Software Design Document

JobMatchU

Version 1.0

Prepared by

Febin P Biju
Rizwan AT
Hains Dolichan
Don Philip

TKM College of Engineering

Table of Contents

1. Introduction

This Software Design Document (SDD) is intended to provide a comprehensive overview of the design and architecture of a software system. The document is aimed at developers, architects, and stakeholders involved in the development of the system. It outlines the key design decisions, architectural patterns, and technologies used to build the system.

The SDD covers various aspects of software design, including system architecture design, application architecture design, GUI design, API design, and technology stack. The document provides detailed descriptions of the design decisions and provides an overview of the system architecture, the application architecture, and the user interface design. Additionally, it provides details about the API design and the technologies used to build the system.

This document aims to serve as a reference guide for the development team throughout the development process. It will help to ensure that the team remains aligned with the design goals and can make informed decisions regarding the implementation of the system.

1.1 Purpose

The purpose of the Software Design Document (SDD) is to provide a comprehensive overview of the design of the software system being developed. It serves as a guide for the developers, testers, and stakeholders involved in the project, and provides a common understanding of the system architecture, application architecture, GUI design, API design, database design, and technology stack. The document outlines the technical aspects of the software and the design decisions that were made during the development process, ensuring that the software meets the specified requirements and performs as expected. The SDD also helps to identify potential issues and risks that may arise during the development and implementation phases, enabling proactive measures to be taken to mitigate them. Overall, the SDD serves as a critical reference document for the development team throughout the software development lifecycle.

1.2 Scope

The scope of this Software Design Document (SDD) is to provide a comprehensive overview of the design and architecture of the proposed application. It covers the system architecture design, application architecture design, GUI design (mockups), API design, and the technology stack that will be used to develop the application. The SDD aims to provide a clear understanding of the design and architecture of the application to the development team and stakeholders involved in the project. It will also serve as a reference guide for the development team throughout the development process. The SDD is intended to be a living document and will be updated as needed throughout the project lifecycle.

1.3 Intended Audience

The intended audience for this SDD includes software developers, College Professors, and other stakeholders involved in the software development process. This document is intended to provide a detailed design of the system architecture, application architecture, GUI design, API design, and technology stack, as well as any other relevant technical details related to the development of the software application.

The audience can review the overall design and technical description and focus on specific components such as the system architecture, application architecture, GUI design, API design, and technology stack. Any concerns or areas of interest that could impact the development or evaluation of the software application can be noted for further discussion and clarification.

1.4 References

- IEEE Standards Association. IEEE Std 1016-2009, IEEE Standard for Information Technology--Systems Design--Software Design Descriptions. IEEE, 2009.

