Master's Project in Web Application Development

# Engineers' Platform

Full Stack Web Application Report

**By:**

Heba Mansour

**Academic group:**

RIM-140930

**Supervisor:**

Dr. Saif Mujahid Abdullah Khael

**2024-2025**

## Abstract

**Engineers' Platform** is a web-based application designed to showcase engineering projects across a wide range of specializations. The platform acts as a bridge between engineers who publish their own projects, and investors who seek qualified professionals to implement their ideas or similar engineering solutions.

One of the primary goals of this platform is to support university students by providing access to a library of real-world project ideas, helping them stay informed about current trends and tools in the labor market. This exposure enhances their practical understanding and skill development.

The platform enables engineers to upload and present their projects to a wider audience, effectively promoting their expertise and attracting potential collaborations. To protect intellectual property, engineers can specify licensing terms when publishing their projects.

For investors, the platform provides the ability to search for existing projects or related ideas. It also allows them to review an engineer's previous work to assess their capabilities and relevant experience before initiating contact or sending an invitation.

To ensure smooth operation and content quality, the platform includes an admin role with comprehensive control over the system. Admins can manage users (engineers and investors), oversee projects, posts, comments, and invitations, and maintain platform integrity by moderating content. Admins can also add or remove other admin accounts as needed.

The platform is fully responsive and optimized for various screen sizes, including desktop, tablet, and mobile. It is developed using **React.js** for the frontend and **Django** for the backend, ensuring a modern, robust, and scalable web application.

# Table of contents

## Introduction

Websites and web applications have become essential tools for global communication and information sharing across all fields. Today, engineers and project owners can easily publish their work and reach a wide audience, especially when shared through platforms designed for that purpose.

**Engineers' Platform** serves as a dedicated space for showcasing engineering projects across various specializations. It enables engineers to add and present their work, while also allowing them to explore and learn from other published projects on the platform.

Investors can browse all available projects, search and filter results based on specific categories or disciplines, and connect with engineers by sending invitations for collaboration. Additionally, investors can publish posts to announce project ideas or seek engineers for specific tasks.

The platform also serves as a valuable resource for university students by offering a collection of project ideas that can inspire their graduation work. It helps them stay informed about the latest developments, tools, and techniques in the engineering job market.

## Project Goals

The goal of the project is to create a complete and comprehensive platform that features a wide range of engineering projects across various categories and disciplines. It provides engineers with a dedicated and organized space to showcase their ideas, promote their expertise, and share their work with potential collaborators and clients. The platform also supports students and investors by giving them access to innovative project ideas and developments in the engineering field.

In addition to enabling engineers to publish their projects, the platform allows investors to post their own ideas or service requests in case they do not find what they are looking for on the platform. This helps connect demand with expertise, ultimately bridging the gap between talent and opportunity in the engineering domain.

To ensure the platform remains well-managed and secure, an admin role has been introduced. Admins have full control over users and content, they can manage (add, edit, or delete) users, projects, posts, invitations, and comments. Admins can also search and browse all content, as

well as add or remove other admin accounts. This ensures the quality, reliability, and integrity of the platform's operations.

## Business Use Case

Many skilled engineers, especially students and early-career professionals, struggle to promote their work or connect with stakeholders who can help realize their ideas. Investors, on the other hand, often seek engineering talent for project execution. This platform serves as a two-way marketplace where:

- Engineers can post projects, receive feedback, and attract investor attention.
- Investors can propose ideas and directly invite engineers to collaborate.

## Problem Statement

Current platforms like GitHub or Behance cater to either developers or designers but lack a specialized, engineering-focused hub that combines project showcasing, investor engagement, and professional networking. **Engineers' Platform** addresses this niche by creating a centralized and interactive digital space.

## System Requirements

### 1. Functional Requirements

These requirements describe what the system should do based on user roles and use cases.

