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.

<u>User Roles and Access Control</u>:

- 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:

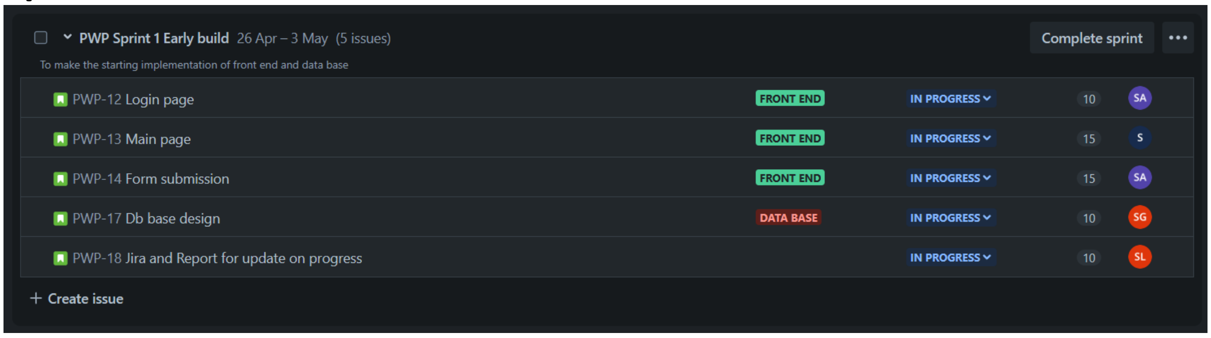
- Maintained detailed documentation of the codebase for documentation of future maintenance and scalability.

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



PWP Sprint 1 Early build 26 Apr – 3 May (5 issues)

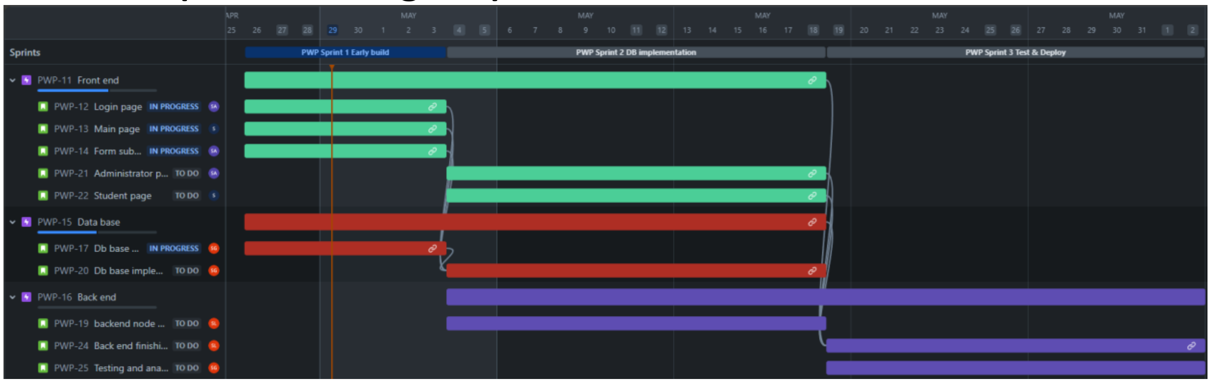
Complete sprint

To make the starting implementation of front end and data base

PWP-12 Login page	FRONT END	IN PROGRESS	10	SA
PWP-13 Main page	FRONT END	IN PROGRESS	15	S
PWP-14 Form submission	FRONT END	IN PROGRESS	15	SA
PWP-17 Db base design	DATA BASE	IN PROGRESS	10	SG
PWP-18 Jira and Report for update on progress		IN PROGRESS	10	SL

+ Create issue

Time line (while in Progress)