2. System Architecture Design

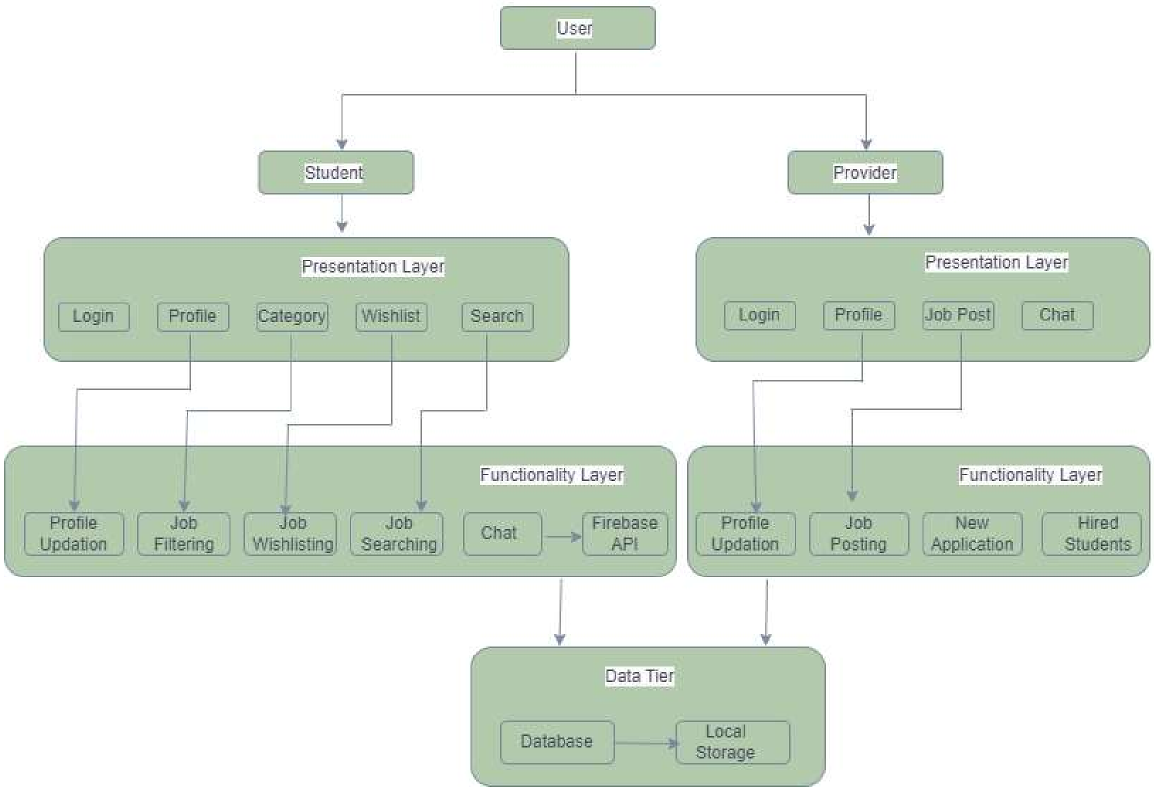
2.1 Description

System architecture includes two types of users: students and job providers. Both user types have different capabilities in the presentation and functionality layers of the application.

The presentation layer for students allows them to log in to the system and create a profile with personal and professional details. They can also view different jobs, filter them based on their preferences, add them to a wish list, and search for jobs that match their skill set. The functionality layer for students includes the ability to update their profile information, filter and sort through job listings, and communicate with job providers through the built-in chat system.

On the other hand, job providers can also log in to the system and create a profile with their company details. They can post jobs that are available, review applications that are received, and maintain details of students that they have hired in the past. The presentation layer for job providers includes the same features as that of students - profile maintenance, job posting, and chat with students. All the user data, such as student profiles, job postings, and application details, are stored in a database that is located in the data tier of the system architecture.

2.2 Architecture



3. Application Architecture Design

JobMatchU will be a website designed to provide part-time jobs for students through which they can gain experience and earnings and for job providers to get employees without any complex procedures. The application architecture will consist of two main components: a frontend built using the ReactJs framework, and a backend built using the Django web framework.

The front end will consist of different views such as the home page, job search page, job detail page, user profile page, etc. Each of these views will be implemented using appropriate components and UI libraries. The front end will also be responsible for handling user authentication and authorization, fetching and displaying data from the backend, and managing user interactions.

The backend will handle various tasks such as managing user authentication and authorization, handling user data, managing job postings and applications, and handling communication between users.

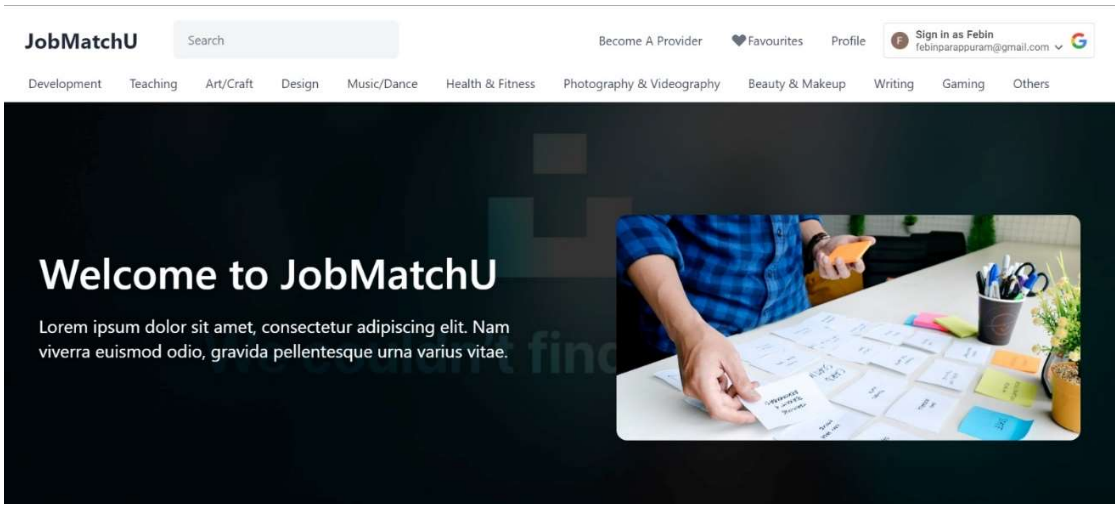
Overall, the application architecture is designed to provide a smooth and user friendly website for students who are interested in doing part-time jobs in their free time along with their studies.