**Engineer Functional Requirements**

- Register and log in to the system as an engineer.
- Reset password
- View and update personal profile details.
- Post a new project.
- Edit or delete an existing project.
- View projects details.

- Comment on posts (add or delete).
- Browse engineering projects and investor posts.
- View profiles of other users (engineers and investors).
- Search for projects and posts using keyword or category.
- View invitations received from investors.

**Investor Functional Requirements**

- Register and log in to the system as an investor.
- Reset password
- View and edit personal profile details.
- Publish a post describing a project idea.
- Edit or delete their posts.
- Browse engineers' projects and posts by other investors.
- Comment on posts (add or delete).
- View profiles of other users.
- Search for engineering projects and investor posts.
- Send invitations to engineers to collaborate.

**Admin Functional Requirements**

- Log in as an administrator.
- Log out.
- Reset password.
- Brows posts and projects.
- Add, update, and delete users (engineers or investors), projects, posts and invitations
- Comments (including user comments)
- Add or remove other admins.
- Search through projects and posts.

## 2. Non-Functional Requirements

These specify how the system performs rather than what it does.

**Performance**

- The platform must support multiple simultaneous users with minimal latency.
- Backend should handle up to 1,000+ active users with acceptable latency.
- Most API responses should complete in under 500ms under normal usage.

**Scalability**

- Backend and database should support modular expansion (e.g., messaging, analytics, admin dashboard enhancements).
- Admin operations should work efficiently across growing datasets.

**Availability**

- Platform should maintain 99.9% uptime.
- Admin actions should be reliably logged for accountability.

**Usability**

- The UI must be responsive across all devices (desktop, tablet, mobile).
- Admin interface should be intuitive for content moderation and user management.

**Security**

- Role-based access control (RBAC): Admins, engineers, and investors have distinct permissions.
- All communications between frontend and backend must be encrypted via HTTPS.
- User input must be validated on both client and server sides to avoid injection and cross-site attacks.

**Maintainability**

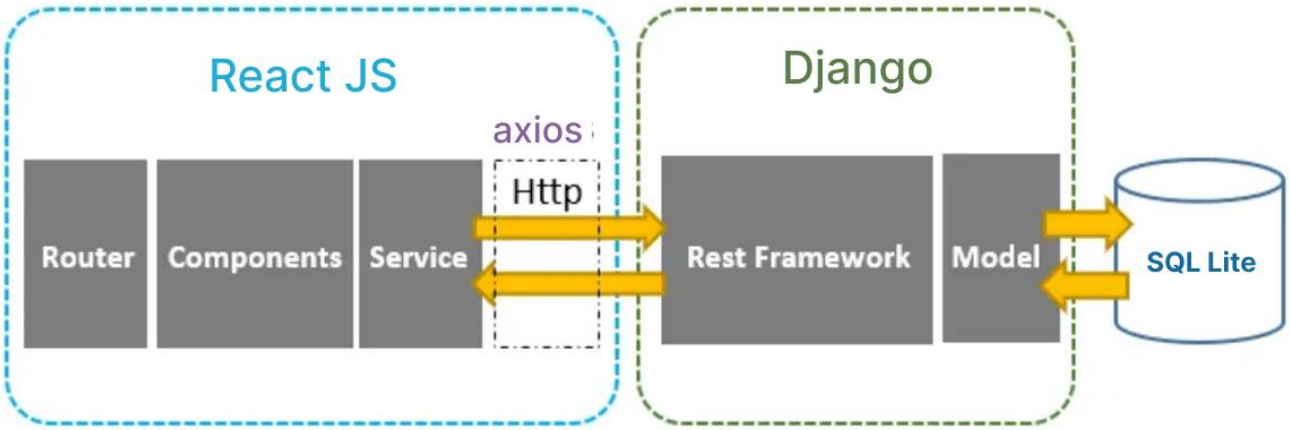
- Codebase must be organized using modular, reusable components.
- APIs and logic should be documented and commented for future developer onboarding.
- Admin actions are centralized for ease of auditing and future expansion.

**Localization (Planned/Future)**

- The system should be adaptable for multi-language support if required in future iterations.

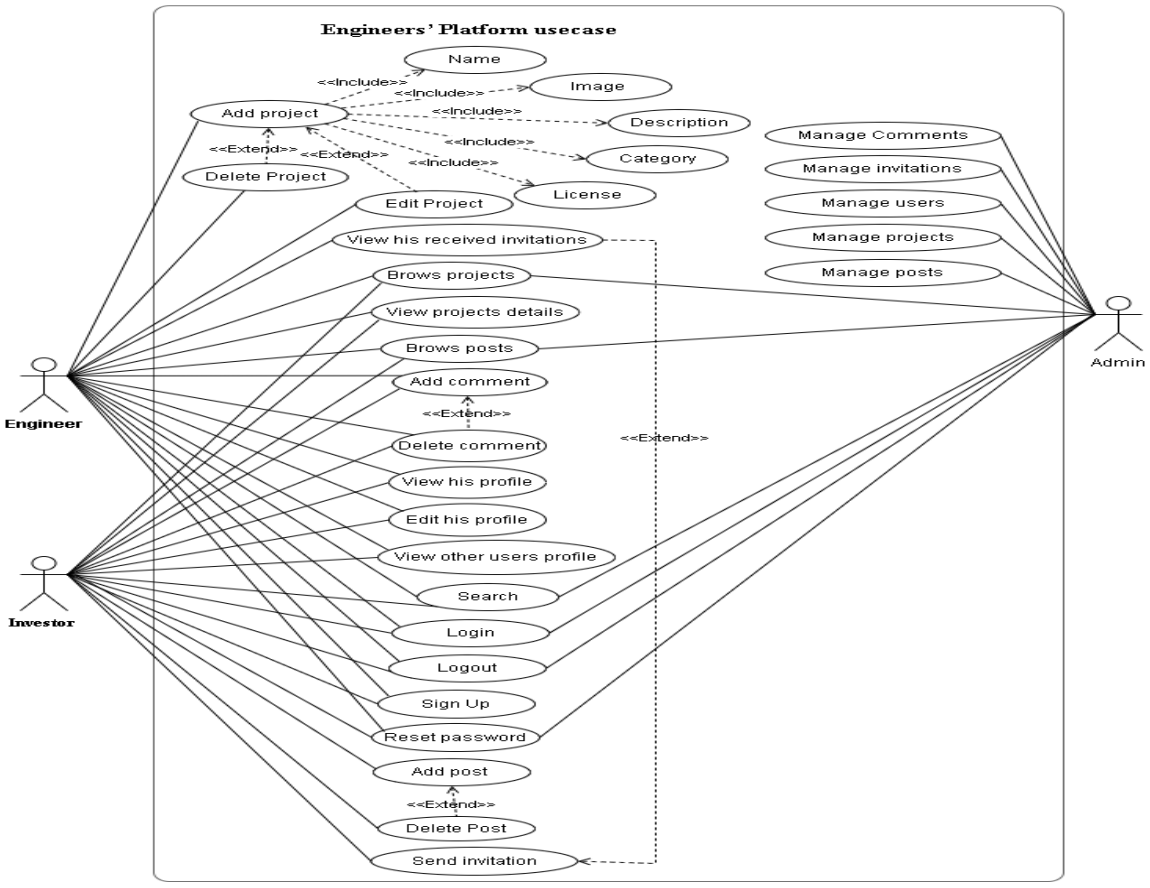
## System architecture

The diagram below represents the system architecture of the Engineers Platform, which expresses the environment used for both front-end and back-end programming and the database.



