**Placement Portal**

**UIT2402 - ADVANCED DATA STRUCTURES AND ALGORTIHM ANALYSIS**

**A PROJECT REPORT**

**Submitted by**

* Shanthanu G 3122225002125
* Shiva Sai Adithiyan S 3122225002126
* Sindhujaa I 3122225002129
* Singaram PL 3122225002130

**SSN COLLEGE OF ENGINEERING,**

**KALAVAKKAM**

****

**JUNE 2024**

**Abstract Of This Project:**

* This project report details the development of a Placement Web Portal, designed to streamline the process of internships and job placements within a college.
* The portal provides a centralized platform for administrators, faculty, and students to access and manage relevant data.
* The platform is designed to accommodate various user roles with specific access controls, including administrators who oversee the placement process, faculty members who guide and support students, and students who manage their profiles and track their placement progress.

**Introduction And Motivation:**

**Introduction:**

The Placement Web Portal project was initiated with the primary aim of addressing the challenges associated with managing the internship and job placement processes within the department.

This project aims to modernize the placement process by developing a web-based platform that integrates all aspects of placement management. The portal will serve as a one-stop solution for students to access job opportunities, for faculty to track student progress, and for administrators to manage the placement process seamlessly.

By providing a centralized platform, it aims to facilitate better communication and data management among students, faculty, and administrators, ultimately improving the placement outcomes.

This initiative reflects the commitment to leveraging technology to enhance the educational experience and support students in their career endeavours.

**Motivation:**

* **Centralization and Efficiency:** A centralized web portal enhances efficiency and streamlines placement operations.
* **Automating Repetitive Tasks:** Automation reduces errors and frees up administrators for strategic tasks.
* **Data Security and Privacy:** Secure authentication and role-based access controls protect student data.
* **Scalability and Flexibility:** Using Firebase for the database and Django for the backend provides scalability and flexibility for future growth.
* **Student Empowerment:** Students get a personalized dashboard for managing profiles and tracking applications.
* **Continuous Improvement:** Agile development ensures the portal evolves based on user feedback.

In conclusion**,** the project modernizes placement processes, enhancing efficiency, security, and user experience.

**Problem Statement:**

To craft a college Placement Web Portal entails designing a centralized platform tailored for administrators, faculty, and students, facilitating seamless communication and access to internship and job opportunities within the institution.

Key functionalities of the portal include filtering student data based on their batch, accessing comprehensive student profiles that detail their academic achievements, placed companies, and internship experiences, and displaying relevant academic and placement information dynamically

**Requirement:**

 Selection of appropriate programming languages, frameworks, and tools:

Python Django for backend development, Firebase for database

management, and HTML/CSS for frontend design.

 Client-side scripting:

Use HTML and CSS for creating dynamic and responsive user interfaces.

 Database management system:

Use Firebase for storing user data and student records.

**User Roles and Access Control:**

* Administrator: Full access to all features and profile management features.
* Student: Ability to manage specific details of their profiles.
* Faculty : Access to view the student profiles.
* **User Authentication:** Secure user authentication mechanisms (e.g., email/password)
* **Dashboard**: Customized dashboard for each batch displaying relevant information.

## Design and Development Solution

### Choice of Method

* **Backend:**
  + **Python Django:** Chosen for its robustness and ease of creating a scalable and maintainable backend.
* **Database:**
  + **Firebase:** Selected for its real-time database capabilities, scalability, and ease of integration with web applications.
* **Client-Side Scripting:**
  + **Languages and Frameworks:**
    - **HTML & CSS:** Core technologies for building the structure and style of web pages.
* **Security Measures:**
  + Implemented secure authentication mechanisms (e.g., email/password).
  + Role-based access control to manage user permissions (Administrator, Student, Faculty).

### Frontend Design

* **Responsive Design:** Ensured optimal viewing across various devices (desktops, tablets, smartphones).
* **User Interface:** Intuitive and easy-to-navigate interfaces for different user roles.
* **Real-time Updates:** Dynamic interactions and real-time data updates using Django and Firebase.

### Justification

* **Django:** Provides a powerful framework for backend development, ensuring smooth integration with Firebase.
* **Firebase:** Offers a flexible and scalable solution for managing data in real-time.
* **HTML & CSS:** Essential for creating responsive and visually appealing user interfaces.

**Process Management:**

* **Sprint Planning:**
  + Defined the project scope and objectives.
  + Divided the project into manageable sprints to ensure continuous progress and timely delivery.

**Sprint Breakdown:**

* **Total Number of Sprints:** 3
  + **Sprint 1 (1 week):** Initial setup, requirement analysis, and basic infrastructure development.
  + **Sprint 2 (2 weeks):** Core feature development, including user authentication, profile management, and job listings.
  + **Sprint 3 (2 weeks):** Integration, testing, debugging, and final deployment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint | Epic | User Story | Requirement/User Story | Essential or Desirable | Description of the Requirement |
| 1. | Frontend Implementation | RS1 | To create a login page and platform tailored for administrators, faculty, and students. | Essential | Implementation of login, user security and access along with displaying details. |
| 2. | Database Implementation | RS2 | To implement structured schema of student details. | Essential | Implementation of database for backend in Firebase. |
| 3. | Back End Implementation | RS3 | To provide comprehensive documentation. | Essential | Filtering data according to user needs. |

**Daily Stand-ups:**

* Short, daily meetings to discuss progress, roadblocks, and plan for the day.

**Iterative Development:**

* Continuous integration and testing to ensure each feature works correctly and meets user requirements.
* Regular feedback from stakeholders to refine and improve the portal iteratively.

**User Stories and Tasks:**

* Created user stories to capture the requirements from the perspective of end-users.
* Broken down user stories into smaller, actionable tasks for developers.

**Testing and Quality Assurance:**

* Conducted unit testing, integration testing, and end-to-end testing to ensure the portal's functionality and performance.
* Gathered feedback from beta testers and made necessary adjustments before the final deployment.