4. GUI Design

4.1 User Interface Design

a. Home Page

Displaying navigation bar, introduction, recommended jobs, picked jobs, footer and job categories.

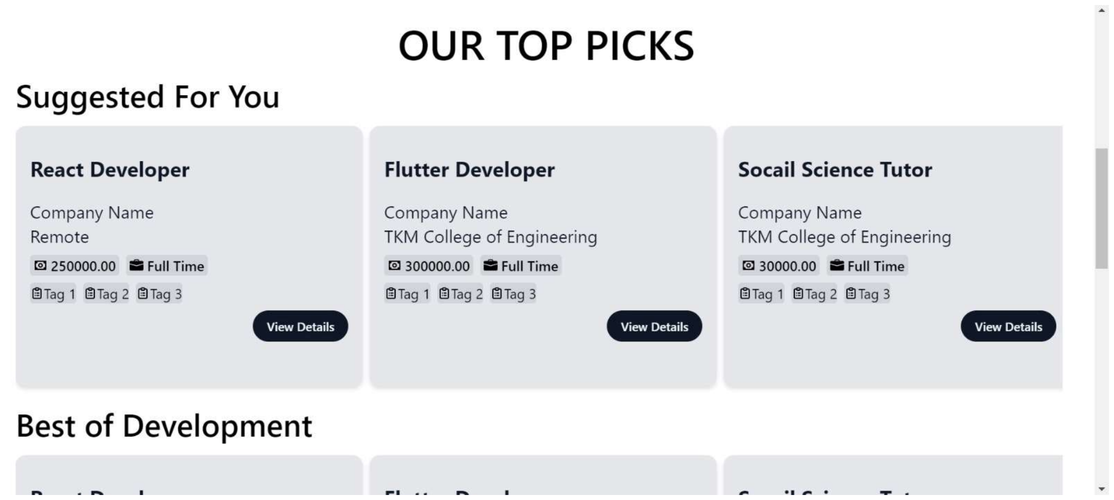


JobMatchU Search Become A Provider Favourites Profile Sign in as Febin febinparappuram@gmail.com

Development Teaching Art/Craft Design Music/Dance Health & Fitness Photography & Videography Beauty & Makeup Writing Gaming Others

Welcome to JobMatchU

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam viverra euismod odio, gravida pellentesque urna varius vitae.



OUR TOP PICKS

Suggested For You

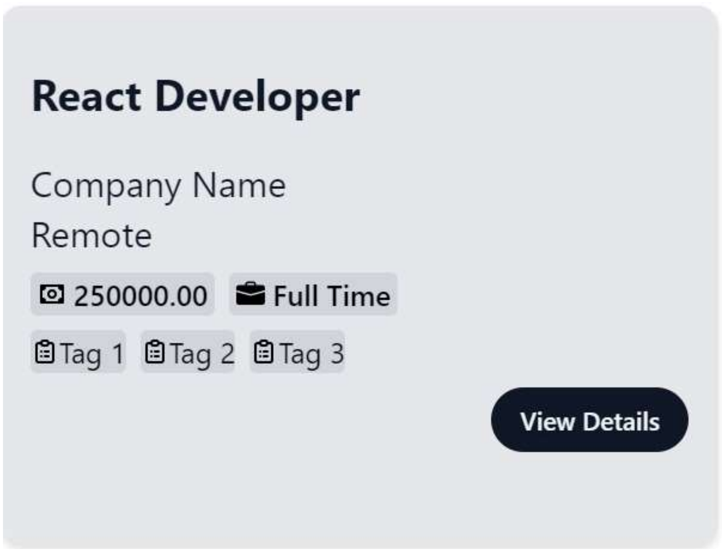
React Developer
Company Name
Remote
250000.00 Full Time
Tag 1 Tag 2 Tag 3
View Details

