Lemmi

Software Architecture Document

Version 2.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 25/11/2021 | 1.0 | Define the goal and constraint for the architecture.  Sketch version for logical view. | TNTDat  NVTien  TNMKhoa  MTCVan |
| 27/11/2021 | 1.1 | Complete the introduction.  Update the goal and constraint of the architecture.  Logical view with component description and class diagram. | TNTDat  NVTien  TNMKhoa  MTCVan  NMThu |
| 12/11/2021 | 2.0 | Remake the logical view.  Complete the deployment view.  Complete the implementation view.  Finish UI design. | NVTien  MTCVan  TNTDat  TNMKhoa  NMThu |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 4

4. Logical View 4

4.1 Component: abc 4

5. Deployment 4

6. Implementation View 4

Software Architecture Document

# Introduction

**1.1 Purpose:**

Its purpose is to build a bridge for developers and non-developers to give a clear understanding of software’s modules and components without showing detailed implementation.

**1.2 Scope:**

An architectural overview of Lemmi which is being developed by our team is highlighted with the support of JSON, ReactJS, MVC model.

**1.3 References:**

* Use Case Spec - Login by Lemmi account, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Register, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Edit user’s profile, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Create a post, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Edit a post, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Comment on a post, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Report a post, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Search, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Filter, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Delete comment, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Report comment, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Edit comment, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Reply comment, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Follow user, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Unfollow user, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Report user, Version 2.0, 2021, VNU-HCMUS.
* Use Case Spec - Forgot Password, Version 2.0, 2021, VNU-HCMUS.

**1.4 Overview:**

This document outlines Lemmi’s software architecture including architectural goals and constraints, use-case model, logical and implementation view, and deployment view.

# Architectural Goals and Constraints

* 1. **Architecture Goals**

**-** The system programming language is mainly javascript.

- The website server uses long-time supported middleware whose version is up to 12/2021.

- Source code containers are structured of MVC model for future development.

* 1. **Architecture Constraints**

- The website server is configured by Express.

- The front-end development of the application uses no client-side framework. This means all files are sent and rendered from the server.

- Authentication: the system provides

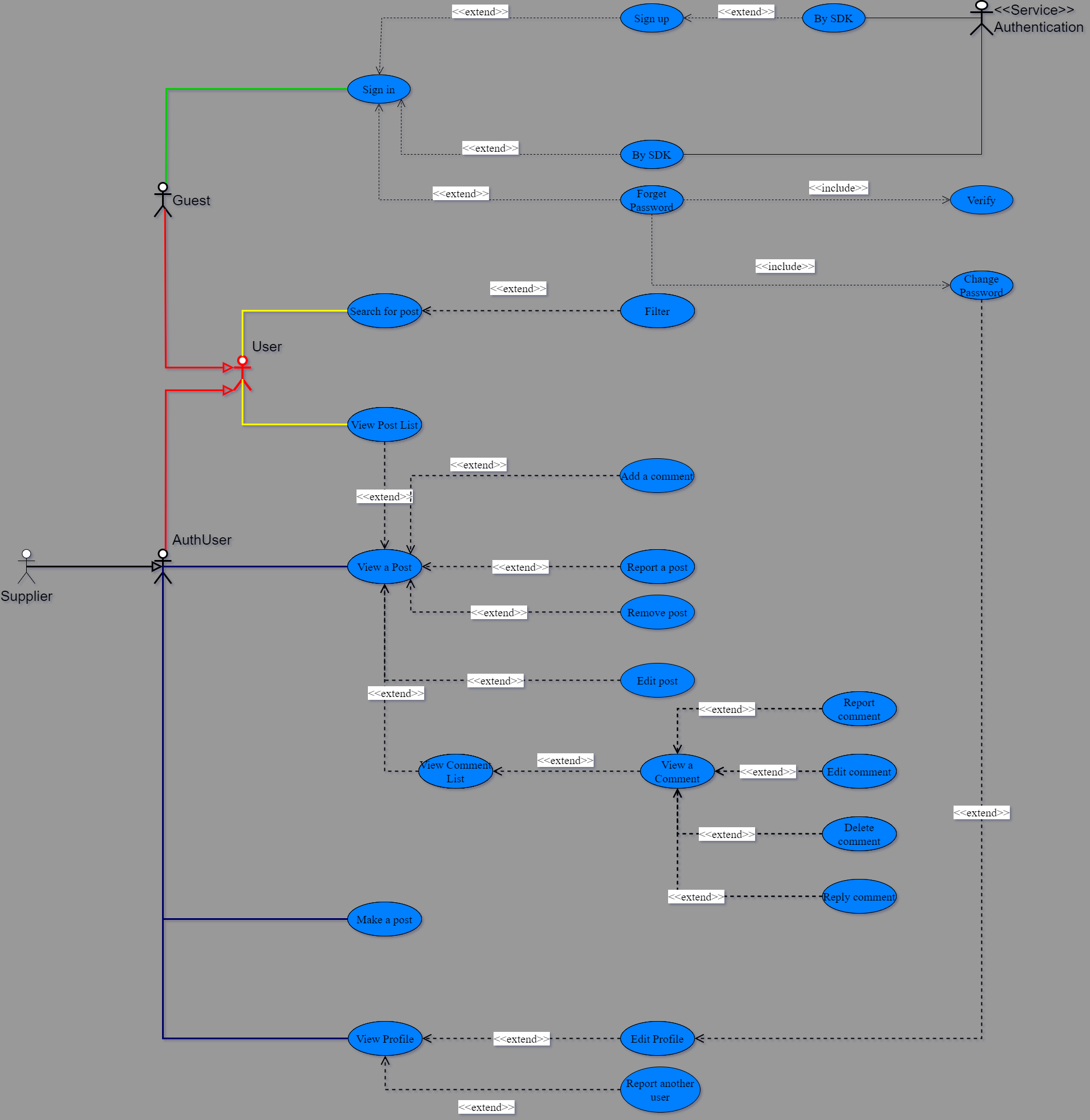
* Login/sign up by traditional account - password
* Login/sign up by Facebook
* Login/sign up by Google

