Mid Term Report

Career Counselling Portal

**BACHELOR OF TECHNOLOGY**

(Computer Science and Engineering.)



**SUBMITTED BY:**

Ravi Kumar Singh (2230840)

Prateek Singh Jadon(2230829)

Payal Rawat (2230823)

Sakshi Kumari (2230857)

Jan 2025

**Under the Guidance of**

Dr. Ashima

Associate Professor

**Department of Computer Science & Engineering**

**Chandigarh Engineering College**

**Jhanjeri, Mohali – 14030**

**INDEX**

**Chapter 1 Introduction**

Objectives of the Project , Tools and Technologies Used

**Chapter 2 System Requirements**

Software and Hardware requirements

**Chapter 3 Software Requirement Analysis**

Problem Definition

Modules and Functionalities

**Chapter 4 Software Design**

Design Overview

Data Flow Diagrams (DFDs)

Database Design

E-R Diagrams

**Chapter 5 Implementation**

Code Structure and Explanation

Tables with Fields and Datatype.

1. **Introduction**

In today's landscape, students encounter numerous challenges when it comes to selecting their career paths. Many find themselves uncertain about their interests, often turning to peers or external influences for guidance rather than exploring their own passions. Enter CareerGPT, designed to assist individuals in uncovering their unique interests and passions, guiding them towards the most suitable career paths.

## Objectives of the Proposed System

* Provide free career counselling to all students in the world.
* Achieve high satisfaction rate among users, measured through feedback and surveys.
* Ensure smooth uptime for the portal to provide continuous and reliable access to career guidance.
* Reduce the cost of career counselling services for users by 100% compared to existing paid models.
* Maintain a response time of under 1 minute for AI-driven career advice queries.
* Support career counselling sessions for international students.
* Reduce the manual effort required by counsellors through automated data management and AI-driven recommendations.
* Provide training and support materials that lead to high satisfaction rate among counsellors using the platform.

1. **System Requirements**

| **Component** | **Recommended Specification** |
| --- | --- |
| **Processor (CPU)** | Multi-core (Quad-core or higher) – e.g., Intel Xeon / AMD Ryzen 5 or above |
| **Memory (RAM)** | At least 16 GB RAM (32 GB preferred for smooth AI + chat handling) |
| **Storage** | SSD with at least 256 GB space (for faster read/write and better performance) |
| **Operating System** | Ubuntu Server 22.04 LTS or latest Windows Server (based on preference) |
| **Deployment** | Can be deployed using Docker or hosted via cloud services (like AWS EC2, DigitalOcean, or Heroku) |
| **Backup & Uptime** | Daily backups and 99.9% uptime for production server recommended |

**3 Software Requirement Analysis**

## 3.1.Problem Statement

In today's educational landscape, students face significant challenges in selecting their career paths, often grappling with uncertainty about their interests and potential professions. Many students rely on peer opinions or external influences rather than exploring their own passions, leading to dissatisfaction and misaligned career choices. The lack of personalized, accessible, and cost-effective career counselling escalates this issue, leaving students without the necessary guidance to make informed decisions. Existing solutions, such as paid AI models and limited dataset services, are often expensive and inaccessible to a broad audience, restricting their utility. Moreover, the absence of continuous support and contextual interactions in these models hinders the counselling process. To address these problems, we are developing the Career Counselling Portal, a platform that leverages the LLAMA2 model to provide free, personalized, and extensive career guidance. This system will facilitate continuous, context-aware interactions, ensuring students receive consistent and tailored advice. By offering a comprehensive and accessible counselling solution, the Career Counselling Portal aims to empower students to make informed career choices aligned with their interests and aspirations, ultimately fostering a more satisfied and productive future workforce.

