Let’s explore BEYOND SPACE

MULTI-USER APPLICATION Noser Young

By Lisa Meergans, Elif Berra Canmaya

Moreno Ramanti, Tennis Susyu

Inhaltsverzeichnis

Abstract

All parts of the document were discussed with the whole team. The goal of this documentation is to describe the planning of “our space” application. This document contains multiple UML-Diagrams (all expected diagrams are included) and tables which describe the behavior and structure of processes in our application.

KEY WORDS: Diagrams, Multi-User Application, Frontend, Backend

Use Cases

UC1

Actor(s)	**User**
Description	User creates own profile
Preconditions	• The user has successfully logged into the system. • The user does not yet have an existing profile. • The backend API is accessible. • The user has the role ROLE_USER.
Postconditions	• The user profile has been successfully saved in the database. • The profile is uniquely assigned to the logged-in user. • The user receives a success message in the front end.
Normal Course	1. The user navigates to the "Create Profile" page. 2. The system displays a form with the following fields: a. Address b. Date of birth c. Profile picture URL 3. The user clicks on "Save". 4. System validates the entries 5. The profile is saved in the database.

	6. The back end sends a success response to the front end. 7. The front end displays a success message.
Alternative Courses	**Invalid date of birth** 1. The date of birth entered results in the age of 13 years. 2. The back end rejects the request. 3. The frontend displays a corresponding error message. **Invalid profile picture URL** 1. The URL entered does not correspond to the valid format. 2. The backend rejects the request. 3. The frontend displays a corresponding error message. **Mandatory fields not filled in** 1. The user leaves a mandatory field empty. 2. The frontend prevents submission and displays a validation message.
Exceptions	• Invalid or expired JWT token (user is logged out).

UC2

Actor(s)	User
Description	User reads and updates own profile
Preconditions	• The user does have an existing profile and is logged in. • The backend API is accessible. • The user has the role USER.
Postconditions	Success case: • The updated profile data is saved. • The user receives a success message. • The updated data is displayed. Error case: • The data is not saved. • The user receives an error message with an explanation.
Normal Course	1. The user logs into the system. 2. The user opens the profile page. 3. The system loads the user's current profile data. 4. The system displays profile data. 5. The user changes the desired fields. 6. The user saves the changes. 7. The system validates the entries. 8. The changes are saved. 9. The system displays a success message and the updated data.
Alternative Courses	No changes made The user leaves the page without saving. No changes are made. Partial changes The user only changes individual fields. Only these fields are updated.
Exceptions	• Validation error • Invalid entries (incorrect URL format, minimum age not reached). • The system displays an error message.

UC3

Actor(s)	User
Description	User deletes own profile
Preconditions	• The user is successfully logged in. • The user profile exists in the system. • The user has authorization to delete their own profile.
Postconditions	• The user's profile has been permanently deleted from the database. • The user is no longer available in the system. • The user receives confirmation of the successful deletion.
Normal Course	1. The user navigates to the profile page. 2. The user clicks on the "Delete profile" action. 3. The system deletes the profile from the database. 4. The system displays a success message. 5. The user is redirected to the public home page or login page.
Alternative Courses	The user clicks on "Cancel". The profile remains unchanged.
Exceptions	• No authorization • The user is attempting to delete someone else's profile. • The system denies access (HTTP 403).

UC4

Actor(s)	Admin
Description	Admin can delete any user profile.
Preconditions	• The backend API is accessible • The user must be an admin • The user must be logged in as an admin
Postconditions	• The selected user profile is deleted • A success message is displayed after deletion • The user list is updated
Normal Course	1. Admin navigates to home page 2. Admin sees all the users listed 3. Admins select's user profile 4. Admin clicks on delete user 5. A confirmation dialog is shown 6. Admin confirms the deletion 7. The user profile is deleted 8. A success message is displayed 9. Admin continues with another process
Alternative Courses	1. Admin clicks on delete user 2. A confirmation dialog is shown 3. Admin cancels the deletion 4. The user profile is not deleted 5. Admin continues with another process
Exceptions	• The user is not an admin • The selected user does not exist • The backend API is not accessible