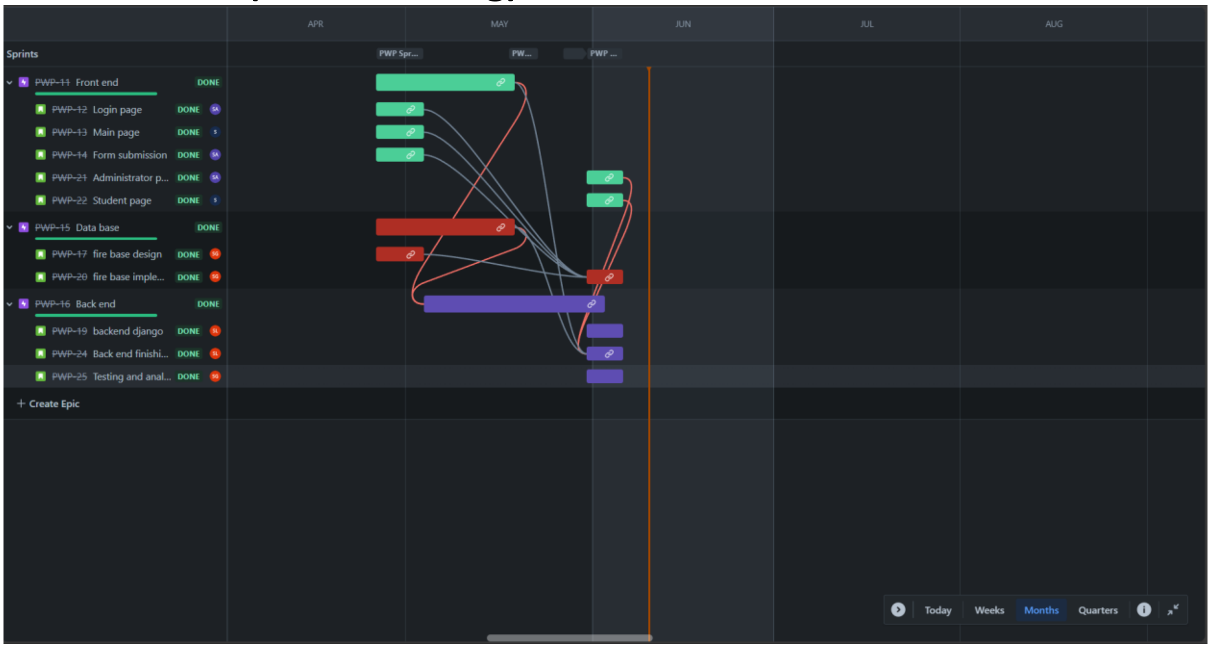


Sprints

PWP Sprint 1 Early build · PWP Sprint 2 DB implementation · PWP Sprint 3 Test & Deploy

PWP-11 Front end		
PWP-12 Login page	IN PROGRESS	SA
PWP-13 Main page	IN PROGRESS	S
PWP-14 Form sub...	IN PROGRESS	SA
PWP-21 Administrator p...	TO DO	SA
PWP-22 Student page	TO DO	S
PWP-15 Data base		
PWP-17 Db base ...	IN PROGRESS	SG
PWP-20 Db base imple...	TO DO	SG
PWP-16 Back end		
PWP-19 backend node ...	TO DO	SL
PWP-24 Back end finishi...	TO DO	SL
PWP-25 Testing and ana...	TO DO	SG

Jira Time line (after finishing)

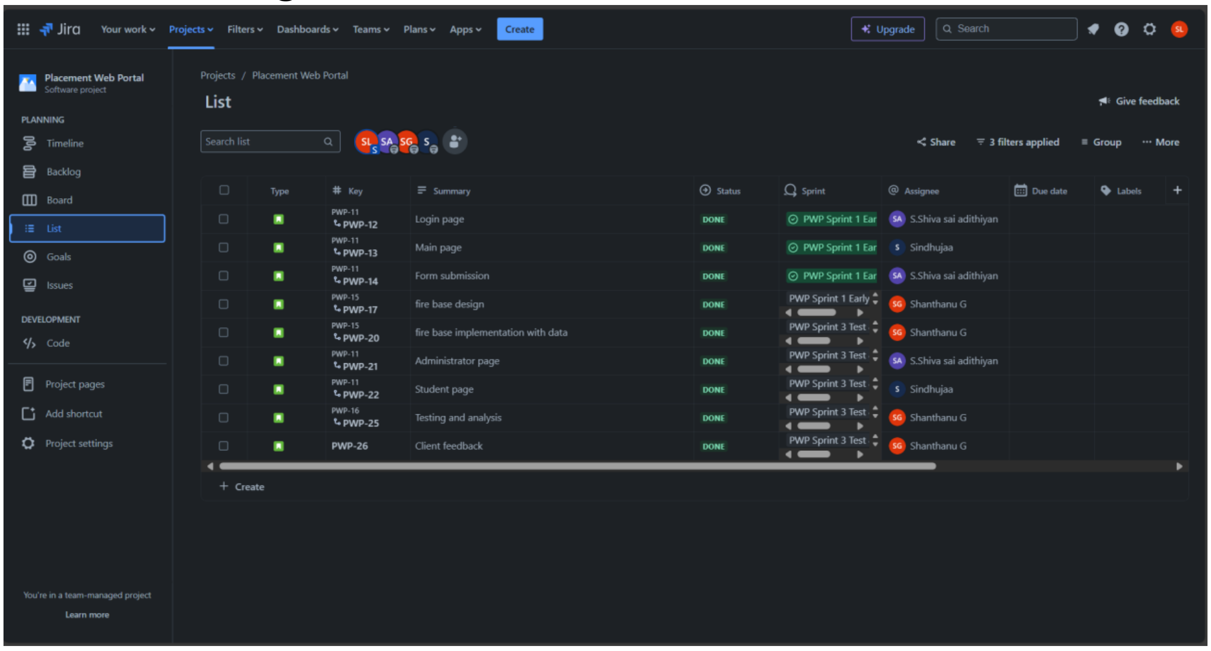


Sprints

~~PWP-11~~ Front end	DONE	
~~PWP-12~~ Login page	DONE	SA
~~PWP-13~~ Main page	DONE	S
~~PWP-14~~ Form submission	DONE	SA
~~PWP-21~~ Administrator p...	DONE	SA
~~PWP-22~~ Student page	DONE	S
~~PWP-15~~ Data base	DONE	
~~PWP-17~~ fire base design	DONE	SG
~~PWP-20~~ fire base imple...	DONE	SG
~~PWP-16~~ Back end	DONE	
~~PWP-19~~ backend django	DONE	SL
~~PWP-24~~ Back end finishi...	DONE	SL
~~PWP-25~~ Testing and anal...	DONE	SG