**Documentation:**

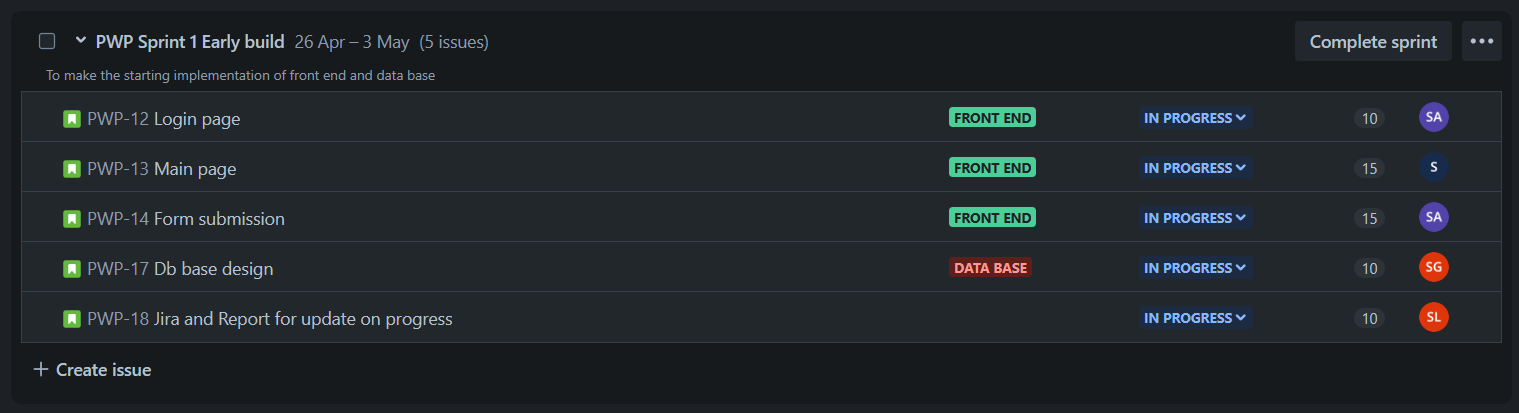
* Maintained detailed documentation of the codebase for documentation of future maintenance and scalability.

**Tools Used:**

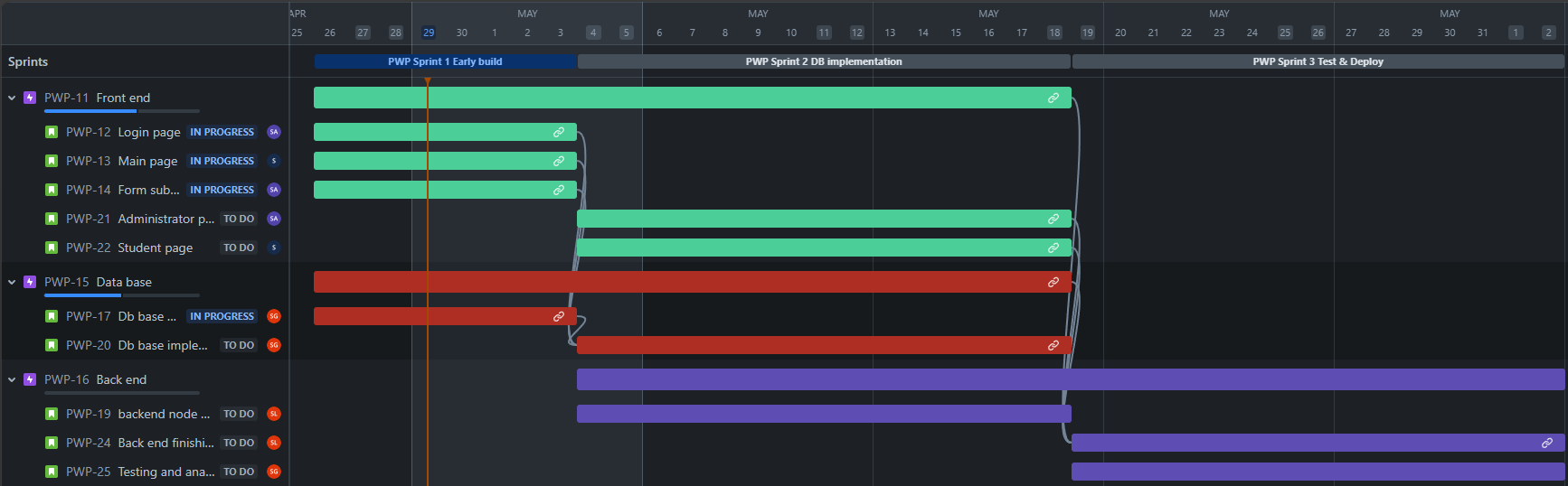
* **Project Management:** JIRA for tracking progress and managing tasks.
* **Version Control:** GitHub for source code management and collaboration.

The development process remained flexible and adaptive to changes, ensuring the final product met the requirements and expectations of all stakeholders.