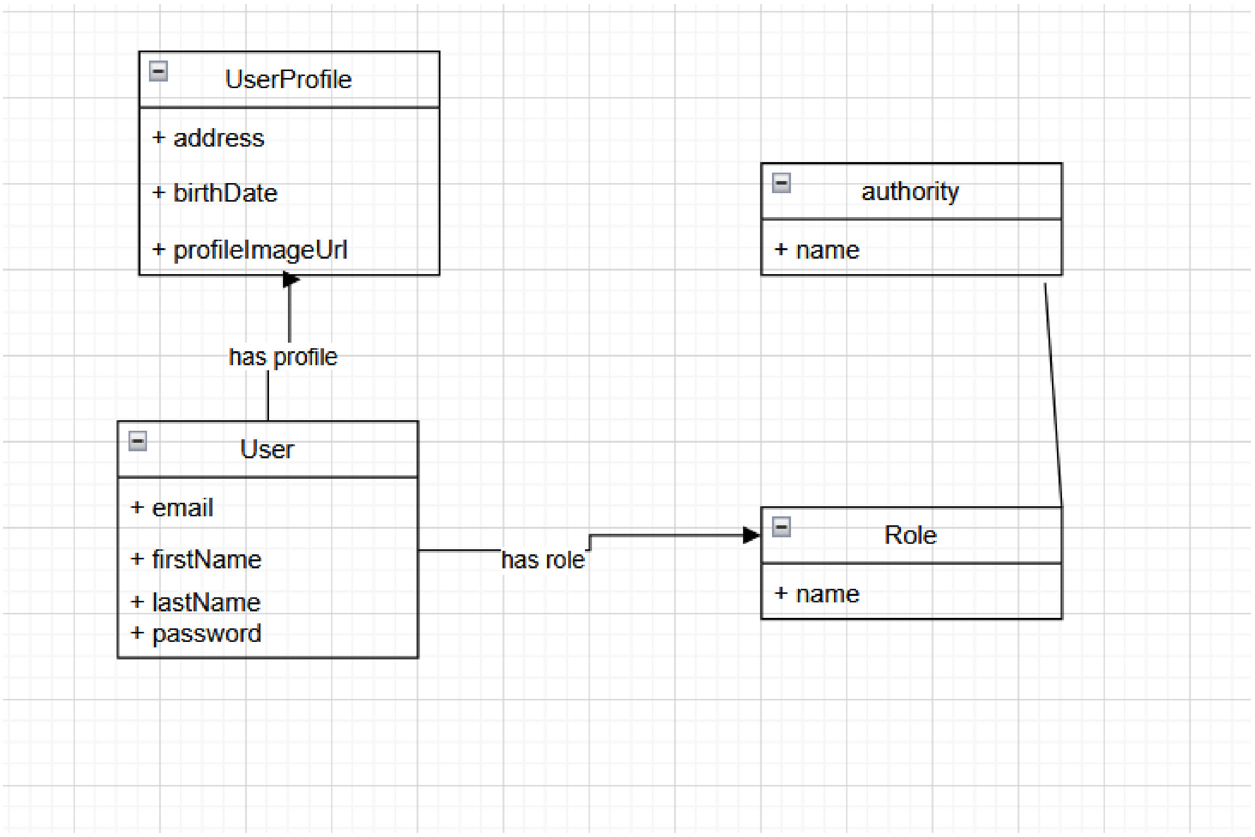
UC5

Actor(s)	Admin
Description	The admin can search for users.
Preconditions	• The backend API is accessible • The user must be an admin • The user must be logged in as an admin
Postconditions	• The user can see user profiles which are sorted based on age and name • The user can filter the data based on name and age • The results are paginated, which means only 10 users are shown per page • The admin can navigate between pages
Normal Course	1. Admin navigates to home page 2. Admin sees all the users listed without searching already 3. Admin can search by name 4. Even after the first letter which is put in Admin gets results 5. Admin filters based on minimum age and maximum age admin can also filter based on name 6. The results are sorted by name and age 7. Per page only 10 results are shown 8. Admin can navigate to the next or previous page using pagination 9. Admin finds the user it was searching for and continues with another process
Alternative Courses	1. The admin doesn’t filter or search specifically 2. Finds the user by scrolling down 3. Admin continues with another process
Exceptions	• The user is not an admin

