SWE 574

Software Development as a Team

Suzan Üsküdarlı

CONNECT THE DOTS

Customer Milestone 1

Report

v1.0.0

Batuhan Cömert
Gamze Güneri
Esra Nur Özüm
Yasemin Tangül
Yusuf Bayam
—Group 1—

Table Of Contents

Purpose of the Report

The purpose of this report is to provide an overview of the current implementation status of the project's functional&non-functional requirements across all major components like backend, web frontend, and mobile application. It aims to track progress made, highlight completed and ongoing features, and identify remaining tasks that need to be addressed before the next release. By presenting a clear picture of development progress, this report ensures alignment among team members and supports effective planning for future iterations.

Project Overview

Connect-The-Dots is a collaborative knowledge-mapping platform designed to help users build meaningful connections between ideas, people, and discussions. By combining an interactive graph-based interface with activity streams and space-based collaboration, the platform enables communities to co-create and explore relationships among concepts in a visually intuitive way. The system includes a Django-based backend, a React web frontend, and a native Android application, all communicating through a shared REST API. Together, these components provide a seamless multi-platform experience for exploring, sharing, and expanding knowledge networks.

Milestone Objectives

The primary objective of the CM1 milestone is to establish a solid foundation for the Connect-The-Dots project by completing the initial setup, research, and planning phases required for subsequent development. Specifically, this milestone aims to:

- **Establish the Project Infrastructure**: Set up the GitHub monorepo, define branching strategies, implement CI/CD pipelines, and prepare the Dockerized development and deployment environments.
- **Conduct Foundational Research and Analysis**: Review related projects, analyze initial functional requirements, and study relevant W3C specifications (such as Activity Streams and WAI Accessibility) to ensure compliance with web standards and inclusive design principles.
- **Design and Planning:** Create system architecture, user experience designs, and project plans outlining timelines, deliverables, and resource allocation for upcoming milestones.
- **Define Testing and Deployment Strategies:** Implement initial unit testing for core functionalities and integrate these into the CI pipeline to support future continuous integration and delivery workflows.

- **Deliver an Initial Deployed Version**: Deploy an initial working version of the application stack (backend, frontend, and mobile integration) to demonstrate the successful setup of the end-to-end environment.

Together, these objectives ensure that the technical, organizational, and design foundations of the project are fully established before transitioning to the implementation and feature development stages.

RAM

Responsibility	**Batuhan**	**Esra**	**Gamze**	**Yasemin**	**Yusuf**
Communicator				X	
Wiki Editor	X	X	X	X	X
Mockups	X	X	X		
Diagrams		X	X		X
Scenarios	X			X	
Presenter	X		X	X	X
Infrastructure				X	
Frontend	X	X	X		X
Backend	X				X
Dockerization				X	X
Deployment				X	
Mobile		X			X
Testing	X	X	X	X	X

Actions Completed

Activity	**Description**	**Status**

Requirements Gathering	Conducted user and stakeholder analysis; identified functional and non-functional requirements.	Completed
Software Requirements Specification (SRS)	Documented user stories, use cases, and detailed requirements in the SRS document.	Completed
System Design	Designed UML diagrams (use case, class, and sequence diagrams) and outlined overall architecture.	Completed
Repository Setup	Initialized GitHub repository with branch management strategy (main/dev/feature).	Completed
Setup the Infrastructure	Prepare the project's development and deployment environment. This may include container setup (e.g., Docker), CI/CD integration, development pipelines, and test environment configuration.	Completed
Updated Requirements	Reflect any feedback, new findings, or design changes in the updated requirement documentation. Maintain traceability between changes and related design or implementation updates.	Completed
Unit Testing	Implement automated tests for core functionalities to verify correctness and reliability. Ensure coverage targets are met and integrate tests into the CI pipeline.	Completed
CI/CD	Define and implement the continuous integration and delivery pipeline. Versioning: Establish semantic versioning or tag conventions. Branch Policy: Define main/develop/feature branch strategies, naming conventions, and protection rules	Completed
Plan	Prepare the overall project plan detailing milestones, timeline, resources, and deliverables. Outline the development roadmap, review checkpoints, and evaluation criteria for each phase.	Completed

Implemented Features

During this milestone, several key components of the system were established and deployed.

- The CI/CD pipeline was configured using GitHub Actions, and the deployment environments were successfully set up.
- An initial version of the Back-Office Dashboard was developed, introducing the Moderator role with basic permissions and workflow logic.
- On the mobile application, user registration and login functionalities were implemented and fully connected to the backend API.
- Additionally, a Color-Blind Mode setting was introduced in the mobile app to improve accessibility and align with WAI guidelines.

Technologies and Tools Used

Component	Technology / Tool
Frontend (Web)	React.js
Mobile Application	Kotlin
Backend	Django
Database	PostgreSQL
Version Control	GitHub
Development Tools	Visual Studio Code, Postman, Android Studio
Project Management	GitHub Projects
Documentation	Notion, figma, Github wiki, Canva
CI/CD	Github Actions
Deployment	AWS
Design	Canva, Figma

Repository and Development Practices

The project is organized in a monorepo structure hosted on GitHub, containing the backend (Django), web frontend (React), and mobile application (Kotlin). This approach enables centralized version control, consistent dependency management, and simplified CI/CD workflows across all components.

We follow a Git-flow-inspired branching strategy, where the main branch represents production-ready code, develop is used for integration testing, and feature branches are created for specific enhancements or fixes. All pull requests are reviewed before merging to ensure code quality and maintain consistent development standards.

Automated CI/CD pipelines are configured using GitHub Actions to run unit tests, and deploy updated services upon successful merges. Each component (backend, frontend, and mobile) has a dedicated workflow to ensure modular testing and build automation.

To ensure maintainability, the team adheres to consistent coding conventions, comprehensive documentation practices (GitHub Wiki, Notion, and inline code comments), and regular code reviews. This workflow fosters collaboration, traceability, and alignment across the development lifecycle.

Plans for next steps / for CM2

For the CM2 milestone, the team will focus on enhancing accessibility, interactivity, and analytical capabilities across the Connect-The-Dots platform.

The web application will be extended with multi-language support, laying the groundwork for future internationalization.

Accessibility will be improved through text-to-speech and speech-to-text features, enabling users to create nodes and boards via voice commands.

The analytics dashboard will be expanded to include detailed space-level metrics such as user count, comments, votes, and node statistics.

The moderation system will be strengthened with new capabilities to archive nodes, spaces, and discussions, along with refined moderator permissions.

A similarity-based search engine will be introduced to help users find related spaces, nodes, and discussions more effectively.

Geolocation features will be integrated into user profiles, nodes, and spaces to enable location-based visualization and context.

An activity stream will be implemented to log and display major in-space events following W3C Activity Streams standards.

Lastly, voting functionality will be added to discussions, allowing users to upvote or downvote content and highlight valuable contributions.

Together, these features will advance the platform’s inclusivity, usability, and community engagement toward a more interactive and data-driven system.

Project Plan

Customer Milestone	**Internal Milestone**	**Goal**	**Tasks**
CM1 *20/10/2025*	IM1 *16/10/2025*	Establish foundational infrastructure, documentation, and initial Backoffice design.	- Repository Creation - CI/CD Test Environment - SRS Documentation Update - Decision of way of working - Android Unit Test for Login/Register - W3C Specifications Research - Plan - Backoffice design / initial stage of development - Color theme (including color-blind accessibility) - Profile page for Android - Unit Testing
CM2 *24/11/2025*	IM2 *20/10/2025 – 03/11/2025*	Implement key web features enhancing usability and discoverability.	- Language support foundation for web - Voting Ability (upvote/downvote discussions) - Similarity Search - Geolocation implementation (user, node, and space level)
CM2 *24/11/2025*	IM3 *03/11/2025 – 17/11/2025*	Introduce accessibility features and moderation tools to ensure inclusivity and governance.	- Text to Speech & Speech to Text - Moderator Role to manage the Space - Activity Stream implementation - Analytics for Space
CM3 *15/12/2025*	IM4 *24/11/2025 – 01/12/2025*	Enhance analytics, activity streams, and voice interaction features.	- Activity Stream improvement - TTS & STT contribution features (discussion, reporting, search, login/register) - Analytics for admin improvement

CM3 *15/12/2025*	IM5 01/12/2025 – *08/12/2025*	Finalize reporting and space management functionalities.	- Reporting Feature to report properties (Node, Space, Discussion, People) - Context Merging & Splitting

Summary Timeline

Phase	Duration	Key Deliverables
IM1 → CM1	10 – 20 Oct 2025	Infrastructure setup, Back-Office design
IM2 + IM3 → CM2	20 Oct – 24 Nov 2025	Language support, Voting, Analytics, Moderator tools
IM4 + IM5 → CM3	24 Nov – 15 Dec 2025	Reporting, Admin analytics, Final release

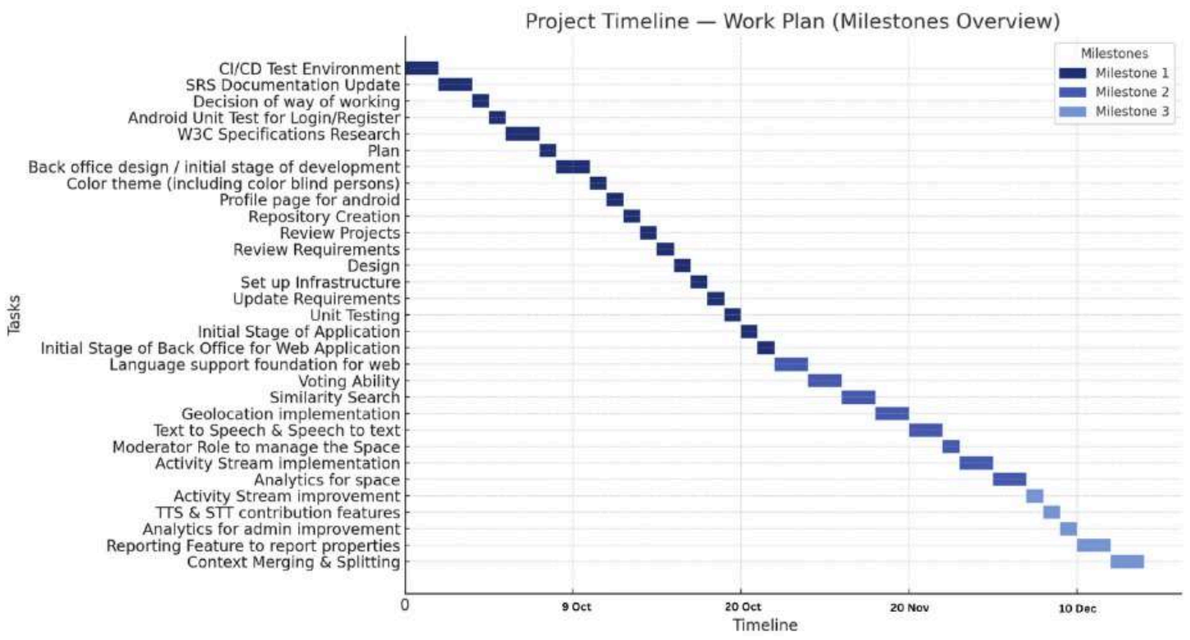


Fig 1: Project timeline overview for each customer milestone

W3C Research

W3C Compliant Color and Activity Stream Research Notes

Quotations

- WCAG 2.2 — Contrast (Minimum, 1.4.3, AA): The visual presentation of text and text images must have a contrast ratio of at least 4.5:1.
- WCAG 2.1 — Non-Text Contrast (1.4.11, AA): User interface components and expressive graphics, must have a contrast ratio of at least 3:1.
- WCAG — Use of Color (1.4.1, A): Information should not be conveyed only with color, an alternative should be found to the information given by color differences.
- WAI-ARIA — Feed Pattern: The element containing the set of articles carries role="feed", each unit of content is in role="article"
- ActivityStreams 2.0 — W3C: AS2 provides a JSON data model and dictionary to represent common online social objects, activities, and actors.

Color and Contrast (compliance with WCAG 2.2)

- Text contrast (1.4.3, AA): ≥ 4.5:1 for normal text, ≥ 3:1 for large text (≥ 18 pt normal, ≥ 14 pt bold). Logo/incidental texts are excluded. These ratios ensure readability in cases such as color blindness and age-related vision decline.
- Non-text contrast (1.4.11, AA): Component borders such as button borders, icon meaning, focus rings must be ≥ 3:1 against the background. Except in passive/disabled situations.
- Not providing information only with color (1.4.1, A): It is insufficient to indicate errors only in red and success only in green. Secondary clues such as text, icon, pattern, underline are mandatory.

Activity Stream — Data Model and Accessible Presentation

- Data model (AS2): Provides JSON schema that represents events with fields actor / verb (type) / object / target / published. This is a portable event logging standard across clients.
- Presentation pattern (ARIA APG): Give role="feed" to the feed container, role="article" to each card. Title, time and content must be in semantic hierarchy. Keyboard access and declared uploads should be provided in infinite scrolling.

Color and Contrast Requirements for Connect the Dots

- AA compliance required: Respect thresholds for text/icon/component borders. (1.4.3, 1.4.11)
- Color-independence at A level: Non-color cues are mandatory for information. (1.4.1)
- Typography contrast: Body text (normal) ≥ 4.5:1; page titles (large text) ≥ 3:1.
- Interactive components: Button border or padding relative to surroundings ≥ 3:1; focus ring ≥ 3:1 in all cases.
- Error: color + icon, textual description.
- Success/Warning: color + icon + text. (1.4.1)
- Blindness scenarios: Verify that node colors are distinguishable (as required by the design process).

Color Palette & Visual Elements

Component	Color	Hex	Contrast Ratio (vs white #FFFFFF)	WCAG 2.1 Reference
Background (main)	Light Gray / Off-white	#F5F5F5	—	1.4.3
Header / Navbar	Charcoal Gray	#2F2F2F	13.39 : 1	1.4.3
Primary Text (body)	Almost Black	#1B1F3B	16.08 : 1	1.4.3
Secondary Text / Labels	Slate Gray	#4A5568	7.53 : 1	1.4.3
Disabled Text	Medium Gray	#656F75	5.14 : 1	1.4.3, 1.4.6
Button Primary (Background)	Deep Blue	#0076B5	text white #FFFFFF = 4.93 : 1	1.4.3
Button Secondary (Background)	Teal Blue	#008296	text black #000000 = 4.53 : 1	1.4.3
Node (Default)	Indigo Blue	#3A0CA3	text white = 11.92 : 1	1.4.11

Node (Selected / Active)	Purple Accent	#7209B7	text white = 8.61 : 1	1.4.11
Edges / Lines	Dark Gray	#374151	10.31 : 1	1.4.11
Card / Panel Background	White	#FFFFFF	—	1.4.3
Input Field Border	Neutral Gray	#68686B	5.55 : 1	1.4.11
Placeholder Text	Gray	#646E7A	4.75 : 1	1.4.3
Alert / Error	Deep Orange / Red	#BD4902	text white = 4.68 : 1	1.4.11, 1.4.3
Success	Emerald Green	#2D6A4F	text white = 6.39 : 1	1.4.11, 1.4.3
Info / Hint	Sky Blue	#215D69	text white = 7.41 : 1	1.4.3
Warning	Amber / Yellow	#8F6701	text white = 4.69 : 1	1.4.3

Activity Stream Data Layer (AS2 compatible)

Example AS2 record for creating a node:

```
{
    "@context": "https://www.w3.org/ns/activitystreams",
    "type": "Create",
    "actor": {"type": "Person", "id": "user:123"},
    "object": {
        "type": "Note",
        "id": "node:987",
        "name": "New Node: 'LLM Sources",
        "content": "New sources added."
    },
    "target": {"type": "Collection", "id": "graph:42"},
    "published": "2025-10-23T14:00:00Z"
}
```

Activity Stream Presentation Layer

There should be a “Load More” button accessible via keyboard.

```
<section role="feed" aria-labelledby="streamTitle" aria-busy="false">
    <h2 id="streamTitle">Event Stream</h2>
    <article role="article" aria-posinset="1" aria-setsize="unknown">
        <h3><a href="/nodes/987">New Node: “LLM Sources”</a></h3>
        <p><time datetime="2025-10-23T14:00:00Z">October 23, 2025</time> —
    Batuhan created a node.</p>
        <button>Open Details</button>
    </article>
    <!-- load more →
    <button id="loadMore">Load More</button>
</section>
```

SRS Review

Software Requirements Review on Functional Requirements

1. **User Roles and Authorization**

   Old version: Only Admin and Regular User roles were available.