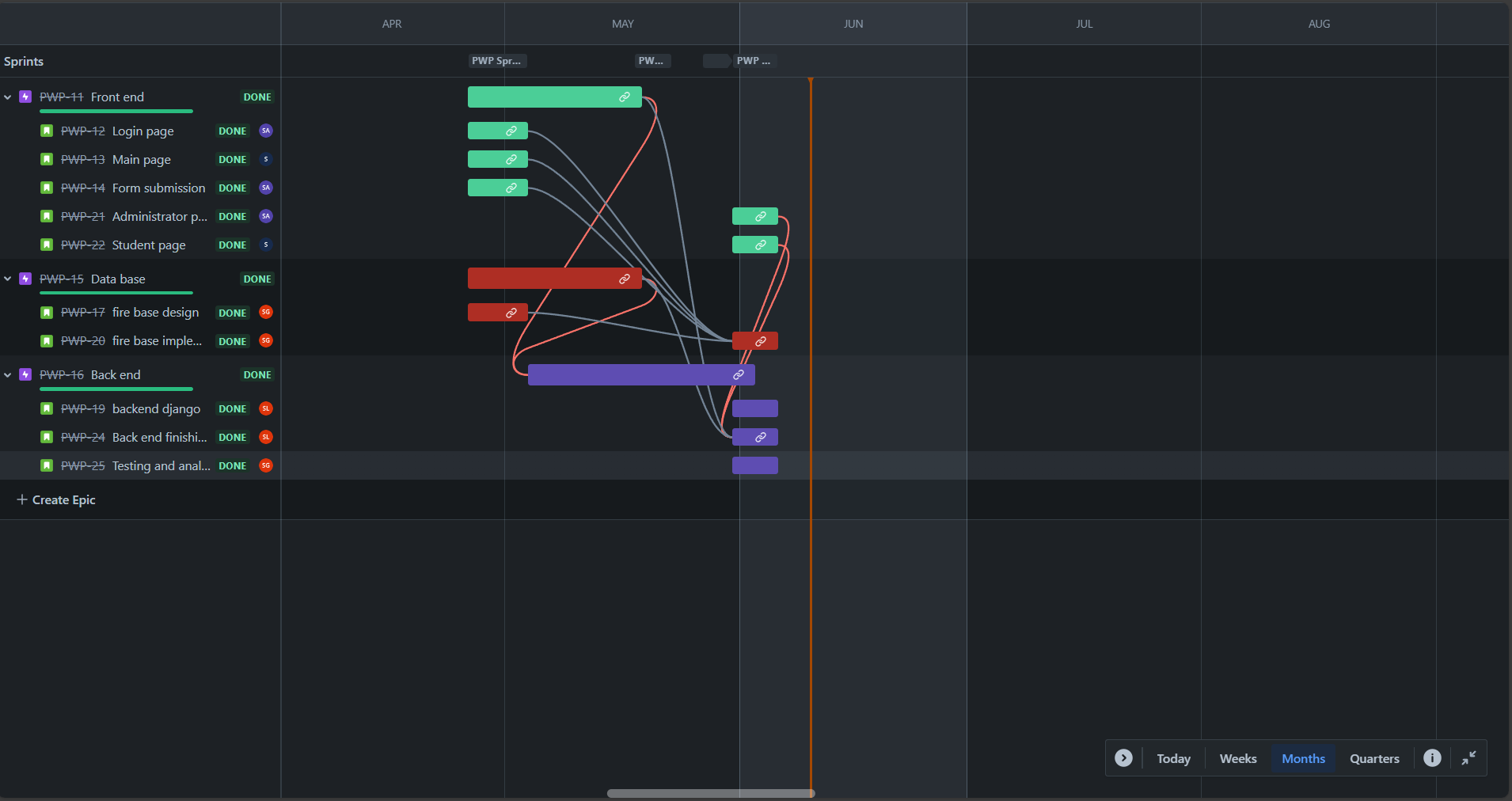
**Sprints**



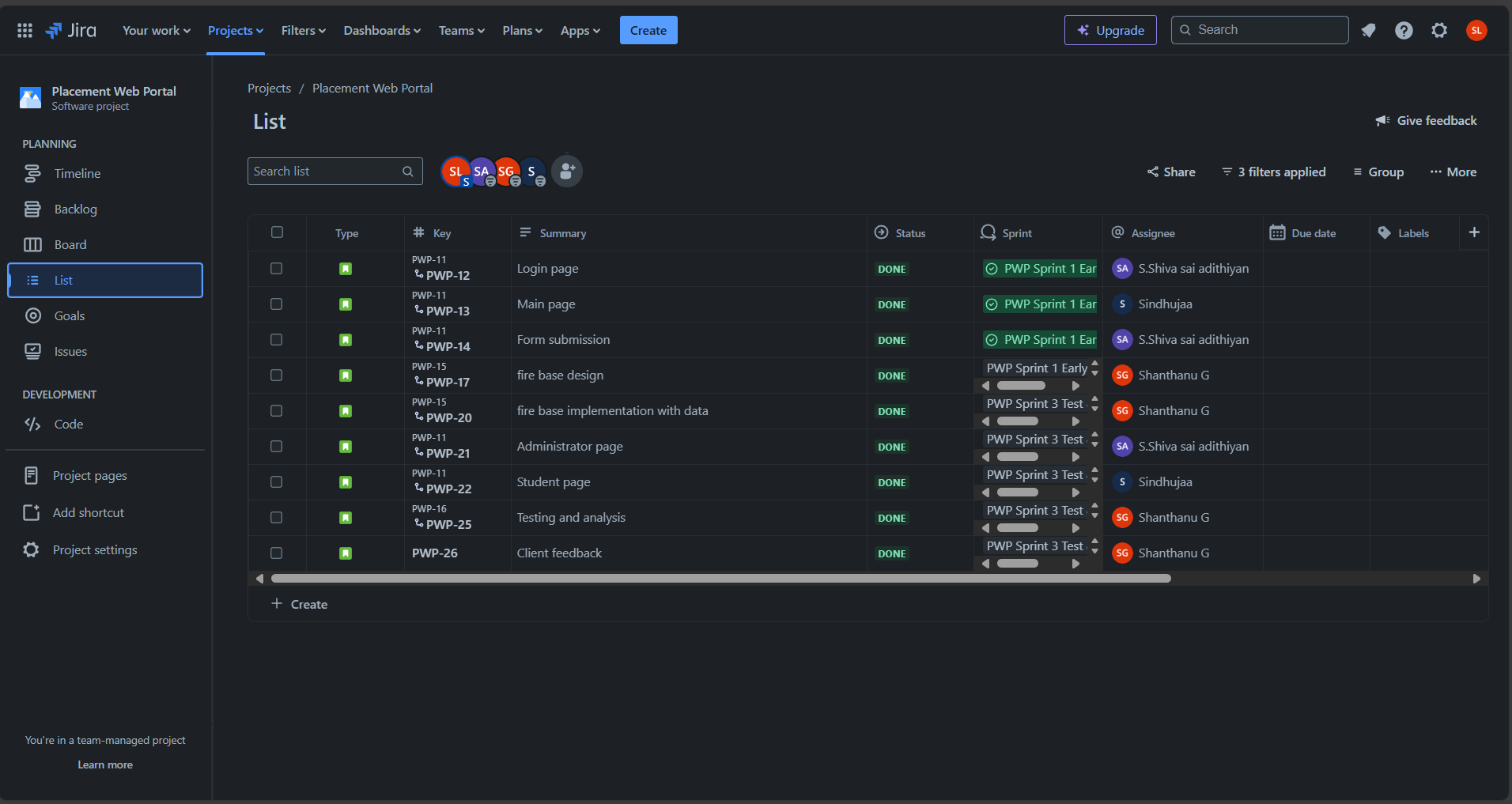
**Time line (while in Progress)**



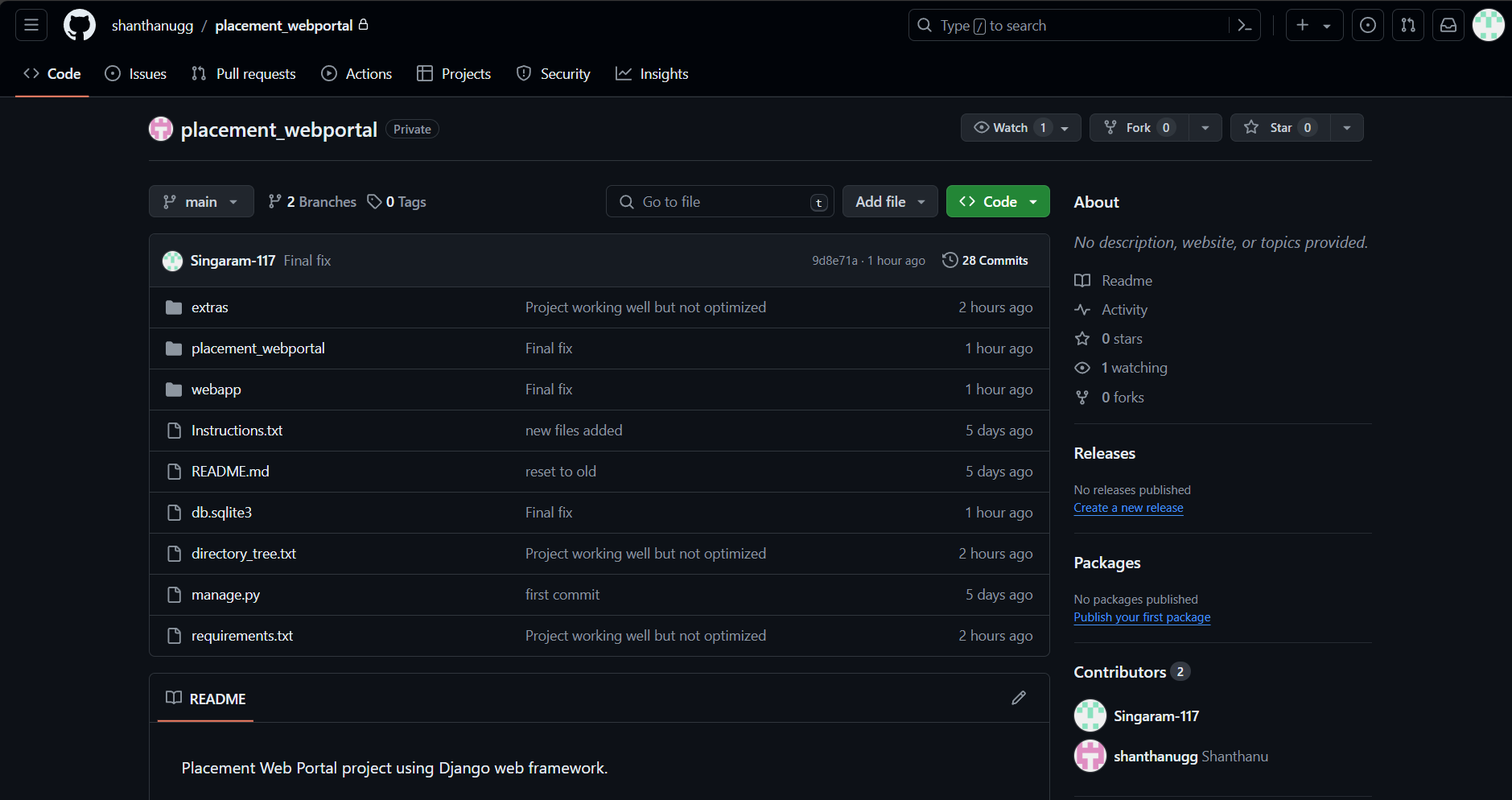
**Jira Time line (after finishing)**

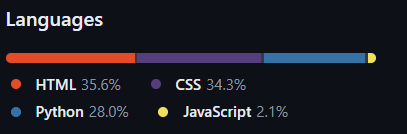
****

**Finished backlogs in list view**

****

**Git hub repository (version control)**

****

****

**Coding and Implementation:**

**from django.shortcuts import render, redirect**

**from django.http import JsonResponse, HttpResponse**

**from django.conf import settings**

**from django.dispatch import receiver**

**from django.core.signals import request\_started, request\_finished**

**from django.contrib.sessions.backends.db import SessionStore**

**from firebase\_admin import credentials, auth, db**

**from django.template.loader import render\_to\_string**

**import requests**

**import firebase\_admin**

**import pandas as pd**

**import os**

**# Initialize Firebase app (ensure the path to your service account key is correct)cred = credentials.Certificate(os.path.join(os.path.dirname(\_\_file\_\_), 'static', 'extras', 'firebase\_key.json'))**

**# Initialize Firebase app (ensure the path to your service account key is correct)**

**cred\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'static', 'extras', 'firebase\_key.json')**

**if not firebase\_admin.\_apps:**

**cred = credentials.Certificate(cred\_path)**

**firebase\_admin.initialize\_app(cred, {**

**'databaseURL': 'https://placement-web-portal-c3159-default-rtdb.asia-southeast1.firebasedatabase.app/'**

**})**

**FIREBASE\_API\_KEY = "YOUR\_FIREBASE\_API\_KEY"**

**def home(request):**

**request.session["home\_visited"] = True**

**request.session["is\_authenticated"] = False**

**years = fetch\_years()**