## 3.2.Problem Solution

* Provide free and accessible career counselling services to students globally.
* Utilize the LLAMA2 model to offer personalized and detailed career guidance.
* Maintain continuous and context-aware interactions to enhance user experience.
* Overcome the limitations of existing paid models and limited dataset services.
* Facilitate the identification of students' interests and passions to guide them towards suitable career paths.
* Ensure a user-friendly interface that simplifies the counselling process.
* Support counsellor registration, focusing on professionals from India, while offering services to international students.
* Achieve a responsive and efficient system through the use of modern web development technologies.
* Develop comprehensive documentation and reports to facilitate future enhancements and scalability.

# Software Design

## 4.1 Architectural Design

**Scenario 1: Website Visit**

Users who simply visit the website will interact solely with the Presentation Layer. This layer is implemented using React JS for the frontend.

**Scenario 2: Using CareerGPT Feature**

1. ***Login/Signup****:*

* When a user wants to use the CareerGPT feature (an AI model providing career suggestions to students), they need to log in or sign up first.
* The request from the Presentation Layer (React JS) will be sent to the Django server.
* The Django server will handle the request and save the user data in the SQLite database via the Django ORM.

1. ***Using CareerGPT****:*

* After logging in, the user can use the CareerGPT feature.
* The React frontend will call the API of the CareerGPT model, hosted on Kaggle, through Ngrok.
* The model will process the user’s query and respond accordingly.
* This interaction involves continuous queries and responses between the user and the model.

**Scenario 3: Chat Functionality**

1. ***Initiating Chat****:*
   1. When a user wants to use the chat functionality, the request will go from the Presentation Layer (React JS) to the Django server in the Business Layer.
   2. The Django server will then forward the request to the SendBird server.

-

* 1. SendBird will create a user and a channel for the chat and return the data to the Django server.
  2. The React frontend will then display this data to the user.

1. ***Real-Time Notifications****:*
   1. Pusher is used for real-time notifications. It employs WebSockets in the background.
   2. When a user or counsellor sends a message, SendBird triggers a webhook (an HTTP callback) to the Django server.

The Django server receives this data and uses Pusher to publish it on a specific channel.

The React frontend subscribes to this channel, receiving the data and displaying it to the user.

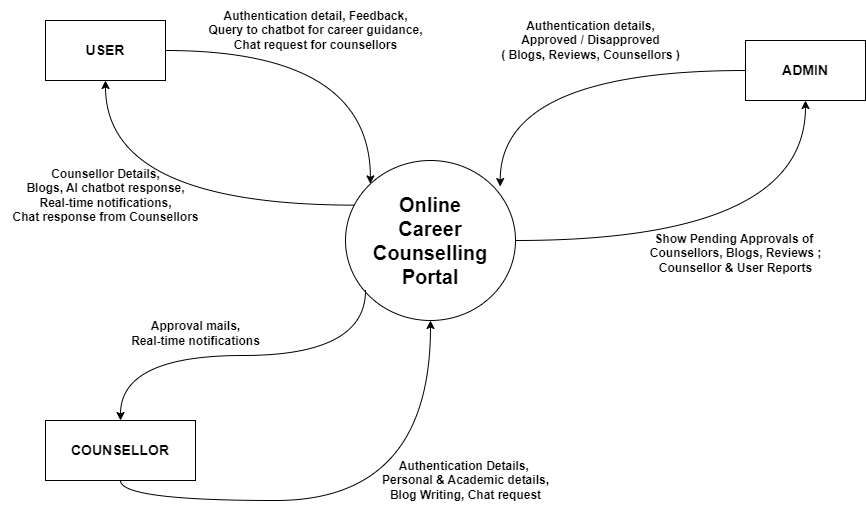
This mechanism ensures that both users and counsellors are notified in real-time about new messages.



Figure 2: Architectural Diagram

**4.2 Data Flow Diagrams**

* **Level 0**



## Data Design

In our career counselling portal, the information domain is transformed into structured data that supports the various functionalities of the system. The design focuses on storing, processing, and organizing data effectively to ensure smooth operation and efficient data retrieval.

