

# Alumni Management and Engagement System

Team 12

Mansi Singh (20162034)

Prateek Saxena (200902016)

Natasha Sehgal (201401237)

Abhishek Shrivastava (20172098)

Vatsal Shah (201401115)





# Overview

- Problem Statement
- Assumptions
- Stakeholders
- Requirements
- Design
- Tech Stack
- Demo



# Problem Statement

## Alumni Management and Engagement System

- Enable alumni to stay in contact with each other and institute, and vice versa
- Contact with and within alumni can be used to explore business connections and to gain references or insights in a new field
- An online application that can be accessed throughout the institute and outside.



# Assumptions

- Users have an internet connection
- Users have a mobile/desktop computing device with a web browser
- Users have prior knowledge and experience of using and navigating through a web portal
- Users are alumni of the institute



# Stakeholders

## Direct stakeholders

- Institute
- Alumni - A pass out of the institute, interacting with peers and institute
- Administrators - Monitor content, control functionality and data through platform
- Developers - Platform maintenance, Scaling

## Indirect Stakeholders

- Companies/Firms - With job postings on the portal



# Functional Requirements

An **alumni** can

- Register and login
- View/post job referrals and posts
- Search for users
- Block user
- Report posts for offensive content
- Subscribe to events posted by the admin.
- Provide feedback

An **admin** can

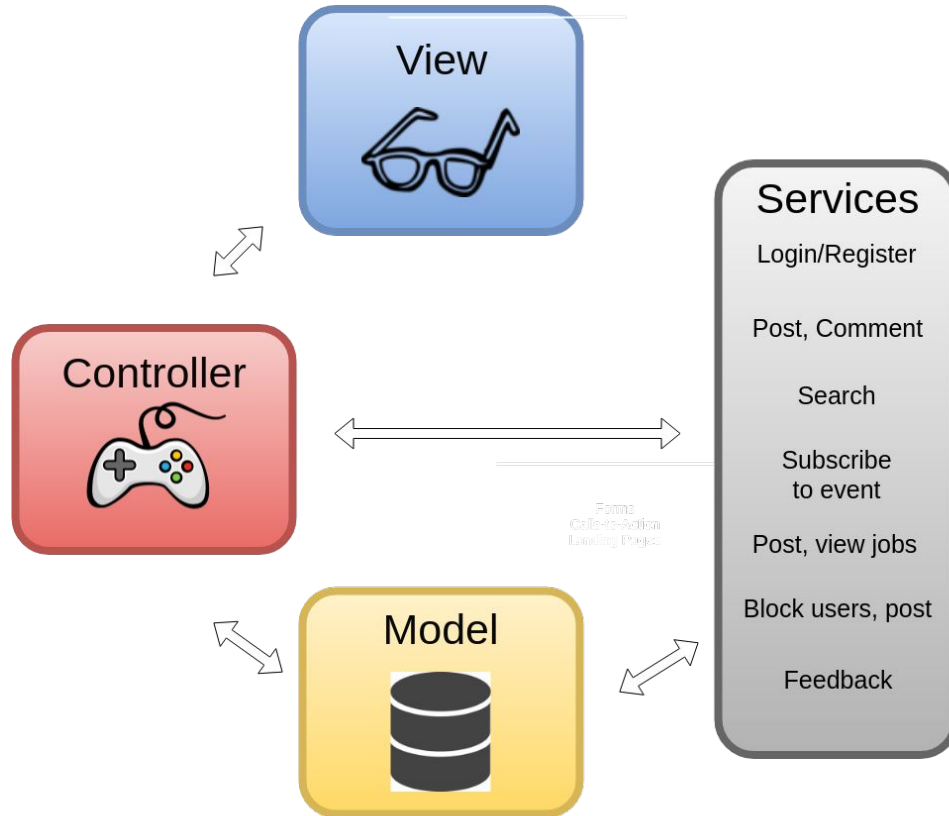
- View/edit job referrals, blog posts
- Respond to feedback, reports
- Remove offensive content
- Create/edit events



# Non-Functional Requirements

- **Performance** -
  - Mean latency is 4.8ms
  - Our system can handle ~350 RPS
  - TP50: 4ms , TP99: 12ms
- **Security** - Authentication and Authorization handled by Django
- **Availability** - System will have 99.9% uptime.
- **Capacity** - System is expected to be able to store upto 50,000 alumni.
- **Maintainability** - Well documented code written clearly and concisely.
- **Usability** - Responsive design.
- **Data Retention Policy** - Automated database backups done using cron jobs. Database log, error log and logging modules will be maintained. Logs older than one year will be deleted.