**year = max(years)**

**context = fetch\_data(year)**

**context["selected\_year"] = year**

**return render(request, 'home.html', context)**

**def student\_login(request):**

**return render(request, 'student\_login.html')**

**def admin\_login(request):**

**if request.method == "POST":**

**email = request.POST.get('admin\_username')**

**password = request.POST.get('admin\_password')**

**# Authenticate with Firebase**

**payload = {**

**"email": email,**

**"password": password,**

**"returnSecureToken": True**

**}**

**response = requests.post(f"https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key={FIREBASE\_API\_KEY}", data=payload)**

**if response.status\_code == 200:**

**id\_token = response.json()['idToken']**

**try:**

**# Verify the ID token**

**decoded\_token = auth.verify\_id\_token(id\_token)**

**uid = decoded\_token['uid']**

**# You can add additional checks, such as checking for a specific role**

**user = auth.get\_user(uid)**

**if email == user.email:**

**# Store authentication status in session**

**request.session['is\_authenticated'] = True**

**return redirect('admin\_dashboard')**

**else:**

**error\_message = "Invalid credentials. Please try again."**

**return render(request, 'administrator\_login.html', {'error\_message': error\_message})**

**except auth.InvalidIdTokenError:**

**error\_message = "Invalid token. Please try again."**

**return render(request, 'administrator\_login.html', {'error\_message': error\_message})**

**else:**

**error\_message = "Authentication failed. Please check your credentials and try again."**

**return render(request, 'administrator\_login.html', {'error\_message': error\_message}) return render(request, 'administrator\_login.html')**