***Major Data or System Entities***

1. **User Data (ACU)**:
   1. Information about users who sign up and log in to the portal, including their name, email, password, role, and creation timestamp.
   2. Stored in the SQLite database using the ACU model.
   3. Processed for authentication, authorization, and interaction with AI models and chat functionalities.
2. **Counsellor Data**:
   1. Additional details for users who are counsellors, including phone number, gender, CNIC, profile picture, CNIC images, approval status, and creation timestamp.
   2. Stored in the Counsellor model and linked to the ACU model.
3. **Qualification Data**:
   1. Academic qualifications of counsellors, including qualification details, field of study, transcript image, and creation timestamp.
   2. Stored in the Qualification model and linked to the Counsellor model.
4. **Working Experience Data**:
   1. Professional experience of counsellors, including institute name, starting and ending years, certificates image, and creation timestamp.
   2. Stored in the WorkingExperience model and linked to the Counsellor model.
5. **Ratings and Reviews**:
   1. Ratings and reviews given by users to counsellors, including rating value, review description, user details, approval status, and creation timestamp.
   2. Stored in the Ratings and Reviews models, linked to the Counsellor and ACU models respectively.
6. **Blog Data**:
   1. Blog posts written by counsellors, including title, author name, area of field, description, cover image, approval status, and creation date.
   2. Stored in the Blogs model and linked to the Counsellor model.
7. **CareerGPT Interaction History**:
   1. History of interactions with the CareerGPT feature, including message ID, message content, message type (bot/user), and associated user.
   2. Stored in the CareerGPTHistory model and linked to the ACU model.
8. **Chat Data**:
   1. Messages exchanged between users and counsellors.
   2. Managed through SendBird and Pusher for real-time updates and notifications.
   3. Stored and retrieved to maintain conversation history.

***Data Storage Items***

1. **SQLite Database**:
   1. Stores all user-related data such as profiles, counsellor details, qualifications, working experiences, ratings, reviews, blogs, and CareerGPT history.
   2. Utilizes Django ORM for handling database operations.
2. **SendBird Server**:
   1. Manages chat-related data, including user creation, channel creation, and message storage.
   2. Provides real-time communication features.
3. **Pusher**:
   1. Handles real-time notifications for chat messages using WebSockets.
   2. Publishes and subscribes to data channels to notify users of new messages.