New version: A three-person hierarchy will be added: System Admin, Moderator, and Regular User.

- Moderators will be authorized only within their own domains, a transition to the RBAC (Role-Based Access Control) approach.
- Administrator actions will be performed with soft delete, preventing data loss and increasing auditability.

2. **Registration & Login**

   New additions:
   - Geolocation data becomes mandatory.
   - Structured data capture (e.g., latitude-longitude) → lays the foundation for future map-based analysis and location-based recommendation systems.

3. **Collaboration Spaces**

- The new version will introduce space merging. When two spaces are merged, discussions are merged chronologically, strengthening the data integrity and versioning mechanism.
- The concept of an "In-Space Activity Stream" has been defined (W3C Activity Streams 2.0 Specification.).
- The "Report" feature will allow users to report inappropriate content, supporting community governance principles.

4. **Graph Presentation**

- The content remains the same, but the terminology has been clarified (bi-directional edges, relation properties).
- Users will be able to perform similarity searches on nodes and edges.

5. **Search Functions**

- The single "Search" section in the old version of requirements has now been split into two:
  - - In-Space Search (local)
  - - In-App Search (global)
- Similarity search integration (will be possibly NLP or cosine similarity-based) → transition to a semantic search infrastructure.

6. **Analytics & Reputation**

   New additions:
   - Space-level and System-level Analytics Dashboards.
   - “Top Contributor” label based on user contribution.

   This demonstrates that the system will not only allow admins and moderators to interact but also the ability to measure and generate insights.

7. **Main Activity Stream**

- An "Activity Stream" will be added to the system at both a global and a space level (global and in-space feed).
- This feature aligns with the discovery-driven design approach, which increases user engagement.

Software Requirements Review on Non-Functional Requirements

1. **Performance**

- Added “Page load time < 3 seconds” criterion (W3C 2.1 guideline.).

2. **Security**

- Encryption and security mechanisms detailed:
  - bcrypt hashing
  - HTTPS mandatory
  - PII encryption at rest

3. **Accessibility**

- W3C-compliant Activity Stream (WCAG 2.1).
- Color-blind-friendly palette (1.4.3 Contrast).
- Speech-to-text and audible feedback support (W3C Web Accessibility Initiative).
- Font and size customization.

The project aims for level AA WCAG compliance.

4. Reliability

- 99.9% uptime target maintained.

5. System Architecture

Area	Old Version	New Version	Inference
Application Type	Web only	Web + Android	Mobile expansion and broader accessibility have been introduced.
Database	PostgreSQL	PostgreSQL	An ACID-compliant relational database has been adopted.
Deployment	Not specified	AWS EC2	A cloud-based CI/CD and mobile distribution pipeline has been established.
Environments	None	Separate Dev/Test and Production environments	Versioning and quality assurance processes have been formalized.

State of Functional Requirements

This document tracks the implementation status of the functional requirements for each component of the application: Backend, Web Frontend, and Mobile App.

Functional Requirement	Description	Backend	Web	Mobile
1.1. Registration And Login				
1.1.1	Register with a unique e-mail, username, and password.	Done ✅	Done ✅	Done ✅
1.1.2	Provide profession/work title and current location during registration.	Done ✅	Done ✅	Done ✅

Functional Requirement	**Description**	**Backend**	**Web**	**Mobile**
1.1.3	Capture location data in a structured GeoLocation format.	In Progress	In Progress	In Progress
1.1.4	Require users to confirm they are over 18 via a checkbox.	Done ✅	Done ✅	Done ✅
1.1.5	Log in using e-mail and password.	Done ✅	Done ✅	Done ✅
1.2. Authorization				
1.2.1	Three user roles: System Admin, Moderator, and Regular User.	Done ✅	N/A	N/A
1.2.2	Regular User can create, join, and contribute to spaces.	Done ✅	Done ✅	Done ✅
1.2.3	Moderator role is assigned on a per-space basis.	In Progress	In Progress	In Progress
1.2.4	Moderator can archive content within their assigned space.	In Progress	Not Done	Not Done
1.2.5	System Admin has universal privileges across the system.	In Progress	In Progress	N/A
1.2.6	The creator of a space is automatically assigned as Moderator.	In Progress	N/A	N/A
1.2.7	Moderators must provide a justification when archiving content.	Not Done	Not Done	Not Done
1.2.8	System Admin can "soft delete" content and users.	In Progress	In Progress	N/A

Functional Requirement	**Description**	**Backend**	**Web**	**Mobile**
1.2.9	System Admin can restore previously archived content.	In Progress	In Progress	N/A
1.2.10	System Admin can manage any user's role from the back-office.	In Progress	In Progress	N/A
1.2.11	Back-office dashboard for Admins to manage the system.	In Progress	In Progress	N/A
1.2.11	Admins view comprehensive analytics, moderators view space-specific data.	In Progress	In Progress	N/A
1.3. Collaboration Spaces				
1.3.1	Create spaces with a title, description, and tags.	Done ✅	Done ✅	Done ✅
1.3.2	Join existing spaces as a collaborator.	Done ✅	Done ✅	Done ✅
1.3.3	Propose merging their space with another.	Not Done	Not Done	Not Done
1.3.4	When merged, discussion threads are combined.	Not Done	N/A	N/A
1.3.5	Reply to discussions in a joined space.	Done ✅	Done ✅	Done ✅
1.3.6	A space has a title, graphical visualization, and discussion area.	Done ✅	Done ✅	Done ✅
1.3.7	A space has tags for categorization and search.	Done ✅	Done ✅	Done ✅

Functional Requirement	**Description**	**Backend**	**Web**	**Mobile**
1.3.8	Store discussions with version history.	Not Done	N/A	N/A
1.3.9	In-Space Activity Stream lists key events within each space.	In Progress	In Progress	In Progress
1.3.10	The stream logs new collaborators, nodes, edges, and discussions.	In Progress	In Progress	In Progress
1.3.11	Display the number of votes on space content.	Not Done	Not Done	Not Done
1.3.12	Report spaces, nodes, and comments for inappropriate content.	Not Done	In Progress	Not Done
1.4. Graph Presentation				
1.4.1	Graphical visualization of nodes and edges in a space.	Done ✅	Done ✅	Done ✅
1.4.2	Add new nodes by connecting them to existing nodes.	Done ✅	Done ✅	Done ✅
1.4.3	Show a bi-directional connection between two nodes as an edge.	Done ✅	Done ✅	Done ✅
1.4.4	Show the relation property of an edge on the graph.	Done ✅	Done ✅	Done ✅
1.4.5	Add, delete, or modify nodes in a joined space.	Done ✅	Done ✅	Done ✅
1.4.6	Add or remove a connection between nodes.	Done ✅	Done ✅	Done ✅
1.5. In-Space Search				

Functional Requirement	Description	Backend	Web	Mobile
1.5.1	Search in spaces by title, description, and tags.	In Progress	Not Done	Not Done
1.5.2	Apply a similarity search among nodes and edges.	Not Done	N/A	N/A
1.5.3	Navigate to a node or edge from the in-space search result.	N/A	Not Done	Not Done
1.6. In-App Search				
1.6.1	Search for spaces, nodes, edges, and profiles by text.	In Progress	Done ✅	Done ✅
1.6.2	Find related content using a similarity search.	Not Done	N/A	N/A
1.6.3	Display search results according to the content type.	Done ✅	Done ✅	Done ✅
1.6.4	Navigate to a profile from search results.	N/A	Done ✅	Done ✅
1.6.5	Navigate to a space from search results.	N/A	Done ✅	Done ✅
1.7. Browsing				
1.7.1	Browse new spaces.	Done ✅	Done ✅	Done ✅
1.7.2	Browse trending spaces.	Not Done	Not Done	Not Done
1.8. User Profile				
1.8.1	Do not expose users' sensitive information on their public profile.	Done ✅	Done ✅	Done ✅

Functional Requirement	**Description**	**Backend**	**Web**	**Mobile**
1.8.2	Show username, age, profession, and location on profiles.	Done ✅	Done ✅	Done ✅
1.8.3	List a user's created spaces, collaborations, and activities.	Done ✅	Done ✅	Done ✅
1.8.4	View other users' profile pages.	Done ✅	Done ✅	Done ✅
1.8.5	Edit own profile information.	Done ✅	Done ✅	Done ✅
1.9. Discussion				
1.9.1	Display the discussion of a space as an activity stream.	Done ✅	Done ✅	Done ✅
1.9.2	Add comments to a discussion in a joined space.	Done ✅	Done ✅	Done ✅
1.9.3	Vote up or down on discussions.	Not Done	Not Done	Not Done
1.10. Analytics				
1.10.1	Space-level analytics dashboard is visible to moderators.	Not Done	In Progress	N/A
1.10.2	The dashboard displays metrics on activity, content, and engagement.	Not Done	In Progress	N/A
1.10.3	System-level analytics dashboard is visible only to System Admins.	Not Done	In Progress	N/A
1.10.4	The dashboard displays platform-wide metrics on users and activity.	Not Done	In Progress	N/A

Functional Requirement	Description	Backend	Web	Mobile
1.11. User Reputation				
1.11.1	Display a "Top Contributor" label on certain user profiles.	Not Done	Not Done	Not Done
1.11.2	A full user scoring and reputation system is planned for a future release.	Not Done	Not Done	Not Done
1.12. Main Activity Stream				
1.12.1	Display a Main Activity Stream on the home page.	Not Done	Not Done	In Progress
1.12.2	The stream aggregates significant public events.	Not Done	N/A	N/A
1.12.3	The stream shows new spaces, trending spaces, and highly-voted content.	Not Done	Not Done	In Progress
1.12.4	The stream is not cluttered with minor events.	Not Done	N/A	N/A
1.12.5	Each item in the stream is a direct link to the content.	N/A	Not Done	In Progress

Project Structure and Repository Setup

For our project, we've organized our code in a monorepo on Github that holds our backend services, web frontend, and native Android application. This setup keeps everything in one place, making it easier for our team to work together and manage the different parts of the project.

The heart of our application is the backend, which we built using Python and the Django framework. We chose Django because it's easy to dive into and helps us build features quickly and reliably. To create the API that our web and mobile apps talk to, we used the Django REST Framework. All our data is stored in a PostgreSQL database, and we use Gunicorn to run the backend in a production environment. To make sure everything runs smoothly everywhere, we've dockerized our backend in a Docker container.

Our web frontend is a single-page application built with React, which allows us to create a dynamic and responsive user interface. We use Vite for our development and build process because it's incredibly fast. For handling navigation within the app, we use React Router, and the core graph visualization is powered by Reactflow. We also write tests for our components using Vitest and the React Testing Library to keep our app stable.

For our mobile app, we're developing a native Android app using Kotlin with MVVM architecture & Jetpack Compose. Using this tech stack we had to start our mobile app from scratch but our past experiences and familiarity with this framework allowed us to quickly implement fully functional features covered with unit tests. The Android app connects to the same Django backend as our web app, providing a seamless experience across platforms.

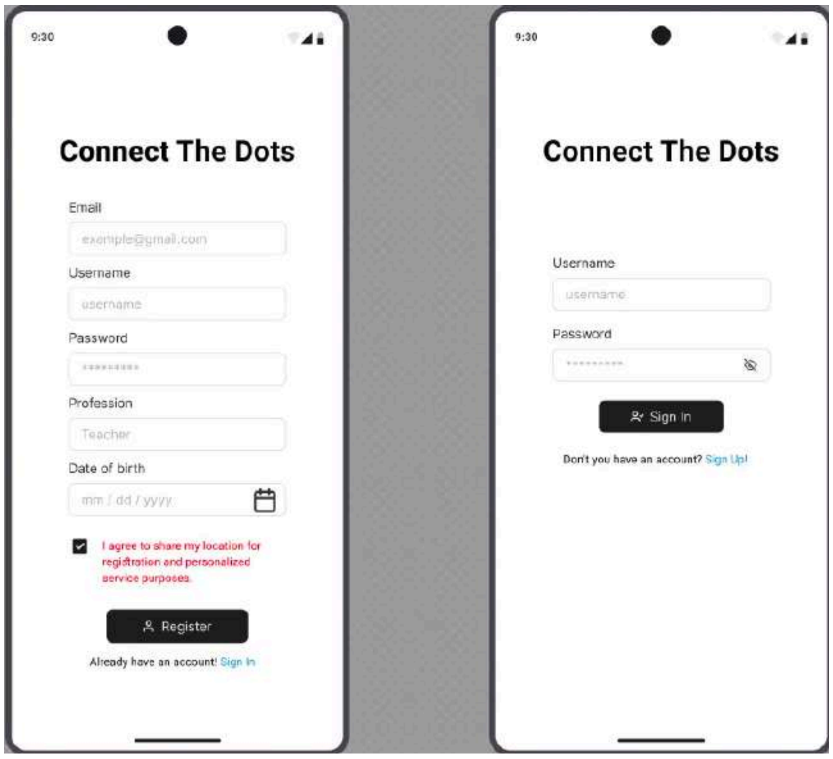
To automate our workflow, we've set up CI/CD with GitHub Actions. Whenever we push new code or create a pull request, automated workflows kick in to run tests for the backend, frontend, and mobile app. When we merge changes into our main branch, these workflows also handle deploying the new versions of our applications. The entire infrastructure is managed with Docker Compose, which lets us spin up our whole application with a single command, making both development and deployment much simpler.

