



IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

ANONYMOUS STUDYING

Prepared by:
Edhem Rogo
Sadžida Rakovac

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Lejla Breščić, Laboratory Assistant
Ajla Korman, Laboratory Assistant

15.06.2025.

TABLE OF CONTENTS

1. Introduction	3
1.1. About the Project	3
1.2. High-level Plan	3
1.2.1. Goal-Oriented Roadmap	3
1.2.2. Release Plan	3
1.2.2.1. Core Platform	3
1.2.2.2. Community Enhancement	4
1.3. Project Requirements	5
1.3.1. Functional Requirements	5
1.3.1.1. Authentication and User Management	5
1.3.1.2. Ticket Management	6
1.3.1.3. Reply and Interaction	7
1.3.1.4. Topic and Tag Management	8
1.3.1.5. Reputation and Moderation	9
1.3.2. Non-Functional Requirements	10
1.3.3. Product Backlog	12
1.4. UML diagrams	13
1.4.1. Class Diagram	13
1.4.2. User Authorization and Profile Creation Activity Diagram	14
1.4.3. Create Ticket Activity Diagram	15
1.4.4. Create and Accept Reply Activity Diagram	16
1.4.5. Create Ticket Sequence Diagram	17
1.4.6. Create Reply Sequence Diagram	18
1.4.7. Change Ticket Status Sequence Diagram	19
2. Project Structure	20
2.1. Technologies	20
2.2. Architectural Pattern	20
2.3. Database Entities	21
2.4. Design Patterns	21
2.5. Project Functionalities and Screenshots	24
2.6. Tests	27
3. Conclusion	28
4. Individual Contributions	28

1. Introduction

<https://github.com/redhem-dev/Anonymous-Studying>

1.1. About the Project

Anonymous Studying is a platform where students can anonymously seek help with academic challenges by creating tickets about problems they're facing. Other users (students or professors) can provide assistance through replies, creating a collaborative learning environment while preserving privacy. This platform is only available for IBU students and professors.

<https://anonymous-studying.vercel.app/>

1.2. High-level Plan

1.2.1. Goal-Oriented Roadmap

DATE The date or time frame when a goal should be met.	May 5, 2025	June 15, 2025
NAME The name of the new release.	Core Platform	Community Enhancement
GOAL The outcome or benefit you want to achieve.	Establish a functional anonymous help platform with essential features	Improve content quality and enhance community
FEATURES The high-level features required to meet the goal.	<ul style="list-style-type: none">• User authentication and profile creation• Basic ticket creation and management• Simple reply functionality• Notification system	<ul style="list-style-type: none">• Search and filtering• Reputation system• Admin dashboard• Performance optimizations• Bookmarks
METRICS The measures to determine if the goal has been met.	10 users	50 users

1.2.2. Release Plan

1.2.2.1. Core Platform

1.2.2.2.1. Base of Core Platform

Timeline: 2 weeks

Objective: Launch a secure, anonymous platform with basic ticket/reply interactions.

Features:

- User authentication via OAuth2
- Anonymous profile creation
- Basic ticket creation
- Simple home page
- Tickets reply functionality

1.2.2.2.2. Adding more features to Core Platform

Timeline: 3 weeks

Objective: Add notification system and ability to update personal information, tickets and replies

Features:

- Editing of personal information, tickets and replies
- Upvote and downvote option for tickets and replies.
- Answer acceptance
- Notification system

1.2.2.2. Community Enhancement

1.2.2.2.1. Participation Encouragement

Timeline: 3 weeks

Objective: Induce participation with a reputation system.

Features:

- Topic management
- Search, filters and sorting
- Reputation system

1.2.2.2.2. Engagement Improvement

Timeline: 2 weeks

Objective: Improve engagement with bookmarks options.

Features:

- Bookmarks

1.3. Project Requirements

1.3.1. Functional Requirements

1.3.1.1. Authentication and User Management

1.3.1.1.1. Authentication

As a user, I want to authenticate using OAuth2 so I can securely access the platform without creating a new password.

Acceptance Criteria

- The system must allow users to log in using OAuth2 authentication providers (Google).
- The login process must redirect users to the selected OAuth2 provider's authorization page.
- If the user logs in for the first time, the system must create a new account using the OAuth2-provided email.
- Upon successful authentication, the system must retrieve and store the user's profile information (e.g., name, email) securely.
- If the user already exists, the system must match the OAuth2 email with the existing account and log them in.
- Users must be redirected to their intended page after successful authentication.

1.3.1.1.2. Username Creation

As a new user, I want to create a username after authentication so I can maintain anonymity while having a recognizable identity.

Acceptance Criteria

- After successful authentication, new users must be prompted to create a unique username.
- The system must verify that the chosen username is not already in use.
- The platform must not display the user's real name or email address publicly.
- The username must adhere to predefined formatting rules (e.g., length, allowed characters).

1.3.1.1.3. Editing Personal Information

As a user, I want to view and edit my profile so I can manage my personal information.

Acceptance Criteria

- Users must be able to access a profile page where their personal information is displayed.
- Users must be able to edit and update usernames.
- The system must validate user input (e.g., username character limits).
- If the update is successful, the system must display a confirmation message.
- If the update fails, an appropriate error message must be shown.

1.3.1.1.4. Notifications

As a user, I want to receive notifications so I can be informed about updates on my tickets and replies.

Acceptance Criteria

- Users must receive notifications for replies and ticket status changes.
- The notification page must be easily accessible from the main navigation.
- Unread notifications must be highlighted until viewed.

1.3.1.1.5. Notifications Settings

As a user, I want to enable/disable notifications so I can manage how I receive updates.

Acceptance Criteria

- Users must be able to access a notification settings page.
- Users must be able to toggle notifications on or off.
- Changes to notification preferences must be saved instantly or upon confirmation.

1.3.1.2. Ticket Management

1.3.1.2.1. Creating a Ticket

As a user, I want to create a new ticket with a title and detailed description so I can explain my academic problem.

Acceptance Criteria

- Users must be able to access a form to create a new ticket.
- The form must include mandatory fields for Title, Description, Topic, and Tags.
- Users must be able to submit the form only if all required fields are filled.
- If any required field is missing, the system must display an appropriate error message.
- Upon submission, the system must save the ticket and assign it a unique ID.
- Users must be redirected to a page where they can view their submitted ticket.

1.3.1.2.2. Ticket Topics

As a user, I want to view all tickets related to a specific topic so I can find relevant discussions.

Acceptance Criteria

- The system must provide a predefined list of topics
- Users must be able to filter and view tickets by topic from the ticket list page.
- Users must be able to switch between different topics easily.
- If no tickets match a topic or search query, the system must display a message indicating no results.