**def student\_profile(request):**

**return render(request, 'student\_profile.html')**

**def admin\_dashboard(request):**

**# Check if user is authenticated**

**try:**

**if not request.session.get('is\_authenticated', False):**

**return redirect('admin\_login') # Redirect to login page if not authenticated**

**if not(request.session.get('home\_visited')):**

**raise Exception**

**except Exception as e:**

**return redirect('admin\_login') # Redirect to login page if not authenticated**

**if request.method == "POST":**

**selected\_year = request.POST.get('year')**

**context = fetch\_data(selected\_year)**

**context['selected\_year'] = selected\_year**

**else:**

**# If the request method is GET, default to the current year**

**selected\_year = '2024' # Change this to your default year**

**context = fetch\_data(selected\_year)**

**context['selected\_year'] = selected\_year**

**context['years\_list'] = fetch\_years()**

**return render(request, 'admin\_dashboard.html', context)**

**def fetch\_data(year):**

**# Fetch data from Firebase**

**result = db.reference(f'/batches/{year}').get()**

**# Convert Firebase data to pandas DataFrame**

**df = pd.DataFrame(result)**

**# Calculate average and highest CTC**

**avg\_ctc = df['CTC'].mean()**

**highest\_ctc = df['CTC'].max()**

**# Generate pie chart data**

**dream\_ctc = ((df['CTC'] >= 500000) & (df['CTC'] <= 1000000)).sum()**

**super\_dream\_ctc = ((df['CTC'] > 1000000) & (df['CTC'] <= 2000000)).sum()**

**marquee\_ctc = (df['CTC'] > 2000000).sum()**

**pie\_chart\_data = [dream\_ctc, super\_dream\_ctc, marquee\_ctc]**

**# Filter top recruiters**

