AMAM

Software Architecture Document

Version 1.0

*[The texts in blue are guidance for filling in the information for each section. Remove everything in blue when writing the document.*

*This template is a simplified version of the Software Architecture Document from the RUP model. ]*

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 6/12/2022 | 1.0 | First version | Từ Khai Hoài, Lâm Hiền Toàn, Nguyễn Việt Hùng, Thái Võ Đức Trọng, Nguyễn Đình Quốc |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**1. Introduction**](#_heading=h.gjdgxs) **4**

[**2. Architectural Goals and Constraints**](#_heading=h.30j0zll) **4**

[1. Design Principles](#_heading=h.deaoxrjcir3t) 4

[2. Resilience and Scalability](#_heading=h.p7mnv3dj7wlt) 4

[3. Usability](#_heading=h.xv7ak1kbjwy2) 4

[4. Security](#_heading=h.l41fqn8dzmwp) 4

[5. Performance](#_heading=h.wim9d2bpsnxs) 4

[**3. Use-Case Model**](#_heading=h.1fob9te) **5**

[**4. Logical View**](#_heading=h.i9opvb2yt6e2) **6**

[4.1 Component: Views - Frontend](#_heading=h.tyjcwt) 6

[4.2 Component: Components - Frontend](#_heading=h.nwpa1tdn7ab2) 6

[4.3 Component: Controllers - Backend](#_heading=h.hklv5ober391) 6

[4.4 Component: Model - Backend](#_heading=h.p2n8agxmzcyv) 7

[4.5 Component: Views - Backend](#_heading=h.m4b204i6t3b2) 7

[4.6 Component: MongoDB](#_heading=h.fj116acw779g) 7

[4.7 Component: Firebase](#_heading=h.zgyjv7p4ymyx) 7

[**5. Deployment**](#_heading=h.3dy6vkm) **7**

[**6. Implementation View**](#_heading=h.1t3h5sf) **7**

Software Architecture Document

# Introduction

The **Software Architecture Document** provides an overview of the architecture of the AMAM project. This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

# Architectural Goals and Constraints

This section describes the software requirements and objectives that significantly impact the architecture.

## Design Principles

The architecture of AskMeAnswerMe encourage the application of the SOLID design principles, which are intended to make the software design more understandable, flexible and maintainable. The broad goal of these design principles is to reduce dependencies to ensure changes can be made to one area of the solution without impacting others.

## Resilience and Scalability

AskMeAnswerMe is a Q&A website for all people in the world. For this reason, it is very important that the platform supports high availability. The strategies that will be employed to accomplish this architectural goal include scalability (ensuring that the platform can scale to meet the demands of the users), resilience to outside attacks (the platform should be resilient towards attacks from third parties with intent to disrupt the services of the platform, like DDoS attacks), and support for having no single point of failure (to ensure that no single component can cause the whole platform to fail if it fails).

## Usability

AskMeAnswerMe system features should be considered in a way to be easy to use. Users just need to log in, and use website features. Website will also contain user manual to help users. The language of site should be international because the users are from different countries.

