

Annexure-III

Hard bound cover

Quiz Website

A project report

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

(Computer Science and Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB



From 03/10/2024 to 04/25/2024

SUBMITTED BY

Name of student

Shivam Nath

Registration Number

12018682

Name of Supervisor

Nitish Kumar

UID of Supervisor

30748

Annexure-V

Declaration by student

To whom so ever it may concern

I, **Shivam Nath, 12018682**, hereby declare that the work done by me on “**Quiz website**” under the supervision of **Nitish Kumar** Lovely professional University, Phagwara, Punjab, is a record of original work for the partial fulfilment of the requirements for the award of the degree, **Bachelor of Technology**.

Shivam Nath (12018682)

Signature of the student

Dated:25/04/2024

Annexure-VI

Declaration by the supervisor

To whom so ever it may concern

This is to certify that **Shivam Nath, 12018682** from Lovely Professional University, Phagwara, Punjab, has worked on “**Quiz website**” under my supervision from. It is further stated that the work carried out by the student is a record of original work to the best of my knowledge for the partial fulfilment of the requirements for the award of the degree, degree name.

Name of Supervisor

Nitish Kumar

UID of Supervisor

30748

Signature of Supervisor

Introduction

The Software Requirement Specification (SRS) document offers a thorough overview of the Quiz website to be developed using the Django framework. Serving as a compass for software engineers, it delineates the intricate requirements necessary for the effective design and implementation of the software product. The Quiz website aspires to revolutionize online quizzes by providing a centralized platform for users to create, discover, and participate in quizzes seamlessly. Through this website, users can explore a variety of quizzes on different topics, create their own quizzes, and challenge friends or other users. The SRS document encompasses all vital facets of the Quiz website, encompassing functional requirements, non-functional requirements, user interface specifications, system constraints, and external interfaces. It is meticulously structured to furnish software engineers with clear and succinct information, empowering them to craft a resilient and user-friendly software solution that caters to the diverse needs of users.

Furthermore, the document will undergo continuous refinement and validation throughout the software development lifecycle to ensure alignment with stakeholder expectations and project objectives, fostering the creation of an engaging and dynamic Quiz website.

1.1 Purpose:

The purpose of this Software Requirement Specification (SRS) document is to delineate the functional and non-functional requirements of the Quiz website developed using the Django framework. It serves as a comprehensive guide for software engineers and developers, furnishing them with a detailed comprehension of the project's scope, objectives, and specific features to be incorporated. Furthermore, the SRS document serves as a point of reference for stakeholders, encompassing project managers, quality assurance teams, and end-users, to guarantee alignment with the intended outcomes and expectations. The intended audience for this SRS document includes:

1. Software Engineers and Developers: Tasked with designing, implementing, and testing the Quiz website in accordance with the stipulated requirements and specifications.
2. Project Managers: Overseeing the project's advancement, resource allocation, and ensuring that development adheres to the defined requirements and timelines.
- 3 Quality Assurance Teams: Responsible for validating the functionality and performance of the Quiz website through meticulous testing methodologies.
4. Stakeholders: Encompassing executives, investors, and other pertinent parties invested in comprehending the project's scope, objectives, and deliverables.
5. End-users: Including individuals who will engage with the Quiz website, ensuring that their needs and expectations are met effectively, whether they are creating, discovering, or participating in quizzes.

1.2 Scope:

(1) The software product to be developed is a Quiz website utilizing the Django framework.

(2) The Quiz website will:

- Provide a user-friendly interface for users to create, discover, and participate in quizzes on various topics.
- Include features such as quiz creation tools, quiz browsing, and participation tracking.
- Offer options for users to share quizzes with friends or other users.
- Ensure data security and privacy by implementing appropriate encryption and access control measures.
- Allow for scalability and flexibility to accommodate future enhancements and integration with external systems.

The Quiz website will not:

- Guarantee quiz completion or specific outcomes for users.
- Provide quiz creation services beyond facilitating the creation and participation in quizzes.
- Offer automated decision-making processes for quiz creation or participation.

(3) Application of the software:

(a) The Quiz website aims to achieve the following benefits, objectives, and goals:

- Provide an engaging platform for users to create, explore, and enjoy quizzes on diverse topics.
- Offer a centralized hub for quiz enthusiasts to discover and share quizzes with others.
- Enable users to showcase their knowledge and creativity through quiz creation and participation.
- Enhance accessibility and user experience by ensuring mobile responsiveness and intuitive navigation.
- Ensure compliance with data privacy regulations and industry standards to safeguard user information.

(b) This scope aligns with the objectives outlined in higher-level specifications, such as the System Requirement Specification, ensuring consistency with project goals and stakeholder expectations.