**top\_recruiters\_count = df['Final Offer'].value\_counts().to\_dict()**

**top\_recruiters\_lst= [key for key in top\_recruiters\_count if top\_recruiters\_count[key] > 10]**

**avg\_floored\_ctc = avg\_ctc // 100000**

**context = {**

**'highest\_ctc': highest\_ctc**

**'avg\_floored\_ctc': avg\_floored\_ctc,**

**'pie\_chart\_data': pie\_chart\_data,**

**'top\_recruiters\_lst': top\_recruiters\_lst**

**}**

**return context**

**def fetch\_years():**

**years\_lst = list(db.reference('/years').get())**

**return years\_lst**

**from django.http import JsonResponse**

**from django.shortcuts import render**

**from django.template.loader import render\_to\_string**

**def yearly\_records(request):**

**if request.method == 'POST':**

**selected\_year = request.POST.get('year')**

**else:**

**selected\_year = '2024' # Default year**

**# Fetch data for the selected year**

**students\_data = fetch\_year\_data(selected\_year)**

**# Fetch list of available years**

**years\_list = fetch\_years()**

**context = {**

**'students': students\_data,**

**'years\_list': years\_list,**

**'selected\_year': selected\_year**

**}**

**if request.headers.get('x-requested-with') == 'XMLHttpRequest':**

**return JsonResponse({'students': render\_to\_string('students\_list.html', {'students': students\_data})})**

**return render(request, 'yearly\_records.html', context)**

**def fetch\_year\_data(year):**

**# Fetch data from Firebase**

**result = db.reference(f'/batches/{year}').get()**

**# Convert Firebase data to a list of dictionaries**

**if result:**

**if isinstance(result, list):**

**data = result**

**else:**

**data = [value for