Software Development Practice (SWE573)

Project Report

Connect The Dots

Prepared by:

Mehmet Yusuf Bayam

Boğaziçi University
Software Engineering MS Program

Spring 2025

18 May 2025

HONOR CODE

Related to the submission of all the project6eliverables for the Swe573 Spring 2025 semester project reported in this report, I Mehmet Yusuf Bayam 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.

Mehmet Yusuf Bayam

Deployment URL: http://54.163.77.79:3000/
Git Repository: https://github.com/yusufbayam/SWE573
Git Tag Version: https://github.com/yusufbayam/SWE573/releases/tag/v0.9

Contents

Chapter 1

Test Credentials

Credentials to test the deployed system with **username: password** format:

- yusuf: ysf12345 (admin account)
- qq: qq (admin account)
- zarife: zarife
- spike: spike

Chapter 2

Declaration of Work

I declare that all work in this project has been performed by myself.

Chapter 3

Overview

Connect-The-Dots is where curious minds come together to build and explore knowledge in a whole new way. Think of it as a collaborative playground where you can map out ideas visually, connecting the dots between concepts to reveal the bigger picture. Connect-The-Dots lets you create dedicated "Spaces" around topics you care about, whether that's renewable energy, fantasy literature, or quantum physics—then transform abstract ideas into interactive visual networks that anyone can understand at a glance.

What makes Connect-The-Dots special is how it bridges the gap between Wikipedia-style facts and your own unique insights. Using seamless integration with Wikidata, you can pull in established concepts with just a few clicks, then customize them to fit your space. Want to show how renewable energy technologies impact climate change metrics? Just search, select and connect. The visual graph makes relationships clear, showing connections that might get lost in traditional text formats.

The magic really happens when people collaborate. While individual users create and contribute nodes and connections, the knowledge belongs to the space itself. Growing and evolving as more people join in. You can discuss ideas and build on others' work, creating a dynamic, living knowledge network that's greater than the sum of its parts. Whether you're a teacher looking to explain complex ideas, a researcher organizing findings, or just someone who loves diving deep into interesting topics, Connect-The-Dots gives you the tools to explore, share and discover in a way that feels both intuitive and powerful.

Chapter 4

Software Requirements Specification

4.1 Glossary

Term	Definition
Regular User	A standard user who can create and participate in collaboration spaces.
Admin User	A privileged user with the ability to manage spaces, discussions and user-generated content.
Space	A collaborative environment where users can add nodes, create discussions and interact.
Graph Visualization	A visual representation of nodes and edges within a collaboration space.
Discussion	A threaded conversation within a space where users can post, reply and vote.
Node	A graphical element in a space representing a concept, idea, or piece of information.
Edge	A connection between two nodes, representing relationships in the graphical visualization.
Tag	Keywords assigned to spaces for categorization and searchability.
Search	A functionality allowing users to find spaces by title/tags and other users by username.
Wikidata	An external structured knowledge base integrated as a data source for the system.

4.2 Functional Requirements

4.2.1 Registration And Login

2.2.1.1. Users shall be able to register to the system with an unique e-mail, unique username, password, age and profession data.

2.2.1.2. Users shall be over 18 years old.

2.2.1.3. Users shall be able to login using their e-mail and password.

4.2.2 Authorization

2.2.2.1. There shall be two types of users in the system; admins and regular users.

2.2.2.2. An admin shall be able to do everything a regular user can.

2.2.2.3. An admin shall be able to delete any irrelevant, inappropriate or incorrect content throughout the app.

4.2.3 Collaboration Spaces

2.2.3.1. A user shall be able to create spaces by providing title, description and tags.

2.2.3.2. A user shall be able to join spaces as collaborator.

2.2.3.3. A user shall be able to participate in discussion in a joined space.

2.2.3.4. A user shall be able to vote up or down on discussions in a joined space.

2.2.3.5. A user shall be able to reply to discussions in a joined space.

2.2.3.6. A space shall have a title, graphical visualization and discussion.

2.2.3.7. A space should have tags to make it easier for users to categorize, search and find spaces in an easier manner.

2.2.3.8. The system shall store discussions with version history.

2.2.3.9. Users shall be able to view the recent activity in a space. This activity shall list recently added content, recent discussions, recently joined collaborators by chronological order.

4.2.4 Graph Presentation

2.2.4.1. The system shall layout the graphical visualization of a space which shows relation between contents of the space by nodes and edges.