1.3 Definitions, Acronyms, and Abbreviation

Definitions:

1. Quiz Website: A web application designed to facilitate the creation, discovery, and participation in quizzes on various topics, typically allowing users to create quizzes, browse existing quizzes, and track their quiz participation.
2. Django: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
3. User: A person who interacts with the quiz website, including quiz creators and participants.
4. Administrator: A user role with elevated privileges, responsible for managing users, quizzes, and other aspects of the website.
5. Authentication: The process of verifying the identity of users accessing the quiz website, typically through login credentials such as username and password.

CRUD Operations: Create, Read, Update, and Delete operations, representing the basic functions for managing data in a database. Acronyms and Abbreviations:

1. SRS: Software Requirements Specification
2. CRUD: Create, Read, Update, Delete
3. DB: Database
4. CSS: Cascading Style Sheets
5. HTML: Hypertext Markup Language
6. ORM: Object-Relational Mapping
7. HTTP: Hypertext Transfer Protocol

1.4 References

1. Django Documentation:

- Title: Django Documentation
- Publishing Organization: Django Software Foundation
- Sources: Available online at <https://docs.djangoproject.com/en/5.0/>

2. GeeksforGeeks Django Tutorials:

- Title: Django Tutorials on GeeksforGeeks
- Publishing Organization: GeeksforGeeks
- Sources: Available online at <https://www.geeksforgeeks.org/django-tutorial/>.

3. Tutorialspoint Django Tutorial:

- Title: Django Tutorial on Tutorialspoint
- Publishing Organization: Tutorialspoint
- Sources: Available online at <https://www.tutorialspoint.com/django/index.htm>.

4. W3Schools Django Tutorial:

- Title: Django Tutorial on W3Schools
- Publishing Organization: W3Schools
- Sources: Available online at <https://www.w3schools.com/django/default.asp>.

1.5 Overview:

(1) The rest of the Software Requirement Specification (SRS) contains detailed information about the Quiz Website developed using Django framework. It includes comprehensive documentation of the functional and non-functional requirements, user interface specifications, system constraints, and external interfaces necessary for designing and implementing the software product effectively. Additionally, the SRS outlines the scope, purpose, and application of the job application portal, providing a holistic view of the project for stakeholders and software engineers alike.

(2) The SRS is organized into several sections, each addressing specific aspects of the job application portal:

- Introduction: Provides an overview of the SRS document and its intended audience.
- Purpose: Defines the purpose of the SRS and identifies its intended audience.

- **Scope:** Describes the software product(s) to be produced, including what it will and will not do, as well as its application and objectives.
- **Definitions, Acronyms, and Abbreviations:** Lists and defines any technical terms, acronyms, or abbreviations used throughout the document.
- **References:** Includes any external documents or resources referenced in the SRS.
- **Functional Requirements:** Details the functional specifications of the job application portal, including user interactions, system behaviours, and data processing.
- **Non-Functional Requirements:** Outlines the non-functional specifications, such as performance, security, usability, and scalability considerations.
- **User Interface Specifications:** Describes the visual and interactive aspects of the user interface, including layout, navigation, and design elements.
- **System Constraints:** Identifies any limitations or constraints imposed by the system architecture, hardware, or software environment.
- **External Interfaces:** Specifies any external systems, APIs, or databases that the job application portal will interact with.
- **Appendices:** Includes additional supplementary information, such as mock-ups, diagrams, or supporting documentation.

By organizing the SRS in this structured manner, it provides a comprehensive and systematic guide for software engineers to design, develop, and test the job application portal effectively, ensuring alignment with stakeholder requirements and project objectives.

2. General Description:

The social media website developed using the Django framework is an online platform designed to facilitate seamless social interaction and content sharing among users. This section of the Software Requirement Specification (SRS) provides an overarching view of the key factors influencing the product and its requirements, offering stakeholders and software engineers a clear understanding of the project's context and objectives.

1.Purpose: The primary purpose of the social media website is to provide a centralized platform where users can connect with friends, share content, and engage in meaningful interactions. By offering features that foster communication and community building, the website aims to enhance the overall user experience and promote user engagement.

2.Target Audience: The target audience for the social media website includes individuals from diverse backgrounds and demographics who are interested in connecting with others and sharing their experiences online. This encompasses users ranging from individuals seeking social connections to content creators looking to reach a broader audience.

3.Functionality: The social media website will offer a wide range of functionality to cater to the diverse needs of its users. This includes features such as user profiles, news feeds, messaging, photo and video sharing, groups and communities, and event management. The website will prioritize user experience by providing intuitive navigation, responsive design, and personalized content recommendations.

4.Security and Privacy: Given the sensitivity of personal information shared on social media platforms, security and privacy considerations are paramount. The website will implement robust

security measures, including encryption protocols, secure authentication mechanisms, and privacy settings, to protect user data and ensure user confidentiality.

5. Scalability and Flexibility: As the social media website is expected to attract a growing user base and evolve over time, scalability and flexibility are crucial aspects of its design. The website will be built with scalable architecture and adaptable infrastructure to accommodate increasing user traffic and support future enhancements or integrations with external platforms or services.