1.3.1.2.3. Editing Tickets

As a ticket author, I want to update my ticket details so I can add clarification or additional information.

Acceptance Criteria

- Only the ticket author must be able to edit their own ticket.
- Admins or moderators must have the ability to delete any ticket or change its topic if necessary.

- Once a ticket is resolved, users must not be able to edit it (unless reopened).
- The system must display a success message upon successful update.
- If an update fails, the system must display an appropriate error message.

1.3.1.2.4. Resolved Ticket

As a ticket author, I want to close my ticket when my problem is resolved so others can easily identify the solution and know no further assistance is needed.

Acceptance Criteria

- The ticket author must be able to mark any reply on their ticket as Accepted when their issue is solved, which makes their Ticket Resolved .
- Resolved tickets must be visually distinct (e.g., a "Resolved" label or different color).
- Users must still be able to view all ticket details and replies after it is resolved.

1.3.1.2.5. Ticket Bookmark

As a user, I want to save and view my bookmarks, so that I can easily access important items later.

Acceptance Criteria

- When a user clicks on the "Bookmark" icon on an item, then the ticket should be saved in bookmarks.
- When the user clicks the "Bookmark" icon again, then the ticket should be removed from bookmarks.
- User must have access to the "Bookmarks" page, where he can see a list of all saved bookmarks.

1.3.1.3. Reply and Interaction

1.3.1.3.1. Ticket Replies

As a user, I want to reply to tickets so I can provide help or ask for clarification.

Acceptance Criteria

- Users must be able to reply to any open ticket.
- The reply form must include a text input field and "Post Reply" button.
- The system must validate that replies are not empty before submission.
- The ticket author must receive a notification when someone replies to their ticket.

1.3.1.3.2. Replies Visibility

As a user, I want to see replies of selected tickets so I can look for a possible solution.

Acceptance Criteria

- Replies must be displayed below ticket
- Each reply must show the author's name, timestamp, and content.
- Tickets must have an option to be sorted by Most Upvoted, Most Popular, Newest

1.3.1.3.3. Upvote

As a user, I want to upvote helpful tickets and replies so quality content gets recognized.

Acceptance Criteria

- Users must be able to upvote both tickets and replies.
- Each ticket and reply must display an upvote button (e.g., a thumbs-up or arrow icon).
- Clicking the upvote button must increase the vote count.
- Clicking the upvote button again must remove the upvote.
- Each ticket and reply must display the current number of upvotes.
- The upvote count must update in real-time when users interact with it.
- Users must not be able to upvote their own tickets or replies.

1.3.1.3.4. Downvote

As a user, I want to downvote unhelpful content so irrelevant or incorrect information is discouraged.

Acceptance Criteria

- Users must be able to downvote both tickets and replies.
- Each ticket and reply must display a downvote button (e.g., a thumbs-down or arrow icon).
- Clicking the downvote button must decrease the vote count.
- Clicking the downvote button again must remove the upvote.
- Each ticket and reply must display the current number of downvotes.
- The downvote count must update in real-time when users interact with it.
- Users must not be able to downvote their own tickets or replies.

1.3.1.3.5. Editing Replies

As a user, I want to edit my replies so I can correct mistakes or add information.

Acceptance Criteria

- Users must be able to edit their own replies.
- Users must be able to delete their own replies if necessary.
- Deleted replies must be replaced with a placeholder message (e.g., "This reply has been deleted").
- Users must confirm changes before saving their edited reply.
- The system must display a success message after a reply is edited.
- If an update fails, the system must notify the user and allow retrying.

1.3.1.4. Topic and Tag Management

1.3.1.4.1. Tickets Tagging

As a user, I want to add tags to my tickets so they can be found more easily through filtering.

Acceptance Criteria

- Users must be able to add tags when creating a ticket.
- Users must be able to select from predefined tags (e.g., "Bug," "Question," "Feature Request") and/or create custom tags.
- The system must limit the number of tags per ticket (e.g., max 5 tags).
- Users must be able to add or remove tags when editing their own ticket.

- Once a ticket is marked as Resolved, users must not be able to modify tags unless reopening the ticket.
- Tags must be case-insensitive to avoid duplication (e.g., "Bug" and "bug" should be treated as the same tag).

1.3.1.4.2. Search Tickets

As a user, I want to search for tickets by tags so I can find content relevant to specific concepts.

Acceptance Criteria

- Users must be able to search for tickets by entering a keyword in the search bar.
- Users must be able to search for tickets by tag or topics.
- Search results must display only tickets that contain the searched keyword.
- Users must be able to clear selected tags to reset the search.
- The results must update dynamically without requiring a page refresh.
- If no tickets match the searched tag, the system must display a message: "No tickets found for the "keyword". Try another search."

1.3.1.4.3. User Tooltip

As a user, I want to view user details when I hover over a username so I can have insight in reputation points.

Acceptance Criteria

- When users hover over a username, a tooltip must appear displaying the user details.
- The tooltip must disappear when the user moves the cursor away.
- The tooltip must not interfere with other interactive elements on the page.

1.3.1.5. Reputation and Moderation

1.3.1.5.1. Earning Reputation Points

As a user, I want to earn reputation points for helpful contributions so my expertise is recognized.

Acceptance Criteria

- Users must earn reputation points based on their contributions.
- Reputation points must be displayed on the user's profile and next to their name on posts.
- If a ticket or reply is deleted, associated reputation points must be deducted.

1.3.2. Non-Functional Requirements

1.3.2.1. Quick Platform Loading

As a user, I want the platform to load quickly (under 2 seconds) so I don't waste time waiting.

Acceptance Criteria

- The platform must use pagination or infinite scrolling to avoid loading excessive data at once.
- The system must return relevant results within 2 seconds for an optimal user experience.
- The search functionality must be optimized.
- The sorting selection must update the ticket list dynamically without requiring a page reload.

1.3.2.2. Usability

As a user, I want the platform to work well on all devices so I can get help anywhere and access it using multiple screen sizes.

Acceptance Criteria

- The platform must have an intuitive user interface.
- The platform must support desktop, tablet, and mobile screen sizes without breaking the layout.
- Elements must resize or reposition dynamically to fit smaller screens without horizontal scrolling.
- Forms and input fields must be optimized for mobile (e.g., numeric keyboard for number inputs).

1.3.2.3. Privacy Protection

As a user, I want my personal data to be securely stored so my privacy is protected.

Acceptance Criteria

- All data transmission must be encrypted using HTTPS
- Access tokens must be securely stored and never exposed to the frontend.
- The system must log out users properly by revoking their access tokens when they log out.
- User personal data (e.g., email, username) must be stored securely using encryption at rest.
- Sensitive data (e.g., passwords, authentication tokens) must be hashed.
- Users must be able to view, update, or delete their personal information.
- Users must have the ability to delete their accounts and associated data permanently.