- Authorization: in order to perform any action besides viewing/searching posts, users must be authenticated.

- Query and request data in the authentication process uses secured middleware to avoid attacks.

- Database systems and front-end development must take notice of handling special characters in posts.

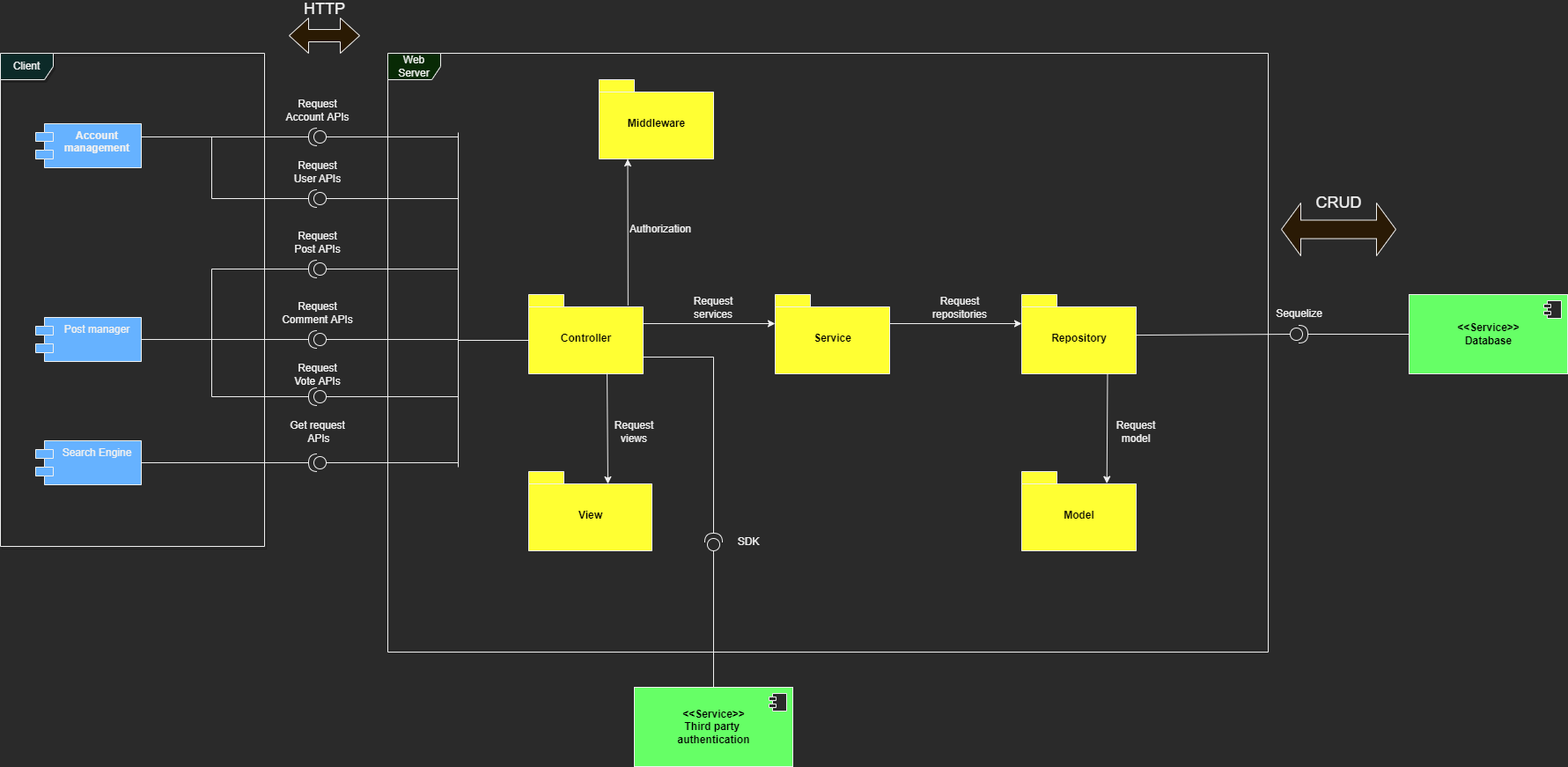
# Use-Case Model



Link: [Use-case.drawio.png](https://drive.google.com/file/d/1xjyIkhckg9Hl7jgcWwHPiijlvMagoL8r/view?usp=sharing)