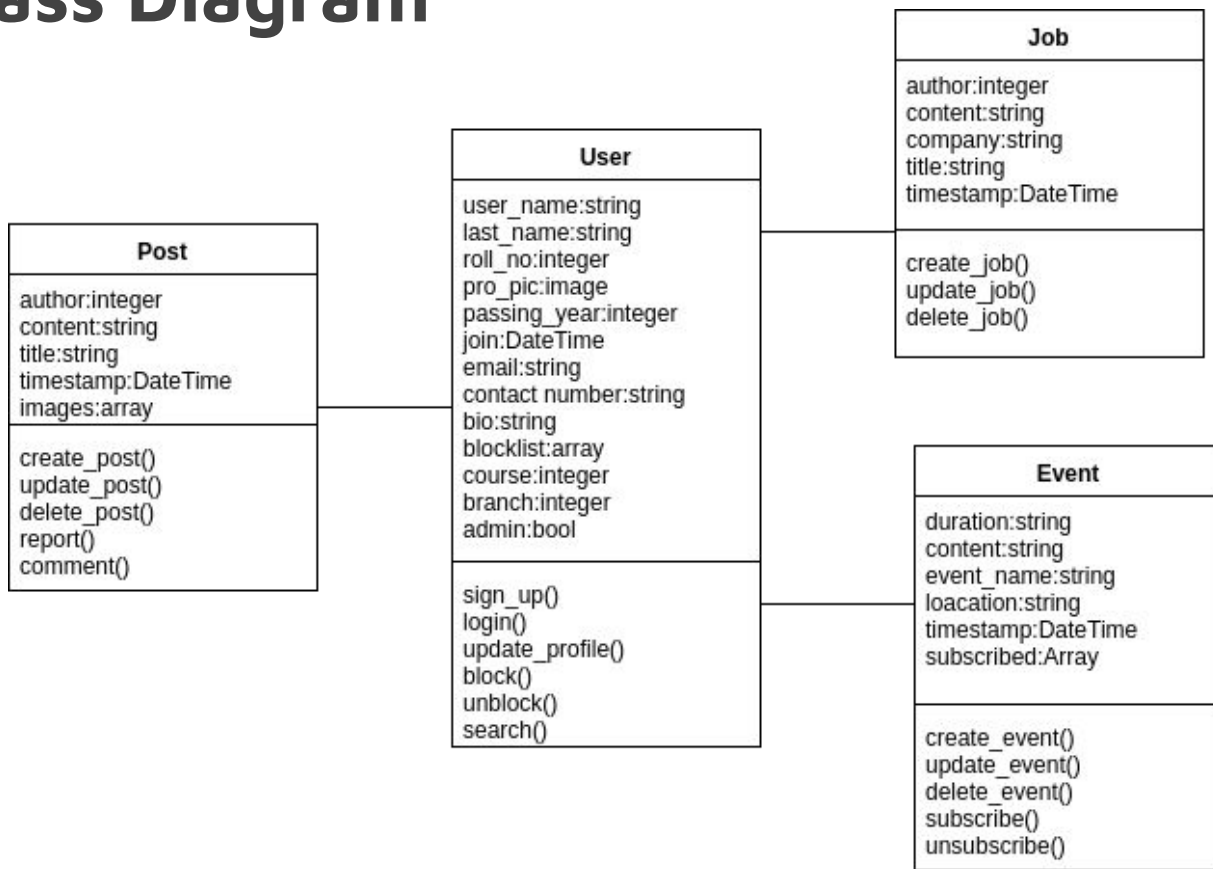
# Architecture - MVC





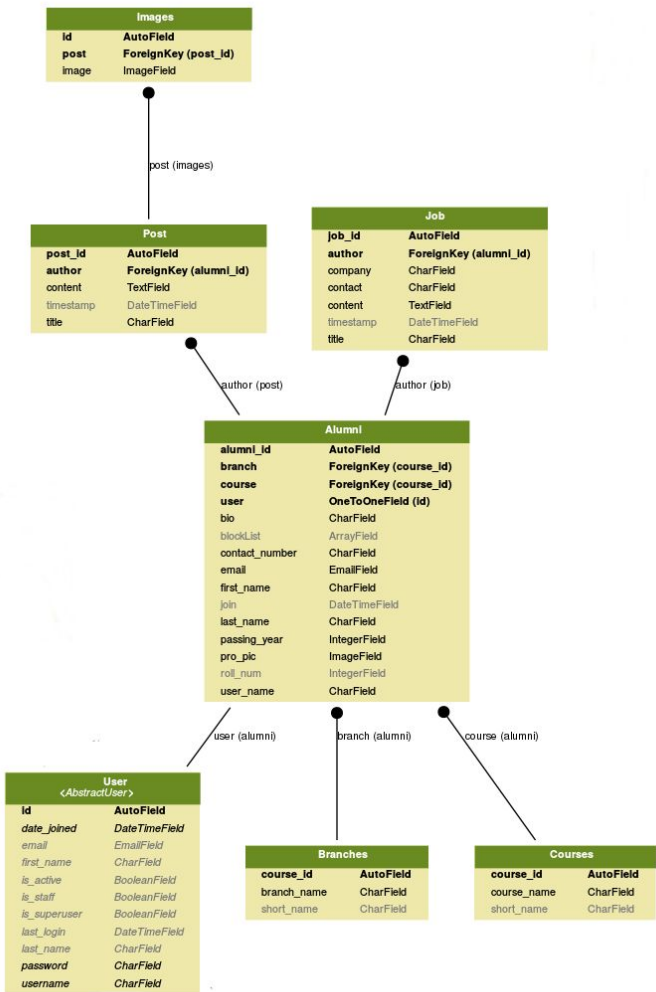


# Class Diagram





# Database Schema





# Tech Stack

- Mostly python, postgresql, js
- **Backend-** Django for creating web service.
  - Postgresql as database
  - Bash scripts for backing up data
  - Hosting will be done using apache2 wsgi
- **Frontend-** Bootstrap, jQuery for handling frontend.
- Pixel based size constraint on photos upload.



**DEMO**