Design

UI Mockups

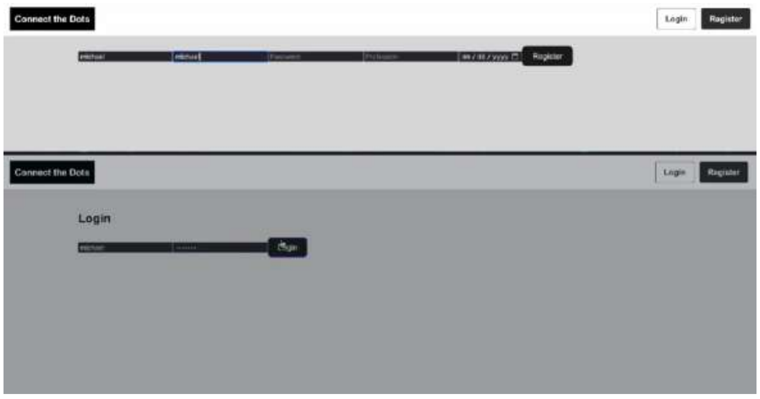
Registration & Login screens

A. Mobile



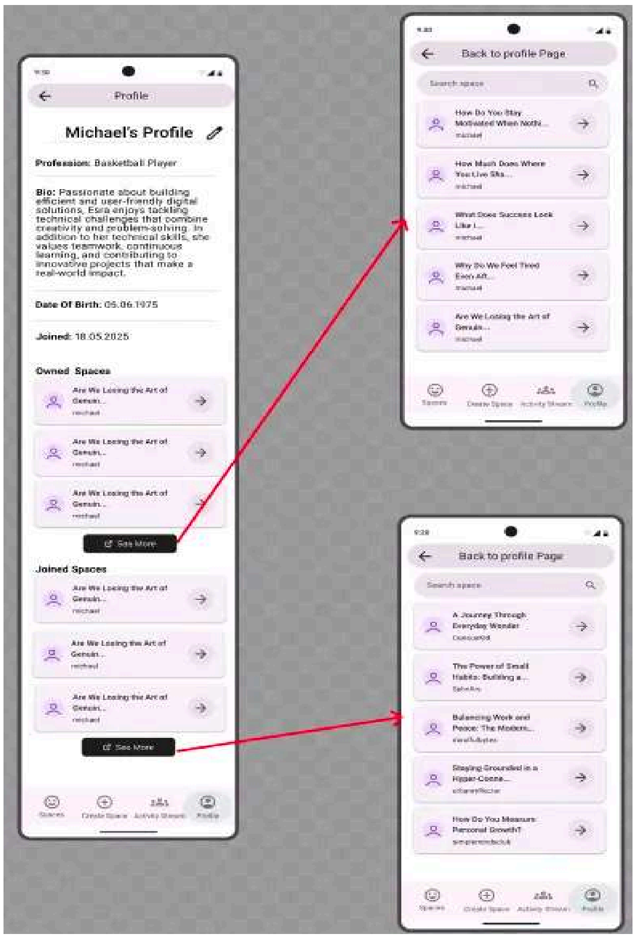
I agree to share my location for registration and personalized service purposes.

B. Web



Profile Screen

A. Mobile


Profile

Michael's Profile

Profession: Basketball Player

Bio: Passionate about building efficient and user-friendly digital solutions, Esra enjoys tackling technical challenges that combine creativity and problem-solving. In addition to her technical skills, she values teamwork, continuous learning, and contributing to innovative projects that make a real-world impact.

Date Of Birth: 05.06.1975

Joined: 18.05.2025

Owned Spaces

Are We Losing the Art of Genuin... michael

Are We Losing the Art of Genuin... michael

Are We Losing the Art of Genuin... michael

See More

Joined Spaces

Are We Losing the Art of Genuin... michael

Are We Losing the Art of Genuin... michael

Are We Losing the Art of Genuin... michael

See More

Back to profile Page

Search space

How Do You Stay Motivated When Nothi... michael

How Much Does Where You Live Sha... michael

What Does Success Look Like I... michael

Why Do We Feel Tired Even Aft... michael

Are We Losing the Art of Genuin... michael

Back to profile Page

Search space

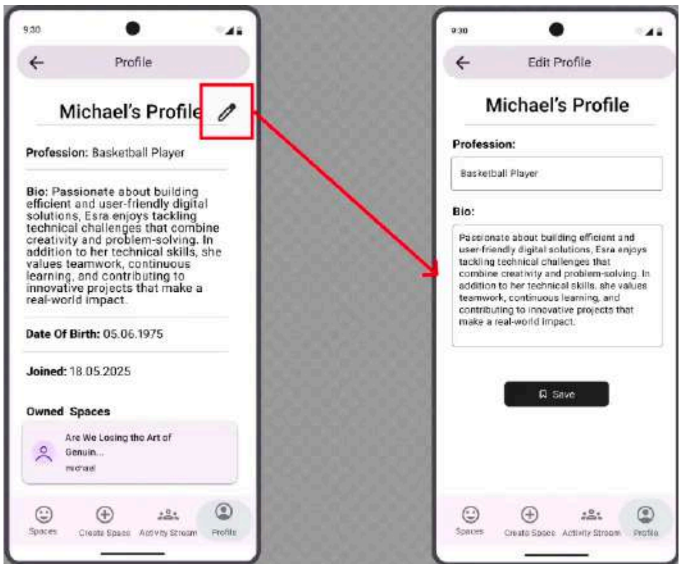
A Journey Through Everyday Wonder

The Power of Small Habits: Building a...

Balancing Work and Peace: The Modern...

Staying Grounded in a Hyper-Conne...

How Do You Measure Personal Growth?


Profile

Michael's Profile

Profession: Basketball Player

Bio: Passionate about building efficient and user-friendly digital solutions, Esra enjoys tackling technical challenges that combine creativity and problem-solving. In addition to her technical skills, she values teamwork, continuous learning, and contributing to innovative projects that make a real-world impact.

Date Of Birth: 05.06.1975

Joined: 18.05.2025

Owned Spaces

Are We Losing the Art of Genuin... michael

Edit Profile

Michael's Profile

Profession:

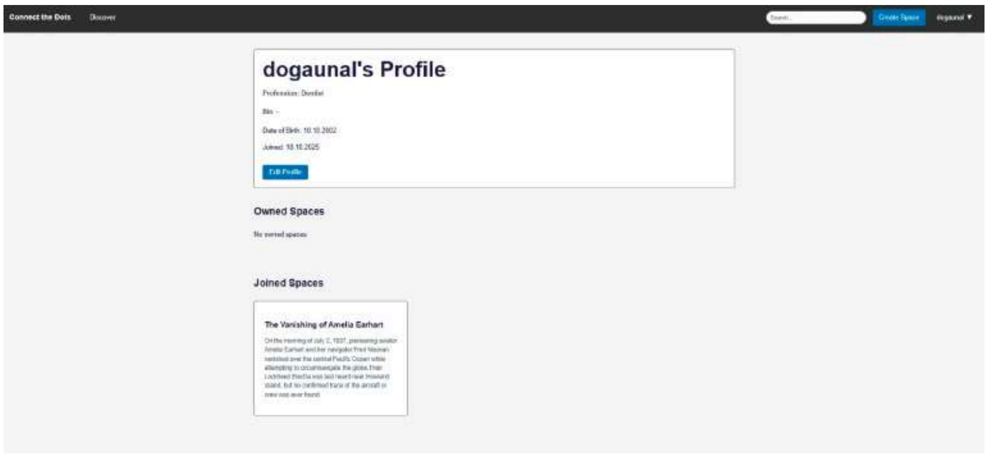
Basketball Player

Bio:

Passionate about building efficient and user-friendly digital solutions, Esra enjoys tackling technical challenges that combine creativity and problem-solving. In addition to her technical skills, she values teamwork, continuous learning, and contributing to innovative projects that make a real-world impact.

Save

B. Web



dogaunal's Profile

Profession: Dentist

Bio: -

Date of Birth: 10.10.2002

Joined: 10.10.2025

Edit Profile

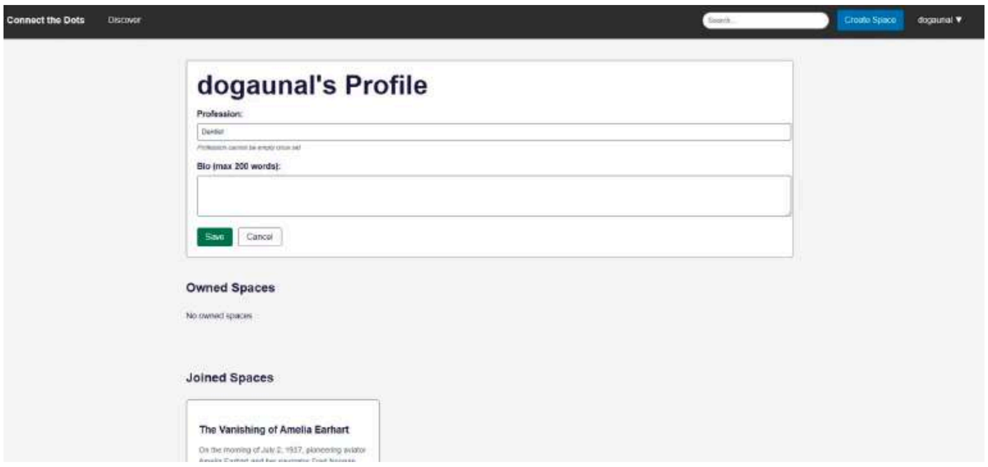
Owned Spaces

No owned spaces

Joined Spaces

The Vanishing of Amelia Earhart

On the morning of July 2, 1937, pioneering aviator Amelia Earhart and her navigator Fred Noonan vanished over the central Pacific Ocean while attempting to circumnavigate the globe. Their Lockheed Electra was last heard near Howland Island, but no confirmed trace of the aircraft or crew was ever found.



Connect the Dots   Discover   Search...   Create Space   dogaunal ▼

dogaunal's Profile

Profession:

Dentist

Profession cannot be empty once set

Bio (max 200 words):