2.1 Product Perspective

The quiz website developed using Django will prioritize delivering a seamless and engaging user experience for both quiz creators and participants. Key features will include robust search functionality with advanced filtering options to help users discover quizzes relevant to their interests efficiently. Quiz creators will have the ability to design detailed quizzes, incorporate multimedia elements, and track quiz engagement, while participants will benefit from interactive quiz experiences and personalized recommendations.

The website will prioritize scalability and flexibility, enabling future enhancements and integration with emerging technologies to adapt to evolving trends in the quiz landscape. Overall, the quiz website will serve as a comprehensive platform that fosters knowledge sharing, entertainment, and community engagement, ultimately enhancing the overall quiz experience for all users involved.

2.2 Product Functions

The quiz website developed using Django will encompass the following core functions:

1. User Registration and Authentication:
 - Allow users to register accounts by providing necessary information.
 - Authenticate users securely during login using credentials.
2. Quiz Creator Features:
 - Create quizzes by specifying quiz title, description, and questions with multiple-choice or open-ended formats.
 - Add multimedia elements such as images or videos to enhance quiz content.
 - Set quiz parameters such as time limits, difficulty levels, and scoring methods.
3. Participant Features:
 - Browse and search for quizzes based on various criteria such as topic, difficulty, or popularity.
 - View detailed information about quizzes including descriptions, number of questions, and average ratings.
 - Participate in quizzes by answering questions and receiving instant feedback on their performance.
4. Administrator Functions:
 - Manage user accounts, including creating, editing, and deactivating accounts.
 - Monitor quiz content and user interactions to ensure compliance with community guidelines.

- Address user inquiries and resolve any issues related to account management or quiz usage.

2.3 User Characteristics

For a social media website using Django:

Users:

1.Regular Users:

- Create profiles with personal information, interests, and preferences.
- Connect with friends and followers, share content, and engage in conversations.
- Customize privacy settings to control visibility of profile information and posts.

2.Content Creators:

- Publish and share multimedia content such as photos, videos, and stories.
- Build a following and engage with audience through likes, comments, and direct messages.
- Access analytics tools to track content performance and audience demographics.

Administrators:

1.CRUD Operations:

- Manage user accounts, including registration, verification, and account suspension.
- Review and moderate user-generated content to enforce community guidelines and policies.
- Monitor platform activity and address user complaints or disputes.

2.Dynamic UI Management:

- Customize the website's layout, themes, and features to enhance user experience and engagement.
- Implement responsive design principles to ensure seamless user interaction across devices and screen sizes.

3.Website Metadata Control:

- Optimize metadata such as page titles, descriptions, and tags to improve search engine visibility and click-through rates.
- Implement meta tags and schema markup to enhance social media sharing and integration with external platforms.

2.4 General Constraints

The development of the job application portal is constrained by:

1.Technology Stack:

- Developed using Django web framework in Python.
- Integration with HTML, CSS, and JavaScript for frontend.

2.Database:

- Relational database management system (RDBMS) like PostgreSQL or MySQL.

3.Scalability:

- Architecture must accommodate future scalability needs.
- Consideration of cloud infrastructure for scalability.

4.Security:

- Adherence to industry-standard security practices.
- Encryption, secure authentication, and regular security audits.

5.Regulatory Compliance:

- Compliance with data protection and privacy laws.
- Adherence to industry-specific regulations.

6.User Experience:

- Responsive and intuitive user interface design.
- Optimization of page load times and system performance.

7.Resource Allocation:

- Sufficient resources including time, budget, and personnel.
- Optimization of resource utilization.

8.Third-Party Integration:

- Integration with third-party services and APIs.
- Consideration of third-party limitations and constraints.

2.5 Assumptions and Dependencies

The requirements stated in this SRS are based on the following assumptions and dependencies:

1.Availability of Required Hardware and Software:

- Assumption: The hardware and software environments necessary for deploying the job application portal will be available as specified.
- Dependency: Any changes or discrepancies in the availability of hardware or software may require adjustments to the system requirements and configurations.

2.Internet Connectivity:

- Assumption: Users will have access to stable internet connectivity to interact with the job application portal.
- Dependency: Reliability of the portal's functionality is dependent on uninterrupted internet access for users.

3.Compliance with Third-Party APIs and Services:

- Assumption: Integration with third-party services and APIs will function as expected.
- Dependency: Any changes to the functionality or availability of third-party APIs and services may impact the functionality of the job application portal and require corresponding modifications.

4.User Data Privacy and Security:

- Assumption: Adequate measures will be in place to ensure the privacy and security of user data collected and processed by the job application portal.
- Dependency: Adherence to data protection regulations and industry standards is necessary to maintain user trust and legal compliance.

5.Stakeholder Collaboration and Feedback:

- Assumption: Stakeholders will actively participate in the requirements gathering process and provide timely feedback during development.
- Dependency: The accuracy and completeness of the SRS rely on effective communication and collaboration with stakeholders throughout the project lifecycle.