# Logical View

This section provides an overview of the main components and packages that solve the functional requirements for the system.



Link: [Logical View.drawio.png](https://drive.google.com/file/d/1iq46ZHgHSVvDOs3vNeCks7H2mRmTmR8W/view?usp=sharing)

## Component: Account management

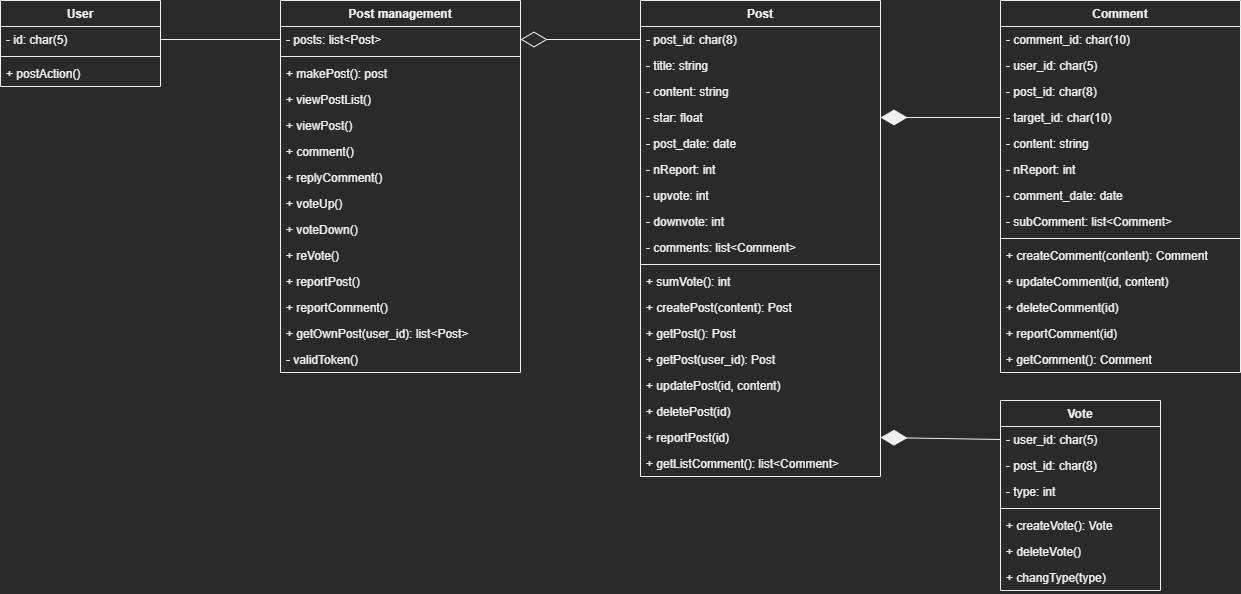
This component focuses on the functional requirements of managing accounts such as login, sign up, show information, reset password, etc.