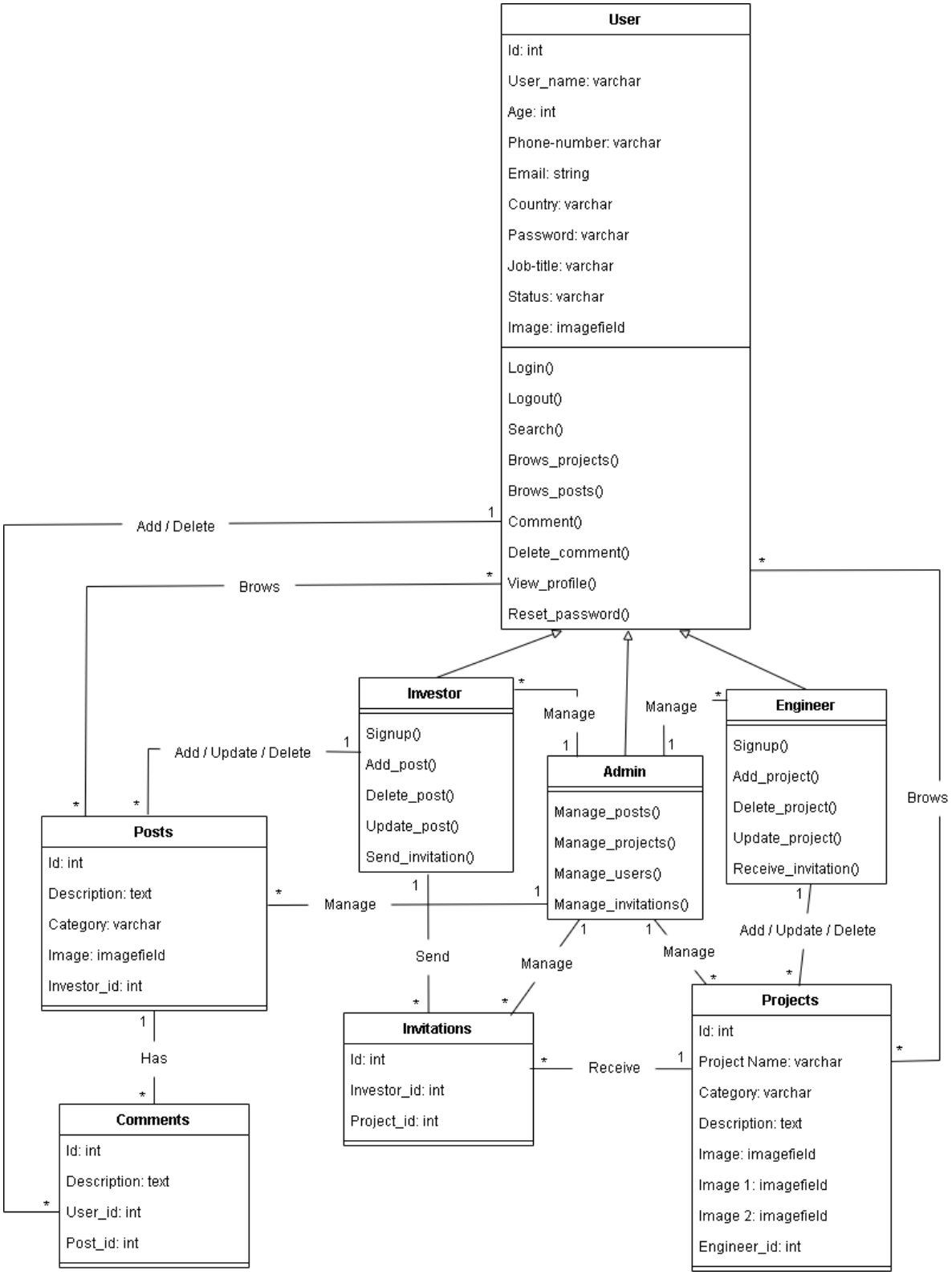

## UML Diagrams - Use case diagram

The diagram below shows the use case diagram for the Engineers Platform, where there are 3 types of users: Engineer, Investor, and Admin, and each user has his own permissions.



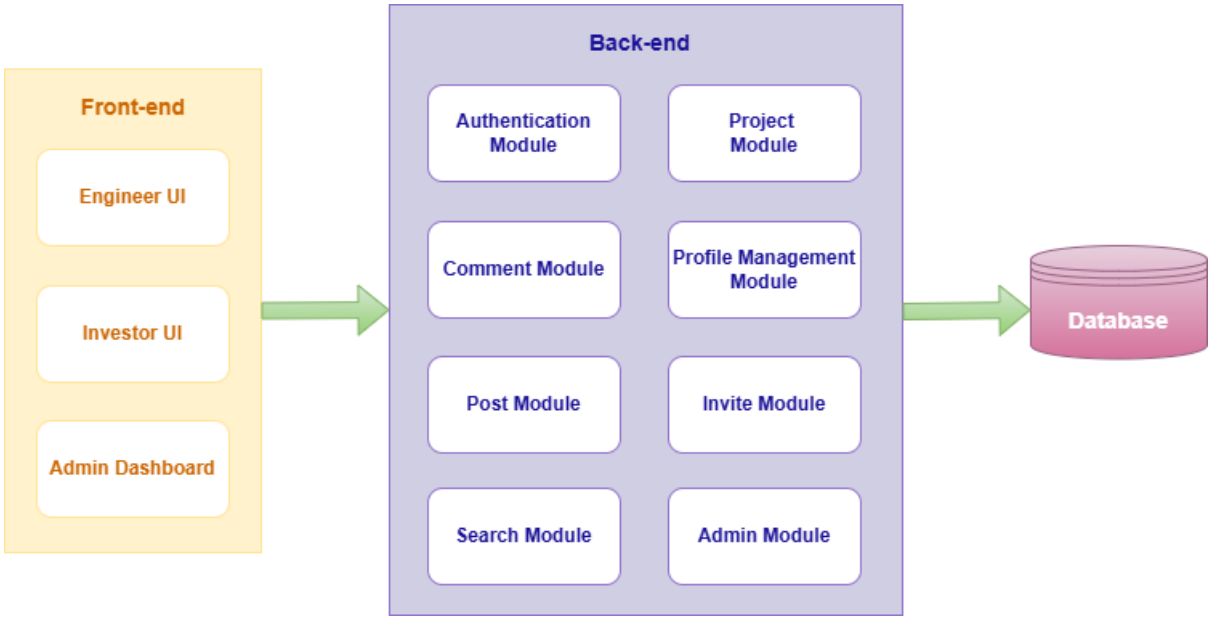
## Class Diagram

The diagram below shows the class diagram of the Engineers Platform, which is the closest representation of the database, where each class represents a table in the database.



## Block Diagram

The diagram below shows the block diagram of the Engineers Platform, which represents the front-end and back-end components with the database and how data is transferred between them in the form of arrows.




## Key Features

1. **User Authentication and Profile Management**

   Users (engineers, investors, admins) can securely register, log in, and manage personal profiles. Authentication is handled using **Rest_framework.authtoken**.

2. **Profile Management**

   All users can view and edit their own profiles. Users can also view other users' profiles based on their role.

3. **Project and Post Management (CRUD)**

   a. Engineers can add, edit, and delete their own projects.

   b. Investors can add, edit, and delete their own posts (project ideas).

   c. Admins can manage (add/edit/delete) **any** project or post.

4. **Commenting System**
    a. Engineers and investors can add and delete comments on posts.
    b. Admins can add or delete **any** comment.
5. **Invitation System**
    a. Investors can send invitations to engineers.
    b. Engineers can view received invitations.
    c. Admins can manage invitations.
6. **Search Functionality**

    All users (including admin) can search for posts and projects by keywords or categories using filter parameters.
7. **Admin Control Panel**
    a. Admins can create, edit, and delete users (engineer/investor).
    b. Admins can promote or delete other admin accounts.
    c. Admins have full control over all content (projects, posts, comments, invitations).
    d. Admins can monitor user activities and search/browse content site-wide.