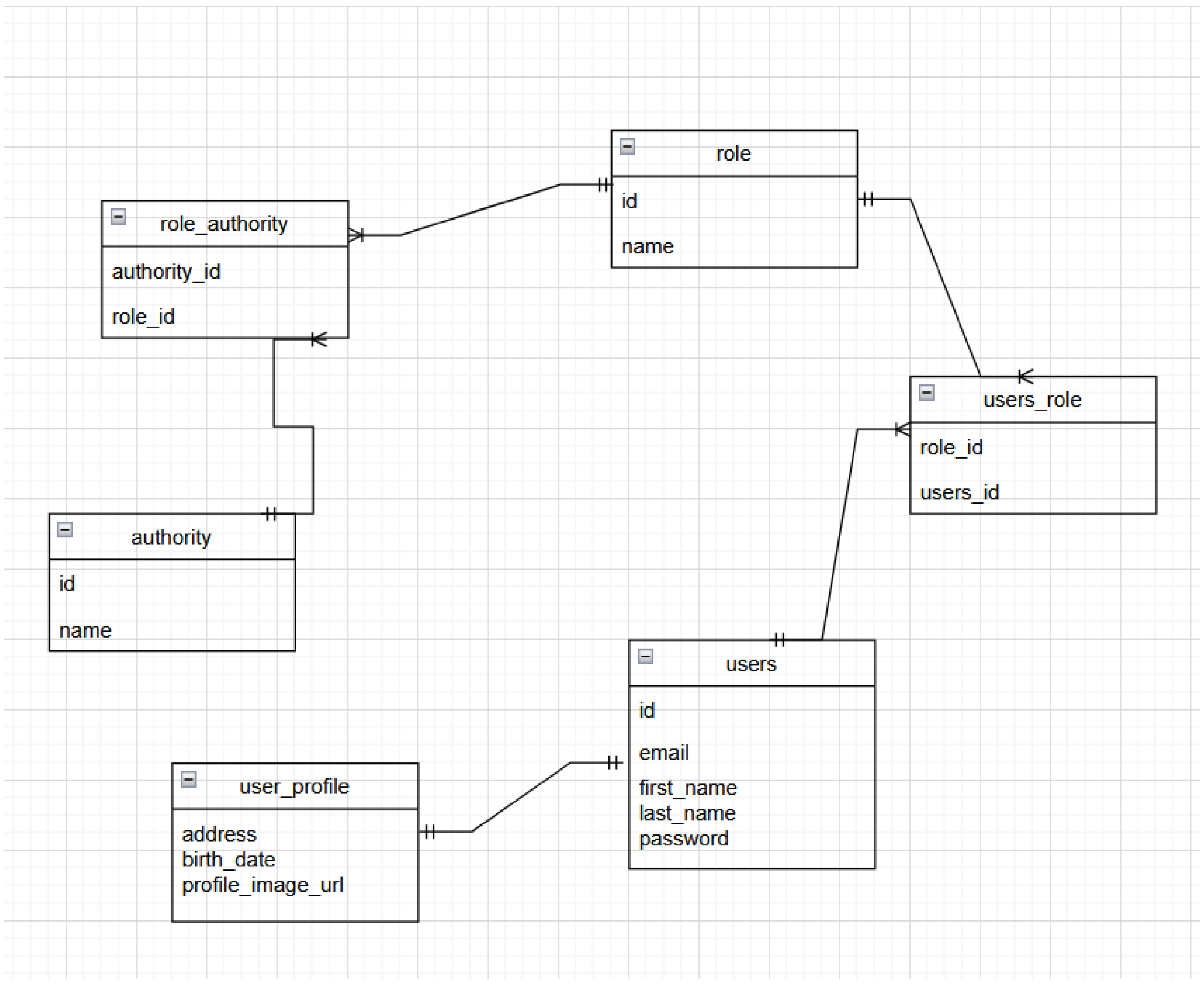
	• There are no existing users • Every user has the same age or name, so the filter is not going to make sense • No users match the search or filter criteria so the filter becomes useless • The backend API is not accessible