Flutter Developer
Company Name
TKM College of Engineering
300000.00 Full Time
Tag 1 Tag 2 Tag 3
View Details

Socail Science Tutor
Company Name
TKM College of Engineering
30000.00 Full Time
Tag 1 Tag 2 Tag 3
View Details

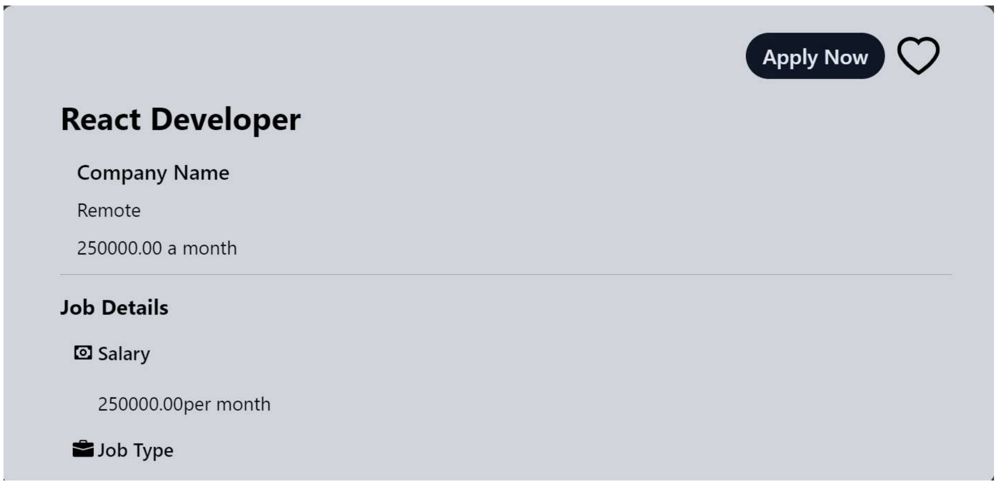
Best of Development

b. Job Card
Provides major information like job title, company name, location, salary, etc. about a particular job and a button to show full details.



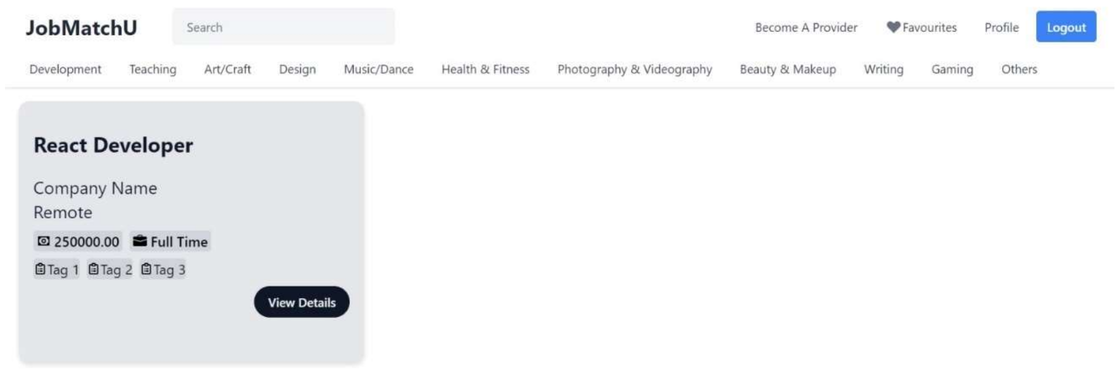
c. Job Description Page

- This page includes all the details of the job including job title, company details, location details, duration of job, salary, description, hiring insights and include 3 buttons: One to apply for the jobs, one to add job to favorites, and another one to close the job description page.



d. Favorites page

- This page includes the wish listed jobs for each user.

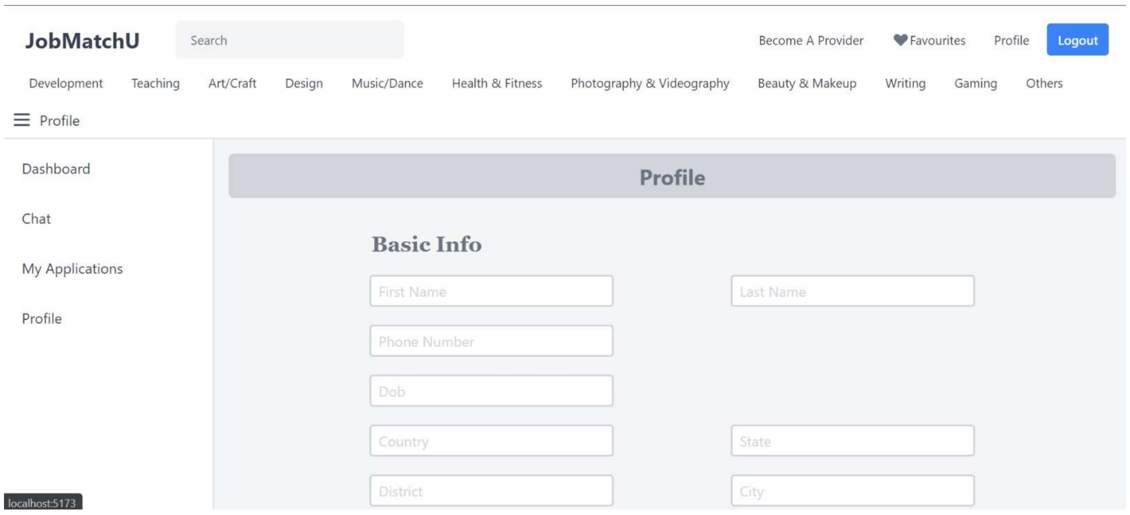


React Developer

Company Name
Remote
250000.00 Full Time
Tag 1 Tag 2 Tag 3

e. Profile Information Page

- This page receives basic information, academic information, work experiences and related documents and saves the details.



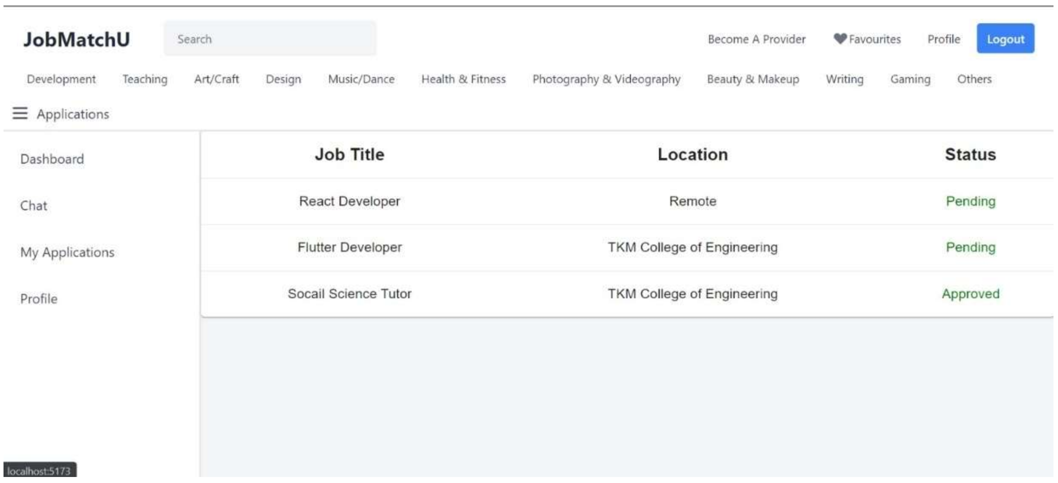
Profile

Basic Info

First Name | Last Name
Phone Number
Dob
Country | State
District | City

f. My Application Page

- This page includes the details of applied jobs and their current status.



Job Title	Location	Status
React Developer	Remote	Pending
Flutter Developer	TKM College of Engineering	Pending
Socail Science Tutor	TKM College of Engineering	Approved

g. Job Category Page

- This page includes the jobs that comes under a specific category

5. Technological Stacks

- ReactJS: A popular JavaScript library for building user interfaces. React allows developers to create reusable UI components and efficiently manage the state of an application.

- Django: A high-level Python web framework that follows the model-template-view architectural pattern. Django is known for its robustness, security, and scalability.

- Netlify: A cloud-based platform for deploying and hosting modern web applications. Netlify offers features such as continuous deployment, automatic SSL certificates, and serverless functions.