2.2.4.2. Users shall be able to add new nodes by connecting them to existing nodes in a space.

2.2.4.3. The system shall show bi-directional connection between two nodes as an edge.

2.2.4.4. The system shall show relation property of an edge on the graph representation.

2.2.4.5. A user shall be able to add, delete or modify nodes in a joined space.

2.2.4.6. A user shall be able to add or remove connection between nodes in a joined space.

4.2.5 Search

2.2.5.1. A user shall be able to search for spaces by title and tags.

2.2.5.2. A user shall be able to search for other users by username.

2.2.5.3. A user shall be able to navigate to a profile from search results.

2.2.5.4. A user shall be able to navigate to a space from search results.

4.2.6 Browsing

2.2.6.1. A user shall be able to browse new spaces.

2.2.6.2. A user shall be able to browse trending spaces.

4.2.7 User Profile

2.2.7.1. The system shall not expose users' real name or any sensitive information.

2.2.7.2. The system shall show users' information related to the application such as username, age and profession.

2.2.7.3. The system shall list users' spaces, collaborations.

2.2.7.4. A user shall be able to view other users' profile page.

2.2.7.5. A user shall be able to edit their profile information.

4.3 Non-functional Requirements

4.3.1 Performance

2.3.1.1. The system should have a responsive UI to support using the project on mobile devices.

2.3.1.2. The system shall have smooth and accessible UI/UX.

4.3.2 Reliability

2.3.2.1. The system shall ensure 99.9% uptime.

4.3.3 Security

2.3.3.1. The collected sensitive data shall be encrypted and then saved in database.

2.3.3.2. The system shall use HTTPS as communication protocol.

4.3.4 Accessibility

2.3.4.1. The system shall have English as main language.

2.3.4.2. The system should have localization for different languages.

4.4 System Architecture

4.4.1 Architecture

2.4.1.1. The system shall be available as a web application.

2.4.1.2. The system shall have backend with Django Python or Spring Boot Java.

2.4.1.3. The system shall use a relational database such as MySQL to store data such as user, space and discussions.

4.4.2 Integration

2.4.2.1. The system shall utilize Wikidata as data source.

Chapter 5

Design Documents

5.1 Architecture Overview

5.1.1 Overview

Connect-The-Dots implements a modern three-tier architecture with clear separation of concerns:

- **Presentation Layer:** React-based Single Page Application (SPA)
- **Application Layer:** Django REST Framework API
- **Data Layer:** PostgreSQL relational database

5.1.2 Component Architecture

Frontend (Presentation Layer)

- **Technology:** ReactJS 18+, React Router, CSS3
- **State Management:** React Context API and local component state
- **Key Components:**
  - User authentication and session management
  - Space display and management UI
  - Graph visualization using React Flow
  - Node and edge detail modals for data manipulation
  - Semantic search with also tag-based filtering capability
- **API Communication:** Fetch API for REST interactions with backend

Backend (Application Layer)

- **Framework:** Django 5.1 with Django REST Framework 3.15
- **Authentication:** JWT-based authentication using SimpleJWT
- **API Design:** RESTful API with ViewSets and serializers

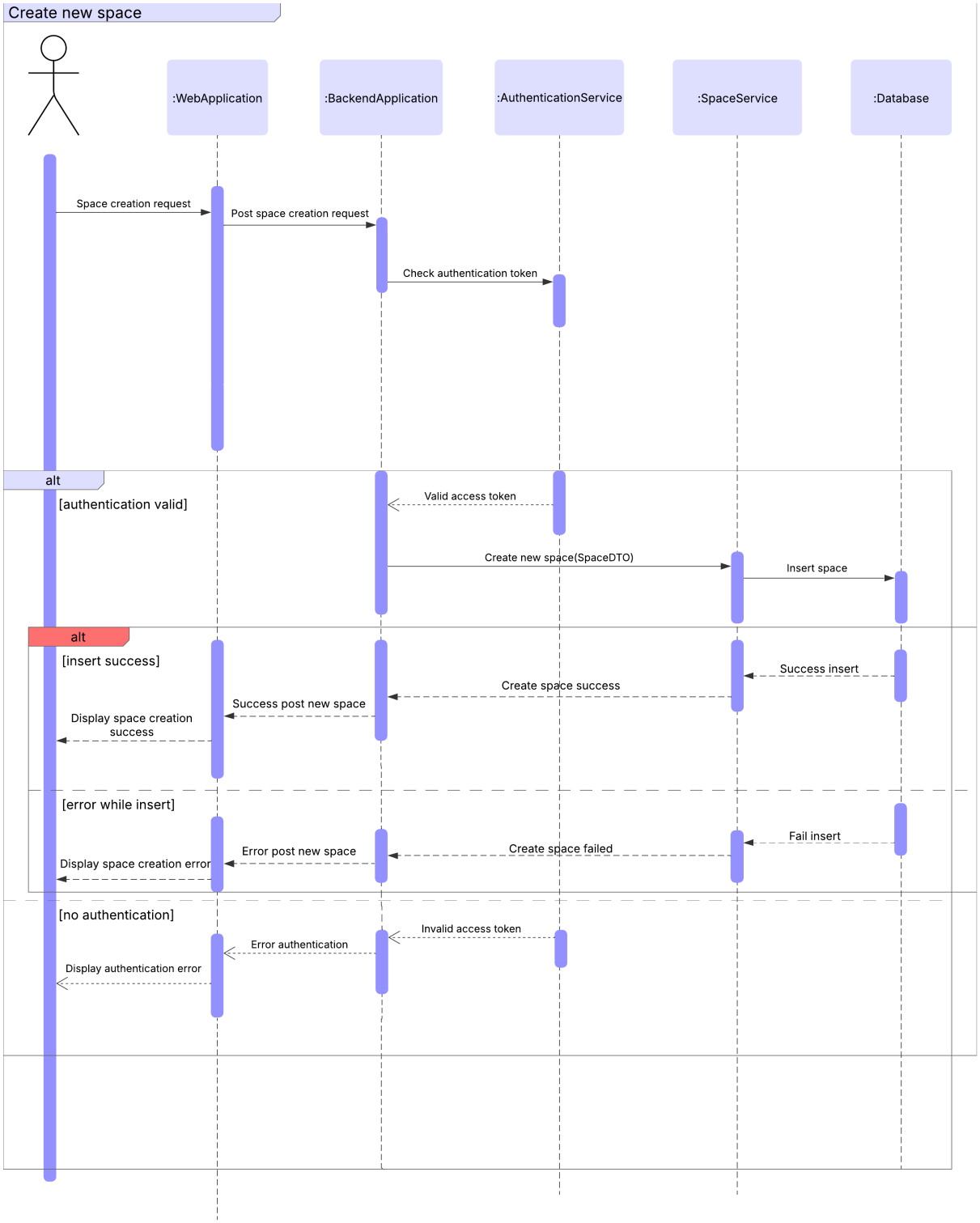
- **Key Modules:**
  - User management and authentication services
  - Space and collaboration management
  - Graph data processing and storage
  - Wikidata integration for node and tag enhancement
  - Space and user search capabilities
- **External Integrations:** Wikidata API for entity data

