Software Engineering

2023/2024

Cookbooked

Documentation, Rev. *2*

Group: *Ariel*

Leader: Cédric LISMONDE

Date: 19th January 2024

Teacher: Nikolina FRID

Contents

1. Documentation change log

Rev.	Change description	Authors	Date
0.1	Project Description	Noëlie COMTE	22.10.2023
0.2	Refining Project Description, functional requirements during a team meeting	Noëlie COMTE, Sonia HAMDI, Florian PAILHE	23.10.2023
0.3	Use case description during a team meeting	Sonia HAMDI, Florian PAILHE	26.10.2023
0.4	Refining Use case description and use case diagram	Sonia HAMDI	27.10.2023
0.5	Software Specification Sequence diagrams	Cédric LISMONDE	05.11.2023
0.6	Architecture Table description and Class diagram during a team meeting	Noëlie COMTE, Sonia HAMDI, Florian PAILHE	12.11.2023

Continued on next page

Continued from previous page

Rev.	**Change description**	**Authors**	**Date**
0.7	Architecture Database Diagram	Lara FERNANDES	16.11.2023
0.8	Architecture and design description and Refining Class diagram	Noëlie COMTE, Sonia HAMDI, Lara FERNANDES, Théau MARGOUTI, Taylan ÜNVER	17.11.2023
0.9	Modification of use case description and use case diagrams	Sonia HAMDI	13.12.2023
1.0			
1.1	Refining use cases	Noëlie COMTE	20.12.2023
1.2	Refining sequence diagram	Cédric LISMONDE	16.01.2024
1.3	State machine diagram and activity diagram	Lara FERNANDES	18.12.2023
1.4	Component diagram	Sonia HAMDI	18.01.2024

Continued on next page

Continued from previous page

Rev.	**Change description**	**Authors**	**Date**
1.5	Tools and technologies	Noëlie COMTE, Taylan ÜNVER	10.01.2024
1.6	Software testing		
1.6.1	Component testing	Noëlie COMTE, Théau MARGOUTI	18.01.2024
1.6.2	System testing	Noëlie COMTE	19.01.2024
1.7	Deployment diagram	Lara FERNANDES	19.01.2024
1.8	Deployment instructions		
1.9	Conclusion and future work	CEDRIC LISMONDE, Noëlie COMTE, Théau MARGOUTI	17.01.2024
2.0			

2. Project assignment description

The objective of this project is to create and implement a website and everything surrounding it, whether it is the coding of the functionalities or the user interface.

CookBooked is a web platform that connects lovers of cooking and baking. CookBooked users will be able to share their favorite recipes, discover new ones, and interact with the recipe authors. This platform aims to connect all cooks while simplifying communication and sharing between users. The recipes are available to all visitors of the website, but registration is required to access all the features provided by the platform. To validate registration, users must input one valid e-mail address, a username, a password, a first name, and a family name which will appear on their profile. Once registered, users will be able to log in by entering their password as well as their username or email address.

Non-registered users can only browse recipes according to the categories available, types of cuisine, or specific ingredients. appetizers, starters, main courses, desserts, pastries, etc. constitute the different categories that can be found on the platform. Among the types of cuisine, users will find Mediterranean, Japanese, Indian, French, American and more...

People who register will be able to access all the functionalities of the platform. They will be able to post their cooking or baking recipes by adding a title to the recipe, and by specifying the various ingredients and the different steps of the recipe. The website will also let them add cooking times and tags such as vegetarian, vegan, gluten-free, etc... Recipe authors will even be able to add photos and videos to the recipes they share with the community.

Logged-in users can chat with other users. Recipe authors can choose if they are available for the various communication options available on the website. Users can communicate via messaging, integrated chat, and video calls. Authors can indicate their availability with precise days or times.

Registered users can also like a recipe, make comments, or save it for later. They can also follow the recipe authors to receive updates and new recipe notifications.

Registered users have a public and a private profile. The public profile cannot be modified and is accessible to other members of the CookBooked community. It contains all the recipes published by the registered user, the people he or she follows, and those who follow him or her as an author. Kind of like an Instagram profile, with recipes instead of posts.

The private profile, instead, can be modified. This allows the user to manage their personal information (first name, last name, username, profile picture, email...), and their communication preferences, but also to access their messaging notifications and updates related to recipes or authors.

The website is maintained by administrators, who are not considered as users. They can manage individual users and recipes. For example, they can change the category of a recipe, delete the recipe, or delete some messages in the chat. They can also temporarily or permanently ban a user for inappropriate activity.

The Cookbooked website respects data protection according to GDPR.

To program this site we had to choose the object-oriented language python or java. And then use CSS, HTML, or JavaScript for the front-end development of the website

Our group decided to program the site in Python and make the user interface in HTML or CSS.

The website must be compatible with different machines, such as mobile phones, tablets, and computers, to allow as many people as possible to use it.

3. Software specification

3.1 Functional requirements

Stakeholders:

1. Users interested in cooking
2. Administrator
3. Developers
4. Google Calendar and Google Meet API providers

Actors and their functional requirements:

1. Unregister user can:
   (a) Create an account to register
   (b) Browse recipes
      i. Search by categories (appetizers, main dishes, desserts...)
      ii. Search by cuisine types (Mediteranean, Japanese, French...)
      iii. Search by specific ingredients
   (c) Consult recipes
2. Register user can:
   (a) Log in
   (b) Post recipes
      i. Add a title
      ii. Add the category (Appetizers, main dishes, desserts...)
      iii. Specify ingredients
      iv. Specify preparation steps
      v. Add cooking time
      vi. Add tags (vegetarian, vegan, gluten-free...)
      vii. Add image(s) related to the recipe
      viii. Add video(s) related to the recipe
   (c) Watch recipes

i. Like a recipe
ii. Comment a recipe
iii. Save a recipe
iv. Interact with authors
    A. Messaging
    B. Chat
    C. Video call

(d) Watch authors profil
  i. Subscribe to users
     A. Receive notifications for new post and message
  ii. See all the posts of the author
  iii. Interact with the author and other users
     A. Messaging
     B. Video call
  iv. Access to his/her public profile

(e) Access and manage his/her private profile
  i. Manage his/her personal information (Username, Name, Surname, Profile Picture, E-mail address)
  ii. Set his/her communication preferences
  iii. Set his/her availability for video calls
  iv. See his/her notifications (messages and video call demands)
  v. Manage his/her posted recipes (Edit and Delete)

3. Administrators can:

  (a) Manage users
    i. Temporary ban users
    ii. Permanent ban users
    iii. Delete users
  (b) Manage recipe
    i. Change the category of a recipe
    ii. Edit a recipe
    iii. Delete a recipe
  (c) Manage chat
    i. Delete message

(d) Receive report from registered users

3.1.1 Use cases

Use case description

UC1-Register

- **Main participant:** Unregistered user
- **Goal:** Get access to the application
- **Participants:** Users
- **Prerequisites:** Unregistered
- **Description of the basic course:**
  1. User goes to the main page of the website
  2. Click on the Login part
  3. Click on Register
  4. The user inputs name, surname, email, picture, username and set a password
  5. System response (checking data)
  6. If the data are correct the account is created
- **Description of possible deviations:**
  - 6.a The username or email is already taken
    - 6.a.a Change your username
    - 6.a.b Go to the login page if it is your username
  - 6.b Incorrect password. Needs uppercase letter and number.
    - 6.b.a Change password

UC2-Login

- **Main participant:** Registered user
- **Goal:** Get access to everything on the website
- **Participants:** Users
- **Prerequisites:** Register
- **Description of the basic course:**
  1. Registered users go to the main page of the website
  2. Click to Login
  3. Enter their username and password
  4. System response(checking username and password)
  5. If the password corresponds to the username in the database, the user is redirected to the main page and can access his profile, see all the recipes, and be able to share and exchange with other users

- **Description of possible deviations:**
  - 3.a The user forgot his password
    - 3.a.a user clicks on ”forgotten password”, enters his email address, and changes his password
  - 5.a The username doesn’t exist
    - 5.a.a User tries to log in with another username
  - 5.b The password is not valid
    - 5.b.a User tries to log in with another password

UC3- Edit user profile

- **Main participant:** Registered user
- **Goal:** change information in a profile
- **Participants:** Users
- **Prerequisites:** user should be logged to the website
- **Description of the basic course:**
  1. By clicking from the main page of the website, the user can access his profile
  2. Click on the edit button on the profile page
  3. New page will be open with the edition possibility
  4. Change some information: name, surname, username, password, email, profile picture, setting communication preferences and availability
     (a) to change the password the user will have to type the old password first before writing a new one
  5. User can save his changes
  6. system response (saving the new data)
  7. If new data are saved, a pop-up message appears and the user is redirected to the main page
- **Description of possible deviations:**
  - 5.a The user forgot to save his changes
    - 5.a.a profile information will remain the same as before

UC4-Look at a recipe

- **Main participant:** Registered user, Unregistered user , Administrators
- **Goal:** Discover recipes
- **Participants:** Users
- **Prerequisites:** Go to the app

- **Description of the basic course:**
  1. User goes to the main page of the website
  2. Click on ”Search” and search for a recipe
  3. Then choose a recipe by clicking on
  4. Then user watches the recipe and can comment or like the recipe
- **Description of possible deviations:**
  - 2.a There are no recipes for your research
    - 2.a.a A message appears, the user can make another search
  - 1.a The user subscribed to another user and can see recipes directly on his main page
    - 1.a.a The user can click on one of these recipes to see it

<u>UC5- Post a recipe</u>

- **Main participant:** Registered user
- **Goal:** Share a recipe
- **Participants:** Users
- **Prerequisites:** User should be logged in and is on the main page
- **Description of the basic course:**
  1. Click on the ”Post a recipe” button
  2. Write a title for the recipe
  3. Add category, type, ingredients, preparation steps, cooking time, tags, image or video related to the recipe
  4. Post the recipe
  5. System response (load data recipe and display the recipe with all the other recipes)
- **Description of possible deviations:**
  - 4.a The user leaves before posting his recipe or click on the ”save in drafts” button
    - 4.a.a the post is saved in its drafts
  - 3.a The user didn’t choose one or several recipe information
    - 3.a.a a message is displayed to remind the user to fill in all the details
  - 3.b One ingredient or tag doesn’t exist
    - 3.b.a The User can add it by clicking on a button and writing the missing data

<u>UC6-React to a recipe</u>

- **Main participant:** Registered user
- **Goal:** Like, comment a recipe
- **Participants:** Users
- **Prerequisites:** Looking at a recipe
- **Description of the basic course:**
  1. The user can click on the like button
  2. The user can add a comment to a recipe in the appropriate section
     (a) User click on the ”add a comment” button
     (b) User write a comment respecting the chat policy
     (c) User clicks on ”post comment”
- **Description of possible deviations:**

2.(a).a The comment doesn’t respect our chat policy

2.(a).a.a Another user can report the comment and the comment will be deleted until the administrator determines if the comment respects the chat policy

UC7- Search users

- **Main participant:** Registered user
- **Goal:** see the profile and recipes of this user
- **Participants:** Users
- **Prerequisites:** User is on the main page
- **Description of the basic course:**
  1. Click on the search bar
  2. Write the username of the person
  3. Search among the proposed users
  4. Click on the researched user to access to his profile page
- **Description of possible deviations:**

4.a The user doesn’t exist in the database

4.a.1 A unsuccessful research message is return

3.a You don’t know his username

3.a.1 Type some keywords related to his recipes to find the user directly with his posts

UC8- Subscribe to users

- **Main participant:** Registered user,
- **Goal:** Connect with other people and receive notifications about future posts

- **Participants:** Users
- **Prerequisites:** Register and on a user page
- **Description of the basic course:**
  1. Click on the ”subscribe” button
  2. Wait until the user answers to request
- **Description of possible deviations:**
  - 2.a The user didn’t answer
  - 2.b The user refused the request

UC9- Set communication preferences

- **Main participant:** Registered user
- **Goal:** avoid inappropriate content
- **Participants:** Users
- **Prerequisites:** Register and go to profile page
- **Description of the basic course:**
  1. Click on set communication preferences button
  2. Choose between the proposed sections and click on the one(s) you want

UC10- Set availability for video calls

- **Main participant:** Registered user
- **Goal:** avoid inappropriate content
- **Participants:** Users
- **Prerequisites:** Register and on profile page
- **Description of the basic course:**
  1. Click on set availability button
  2. Click on the times where you are available on the calendar section
  3. The section time will change of color in the calendar

UC11- Messaging other users

- **Main participant:** Registered user
- **Goal:** Talk to other users
- **Participants:** Users
- **Prerequisites:** Register and go to other user profile
- **Description of the basic course:**
  1. Click on message
  2. Write a message

3. Send a message

- **Description of possible deviations:**

  2.a The message was not sent

    1. Reconnect to the web page (maybe problem of connection)
    2. Send again the same message

UC12- Managing chat

- **Main participant:** Administrators
- **Goal:** Check if the chat rules are respected
- **Participants:** Administrators
- **Prerequisites:** Be an administrator
- **Description of the basic course:**
  1. Manage chat and delete message which are inappropriate
- **Description of possible deviations:**

  2.b This chat as already been deleted

UC13- Managing recipes

- **Main participant:** Administrators
- **Goal:** Check if the content of a recipe is not appropriate
- **Participants:** Administrators
- **Prerequisites:** Be an administrator
- **Description of the basic course:**
  1. Change the category of a recipe or delete a recipe
  2. Manage recipes and delete recipes which are inappropriate

UC14- Managing users

- **Main participant:** Administrators
- **Goal:** Manage the behaviour of users
- **Participants:** Administrators
- **Prerequisites:** Be an administrators
- **Description of the basic course:**
  1. Temporary ban users or permanent ban
- **Description of possible deviations:**

  1.a This user is already banned

UC15- Report users

- **Main participant:** Registered user
- **Goal:** avoid inappropriate content
- **Participants:** Users
- **Prerequisites:** Register
- **Description of the basic course:**
  1. if a user sees inappropriate content in a post, whether in chat or in the recipe itself, they can click on it and report it
  2. a notification is send to administrators

UC16- Call other users

- **Main participant:** Registered user
- **Goal:** Talk to other users
- **Participants:** Users
- **Prerequisites:** Register and go on other user profile
- **Description of the basic course:**
  1. Click on call
  2. Wait for the other user to answer
  3. If the other user accepts the call, the call can start
- **Description of possible deviations:**
  3.a The other user didn't answer or decline the call

Use case diagrams

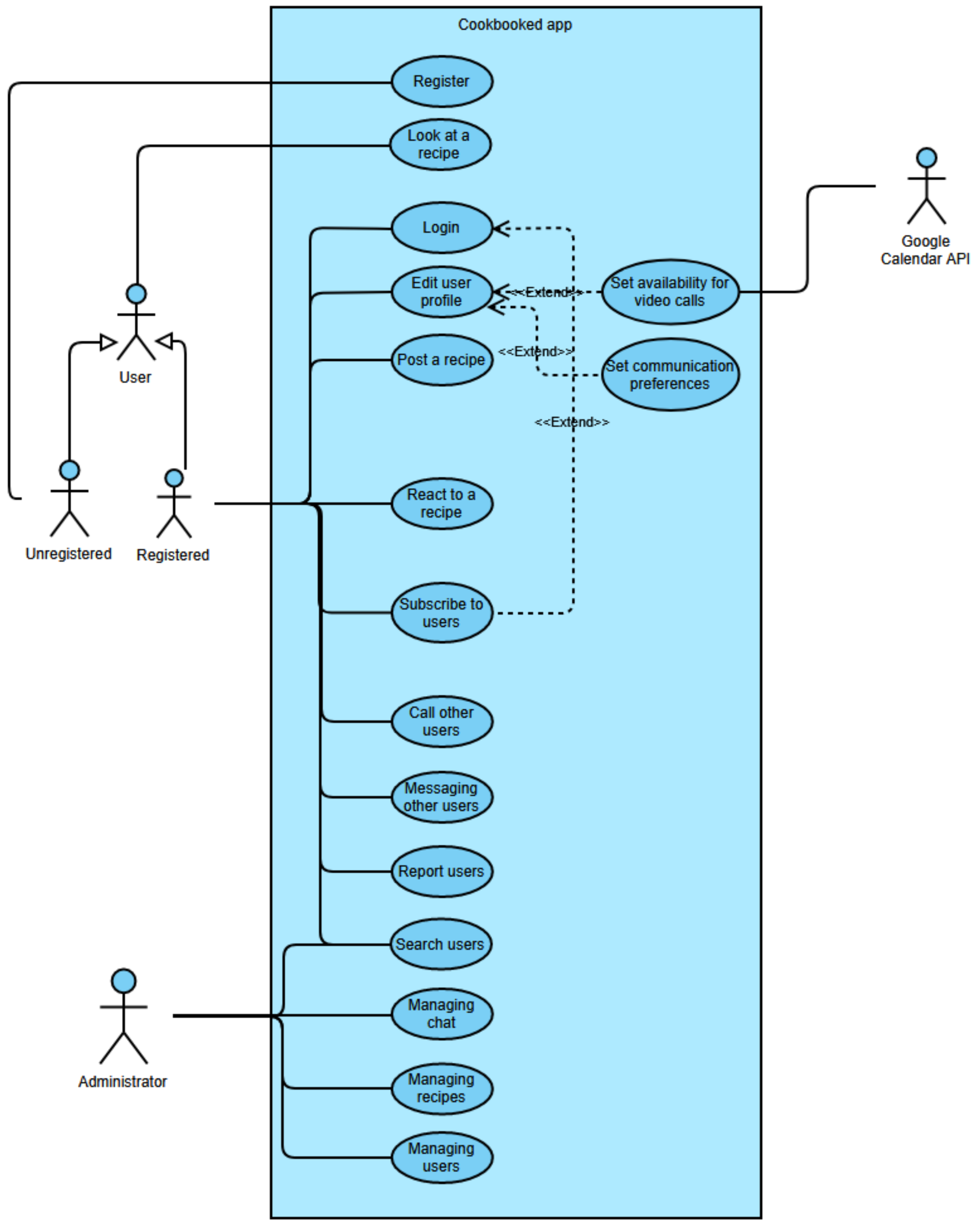


Figure 3.1: Use Case diagrams

3.1.2 Sequence diagrams

Unregistered users can only browse recipes, cuisine types, or specific ingredients.

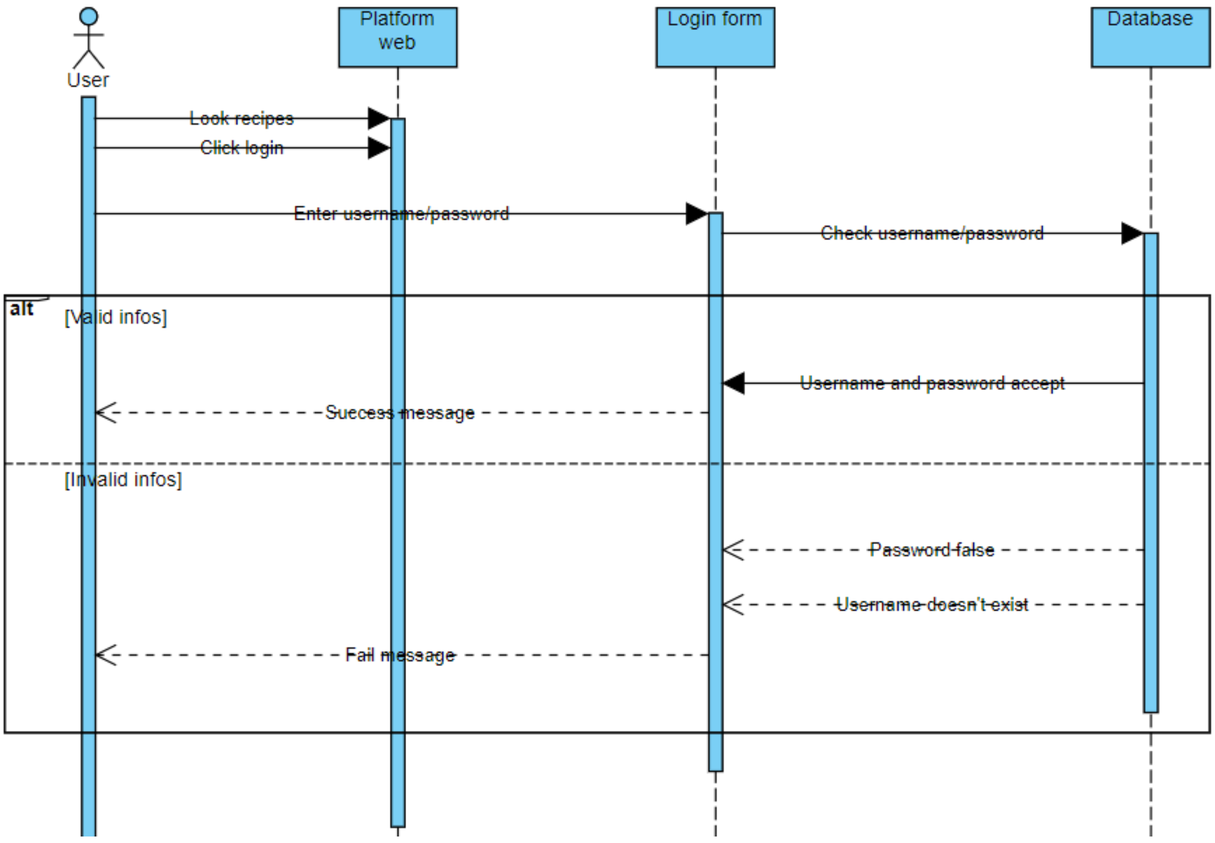


Figure 3.2: Sequence diagram - Login

Once registered, users can post their cooking and baking recipes. They can provide recipe details such as the title, ingredients, preparation steps, cooking time, and tags. Users can also add images and videos related to the recipe.