Link: [Account management.drawio.png](https://drive.google.com/file/d/1wbWfsc_aOTWlVfpefM43vYgU3KP69Vj_/view?usp=sharing)

## Component: Post management

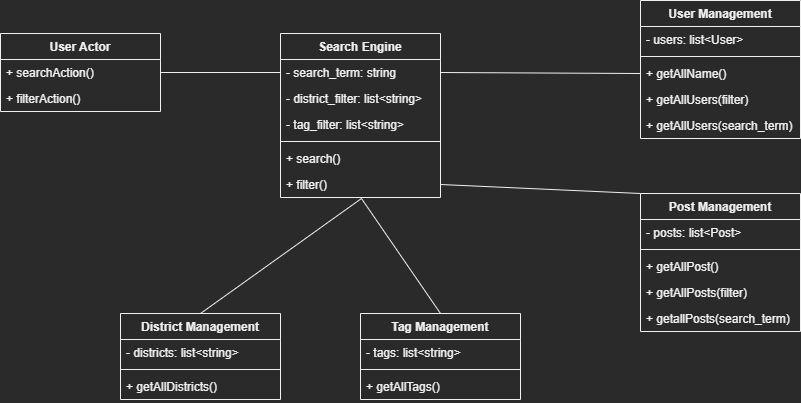
This component illustrates the post’s functionality such as creating a post, getting own posts, etc.



Link: [Post management.drawio.png](https://drive.google.com/file/d/15hVpujulXP9ZUp9-QJgXLGNogQyONLOv/view?usp=sharing)

## Component: Search engine

This section demonstrates the search engine component which provides search and filter functions.



Link: [Search engine.drawio.png](https://drive.google.com/file/d/1hVfYpUctWDfjLBvE-VI1jg-5l-Nye8yK/view?usp=sharing)

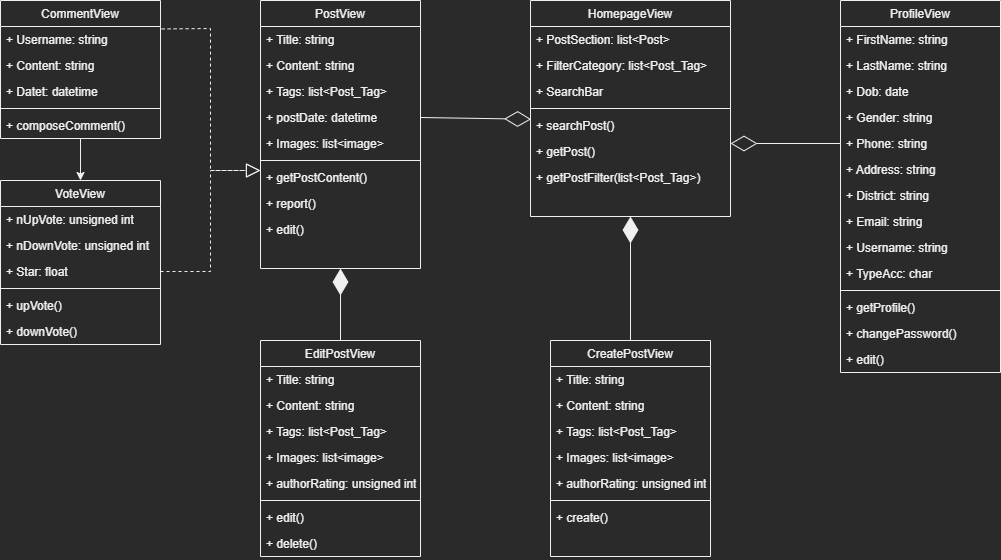
## Package: View

HomepageView includes CreatePostView, ProfileView, PostView.

* In order to get to CreatePostView, users must be in HomepageView.
* From HomepageView, users can navigate to ProfileView or PostView.

PostView has CommentView and VoteView as children and includes EditPostView.

* In order to get to EditPostView, users must be in PostView.