## Implementation

- **Frontend**
    - **Framework**: React.js (with React Router for navigation)
    - **Languages**: JavaScript
- **Backend**
    - **Framework**: Django
    - **Language**: Python
    - **Database**: SQL Lite

- **Support Tool**
  - **Prototype design:** Figma
  - **Code editor:** VS code
  - **Web browser:** Chrome
  - **Documentation:** MS office 2019
- **Hosting**: Render.com
- **Version Control**: GitHub

## Key Implementation Notes

❖ **Modular API Design**

Each core entity( User, Project, Post, Comment, Invitation) has its own model, serializer, viewset, and route, ensuring clean and scalable API design.

❖ **Role-Based Access Control (RBAC)**

- Permissions are managed via Django REST Framework using custom permission classes.
- Engineers and investors can only access and modify their own content.
- Admins bypass ownership restrictions and have global access to user and content management features.

❖ **Authentication**

Implemented using **Rest_framework.authtoken** for secure token-based login.

❖ **Search and Filtering**

Search features are powered by django-filter, allowing filtering on project/post attributes like name, category.

❖ **Admin Functions**

Admins interact with content either through:

- DRF API endpoints secured with permission logic.
- Or the Django Admin panel for direct model-level control.

❖ **User Profiles**

Each user can view their profile and others', based on system role. Admins have access to all user data.

# Testing & Deployment

## Testing Summary

This project includes automated unit and integration tests to ensure backend reliability and correct API behavior.

- **Authentication**
  - Signup: Validates user creation, token generation, duplicate handling
  - Login: Verifies credentials, token issuance
  - Password Reset: Token generation and password update via reset flow
- **User & Profile**
  - Profile data retrieval and updates
  - Change password and image upload
- **Project & Post Management**
  - Add/edit/delete projects
  - Add and view posts and comments
  - Validate required fields and relations (author, category)
- **Invitations**
  - Create invitations for engineers
  - Check invitation filtering by user and project
- **Permissions & Validation**
  - Auth middleware enforcement (token required)
  - Field-level validation via serializers

### Running the Tests

To execute tests: **python manage.py test**

## Docker Setup

The project uses Docker and docker-compose: frontend (React), backend (Django), Redis (for caching), and Nginx (for serving the frontend in production).

### Services

- frontend Built with React (Vite), served via Nginx
- backend Django + DRF, served via Gunicorn

- ❖ nginx Reverse proxy and static asset handler
- ❖ redis In-memory cache for optimizing backend

### Deployment

This project is containerized for flexible deployment and is prepared for hosting via both Docker-based and FTP-based environments.

**GitHub Deployment**

https://github.com/heba-queen/engineers-platform.git

## Conclusion

- Integration between Django backend and React frontend requires careful API planning.
- Role-based permission systems need to be granular to prevent unintended access.
- Deployment pipelines with Docker streamline testing and production readiness.
- Real-world feedback from users is essential for UI/UX and feature relevance.

## Future Work

- Implement messaging system between engineers and investors.
- Add rating/review system for projects.
- Integrate real-time notifications.
- Advanced search with AI-based recommendations.

## References


1. Prototype Reference: https://www.figma.com/design/t7VgfcVy3hLhReb5Mf9BDR/Engineers'-Platform?node-id=0-1&p=f&t=YcPYSuGxDOxJ4e9S-0
2. Django REST Framework– https://www.django-rest-framework.org/
3. React.js– https://reactjs.org/
4. Docker Docs – https://docs.docker.com/
5. UML Use Case Diagram Guidelines – Visual Paradigm https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
6. UML Class Diagram Reference – Creately https://creately.com/diagram-type/class-diagram
7. System Block Diagrams in Software Architecture – IBM Developer https://developer.ibm.com/articles/uml-diagrams-software-architecture/
8. Creating System Architecture Diagrams – Draw.io https://drawio-app.com/blog/software-architecture-diagrams/