1.3.2.4. Scalability

As a developer, I want the system to handle more users and content as the platform grows, so that it keeps working fast and smoothly even when many people are using it.

Acceptance Criteria

- Architecture must support growing numbers of users and content.
- Database schema design must be efficient.

1.3.2.4. Error Handling

As a user, I want clear error messages when something goes wrong.

Acceptance Criteria

- User-friendly error messages for common issues.
- No exposure of sensitive information in error responses.

1.3.2.5. Security

As a user, I want my data to be protected from unauthorized access, so I can trust the platform.

Acceptance Criteria

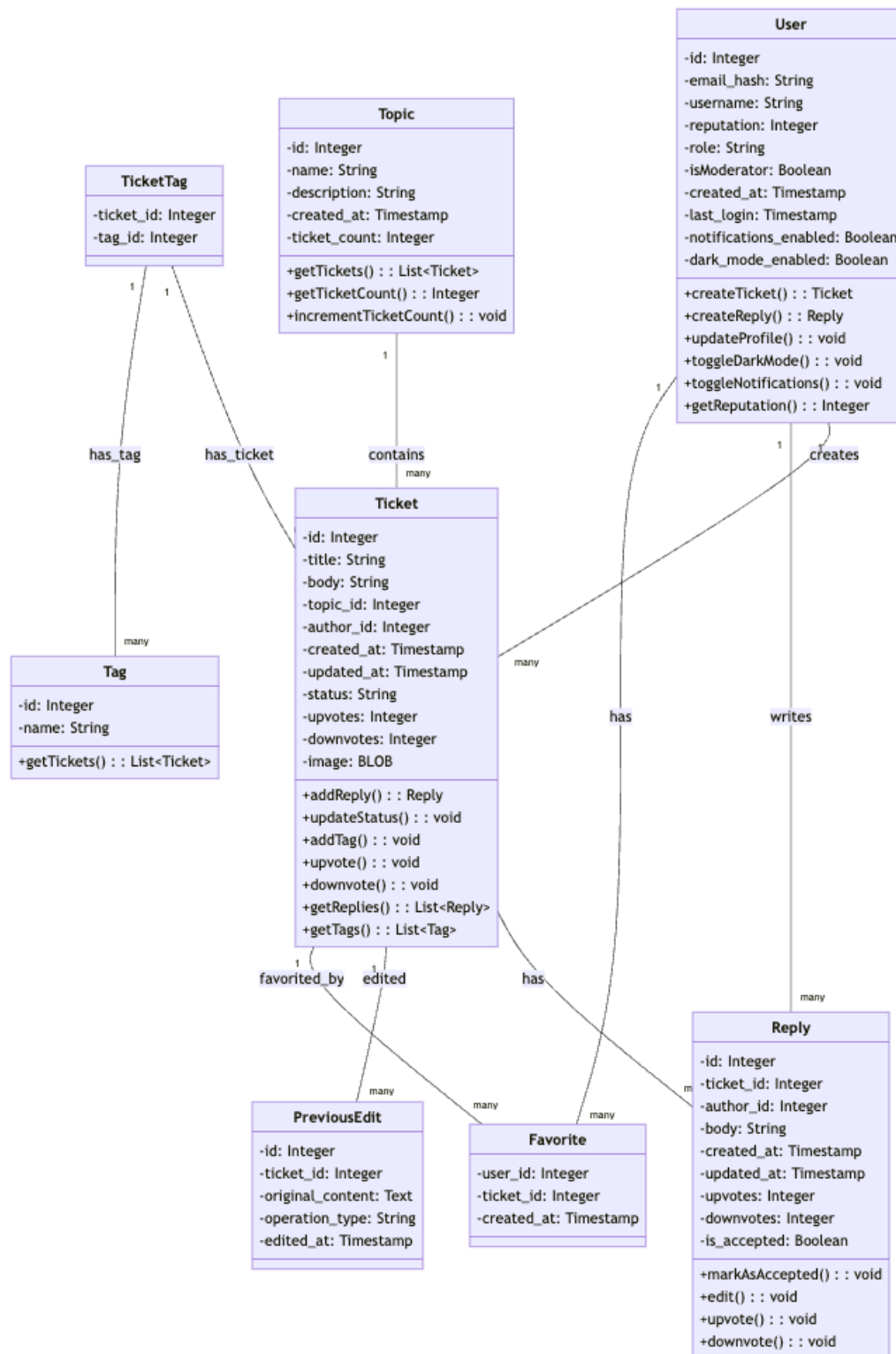
- All endpoints must be protected against common attacks (e.g., SQL Injection, XSS, CSRF).
- Authentication tokens must be validated before accessing protected routes.

1.3.3. Product Backlog

No.	Feature	Role	Business Value	User Impact
1.	Authentication	User	5	5
2.	Username Creation	User	5	5
3.	Creating a Ticket	User	5	5
4.	Ticket Replies	User	5	5
5.	Replies Visibility	User	5	5
6.	Upvote	User	5	5
7.	Downvote	User	5	5
8.	Search Tickets	User	5	5
9.	Privacy Protection	User	5	5
10.	Security	User	5	5
11.	Scalability	Developer	5	4
12.	Ticket Topics	User	4	5
13.	Resolved Ticket	Ticket Author	4	5
14.	Responsivity	User	4	5
15.	Tickets Tagging	User	4	4
16.	Earning Reputation Points	User	4	4
17.	Editing Personal Information	User	3	5
18.	Editing Replies	User	3	5
19.	Editing Tickets	Ticket Author	3	5
20.	Quick Platform Loading	User	3	5
21.	Notifications	User	3	4
22.	Ticket Bookmark	User	3	4
23.	Notifications Settings	User	2	4
24.	Error Handling	User	2	3
25.	User Tooltip	User	1	3