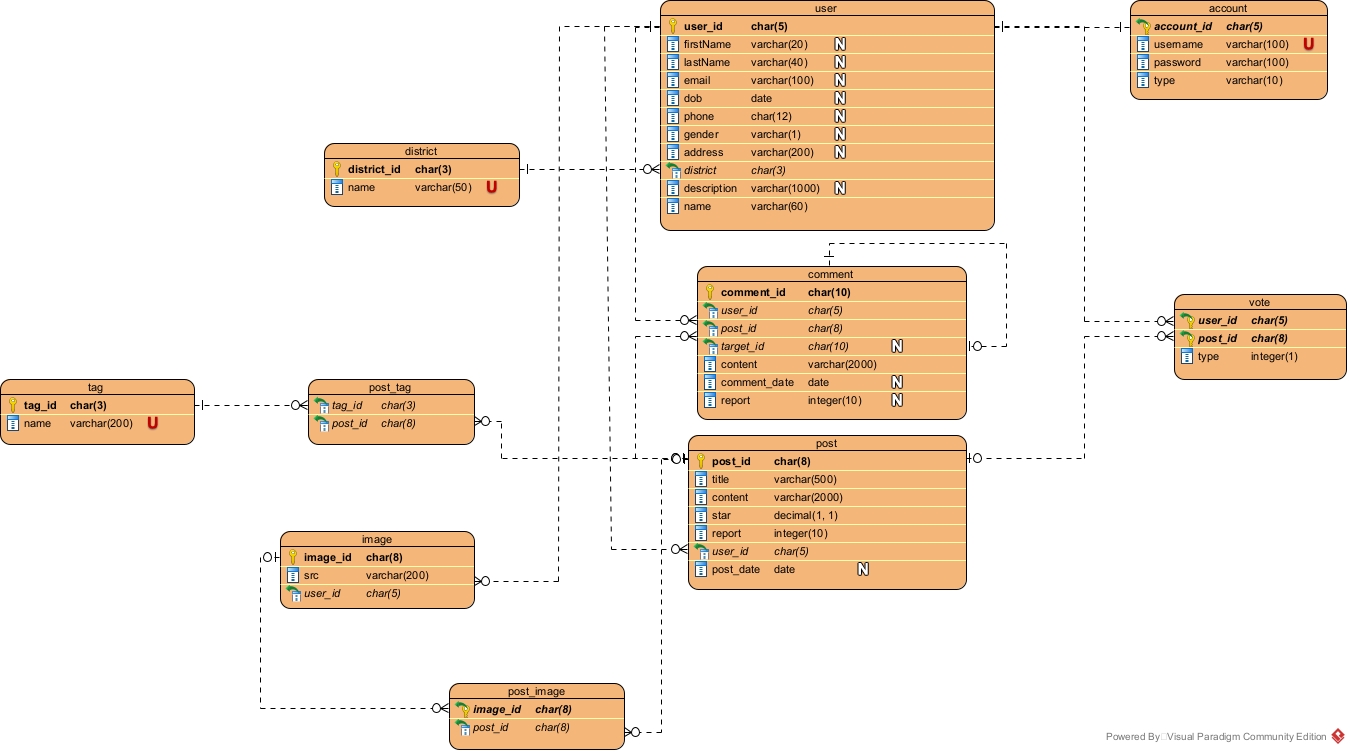


Link: [View.drawio.png](https://drive.google.com/file/d/1K36pxaELjoMMh0hSuBqc84S3NjqWs-e8/view?usp=sharing)