Here's how you can structure these sections for your Django job application portal:

3.1 External Interface Requirements

3.1.1 User Interfaces

- The user interface of the job application portal should be intuitive and easy to navigate for both job seekers and employers.
- It should support various functionalities such as user registration, job search, application submission, and job listing management.

3.1.2 Hardware Interfaces

- The job application portal should be compatible with standard hardware configurations commonly used for web browsing, including desktop computers, laptops, tablets, and smartphones.
- No specific hardware interfaces are required.

3.1.3 Software Interfaces

- Integration with a relational database management system (RDBMS) such as PostgreSQL or MySQL for data storage.

- Utilization of Django's built-in authentication system for user authentication and authorization.

3.1.4 Communications Interfaces

- The portal should support communication between users (e.g., messaging between employers and job seekers).
- Integration with email services for notifications and communication with users.

3.2 Functional Requirements

3.2.1 Job Listing Management

- **Introduction:** Employers should be able to post, edit, and remove job listings.
- **Inputs:** Job details including title, description, requirements, and application instructions.
- **Processing:** Validation of inputs, storing job listings in the database.
- **Outputs:** Published job listings visible to job seekers.
- **Error Handling:** Notify users of errors during input validation or database operations.

3.2.2 Job Search and Application

- **Introduction:** Job seekers should be able to search for job listings and submit applications.
- **Inputs:** Search criteria (e.g., job title, location), application details (resume, cover letter).
- **Processing:** Matching search criteria to job listings, storing application details.
- **Outputs:** Display search results, confirmations for successful application submissions.
- **Error Handling:** Notify users of search query errors or application submission failures.

3.5 Non-Functional Requirements

3.5.1 Performance

- Job search results should be returned within 2 seconds.
- Application submission should not take more than 5 seconds to process.

3.5.2 Reliability

- The job application portal should have a minimum uptime of 99.9%.
- It should be capable of handling concurrent user interactions without system crashes.

3.5.3 Availability

- The portal should be accessible 24/7 except during scheduled maintenance windows.

3.5.4 Security

- User data should be encrypted during transmission and storage.
- Access to sensitive data and functionalities should be restricted based on user roles and permissions.

3.5.5 Maintainability

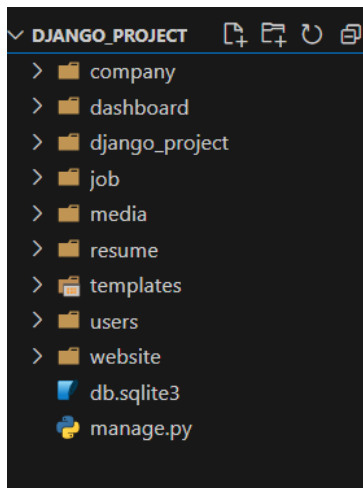
- Code should be well-organized and documented to facilitate future updates and maintenance.
- Regular code reviews and testing should be conducted to identify and address issues promptly.

3.5.6 Portability

- The job application portal should be compatible with major web browsers (Chrome, Firefox, Safari) and operating systems (Windows, macOS, Linux).

3.7 Design Constraints

- Compliance with Django's MVC (Model-View-Controller) architecture.
- Adherence to Django's coding standards and best practices.
- Utilization of Django's ORM (Object-Relational Mapping) for database interactions.



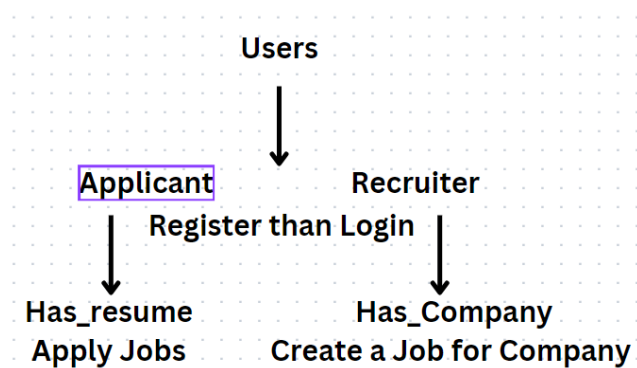
```
users > models.py > ...
1  from django.db import models
2  from django.contrib.auth.models import AbstractUser
3
4  class User(AbstractUser):
5      email = models.EmailField(unique=True)
6      is_recruiter=models.BooleanField(default=False)
7      is_applicant=models.BooleanField(default=False)
8
9      has_resume=models.BooleanField(default=False)
10     has_company=models.BooleanField(default=False)
11
12
13
```

```
company > models.py > Company
1 from django.db import models
2 from users.models import User
3
4 class Company(models.Model):
5     user=models.OneToOneField(User,on_delete=models.CASCADE)
6     name=models.CharField(max_length=100,null=True,blank=True)
7     est_date=models.PositiveIntegerField(null=True,blank=True)
8     city=models.CharField(max_length=100,null=True,blank=True)
9     state=models.CharField(max_length=100,null=True,blank=True)
10
11     def __str__(self):
12         return self.name
```