Diagrams/Models

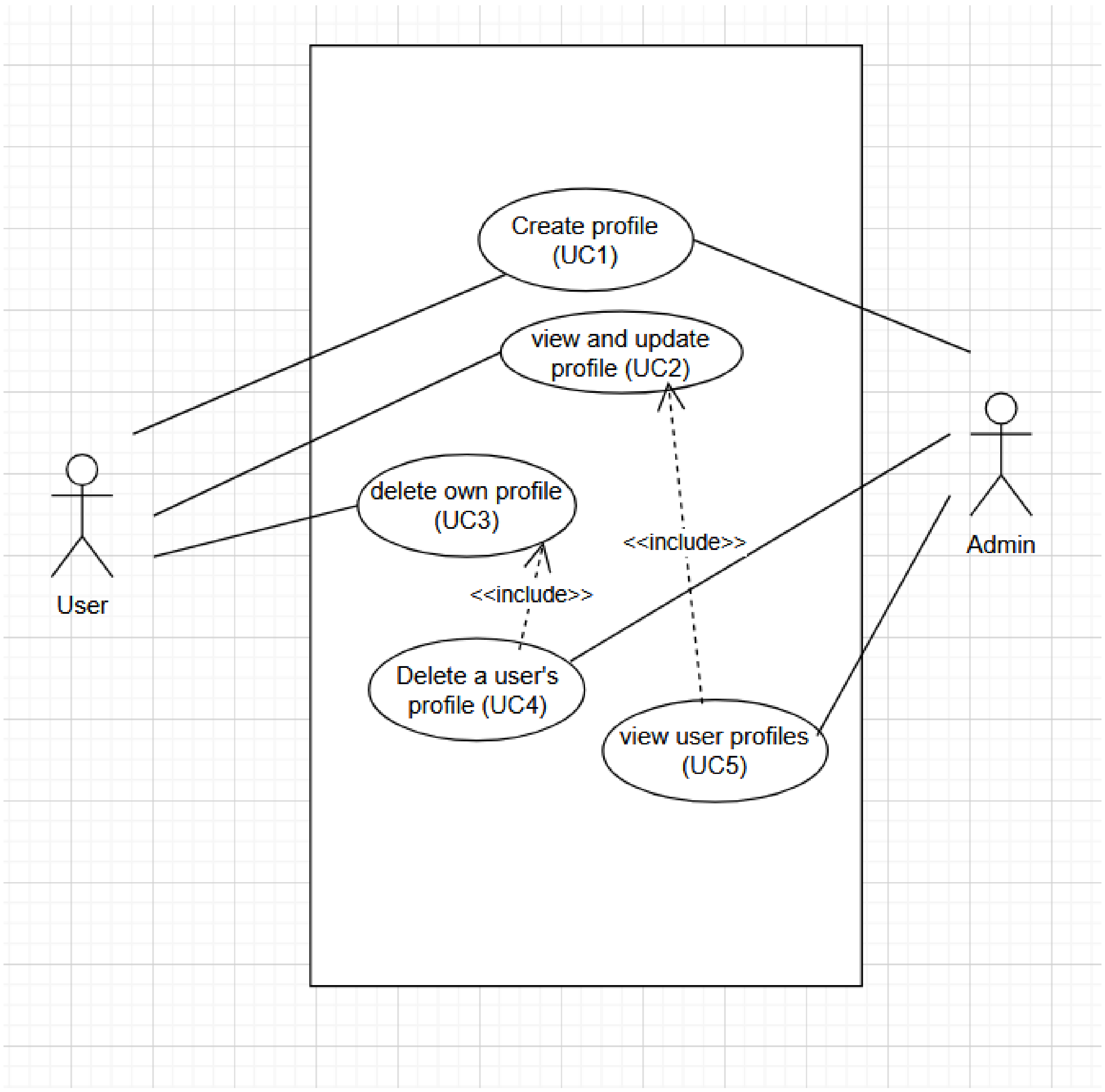
Domain model



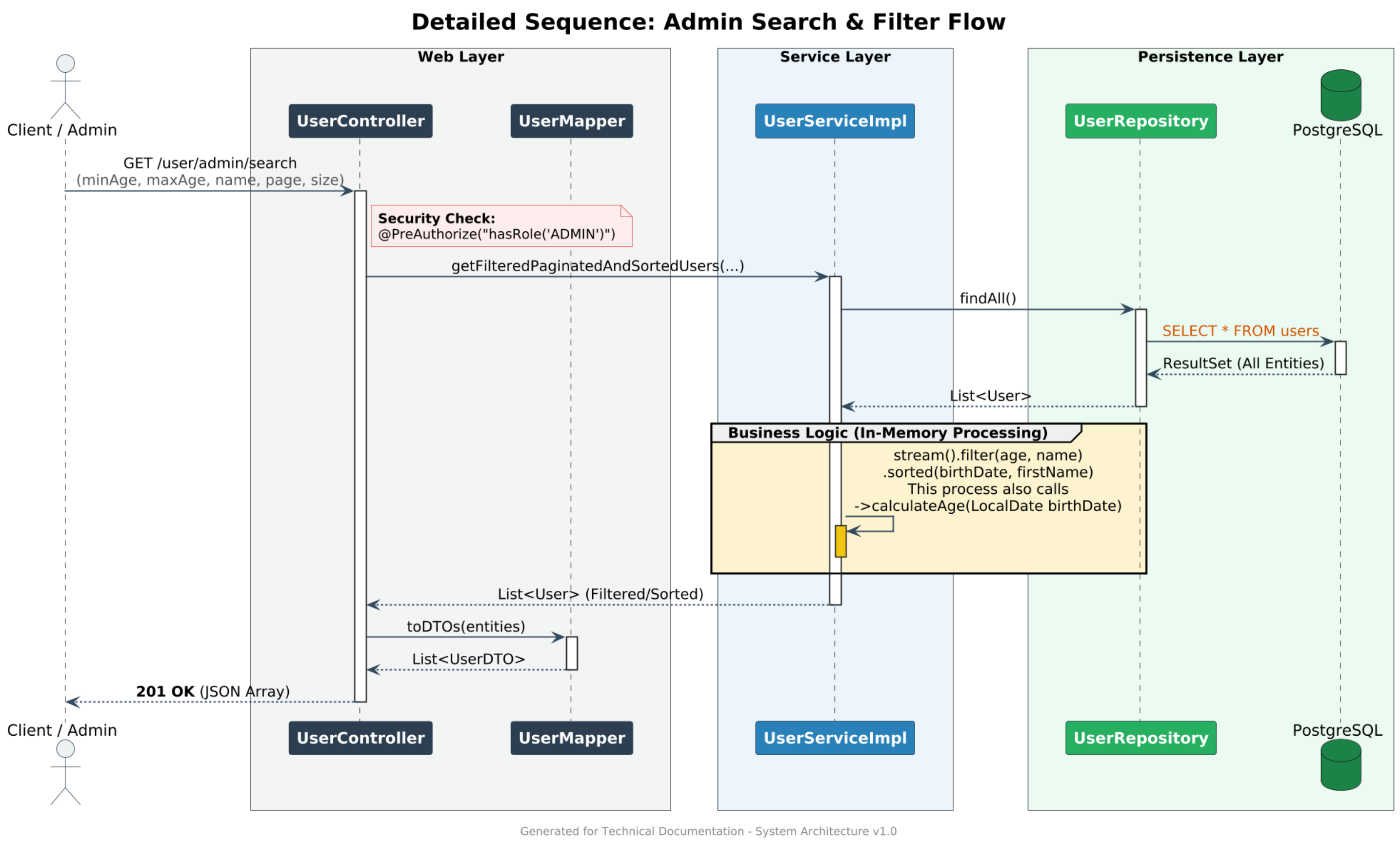
ERD


role
id
name
role_authority
authority_id
role_id
users_role
role_id
users_id
authority
id
name
users
id
email
first_name
last_name
password
user_profile
address
birth_date
profile_image_url

Use case diagram



Sequence Diagram


Detailed Sequence: Admin Search & Filter Flow
Web Layer
Service Layer
Persistence Layer
Client / Admin
UserController
UserMapper
UserServiceImpl
UserRepository
PostgreSQL
GET /user/admin/search
(minAge, maxAge, name, page, size)
Security Check:
@PreAuthorize("hasRole('ADMIN')")
getFilteredPaginatedAndSortedUsers(...)
findAll()
SELECT * FROM users
ResultSet (All Entities)
List<User>
Business Logic (In-Memory Processing)
stream().filter(age, name)
.sorted(birthDate, firstName)
This process also calls
->calculateAge(LocalDate birthDate)
List<User> (Filtered/Sorted)
toDTOs(entities)
List<UserDTO>
201 OK (JSON Array)
Generated for Technical Documentation - System Architecture v1.0

This sequence diagram was created with [PlantUML Editor Online - Free & Fast UML Diagram Tool | PlantText]