Database (Data Layer)

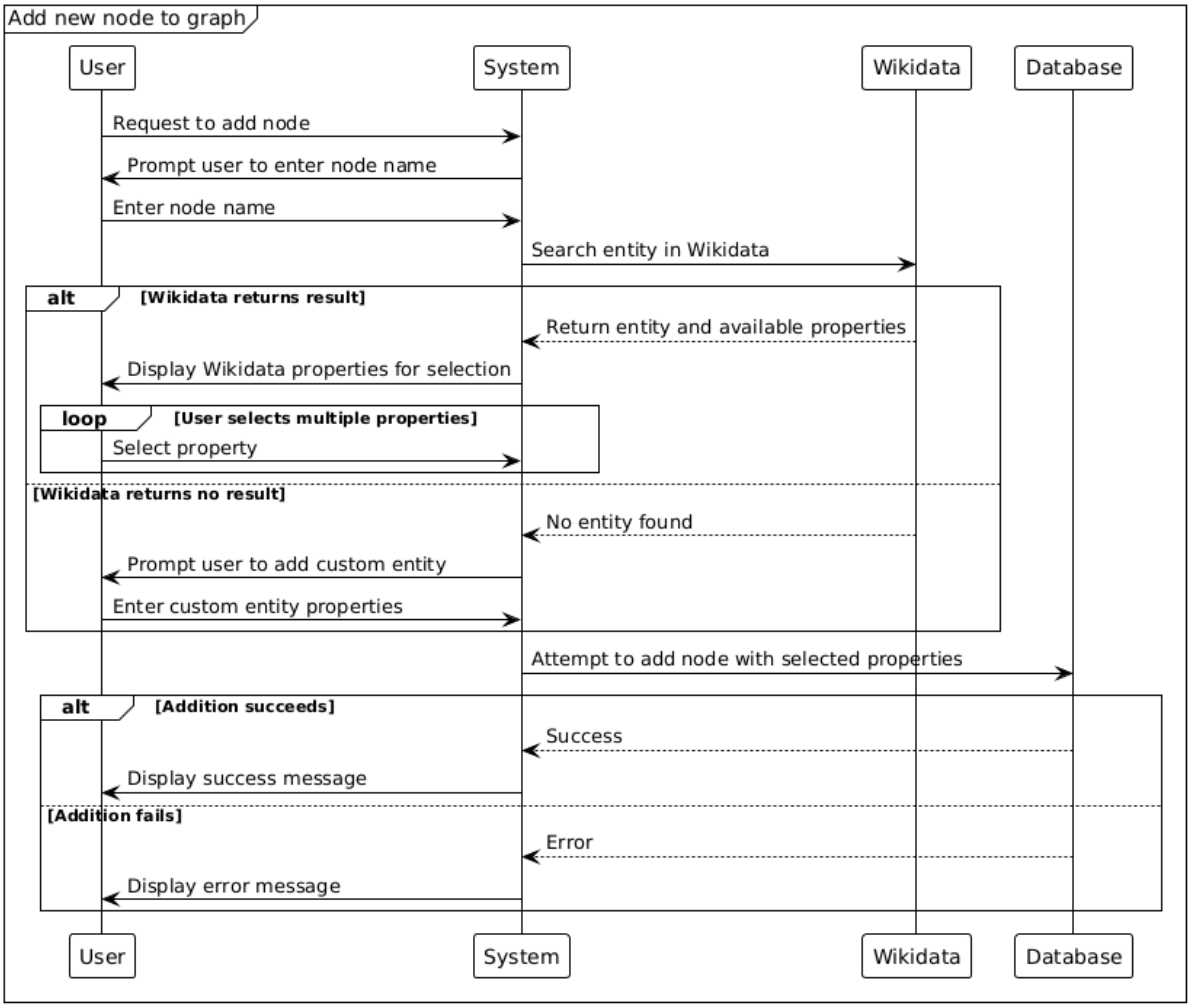
- **DBMS:** PostgreSQL 16
- **Schema Design:** Relational model with the following key entities:
  - Users and Profiles
  - Spaces and Tags
  - Nodes and Edges (graph components)
  - Properties (metadata for nodes)
- **Data Access:** Django ORM with optimized queries