Save   Cancel

Owned Spaces

No owned spaces

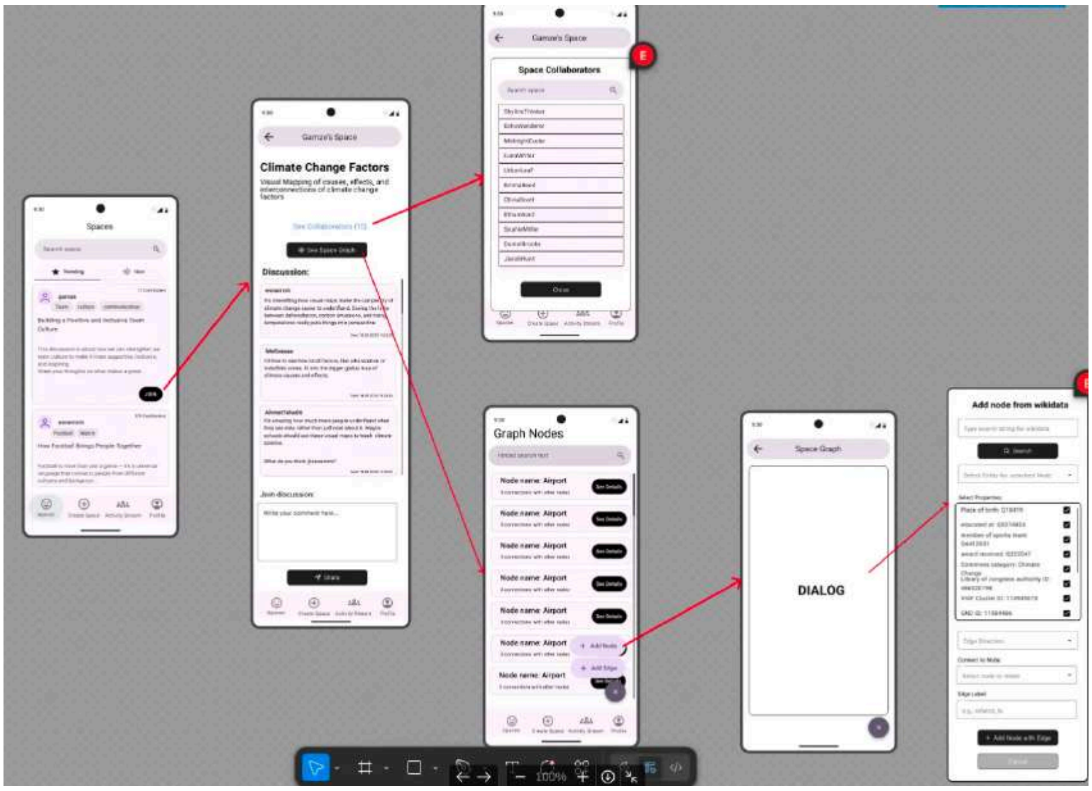
Joined Spaces

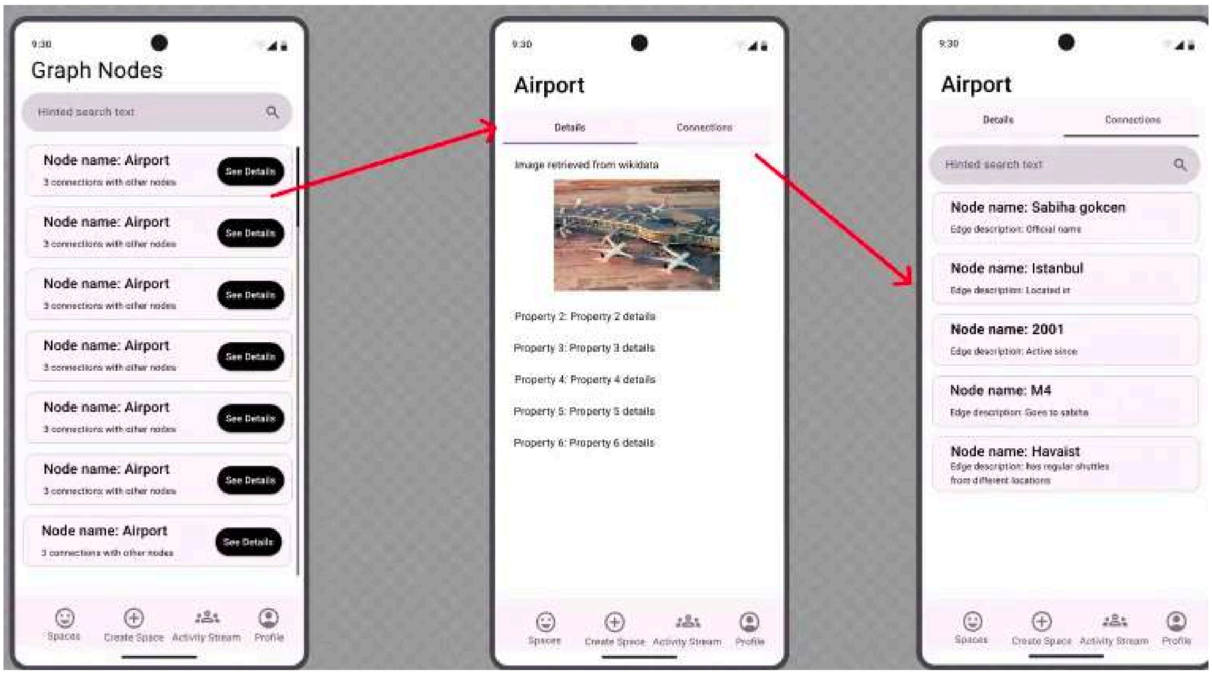
The Vanishing of Amelia Earhart

On the morning of July 2, 1937, pioneering aviator

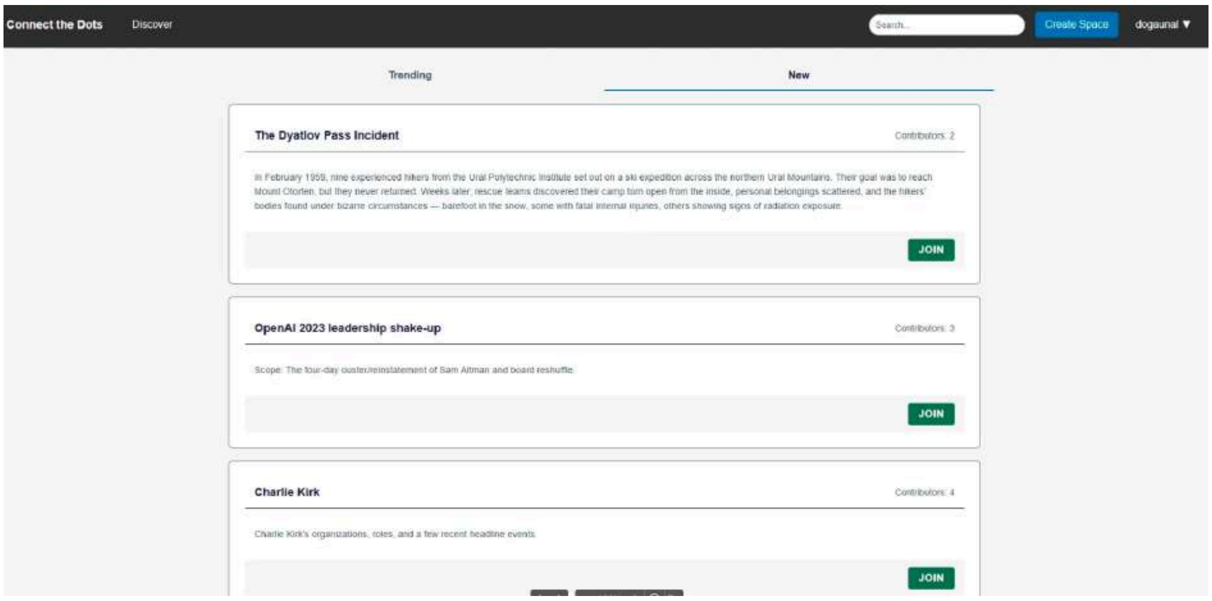
Space and Space Management

A. Mobile




9:30
Graph Nodes
Hinted search text
Node name: Airport
3 connections with other nodes
See Details
Spaces
Create Space
Activity Stream
Profile
Airport
Details
Connections
Image retrieved from wikidata
Property 2: Property 2 details
Property 3: Property 3 details
Property 4: Property 4 details
Property 5: Property 5 details
Property 6: Property 6 details
Node name: Sabiha gokcen
Edge description: Official name
Node name: Istanbul
Edge description: Located in
Node name: 2001
Edge description: Active since
Node name: M4
Edge description: Goes to sabiha
Node name: Havaist
Edge description: has regular shuttles from different locations

B. Web



The Dyatlov Pass Incident Contributors: 2

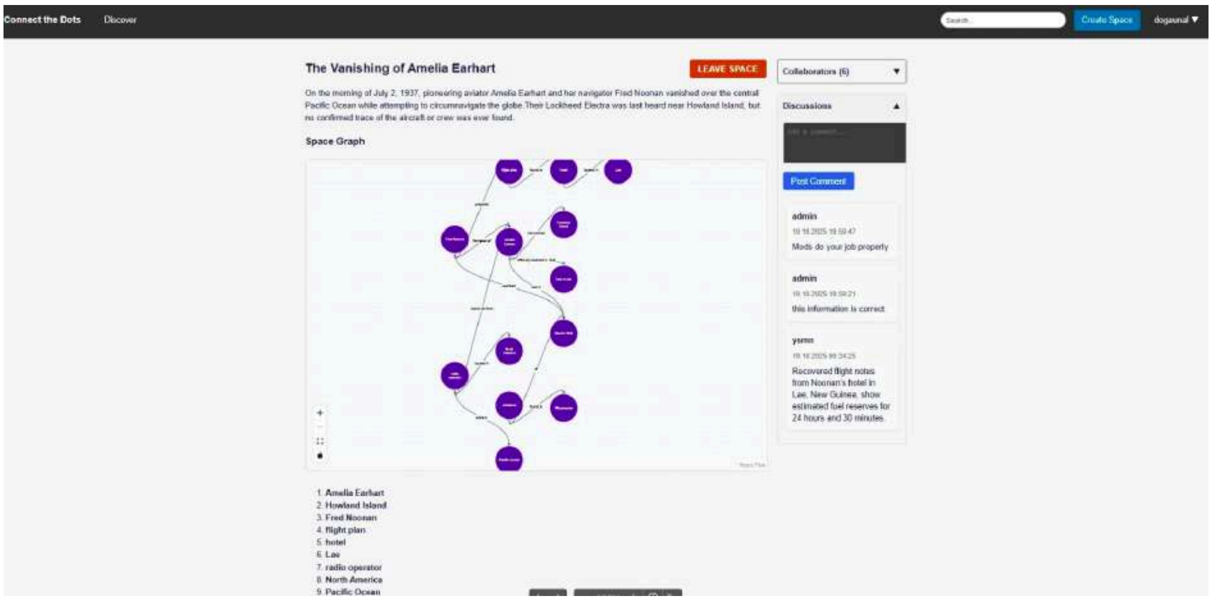
In February 1959, nine experienced hikers from the Ural Polytechnic Institute set out on a ski expedition across the northern Ural Mountains. Their goal was to reach Mount Otorten, but they never returned. Weeks later, rescue teams discovered their camp torn open from the inside, personal belongings scattered, and the hikers' bodies found under bizarre circumstances — barefoot in the snow, some with fatal internal injuries, others showing signs of radiation exposure.

OpenAI 2023 leadership shake-up Contributors: 3

Scope: The four-day ouster/reinstatement of Sam Altman and board reshuffle

Charlie Kirk Contributors: 4

Charlie Kirk's organizations, roles, and a few recent headline events



The Vanishing of Amelia Earhart

On the morning of July 2, 1937, pioneering aviator Amelia Earhart and her navigator Fred Noonan vanished over the central Pacific Ocean while attempting to circumnavigate the globe. Their Lockheed Electra was last heard near Howland Island, but no confirmed trace of the aircraft or crew was ever found.

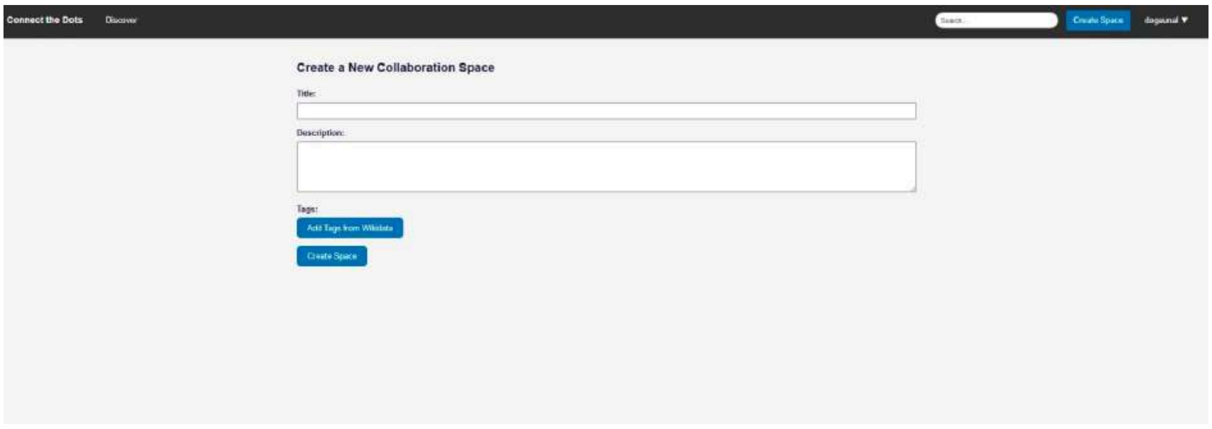
Space Graph

1. Amelia Earhart
2. Howland Island
3. Fred Noonan
4. flight plan
5. hotel
6. Lae
7. radio operator
8. North America
9. Pacific Ocean

admin
19.10.2025 19:59:47
Mods do your job properly

admin
19.10.2025 19:59:21
this information is correct

ysmn
19.10.2025 00:24:25
Recovered flight notes from Noonan's hotel in Lae, New Guinea, show estimated fuel reserves for 24 hours and 30 minutes.



Create a New Collaboration Space

Title:

Description:

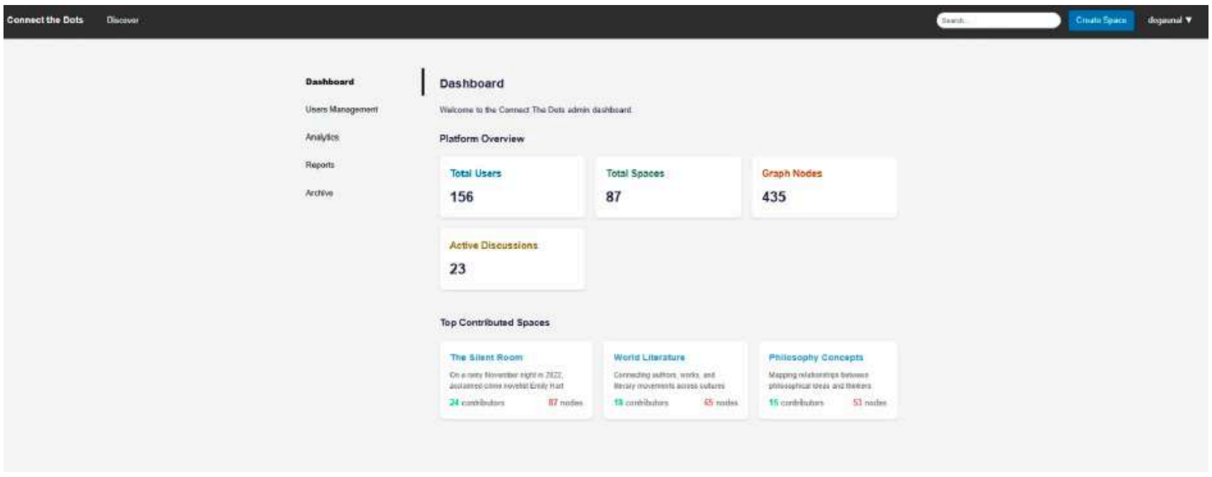
Tags:

Add Tags from Wikidata

Create Space

Back Office (Only Available in Web)

A. Dashboard section where we keep the total overview of the system



Dashboard

Welcome to the Connect The Dots admin dashboard.

Platform Overview

Total Users
156

Total Spaces
87

Graph Nodes
435

Active Discussions
23

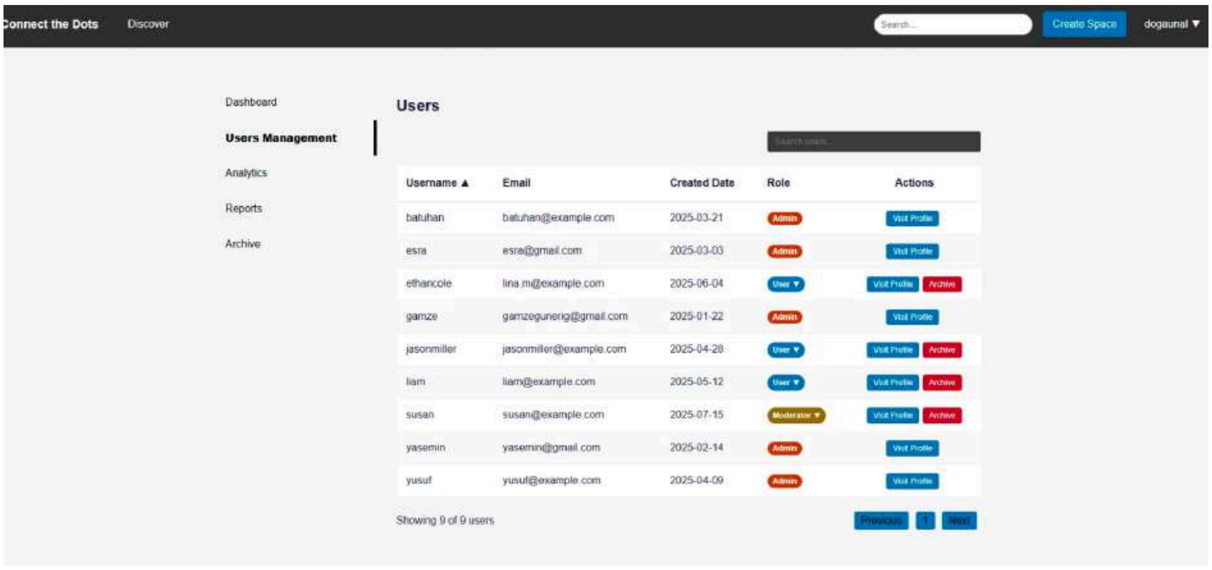
Top Contributed Spaces

The Silent Room
On a rainy November night in 2022, acclaimed crime novelist Emily Hart
24 contributors 87 nodes

World Literature
Connecting authors, works, and literary movements across cultures
18 contributors 65 nodes

Philosophy Concepts
Mapping relationships between philosophical ideas and thinkers
15 contributors 53 nodes

B. User management section where we see all system users, their roles, emails and possible actions that can be performed on them

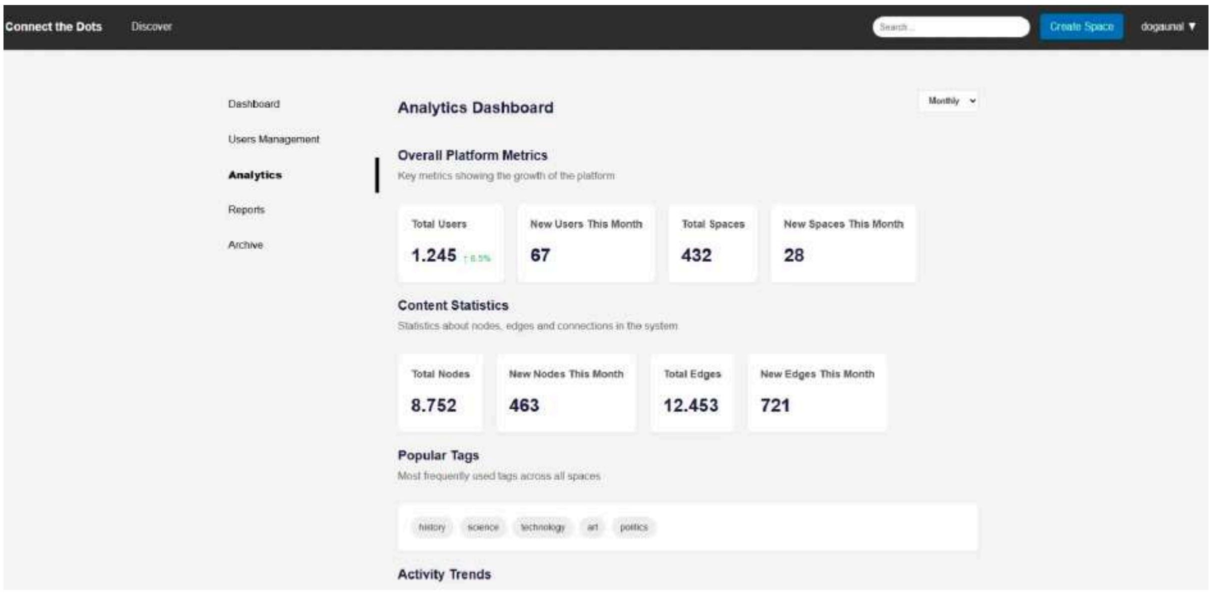


Users

Username ▲	Email	Created Date	Role	Actions
batuhan	batuhan@example.com	2025-03-21	Admin	Visit Profile
esra	esra@gmail.com	2025-03-03	Admin	Visit Profile
ethancole	lina.m@example.com	2025-06-04	User ▼	Visit Profile Archive
gamze	gamzegunerig@gmail.com	2025-01-22	Admin	Visit Profile
jasonmiller	jasonmiller@example.com	2025-04-28	User ▼	Visit Profile Archive
liam	liam@example.com	2025-05-12	User ▼	Visit Profile Archive
susan	susan@example.com	2025-07-15	Moderator ▼	Visit Profile Archive
yasemin	yasemin@gmail.com	2025-02-14	Admin	Visit Profile
yusuf	yusuf@example.com	2025-04-09	Admin	Visit Profile

Showing 9 of 9 users

C. Analytics section, where we see overall statistics of the system and system usage



Analytics Dashboard

Overall Platform Metrics

Key metrics showing the growth of the platform

Total Users	New Users This Month	Total Spaces	New Spaces This Month
1.245 ↑ 8.5%	67	432	28

Content Statistics

Statistics about nodes, edges and connections in the system

Total Nodes	New Nodes This Month	Total Edges	New Edges This Month
8.752	463	12.453	721

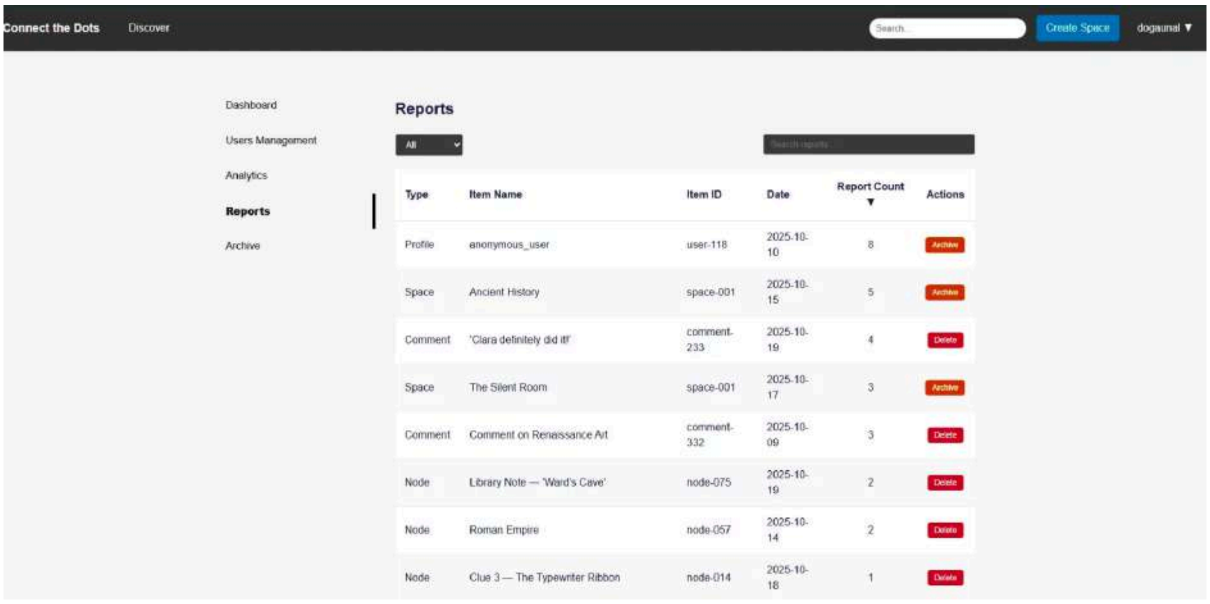
Popular Tags

Most frequently used tags across all spaces

history science technology art politics

Activity Trends

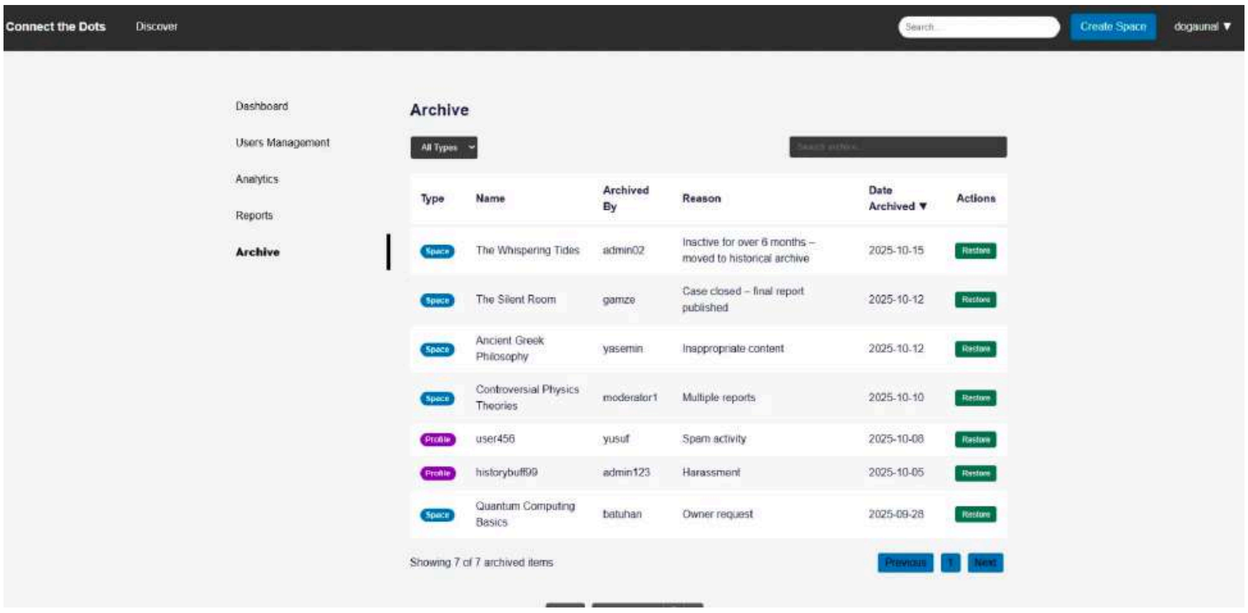
D. Reports where we see all reported system components, such as comments, users, spaces, etc.



Reports

Type	Item Name	Item ID	Date	Report Count ▼	Actions
Profile	anonymous_user	user-118	2025-10-10	8	Archive
Space	Ancient History	space-001	2025-10-15	5	Archive
Comment	'Clara definitely did it!'	comment-233	2025-10-19	4	Delete
Space	The Silent Room	space-001	2025-10-17	3	Archive
Comment	Comment on Renaissance Art	comment-332	2025-10-09	3	Delete
Node	Library Note — 'Ward's Cave'	node-075	2025-10-19	2	Delete
Node	Roman Empire	node-057	2025-10-14	2	Delete
Node	Clue 3 — The Typewriter Ribbon	node-014	2025-10-18	1	Delete

E. Archive section where we see archived system components.



Archive

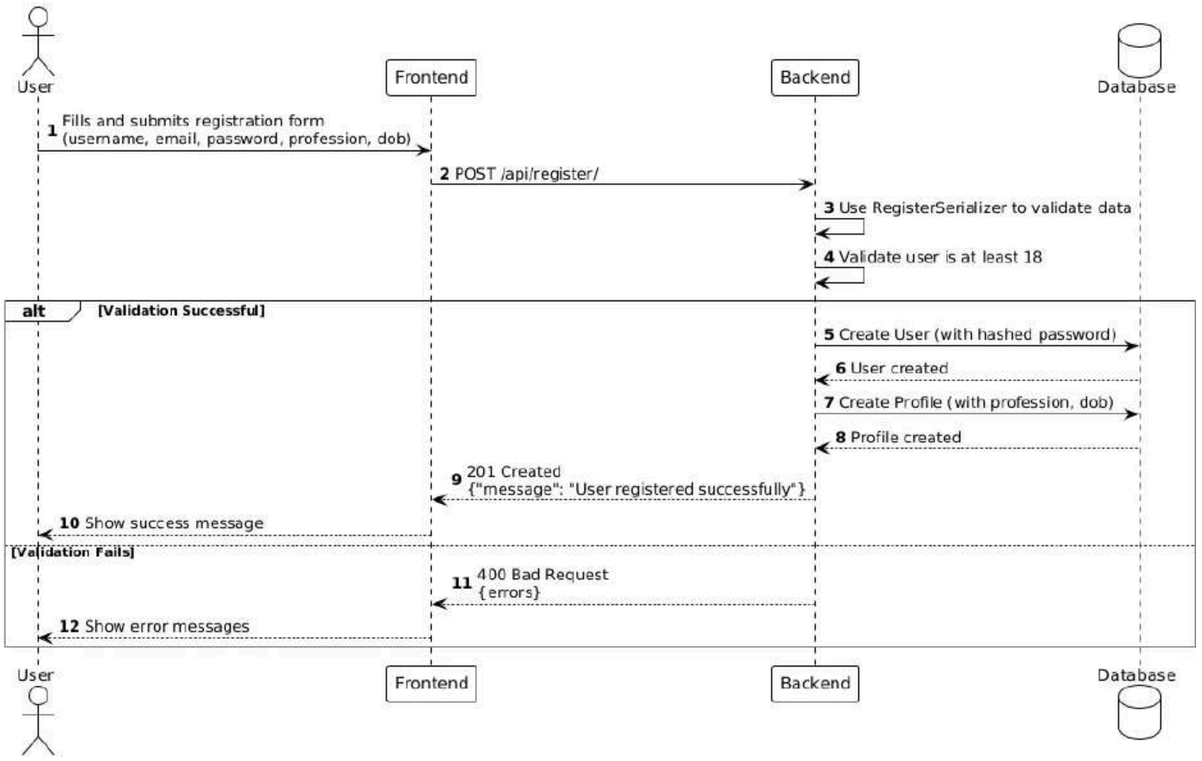
Type	Name	Archived By	Reason	Date Archived ▼	Actions
Space	The Whispering Tides	admin02	Inactive for over 6 months – moved to historical archive	2025-10-15	Restore
Space	The Silent Room	gamze	Case closed – final report published	2025-10-12	Restore
Space	Ancient Greek Philosophy	yasemin	Inappropriate content	2025-10-12	Restore
Space	Controversial Physics Theories	moderator1	Multiple reports	2025-10-10	Restore
Profile	user456	yusuf	Spam activity	2025-10-08	Restore
Profile	historybuff99	admin123	Harassment	2025-10-05	Restore
Space	Quantum Computing Basics	batuhan	Owner request	2025-09-28	Restore

Showing 7 of 7 archived items

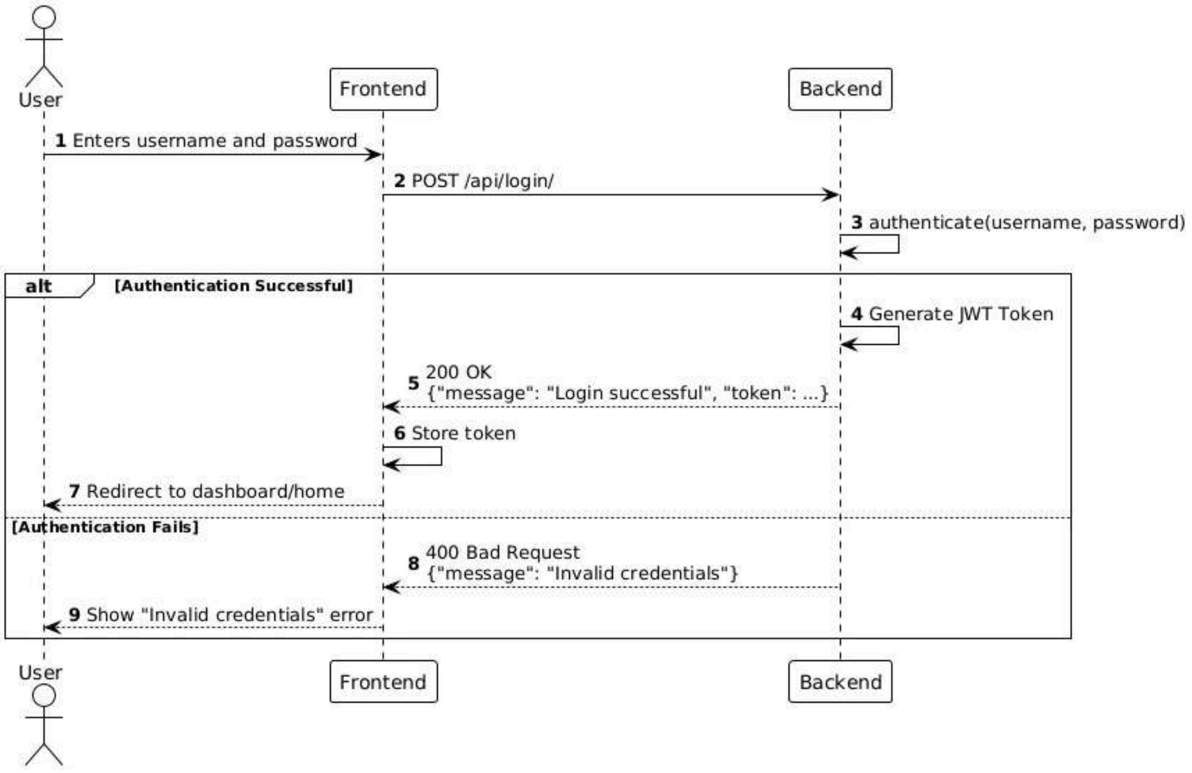
UML Diagrams

Sequence Diagrams

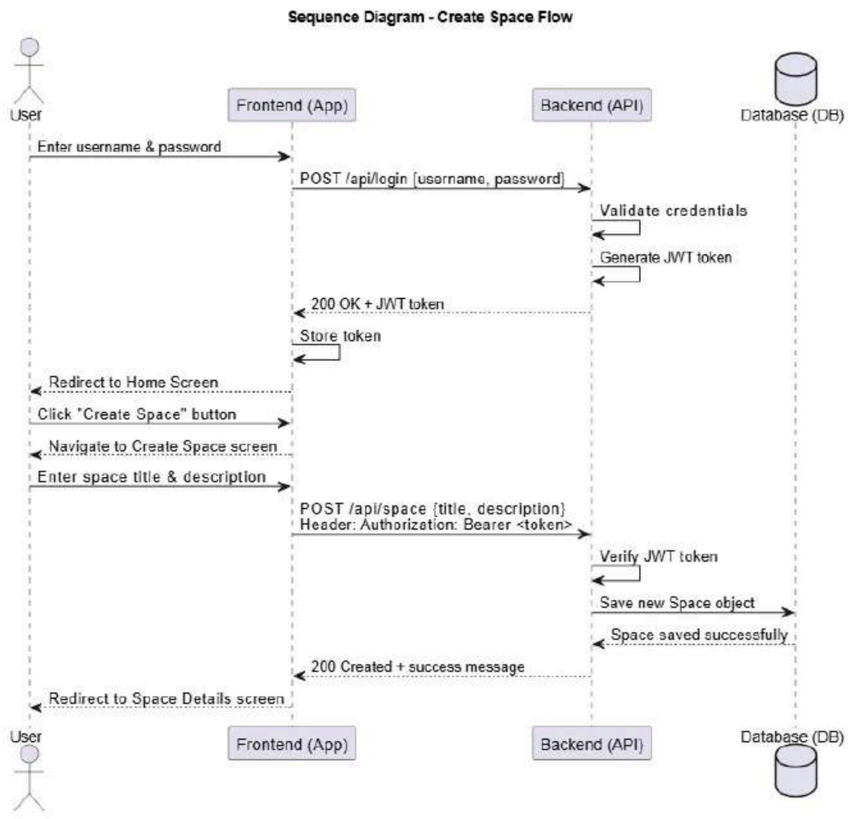
A. Registration


User
Frontend
Backend
Database
1 Fills and submits registration form (username, email, password, profession, dob)
2 POST /api/register/
3 Use RegisterSerializer to validate data
4 Validate user is at least 18
alt [Validation Successful]
5 Create User (with hashed password)
6 User created
7 Create Profile (with profession, dob)
8 Profile created
9 201 Created {"message": "User registered successfully"}
10 Show success message
[Validation Fails]
11 400 Bad Request {errors}
12 Show error messages