## Security

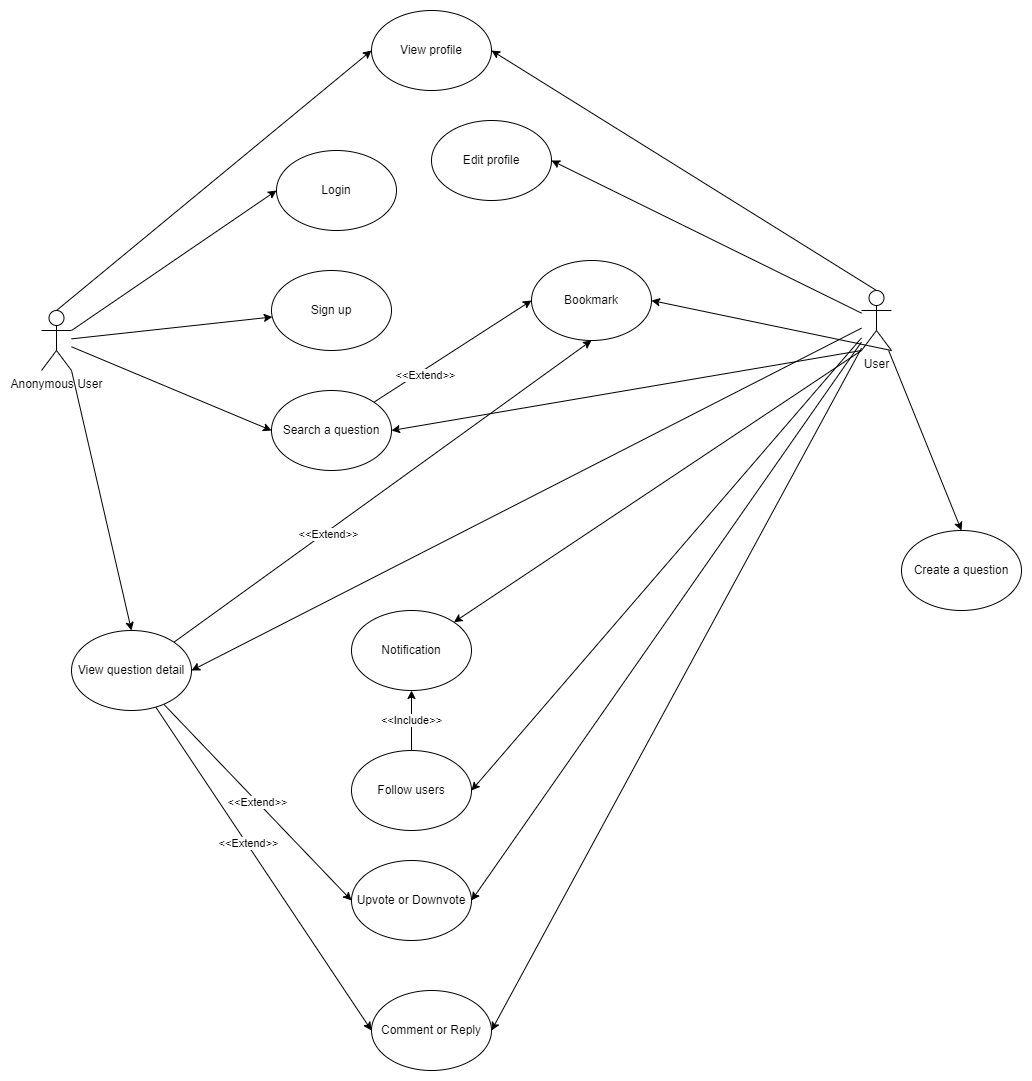
The application must implement basic security behaviors:

* Authentication: Login using at least a user name and a password
* Authorization: anonymous user should not be able to do some actions like comment or post questions, or