key, value in result.items()]**

**# Clean up keys to remove spaces and other problematic characters**

**cleaned\_data = []**

**for student in data:**

**cleaned\_student = {}**

**for key, value in student.items():**

**clean\_key = key.replace(" ", "").replace("|", "").replace("Campus", "Campus")**

**cleaned\_student[clean\_key] = value**

**cleaned\_data.append(cleaned\_student)**

**return cleaned\_data**

**else:**

**return []**

**from .models import Student**

**def student\_profile(request, roll\_number):**

**try:**

**student = Student.objects.get(roll\_number=roll\_number)**

**except Student.DoesNotExist:**

**# Handle the case where the student does not exist**

**return render(request, 'student\_not\_found.html')**

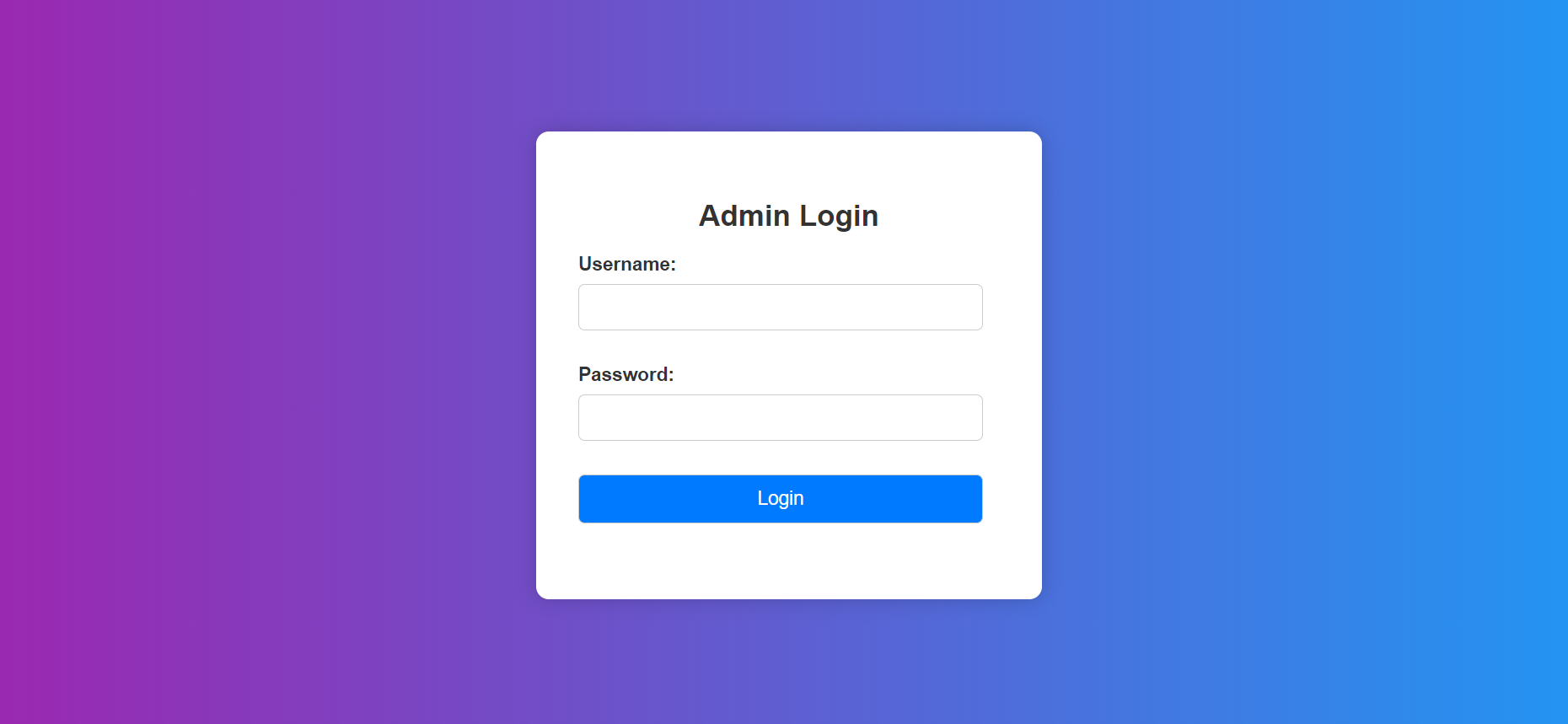
**context = {**

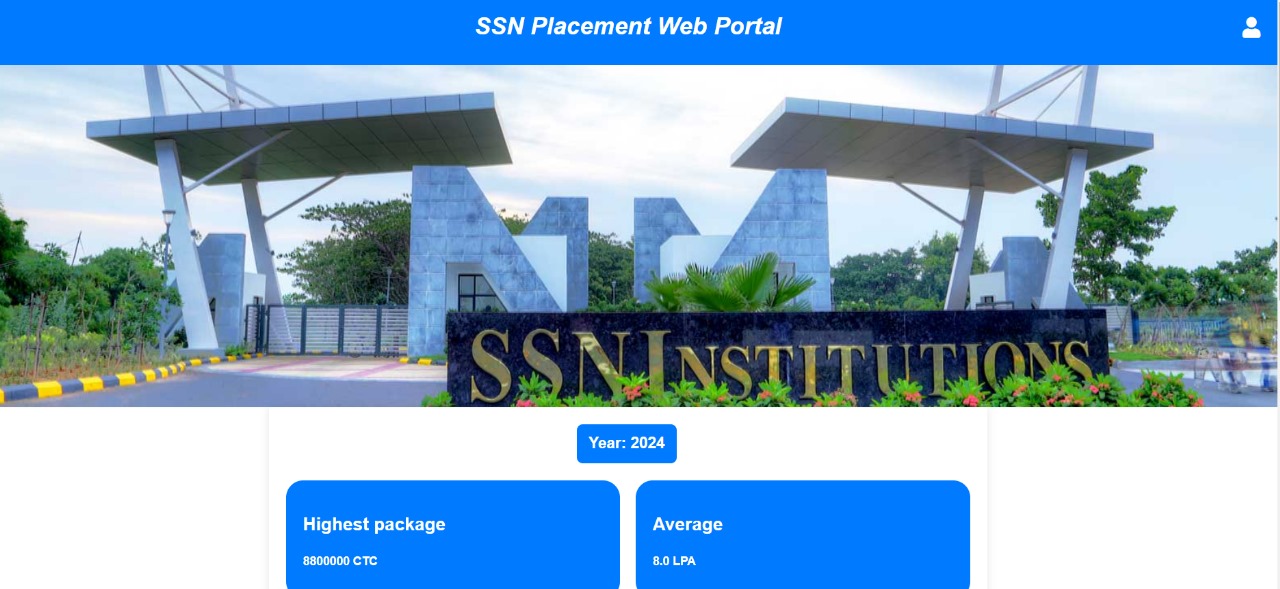
**'student': student**

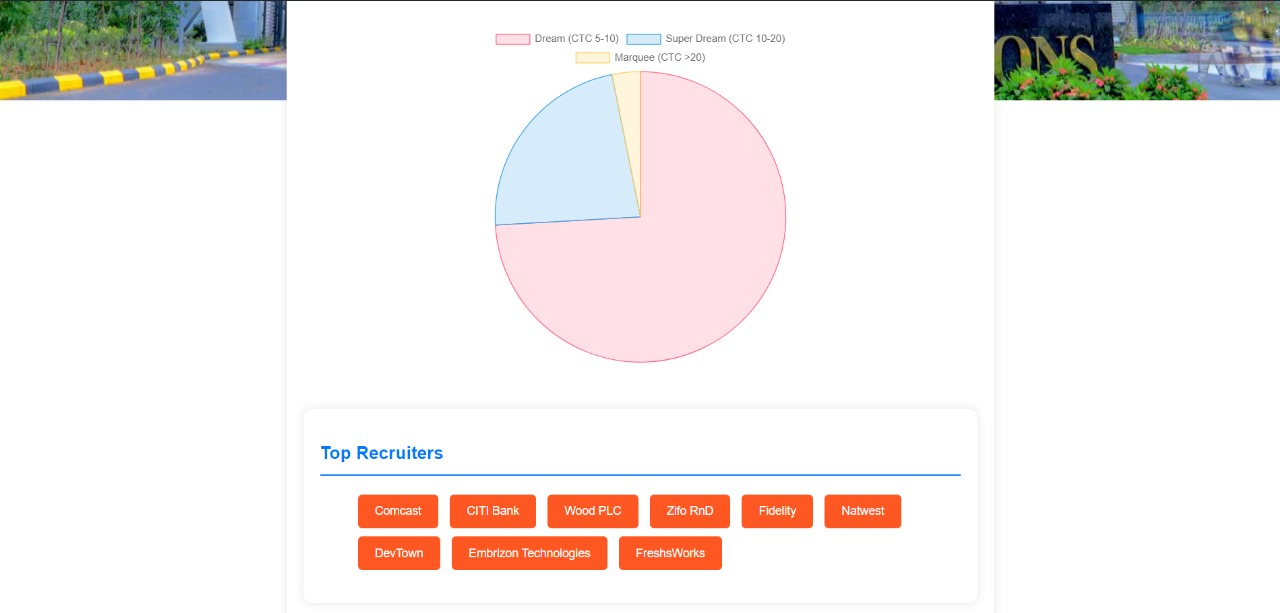
**}**

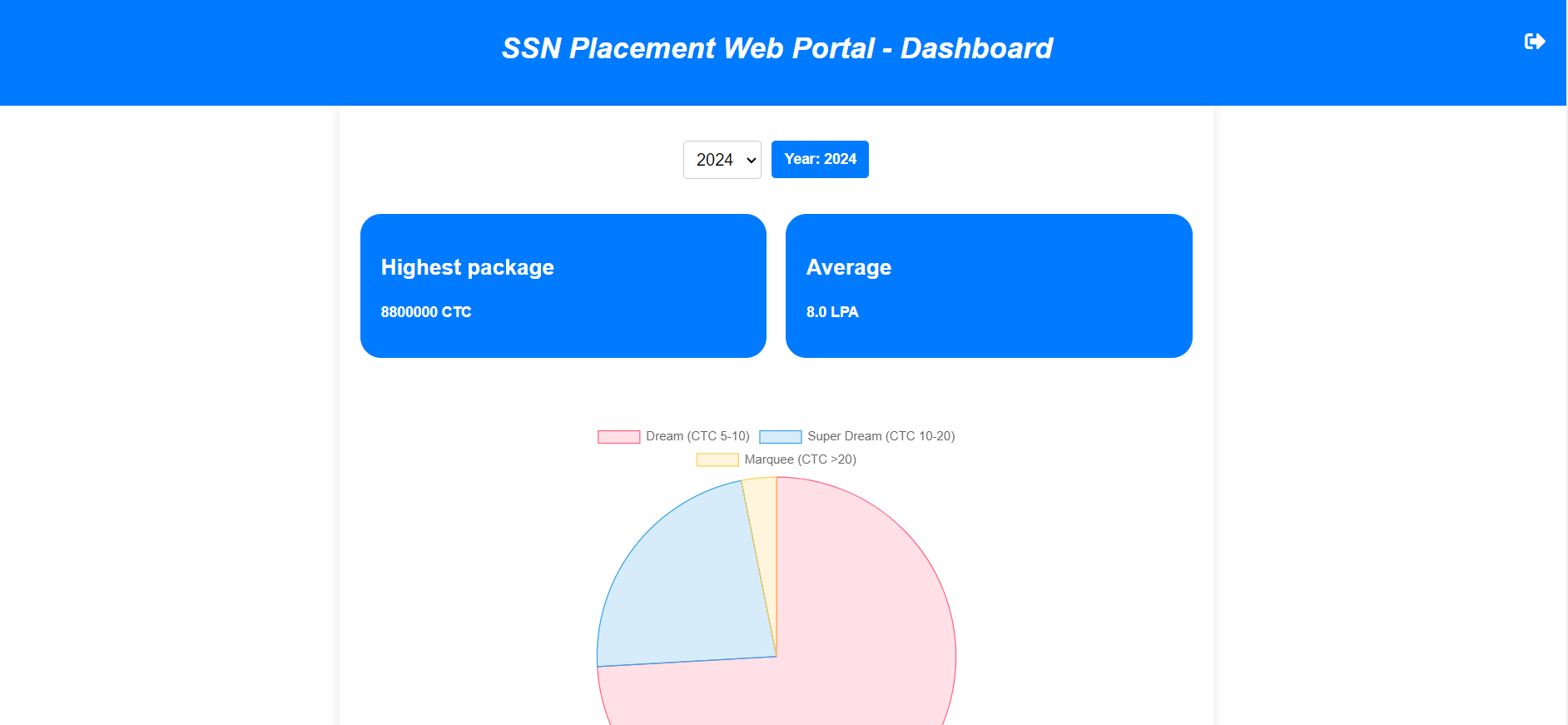
**return render(request, 'stu\_dentprofile.html', context)**

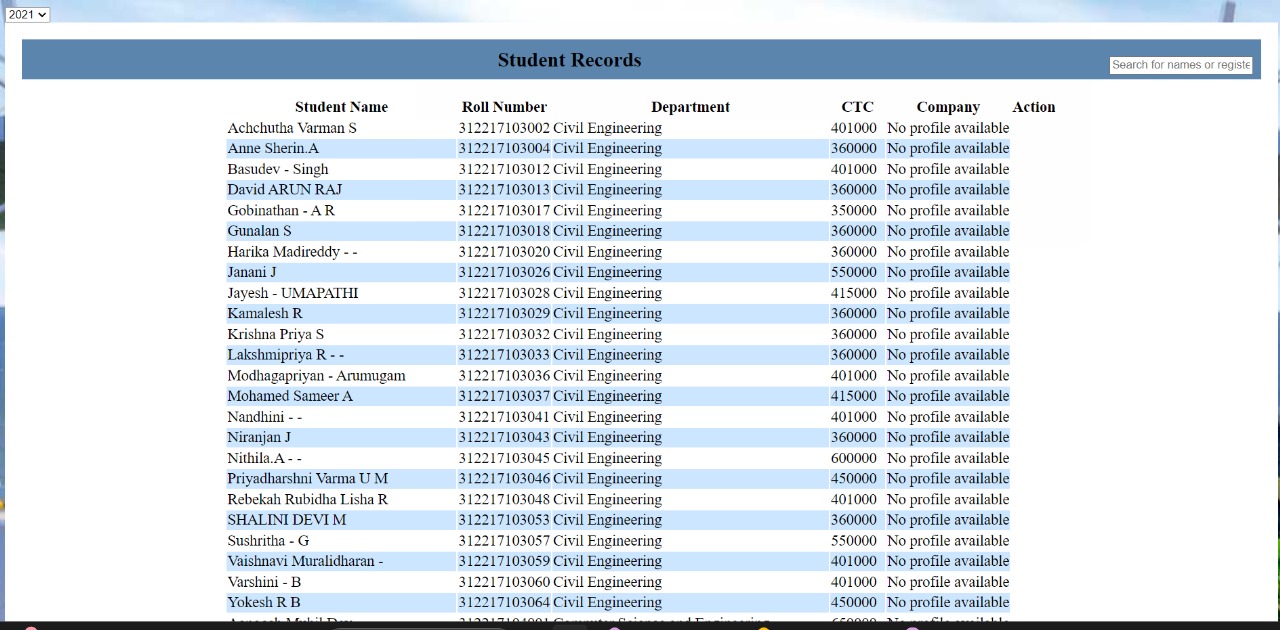
**Results:**

****





****



### Gap Analysis for Placement Portal Project

#### 1. ****Project Objectives and Scope****

* **Current State**: The project aims to create a Placement Web Portal to streamline the process of internships and job placements within the college, offering a centralized platform for administrators, faculty, and students.
* **Gap**: The project needs to explicitly define detailed objectives and scope, including specific functionalities, user roles, and features to be implemented. Clarification is needed on the boundaries of the project to avoid scope creep.