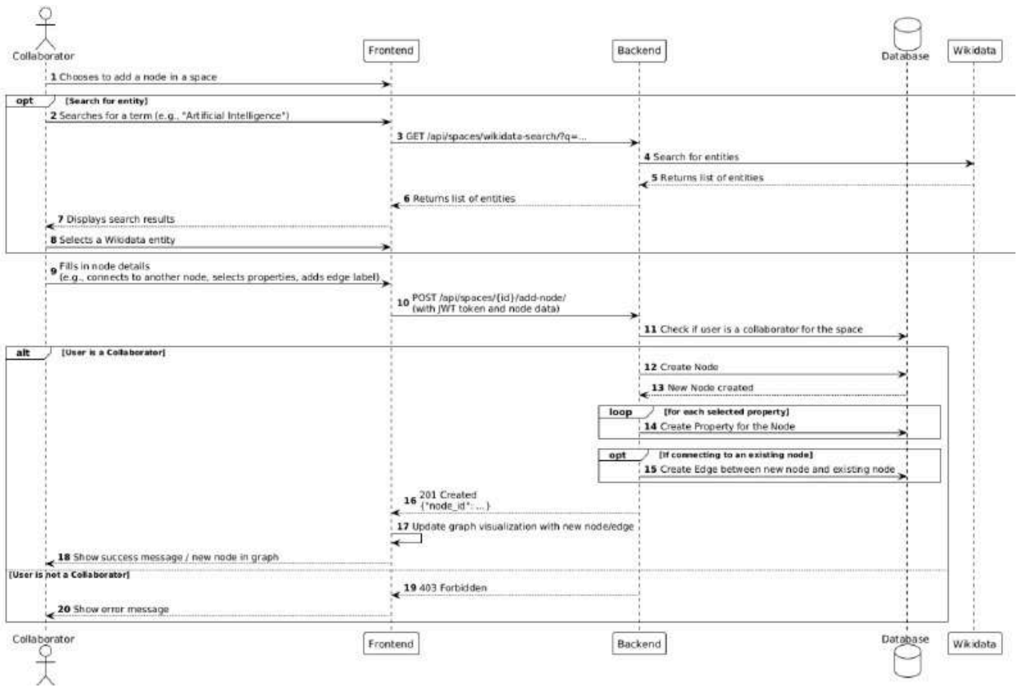
B. Login


User
Frontend
Backend
1 Enters username and password
2 POST /api/login/
3 authenticate(username, password)
alt [Authentication Successful]
4 Generate JWT Token
5 200 OK {"message": "Login successful", "token": ...}
6 Store token
7 Redirect to dashboard/home
[Authentication Fails]
8 400 Bad Request {"message": "Invalid credentials"}
9 Show "Invalid credentials" error

C. Create Space

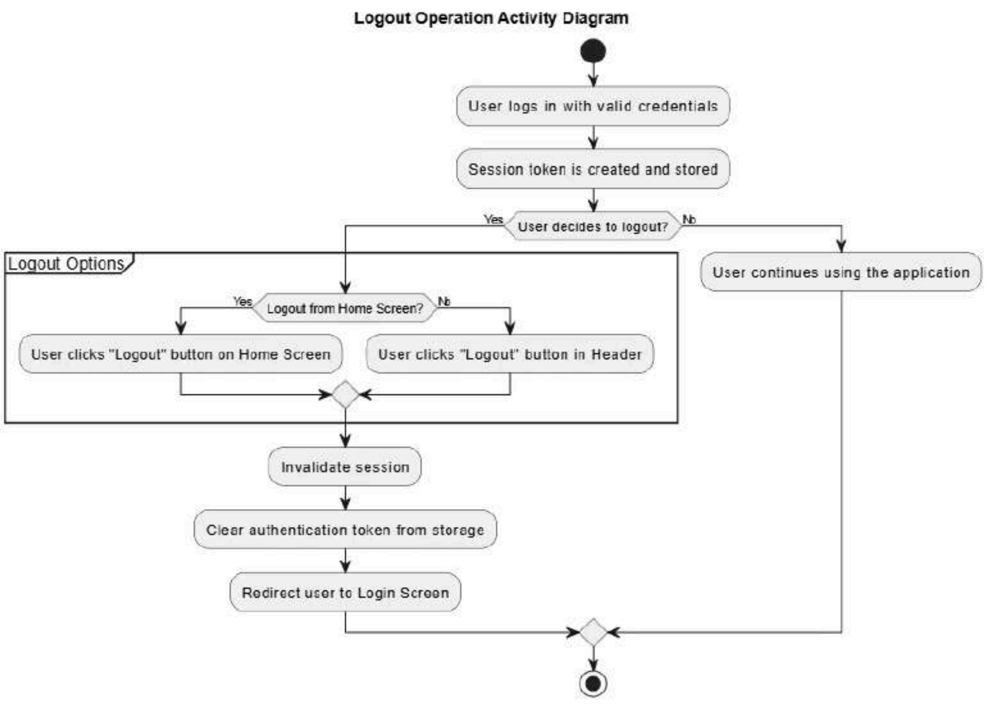

Sequence Diagram - Create Space Flow
User
Frontend (App)
Backend (API)
Database (DB)
Enter username & password
POST /api/login {username, password}
Validate credentials
Generate JWT token
200 OK + JWT token
Store token
Redirect to Home Screen
Click "Create Space" button
Navigate to Create Space screen
Enter space title & description
POST /api/space {title, description}
Header: Authorization: Bearer <token>
Verify JWT token
Save new Space object
Space saved successfully
200 Created + success message
Redirect to Space Details screen
User
Frontend (App)
Backend (API)
Database (DB)

D. Adding Node to Space Graph

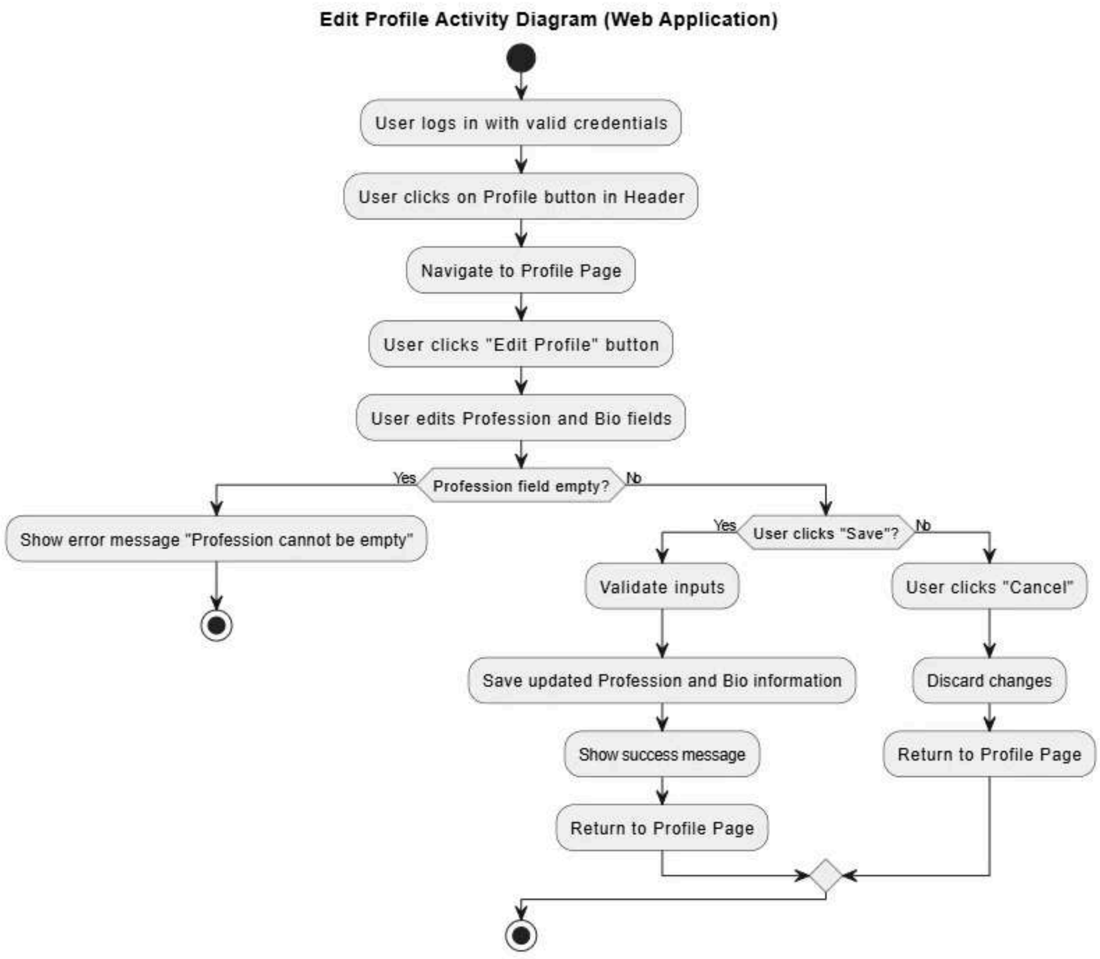

Collaborator
Frontend
Backend
Database
Wikidata
1 Chooses to add a node in a space
opt
[Search for entity]
2 Searches for a term (e.g., "Artificial Intelligence")
3 GET /api/spaces/wikidata-search/?q=...
4 Search for entities
5 Returns list of entities
6 Returns list of entities
7 Displays search results
8 Selects a Wikidata entity
9 Fills in node details
(e.g., connects to another node, selects properties, adds edge label)
10 POST /api/spaces/{id}/add-node/
(with JWT token and node data)
11 Check if user is a collaborator for the space
alt
[User is a Collaborator]
12 Create Node
13 New Node created
loop
[for each selected property]
14 Create Property for the Node
opt
[If connecting to an existing node]
15 Create Edge between new node and existing node
16 201 Created
{"node_id": ...}
17 Update graph visualization with new node/edge
18 Show success message / new node in graph
[User is not a Collaborator]
19 403 Forbidden
20 Show error message
Collaborator
Frontend
Backend
Database
Wikidata

Activity Diagrams

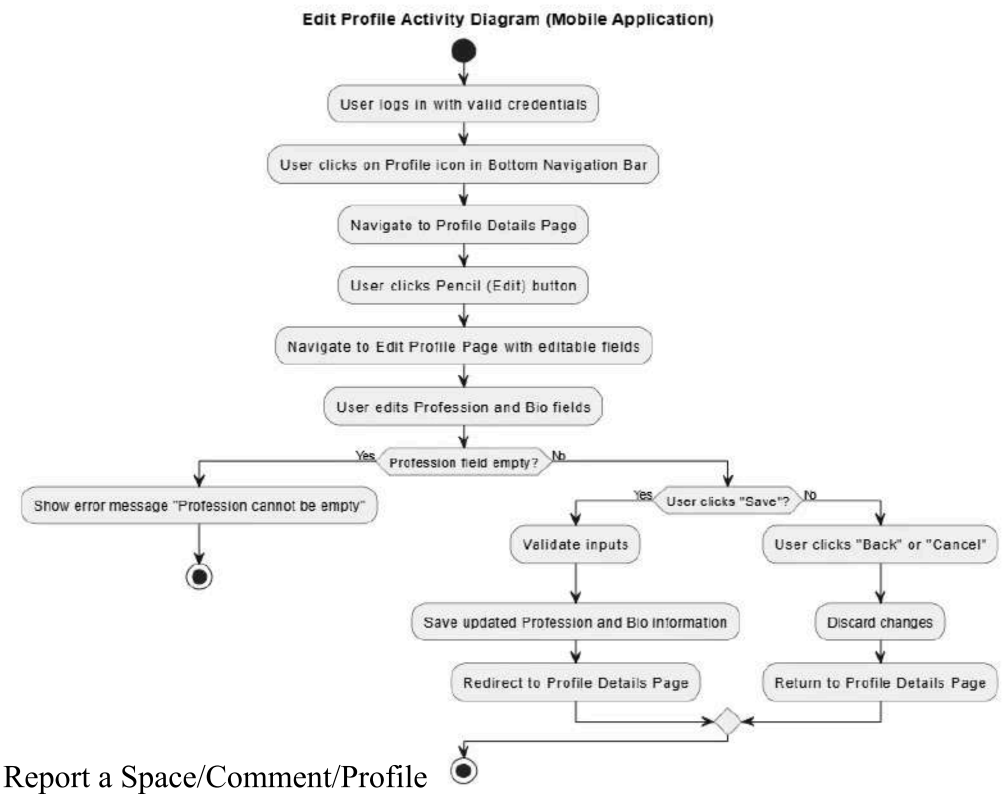
A. Logout


Logout Operation Activity Diagram
User logs in with valid credentials
Session token is created and stored
User decides to logout?
Yes
No
Logout Options
Logout from Home Screen?
Yes
No
User clicks "Logout" button on Home Screen
User clicks "Logout" button in Header
User continues using the application
Invalidate session
Clear authentication token from storage
Redirect user to Login Screen

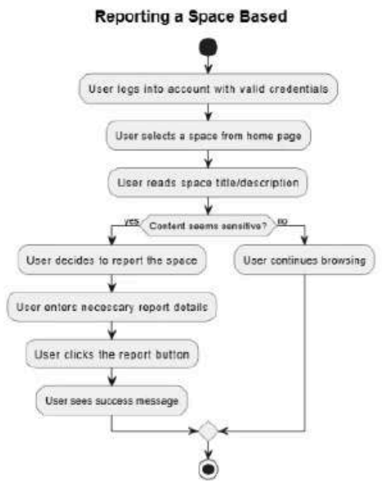
B. Edit Profile (Web)