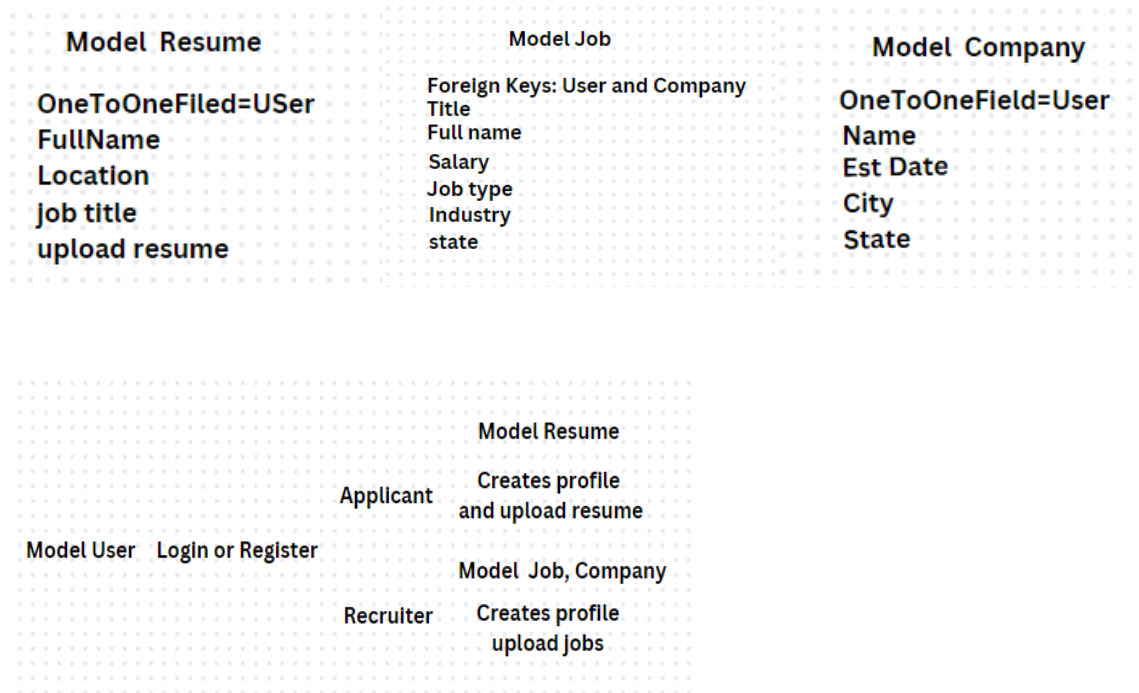
```
resume > models.py > Resume
1 from django.db import models
2 from users.models import User
3
4 class Resume(models.Model):
5     user=models.OneToOneField(User,on_delete=models.CASCADE)
6     first_name=models.CharField(max_length=100,null=True,blank=True)
7     surname=models.CharField(max_length=100,null=True,blank=True)
8     location=models.CharField(max_length=100,null=True,blank=True)
9     job_title=models.CharField(max_length=100,null=True,blank=True)
10     upload_resume=models.FileField(upload_to='resume',null=True,blank=True)
11
12     #insert cv
13     def __str__(self):
14         return f'{self.first_name} {self.surname}'
15
16
```

```
class Job(models.Model):
    job_type_choices=(
        ('Remote','Remote'),
        ('Onsite','Onsite'),
        ('Hybrid','Hybrid'),
    )
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    company=models.ForeignKey(Company,on_delete=models.CASCADE)
    title=models.CharField(max_length=100)
    city=models.CharField(max_length=100)
    salary=models.PositiveBigIntegerField(default=35000)
    requirements=models.TextField()
    ideal_candidate=models.TextField()
    is_available=models.BooleanField(default=True)
    timestamp=models.DateTimeField(auto_now_add=True)
    industry=models.ForeignKey(Industry,on_delete=models.DO_NOTHING,null=True,blank=True)
    state=models.ForeignKey(State,on_delete=models.DO_NOTHING,null=True,blank=True)
    job_type=models.CharField(max_length=20,choices=job_type_choices,null=True, blank=True)

    def __str__(self):
        return self.title
```



Models:



Analysis Models for Django Job Application Portal

1. Use Case Diagram

- **Introduction:** The use case diagram for the Django job application portal illustrates the interactions between actors (job seekers, employers, administrators) and the system. It highlights the functionalities and features of the portal from a user's perspective.
- **Narrative Description:** Actors interact with the system through use cases such as "Search Job Listings," "Submit Application," "Post Job Listing," and "Manage User Account." These use cases represent the core functionalities of the portal as outlined in the SRS. The use case diagram aids in understanding how different actors engage with the system to achieve their goals, ensuring alignment with the specified requirements.