1.4. UML diagrams

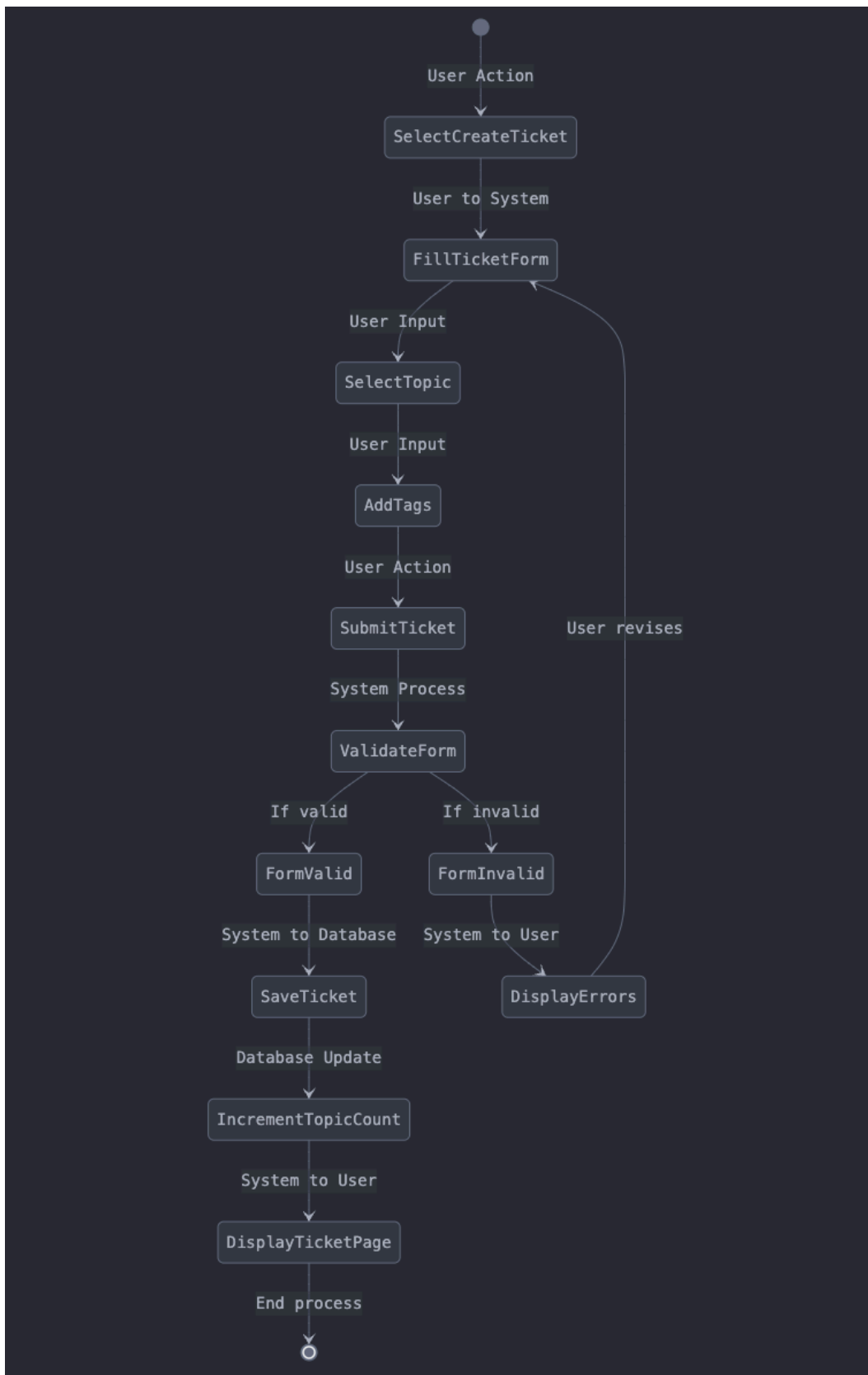
1.4.1. Class Diagram



1.4.2. User Authorization and Profile Creation Activity Diagram



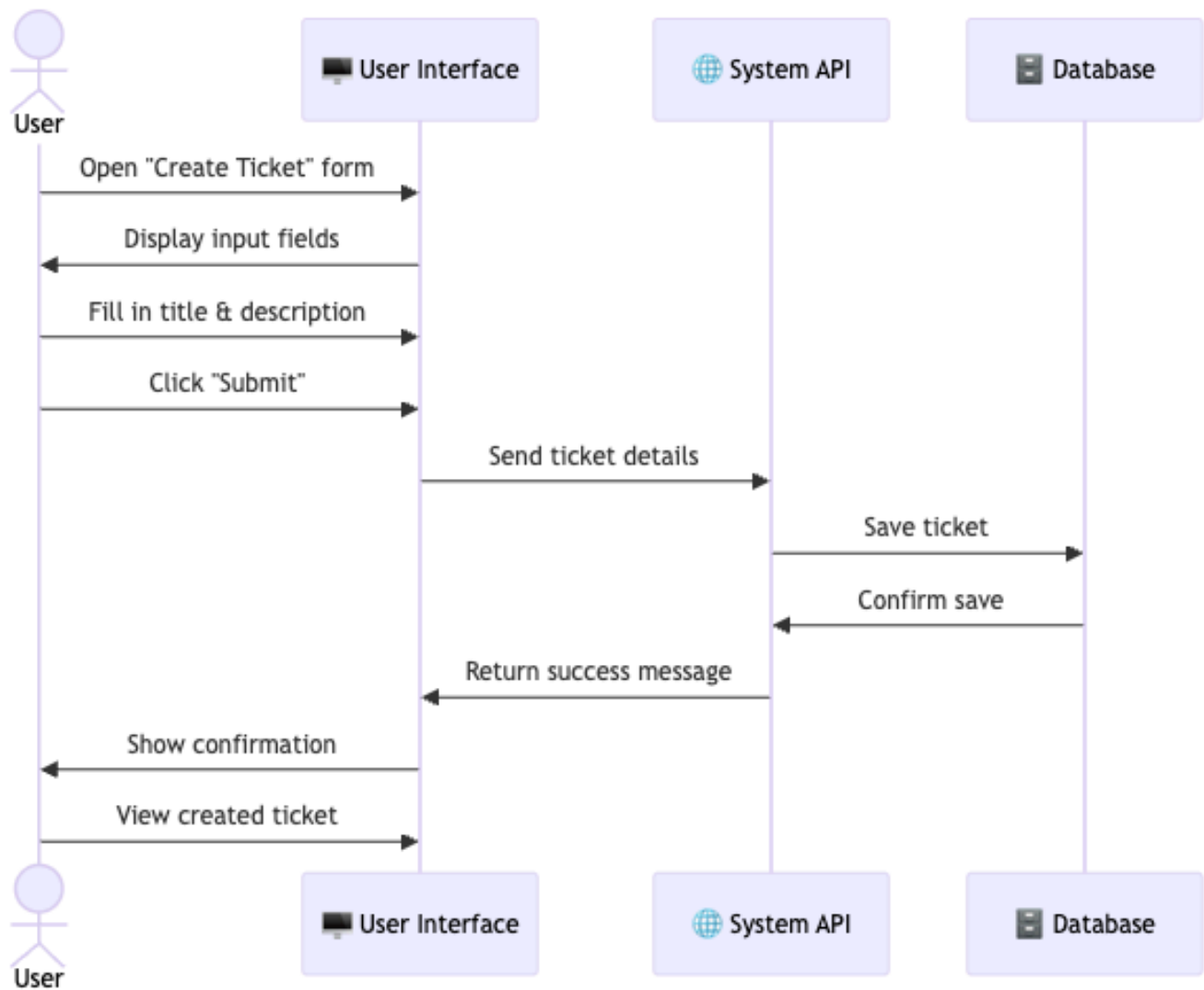
1.4.3. Create Ticket Activity Diagram



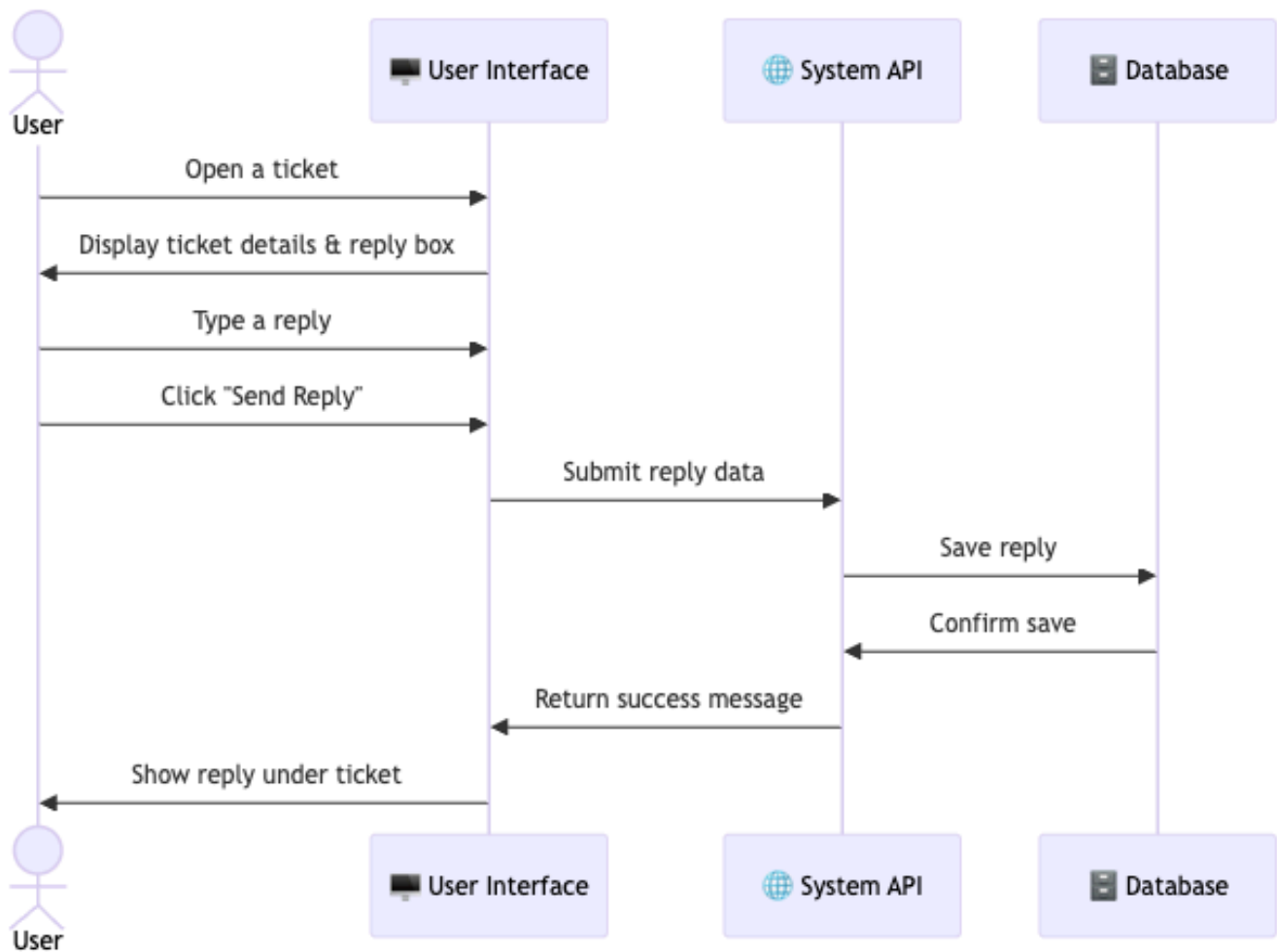
1.4.4. Create and Accept Reply Activity Diagram



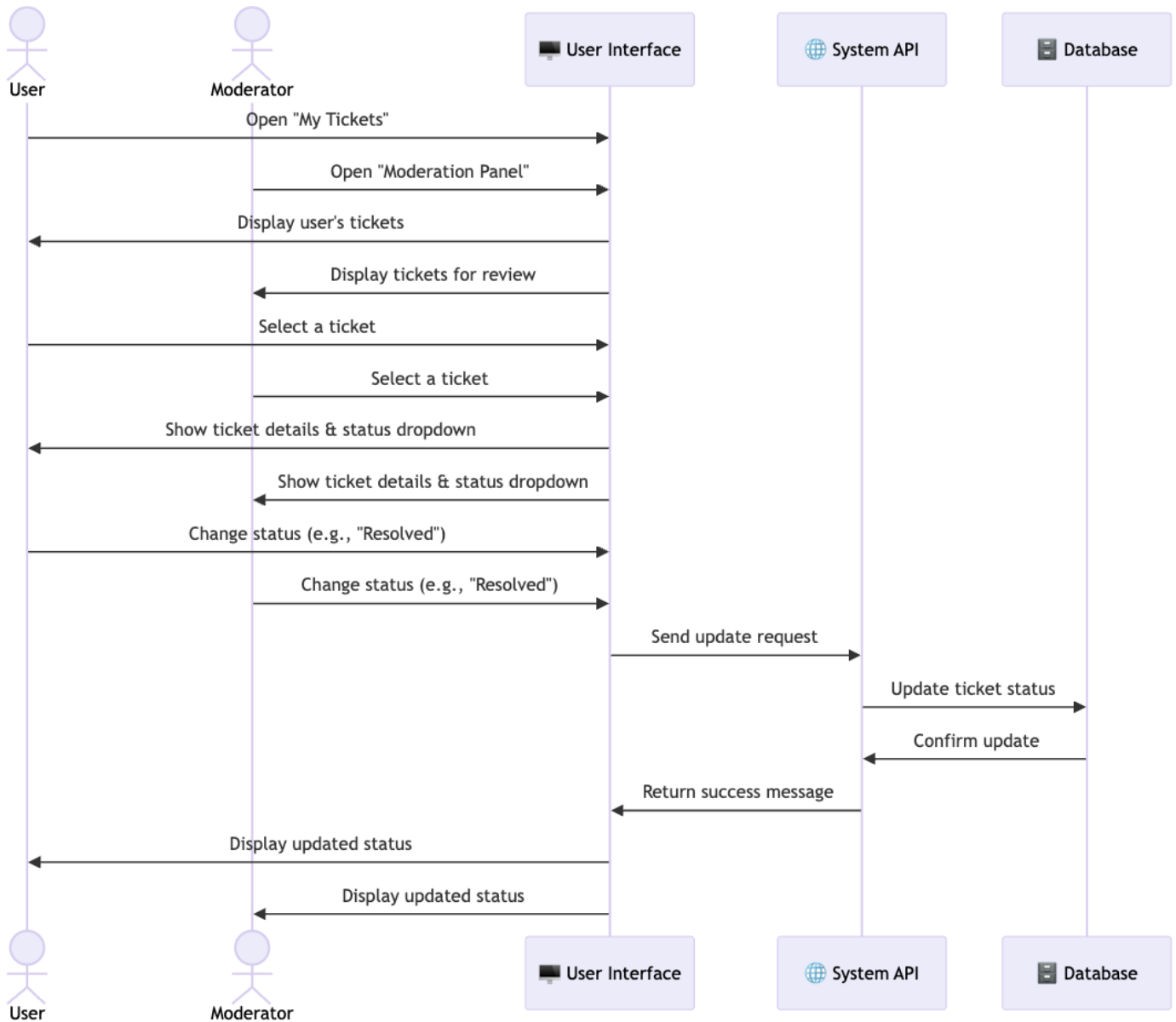
1.4.5. Create Ticket Sequence Diagram



1.4.6. Create Reply Sequence Diagram



1.4.7. Change Ticket Status Sequence Diagram



2. Project Structure

2.1. Technologies

The Anonymous Studying platform is built using a full-stack technology set to create a robust and user-friendly platform. The backend is powered by Node.js and Express.js, with MySQL as the relational database. Authentication is handled with Google OAuth 2.0, ensuring secure access for IBU students and professors, and session management is enabled through Express Session.

On the frontend, the project uses React (v19) with Vite for fast builds. Styling is done with Tailwind CSS. State is managed using React Hooks, and Axios is used for HTTP requests. UI interactivity is enhanced with React Hot Toast for notifications.

Version control is managed via Git, with the codebase hosted on GitHub, and npm is used for package management.

This stack ensures a secure, scalable, and responsive platform tailored for anonymous academic collaboration within the IBU community.

2.2. Architectural Pattern

The Anonymous Studying platform follows a Model-View-Controller (MVC) architectural pattern to ensure a clean separation of concerns between the data layer, business logic, and user interface. This structure is ideal for maintaining a scalable and organized codebase.

- **Model Layer** (backend/models/) manages database structures and interactions using raw SQL with mysql2. It includes models for users, tickets, replies, topics, and voting.
- **Controller Layer** (backend/controllers/) contains business logic such as user authentication, ticket creation, and reply handling.
- **Routing Layer** (backend/routes/) defines RESTful API endpoints and connects them to the appropriate controllers.
- **Frontend** is built with React (v19), using functional components, hooks, and React Router DOM for navigation. Tailwind CSS and reusable components ensure a responsive and consistent UI.
- **Database** is a MySQL relational database with secure, parameterized queries and connection pooling.
- **Authentication** is handled via Google OAuth 2.0 (using Passport.js) and protected with session-based middleware.
- **API Design** follows RESTful principles with clear endpoints, proper HTTP methods, and robust error handling.

This architecture provides a solid foundation for the application, balancing simplicity with the ability to scale as needed. The clear separation of concerns makes it easier to maintain, test, and extend the application over time.

2.3. Database Entities

- users
- tickets -> Represents student-submitted help requests or academic problems.
- replies -> Stores responses to tickets
- topics -> Categories or academic subjects under which tickets (questions/problems) are grouped.
- tags -> Labels used to describe or classify tickets for better filtering and searching.
- ticket_tags -> Table for the many-to-many relationship between tickets and tags. Ensures that a ticket can have multiple tags and vice versa.
- favorites -> Tracks which users have favorited (saved) which tickets. Used to mark tickets for quick access or future reference.
- previous_edits -> Logs history of edits to tickets and replies.

2.4. Design Patterns

- **Custom Hooks Pattern:** used in the frontend, in files:
 - useAuth.js - Authentication state management
 - useTickets.js - Ticket data fetching and state
 - useTopics.js - Topic data management
 - useUserProfile.js - User profile data and operations
 - useCreateTicket.js - Ticket creation logic
 - useUpdateTicket.js - Ticket update logic
 - useUserId.js - User ID management

Why used:

- Encapsulates and reuses stateful logic across components
- Keeps components clean and focused on rendering
- Makes state management more maintainable

- **Higher-Order Component (HOC) Pattern:** used in frontend in file RequireAuth.jsx - Wraps protected routes to ensure authentication

Why used:

- Provides a clean way to protect routes
- Centralizes authentication logic

- **Container/Presentational Pattern:** used in frontend, in files:
 - Dashboard.jsx - Container
 - components folder (various UI components) - Presentational

Why used:

- Separates data-fetching from UI rendering

- Makes components more reusable and easier to test
- Improves code organization and maintainability

- **Provider Pattern:** used in the frontend, in file AuthContext.jsx - Uses React Context API

Why used:

- Provides global state management for authentication
- Makes auth state accessible from any component

- **Repository Pattern:** used in backend/models/, files User.js, Ticket.js, Reply.js, etc.
- Each model handles its own data access

Why used:

- Abstracts data access logic
- Centralizes database operations
- Makes it easier to change data sources if needed

- **Middleware Pattern:** used in the backend, in files:
 - auth.js - Authentication middleware
 - errorHandler.js - Centralized error handling

Why used:

- Handles cross-cutting concerns like authentication and error handling
- Keeps route handlers clean and focused
- Promotes code reuse

- **Factory Pattern:** used in backend/controllers/ - Each controller acts as a factory for handling specific routes

Why used:

- Encapsulates related functionality
- Makes it easier to manage dependencies
- Improves testability

- **Strategy Pattern:** used in backend/config/passport.js - Implements different authentication strategies (Google OAuth, local)

Why used:

- Makes it easy to add new authentication methods
- Keeps authentication logic modular and maintainable

- **Observer Pattern:** used in:
 - frontend - React's state and effect system
 - backend - Event emitters for real-time updates

Why used:

- Enables reactive UI updates
- Decouples components from state management
- Makes it easier to handle asynchronous operations

- **Dependency Injection:** used in backend/app.js - Express app configuration. Route definitions throughout the application

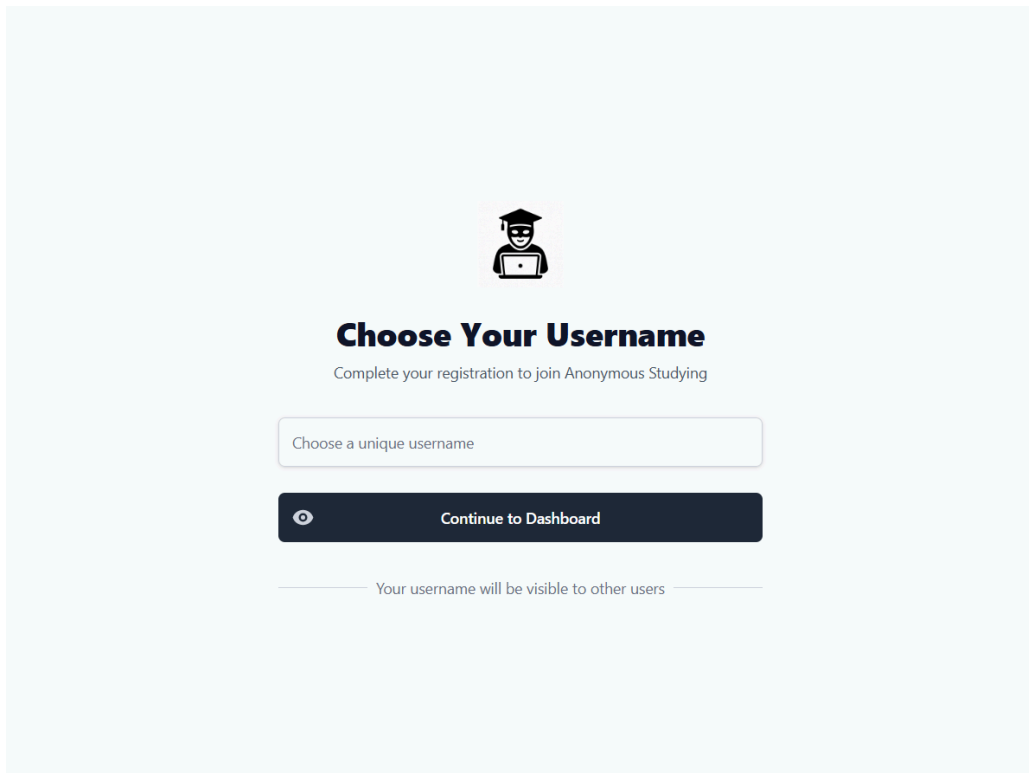
Why used

- Makes dependencies explicit
- Improves testability
- Makes it easier to swap implementations

The combination of these patterns creates a robust, maintainable, and scalable application architecture that follows modern best practices for both frontend and backend development.

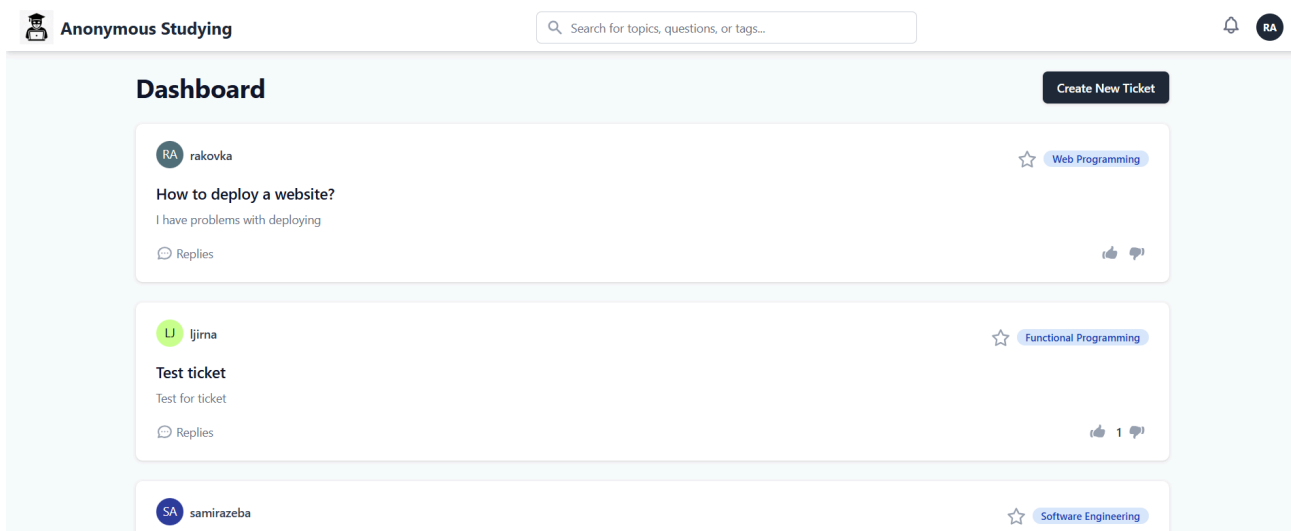
2.5. Project Functionalities and Screenshots

- User Anonymity
 - Secure OAuth authentication
 - Email hashing for privacy protection

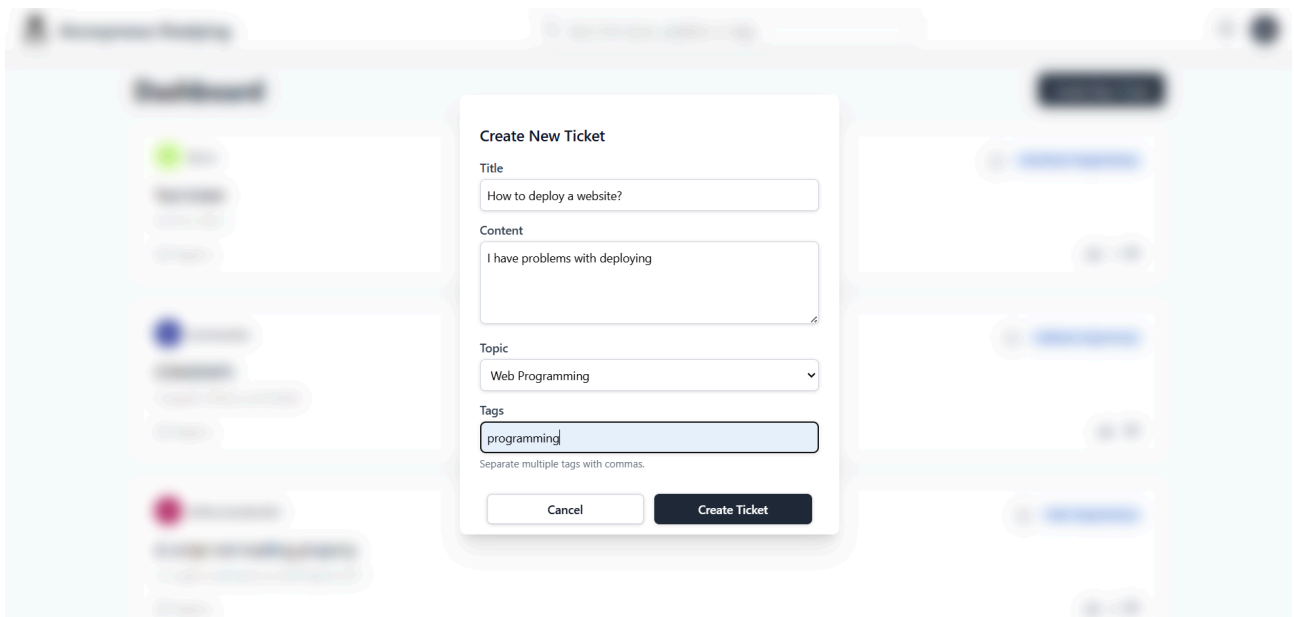



The screenshot shows a registration page with a light blue background. At the top center is an icon of a person wearing a graduation cap and holding a laptop. Below the icon, the heading "Choose Your Username" is displayed in bold. Underneath the heading is the text "Complete your registration to join Anonymous Studying". A text input field with the placeholder "Choose a unique username" is centered. Below the input field is a dark blue button with a white eye icon and the text "Continue to Dashboard". At the bottom, a horizontal line separates the button from the text "Your username will be visible to other users".

- Question and Answer System
 - Ticket-based support system
 - Anonymous posting of questions
 - Community-driven answers





The screenshot shows the "Anonymous Studying" dashboard. At the top left is the site logo and name. At the top right is a search bar with the placeholder "Search for topics, questions, or tags...". Below the search bar is a "Create New Ticket" button. The main content area is titled "Dashboard" and contains a list of questions. Each question card shows the user's profile picture and name, the question title, the question text, the number of replies, and the topic tag. The first question is by user "RA rakovka" titled "How to deploy a website?" with the text "I have problems with deploying" and 1 reply. The second question is by user "U ljirna" titled "Test ticket" with the text "Test for ticket" and 1 reply. The third question is by user "SA samirazebe" titled "Test ticket" with the text "Test for ticket" and 1 reply. The topics are "Web Programming", "Functional Programming", and "Software Engineering" respectively.






 Anonymous Studying

Search for topics, questions, or tags...

How to deploy a website?



Asked Jun 15, 2025, 02:02 AM by  rakovka Web Programming

  rakovka

Web Programming


I have problems with deploying

Jun 15, 2025, 02:02 AM


 


0 Replies

Add Reply



 Posting as:
rakovka

Write your answer here...






 Anonymous Studying

Search for topics, questions, or tags...

CO project issue



Asked Jun 10, 2025, 05:45 PM by  edhemmomcina Mathematics

  edhemmomcina


Mathematics

I cannot create this program to work on my machine. Does anybody have experience with linux?

Jun 10, 2025, 05:45 PM

 2 



1 Replies

 balibegipo

Accepted Answer


No experience with linux but some with co project programm. Make sure to specify the issue so i can help frther.



Jun 10, 2025, 10:36 PM

 1 

This ticket has been resolved with an accepted answer.


- Search

**Anonymous Studying**



Search Results for "project"


Tickets



**edhemstudentski**


Mathematics

Regarding Web Programming Project

Having issues initialising spapp on the project. Anybody have any suggestions?

 Replies


 2  0



**edhemmorcina**

Mathematics


CO project issue



I cannot create this program to work on my machine. Does anybody have experience with linux?


 Replies

 2  0

- Account

**Anonymous Studying**




**rakovka**
0 reputation points

[Settings](#) [Favorites](#) [Previous Tickets](#)


Account Settings

Profile Picture


**RA** Your avatar is automatically generated from your username.

Current Username

New Username

 Anonymous Studying

Search for topics, questions, or tags...


 **rakovka**
0 reputation points

Settings **Favorites** Previous Tickets


Favorite Tickets

[Euclidian algorithm](#) ★
By balibegipo • Physics 6/10/2025
2 upvotes 0 replies

[Regarding Web Programming Project](#) ★
By edhemstudentski • Mathematics 6/11/2025
2 upvotes 0 replies

 Anonymous Studying

Search for topics, questions, or tags...

 **rakovka**
0 reputation points

Settings Favorites **Previous Tickets**

Your Previous Tickets

[How to deploy a website?](#) 0 0 0 0
General • open 6/15/2025

2.6. Tests

This project uses Selenium-based end-to-end tests, located in the **/tests** directory, to verify user flows in the application through real browser automation. These tests simulate real user interactions such as signing in with Google, creating tickets, replying, favoriting, and managing user accounts.

3. Conclusion

The Anonymous Studying platform successfully delivers a secure, anonymous environment for IBU students and professors to exchange academic support. Core features like ticket creation, anonymous replies, and authentication via Google OAuth were implemented effectively, resulting in a functional and user-friendly system.

The project architecture, built with React, Node.js, and MySQL, and structured using design patterns like MVC and Custom Hooks, ensured scalability and maintainability. While the integration of anonymous interactions and secure access was challenging, it was achieved with satisfying results.

Future improvements could include advanced moderation tools, and smarter content recommendations. Overall, the project met its objectives and provides a strong foundation for continued development and impact within the IBU community.

4. Individual Contributions

Edhem Rogo (50%)

He was primarily responsible for the frontend development, authentication, and client-side logic. His contributions included:

- **Frontend Development:** Built the user interface using React (v19), with routing, views, and components styled using Tailwind CSS.
- **Custom Hooks and State Management:** Developed custom hooks (e.g., useAuth, useTickets, useTopics) to manage authentication state, ticket interactions, and topic filtering.
- **Component Structure and Reusability:** Applied the Container/Presentational and Provider patterns to keep components modular and maintainable.
- **Collaboration:** Worked closely with the second other to ensure UI elements correctly displayed the data retrieved from the backend, and provided feedback to improve API structure and usability..

Sadžida Rakovac (50%)

She focused on the backend development, system design, testing and UI/UX aspects of the project. Her contributions included:

- **Project and Release Planning:** Defined the overall release plan, including the core platform milestones and progressive feature enhancements.
- **Backend Development:** Implemented the core backend logic using Node.js and Express.
- **Testing:** Designed and executed end-to-end tests using Selenium to simulate user interactions like login, ticket creation, and replies.
- **Collaboration:** Coordinated with another student to ensure the frontend design aligned with the backend functionalities, and supported integration through testing and feedback.