- Docker: A containerization technology that allows developers to package their applications and dependencies into isolated containers. Docker containers are lightweight, portable, and easy to deploy across different environments.

- Microsoft Azure: A cloud computing platform and service offered by Microsoft. Azure provides a wide range of services such as virtual machines, storage, databases, and AI tools.

- Figma: A cloud-based design tool used for creating user interfaces, prototypes, and collaborative design projects. Figma allows designers and developers to work together in real-time and easily share design assets.

6. Data Design

- Data Storage
  - The application will need to store various types of data, including user account information and email messages. To do this, we will use a relational database management system (RDBMS) such as PostgreSQL. The database will have the following tables:
  - Users: Stores user account information such as userid, name, email and other basic details.
  - Jobs: Stores the details like job title, job_id, duration etc. of each job.
  - Job Category: Stores information like category_id, job_id, etc. each category of job
  - Wishlist: To keep track of wish listed jobs by each user.
  - Company: Stores job providers' information like name, email, size, etc.
  - Apply: Stores the details of applied jobs including user_id, job_id, etc.

- Data Access
  - To access the data stored in the RDBMS, the Django ORM provides several APIs such as QuerySet, and Model. This will allow us to access the data usingobject-oriented programming techniques.

- Data Backup and Recovery
  - To ensure that user data is not lost in the event of a system failure, we will implement regular backups of the database. We will also develop a recovery plan in case of a disaster, such as restoring the database from a backup or using a replication solution to maintain multiple copies of the data. We will use Django's built-in dumpdata and loaddata commands to backup and restore the database.

7. Error Handling Design

- Centralized Error Logging: Set up a centralized error logging system that will capture and store all application errors. This system can be used to monitor application errors and to track down and fix issues.

- User-Friendly Error Messages: Ensure that error messages are clear, concise, and user-friendly. They should describe the error in simple language and provide guidance on how to fix the problem.

- Input Validation: Validate user input on both the client and server sides to prevent invalid data from being entered into the system.

- Try/Catch Blocks: Use try/catch blocks to handle exceptions and errors that may occur during application execution. This will help prevent crashes and keep the application running smoothly.

- Error Pages: Create custom error pages that provide helpful information and links to resources that can help users resolve issues.

8. Performance Design

- Caching: Caching can be implemented to improve the performance of the system. It can reduce the load on the server by storing frequently accessed data in a cache. Caching can be implemented at various levels such as application level, database level, or server level.

- Load Balancing: Load balancing can be used to distribute the incoming traffic across multiple servers. This can help in improving the performance of the system by reducing the load on individual servers. Load balancing can be implemented using hardware or software-based solutions.

- Database Optimization: The performance of the system can be improved by optimizing the database queries. This can be done by indexing the frequently used columns, avoiding subqueries, and using the appropriate data types.

- Asynchronous Processing: Asynchronous processing can be used to improve the performance of the system by executing long-running tasks in the background. This can be achieved by using tools such as Celery, which allows us to run tasks asynchronously.

- Caching of Static Content: Static content such as images, CSS files, and JavaScript files can be cached using CDNs (Content Delivery Networks). This can help in reducing the load on the server and improving the performance of the system.

- Compression: Compression techniques such as Gzip can be used to compress the response data before sending it to the client. This can help in reducing the network bandwidth required for transmitting the data and improving the performance of the system.

- Monitoring and Alerting: Continuous monitoring of the system can help in identifying performance bottlenecks and potential failures. Alerts can be set up to notify the administrators in case of any performance degradation or system failures.

9. Security Design

- Authentication and Authorization: Implement a strong authentication and authorization mechanism to ensure that only authorized users can access the system. Use secure password storage and encryption techniques to protect user credentials.

- Data Encryption: Encrypt sensitive user information such as personal details, educational qualifications, and job-related information while storing it in the database. Use industry-standard encryption algorithms to ensure data confidentiality.

- Secure Communication: Use secure communication protocols such as HTTPS to protect the data transmitted between the client and server. Implement SSL/TLS certificates to ensure secure communication.

- Input Validation: Implement input validation techniques to prevent any malicious inputs from users. Validate input data such as email addresses, phone numbers, and other user-entered data to prevent injection attacks.

- Role-Based Access Control: Implement a role-based access control mechanism to ensure that users have the appropriate level of access based on their roles. This will help prevent unauthorized access to sensitive data.

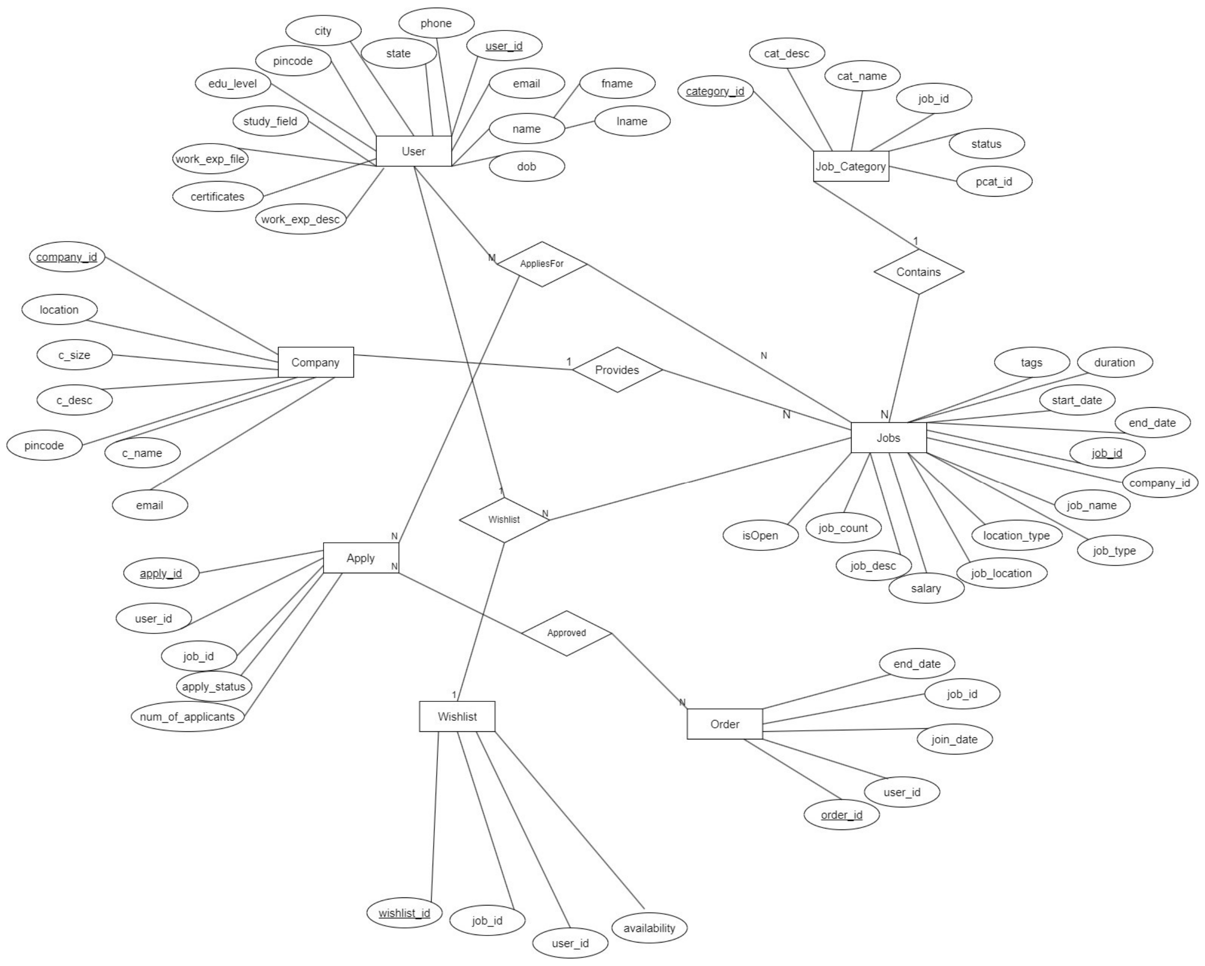
- Security Testing: Conduct regular security testing and vulnerability assessments to identify and address any potential security vulnerabilities in the system. Use tools such as penetration testing and vulnerability scanning to test the system's security.