#### 2. ****Requirements Gathering and Analysis****

* **Current State**: Basic requirements have been outlined, including user roles, access control, and database management.
* **Gap**: Detailed functional and non-functional requirements need to be documented. Specific user stories and acceptance criteria should be developed to ensure all stakeholder needs are met.

#### 3. ****System Design and Architecture****

* **Current State**: The project uses Django for backend development, Firebase for database management, and HTML/CSS for frontend design.
* **Gap**: Comprehensive system architecture diagrams, including data flow, component interactions, and deployment architecture, are missing. There is a need for detailed design documentation covering all major system components and their interactions.

#### 4. ****Implementation Plan****

* **Current State**: The project has identified core technologies and provided a brief implementation plan.
* **Gap**: A more detailed implementation plan is required, including task breakdown, timelines, resource allocation, and milestones. The plan should also include a risk management strategy to address potential challenges.

#### 5. ****Testing and Quality Assurance****

* **Current State**: Basic testing approaches like unit testing, integration testing, and end-to-end testing have been mentioned.
* **Gap**: A comprehensive testing strategy, including test plans, test cases, and a bug tracking system, needs to be developed. Quality assurance processes should be defined to ensure the portal meets all functional and performance requirements.

#### 6. ****Security and Privacy****

* **Current State**: Secure authentication and role-based access controls have been implemented.
* **Gap**: There needs to be a thorough security assessment, including penetration testing and security reviews. Detailed plans for data encryption, secure data storage, and compliance with relevant data protection regulations are required.

#### 7. ****User Interface and User Experience (UI/UX)****

* **Current State**: The project aims to create a responsive and intuitive user interface.
* **Gap**: Detailed UI/UX design guidelines, wireframes, and prototypes are necessary. User feedback mechanisms should be established to continuously improve the user experience.

#### 8. ****Documentation****

* **Current State**: Some documentation has been provided, including coding implementation and process management.
* **Gap**: Comprehensive documentation covering all aspects of the project, including user manuals, developer guides, and maintenance documentation, is needed. This documentation should be kept up-to-date throughout the project lifecycle.

#### 9. ****Project Management and Communication****

* **Current State**: The project uses JIRA for task management and GitHub for version control.
* **Gap**: A detailed project management plan, including communication strategies, regular status updates, and stakeholder engagement plans, is required. Clear roles and responsibilities for team members should be defined.

#### 10. ****Future Work and Enhancements****

* **Current State**: Suggestions for future enhancements like advanced analytics and additional student features have been made.
* **Gap**: A roadmap for future enhancements should be developed, prioritizing features based on user feedback and project goals. Potential integration with other systems and long-term maintenance plans should also be considered.

#### To bridge the gaps, the project needs a more detailed and structured approach in areas like requirements gathering, system design, implementation planning, testing, security, UI/UX, documentation, and project management. Addressing these gaps will ensure the Placement Portal project meets its objectives efficiently and effectively, providing a robust solution for managing college placements.

**Conclusion and Future work:**

**Conclusion:**

The Placement Web Portal project significantly enhances the efficiency of the placement process by centralizing all relevant information, automating repetitive tasks, and streamlining communication between administrators, faculty, and students.

The portal reduces administrative burdens, minimizes errors, and accelerates data retrieval, making the placement process more efficient and transparent. Thus the platform is designed to accommodate various user roles with specific access controls, including administrators who oversee the placement process, faculty members who guide and support students, and students who manage their profiles and track their placement progress.

By empowering students with a personalized dashboard and adopting an agile development approach, the portal remains adaptive, user-friendly, and continuously improving.

**Future Work:**

* **Advanced Analytics:** Integrate advanced analytics to provide insights into placement trends, student performance, and employer preferences, aiding data-driven decision-making.
* **Enhanced Student Features:** Develop additional features for students, such as resume-building tools, interview preparation resources, and personalized job recommendations based on their profiles and interests.

**References:**

**HTML & CSS**

1. **YouTube**
   * [HTML Full Course - Build a Website Tutorial](https://www.youtube.com/watch?v=pQN-pnXPaVg) by freeCodeCamp.org
   * [CSS Crash Course For Absolute Beginners](https://www.youtube.com/watch?v=yfoY53QXEnI) by Traversy Media
2. **Websites**
   * [MDN Web Docs: HTML](https://developer.mozilla.org/en-US/docs/Web/HTML)
   * [MDN Web Docs: CSS](https://developer.mozilla.org/en-US/docs/Web/CSS)

#### Python Django

1. **YouTube**
   * [Django Crash Course by Traversy Media](https://www.youtube.com/watch?v=F5mRW0jo-U4)
   * [Django for Beginners by The Net Ninja](https://www.youtube.com/watch?v=UmljXZIypDc)
2. **Websites**
   * [Django Official Documentation](https://docs.djangoproject.com/en/stable/)
3. **GitHub**
   * [Django Official Repository](https://github.com/django/django)
   * [Django Rest Framework](https://github.com/encode/django-rest-framework)

#### Firebase

1. **YouTube**
   * [Firebase Firestore Tutorial by Academind](https://www.youtube.com/watch?v=TlB_eWDSMt4)
   * [Firebase Authentication Tutorial by The Net Ninja](https://www.youtube.com/watch?v=PKwu15ldZ7k)

**Client Certificate:**

**Name of the project: Placement Web Portal**

#### Members:

1. Shanthanu G 3122225002125
2. Shiva Sai Adithiyan S 3122225002126
3. Sindhujaa I 3122225002129
4. Singaram PL 3122225002130

**Client details:** Dr. V.Arulkumar / Dr.A.Sandana Karuppan

#### Rating System - 1: Strongly disagree 2: Disagree 3: Neutral 4: Agree 5: Strongly Agree

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Questions** | **1** | **2** | **3** | **4** | **5** |
| The problem was well discussed and, the requirements and goals were clear. |  |  |  |  |  |
| The project plan was well defined and communicated from the start. |  |  |  |  |  |
| The resources were adequate for achieving the goals. |  |  |  |  |  |
| The original timeline was realistic and was followed. |  |  |  |  |  |
| The teamwork was well demonstrated. |  |  |  |  |  |
| The client was communicated on regular intervals and given updates on the progress of the project. |  |  |  |  |  |
| The expected project requirements have been satisfied. |  |  |  |  |  |

Client signature: