

CAREER-X

A Comprehensive Student Recruitment and Internship Management System

Sruthi Mandalapu

R11906160

Advanced Data Base Management Systems, CS5356

Table of Contents

1 Introduction	2
1.1 Team Composition	2
2 Project Description	2
2.1 Database Overview	2
2.2 Technical Specifications	2
2.3 Challenges and Solutions	4
3 Personal Contribution	4
3.1 Roles and Responsibilities	4
3.2 Work Process	4
3.3 Technical Implementation	5
4 Reflection	8
4.1 Learning Outcomes	8
4.2 Teamwork and Collaboration	8
4.3 Challenges Faced	10
4.4 Skills Developed	10
4.5 Future Application	13
5 Conclusion	13
6 Appendices	13
7 References	21

1. Introduction:

CareerX, which solely introduces an ease of access in perspective of a student, company and admin. In this project, we attempt to address the gap between the Job Seeker and the Recruiter. This is done by taking into consideration the details provided by both, the Job Seeker and the Recruiter, and, by applying a variety of different functionality in order to cater to each and everyone's individual needs and wishes [1]. We also introduce Admin who would exclusively monitor the student and company by typically posting some updates.

1.1 Team Composition:

As a team, 80 percent of UI part is taken care by one of the team mate Sankeerthana, it includes UI structuring for Student, Admin and Company. This uses complete structuring, styling and designing the pages. She was also involved in final testing the project. Other team mate Imran have contributed Company regard the backend logic. Imran taken care Internship, Jobs, Internship Applications, Job Applications backend logic. In later part, helped in preparing the structure of the database and prepared slides for presentation. I have contributed to make up entire view of initial part of application and in later part had given all crud operations for the all the models, that could be directly accessed to backend logic, by simply calling the functions. And helped loading all the dummy data to test the application. Juan have taken care of few merging aspects and used crud functionality and Imran backend logic, to make up entire backend logic for the application. He does also involve in testing the application and fixing the bugs.

2. Project Description

2.1 Database Overview

In our project, we have used sqlite as a database. This is an inbuilt database which already available in Django. This allows Object Relational Mapping to access the contents with in database. All the tables are solely described as Models. An object-relational application combines artefacts from both object and relational paradigms. Essentially an object-relational application is one in which a program written using an object-oriented language uses a relational database for storage and retrieval [2]. This ORM ensures consistency, reduces code, and supports relationships (e.g., One-to-One). Looking through the Django framework, models represent database tables, making database interactions seamless and efficient.

2.2 Technical Specifications

Below are the name of tables and column names listed. Every table performs either any of the operation "ADD, CREATE, UPDATE, DELETE, ACCESS". 'Table – 1' explains these 4 operations, where which User is allowed to perform what functionality is mentioned. And also, the overview of the database connections is explained in ER-diagram in 'Figure – 1'.

- Student: student_id, full_name, email, contact_number, date_of_birth, gender, r_number, department, cgpa, password
- Company: company_id, company_name, email, contact_number, street_number, city, state, country, pincode, password
- Admin: full_name, email, contact_number, age, gender, password.
- Internship: internship_id, internship_role, description, internship_type, location, stipend, start_date, duration_months, last_date_to_apply, posted_date, company(foreign key), created_by(foreign key).
- Internship Applications: internship_application_id, internship(foreign key), student(foreign key), date_of_applied, status
- Job: job_id, job_role, description, job_type, location, salary, start_date, last_date_to_apply, posted_date, company (foreign key), created_by (foreign key).

- Job Applications: job_application_id, job (foreign key), student (foreign key), date_of_applied, status.
- Notice: notice_id, announcement_text, created_by (foreign key), recipient (foreign key), date_created.
- Event: event_id, title, description, date, and location.

Users	Internship	Intern Applications	Job	Job Applications	Events	Notices
Student	View, Apply	NA	View, Apply	NA	View	View
Company	Add, View, Update, Delete	View, Update	Add, View, Update, Delet	View, Update	View	View
Admin	View	NA	View	NA	Add, View, Update, Delete	Add, View, Update, Delete

Table: 1

The above table – 1 describes the permissions and functionalities assigned to different user roles (Student, Company, Admin) across various modules in a system. Students can view and apply for internships and jobs, view events, and notices but cannot modify or manage any content. Companies have broader permissions; they can add, view, update, and delete internships, job listings, and their applications, while also viewing events and notices. Admins hold the highest level of control, being able to view internships and jobs but having full rights to manage events and notices by adding, viewing, updating, and deleting them. This hierarchy reflects a role-based access control system, where actions are limited based on the user type. Each module's operations align with the responsibilities of the associated role, ensuring organized management. Overall, the table defines the functional boundaries and ensures data integrity by restricting sensitive actions to authorized roles.

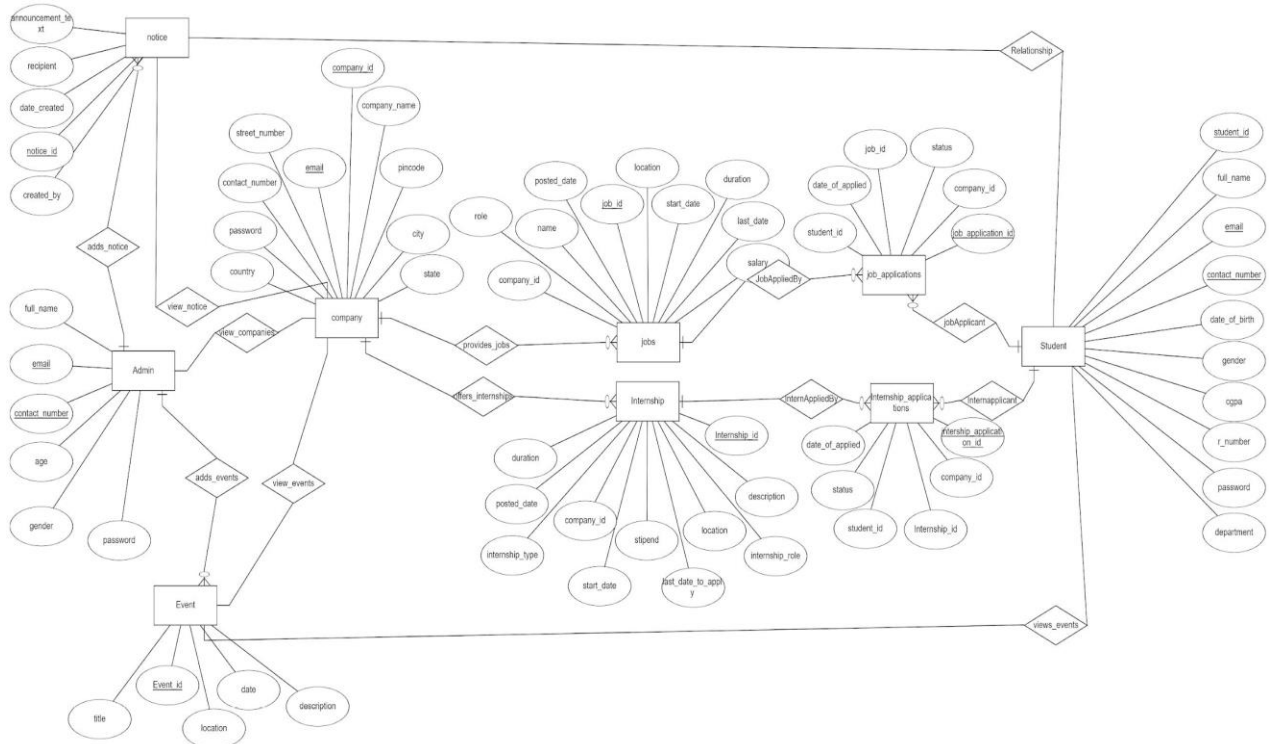


Figure: 1

ER diagram in general scenario illustrates the relationships between different entities, our system is designed for managing students, internships, jobs, events, and notices. The primary entities are Student, Admin, Company, Internship, Job, Event, and Notice, each linked by relationships and defined by their attributes.

The student entity contains attributes such as `student_id`, `name`, `email`, `password`, and others, representing personal and login information. Students are connected to Internship and Job entities through applications, indicating they can apply for internships and job roles. The Internship and Job entities include attributes like `job_id`, `title`, `location`, and application-specific details. The relationship between students and these entities is further detailed with attributes such as `status` and `application_id` to track applications.

The company entity is linked to Internship and Job entities through relationships that allow companies to manage these postings. Each company has attributes like `company_id`, `name`, `email`, and `contact_number`, allowing them to perform operations like creating, updating, or deleting their listings.

The admin entity plays a supervisory role and is connected to the Event and Notice entities. Admins can create, update, and delete events and notices, which are visible to students and companies. The Event entity contains attributes like `event_id`, `title`, `date`, and `location`, while the Notice entity includes details such as `notice_id`, `subject`, and `description`.

Overall, the above diagram represents a structured hierarchy of roles and permissions. Students interact with internships, jobs, and events in a limited capacity. Companies manage their own postings while interacting with students. Admins oversee the system, ensuring proper management of events and notices. Relationships, denoted by diamond shapes, define the nature of interactions between entities, supported by attributes for further detailing. This ER diagram showcases the interconnected functionality and role-specific responsibilities in the system.

2.3 Challenges and Solutions

When we consider the database, initially learning the things from start, understanding the workflow of object relational model had taken lot of time. Later, I have used 'Python Shell' which is as an ease of access. This does directly give the output at each line. Initially, I used to write the entire backend functionality and try to access the data based on operation required. This would lead a bit confusion. Hence, I had manually written all the crud functionality for each table. So, that this doesn't require to write special logic for everything to access from the database, as it is already written, without any delay we could directly call the functionality.

3. Personal Contribution

3.1 Role and Responsibilities

Considering the overall application, I have dedicated my role to finish front-end, back-end, database and loading the test set for the initial part of application. For the next part, I have worked on covering each database table crud operations. These are key part in accessing the data to the application. Lastly, I have tested the entire application by adding sample data to the application.

3.2 Work Process

As a part of every application in daily life, each application asks for user to Login, Register, and reset the password. This is a basic key part for every application. Similarly, the basic UI for the application is really important. This is used as a base to design the all pages to be alike. All the pages use the similar formatting style. This enhances look and feasibility of the application. So, as a kick start for the application I have worked on UI formatting that would be taken as a base for the entire application. Successively, after having an idea related to UI part, I have started formatting the fields, buttons within the pages. Then, after entire front part is designed successfully. I started the backend, linking each field by accessing, updating the data. Here, I have finished the initial part of application which includes "Index, Login, Student Registration, Company Registration, Reset Password" pages explain about entire front-end, back-end functionality. I have loaded basic 2 rows to test these pages, which are successfully working on test data.

For Company, Student, Admin one of my team-mates contributed front end pages. For each page with the respective Models, I have designed CRUD operations for each table. These are crucial part when a user is trying to build any application. Any application needs data to be either created, accessed, added, updated or deleted. So, I have contributed each model performing all the creating, accessing, adding, updating, deleting functionality. In order, to test the application, I have loaded with sample dataset with 10 to 20 rows based on the model.

During the work of backend, I have tested the application frequently to make sure all the functionality is working fine. I reported few bugs, which is not working properly. These bugs are assigned to everyone and is resolved in later part of testing. Below are things I have made more insights while I was working through the application.

3.3 Technical Implementation

UI Formatting:

We have used HTML, CSS, Bootstrap for styling the webpage, and small validations using javascript. Below are the set of UI pages I have worked:

Filed	Description	Format
Textbox	Asks for user to enter details, based on requirement	<input type="text">
Radio Button	It is a selective choice, user need to select one among all options	<input type="radio">
Button	It is allows forms to perform an action like submit/redirecting	<input type="button">

Table: 2

- Index Page:
 - Buttons: Login, Student Registration, Company Registration
- Login Page:
 - Textboxes: Username, Password
 - Radio buttons: Login Type (Student, Company, Admin)
 - Button: Login, Cancel, Can't Access Account
- Reset Password:
 - Textboxes: Email address, New password, Confirm new password
 - Radio buttons: User type (Student, Company, Admin)
 - Buttons: Reset, Go back
- Student Registration:
 - Textboxes: Full name, Email address, Contact number, Date of birth, Gender, R-Number, Department, CGPA, Password, Confirm Password
 - Button: Register, Cancel
- Company Registration:
 - Textboxes: Company name, Email address, Contact number, Street number, City, State, Country, Pincode, Password, Confirm password
 - Button: Register, Cancel
- Javascript Validations:
 - When we observe reset password, student register, company register asks for password and confirm password. Both the fields should be validated as same, so we have used `document.getElementById('id_name').value` which retrieves the value in that field, if both the fields have same value, the flow works fine, able to do the functionality, in other scenario we have produced an alert message "Mismatch".

Backend Resources:

We have used python, django as backend framework. Diving through the backend, I would explain in 2 things:

- Views
 - This allows in connecting the database models with UI part, and allows in communication acting as a middle ware between UI and database.
 - In order to retrieve the html file we use `render(request, "name_of_file.html")`
 - As Django is Object Relational Mapping, it doesn't need any special connections for the database. We can access the tables in the form of Models. Usually database tables are defined as models in ORM.
- Urls
 - For routing each page, we need connections to url that is loaded within the web page
 - For a login page it is defined as: `path('login/', views.login_view, name='login')`

Database Models:

- Database Models are declared like a class in python, but inherits class 'Model'. The below is the Model for creating the table Student:

```
class Student(models.Model):
    student_id = models.IntegerField(primary_key=True) #custom primary Key
    full_name = models.CharField(max_length=100)
    email = models.CharField(max_length=100, unique=True)
    contact_number = models.BigIntegerField()
    date_of_birth = models.DateField()
    gender = models.CharField(max_length=100)
    r_number = models.CharField(max_length=100)
    department = models.CharField(max_length=100)
    cgpa = models.CharField(max_length=100)
    password = models.CharField(max_length=100)
```
- As we can see in the above Student Model, `student_id`, `full_name`, `email`, `contact_number`, `date_of_birth`, `gender`, `r_number`, `department`, `cgpa`, `password` are the column names of the Student Model. And also, on observation type of field is mentioned, restricting the length along with constraints as primary key, foreign key or unique.

CRUD Operations:

Create, Read, Update, and Delete (CRUD) are the four basic functions that models should be able to perform. Below is the brief explanation of each:

- Create:
 - Description:
 - In databases, the Create operation involves inserting a new record into a table. This process typically requires specifying values for each field in the table. For instance, in a relational database, a SQL INSERT statement accomplishes this task. The database management system (DBMS) ensures that the new record adheres to the defined schema and constraints, such as primary keys and foreign keys.
 - Examples:
 - Adding a new user to a web application's database.
 - Inserting a new product into an inventory management system.
 - Creating a new blog post in a content management system.
 - Functionality:
 - As Django uses ORM for accessing the database. Here is how we insert the data into the model
 - `new_student = Student (student_id='---', full_name='---', email='---', contact_number='---', date_of_birth='---', gender='---', r_number='---', department='---', cgpa='---', password='---')`
 - `new_student.save()`

- Read:
 - Description
 - In databases, the Read operation involves executing a query to fetch data from one or more tables. For relational databases, a SQL SELECT statement performs this function. The DBMS processes the query and returns the requested data, ensuring that it meets any specified criteria or filters.
 - Examples:
 - Retrieving a user's profile information from a social media platform.
 - Displaying a list of products in an online store.
 - Fetching transaction history for a bank account.
 - Functionality:
 - As Django uses ORM for accessing the database. Here is how we read the data into the model
 - `student=Student.objects.get(student_id=student_id)`
 - `full-name = student.full_name, email = student.email, date_of_birth = student.date_of_birth, gender = student.gender, r_number = student.r_number, department = student.department, cgpa = student.cgpa`
- Update:
 - Description
 - In databases, the Update operation involves altering one or more fields of an existing record. For relational databases, a SQL UPDATE statement performs this task. The DBMS ensures that the updated record adheres to the defined schema and constraints, maintaining data integrity.
 - Examples:
 - Changing a user's email address in a web application's database.
 - Updating the price of a product in an inventory management system.
 - Modifying the content of a blog post in a content management system.
 - Functionality:
 - As Django uses ORM for accessing the database. Here is how we update the data into the model
 - `student=Student.objects.get(student_id=student_id)`
 - `student.full_name = full_name , student.email = email, student.date_of_birth = date_of_birth, student.gender = gender, student.r_number = r_number, student.department = department, student.cgpa = cgpa`
 - `student.save()`
- Delete:
 - Description
 - In databases, the Delete operation involves executing a command to remove one or more records from a table. For relational databases, a SQL DELETE statement performs this task. The database management system (DBMS) ensures that the deleted record no longer exists in the table while maintaining the integrity of the remaining data.
 - Examples:
 - Removing a user's account from a web application's database.
 - Deleting an out-of-stock product from an inventory management system.
 - Erasing an outdated blog post from a content management system.
 - Functionality:
 - As Django uses ORM for accessing the database. Here is how we delete the data into the model
 - `student=Student.objects.get(student_id=student_id)`

- `student.delete()`

Data Loading:

As a part of testing I have loaded basic dummy data, this allows to verify the flow of application is working fine or not, ensuring the feasibility of the application. Below are the things to be followed:

- Perform migrations
 - Migrations the terms itself says moving or applying, after scripting a model in backend, we need to make it functionally work, by performing migrations, below is how we make migrations
 - Make migrations: `python manage.py makemigrations`
 - Apply migrations: `python manage.py migrate`
 - Verify migrations: `python manage.py showmigrations`
- Usage of Python Shell
 - Open the shell: `python manage.py shell`
 - Query the database: `from your_app.models import YourModel` → `from home.models import Student`
 - Load the data: Use this, load the data using shell prompt


```
dummy_students = [
    {
        "student_id": 1, "full_name": "Alice Johnson", "email": "alice@gmail.com",
        "contact_number": 1234567890, "date_of_birth": date(2000, 1, 1), "gender": "Female",
        "r_number": "R001", "department": "Computer Science", "cgpa": "3.8", "password": "password1"
    },
    {
        "student_id": 2, "full_name": "Bob Smith", "email": "bob@gmail.com",
        "contact_number": 1234567891, "date_of_birth": date(2001, 2, 2), "gender": "Male",
        "r_number": "R002", "department": "Mech Engineering", "cgpa": "3.5", "password": "password2"
    }
]
for student_data in dummy_students:
    student = Student(**student_data)
    student.save()
```

4. Reflection

4.1 Learning Outcomes

As this a team project, I have learnt team work, manageability of work, progress, reporting the issues, maintaining the updates. All these things are planned and started developing the application. As, we all know git is the crucial part while working on any project, and also everyone at least knew basic git resources, we could ensure enhancing the skills to access the git and explore the git functionalities much more. Apart from these, when I look technically, as a Django is completely new technology, I have watched number of videos and multiple online resources to work on this application. This had helped me to learn new technology.

4.2 Teamwork and Collaboration

As a team we have collaborated frequently ensuring the progress of the team and the work flow, we had a regular group meeting, and also in situation like a testing, or having any bugs – we used to have a meeting with each individual if any part of the workflow not going fine. Below is the collaborative work for the pages we worked on:

Index page: The index page contains 3 set of buttons, Login, Student Registration, Company Registration. Based on the button opted, it redirects to respective pages like Login, Student Registration and Company Registration. This is the initial page loaded when the web app runs.

Login page: This page contains set of fields like email and password and type of the user. All the fields are required, the user need to enter all the fields. If the fields given by user are correct it redirects to respective login page it might be Student dashboard, Company dashboard, Admin dashboard. If the fields are incorrect, it redirects to an error message. This does also contain buttons, if user forgot the password, he could able to reset the password by clicking “Can’t Access Account”. If the user wants to go back to the index page, he simply clicks on “cancel” button.

Student Registration Page: The student registration page contains set of basic details to be entered by the student. All the fields are required, if fields are not entered, or in scenario when confirm password, password are not matched throws a javascript validations message to ensure to provide a validated information. On giving successfully correct information redirects to login and asks the student to login.

Company Registration Page: This does also give similar functionality just like that of Student Registration Page, it does also ask for company basic details, even here all the fields are required and it does also contain javascript validation. On successful registration from the company, asks the company to login.

Password Reset Page: It allows the user in scenarios to reset if user forgets the password. It asks for new password, confirm new password. If both passwords are same, it redirects to login page. This page also contains back button, if the user wants to get back to the index page or login page.

Student Dashboard Page: This page contains set of buttons that are accessible to the student. It contains profile, internship, jobs, notices, companies, events and logout. Based on the button, the web app redirects to respective pages.

Student Profile Page: This page allows the student to view his/her details. The student could able to view the details along with that, he could also able to update the details if necessary. Go back button in Student profile page redirects back to dashboard.

Student Internship Page: It allows in viewing the basic details of the internship “Internship id, role, description, duration, type, location, stipend, start date, Apply by, company name, actions”. Actions gives the status ‘expired, apply, pending and approved’. Search by role allows to search the internships based on roles. Back to dashboard gets back to dashboard page.

Student Job Page: Just like internship, it allows in showing the basic details of the job and if student is interested, he would apply to that job. It does also contain search bar and a button to get to dashboard. As above it also contains similar actions.

Student Notices and Events Page: The student receives the notices from the admin. These are posted by admin. The student can only view them. Student has no other operations on it. Similarly, the student can only view the events. This page describes any ongoing events like workshops and conferences. These are also posted by admin. Student can only view these events, he cannot make operations on these events.

Company Dashboard Page: The company dashboard displays set of buttons where company is accessible to make actions. It provides company to view profile, internships, jobs, notices, applicants, events and logout. Based on the button, the web app redirects to respective pages.

Company Profile Page: This page allows the company to view his/her details. The company could able to view the details along with that, he could also able to update the details if necessary. Go back button in Company profile page redirects back to dashboard.

Company Internship Page: It allows in viewing the basic details of the internship “Internship id, role, description, duration, type, location, stipend, start date, Apply by, company name, actions”. Actions gives operations like edit and delete for the internship posted by that company. The company could also add new internship. Search by role allows to search the internships based on roles. Back to dashboard gets back to dashboard page.

Add Internship Page: This page asks for internship role, description, duration, type, location, stipend, start date, Apply by. This page is allowed only by a company. This adds a new internship.

Company Job page: It allows in viewing the basic details of the in job “Job id, role, description, duration, type, location, stipend, start date, Apply by, company name, actions”. Actions gives operations like edit and delete for the internship posted by that company. The company could also add new job. Search by role allows to search the jobs based on roles. Back to dashboard gets back to dashboard page.

Add Job Page: This page asks for job role, description, duration, type, location, stipend, start date, Apply by. This page is allowed only by a company. This adds a new job.

Company Applicants Page: This page contains Type, Role, Applicant, Date Applied, Status. Status can be accepted or pending. On changing the status user should click the save button which enables in saving the changes. Back to dashboard buttons redirects back to dashboard.

Admin Dashboard Page: The admin dashboard displays set of buttons where admin is accessible to make actions. It provides admin to view profile, internships, jobs, notices, applicants, events and logout. Based on the button, the web app redirects to respective pages.

Admin Events and add event Page: The admin could view the event details, and also admin could able to add, edit and delete the events. Add event page asks for event title, description, date and location. After giving all the details, new event gets updated, all the users “student, company, admin” can view newly updated event. Update event page also provides similar function, it allows to edit the existing added event, where as delete button directly deletes the event.

Admin View Companies, Students Page: Admin can only view the companies’ basic details like company id, company name, email, country. Similarly, the admin can view the student basic details also.

Admin Notifications Page: It provides add, update and delete notices. Add notice allows in adding a new announcement. Update notice allows in updating the notice. And delete notice removes a particular notice.

Admin Internship, Job Pages: Admin can view all the jobs and internships posted by the companies and their details. Admin can only view these pages. Admin has no operations like update or delete.

4.3 Challenges Faced

This Django framework is completely a new technology. I have learnt from scratch, initially after declaring the templates, in order to ensure css styling is working, the page should be declared with {% static %}, I have faced few issues and took me long time, later it was fixed. After testing, the application is encountered with many bugs, it took longer time to fix the UI and pages to redirect. As it is an existing part, all the flow went correctly, but to identify and rectify the issues had taken me long time than expected. Later, after resolving the issue, I found to be too simple.

4.4 Skills Developed

I have learnt a new technology, usage of Django for backend development. As Django uses ORM which is Object Relational Mapping, this is a new framework for accessing the database. I have learnt using orm and its

functionality of accessing the data. Here is my step-by-step process of new learning aspects and insights while working in this web-app:

Step1: Learning Django and its installations in python

In my system I have already python installations, to ensure the Django framework to be installed and able to run, we use the following commands:

- pip install Django
- django-admin --version
- django-admin startproject myproject
- cd myproject
- python manage.py runserver
- Visit - <http://127.0.0.1:8000/>

Django is installed using pip, Python's package manager, which fetches the framework and its dependencies. On running the command pip install Django, allows in fetching the latest version of Django. The Django installation is verified by checking its version. Using the django-admin tool with the startproject command, a new project is initialized, creating essential configuration files and the required directory structure. The development server is started using python manage.py runserver, making the project accessible locally for further development and testing. This allows access to the application through a browser at the default address <http://127.0.0.1:8000/>. The development server facilitates testing changes in real time without needing deployment. It auto-reloads with every modification, ensuring a seamless development experience. However, this server is meant only for development and not for production environments.

Step2: MVT Usage

Django is a web framework that uses the MVT architecture to simplify web development. After installing Django, we start by creating a project and running a server to test the setup. Apps are created within the project, and linked to the database by defining models and running migrations. The framework integrates templates, views, and URLs to handle the application's frontend and backend logic. The Django shell allows quick database operations and debugging. This setup streamlines development by providing robust tools for efficient application building.

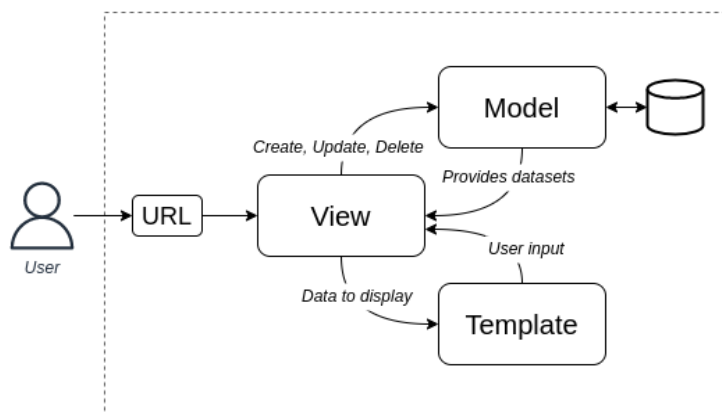


Figure: 2

The diagram represents the Model-View-Template (MVT) architecture, which is the core pattern used in Django for developing web applications. The flow begins when a user makes a request by entering a URL in their browser. This request is then processed by the URL system, which matches the requested URL to a specific view. The view acts as the controller in the MVT pattern, receiving the user's request, handling any necessary logic, and interacting with the model to retrieve or update data. The model is responsible for managing the application's data, which typically involves querying the database or performing data operations like creating, updating, or deleting records.

Once the view has the necessary data, it passes it to the template, which is responsible for rendering the user interface. The template takes the data provided by the view and generates the appropriate HTML or other formats for display on the user's screen. In this flow, the model interacts with the database to perform the actual data operations, ensuring that the view is working with up-to-date information.

This separation explains - models handling data, views managing logic, and templates focusing on presentation - ensures a clean and maintainable architecture in Django applications. It allows developers to efficiently manage the application's components and improve the overall maintainability of the codebase.

Step3: Templates and Static files for styling

In Django, templates for HTML files are a fundamental part of rendering web pages dynamically. HTML templates are used to design the structure of the website, while allowing the integration of dynamic content using Django's template language. These templates contain standard HTML code alongside placeholders (using double curly braces like `{{ }}`) and control structures (like `{% for %}` and `{% if %}`). These curly braces are used when displaying some content on webpage. It allows placeholders like `{{ variable }}` to display data and `{% logic %}` tags for conditions or loops. For example, job listings fetched from the database can be rendered in a list format, and a message like "No jobs available" can be shown if no data exists.

Static files, on the other hand, refer to CSS, JavaScript, and images that are used to style and enhance the user interface. These files remain unchanged and are served directly to the user's browser. Django organizes static files within a static directory, and they are referenced in templates using Django's `{% static %}` template tag. This allows to manage and include assets like custom CSS styles or JavaScript code seamlessly.

By combining templates and static files, Django ensures a clean separation of concerns, allowing developers to focus on backend logic while maintaining an appealing and responsive user interface. Templates generate the structure and dynamic content, while static files handle the aesthetics and functionality of the site. Together, they create a cohesive and interactive web application.

Step4: ORM Usage and depth understanding

Object/Relational Mapping (ORM) provides a methodology and mechanism for object-oriented systems to hold their long-term data safely in a database, with transactional control over it, yet have it expressed when needed in program objects. Instead of bundles of special code for this, ORM encourages models and use of constraints for the application, which then runs in a context set up by the ORM. Today's web applications are particularly well-suited to this approach, as they are necessarily multithreaded [3].

Django's ORM allows seamless interaction with the database using Python code instead of SQL. For example, a Student model defines fields like name, age, email, and grade, representing columns in the database. This can create, retrieve, update, or delete records effortlessly. For instance, a new student can be added using `Student(name="Alice", age=20, ...)` followed by `.save()`, and records can be fetched with `Student.objects.all()`. Similarly, updates and deletions are performed by modifying or removing objects. The ORM automatically handles these operations, converting Python commands into SQL queries, enabling efficient database management without writing raw SQL.

Step5: Collaborating the front-end, back-end and database Models to work

Collaborating the front-end, back-end, and database models in Django involves integrating templates, views, and models to create a dynamic web application. The front-end is built using HTML templates, styled with CSS, and optionally enhanced with JavaScript. These templates are rendered by views, which act as the intermediary between the front-end and back-end logic. The back-end handles user requests through URL routing and processes them using views, which interact with the database models. Models define the structure of the data, and Django's ORM manages database queries seamlessly. For example, when a user submits a form, the view processes the input, interacts with the model to save or retrieve data, and renders the appropriate template to display results. This collaboration ensures a smooth workflow where the front-end, back-end, and database work to deliver a functional web application.

4.5 Future Application

As this is a job portal application, where student can look through internships and jobs easily, we could also create a collaborated setup of conducting interview panel in the same interface by allowing a face-to-face conversation. In database perspective, we could extend this allowing to fill necessary details on clicking button “apply”. On clicking apply allowing the student to upload resume, transcripts and few basic details that are required for a specific company.

5. Conclusion

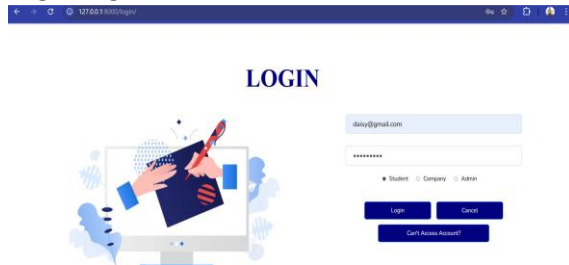
To sum up, this ‘CAREERX’ provides a collaborated environment for student, admin and company, who could access all the resources alone in one single web application. This provides all the necessary functionality for the student where student can access all the events, notifications that could provides updates, alongside student could also check for job postings and apply immediately. Similarly, within the same application company updates the new postings, and receives the applicants applied by the student. Admin monitors both student and company posting new events or updates if any. Overall, this application provides a feasible way with multiple features accessed by Student, Company and Admin all together. On an overall experience fxrom this application, I have learnt new technology, team work, collaboration via github. To conclude, I have experienced a good interaction and coordination with team mates.

6. Appendices

Index Page:



Login Page:



Password Reset:

127.0.0.1:8000/password_reset/




Reset your Password

[Student](#) [Company](#) [Admin](#)

[Reset](#) [Go Back](#)

Student Registration:

127.0.0.1:8000/student_register/




STUDENT REGISTRATION

[Register](#) [Cancel](#)

Company Registration:

127.0.0.1:8000/company_register/




COMPANY REGISTRATION

[Register](#) [Cancel](#)

Student Dashboard:

127.0.0.1:8000/student_dashboard/



Hi, Daisy Green !

Welcome to your Dashboard

- MY PROFILE
- INTERSHIPS
- JOBS
- NOTICES
- COMPANIES
- EVENTS
- LOGOUT

Student Profile:

127.0.0.1:8000/view_student_profile/

MY PROFILE



Daisy Green

daisy@gmail.com

123456789

04/04/1999

Female

8004

Chemistry

3.90

Update

Go Back

Apply Internship:

127.0.0.1:8000/view_internships/

INTERNSHIPS

Search by Role

Back to Dashboard

Internship ID	Role	Description	Duration (Months)	Type	Location	Stipend	Start Date	Apply By	Company	Actions
I-001	Software Developer Intern	Assist in software development projects.	6	Full Time	Remote	1500	Dec. 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	Expired
INT-3-20241116000000	Software Engineer	intern plus job offers	5	Full Time	In Office	50000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	Apply

Apply Job:

127.0.0.1:8000/view_jobs/

JOBS

Search by Role

Back to Dashboard

Job ID	Job Role	Description	Type	Location	Salary	Start Date	Apply By	Company	Actions
J-1-Softw	Software Developer	Develop and maintain software applications.	Full Time	Remote	70000	Dec. 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	Expired
JOB-1-20241116000000	Senior Software Tester	senior tester	Full Time	Hybrid	49999	March 3, 2025	Feb. 2, 2025	Tech Solutions Inc.	Apply
JOB-3-20241116000000	Junior Software Tester	junior tester	Full Time	Remote	35000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	Apply

Student Notices Page:

127.0.0.1:8000/view_notices/

NOTICES

Back to Dashboard

Notice ID	Announcement	Date
1	Midterm exams will start next week. Please prepare	2024-11-14

Student Events Page:

127.0.0.1:8000/view_events/

EVENTS

Back to Dashboard

Event ID	Title	Description	Date	Location
1	Tech Conference 2025	Annual tech conference featuring industry leaders.	March 15, 2025	San Francisco
2	AI Workshop	Hands-on workshop for AI and machine learning enthusiasts.	April 20, 2024	New York City
3	Data Science Bootcamp	Intensive bootcamp covering data science and analytics.	May 25, 2024	Chicago
5	Blockchain Summit	Summit discussing the latest trends in blockchain technology.	July 5, 2024	Los Angeles
6	Tech Conference 2026	Meetup with awarded professionals in the Computer Science field.	June 6, 2026	Texas Tech University
7	event 1	fest	Dec. 17, 2024	ttu

Company Dashboard Page:

127.0.0.1:8000/company_dashboard/

Hi, Tech Solutions Inc. !
Welcome to your Company Dashboard



MY PROFILE

INTERSHIPS

JOBS

NOTICE

APPLICANTS

EVENTS

LOGOUT

Company Profile Page:

127.0.0.1:8000/company_profile/

MY PROFILE



Tech Solutions Inc.

contact@techsolutions.com

1234567890

1234 Elm St

New York

NY

USA

10001

UpdateGo Back

Company Internships Page:

127.0.0.1:8000/view_internships/

INTERNSHIPS


Search by RoleAdd New InternshipBack to Dashboard

Internship ID	Role	Description	Duration (Months)	Type	Location	Stipend	Start Date	Apply By	Company	Actions
I-001	Software Developer Intern	Assist in software development projects.	6	Full Time	Remote	1500	Dec. 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	<div>Edit</div> <div>Delete</div>
INT-3-20241116000000	Software Engineer	intern plus job offers	5	Full Time	In Office	50000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	

16

Company Add Internship Page:

ADD INTERNSHIP



Internship Type:
☐ Full Time ☐ Part Time

Location:
☐ Remote ☐ In Office ☐ Hybrid

Start Date:

Enter Duration (in months)

Last Date to Apply:


Company Jobs Page:

JOBS

Job ID	Job Role	Description	Type	Location	Salary	Start Date	Apply By	Company	Actions
J-1-Softw	Software Developer	Develop and maintain software applications.	Full Time	Remote	70000	Dec. 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
JOB-1-20241116000000	Senior Software Tester	senior tester	Full Time	Hybrid	49999	March 3, 2025	Feb. 2, 2025	Tech Solutions Inc.	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
JOB-3-20241116000000	Junior Software Tester	junior tester	Full Time	Remote	35000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	

Company Add Job Page:

ADD JOB



Job Type:
☐ Full Time ☐ Part Time

Location:
☐ Remote ☐ In Office ☐ Hybrid

Start Date:

Last Date to Apply:

Company View Applicants:

APPLICANTS

Job Applicants

Type	Role	Applicant	Date Applied	Status	Actions
Job	Software Developer	Alice Johnsons	Nov. 16, 2024, 12:42 p.m.	Accepted	<input type="button" value="View"/>
Job	Software Developer	Bob Smith	Nov. 16, 2024, 1:24 p.m.	Pending	<input type="button" value="View"/>
Job	Senior Software Tester	Chiata	Nov. 24, 2024, 7:13 p.m.	Pending	<input type="button" value="View"/>

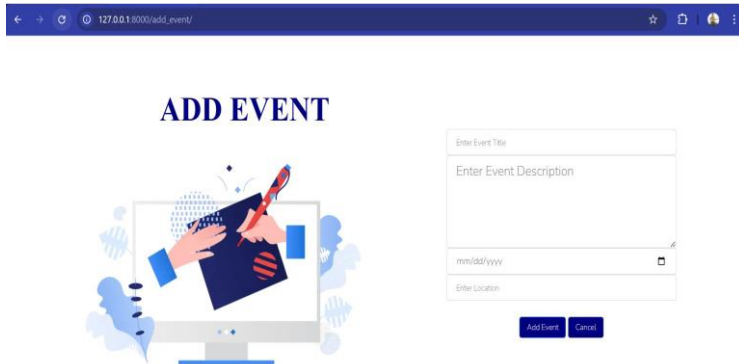
Internship Applicants

Type	Role	Applicant	Date Applied	Status	Actions
No internship applications available.					

Admin Dashboard:



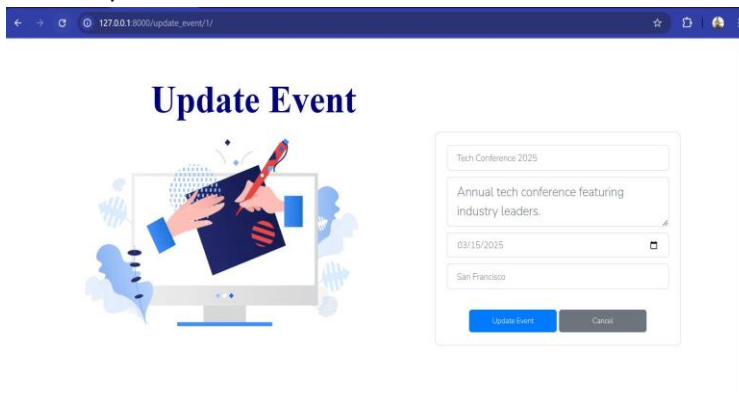
Admin Add Events:



Admin Manage Events:



Admin Update Events:



Admin View Companies:

127.0.0.1:8000/view_companies/

COMPANIES

Back to Dashboard

Company ID	Company Name	Email	Country
1	Tech Solutions Inc.	contact@techsolutions.com	USA
2	Green Energy Corp.	info@greenenergy.com	USA
3	Health Plus Ltd.	support@healthplus.com	USA
4	EduSmart Co.	contact@edusmart.com	USA
5	Global Tech Ltd.	info@globaltech.com	USA

Admin View Students:

127.0.0.1:8000/view_students/

STUDENTS

Back to Dashboard

Student ID	Students Name	Email	Actions
2	Alice Johnsons	alice@gmail.com	<div>View</div>
3	Bob Smith	bob@gmail.com	<div>View</div>
4	Charlie Brown	charlie@gmail.com	<div>View</div>
5	Daisy Green	daisy@gmail.com	<div>View</div>
6	Chinta	chinta@gmail.com	<div>View</div>

Admin Manage Notices:

127.0.0.1:8000/view_notices/

NOTICES

Add New Notice


Back to Dashboard

Notice ID	Announcement	Date	Actions
1	Midterm exams will start next week. Please prepare.	2024-11-14	<div><div>Edit</div><div>Delete</div></div>

Admin Add Notice:

127.0.0.1:8000/add_notice/

ADD NOTICE



Enter announcement text

Add Notice

Cancel

19

127.0.0.1:3000/view_internships/

Back to Dashboard

Search by Role


INTERNSHIPS

Internship ID	Role	Description	Duration (Months)	Type	Location	Stipend	Start Date	Apply By	Company	Actions
I-001	Software Developer Intern	Assist in software development projects.	6	Full Time	Remote	1500	Dec 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	
INT-3-20241116000000	Software Engineer	intern plus job offers	5	Full Time	In Office	50000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	

JOBS									
<input type="text"/> Search by Role <button>Back to Dashboard</button>									
Job ID	Job Role	Description	Type	Location	Salary	Start Date	Apply By	Company	Actions
J-1-Softw	Software Developer	Develop and maintain software applications.	Full Time	Remote	70000	Dec. 1, 2024	Nov. 20, 2024	Tech Solutions Inc.	
JOB-1- 20241116000000	Senior Software Tester	senior tester	Full Time	Hybrid	49999	March 3, 2025	Feb. 2, 2025	Tech Solutions Inc.	
JOB-3- 20241116000000	Junior Software Tester	junior tester	Full Time	Remote	35000	March 3, 2025	Feb. 2, 2025	Health Plus Ltd.	




The screenshot shows the PyCharm IDE interface with the following components:

- File Explorer (Left):** Displays the project structure for 'F:\CS356project'. The 'migrations' folder is expanded, showing files like '0001_initial.py', '0002_admin.py', '0003_app.py', '0004_tests.py', '0005_auth.py', '0006_auth_user_permissions.py', '0007_auth_admin_log.py', '0008_auth_content_type.py', '0009_auth_permission.py', '0010_auth_group.py', '0011_auth_user.py', '0012_auth_admin.py', '0013_auth_company.py', '0014_auth_admin.py', '0015_auth_student.py', '0016_auth_internship.py', '0017_auth_notice.py', '0018_auth_jobapplications.py', '0019_auth_internshipapplications.py', and '0020_django_session.py'.
- SQL Editor (Middle):** Shows a SQL query: `SELECT name FROM sqlite_master WHERE type='table';`. The query is executed, and the results are displayed in a table with two columns: 'id' and 'name'.
- SQL Results (Right):** Displays the output of the query, showing a list of tables in the database: 'django_migrations', 'sqlite_sequence', 'auth_group_permissions', 'auth_user_groups', 'auth_user_user_permissions', 'django_admin_log', 'django_content_type', 'auth_permission', 'auth_group', 'auth_user', 'home_admin', 'home_company', 'home_admin', 'home_student', 'home_internship', 'home_job', 'home_notice', 'home_jobapplications', 'home_internshipapplications', and 'django_session'.



CSS3536 project

Public

main

6 Branches

0 Tags

+

Add file

→

Code

juan-mgc



interfaces updated

16b2a08 · 2 hours ago

19 Commits

home	Interfaces updated	2 hours ago
mySite	Interfaces updated	2 hours ago
README.md	merged html and login styling, fixed/added logic of addEdit...	2 weeks ago
db.sqlite3	interfaces updated	2 hours ago
manage.py	initial login pages created	2 months ago
populateDB.txt	improved layout, fixed issues with job and internship input b...	last week

README

CSS3536 Project

Requirements

To install the required dependencies, use the following command:

7. References

- [1] Pinjari, Mustafa, et al. "Online job portal." *International Research Journal of Engineering and Technology* 6.4 (2019).
- [2] Ireland, Christopher, et al. "Understanding object-relational mapping: A framework based approach." *International Journal On Advances in Software* 2.2 (2009).
- [3] O'Neil, Elizabeth J. "Object/relational mapping 2008: hibernate and the entity data model (edm)." *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008.
- [4] AlDeen, Omar Nizam, Yaseen Dergham, and Maxeem Deeb. "Online Job Portal."
- [5] <https://docs.djangoproject.com/en/5.1/intro/tutorial01/>
- [6] https://www.youtube.com/watch?v=cW1_OoTiWg8&t=518s
- [7] <https://docs.djangoproject.com/en/5.1/topics/migrations/>
- [8] <https://djangocentral.com/create-a-hello-world-django-application/>