## 

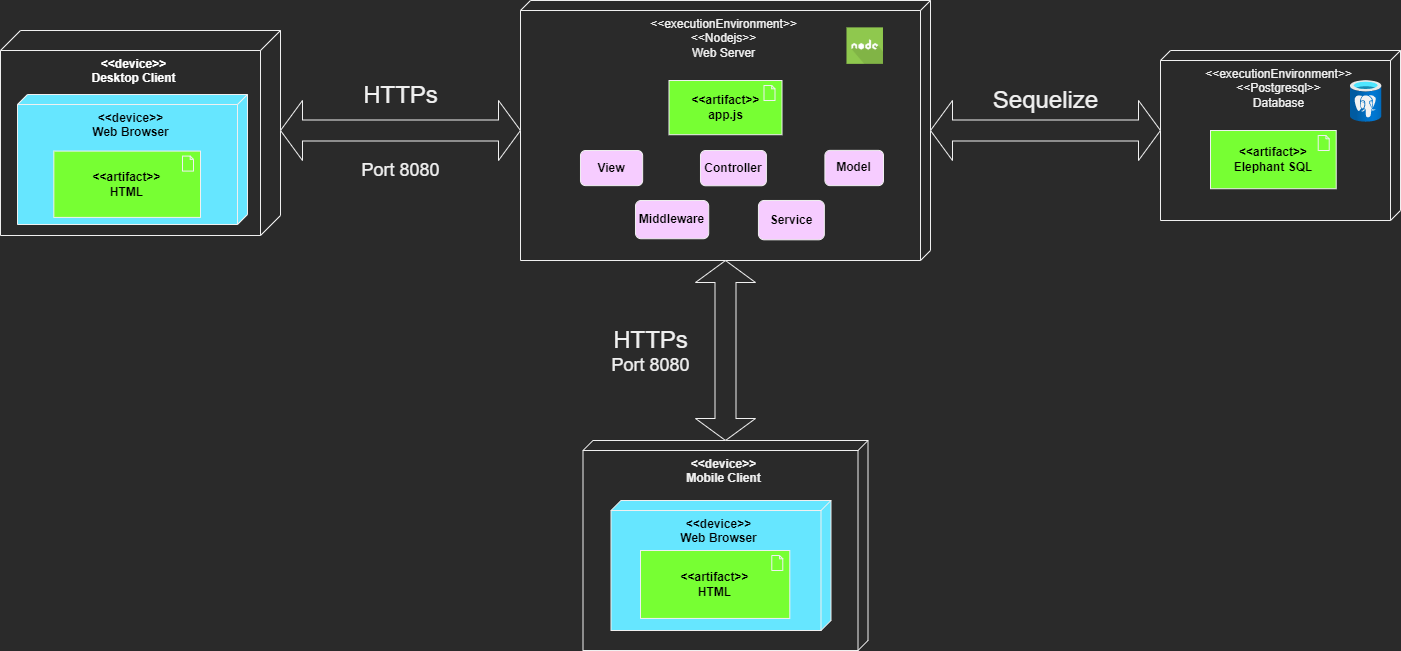
## Database Entity Relation (Package model):

This diagram shows the structure of the entities in the database and the relation between them. It also represents the data model in the package model that use in the website server.



Link: [ER.jpg](https://drive.google.com/file/d/1VDCeD7p3wpykv6pYdx7NUCT-bM5Ubk4Q/view?usp=sharing)

# Deployment

This section illustrates the organization of the software which includes subsystems, execution environment, device, framework, etc.  
 

Link: [Deployment View.drawio.png](https://drive.google.com/file/d/1qsKGixG2X1Q1mWC4ToTUtBdT8TkPzzId/view?usp=sharing)

# Implementation View

## Overview

This section illustrates how the folder structure is implemented.

All folders are contained in the src/Lemmi folder. However, there are three main groups of folders.

## 

Server folders:

* Route: define the main routes of main APIs.
* Controller: define APIs method for each main route.
* Config: set up files (connect database, error).
* Middleware: authorization (generate token, valid token).

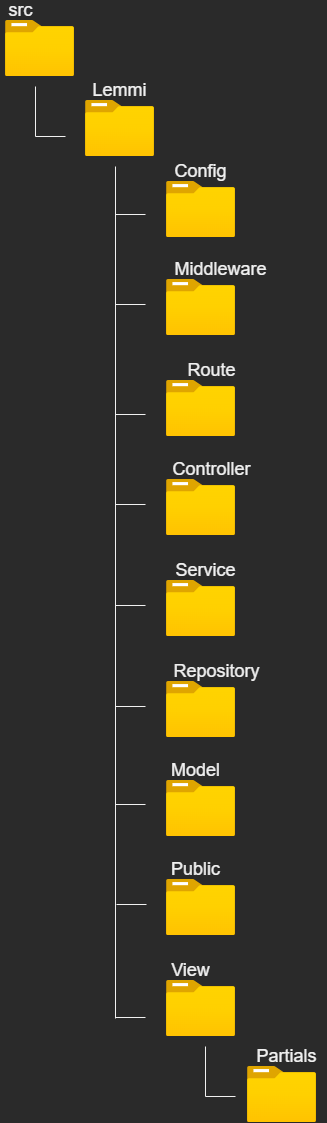
Layout folders:

* Public: contain Javascript, CSS files.
* View: contains HTML-EJS templates.
* Partials: contains reusable components.

Data service folders:

* Service: provide APIs service that is related to data.
* Repository: provide CRUD functions to manipulate the database.
* Model: provide object models for each data queried from the database.

## Demonstration



Link: [Implementation View.drawio.png](https://drive.google.com/file/d/1jsrOU2miSbaquNltjpYlcSJZWTk4m-nv/view?usp=sharing)