Testing

Detailed Testing of UC2

Endpoint	Testscenario	Expected Result	Technology
[PUT] /user/editUser	UC2 (Success): Complete Update. User is logged in and changes all fields (Address, Birth Date, Image URL) to valid new values.	All data is persisted. UI shows a success notification. Status 200 OK.	Cypress
[PUT] /user/editUser	UC2 (Success): Partial Update. User only changes the Address but leaves other fields as they were.	Only the Address is updated; other fields remain unchanged. Status 200 OK.	Cypress
[PUT] /user/editUser	UC2 (Validation): Empty Address. User deletes the content of the address field and tries to save.	The Submit button is disabled. An error message "Address is mandatory" appears in red below the input.	Cypress
[PUT] /user/editUser	UC2 (Validation): Invalid URL. User enters a string that is not a URL (e.g., "my_photo") into the Profile Image URL field.	Validation fails. Error message "Invalid URL format" (Yup/Frontend) is displayed. Request is not sent.	Cypress
[PUT] /user/editUser	UC2 (Business Logic): Age < 13. User changes their birth	Frontend prevents submission OR Backend returns 400 Bad	Cypress

	date to a year that makes them 12 years old.	Request with message "Age must be at least 13".	
[PUT] /user/editUser	UC2 (Security): Forbidden ID Access. User A is logged in but sends a PUT request with the ID of User B in the URL.	The @userPermissionEvaluator catches the mismatch. Access denied. Status 403 Forbidden.	Postman
[PUT] /user/editUser	UC2 (Security): Admin Role Access. Admin is logged in and tries to update a standard user's profile via the standard user edit route.	Depending on permissions: If Admin has no USER_MODIFY for that specific ID, access is denied. Status 403.	Postman
[PUT] /user/editUser	UC2 (Persistence): Refresh check. User updates profile, sees success message, and then refreshes the page.	The page reloads and displays the newly updated data fetched from the database, not old cached data.	Cypress
[GET] /user/profile	UC2 (Read): Initial State. User visits their profile page for the first time after login.	The UI correctly maps the Backend DTO fields to the profile labels. Status 200 OK.	Cypress