2. Sequence Diagram

- **Introduction:** Sequence diagrams for the Django job application portal illustrate the sequence of interactions between components such as views, models, and databases during specific scenarios or user actions.
- **Narrative Description:** For example, a sequence diagram depicting the process of job application submission would show interactions between the user interface, Django views handling form submissions, and database models for storing application data. It outlines the sequence of messages exchanged between these components, including user input, data validation, and database updates. This model helps validate requirements related to input processing, data flow, and system response times.

3. Entity-Relationship Diagram (ERD)

- **Introduction:** The ER diagram for the Django job application portal represents the structure of the database schema, including entities, attributes, and relationships between them.
- **Narrative Description:** Entities such as User, JobListing, Application, and Administrator are depicted, along with their attributes and relationships. For example, the ER diagram shows that a User can have multiple Applications, and a JobListing can have multiple Applications from different Users. It ensures that the database schema supports the data requirements specified in the SRS and facilitates efficient data management within the Django framework.

4. Class Diagram

- **Introduction:** Class diagrams for the Django job application portal illustrate the object-oriented design of the system, including classes, attributes, methods, and relationships.
- **Narrative Description:** Classes such as User, JobListing, Application, and Administrator are represented, along with their attributes and methods. Associations between classes, such as ownership of job listings by employers or relationships between users and applications, are depicted. This model provides a blueprint for implementing the system's functionality within the Django framework, ensuring that classes and their relationships align with the specified requirements in the SRS.

A. Appendices

A.1 Conceptual Documents

- This appendix includes conceptual documents such as system architecture diagrams, wireframes, and mockups. These documents provide a high-level overview of the system's design and functionality.

A.2 Marketing Materials

- This appendix contains marketing materials such as brochures, flyers, and promotional videos. These materials showcase the features and benefits of the job application portal to potential users and stakeholders.

A.3 Meeting Minutes

- This appendix includes minutes of meetings with the customer(s) and other stakeholders. It documents discussions, decisions, and action items related to the development of the job application portal, providing transparency and accountability throughout the project lifecycle.

A.4 User Feedback

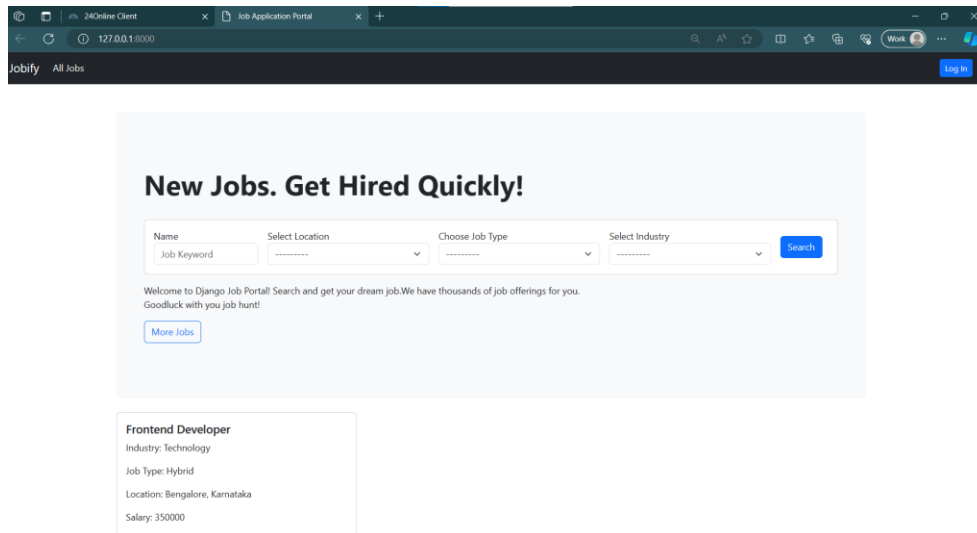
- This appendix contains feedback collected from users during the testing and evaluation phases of the project. It includes user surveys, usability testing reports, and user interviews, highlighting areas for improvement and informing future iterations of the portal.

A.5 Technical Documentation

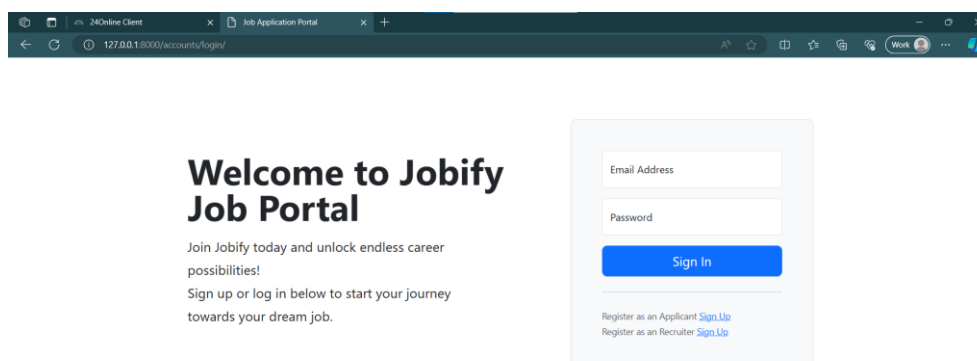
- This appendix includes technical documentation such as API specifications, database schemas, and deployment guides. It provides detailed information for developers, system administrators, and other technical stakeholders involved in the implementation and maintenance of the job application portal.

Output Screenshots:

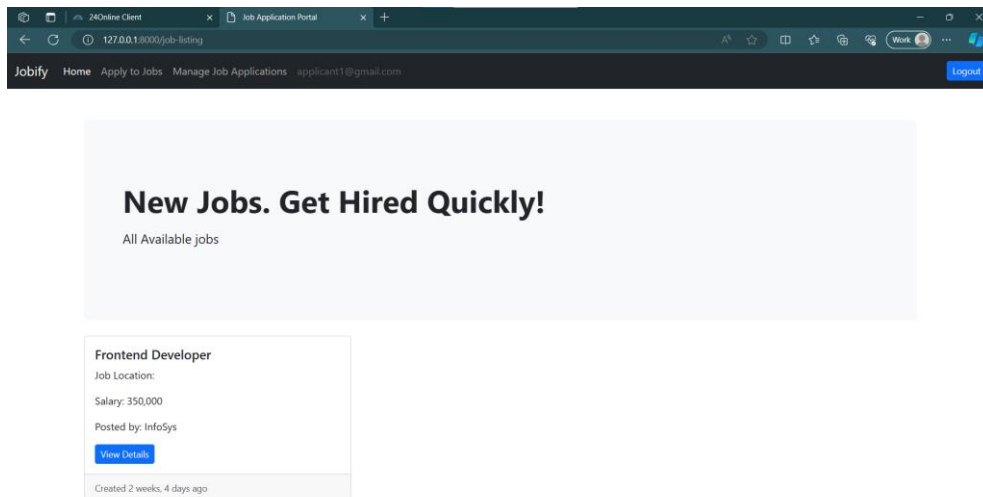
Home Page:



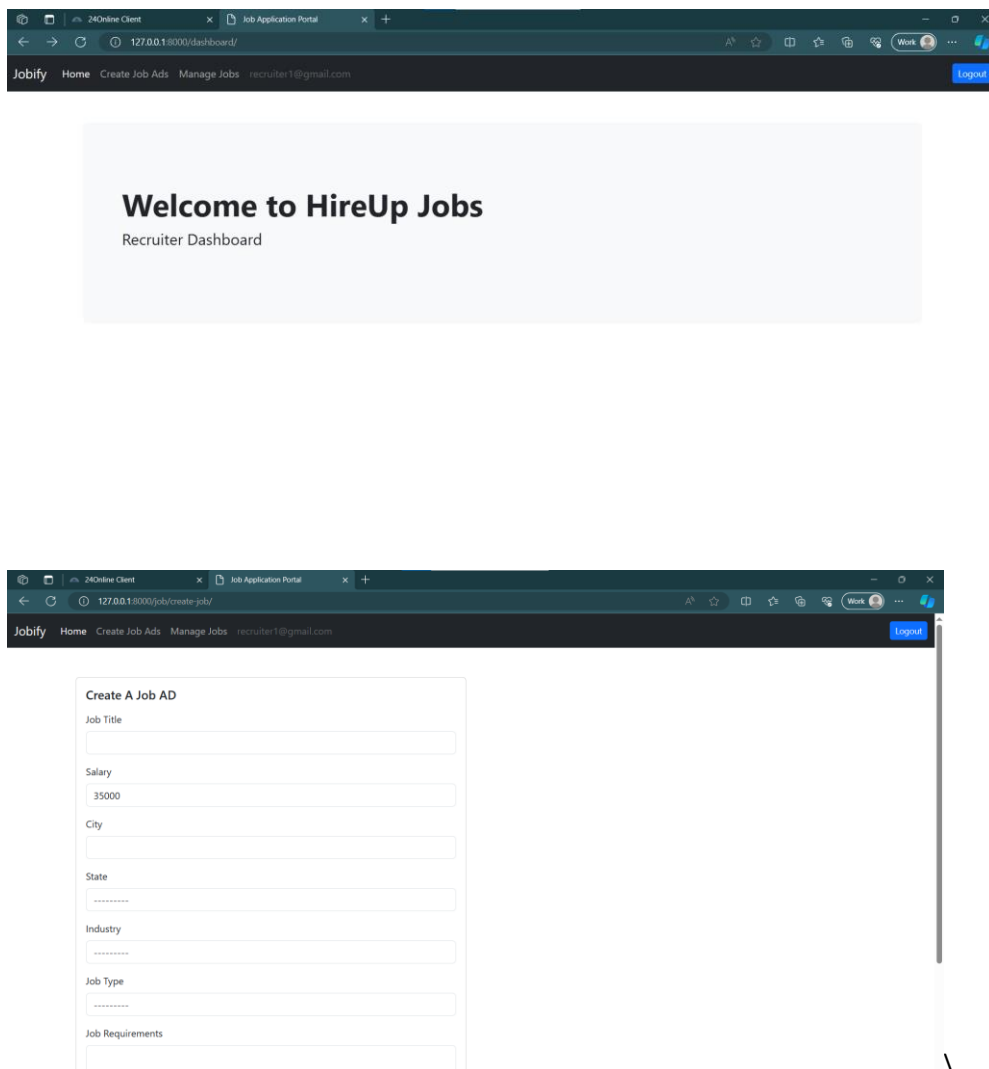
Signup/Login Page:



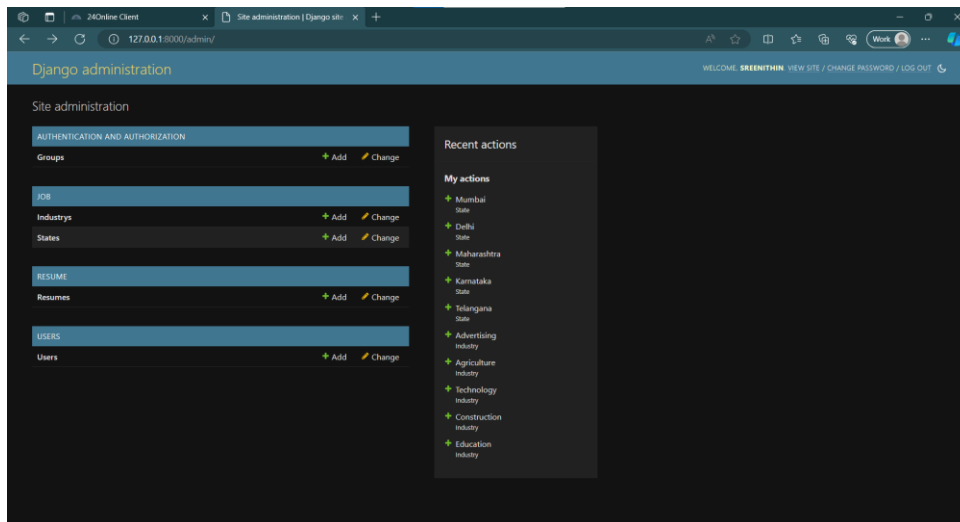
Applicant Dashboard:



Recruiter Dashboard:



Admin Portal



GitHub Link:

<https://github.com/kantesreenithin/Django-Job-Application-Portal>

Conclusion

The quiz website developed using Django serves as an essential platform for facilitating knowledge sharing, engagement, and entertainment among users. Its intuitive interface and diverse features streamline the quiz creation and participation process for creators and participants alike. By providing a centralized hub for discovering, creating, and participating in quizzes, the website contributes to fostering a vibrant and interactive quiz community.

However, to maintain its relevance and effectiveness in the evolving landscape of online quizzes, the website requires continuous refinement and enhancement. Addressing areas for improvement and implementing recommended enhancements can further elevate the website's impact, user satisfaction, and overall engagement.

Enhancements such as improving search functionality to deliver more relevant quiz recommendations and simplifying the quiz creation process for creators can enhance user

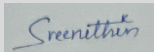
experience. Additionally, providing participants with personalized recommendations based on their interests and past quiz performance can increase user engagement and retention.

Moreover, embracing innovative features such as real-time multiplayer quizzes or integration with social media platforms can broaden the website's appeal and attract a wider audience. By staying responsive to user feedback and incorporating user-centric design principles, the website can continue to evolve and thrive as a leading destination for quiz enthusiasts seeking entertainment and enrichment.

Client Approval Form

PROJECT NAME	Quiz website		
JOB LOCATION	Jalandhar		
EST. START DATE	1 st MAR 2024	EST. FINISH DATE	22 st APR 2024
PROJECT LEADER	Shivam Nath	COMPANY	Jobify
CONTACT NAME	Nagarjuna	ADDRESS	Lovely Professional University
PHONE	9848780624		
EMAIL	nagarjuna@gmail.com		

SUMMARY	Job Application Portal Using Django
DESIRED OUTCOME	Need of Fully Functional Website along with backend
ACTION TO COMPLETION	Developed website using Django framework that fulfilled their needs.
BENEFITS OF PROJECT	Easy to manage and update in future with less to no expenditure.
PROJECTED SCHEDULE	Under one month
PROJECTED BUDGET	200-500 rupees
PROJECTED TEAM AND RESOURCE REQUIREMENTS	One person and a computer
PROPOSAL MAY BE WITHDRAWN IF NOT ACCEPTED BY DATE OF	
22 th APR 2024	

ACCEPTANCE OF PROPOSAL			
AUTHORIZED CLIENT SIGNATURE		DATE OF ACCEPTANCE	22 rd APR 2024