users cannot edit or change the questions or profile of another user.

## Performance

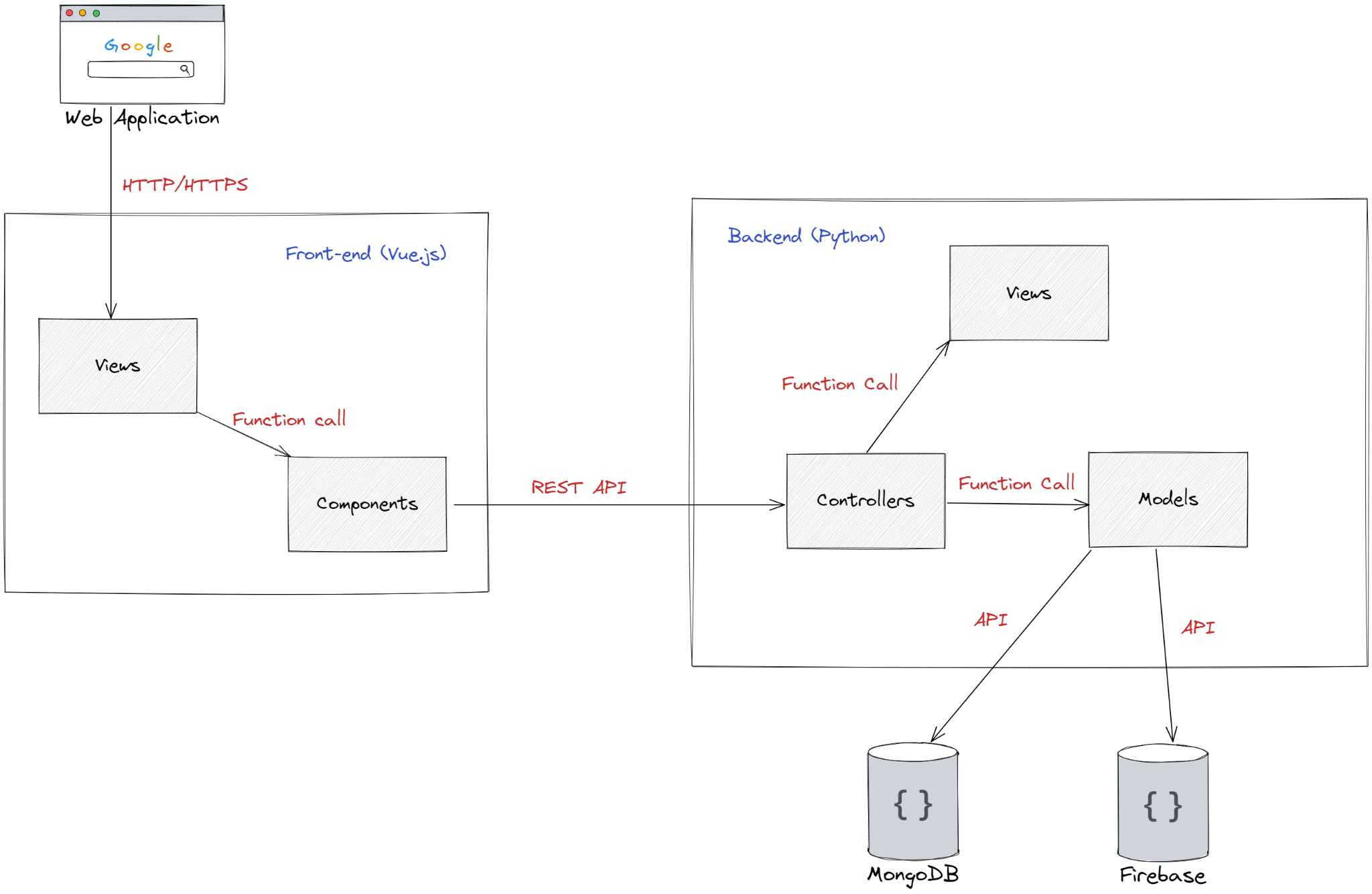
The application should perform fast with low latency and should be able to perform normally under high traffic with huge number of users and requests.