5.2 Sequence Diagrams

5.2.1 Create New Space



5.2.2 Add new node to graph



5.2.3 ERD

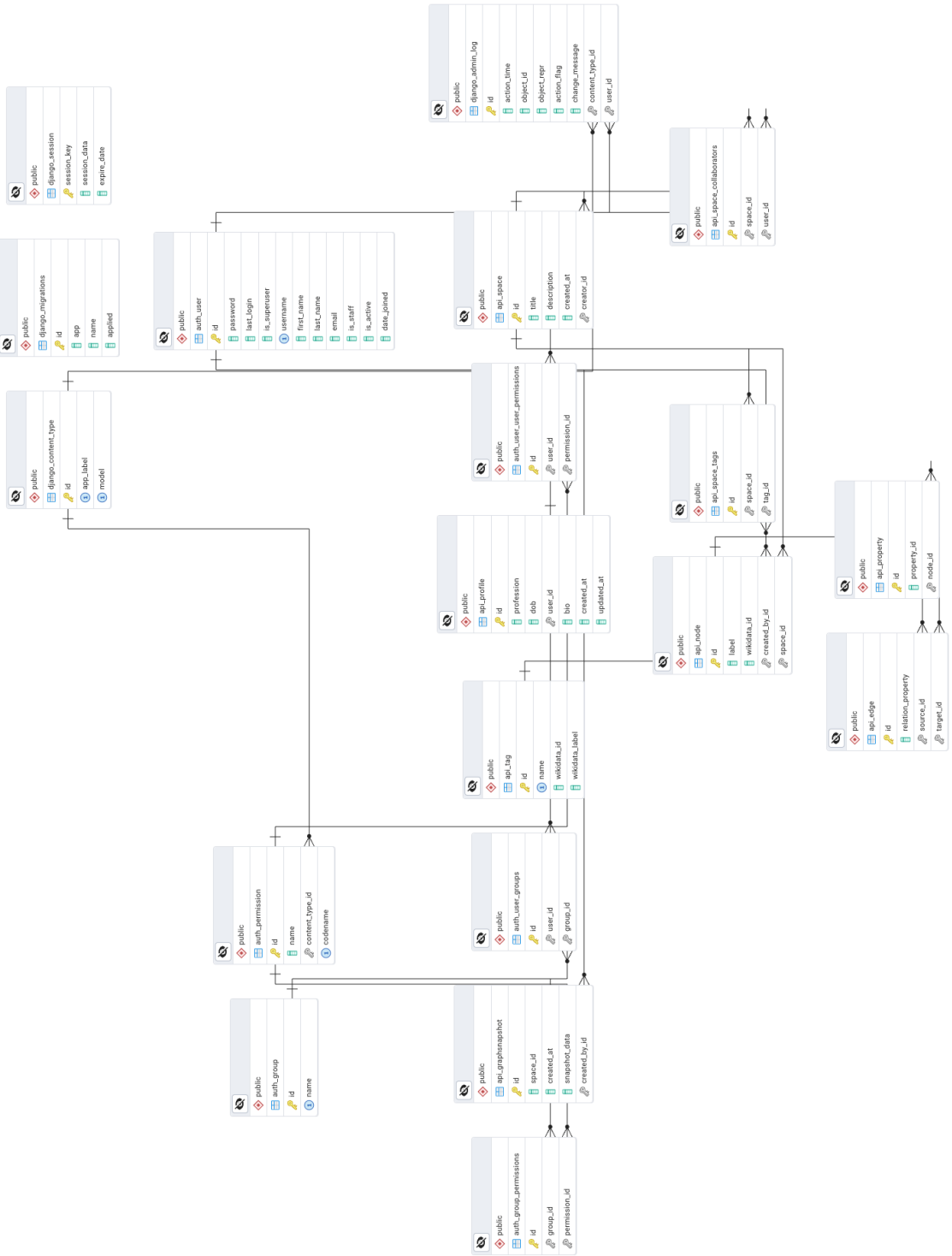


Figure 5.1: Entity Relationship Diagram

5.3 API Design

5.3.1 Authentication Endpoints

- **POST /api/register/**
  - *Description:* Register a new user account
  - *Permissions:* Public access
- **POST /api/login/**
  - *Description:* Authenticate user and obtain JWT tokens
  - *Permissions:* Public access

5.3.2 Space Management

- **GET /api/spaces/**
  - *Description:* List all spaces (with optional filtering)
  - *Permissions:* Authenticated users
- **GET /api/spaces/new**
  - *Description:* List spaces created in last 7 days ordered by date of creation
  - *Permissions:* Authenticated users
- **GET /api/spaces/trending**
  - *Description:* List spaces sorted by highest number of contributors
  - *Permissions:* Authenticated users
- **POST /api/spaces/**
  - *Description:* Create a new space
  - *Permissions:* Authenticated users
- **GET /api/spaces/{id}/**
  - *Description:* Retrieve detailed information about a specific space
  - *Path Parameters:* id (integer)
  - *Permissions:* Authenticated users
- **PUT /api/spaces/{id}/**
  - *Description:* Update space details
  - *Path Parameters:* id (integer)
  - *Permissions:* Space creator only
- **DELETE /api/spaces/{id}/**
  - *Description:* Delete a space