### **4.3.1 Data Dictionary**

***Entities and Attributes***

1. **ACU (User)**
   1. **id** (Integer): Unique identifier for the user.
   2. **name** (Text): User's name.
   3. **email** (EmailField): User's email address, unique.
   4. **password** (Text): Encrypted password for the user.
   5. **role** (CharField): Role of the user, default is 'U'.
   6. **created\_at** (DateTimeField): Timestamp when the user was created.
2. **Counsellor**
   1. **id** (Integer): Unique identifier for the counsellor.
   2. **counsellor\_id** (ForeignKey): Link to the ACU model.
   3. **phone\_no** (Text): Phone number of the counsellor.
   4. **gender** (CharField): Gender of the counsellor.
   5. **cnic** (CharField): CNIC of the counsellor.
   6. **profile\_pic** (Text): Profile picture URL.
   7. **cnic\_front\_img** (Text): CNIC front image URL.
   8. **cnic\_back\_img** (Text): CNIC back image URL.
   9. **is\_approved** (BooleanField): Approval status of the counsellor.
   10. **created\_at** (DateTimeField): Timestamp when the counsellor was created.
3. **Qualification**
   1. **counsellor\_id** (OneToOneField): Link to the Counsellor model.
   2. **qualification** (Text): Qualification details.
   3. **field\_of\_study** (Text): Field of study.
   4. **transcript\_img** (Text): Transcript image URL.
   5. **created\_at** (DateTimeField): Timestamp when the qualification was created.
4. **WorkingExperience**
   1. **id** (Integer): Unique identifier for the working experience.
   2. **counsellor\_id** (ForeignKey): Link to the Counsellor model.
   3. **institute\_name** (Text): Name of the institute.
   4. **starting\_year** (Text): Starting year of the experience.
   5. **ending\_year** (Text): Ending year of the experience.
   6. **certificates\_image** (Text): Certificates image URL.
   7. **created\_at** (DateTimeField): Timestamp when the experience was created.
5. **Ratings**
   1. **id** (Integer): Unique identifier for the rating.
   2. **counsellor\_id** (ForeignKey): Link to the Counsellor model.
   3. **rating** (Integer): Rating value.
   4. **review\_description** (Text): Description of the review.
   5. **user\_id** (ForeignKey): Link to the ACU model.
   6. **created\_at** (DateTimeField): Timestamp when the rating was created.
6. **Reviews**
   1. **id** (Integer): Unique identifier for the review.
   2. **user\_id** (ForeignKey): Link to the ACU model.
   3. **reviewer\_name** (Text): Name of the reviewer.
   4. **reviewer\_email** (EmailField): Email of the reviewer.
   5. **reviewer\_description** (Text): Description of the review.
   6. **is\_approved** (BooleanField): Approval status of the review.
   7. **created\_at** (DateTimeField): Timestamp when the review was created.
7. **Blogs**
   1. **id** (Integer): Unique identifier for the blog.
   2. **counsellor\_id** (ForeignKey): Link to the Counsellor model.
   3. **title** (Text): Title of the blog.
   4. **author\_name** (Text): Author's name.
   5. **area\_of\_field** (Text): Area of field.
   6. **description** (Text): Description of the blog.
   7. **cover\_image** (Text): Cover image URL.
   8. **is\_approved** (BooleanField): Approval status of the blog.
   9. **created\_at** (DateField): Date when the blog was created.
8. **CareerGPTHistory**
   1. **id** (Integer): Unique identifier for the history record.
   2. **user\_id** (ForeignKey): Link to the ACU model.
   3. **msgId** (BigInteger): Message ID.
   4. **message** (Text): Message content.
   5. **type** (CharField): Type of the message (bot/user).
   6. **created\_at** (DateTimeField): Timestamp when the history was recorded.