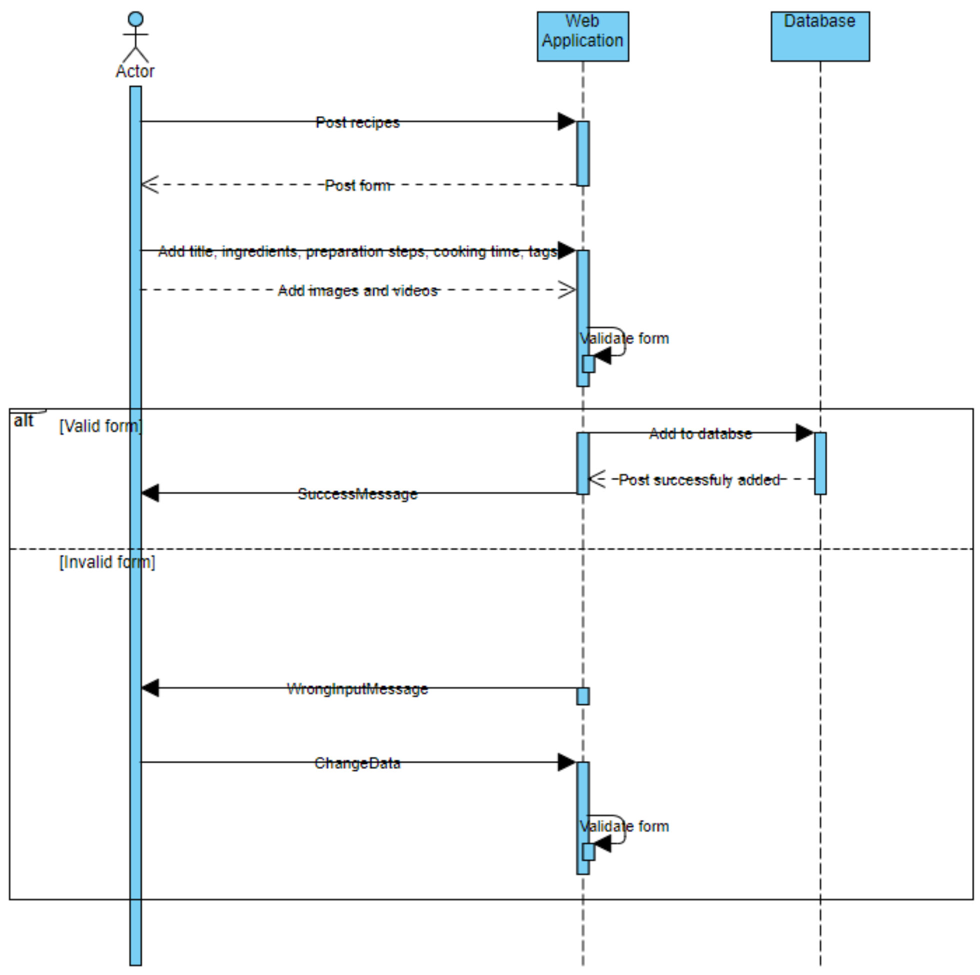


Figure 3.3: Sequence diagram - Add recipe

Registered users can like, comment on, and save recipes for future reference. Users can follow their favorite recipe authors to receive updates on new recipes.

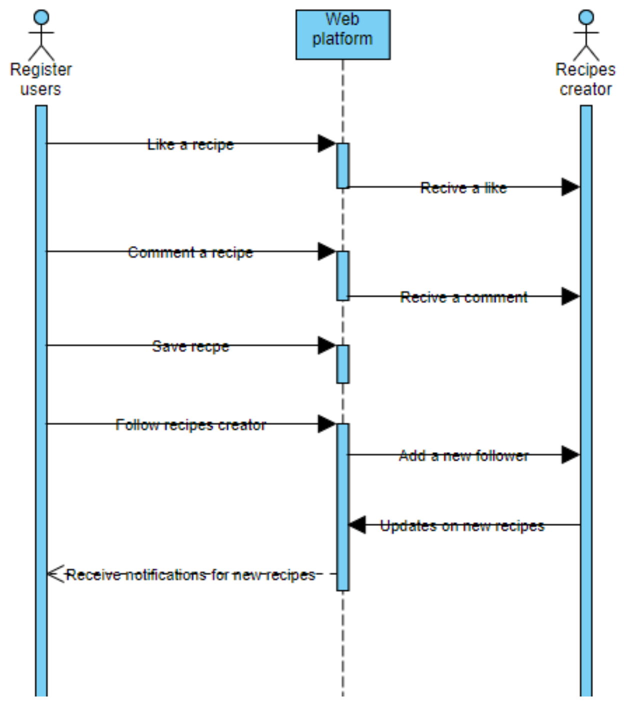


Figure 3.4: Sequence diagram - Interactions with recipes creators

3.2 Other requirements

The website will support only English language and will be accessible on different devices such as mobile phones, computers and tablets. The website is only available on web browser such as Chrome, Firefox, Safari, Edge, etc. The maximum supported number of users is approximately of 1000 users. To protect our users, we will use encrypted passwords.

4. System architecture and design

Our project is a strong and extensible web application designed with a modern architecture. The combination of React for frontend, Django for the backend, PostgreSQL for the database and Tailwind CSS for styling, creates a powerful user experience.

The client side (frontend) of our application is constructed with React. React is not just a library, it is an important JavaScript framework designed to create a software's user interface. Almost all React architectures include:

- Project Directory: contains all configuration documentation, files and other resources.
- src: source directory where the main coding is developed
- Assets: where we store assets that maintain statics like images.
- Components: directory that stores reusable components.
- Services: directory that manages various services or functionalities that aren't directly tied to user interface components, like authentication services.
- Store: where is commonly utilized handling the state of your application.
- Utils: this directory is frequently utilized for handling utility functions, that can appear in various sections of your application.
- Views: where you structure the more advanced components that came from the 'Components' directory and engage with services from the 'Services' directory.

The diagram below represents how this architecture works.

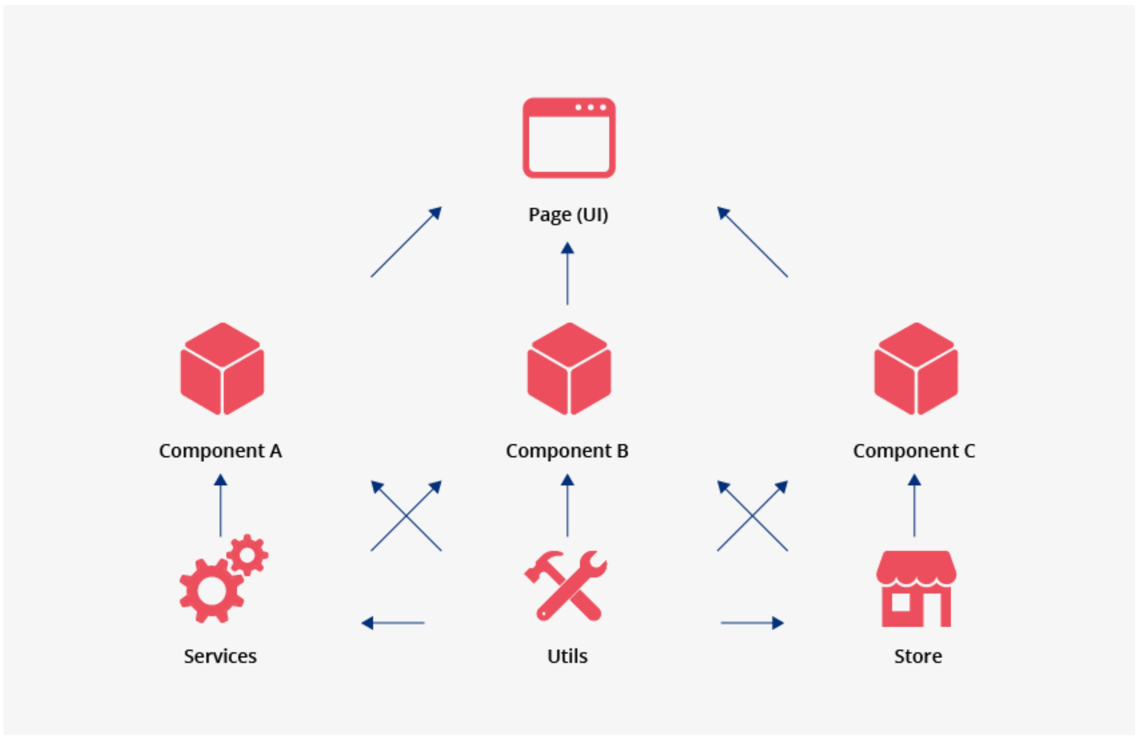


Figure 4.1: React architecture

The backend side of our project was developed using Python Django Framework, more precisely the Model-Template-View architecture pattern. Our innovative web application is divided into 3 parts:

- Model: is responsible for manage the data of the application.
- Template: outlines how data should be presented to the user
- View: controls what the user sees