  - *Permissions:* Space creator only

- **POST /api/spaces/{id}/join/**
  - *Description:* Join a space as collaborator
  - *Path Parameters:* id (integer)
  - *Permissions:* Authenticated users

- **POST /api/spaces/{id}/leave/**
  - *Description:* Leave a space as collaborator
  - *Path Parameters:* id (integer)
  - *Permissions:* Space collaborators

5.3.3 Graph Data Management

- **GET /api/spaces/{id}/graph/**
  - *Description:* Retrieve the complete graph structure for a space
  - *Path Parameters:* id (integer)
  - *Permissions:* Authenticated users

- **POST /api/spaces/{id}/nodes/**
  - *Description:* Add a new node to a space
  - *Path Parameters:* id (integer)
  - *Permissions:* Space creator and collaborators

- **POST /api/spaces/{id}/edges/**
  - *Description:* Create a relationship between two nodes
  - *Path Parameters:* id (integer)
  - *Permissions:* Space creator and collaborators

- **DELETE /api/nodes/{id}/**
  - *Description:* Remove a node from a space
  - *Path Parameters:* id (integer)
  - *Permissions:* Space creator and collaborators

- **DELETE /api/edges/{id}/**
  - *Description:* Remove a relationship between nodes
  - *Path Parameters:* id (integer)
  - *Permissions:* Space creator and collaborators

5.3.4 Tag Management

- **GET /api/tags/**
  - *Description:* List all available tags
  - *Permissions:* Authenticated users
- **POST /api/tags/**
  - *Description:* Create a new tag
  - *Permissions:* Authenticated users

5.3.5 User Profile Management

- **GET /api/profiles/**
  - *Description:* List all user profiles (admin only)
  - *Permissions:* Admin users
- **GET /api/profiles/me/**
  - *Description:* Retrieve current user's profile
  - *Permissions:* Authenticated users (own profile)
- **PUT /api/profiles/me/**
  - *Description:* Update current user's profile
  - *Permissions:* Authenticated users (own profile)

5.3.6 Search

- **GET /api/search/**
  - *Description:* Global search across spaces, tags and users
  - *Query Parameters:* q (search query)
  - *Permissions:* Authenticated users
- **GET /api/search/wikidata/**
  - *Description:* Search Wikidata for entities
  - *Query Parameters:* q (search query)
  - *Permissions:* Authenticated users
- **GET /api/search/wikidata/{entity_id}/properties/**
  - *Description:* Retrieve available properties for a Wikidata entity
  - *Path Parameters:* entity_id (string)
  - *Permissions:* Authenticated users

Chapter 6

Project Status

6.1 Requirements Status

6.1.1 Registration and Login Requirements Status

Requirement	Status	Notes
1.1.1. Users shall be able to register to the system with an unique e-mail, unique username, password, age and profession data.	Completed	-
1.1.2. Users shall be over 18 years old.	Completed	-
1.1.3. Users shall be able to login using their e-mail and password.	Completed	Login is with username and password instead.

6.1.2 Authorization Requirements Status

Requirement	Status	Notes
1.2.1. There shall be two types of users in the system; admins and regular users.	Completed	-
1.2.2. An admin shall be able to do everything a regular user can.	Completed	-
1.2.3. An admin shall be able to delete any irrelevant, inappropriate or incorrect content throughout the app.	Completed	-

6.1.3 Collaboration Spaces Requirements Status

Requirement	Status	Notes
1.3.1. A user shall be able to create spaces by providing title, description and tags.	Completed	-
1.3.2. A user shall be able to join spaces as collaborator.	Completed	-
1.3.3. A user shall be able to participate in discussion in a joined space.	Completed	-
1.3.4. A user shall be able to vote up or down on discussions in a joined space.	Not Completed	-
1.3.5. A user shall be able to reply to discussions in a joined space.	Completed	-
1.3.6. A space shall have a title, graphical visualization and discussion.	Completed	-
1.3.7. A space should have tags to make it easier for users to categorize, search and find spaces in an easier manner.	Completed	-
1.3.8. The system shall store discussions with version history.	Completed	-
1.3.9. Users shall be able to view the recent activity in a space. This activity shall list recently added content, recent discussions, recently joined collaborators by chronological order.	Not Completed	-

6.1.4 Graph Presentation Requirements Status

Requirement	Status	Notes
1.4.1. The system shall layout the graphical visualization of a space which shows relation between contents of the space by nodes and edges.	Completed	-
1.4.2. Users shall be able to add new nodes by connecting them to existing nodes in a space.	Completed	-
1.4.3. The system shall show bi-directional connection between two nodes as an edge.	Completed	-
1.4.4. The system shall show relation property of an edge on the graph representation.	Completed	-
1.4.5. A user shall be able to add, delete or modify nodes in a joined space.	Completed	-
1.4.6. A user shall be able to add or remove connection between nodes in a joined space.	Completed	-