# Use-Case Model

**

# 

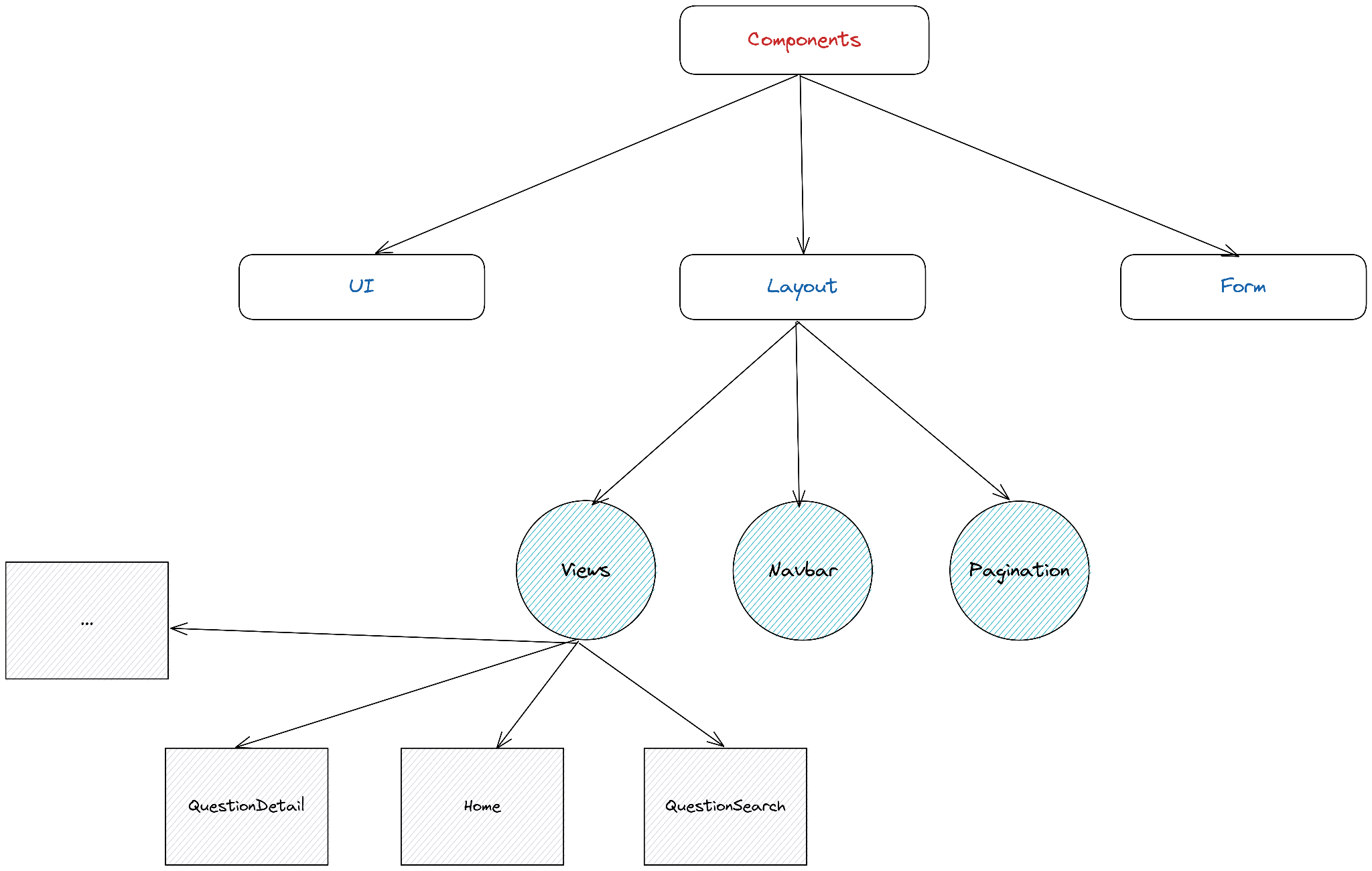
# Logical View



## Component: Views - Frontend

Views in front-end define the UI layout and content to display in the web application for each page. Each view contains some UI, layout, and logic Vue components. Those components are fetched through the function call in the view component.

## Component: Components - Frontend



Components in front-end can be:

* UI components that define some common elements like button, tag, …
* Layout components that define the page layout for each separate view
* Logic/Content components that are used to fetch the content for each page/view from the back-end and display the content in the page.

Components interact with each other through function calls. For the components that handle data fetching and authentication, it will be interact with the back-end components through REST API calls.

## Component: Controllers - Backend

The controllers are responsible for handling the API requests from the front-end components, as well as act as a link between views and models through function calls.

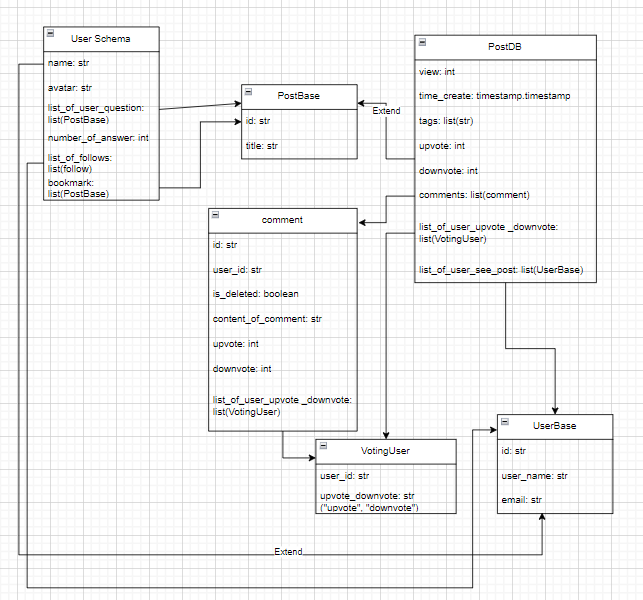
## Component: Model - Backend

The model in the backend side is responsible for how the data from the requirements will be represented, sent to MongoDB for storing, and sent to FireBase for the authentication, signing in, signing up process. Model will get data from the client via controllers and then will be sent to MongoDB or FireBase via APIs provided in those technologies.

## Component: Views - Backend

The views are responsible for how the data from the model will be represented and sent to the front-end. Views will get the data from models through controllers, after that it will decide which field/attribute will be used in the response.

This is the class diagram for this:



## Component: MongoDB

MongoDB is the document-based database or NoSQL database used to store data from models as documents via controllers. MongoDB can be scaled horizontally thanks to the use of a distributed system behind the scenes.

## Component: Firebase

Firebase is used to authenticate with email and password. Firebase supports the authentication, signing in and signing up of users’ accounts from the clients with a set of abundant APIs.

# Deployment

*[Leave this section blank for PA3.]*

# Implementation View

*[Leave this section blank for PA3.]*