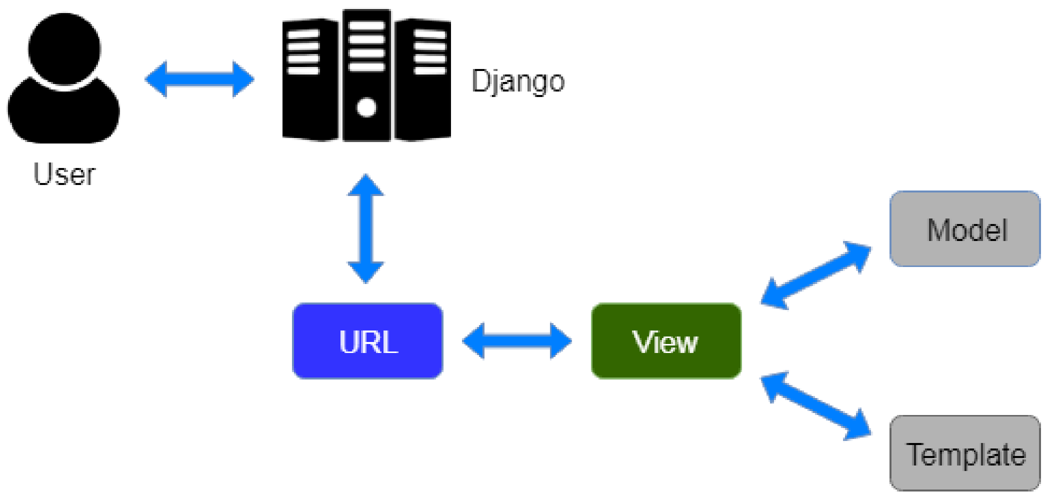


Figure 4.2: Model-Template-View architecture

4.1 Database

We use the PostgreSQL database on our server for storing structured data.

4.1.1 Table description

Primary keys are in light green and foreign keys are in blue.

Our first table is Register User. Register user is defined by their ID, username, name and surname, and email. He needs to have a password and can have a profile picture. He also can be limited or banned.

Register User		
user_id	INT	ID user
username	VARCHAR	username
password	VARCHAR	user password
email	VARCHAR	usermail
name	VARCHAR	user name
surname	VARCHAR	user surname
profile picture	VARCHAR	picture of the user
isbanned	BOOLEAN	if the user is banned from the website, boolean response
islimited	BOOLEAN	if the user is limited on the website, boolean response

The recipe table is defined by its id and is connected with the user table by user_id but also with ingredients_id from the ingredient table. A recipe has a title, category, cuisine type, cooking time, and preparation steps. The recipe is also connected to the tag table. Recipe can also get images videos and likes.

Recipe		
recipe_id	INT	recipe_id

Continued on next page

Continued from previous page

Recipe		
user_id	VARCHAR	user_id
ingredients_id	VARCHAR	ingredient
recipe_title	VARCHAR	recipe_title
category	VARCHAR	category
cuisine_type	VARCHAR	Type of cuisine
cooking_time	VARCHAR	cooking_time
preparation_step	VARCHAR	preparation_step
tag_list	TAG	tags
image	VARCHAR	recipe image
video	VARCHAR	recipe video
nb_like	INT	number of likes

The ingredient table is defined by the id and contains the ingredient name. This table is useful to permit the user to add missing ingredients in the database when he creates a recipe and will be useful to define some tags automatically.

Ingredient		
ingredients_id	INT	recipe_id
ingredient_name	VARCHAR	ingredient name
_tag_list	TAG	tag to which the ingredient is linked

The tag table is defined by its id and name. And as the ingredient table, this table is useful to add missing tags in the database from the creation of recipe.

Tag		
tag_id	INT	the tag id
tag_name	INT	the name of the tag

The table admin is defined by the id. The admin table don't have other variable because we don't need more.

Admin		
admin_id	INT	the administrator id

4.1.2 Database diagram

The database diagram presents the graphical structure of the database. The primary and foreign keys are highlighted and tables linked.
We create a many-to-one relationship indicating that for a single user we can have multiple recipes.
Creating the table JointRecipeIngredient, we indicate that multiple ingredients can be associated to a single recipe and that multiple recipes can use the same ingredient. For example, if we have a "Fruit Salad" and a "Smoothie" they can use the same ingredient and "Fruit Salad" can have as many as ingredients does it need.
Creating the table JointRecipeTag, we indicate that we can associate multiple tags with a single recipe and that multiple tasks can have the same tag. For example, if we have multiple recipes related we can tag them the same tag like "Fast Food" and a unique recipe can have different tags like "Healthy" , "Vegetarian", etc. The graphic was developed using draw.io, as show below.