6.1.5 Search Requirements Status

Requirement	Status	Notes
1.5.1. A user shall be able to search for spaces by title and tags.	Completed	-
1.5.2. A user shall be able to search for other users by username.	Completed	-
1.5.3. A user shall be able to navigate to a profile from search results.	Completed	-
1.5.4. A user shall be able to navigate to a space from search results.	Completed	-

6.1.6 Browsing Requirements Status

Requirement	Status	Notes
1.6.1. A user shall be able to browse new spaces.	Completed	-
1.6.2. A user shall be able to browse trending spaces.	Completed	-

6.1.7 User Profile Requirements Status

Requirement	Status	Notes
1.7.1. The system shall not expose users' real name or any sensitive information.	Completed	-
1.7.2. The system shall show users' information related to the application such as username, age and profession.	Completed	-
1.7.3. The system shall list users' spaces, collaborations.	Completed	-
1.7.4. A user shall be able to view other users' profile page.	Completed	-
1.7.5. A user shall be able to edit their profile information.	Not Completed	-

6.1.8 Performance Requirements Status

Requirement	Status	Notes
2.1.1. The system should have a responsive UI to support using the project on mobile devices.	Not Completed	-
2.1.2. The system shall have smooth and accessible UI/UX.	Completed	-

6.1.9 Reliability Requirements Status

Requirement	Status	Notes
2.2.1. The system shall ensure 99.9% uptime.	Completed	-

6.1.10 Security Requirements Status

Requirement	Status	Notes
2.3.1. The collected sensitive data shall be encrypted and then saved in database.	Completed	-
2.3.2. The system shall use HTTPS as communication protocol.	Not Complete	It couldn't be completed since it required me to have domain name.

6.1.11 Accessibility Requirements Status

Requirement	Status	Notes
2.4.1. The system shall have English as main language.	Completed	-
2.4.2. The system should have localization for different languages.	Not Completed	-

6.1.12 Architecture Requirements Status

Requirement	Status	Notes
3.1.1. The system shall be available as a web application.	Completed	-
3.1.2. The system shall have backend with Django Python or Spring Boot Java.	Completed	Django Python have been used as backend.
3.1.3. The system shall use a relational database such as MySQL to store data such as user, space and discussions.	Completed	PostGRE SQL have been used.

6.1.13 Integration Requirements Status

Requirement	Status	Notes
3.2.1. The system shall utilize Wikidata as data source.	Completed	-

6.2 Deployment Status

- **Deployment URL:** http://54.163.77.79:3000/
- **Dockerized:** Yes

Chapter 7

Installation Instructions

7.1 Prerequisites

- **Docker and Docker Compose**
- **Git**

7.1.1 Local Development Setup

1. Clone the repository:

```
git clone https://github.com/yusufbayam/SWE573
cd SWE573
```

2. Create environment files:
   - Create `backend/.env.docker` for development
   - Example content:

```
DJANGO_SECRET_KEY=dev-key-for-local-testing
DJANGO_DEBUG=True

POSTGRES_USER=myuser
POSTGRES_PASSWORD=pass
POSTGRES_DB=mydb
POSTGRES_HOST=db
POSTGRES_PORT=5432
```

3. Build and start the containers:

```
docker-compose -f docker-compose.yaml up --build -d
```

4. Access the application:
   - Frontend: http://localhost:3000
   - Backend API: http://localhost:8000/api
   - Backend Admin: http://localhost:8000/admin

7.1.2 Production Deployment

1. Set up a server with Docker and Docker Compose
2. Clone the repository and cd into root folder of repository:

```
git clone https://github.com/yusufbayam/SWE573
cd SWE573
```

3. Create `backend/.env.docker.prod` with production settings:

```
DJANGO_SECRET_KEY=your-very-secret-production-key
DJANGO_DEBUG=False

POSTGRES_USER=myuser
POSTGRES_PASSWORD=pass
POSTGRES_DB=mydb
POSTGRES_HOST=db
POSTGRES_PORT=5432
```

4. Run the deployment script:

```
chmod +x deploy.sh
./deploy.sh
```

7.1.3 Manual Database Migration

Migrations are already included in the deployment script but if needed, you can manually run database migrations:

```
docker exec swe573-api-1 python3 manage.py makemigrations
docker exec swe573-api-1 python3 manage.py migrate
```

7.2 User Manual

7.2.1 Getting Started

Creating an Account

1. Navigate to the Connect-The-Dots homepage.
2. Click the "Register" button.
3. Fill in the required fields:
    - Username (must be unique)
    - Email address (must be unique)

- Password
- Profession
- Date of birth
- Password
- Confirm password

4. Click "Register" to complete the registration.
5. You will see registration is successful text.

Logging In

1. Navigate to the Connect-The-Dots homepage.
2. Click the "Login" button.
3. Enter your username and password.
4. Click "Login" button to access your account.

Viewing Your Profile

- Profession: Professional title or area of expertise.
- Bio: Brief description about user.
- Date of Birth: In DD.MM.YYYY format.
- Joined date: In DD.MM.YYYY format.

7.2.2 Exploring the Platform

Browsing Spaces

1. From the dashboard, you can see:
    - Popular spaces across the platform
    - New spaces created in last 7 days across the platform
2. Use the search bar at the top to find spaces and users by keywords.
3. Click on any space card to view its details and knowledge graph.
4. Click on any user card to view its profile.