10. Maintenance Design

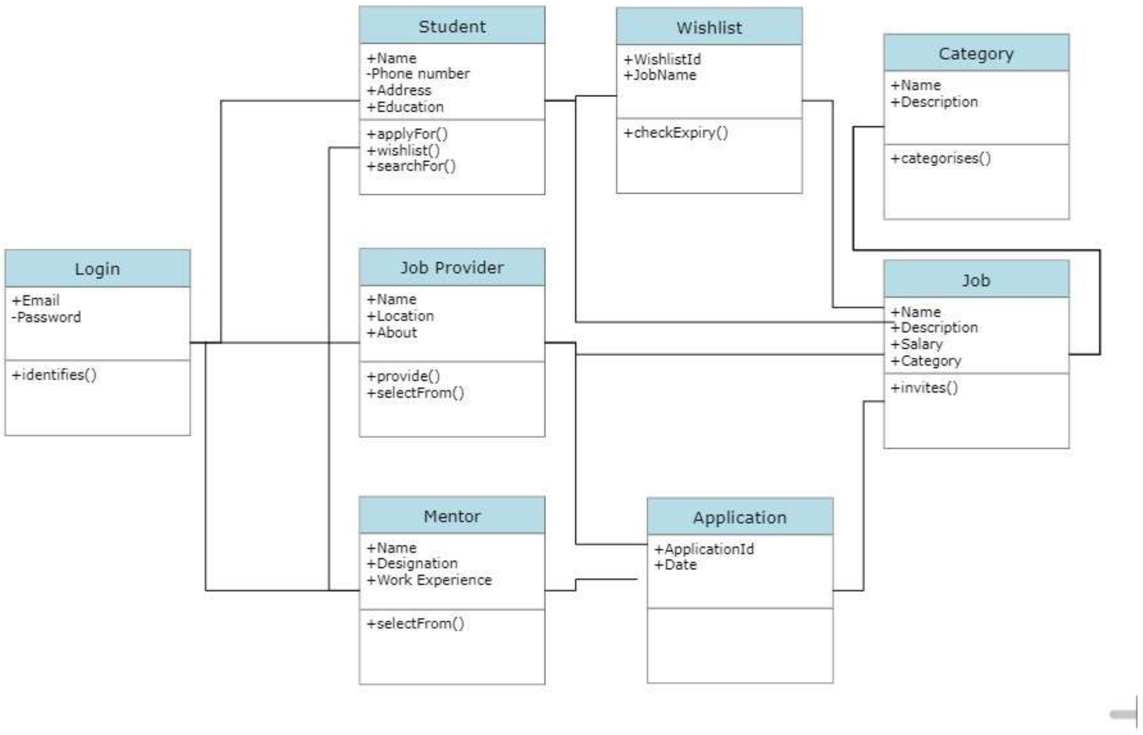
- Authentication and Authorization: Implement a strong authentication and authorization mechanism to ensure that only authorized users can access the system. Use secure password storage and encryption techniques to protect user credentials.
- Data Encryption: Encrypt sensitive user information such as personal details, educational qualifications, and job-related information while storing it in the database. Use industry-standard encryption algorithms to ensure data confidentiality.
- Secure Communication: Use secure communication protocols such as HTTPS to protect the data transmitted between the client and server. Implement SSL/TLS certificates to ensure secure communication.
- Input Validation: Implement input validation techniques to prevent any malicious inputs from users. Validate input data such as email addresses, phone numbers, and other user-entered data to prevent injection attacks.
- Role-Based Access Control: Implement a role-based access control mechanism to ensure that users have the appropriate level of access based on their roles. This will help prevent unauthorized access to sensitive data.
- Security Testing: Conduct regular security testing and vulnerability assessments to identify and address any potential security vulnerabilities in the system. Use tools such as penetration testing and vulnerability scanning to test the system's security

11. Appendices

- ER Diagram



- UML Class Diagram


Student
+Name
-Phone number
+Address
+Education
+applyFor()
+wishlist()
+searchFor()
Wishlist
+WishlistId
+JobName
+checkExpiry()
Category
+Name
+Description
+categorises()
Login
+Email
-Password
+identifies()
Job Provider
+Name
+Location
+About
+provide()
+selectFrom()
Job
+Name
+Description
+Salary
+Category
+invites()
Mentor
+Name
+Designation
+Work Experience
+selectFrom()
Application
+ApplicationId
+Date