Superficial testing of all other UC’s

Endpoint	Testscenario	Expected Result	Technology
[POST] /user/register	UC1: Positive Case. Registration of a new user with valid data (Age ≥ 13, valid URL format).	User is created, password is encrypted in the DB, Status 201 Created.	Postman / JUnit
[POST] /user/register	UC1: Edge Case (Age). Registration with a birth date that results in an age under 13.	Validation fails. Status 400 Bad Request with message "Age must be at least 13".	Postman
[DELETE] /user/admin/{id}	UC4: Admin Authority. An admin deletes any user profile via its UUID.	User is removed from the database. Status 204 No Content.	Postman
[DELETE] /user/admin/{id}	UC4: Permission Check. A standard user attempts to use the admin delete endpoint.	Access denied. Status 403 Forbidden (missing USER_DEACTIVATE authority).	Postman
[GET] /user/admin/search	UC5: Search & Filter. Admin filters users by minAge=20 and firstName=Max.	A paginated list of matching users is returned. Status 200 OK.	Postman
[DELETE] /user/admin/{id}*	UC3 (Success): Deletion with Confirmation. User clicks the "Delete Profile" button, confirms the dialog, and the profile is removed.	The user is logged out, redirected to the home/login page, and a success message appears.	Cypress
[DELETE] /user/admin/{id}*	UC3 (Interaction): Cancel Deletion. User clicks "Delete Profile",	The dialog closes. The user remains logged in,	Cypress

	but clicks "Cancel" in the confirmation dialog.	and the profile is not deleted.	
[DELETE] /user/admin/{id}*	UC3 (Security): Unauthorized Deletion. User A attempts to send a delete request for User B's ID via the API.	Access denied. Status 403 Forbidden. (Ensures users can't delete each other).	Postman
[DELETE] /user/admin/{id}*	UC3 (Success): Deletion with Confirmation. User clicks the "Delete Profile" button, confirms the dialog, and the profile is removed.	The user is logged out, redirected to the home/login page, and a success message appears.	Cypress

Blackbox Testing whole Admin Process

ID	Feature	User Action	Observed Result	Status
01	**Authentication**	Sign in with email and password.	Successful login: Navigation bar becomes visible (Home, Profile, Admin, Logout).	**Pass**
02	**Navigation**	Visual inspection of UI.	Minimalist design: navigation bar is non-intrusive with a subtle separator line.	**Pass**
03	**User Creation**	Locate and click "Add User."	High discoverability via green color-coding. Form fields for bio-data and URL are present.	**Pass**
04	**Form Logic**	Inputting data & Image URL.	User correctly identified "Red" as Cancel/Stop and "Green" as Submit. Image URL linked successfully.	**Pass**
05	**Data Persistence**	Check User Profile list.	New user appears immediately with name, email, and the linked profile picture.	**Pass**
06	**Role Management**	Assign "Admin" role to user.	Admin dashboard allows toggling roles (User, Admin, Default). Updates reflect instantly.	**Pass**
07	**Security/Deletion**	Click trash icon to delete user.	System triggered a confirmation modal: *"Are you sure you want to delete this user?"*	**Pass**
08	**Data Integrity**	Confirm deletion.	User is removed from both the Admin Dashboard and the general user list.	**Pass**

Summary

In this section of the documentation, we discuss several aspects of the project, with a particular focus on team collaboration and the key factors involved in its development.

The project was divided fairly among all team members. To ensure efficient progress, we agreed that small decisions which did not affect the overall outcome could be made independently by the responsible team member. This approach helped us save time and work more efficiently. It also led to better decisions, as the person working on a specific task usually had the most relevant knowledge. Since these decisions did not impact other parts of the project or other team members, this method proved to be very effective.

For larger decisions that affected multiple components or team members, decisions were made democratically. We invested significant effort in maintaining clear and effective communication throughout the project, which was successfully implemented in practice. As a result, we did not encounter any communication issues, allowing us to save considerable time.

The key aspects of this project were meeting all expectations and delivering the best possible user experience, which placed a strong emphasis on thorough testing. For testing, we used Postman, Cypress, and black-box testing. The black box testing process was conducted according to Google standards, as one team member had previously attended a UI/UX lab at Google.