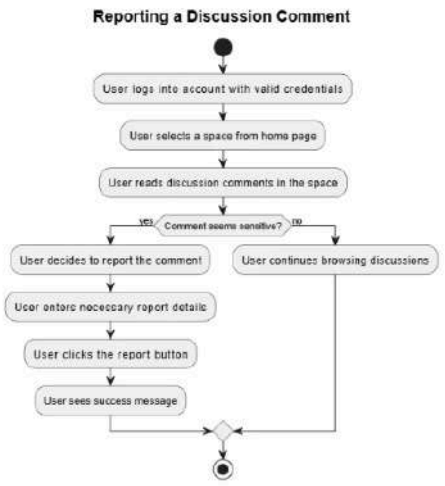

Edit Profile Activity Diagram (Web Application)
User logs in with valid credentials
User clicks on Profile button in Header
Navigate to Profile Page
User clicks "Edit Profile" button
User edits Profession and Bio fields
Profession field empty?
Yes
No
Show error message "Profession cannot be empty"
User clicks "Save"?
Yes
No
Validate inputs
Save updated Profession and Bio information
Show success message
Return to Profile Page
User clicks "Cancel"
Discard changes
Return to Profile Page

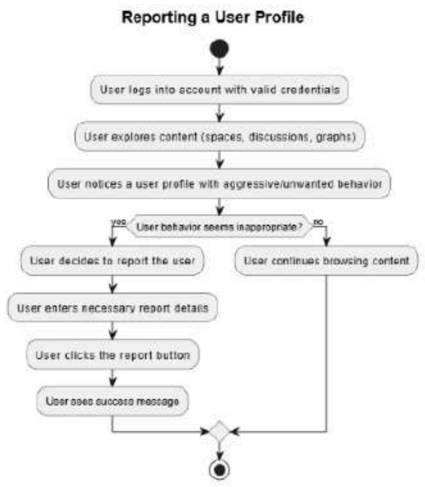
C. Edit Profile (Mobile)


Edit Profile Activity Diagram (Mobile Application)
User logs in with valid credentials
User clicks on Profile icon in Bottom Navigation Bar
Navigate to Profile Details Page
User clicks Pencil (Edit) button
Navigate to Edit Profile Page with editable fields
User edits Profession and Bio fields
Yes
Profession field empty?
No
Show error message "Profession cannot be empty"
Yes
User clicks "Save"?
No
Validate inputs
User clicks "Back" or "Cancel"
Save updated Profession and Bio information
Discard changes
Redirect to Profile Details Page
Return to Profile Details Page

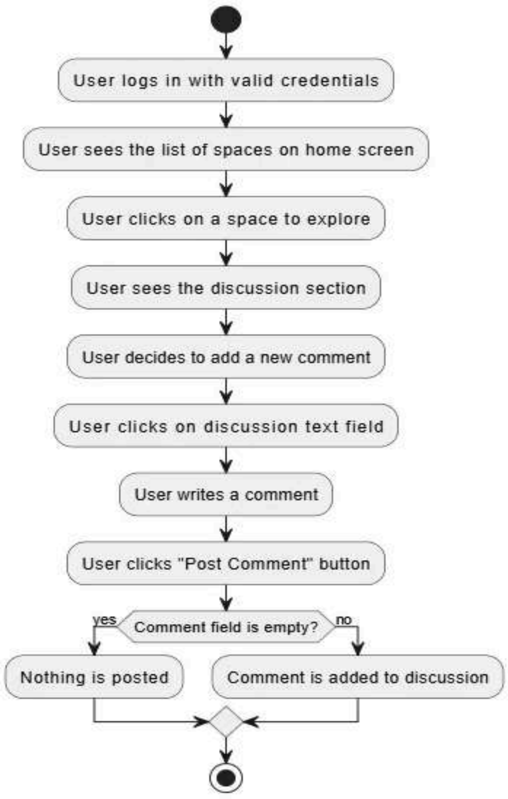
D. Report a Space/Comment/Profile


Reporting a Space Based
User logs into account with valid credentials
User selects a space from home page
User reads space title/description
yes
Content seems sensitive?
no
User decides to report the space
User continues browsing
User enters necessary report details
User clicks the report button
User sees success message


Reporting a Discussion Comment
User logs into account with valid credentials
User selects a space from home page
User reads discussion comments in the space
yes
Comment seems sensitive?
no
User decides to report the comment
User continues browsing discussions
User enters necessary report details
User clicks the report button
User sees success message


Reporting a User Profile
User logs into account with valid credentials
User explores content (spaces, discussions, graphs)
User notices a user profile with aggressive/unwanted behavior
yes
User behavior seems inappropriate?
no
User decides to report the user
User continues browsing content
User enters necessary report details
User clicks the report button
User sees success message

E. Adding a Comment on Discussion


User logs in with valid credentials
User sees the list of spaces on home screen
User clicks on a space to explore
User sees the discussion section
User decides to add a new comment
User clicks on discussion text field
User writes a comment
User clicks "Post Comment" button
Comment field is empty?
yes
no
Nothing is posted
Comment is added to discussion

Use Case Scenarios

1. Dyatlov Pass

Dimitri Petrov, a 29-year-old history teacher in Moskov, often used **Connect-The-Dots** to prepare visual lessons for his students about unsolved events.

After a long day at school, he sat by the window of his small apartment overlooking the skyline and saw a news headline:

> "Russian authorities reopen Dyatlov Pass investigation"

He opened the **Connect-The-Dots** app and searched **Dyatlov Pass**.

There wasnt a such board, he decided to create the space and added the following as description:

In February 1959, nine experienced hikers from the Ural Polytechnic Institute set out on a ski expedition across the northern Ural Mountains.

Their goal was to reach Mount Otorten, but they never returned.

Weeks later, rescue teams discovered their camp torn open from the inside, personal belongings scattered, and the hikers' bodies found under bizarre circumstances — barefoot in the snow, some with fatal internal injuries, others showing signs of radiation exposure.

Dimitri named the board as The Dyatlov Pass Incident. Since he was familiar with the background of the incident, thus, he started to create the node

Firstly, he decided the add node as Dyatlov Pass, and searched from wikidata.

The first clue for the incident was the tent which found Dyatlov Pass in a weird condition, he added this as second node and connected the nodes with the relation of "found at"

Second clue was the footprints which starting from tent to nearby forest, barefood, he added

node searched footprint from wikidata, connected the node with tent, and added the relation as

"starting_from". Than, he searched forest from wikidata, connected with the footprint and added the relation as "near_by".

And lastly, he decided to add a node for *Yuri Doroshenko, which his body* found at the edge of the forest.

2. The Vanishing of Amelia Earhart

Storyline:

Samantha lives a quiet, simple life by the Pacific.

Every morning, she bikes five miles to the **Marine Data Center**, where she worked as a **data technician**.

Her job is repetitive — checking sonar maps, cleaning metadata, and fixing ocean files.

One night, a coworker told her about *Connect-The-Dots, which is an online app where people collaboratively solve or think about some topics.*

After a while, when she was finishing her routine check, she saw a headline:

> “Researchers believe they’ve found Amelia Earhart’s plane near Pacific ocean.”

Out of curiosity, she logged into *Connect-The-Dots*, and found “The Vanishing of Amelia Earhart,” and opened the existing board.

Then she saw a breaking news about a metallic object that is found by Nikumaroro.

Clue 1:
At 8:43 AM, Earhart’s voice was heard over shortwave:

> “We are on the line 157-337… we will repeat this message… we are running north and south.”
> Signal faded abruptly. No further contact.

→ *Amelia Earhart → Howland Island* (**last contact**)

Clue 2:
Recovered flight notes from Noonan’s hotel in Lae, New Guinea, show estimated fuel reserves for

24 hours and 30 minutes.

Analysts believe they may have missed Howland by 50–100 miles.

→*Fred Noonan → Flight Plan*(**navigator record**)

Clue 3:
Between July 3–5, several amateur operators in North America and the Pacific reported hearing Earhart’s distress calls:

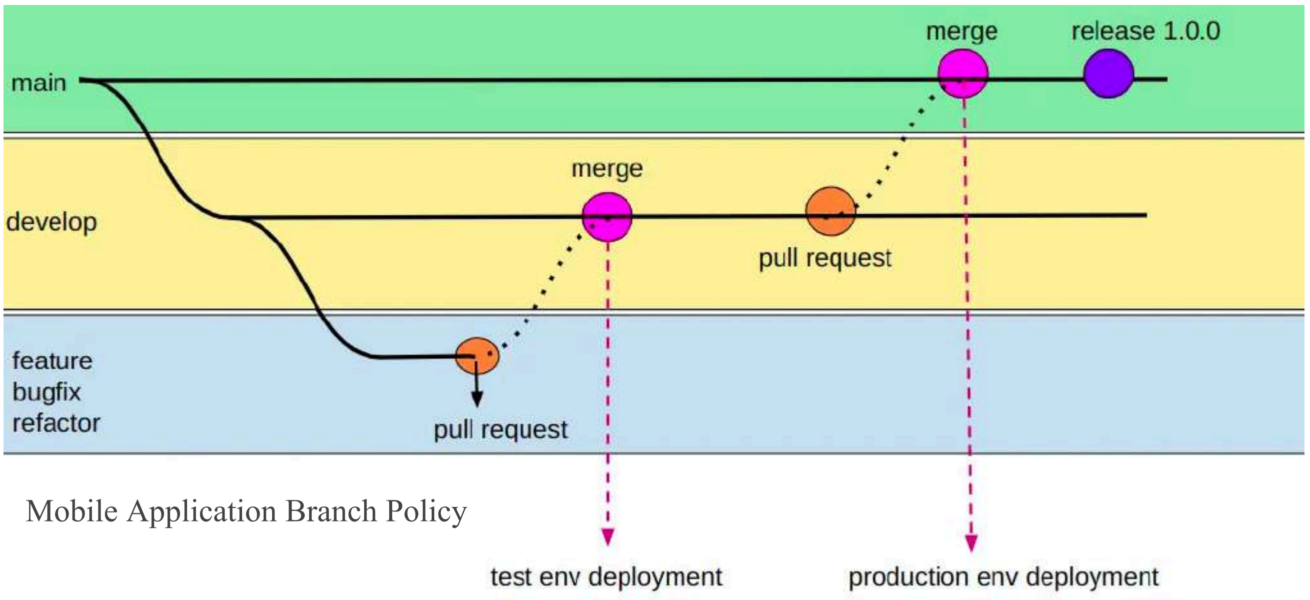
> “This is Amelia Earhart… we can’t last long.”
> Most were dismissed as atmospheric interference, but modern analysis suggests at least two may have been authentic.

→ *Radio Operators → Amelia Earhart* (**possible transmission reception**)

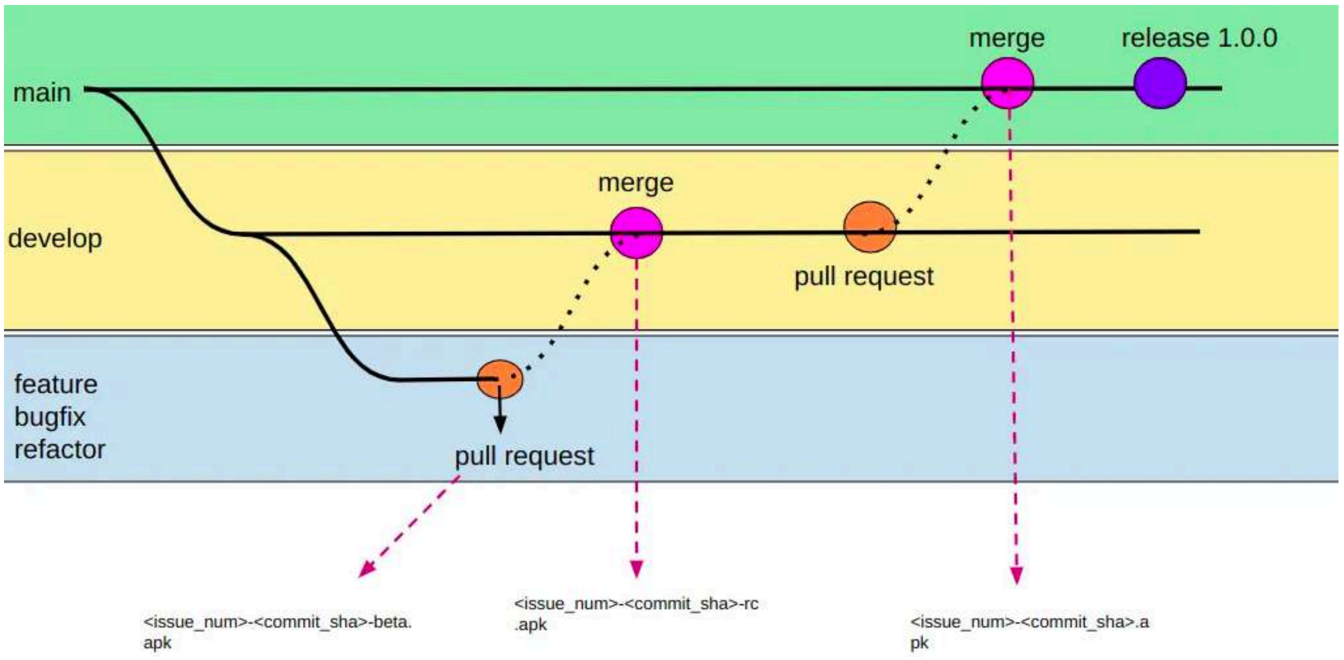
Infrastructure & Unit Testing

Branch Strategy & Environments

Web Application Branch Policy



Mobile Application Branch Policy


main
merge
release 1.0.0
develop
merge
pull request
feature
bugfix
refactor
pull request
<issue_num>-<commit_sha>-beta.apk
<issue_num>-<commit_sha>-rc.apk
<issue_num>-<commit_sha>.apk

Branch	Purpose	Environment	Pipeline Trigger	Deployment Target

feature/ bugfix/ refactor/	Isolated development work	none for web app, apk build for mobile app	PR opened & commit push → CI run	*No deployment* - mobile app artifacts (e.g. beta APK) generated
develop	Integration of all features before release	Test/Staging	Merge into develop → CI/CD	Deploys automatically to test/staging environment -mobile app artifacts (e.g. rc APK) generated
main	Stable release branch	Production	Merge into main → Release pipeline	Deploys automatically to production environment - mobile app artifacts (e.g. APK) generated

Stage	**Trigger**	**Action**	**Infra Resource**
Feature → PR **Feature Push**	pull_request : develop push: feature	Build + Unit Tests	GitHub Actions Runner Github Actions Artifact Docker Compose
Develop Merge	pull_request: main push: develop	Build + Unit Tests + Deploy to Test Env	Github Actions Runner Github Actions Artifact Amazon EC2 Docker Compose
Main Merge	push: main	Build + Unit Tests + Deploy to Production Env	Github Actions Runner Github Actions Artifact Amazon EC2 Docker Compose

Environment Segregation

Each branch corresponds to a different environment:

- **Feature branches → ephemeral (CI only)**
- **Develop → test environment**
- **Main → production environment**

Separate Docker Compose profiles for the environments & different django settings.

Separate deployment instances for web application. (test:http://16.171.162.104:3000/
production:http://13.53.174.44:3000/)

Deployment Automation

GitHub Actions environment protection rules.

