFalse(loginActivity.isSuccessful());
}
private Task<AuthResult> mockTask(boolean success, Exception exception) {
    Task<AuthResult> task = mock(Task.class);
    when(task.isSuccessful()).thenReturn(success);
    when(task.getException()).thenReturn(exception);
    return task;
}
```

5.2.2 System testing

The system test should be performed and described using the Selenium footnote `https://www.seleniumhq.org/` *framework. Develop a* ***minimum 4 test cases*** *that will examine regular cases, boundary conditions, and call functionality that is not implemented / cause an error to see how the system responds when something is not fully realized. The test case should consist of an input (eg username and password), the expected output or result, the test step and the output or result obtained.*

The creation of test cases using the Selenium framework can be performed using one of the following two tools:

- *browser extension* ***Selenium IDE*** *- recording user actions for automatic exam repetition*
- ***Selenium WebDriver*** *- support for writing exams in Java, PHP languages using a special programming interface*

Details on the use of the Selenium tool will be presented in a special lecture during the semester.

5.3 Deployment diagram

part of the 2nd revision

*You need to insert a **specification** layout diagram and describe it. It is possible to insert an instance layout diagram instead of a specification layout diagram, provided that this diagram better describes some important part of the system.*

5.4 Deployment instructions

part of the 2nd revision

In this chapter it is necessary to give instructions for deployment of the realized application. For example, for web applications, describe the process by which the source code leads to a fully set up database and server that responds to user queries. For a mobile application, the process by which the application is built and placed on one of the stores. For a desktop application, the process by which an application is installed on a computer. If mobile and desktop applications communicate with the server and / or database, describe the procedure for setting them up. When creating instructions, it is recommended that ***highlight installation steps using hints*** *and use* ***screenshots*** *as much as possible to make the instructions clear and easy to follow.*

The completed application must be running on a publicly available server. Students are encouraged to use one of the following free services: Amazon AWS, Microsoft Azure or Heroku. Mobile apps should be released on the F-Droid, Google Play or Amazon App Store.

6. Conclusion and future work

part of the 2nd revision

In this chapter it is necessary to write a review of the time of project assignment, what technical challenges have been identified, whether they have been solved or how they could be solved, what knowledge was acquired during project development, what knowledge would be especially needed for faster and better project implementation. and what would be the prospects for continuing work in the project team.

It is necessary to accurately list the functionalities that are not implemented in the realized application.

References

Continuous updating
List all references and literature that helped in the realization of the project.

1. Oblikovanje programske potpore, FER ZEMRIS, http://www.fer.hr/predmet/opp
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Index of figures

3.1 Enter Caption 14
3.2 Enter Caption 16

4.1 Class diagram 20
4.2 State machine diagram 21
4.3 Activity diagram 22
4.4 Activity diagram 23
4.5 Activity diagram 24
4.6 Component diagram 25

Appendix: Group activity

Meeting log

Continuous updating

It's necessary to frequently update the meeting log according to the template.

1. meeting
    - Date: in this format: January 19, 2024
    - Attendees: J.Doe, ...
    - Meeting subjects:
        - subject description
        - subject description
2. meeting
    - Date: in this format: January 19, 2024
    - Attendees: J.Doe, ...
    - Meeting subjects:
        - subject description
        - subject description

Activity table

Continuous updating

Note: Activity contributions should be entered as hours contributed by person and activity.

	Team lead	**Team member**	**Ime Prezime**	**Team member**	**Team member**	**Team member**	**Team member**
Project management							
Project task description							
Functional requirements							
Individual patterns description							
Patterns diagram							
Sequence diagram							
Other requirements description							
System architecture and design							
Database							
Class diagram							
State diagram							
Activity diagram							
Components diagram							
Used technologies and tools							
Solution testing							
Layout diagram							
Deployment instructions							
Meeting log							

Continued on next page

Continued from previous page

	Team lead	**Team member**	**Ime Prezime**	**Team member**	**Team member**	**Team member**	**Team member**
Conclusion and future work							
References							
Additional task examples							
Welcome page creation							
Database creation							
Connecting to the database							
back end							

Change log diagrams

part of the second revision

Import generated change log diagrams from GitLab to this chapter. Diagrams can be reached at GitLab at Repository/Contributors.