SWE 573

Software Development Practice

Connect the Dots

http://52.202.203.12:3000/

Project Report

by

Batuhan CÖMERT

2023719147

GitHub Repo	: https://github.com/comertbatuhan/573
GitHub Tag Version	: https://github.com/comertbatuhan/573/releases/tag/v0.9
Course Instructor	: Prof. Dr. Suzan Üsküdarlı
Submitted to	: Moodle
Delivered to	: Prof. Dr. Suzan Üsküdarlı
Date of Submission	: 18/05/2025
Date of Delivery	: 20/05/2025

Department of Computer Engineering

Boğaziçi University

Bebek, Istanbul

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 Spring 2025 semester project reported in this report, I Batuhan Cömert declare that:

- I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the Spring 2025 semester.

- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself.

- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

Batuhan Cömert

[illegible]

Please, use the following username and password for testing the deployed application.

Username: “usertest”

Password: “Usertest123.”

Also, please check the below URL for demo video:

https://drive.google.com/file/d/1ZlQJGnsMiL8c0vIJxQsRhs7VUbbFbWKU/view?usp=sharing

Contents

1. Overview

One of the most important benefits of the internet today is to access to information. However, we are not always able to access information in a structured form, and even if we do, we cannot trust it. This is where wikidata comes into play. Wikidata is a collection of "facts" where related facts are linked to each other. This way, it represents a lot of information in a highly organized matter. But what about the topics, people or incidents that are not included in wikidata? This is where connecthedots comes into play. Thanks to the Connecthedots application, we give people the chance to produce and share information collaboratively. The wikidata is utilized while doing this.

2. Software Requirements Specification

It is important to mention that the requirements specified below are frozen at the 1st month of the 2025 Spring semester. Throughout the semester, the requirements became much more clear as discussed in classes. However, for the sake of best practices, the requirements below haven't been touched for months since the requirements specification is "the contract" between customers, stakeholders, and software team. For this reason, it should be noted that the specified requirements below would have been much more pinpoint if I were to change them.

Functional Requirements

2.1. User Management

2.1.1. Users should be able to sign up, log in, and log out.

2.1.2. Users should be able to update their profile information.

2.1.3. Users should be able to delete their account.

2.2. Node Management

2.2.1. Users should be able to create nodes by entering a name (e.g., "Chicago").

2.2.2. When creating a node, the system should retrieve attributes from Wikidata.

2.2.3. If no data is found in Wikidata, the user should manually enter attributes.

2.2.4. If data is found in Wikidata, the user still should be able to manually enter attributes.

2.2.5. Users should be able to edit node attributes.

2.2.6. Users should be able to delete nodes.

2.2.7. The name of nodes should be able to be seen on the node like "Chicago" written just above a node.

2.2.8. Users should be able to connect two existing nodes.

2.2.9. While connecting nodes, users must define the relationship (e.g., "Emily was born in Chicago").

2.2.10. Users should be able to edit the relationship between connected nodes.

2.2.11. Users should be able to delete a connection between two nodes.

2.3. Discussion Page

2.3.1. Users should be able to create posts in discussion pages.

2.3.2. Users should be able to edit their posts and comments.

2.3.3. Users should be able to delete their posts.

2.4. Search & Navigation

2.4.1. Users should be able to search discussion posts.

2.5. Data Integrity & Moderation

2.5.1. There should be a single type of user that is "regular user" where each user is able to use all functionalities.

Non-Functional Requirements

System constraints and performance expectations.

2.6. Performance

2.6.1. The system should handle at least 100 concurrent users smoothly.

2.7. Scalability

2.7.1. The backend should support scaling using Docker.

2.8. Security

2.8.1. Rate-limiting should be enforced to prevent API abuse.

2.9. Reliability

2.9.1. If Wikidata is unavailable, the system should gracefully handle failures and allow manual entry.

2.9.2. The system should support database backups and restore functionality.

2.10. Maintainability

2.10.1. The system should be modular, allowing easy updates and feature additions.

2.10.2. Logs should be maintained for monitoring errors and debugging.

2.10.3. The frontend and backend should be containerized using Docker.

2.11. Usability

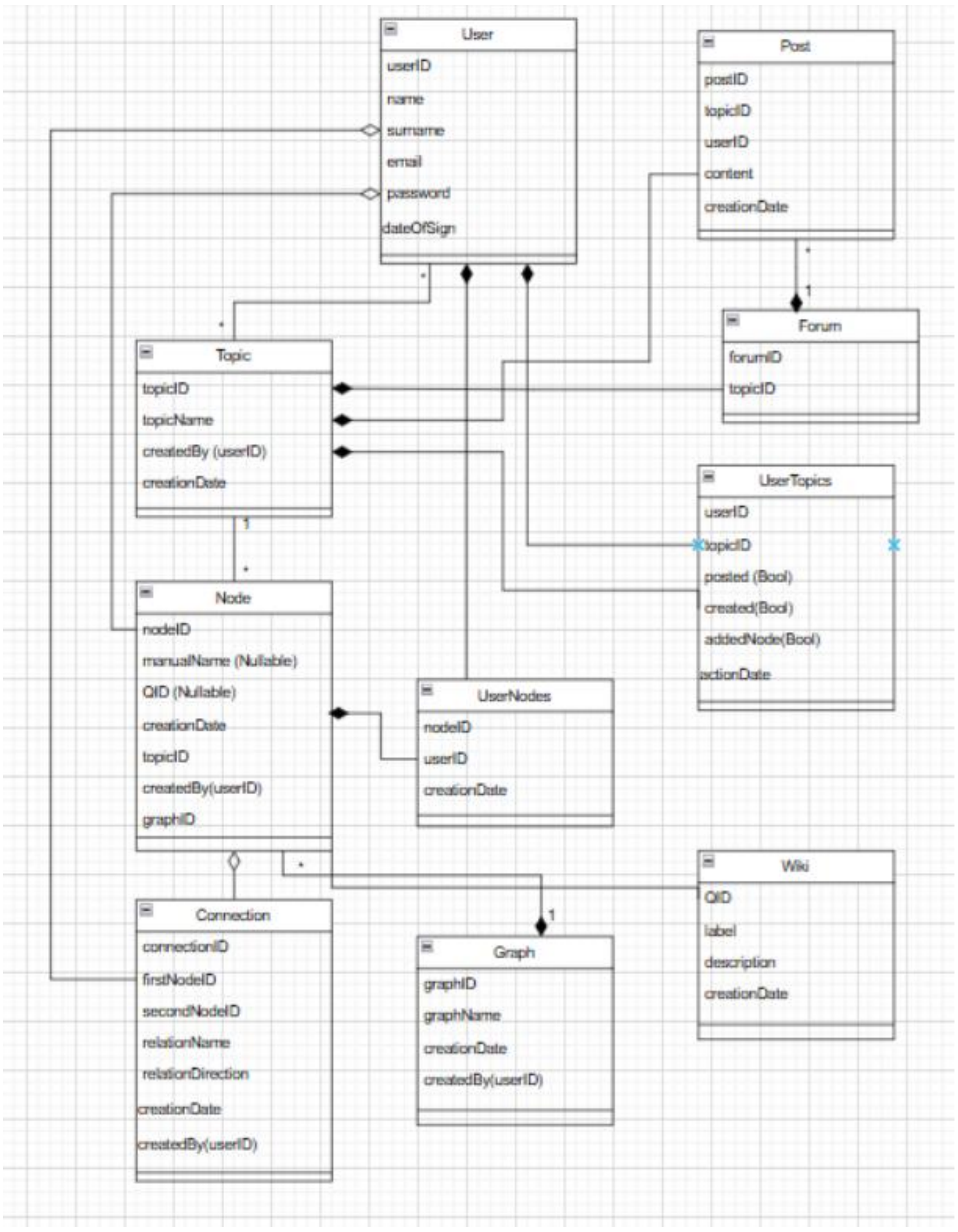
2.11.1. The UI should be minimal but functional for easy node creation and connection.

2.11.2. Error messages should be clear and actionable.

2.11.3. Users should be guided when creating nodes, relationship between nodes, post on discussion page.

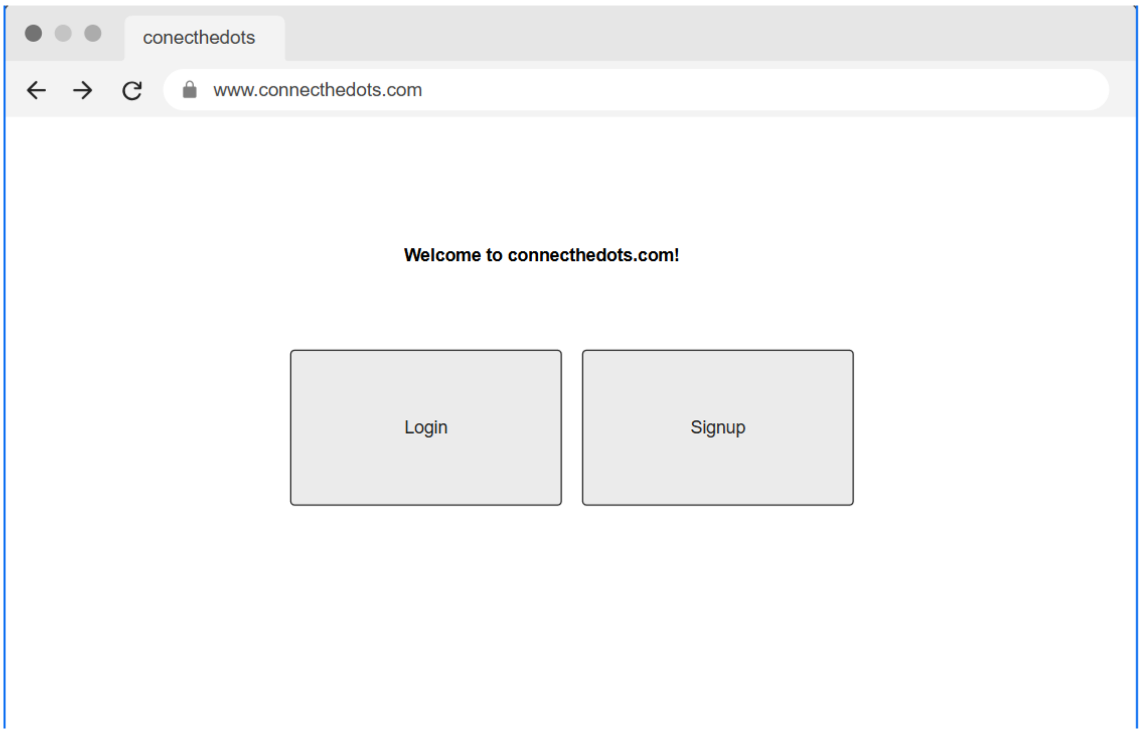
3. Design Documents

3.1. UML Class Diagram

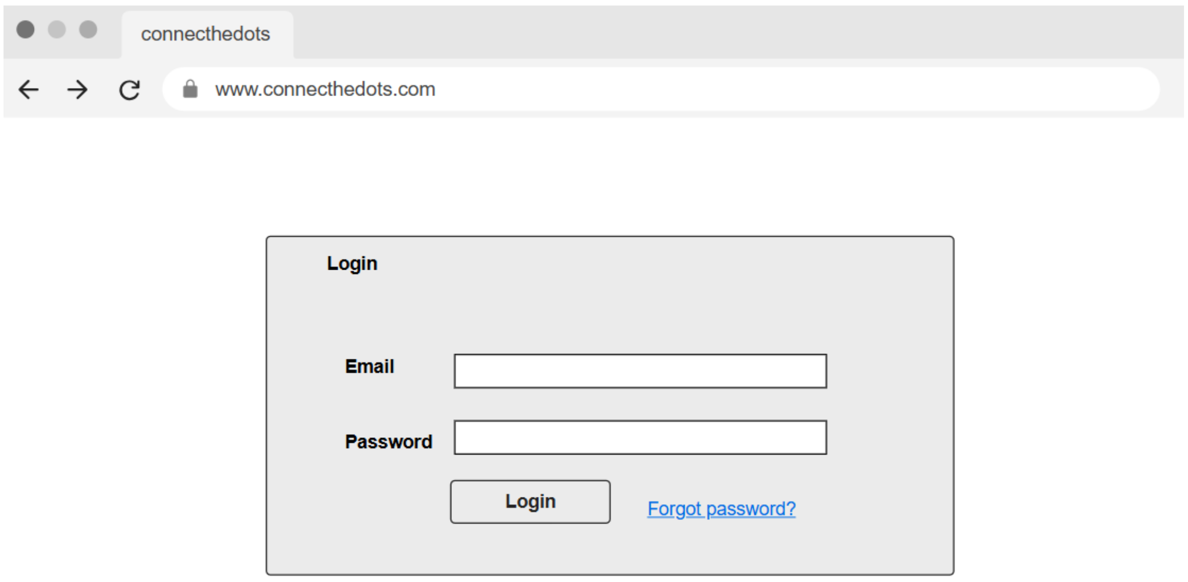


3.2. Mockups

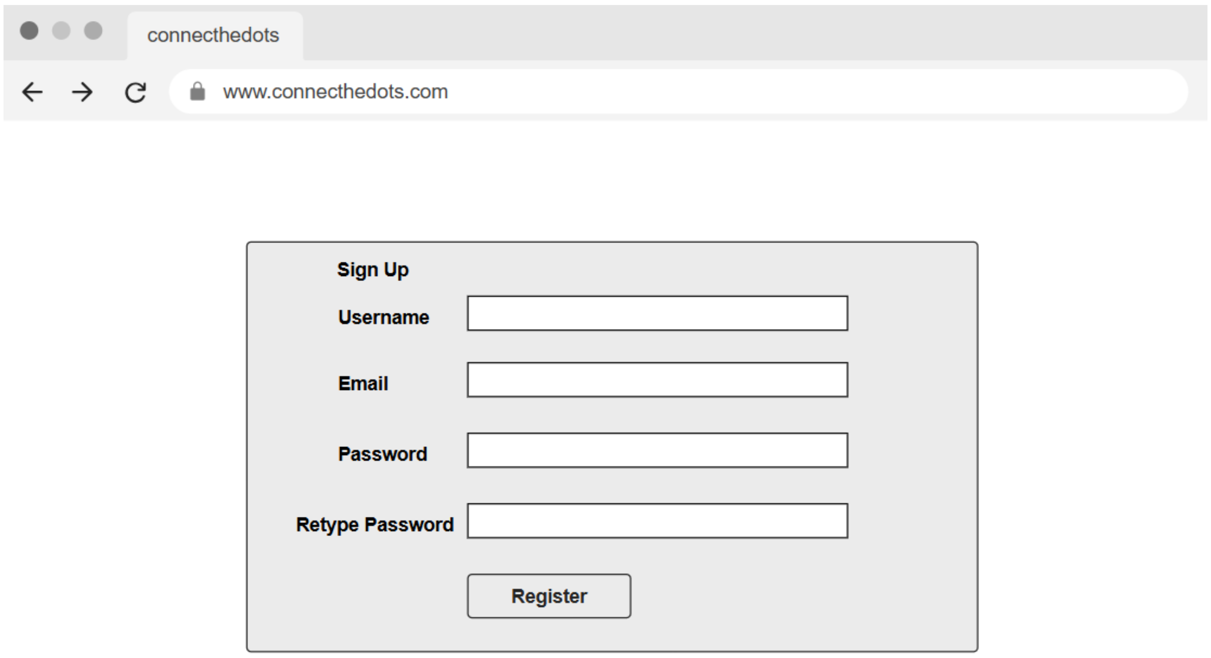
3.2.1. Home Page



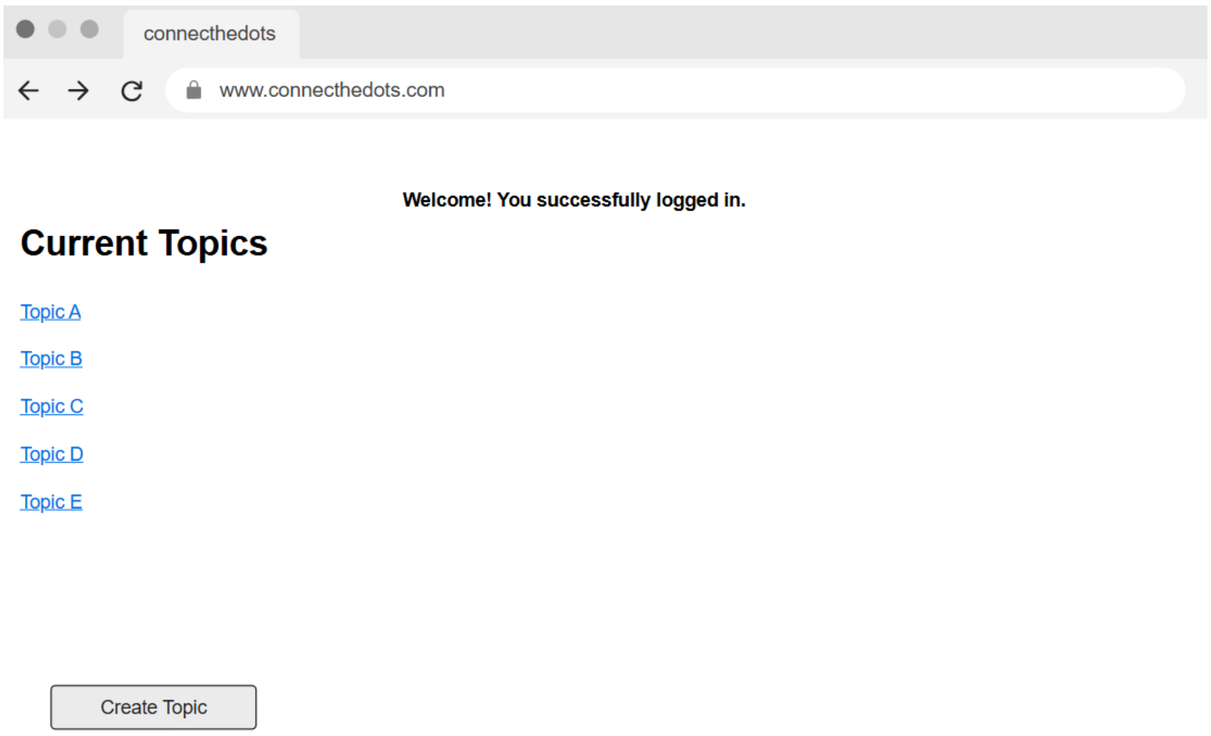
3.2.2. Log-in Page



3.2.3. Sign-up Page



3.2.4. Dashboard Page


Welcome! You successfully logged in.

Current Topics

Topic A

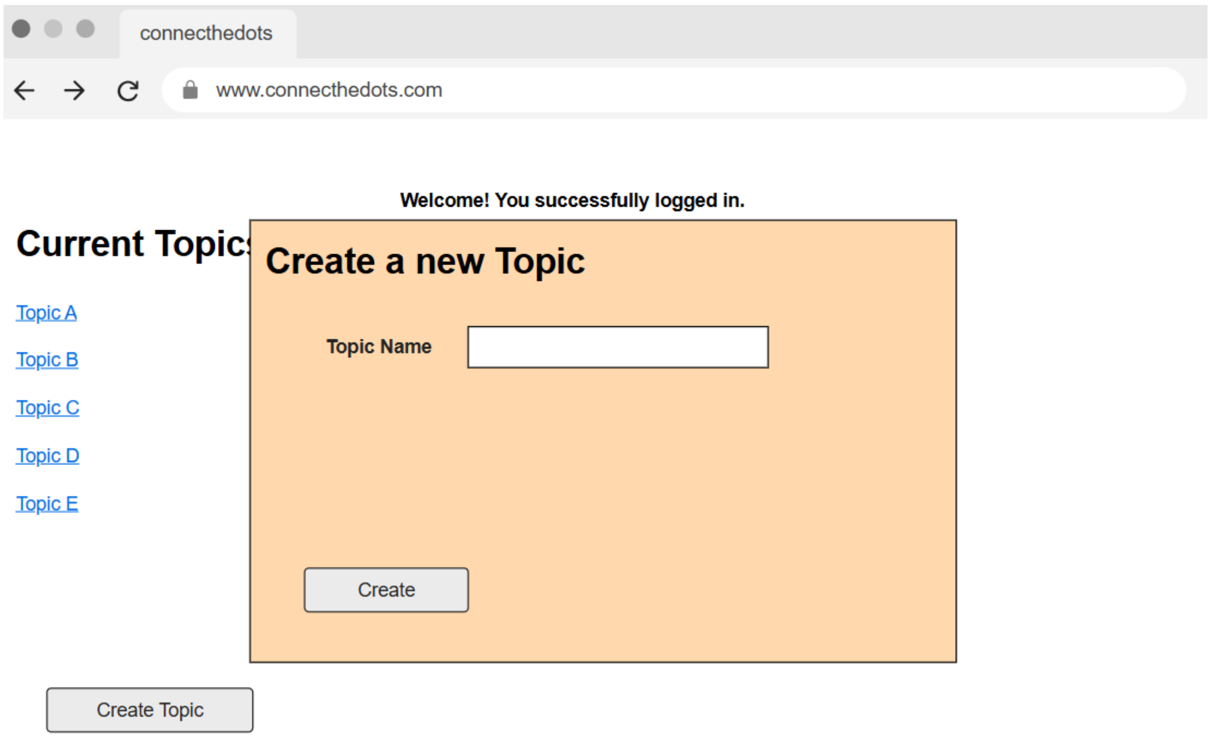
Topic B

Topic C

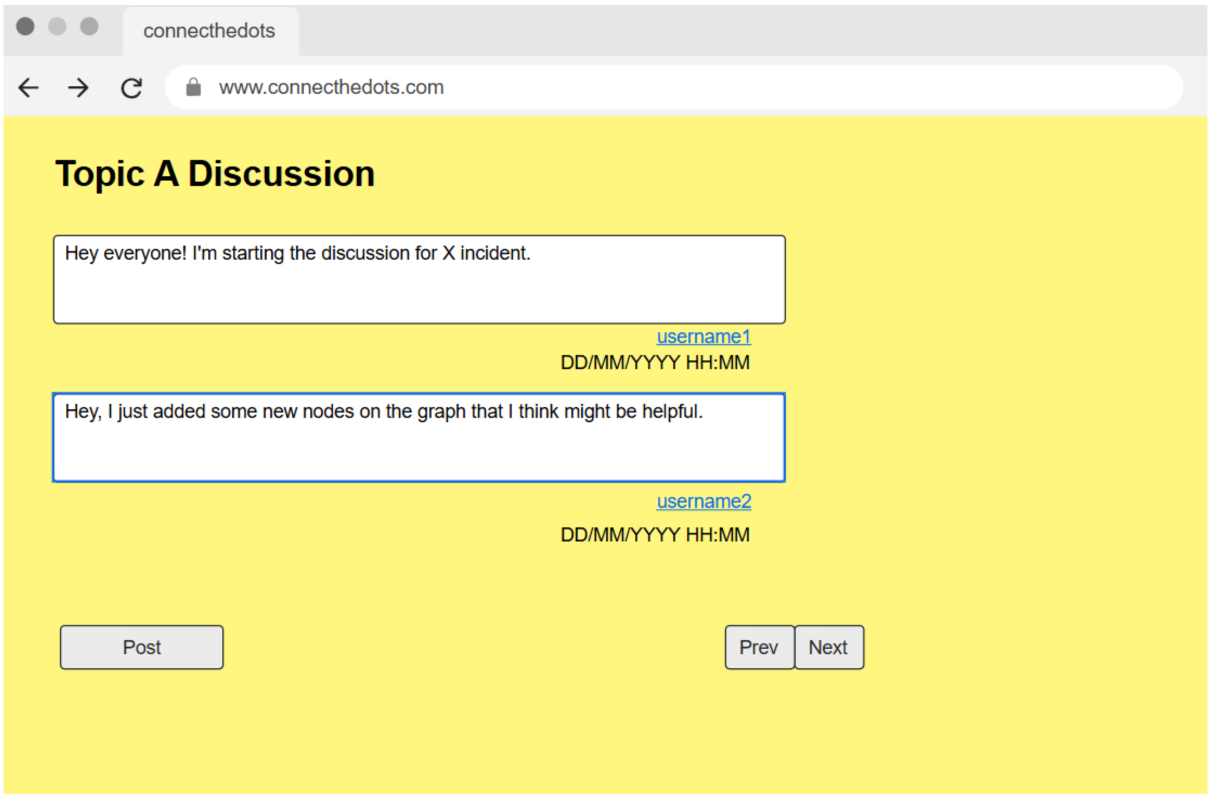
Topic D

Topic E

3.2.5. Topic Creation



3.2.6. Discussion Page



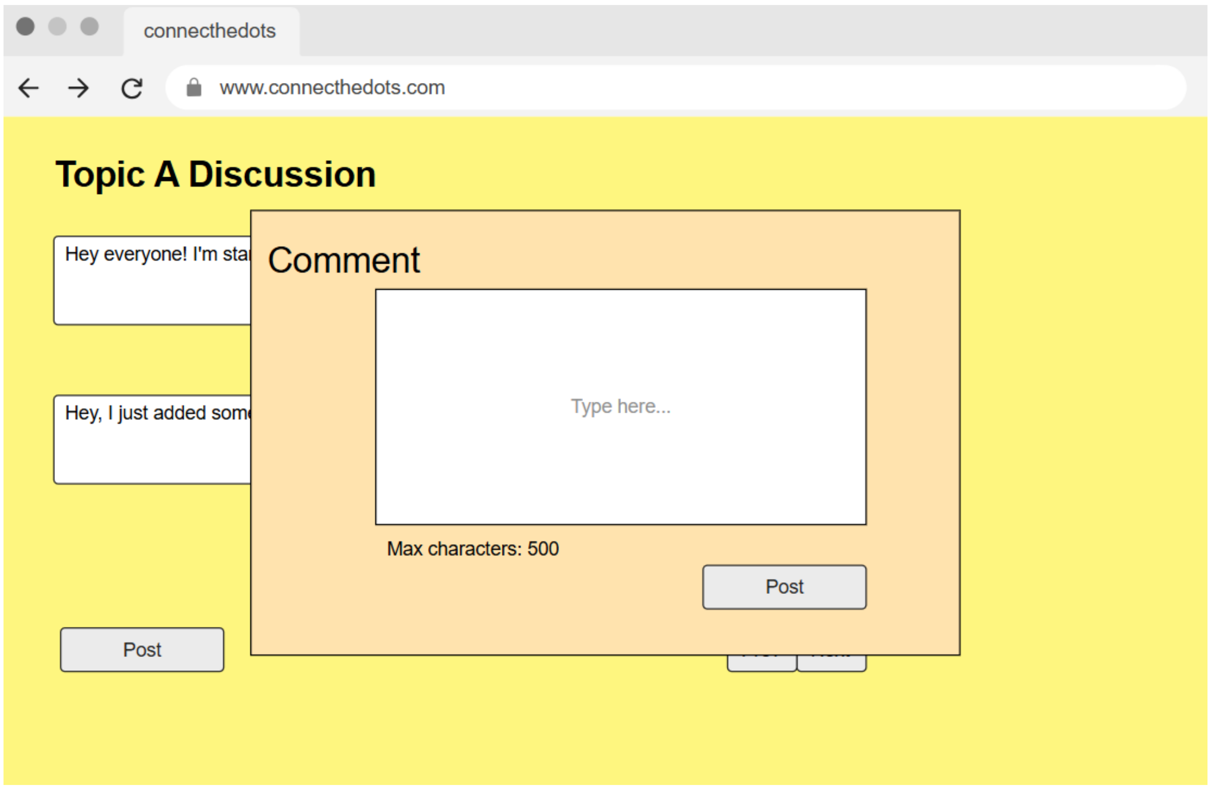
Hey everyone! I'm starting the discussion for X incident.

username1
DD/MM/YYYY HH:MM

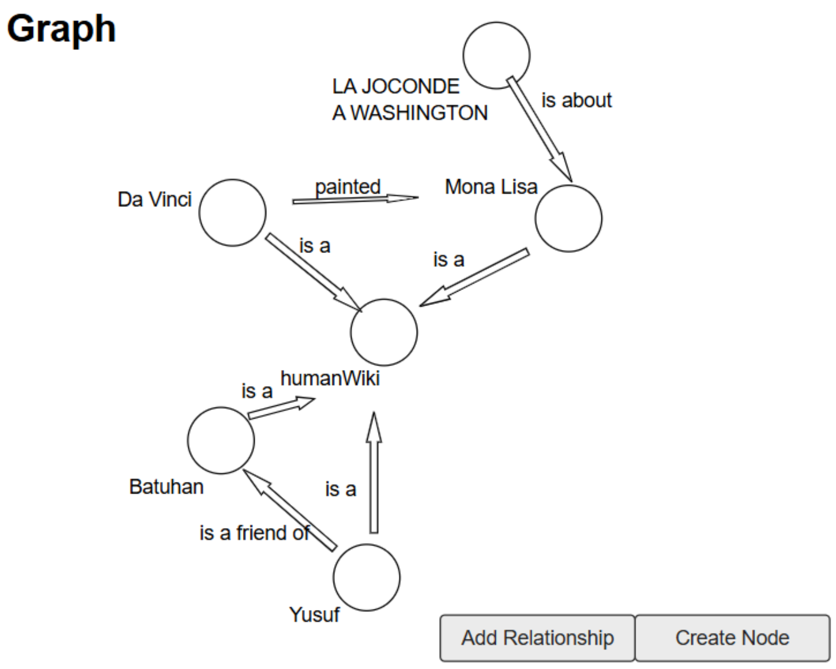
Hey, I just added some new nodes on the graph that I think might be helpful.