Separate workflows:

- Web Application(also contains cd parts):
  - ci_feature.yml
  - ci_develop.yml
  - ci_main.yml
- Mobile Application(contains .apk build and artifact push):
  - ci_feature_mobile.yml
  - ci_develop_mobile.yml
  - ci_main_mobile.yml

Artifact Lifecycle Management (Mobile Application)

beta.apk → rc.apk → release.apk

Unit Tests

Unit-Testing Integration within the CI/CD Pipeline (Web Application)

The continuous integration and continuous deployment (CI/CD) workflows incorporate a fully automated unit-testing process that operates across all active development branches (feature/, *bugfix/*, refactor/*, develop, and main). Each pipeline execution automatically triggers the testing stage, ensuring that any modification to the codebase is systematically verified for functional correctness and reliability prior to merging or deployment. This approach enforces a consistent quality gate throughout the software lifecycle and minimizes the risk of introducing regressions into shared branches.

A. Execution Environment

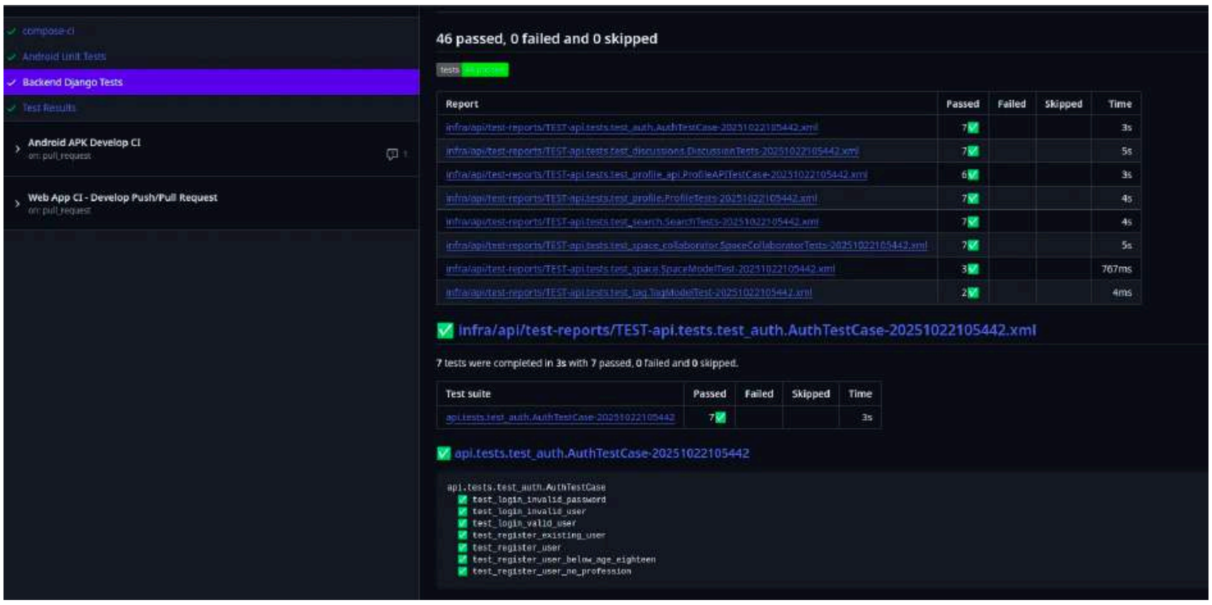
The unit tests are executed within an ephemeral, containerized environment orchestrated by Docker Compose. This setup reproduces the complete production topology, comprising the API service, database, and frontend components, thereby maintaining strict infrastructure parity between testing and deployment environments. Such isolation guarantees reproducibility and prevents host-specific dependencies from affecting test outcomes.

B. Test Workflow

The automated workflow follows a sequential, fully scripted procedure:

- Initialization and readiness verification of the PostgreSQL service container.
- Application of Django database migrations to construct the test schema.
- Execution of the test suite inside the API container using the coverage.py framework.
- Generation of multi-format coverage artifacts (XML, HTML, and Markdown).
- Publication of summarized test outcomes to the GitHub Actions job summary and the "Tests" dashboard.
- Archival of all reports as persistent CI artifacts for subsequent inspection and traceability.

Example unit test results:



46 passed, 0 failed and 0 skipped

Report	Passed	Failed	Skipped	Time
infra/api/test-reports/TEST-api.tests.test_auth.AuthTestCase-20251022105442.xml	7✅			3s
infra/api/test-reports/TEST-api.tests.test_discussions.DiscussionTests-20251022105442.xml	7✅			5s
infra/api/test-reports/TEST-api.tests.test_profile_api.ProfileAPITestCase-20251022105442.xml	6✅			3s
infra/api/test-reports/TEST-api.tests.test_profile.ProfileTests-20251022105442.xml	7✅			4s
infra/api/test-reports/TEST-api.tests.test_search.SearchTests-20251022105442.xml	7✅			4s
infra/api/test-reports/TEST-api.tests.test_space_collaborator.SpaceCollaboratorTests-20251022105442.xml	7✅			5s
infra/api/test-reports/TEST-api.tests.test_space.SpaceModelTest-20251022105442.xml	3✅			767ms
infra/api/test-reports/TEST-api.tests.test_tag.TagModelTest-20251022105442.xml	2✅			4ms

✅ infra/api/test-reports/TEST-api.tests.test_auth.AuthTestCase-20251022105442.xml

7 tests were completed in 3s with 7 passed, 0 failed and 0 skipped.

Test suite	Passed	Failed	Skipped	Time
api.tests.test_auth.AuthTestCase-20251022105442	7✅			3s

✅ api.tests.test_auth.AuthTestCase-20251022105442

```
api.tests.test_auth.AuthTestCase
  ✅ test_login_invalid_password
  ✅ test_login_invalid_user
  ✅ test_login_valid_user
  ✅ test_register_existing_user
  ✅ test_register_user
  ✅ test_register_user_below_age_eighteen
  ✅ test_register_user_no_profession
```

Collectively, this architecture establishes a reproducible, transparent, and infrastructure-aware testing mechanism that reinforces software reliability while enabling continuous quality assessment throughout the development and deployment lifecycle.

Unit-Testing Integration within the CI/CD Pipeline (Mobile Application)

The Android continuous-integration subsystem embeds automated unit testing as a core validation mechanism across all pipeline stages.

Each workflow employs the same fundamental testing framework, ensuring that code correctness and stability are continuously assessed from early development through production-release preparation.

A. Mobile Unified Testing Framework

All pipelines execute unit tests under consistent, reproducible environments on GitHub-hosted Ubuntu runners configured with the Android SDK and JDK 17.

The testing process follows a standardized sequence:

- Source Checkout : retrieves the corresponding branch or pull-request commit.
- Gradle Initialization : validates the Gradle wrapper and restores cached dependencies.
- Build Stage : compiles the application in debug mode (assemble…Debug) for the appropriate flavor.
- Test Execution : invokes the Gradle test task (test…DebugUnitTest) to execute logic-level and ViewModel tests in isolation from device hardware.

Result Collection and Publication :

- dorny/test-reporter: summarizes results in the GitHub job output.
- EnricoMi/publish-unit-test-result-action: exposes detailed outcomes on the Tests tab and in pull-request comments.
- Artifact Preservation : generated APKs and test reports are retained as build artifacts for subsequent analysis and traceability.
- This shared structure provides deterministic testing conditions and immediate feedback to developers, reinforcing the project's continuous-quality-assurance principles.

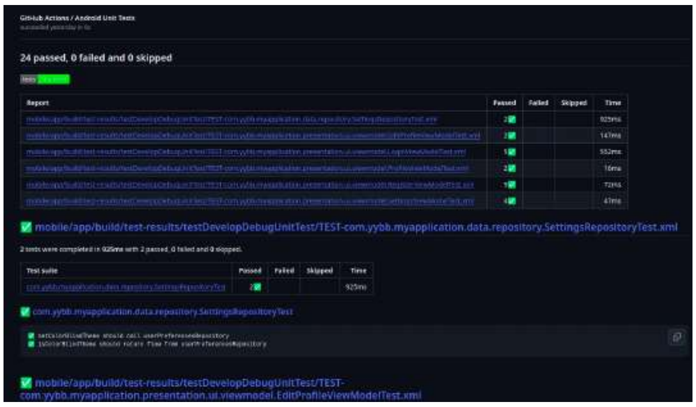
B. Branch-Specific Testing Objectives

Feature/Bugfix/Refactor CI: Executes unit tests on all incremental development branches to detect functional regressions at the earliest possible stage. It validates isolated features prior to integration and ensures that every code contribution satisfies the established behavioral contracts.

Develop CI runs the same Gradle test suite when merging into the develop branch, verifying the combined integrity of multiple features. This acts as an integration-testing checkpoint, where the cumulative effects of concurrent changes are evaluated under the shared development configuration.

Main CI Performs unit testing on the production build flavor (testProductionDebugUnitTest) before code promotion from develop to main. This stage serves as a release-assurance mechanism, confirming that no regressions persist in the final production candidate.

Example unit test results:


GitHub Actions / Android Unit Tests

24 passed, 0 failed and 0 skipped

Report	Passed	Failed	Skipped	Time
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.data.repository.SettingsRepositoryTest.xml	2✓			925ms
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.EditProfileViewModelTest.xml	2✓			147ms
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.LoginViewModelTest.xml	5✓			552ms
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.ProfileViewModelTest.xml	2✓			16ms
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.[illegible]	9✓			72ms
mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.SettingsViewModelTest.xml	4✓			47ms

✓ mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.data.repository.SettingsRepositoryTest.xml

2 tests were completed in 925ms with 2 passed, 0 failed and 0 skipped.

Test suite	Passed	Failed	Skipped	Time
com.yybb.myapplication.data.repository.SettingsRepositoryTest	2✓			925ms

✓ com.yybb.myapplication.data.repository.SettingsRepositoryTest

```
✓ setColorBlindTheme should call userPreferencesRepository
✓ isColorBlindTheme should return flow from userPreferencesRepository
```

✓ mobile/app/build/test-results/testDevelopDebugUnitTest/TEST-com.yybb.myapplication.presentation.ui.viewmodel.EditProfileViewModelTest.xml

Individual Contributions

Yasemin Tangül

Contribution	Related Issue
Organizing folder structure for maintainable repository Creating issue template for feature issues	Issue #2
Building CI pipeline for web application and mobile application. Versioning for mobile application .apk Unit test integration to CI pipeline for both mobile and web application	Issue #1

Research on W3C accessibility standards	Issue #5
Research on collaborative tools for documentation	Issue #8
Review & Update SRS based on the new requirements	Issue #10
Building CD Pipeline for Web Application Creating deployment environment for both test and production(EC2) Preparing deployment servers for web application Unit test report generation CD Pipeline integration to deployment environment Creating secret variables on Github	Issue #18
Adding milestones and issues according to the planning	Issue #27

Gamze Güneri

Contribution	**Related Issue**
Establish branch policy	#Issue 4
Research W3C accessibility standards	#Issue 5
Research mobile app frameworks	#Issue 6
Update Project Wiki Structure with Navigation Links and Additional Pages	#Issue 9
Review and Update SRS with New Features	#Issue 10
Create Figma Project and Design System	#Issue 12
Design Mobile UI Mockups	#Issue 13
Flow Charts and User Stories will be Documented	#Issue 17
Initialize Back-Office Web Frontend Implementation	#Issue 48
Creating UML diagrams	#Issue 50

Yusuf Bayam

Contribution	Related Issue
Setup Github project & Repository	Repository on Github
Establish branch policy	Issue #4
Research W3C accessibility standards	Issue #5
Research mobile app frameworks	Issue #6
Setup Slack Workspace and Communication Channels	Issue #7
Review and Update SRS with New Features	Issue #10
Setup initial mobile application project	Issue #16
Refine & Update Requirements document according to Customer Meeting	Issue #23
User profile screen with unit tests for mobile application	Issue #28
App theme for color-blind support implementation for android in settings screen	Issue #29
Connect Login, Register and Profile Screens to Backend (Android)	Issue #58
Creating UML diagrams	Issue #50
Setup Development and Production Environments for Android	Issue #64

Esra Nur Özüm

Contribution	Related Issue
Creating Issue Tags	Issue #3
Research W3C accessibility standards	Issue #5
Research mobile app frameworks	Issue #6
Review and update the SRS with new features	Issue #10
Designing Mockup UIs for Mobile	Issue #13

Building the initial phase of the mobile application	Issue #16
Implementation of the registration screen on mobile	Issue #24
Implementation of the login screen on mobile	Issue #25
Addding unit tests for login and registration screens	Issue #32
Creating UML diagrams	Issue #50
Creating a milestone report template	Issue #51
Creating a space graph representation on mobile	Issue #57

Batuhan Cömert

Contribution	**Related Issue**
Create issue tags	Issue #3
Research W3C accessibility standards	Issue #5
Research regarding colors for color blind people	Issue #20
Color theme implementation for web	Issue #30
Review of Web Application Unit Tests	Issue #31
Fixing Wikidata Search API Bug	Issue #45
Refactor Color Palette Implementation	Issue #60
Review SRS According to New Requirements	Issue #10

CM1 Team Retrospective

What went well?

- Meeting planning was decided nicely.

    - Communication among the team members was generally efficient and effective. Attendance at regular meetings was consistently high, and prompt actions were taken on the topics discussed. The frequency of the weekly meetings was well planned, allowing sufficient time afterward for development and task management activities.

- Making the project up-and-running was very fast.
    - The project was set up and launched quickly and efficiently, along with its overall structure. The dockerization and deployment stages were handled with the same level of efficiency and were completed without significant time loss.

- We were able to deliver everything included into the CM1
    - All the tasks determined for Customer Milestone 1 during the planning phase were successfully completed and documented. In addition to the timeframe allocated for the customer milestone, the internally defined milestones enabled efficient distribution and completion of task management activities.

- Meeting notes, templates, wikis we prepared well
    - The importance of documentation in the team project was recognized, and with the help of documentation tools, records were kept for all tasks performed. Tools such as Figma, Wiki, and Notion were utilized to ensure that the documentation remained organized and easily manageable.

- Planning (including new features) was good.
    - Given that the project had two significant customer milestones and would undergo an approximately three-month development period, a detailed plan was prepared by the team members accordingly. In addition to the customer milestones, internal milestones were introduced to create a realistic development environment and to maintain a balanced workload. Tasks for improving existing features and adding new ones were taken into account, and they were distributed as evenly as possible across the defined milestones.

What needs improvement?

- We can start presentation planning much more early.
- We need to spend much more time to the design
    - When deciding on the designs, which constitute a guiding and essential part of the project, making decisions collaboratively as a team can help prevent potential issues in later stages and facilitate reaching effective solutions through brainstorming. Within this framework, it can be stated that the team can spend more time on the designs, discussing their pros and cons in greater detail.
- We need to prioritize the issues.

    - We could assign priority levels to the tasks identified during the planning phase and selected individually during development more carefully. This would be highly beneficial as the project grows further.
- We need to make sure the unit tests are covering the code.
    - Unit tests, which are a crucial part of the project, are highly effective and helpful in identifying issues or determining whether a new change impacts the entire system. In this context, we should aim to maintain the coverage of the written unit tests at the highest possible level. As the project grows, these tests can help reduce our workload to detect any broken points in the project.

Action items

- [ ] Start preparing the presentation flow for the customer milestones before 1 week.
- [ ] Supplement design related discussions with graphical tools.
- [ ] Check code coverage while writing new scripts (Add new unit tests if needed)