Navigating the Dashboard

- **Discover**: Overview of recently created and featured spaces
- **Profile**: Navigates to the current user's profile
- **Search**: Search among spaces and user's with an input text
- **Create Space**: Navigate to a page where users can create a new space

Accessing the Search Function

1. The search bar is accessible from the header on any page within the application
2. Enter your search terms in the input field
3. Click the "Search" button or press Enter to execute your search
4. You will be redirected to the search results page

Search Results Interface

1. The search results page displays two main tabs:
    - **Spaces**: Shows spaces matching your search query
    - **Users**: Shows user profiles matching your search query
2. Each tab displays the number of results found in parentheses
3. Click on either tab to switch between viewing space results and user results
4. Results are displayed in card format for easy scanning

What Can Be Searched

The search function performs a comprehensive search across multiple data types:

- **Space Titles**: Finds spaces with matching words in their titles
- **Space Descriptions**: Searches the content of space descriptions
- **Tags**: Identifies spaces tagged with keywords matching your search
- **Usernames**: Locates users whose usernames match your search terms

7.2.3 Creating and Managing Spaces

Creating a New Space

1. From the dashboard, click the "Create Space" button.
2. Enter the following information:
    - Title: A clear, descriptive name for your space
    - Description: An overview of the space's purpose and content
    - Tags: Search from Wikidata and select one of the resuil from dropdown list. After selecting from Wikidata it is possible to change the display label of tag. It will still use the Wikidata id of selected item, display label is only for displaying purposes.
3. Click "Create Space" to set up your new space.
4. You will be redirected to the space's graph view, ready for you to start adding content.

Space Management Options

- **Delete Space**: Permanently remove the space (admin or collaborators only)

7.2.4 Building Knowledge Graphs

Understanding the Graph Interface

- **Graph Canvas**: The main area displaying nodes and connections
- **Controls**: Navigation tools for panning and zooming
- **Nodes List**: Shows all nodes that have been added to the space

Adding Nodes from Wikidata

1. In the Space Detail view, find the "Add Node from Wikidata" section (only visible if you're a collaborator)
2. Enter a search term in the search field and click "Search"
3. From the dropdown that appears, select an entity of interest
4. Once selected, you'll see the entity's available properties
5. Click on individual properties to select/deselect them for inclusion
6. If you want to connect this node to an existing node:
    - Choose the direction of the relationship using the "Edge Direction" button
    - Select an existing node from the "Connect To Node" dropdown
    - Enter a descriptive label for the relationship in the "Edge Label" field
7. Click "Add Node with Edge" to create the node (and optional connection)
8. The graph will refresh to show your new addition

Editing Node Details

1. Click on any node in the graph visualization to open its detail panel
2. The Node Detail modal shows:
    - Basic node information (label, Wikidata ID if applicable)
    - Current properties with their values
    - Available properties that can be added
3. Click on properties in the list to select or deselect them
4. Click "Save Changes" to update the node with your selected properties
5. To delete a node, scroll to the "Danger Zone" section and click "Delete Node"
    - You'll need to confirm deletion by clicking a second time
    - Note that deleting a node will also remove all its connections

Managing Edges (Relationships)

1. Click on any edge in the graph to open its detail panel
2. The Edge Detail modal shows:
    - The connected nodes (source and target)
    - The current relationship label
    - Direction controls
3. Edit the edge label by typing in the text field
4. Change the direction by clicking the direction toggle button
5. Click "Update Label & Direction" to save your changes
6. To delete an edge, use the "Delete Edge" button in the Danger Zone
    - You'll need to confirm deletion by clicking a second time

Display the Graph

- **Auto-Arrange**: Node positions will be auto arranged by their relations to other nodes automatically whenever spaces are modified(Node add, edit or delete).
- **Pan View**: Click and drag on empty canvas space to move the entire view
- **Zoom**: Use the scroll wheel or zoom controls to adjust detail level

7.2.5 Collaboration Features

Joining as Collaborator

1. A non-collaborator user can join to a space as collaborator via home page or space details.
2. There is a join button at the Space card listing for joinable spaces on the home page.
3. There is a join button on the Space details page for the joinable ones.

Collaboration Permissions

- **Creator**: Full control over the space, including deletion.
- **Collaborators**: Can add/edit nodes and relationships.
- **Viewers**: Can view but not modify the space.

Real-time Collaboration

- Changes made by other collaborators are directly visible after refreshing the space details page.
- The list of active collaborators appears in the top-right corner.

Participating in Discussions

1. Each space includes a discussion section where collaborators can share ideas and feedback.
2. To view discussions:
    - Navigate to a space's details page
    - Locate the "Discussions" panel below list of collaborators
    - All users (even non-collaborators) can view existing discussions
3. To add a new comment to the discussion:
    - You must be a collaborator of the space
    - Locate the comment form at the top of the discussions panel
    - Type your message in the text area
    - Click "Post Comment" to submit your contribution
    - Your comment will immediately appear in the discussion thread
4. Discussion features:
    - Comments are displayed in chronological order with the most recent at the top
    - Each comment shows the author's username and timestamp
    - Your own comments are highlighted with a different background color
    - The discussion thread automatically updates when new comments are added
5. If you are not a collaborator:
    - You will see a message indicating "Join as a collaborator to participate in discussions"
    - You can still read all existing discussions, but cannot post new comments
    - Use the "JOIN" button to become a collaborator and gain commenting privileges

Chapter 8

Testing

8.1 Unit Testing

Connect-The-Dots implements a comprehensive unit testing framework using industry-standard testing tools for both frontend and backend components. The test suite is designed to validate core functionality, ensure API reliability and maintain code quality throughout development.