username2
DD/MM/YYYY HH:MM

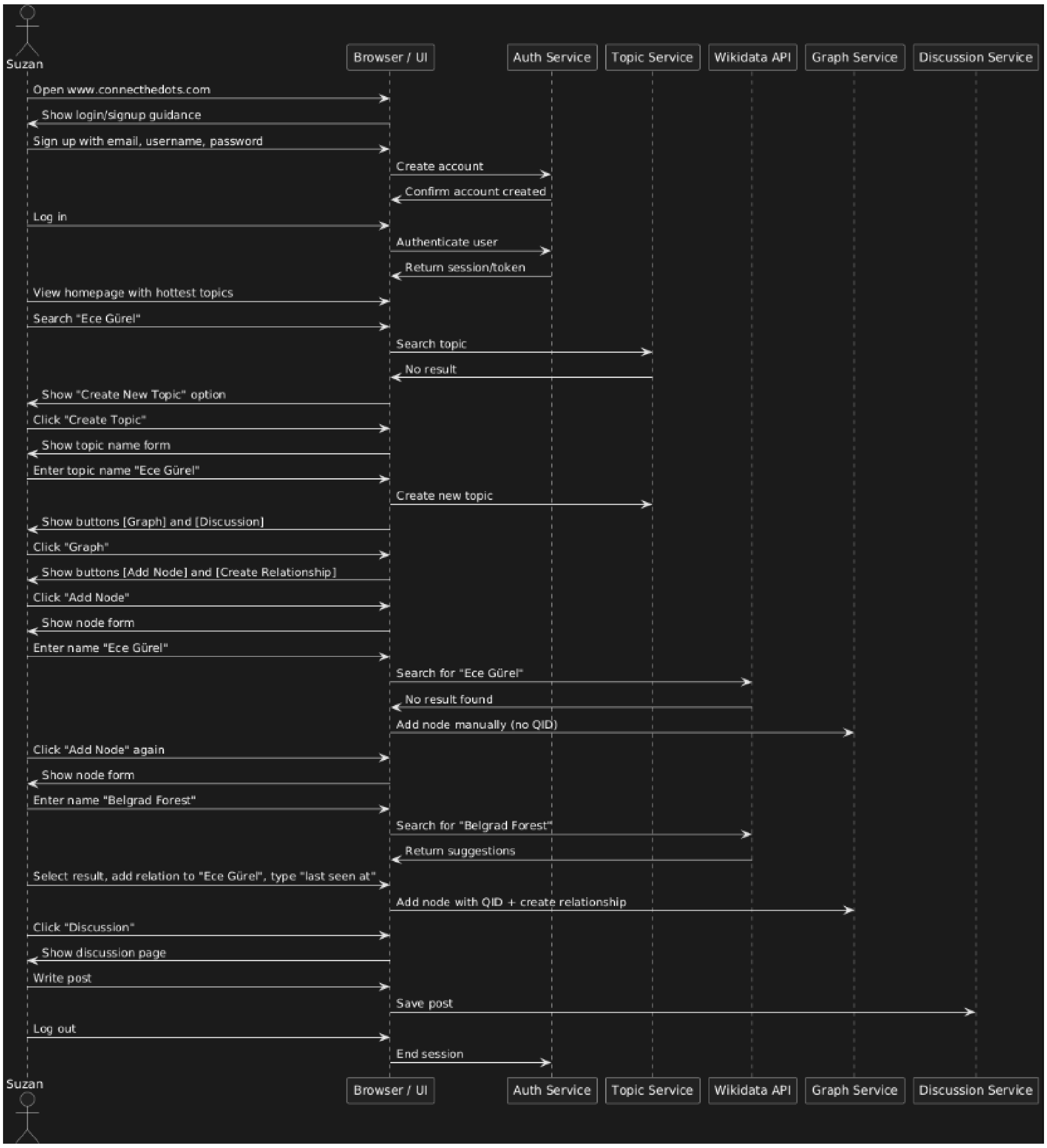
3.2.7. Posting on Discussion Page



3.2.8. Graph Page



3.3. Sequence Diagram



4. Status of the Project

There are two remaining requirements **2.6.1, 2.8.1** and one updated requirement **2.4.1**. The requirement **2.4.1** — “Users should be able to search discussion posts.” — implemented as “Users should be able to search nodes that are available on graphs.”. The requirement **2.6.1** — “The system should handle at least 100 concurrent users smoothly.” — hasn’t been tested, hence it is not known that whether it is succeeded or not. Lastly, **2.8.1** — “Rate limiting should be enforced to prevent API abuse." — is not explicitly implemented. All requirements except these three we defined for our project are now complete: each has been documented, passed unit tests, and has been deployed seamlessly to the production environment. The live version of the application can be accessed at http://52.202.203.12:3000/. Since all services run in Docker containers, scalability and portability criteria are fully met. In summary, we have completed approximately 98% of our project schedule, and the application operates smoothly.

5. Installation Instructions

Tool	Tested Version	Notes
Docker	>=24	Install Docker
Docker Compose	V2	
Git	>= 2.40	For cloning & tagging

Table 5. 1. *Required Tool Versions*

5.1. Code Snippets

Clone Repo using Bash / macOS / Linux

```
git clone https://github.com/comertbatuhan/573.git
cd 573
git checkout v0.9
```

Create .env file

```
POSTGRES_DB=connecthedots
POSTGRES_USER=postgres
POSTGRES_PASSWORD=REPLACE_ME
POSTGRES_HOST=db
POSTGRES_PORT=5432
DATABASE_URL=postgres://postgres:REPLACE_ME@db:5432/connecthedots
DJANGO_SETTINGS_MODULE=connecthedots.settings
DJANGO_DEBUG=False
SECRET_KEY=REPLACE_ME
ALLOWED_HOSTS=localhost,127.0.0.1
CORS_ALLOWED_ORIGINS=http://localhost:3000
REACT_APP_API_BASE=http://localhost:8000
```

Overwrite config.js

```
const API_URL =
  process.env.REACT_APP_API_BASE?.trim() || DEFAULT_URL;
export default API_URL;
```

Building & Running Containers using Bash

```
docker compose up --build -d
docker compose exec backend python manage.py migrate
docker compose exec backend python manage.py createsuperuser
```

6. User Manual

To start using the application, create an account on the “Sign Up” page or “Login” with your existing account. After logging in, you will be automatically directed to the Dashboard. You can see popular topics (“Most Interacted Topics”) on the Dashboard, search for a title with the search bar, or open a new topic by typing a name and description with no spaces and underscores with the “Create Topic” button. Clicking on a topic opens the Forum view: where the latest posts are listed in chronological order. You can write a new message of up to 500 characters with the “Create Post” button, and use the three dots at the top right of the card to edit/delete the post. You can link to other topics by typing @TitleName in the text, and quickly switch to the relevant forum by clicking on the colored tag.

The “Go to Knowledge Graph” button on the forum page takes you to the Graph screen of the same topic. Here you'll see an interactive chart with drag-and-drop support. When adding a new node with “Add Node”, when you type at least three characters, a Wikidata search is automatically performed—if one of the results is selected, the description field will be filled automatically. With the “Add Edge” option, first search and select the source and target nodes, then enter the relationship name (e.g. lives in) and save. The edges appear animated and you can edit or delete them with the pencil-trash icons on the node/edge cards. When you click on a node, the panel that opens on the right shows the description, the Wikidata link, and a list of associated nodes. The coordinates of each node you reposition by dragging are automatically saved.

Navigation is done everywhere with the buttons at the top: “Back to Forum” returns from the graphic to the forum, “Back to Dashboard” returns from both the forum and the graphic to the main panel. You can manage personal information such as password and log out from your profile page. Thus, the application. It combines topic-focused forum discussion with real-time infographics, allowing you to explore content in both textual and visual form.

7. Test Results

Backend

```
Found 17 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.................
----------------------------------------------------------------------
Ran 17 tests in 25.050s

OK
Destroying test database for alias 'default'...
```

Frontend

```
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        15.321 s
Ran all test suites related to changed files.
```

References

Django Documentation. (2025). Retrieved from https://learndjango.com/tutorials/django-login-and-logout-tutorial

Moqups. (2025). Retrieved from https://moqups.com/

Wikidata. (2025). Retrieved from https://www.wikidata.org/wiki/Wikidata:Main_Page