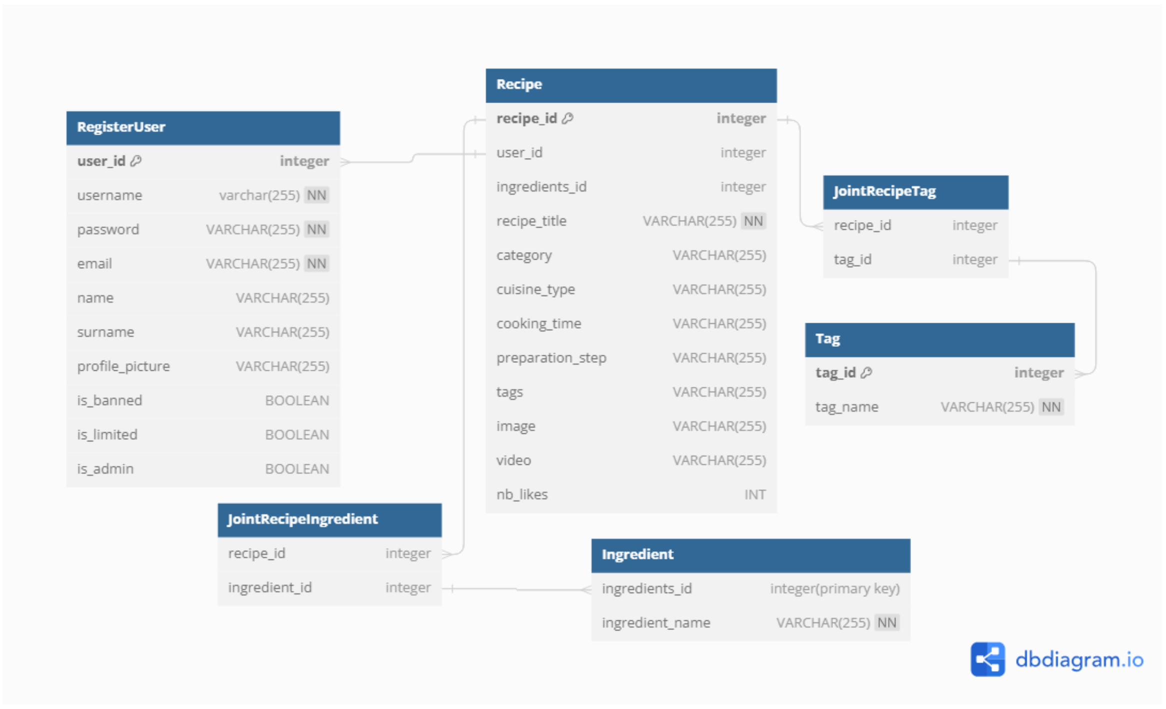


Figure 4.3: Database diagram

4.2 Class diagram

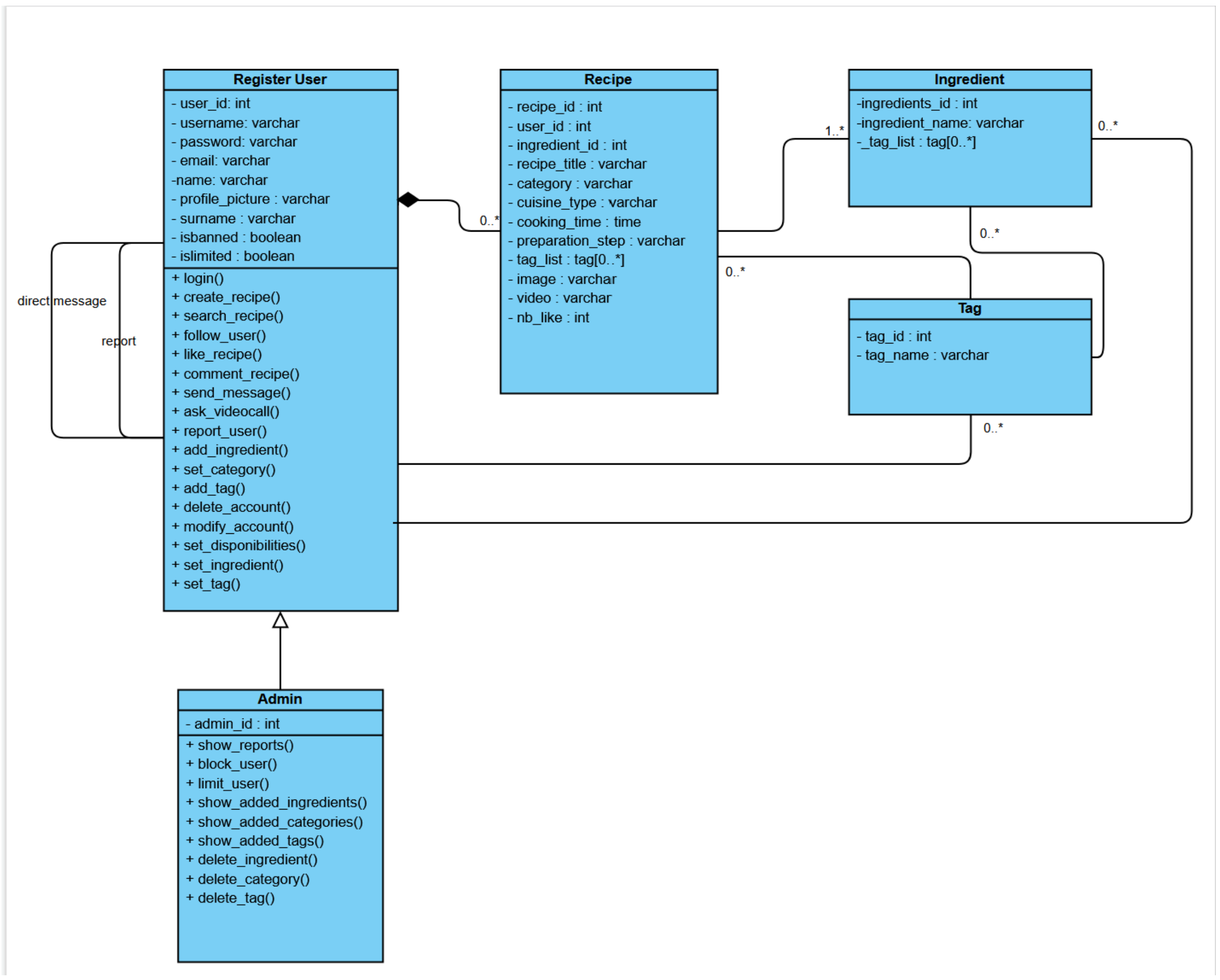


Figure 4.4: Class Diagram

4.3 State machine diagram

A state machine diagram is used to describe the dynamic behavior of classifiers such as classes, use cases and systems and describes all possible states of an object and the paths to get from one state to another.
Firstly, the user needs to login into its account, in order to have access to the profile page. Afterwards, the default state corresponds to the user's profile page, from where the user can access multiple functionalities such as Edit Profile, Add Post and React on Post.
The state ”Edit Profile” gives the user access to the respective editing page, where he can manage his personal information, set his communication preferences and availability and edit or delete recipes. After editing the profile, the user selects the ”Save” option to apply the changes and is redirected back to the default space (”User's profile). The state ”Add Post” allows the user to insert a post on their profile, then after saving the post and returning to ”User's Profile”. Then, the user can select a post to react on. This is done on the ”React on Post” state, where the user can check all the reactions including likes and comments from the selected post and add a comment by clicking on the ”Add Reaction” option.

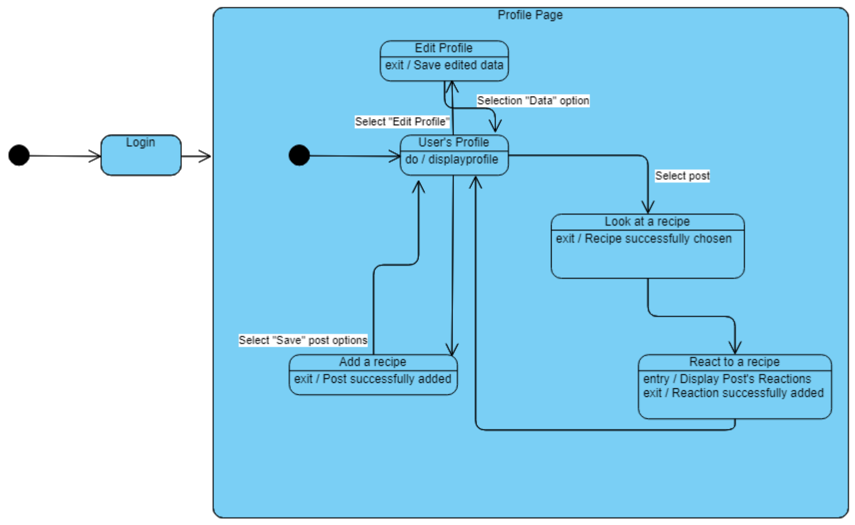


Figure 4.5: State Machine Diagram

4.4 Activity diagram

An activity diagram is about the flow of control or object from one activity to another. The process starts when the user navigates to their profile, the application front-end will request the back-end to recover the user's profile data in order to display it. Then, the user must enter the post details about the recipe and post it. The front-end will request the back-end to store the data about this post. If the creation of the post is successful or not, an error message can be shown in case of failure or the page of the profile will be refreshed with the new post added.

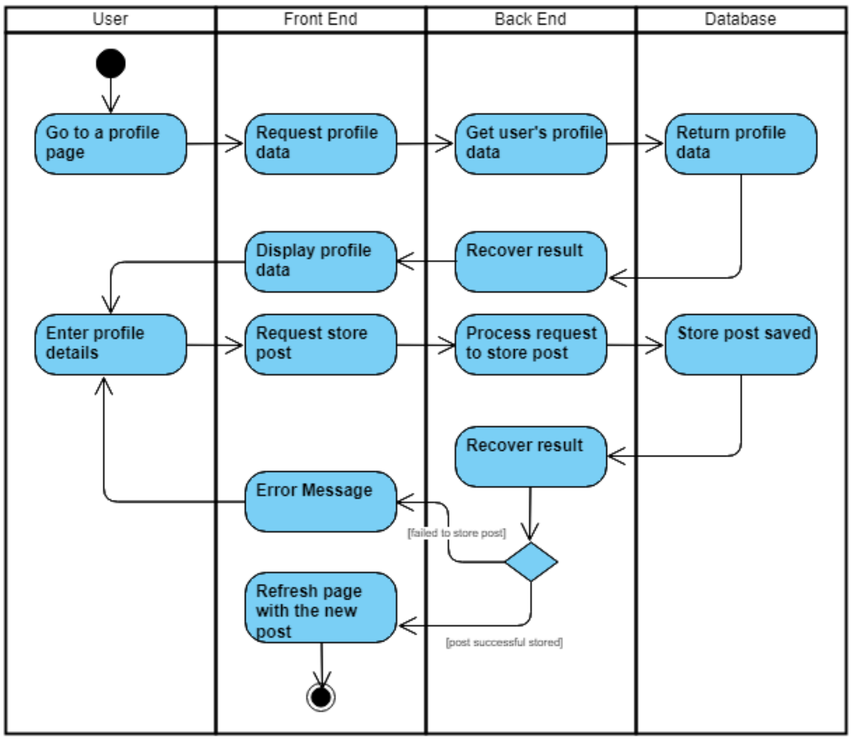


Figure 4.6: Activity Diagram

4.5 Component diagram

The component illustrates the structural organization of components within our software system. In our software for example we have two big components, the front-end in which the users interact directly, and the back-end responsible for the server-side, database interaction, etc. In the front-end, we have 3 main components, the Recipe View, the Registration form and the Profile View. If users do some changes in the front-end, the back-end will permit to change the data and store it.

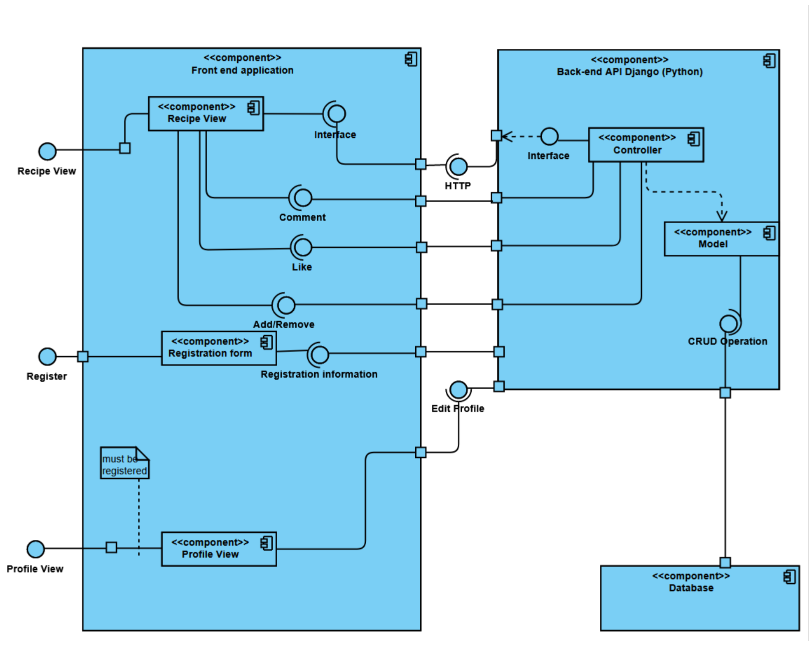


Figure 4.7: Component Diagram

5. Implementation and user interface

5.1 Tools and technologies

To work on the project we used several software and online applications.

- **Overleaf** is an online LaTeX editor which we use to create the documentation of our application.
  We used it during the entire process of our teamwork.
  https://www.overleaf.com

- **Grammarly** is a typing assistant, it reviews spelling, grammar, punctuation, clarity, engagement, and delivery mistakes in English texts, detects plagiarism, and suggests replacements for the identified errors.
  We used it to review our documentation.
  This tool is available online and as a software.
  https://app.grammarly.com/

- **VisualParadigm** is a productivity suite that proposes many tools such as PDF, documents, spreadsheets or PPT editors, graphic design, or diagramming tools... We only used the online diagramming tool to make our diagrams (use case, sequence, class...). The online mode permits you to collaborate with your team.
  We used this tool with its online version but you can also find it as a software.
  https://online.visual-paradigm.com/

- **Dbdiagram.io** is a Database Relationship Diagram Design tool where you can code to create your database diagram.
  We used this tool to create our database diagram.
  https://dbdiagram.io/home

- **GitHub** is a platform that allows developers to create, store, and manage their code. It is commonly used to host open-source software development projects.

We used it to share our code and documentation with our team.
https://github.com/

- **Spider** is an open-source cross-platform for scientific programming in the Python language. We used it to code the back-end of the web app using the **Django** framework.
  https://www.spyder-ide.org/
  https://www.djangoproject.com/
- **VsCode** is a code editor redefined and optimized for building and debugging modern and cloud applications. We used it to code the front-end of the web app in HTML, CSS, and JavaScript languages.
  https://code.visualstudio.com/
- **NeoVim** is a highly extensible light-weight code editor. We used it to code the front-end of the web app in HTML, CSS, and JavaScript languages, back-end with Python and deployment configurations.
  https://neovim.io/
- **SQLite** is a library of a database engine accessible with SQL programming language. We used it to implement the mock database of our project.
  https://www.sqlite.org/index.html
- **PostgreSQL** is database engine accessible with SQL programming language. We used it to implement the database of our project.
  https://www.postgresql.org/
- **Computer Shell** is a computer program that presents a command line interface that allows you to control your computer using commands entered with a keyboard instead of controlling graphical user interfaces (GUIs) with a mouse/keyboard combination. We are using Windows computers, so we downloaded a separate program to access the shell.
- **Selenium IDE** is an open-source record and playback test automation for the web. We used it to do our system tests.
  https://www.selenium.dev/selenium-ide/

5.2 Software testing

Software testing is a process used to verify whether the software aligns with the anticipated requirements and to ascertain that it is free from defects. This phase aims to detect errors, discrepancies, or overlooked requirements when compared to the specified requirements. This phase is further categorized into component testing and system testing.

5.2.1 Component testing

Component testing is characterized as a type of software testing where each component undergoes testing independently, without integration with other components. A total of 7 test cases were executed, focusing on the custom user model, view for posting a recipe, and the like view as the components under examination.

- **Unit Test 1: Create a custom user (model)**
  Test 1.1: Check if the 'username' label is correctly defined
  Input: username
  Expected Result: username
  Actual Result: username
  Status: passed

  Test 1.2: Check if the 'created' label is correctly defined
  Input: created
  Expected Result: created
  Actual Result: created
  Status: passed

  Test 1.3: Check if the password maximum length is correctly defined
  Input: azertyuiopmlkjhgfdsqwxcvbnazert
  Expected Result: sign up successfully
  Actual Result: sign up successfully

Status: passed

Test 1.4: Custom user print method
Input: id = 3
Expected Result: place@holder.com
Actual Result: place@holder.com
Status: passed

- **Unit Test 2: Posting a recipe (view)**
  Test 2.1: Check the created recipe URL location
  Input: created recipe id
  Expected Result: a random uuid
  Actual Result: 047e9515-656c-477a-ae71-2dbe7700177b
  Status: passed

- **Unit Test 3: Like (view)**
  Test 3.1: Check if like button add one like
  Input: Clicked on like
  Expected Result: number of likes goes up
  Actual Result: number of likes goes up
  Status: passed

  Test 3.2: Check if redirect correctly while on comment page
  Input: Clicked on like
  Expected Result: number of likes goes up, no redirection
  Actual Result: number of likes goes up, redirected to index page
  Status: failed

5.2.2 System testing

The System testing was conducted in Selenium framework. The methods chosen to be tested were: Login, Search user, and (un)following user.

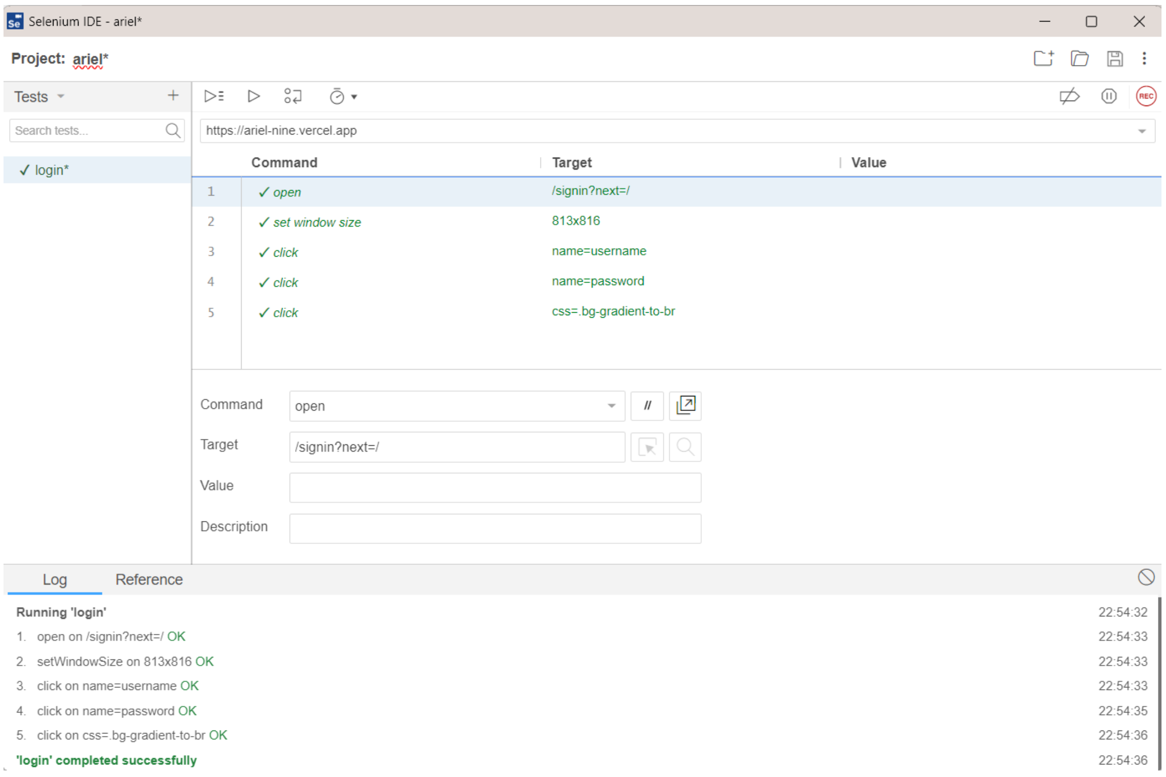
System Test 1: Login

Test 1.1: Correct credentials

Expected Result: Access feed

Actual Result: Access feed

Status: Passed



Selenium IDE - ariel*

Project: ariel*

https://ariel-nine.vercel.app

	Command	Target	Value
1	✓ open	/signin?next=/	
2	✓ set window size	813x816	
3	✓ click	name=username	
4	✓ click	name=password	
5	✓ click	css=.bg-gradient-to-br	

Command: open
Target: /signin?next=/
Value:
Description:

```
Running 'login'                                   22:54:32
1. open on /signin?next=/ OK                      22:54:33
2. setWindowSize on 813x816 OK                    22:54:33
3. click on name=username OK                      22:54:33
4. click on name=password OK                      22:54:35
5. click on css=.bg-gradient-to-br OK             22:54:36
'login' completed successfully                    22:54:36
```

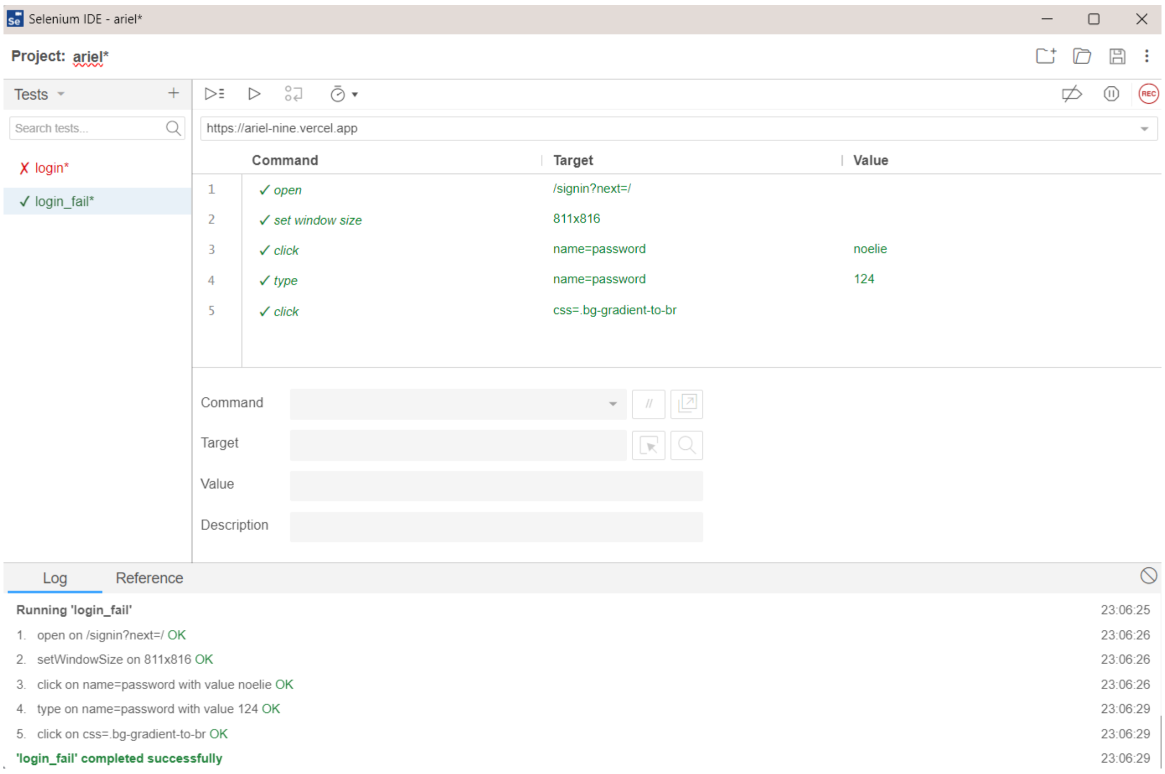
Figure 5.1: System test 1.1

Test 1.2: Incorrect credentials

Expected Result: Login failed

Actual Result: Login failed

Status: Passed



	Command	Target	Value
1	✓ open	/signin?next=/	
2	✓ set window size	811x816	
3	✓ click	name=password	noelie
4	✓ type	name=password	124
5	✓ click	css=.bg-gradient-to-br	

```
Running 'login_fail'                                        23:06:25
1. open on /signin?next=/ OK                                23:06:26
2. setWindowSize on 811x816 OK                              23:06:26
3. click on name=password with value noelie OK              23:06:26
4. type on name=password with value 124 OK                  23:06:29
5. click on css=.bg-gradient-to-br OK                       23:06:29
'login_fail' completed successfully                         23:06:29
```

Figure 5.2: Enter System test 1.2

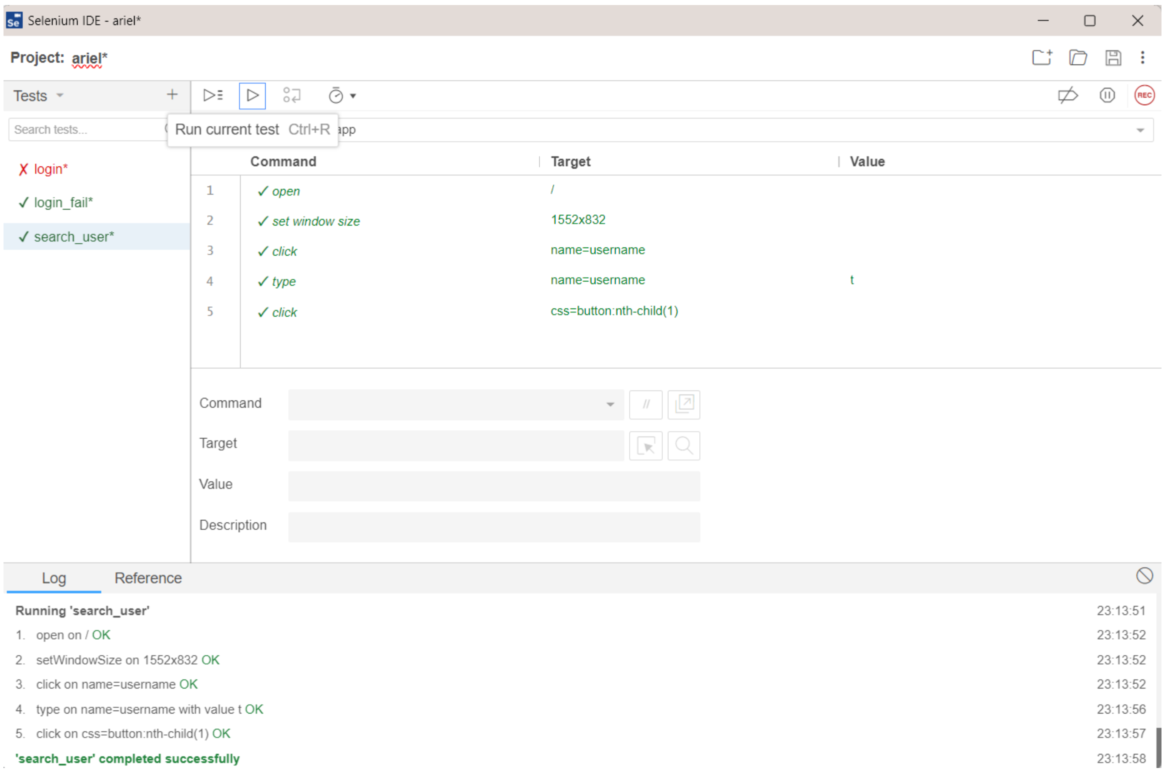
System Test 2: Search user

Test 2.1: Search user

Expected Result: user(s) find

Actual Result: user(s) find

Status: Passed



	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x832	
3	✓ click	name=username	
4	✓ type	name=username	t
5	✓ click	css=button:nth-child(1)	

Tests:
- ✗ login*
- ✓ login_fail*
- ✓ search_user*

```
Running 'search_user'                                          23:13:51
1. open on / OK                                                23:13:52
2. setWindowSize on 1552x832 OK                                23:13:52
3. click on name=username OK                                   23:13:52
4. type on name=username with value t OK                       23:13:56
5. click on css=button:nth-child(1) OK                         23:13:57
'search_user' completed successfully                           23:13:58
```

Figure 5.3: System test 2.1

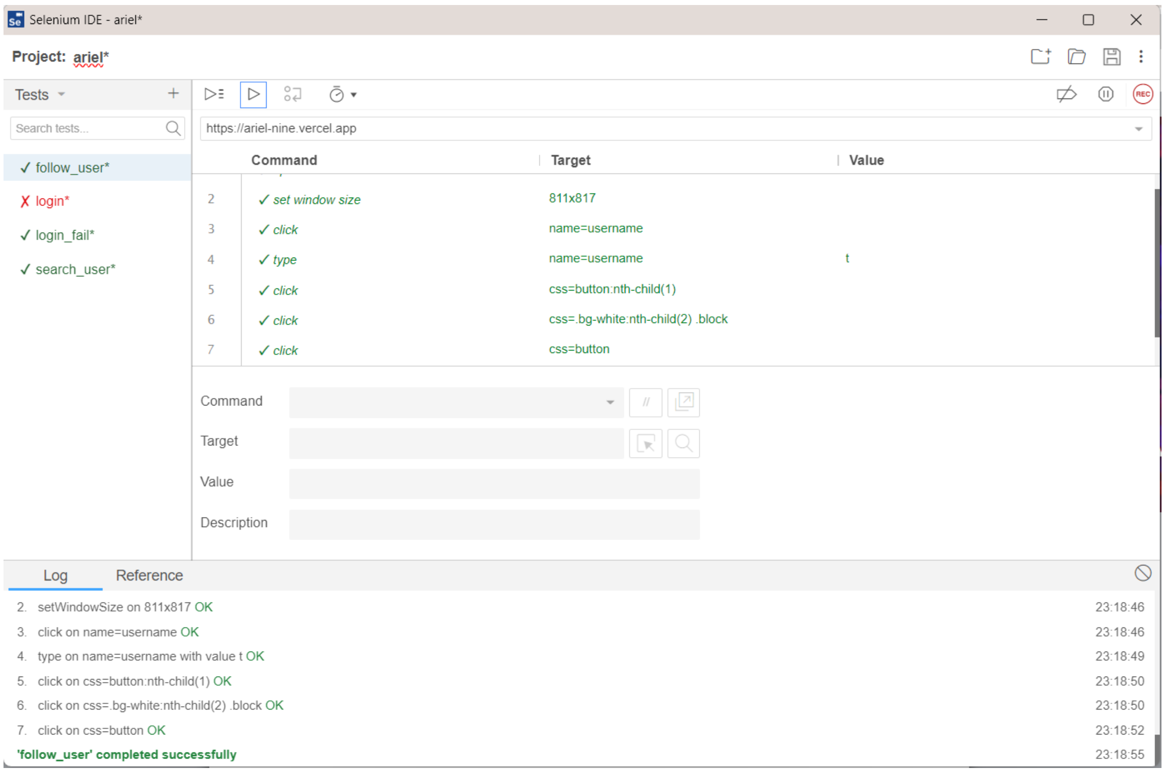
System Test 3: Following user

Test 3.1: follow user

Expected Result: user is followed

Actual Result: user is followed

Status: Passed



	Command	Target	Value
2	✓ set window size	811x817	
3	✓ click	name=username	
4	✓ type	name=username	t
5	✓ click	css=button:nth-child(1)	
6	✓ click	css=.bg-white:nth-child(2) .block	
7	✓ click	css=button	

```
2.  setWindowSize on 811x817 OK                                  23:18:46
3.  click on name=username OK                                    23:18:46
4.  type on name=username with value t OK                        23:18:49
5.  click on css=button:nth-child(1) OK                          23:18:50
6.  click on css=.bg-white:nth-child(2) .block OK                23:18:50
7.  click on css=button OK                                       23:18:52
'follow_user' completed successfully                             23:18:55
```

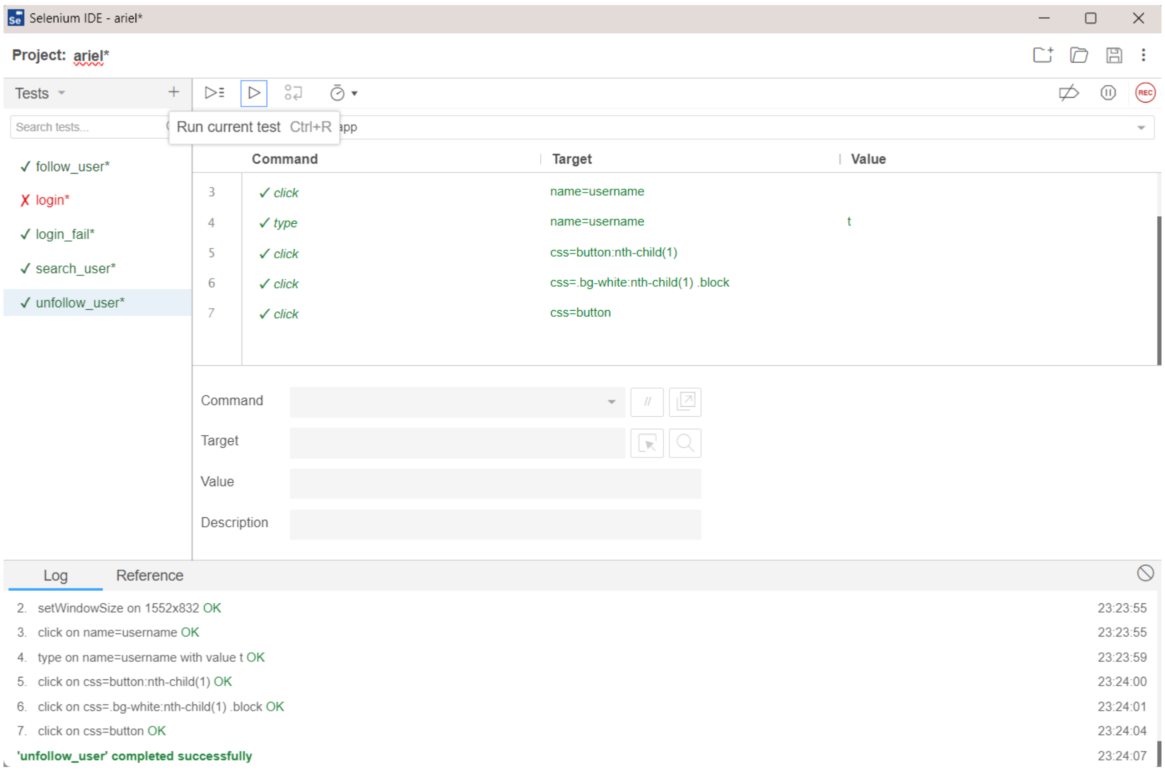
Figure 5.4: System test 3.1

Test 3.2: unfollow user

Expected Result: user is unfollowed

Actual Result: user is unfollowed

Status: Passed



✓ follow_user*
✗ login*
✓ login_fail*
✓ search_user*
✓ unfollow_user*

	Command	Target	Value
3	✓ click	name=username	
4	✓ type	name=username	t
5	✓ click	css=button:nth-child(1)	
6	✓ click	css=.bg-white:nth-child(1) .block	
7	✓ click	css=button	

```
2.  setWindowSize on 1552x832 OK                              23:23:55
3.  click on name=username OK                                 23:23:55
4.  type on name=username with value t OK                     23:23:59
5.  click on css=button:nth-child(1) OK                       23:24:00
6.  click on css=.bg-white:nth-child(1) .block OK             23:24:01
7.  click on css=button OK                                    23:24:04
'unfollow_user' completed successfully                        23:24:07
```

Figure 5.5: System test 3.2

5.3 Deployment diagram

Deployment diagrams represent the setup of runtime processing nodes and the software elements, processes, and objects that reside in these nodes. A deployment diagram is a graph of nodes that are connected by communication links. Nodes can contain component instances, components can contain objects, and components are connected to other components by dashed arrows.
The frontend container contains an artifact, which is the React part and the backend contains another artifact, which is the Django part. These two artifacts communicate using HTTP requests. The backend is connected to the PostgreSQL database with TCP/IP. Our web application was deployed on Vercel. Using a web browser, users can also have access to our application.

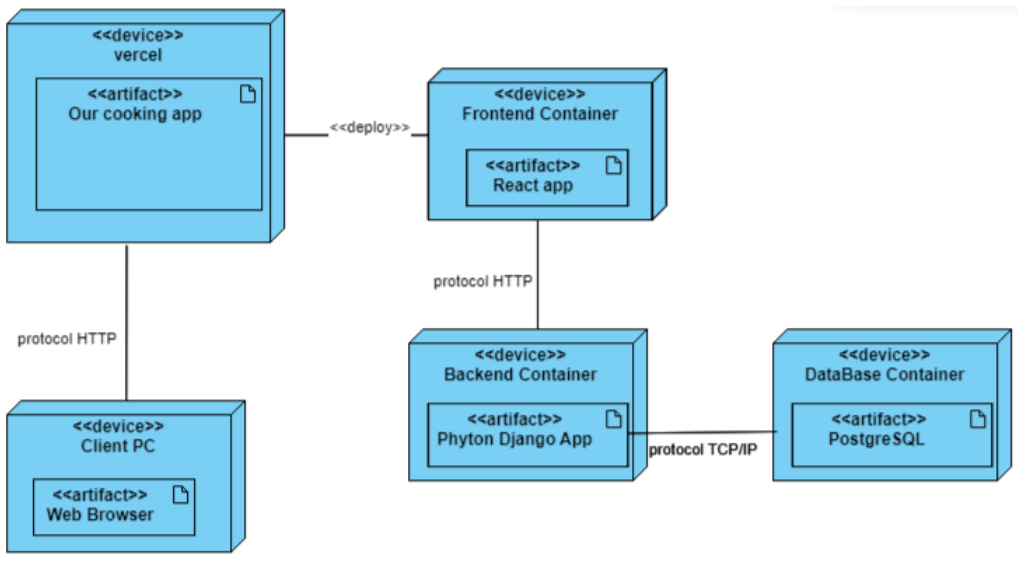


Figure 5.6: Deployment Diagram

5.4 Deployment instructions

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 in one of the stores. A desktop application is 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

In summary, the project aimed to create a collaborative web application for food enthusiastic. Users can create or look at recipes. The project unfolded in two phases. Initially, the focus was on meticulous documentation, and designing the web application. The close collaboration between front-end and the back-end team ensure the smooth functioning of the web application. To ensure this, the communication was essential between members of the team. Anticipating future improvements, a critical consideration involves the introduction of an administrator role to uphold post quality. This envisages a system where users can report recipes or creators, with the administrator taking appropriate actions. Elevating user experience, we propose incorporating videos and messages to enhance the utility and visual appeal of posts. Additionally, the ability to save recipes and receive notifications emerges as essential features for user engagement. Addressing security concerns, implementing password conditions and a forgotten password option would fortify the system. Furthermore, optimizing the registration process by incorporating email registration would contribute to a more streamlined and user-friendly onboarding experience. Allowing users to edit their usernames, set communication preferences, and manage availability directly from their profiles adds a personal touch. Getting rid of pop-up messages for profile changes makes the interface smoother and improves user interaction. Moreover, enabling access to the site without requiring users to be constantly connected provides more flexibility for a wider range of user preferences and situations.

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

Appendix: Group activity

Meeting log

1. meeting
   - Date: October 23, 2023
   - Attendees: Noëlie COMTE, Sonia HAMDI, Florian PAILHE
   - Meeting subjects:
     * Project description
     * Functional requirements
2. meeting
   - Date: October 26, 2023
   - Attendees: Sonia HAMDI, Florian PAILHE
   - Meeting subjects:
     * Use case description
3. meeting
   - Date: November 12, 2023
   - Attendees: Noëlie COMTE, Sonia HAMDI, Florian PAILHE
   - Meeting subjects:
     * Architecture table description
     * Class diagram
4. meeting
   - Date: January 18, 2024
   - Attendees: Noëlie COMTE, Sonia HAMDI, Cédric LISMONDE, Théau MARGOUTI
   - Meeting subjects:
     * Review of documentation
     * Software testing

Activity table

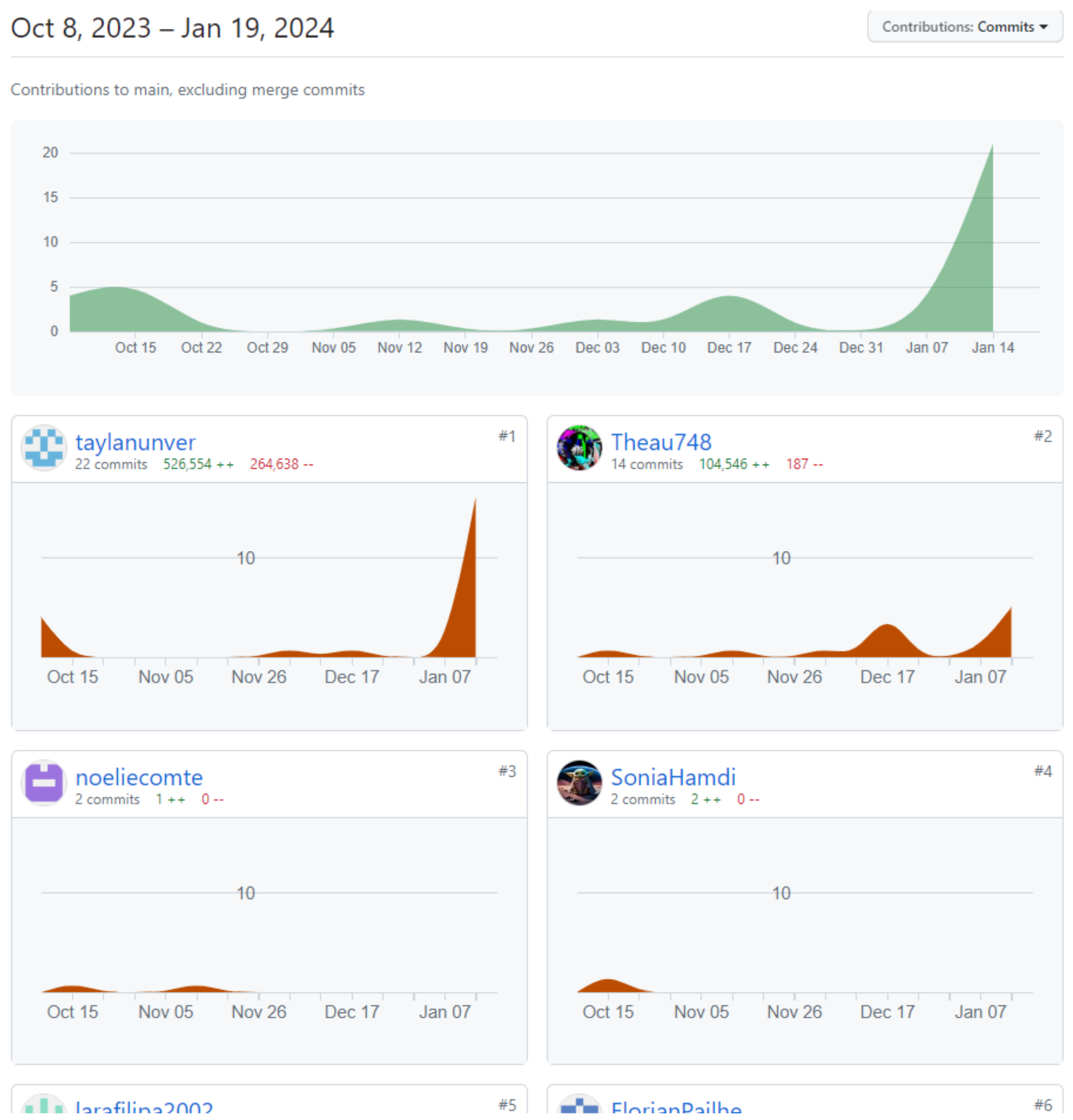
	Cédric LISMONDE	Sonia HAMDI	Noëlie COMTE	Florian PAILHE	Théau MARGOUTY	Lara FERNANDES	Taylan ÜNVER
Project management	0	4	5	0	0	0	0
Project task description	0	0.5	2.5	0	0	0	0
Functional requirements	0	2	2	2	0	0	0
Individual patterns description	0	3	3	3	0	0	0
Patterns diagram	0	1.5	0	0.5	0	0	0
Sequence diagram	5	0	0	0	0	0	0
Other requirements description	0	0.5	0.5	0	0	0	0
System architecture and design	0	0	0	0	0	1	0
Database	0	3.5	3.5	3.5	0	2.5	0
Class diagram	0	2.5	2.5	1	1	0	0
State diagram	0	0	0	0	0	3	0
Activity diagram	0	0	0	0	0	3	0
Components diagram	0	3	0	0	0	0	0
Used technologies and tools	0	0.5	2.5	0	0	0	0
Solution testing	0	0	3	0	1	0	0
Deployment diagram	0	0	0	0	0	3	0
Deployment instructions	0	0	0	0	0	0	
Meeting log	0	0	0.5	0	0	0	0
Conclusion and future work	1	0	0.1	0	0.1	0	0
References	0	0	0	0	0	0	0

Continued on next page

Continued from previous page

	Cédric LISMONDE	**Sonia HAMDI**	**Noëlie COMTE**	**Florian PAILHE**	**Théau MARGOUTY**	**Lara FERNANDES**	**Taylan ÜNVER**
Frontend Login and Registration page	0	0	0	0	0.5	0	0
Backend Login and Registration page	0	0	0	0	4	0	0
Frontend User profile page	0	0	0	0	0.5	0	0
Backend User profile page	0	0	0	0	3	0	0
Frontend feed page	0	0	0	0	0.5	0	0
Backend feed page	0	0	0	0	4	0	0
Frontend search user and recipe page	0	0	0	0	0.5	0	0
Backend search user and recipe page	0	0	0	0	2	0	0

Change log diagrams



Oct 8, 2023 – Jan 19, 2024

Contributions: Commits

Contributions to main, excluding merge commits

taylanunver #1
22 commits 526,554 ++ 264,638 --

Theau748 #2
14 commits 104,546 ++ 187 --

noeliecomte #3
2 commits 1 ++ 0 --

SoniaHamdi #4
2 commits 2 ++ 0 --

larafilipa2002 #5

FlorianPailhe #6