8.1.1 Backend Testing

The backend testing suite utilizes Django's test framework alongside Django REST Framework's APITestCase class. Test coverage focuses on critical system components:

- **Authentication System**: Tests verify user registration validation rules, login functionality and JWT token generation. Test cases include valid credentials, invalid credentials and edge cases such as duplicate usernames.
- **Search Functionality**: Comprehensive tests validate search capabilities across multiple data types including:
  - Searching spaces by title
  - Searching spaces by description content
  - Searching spaces by associated tags
  - Searching for users by username
  - Handling edge cases (empty queries, no results)
  - Access control (authenticated vs. unauthenticated requests)
- **Space Collaboration**: Tests verify the space collaboration system, including:
  - Joining and leaving spaces
  - Collaborator permissions
  - Access control for space operations
- **Profile Management**: Tests for user profile creation, update operations and validation rules including:

  - Profile creation on user registration
  - Profession field validation
  - Date of birth validation (age restrictions)

- **Space and Tag Management**: Basic CRUD operations for spaces and tags are tested to ensure data integrity and proper relationships.

8.1.2 Frontend Testing

The frontend test suite utilizes React Testing Library and Vitest to test component rendering, user interactions and integration with the backend API. Key areas of testing include:

- **Authentication Components**: Tests for Login and Registration components verify form submissions, validation messages and successful authentication flows.
- **Space Details View**: Extensive testing of the Space Details component includes:
  - Proper rendering of space information
  - Collaborator-specific UI elements
  - Graph visualization components
  - Node and edge interaction
- **Home Page**: Tests verify proper display of trending and new spaces, search functionality and navigation.
- **Search Functionality**: Dedicated tests for the search component validate:
  - Search form submission
  - Results display
  - Tab switching between space and user results
  - Navigation to search results
- **Profile Page**: Tests verify profile information display and associated spaces.
- **Utility Functions**: Date formatting and other utility functions have dedicated unit tests.

8.1.3 Test Environment

The testing environment includes:

- **Mocking**: External dependencies (API calls, local storage, browser APIs) are mocked to isolate component behavior.
- **Test Data**: Comprehensive test fixtures create realistic test scenarios.
- **Setup and Teardown**: Proper test isolation through setup and teardown procedures.

8.1.4 Test Execution Results

All unit tests pass successfully, with the testing suite achieving good coverage of critical system functionality. The continuous integration process runs these tests automatically to prevent regressions during development.

8.2 User Testing

8.2.1 Testing Methodology

User testing for Connect-The-Dots followed a systematic approach to evaluate the platform's core functionalities, focusing on real-world usage scenarios across all major features.

8.2.2 Scope of Testing

The testing covered all primary features of the platform:

- User authentication (registration and login)
- Profile management
- Space creation and management
- Node & edge building and manipulation among them
- Collaboration features
- Search functionality
- Wikidata integration

8.2.3 Results and Findings

The results showed that the core functionalities of the platform perform reliably and meet the expected requirements of the project. Specifically:

- **Authentication System**: Users were able to register successfully and log in without errors. The JWT-based authentication system maintained sessions appropriately and handled invalid credentials securely.
- **Profile Management**: Users successfully managed their profiles, including editing profile details such as profession and bio information. Navigation to personalized dashboards and viewing created/joined spaces worked as expected.
- **Space Creation**: The functionality for creating and updating spaces worked well, with users able to set up new knowledge spaces, add descriptive information and assign relevant tags without difficulties.
- **Knowledge Graph Building**: Users could effectively add nodes from Wikidata, select relevant properties and establish connections between nodes. The visualization rendered properly across different browsers and screen sizes.

- **Collaboration**: The joining/leaving spaces functionality operated correctly, with proper permission management ensuring only authorized users could modify space content. Collaborator lists updated appropriately when users joined or left spaces.
- **Graph Interaction**: Users could interact with the graph visualization, including selecting nodes, viewing details and managing relationships between concepts.

8.2.4 Areas for Improvement

Despite overall positive results, several areas requiring improvement were identified:

- **Error Handling**: The platform lacks comprehensive response mechanisms for page errors such as 404 (not found) or 500 (server error). Users encountered generic error messages that did not provide clear guidance on how to resolve issues.
- **Wikidata Property Selection**: The interface for selecting properties from Wikidata entities proved confusing for some users. Inadequate explanations of what each property represents.

8.2.5 Conclusion

The user testing demonstrated that Connect-The-Dots successfully delivers on its primary objective of enabling collaborative knowledge space creation and visualization. While improvements are needed in property adding and error handling, the platform's core features perform reliably and provide a solid foundation for collaborative knowledge building. The identified issues have been documented.