***4.4 Entity Relationship Diagram***

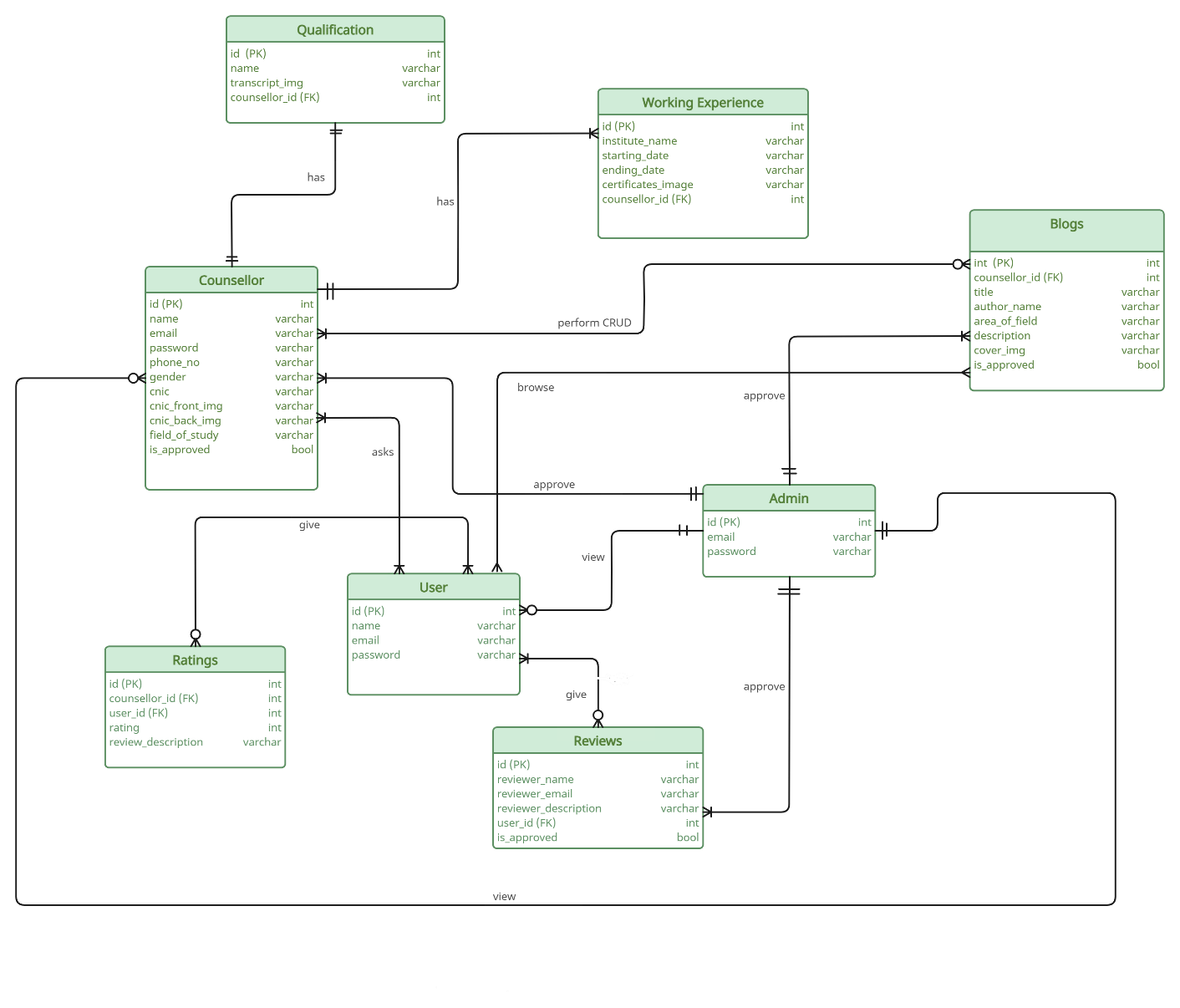


Figure 6: Entity Relationship Diagra

# Implementation

## Algorithm

Table 5: Algorithms

|  |
| --- |
| **Algorithm 1 User Registration and Authentication** |
| **Input:** Request containing user data |
| **Output:** HTTP Response with success or error message |
| **1: FUNCTION generate\_otp():**  **2: RETURN random\_number\_between(1000, 9999)**  **3:**  **4: FUNCTION sendOTP(request):**  **5: IF request.method == 'POST':**  **6: data = parse\_json(request.body)**  **7: email = data['email']**  **8: otp = generate\_otp()**  **9: send\_email('Verification Code', 'Your verification code is: ' + otp, email)**  **10: RETURN HTTPResponse({'otp': otp})**  **11: ELSE:**  **12: RETURN HTTPResponse({'error': 'Method not allowed'}, status=405)**  **13:**  **14: FUNCTION registerUser(request):**  **15: IF request.method == 'POST':**  **16: data = parse\_json(request.body)**  **17: signup\_data = data['signupData']**  **18: name = signup\_data['name']**  **19: email = signup\_data['email']**  **20: password = hash\_password(signup\_data['password'])**  **21: create\_user(name, email, password)**  **22: RETURN HTTPResponse({'status': 'success'})**  **23: ELSE:**  **24: RETURN HTTPResponse({'error': 'Method not allowed'}, status=405)**  **25:**  **26: FUNCTION checkEmail(request):**  **27: IF request.method == 'POST':**  **28: data = parse\_json(request.body)**  **29: email = data['email']**  **30: IF email\_exists(email):**  **31: RETURN HTTPResponse({'isExist': True})**  **32: ELSE:**  **33: RETURN HTTPResponse({'isExist': False})**  **34: ELSE:**  **35: RETURN HTTPResponse({'error': 'Method not allowed'}, status=405)** |

### **Home Screen**

Home Screen presents the navigation bar and popular blogs and latest reviews for

the user to have a quick view of blogs and other informative data.



Figure 7: Home Screen