+ Create Epic

Finished backlogs in list view

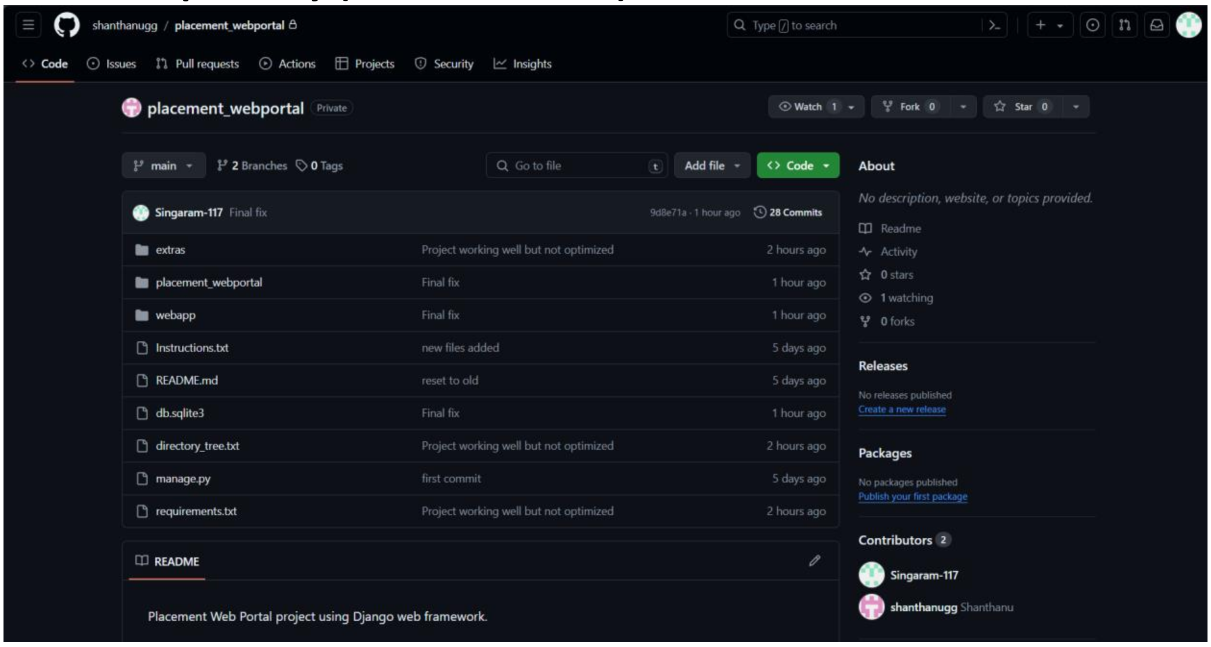


Projects / Placement Web Portal

List

Type	Key	Summary	Status	Sprint	Assignee	Due date	Labels
🟩	PWP-11 PWP-12	Login page	DONE	PWP Sprint 1 Ear	S.Shiva sai adithiyan		
🟩	PWP-11 PWP-13	Main page	DONE	PWP Sprint 1 Ear	Sindhujaa		
🟩	PWP-11 PWP-14	Form submission	DONE	PWP Sprint 1 Ear	S.Shiva sai adithiyan		
🟩	PWP-15 PWP-17	fire base design	DONE	PWP Sprint 1 Early	Shanthanu G		
🟩	PWP-15 PWP-20	fire base implementation with data	DONE	PWP Sprint 3 Test	Shanthanu G		
🟩	PWP-11 PWP-21	Administrator page	DONE	PWP Sprint 3 Test	S.Shiva sai adithiyan		
🟩	PWP-11 PWP-22	Student page	DONE	PWP Sprint 3 Test	Sindhujaa		
🟩	PWP-16 PWP-25	Testing and analysis	DONE	PWP Sprint 3 Test	Shanthanu G		
🟩	PWP-26	Client feedback	DONE	PWP Sprint 3 Test	Shanthanu G		

Git hub repository (version control)



shanthanugg / placement_webportal

placement_webportal Private

main · 2 Branches · 0 Tags

Singaram-117 Final fix · 9d8e71a · 1 hour ago · 28 Commits

extras	Project working well but not optimized	2 hours ago
placement_webportal	Final fix	1 hour ago
webapp	Final fix	1 hour ago
Instructions.txt	new files added	5 days ago
README.md	reset to old	5 days ago
db.sqlite3	Final fix	1 hour ago
directory_tree.txt	Project working well but not optimized	2 hours ago
manage.py	first commit	5 days ago
requirements.txt	Project working well but not optimized	2 hours ago

README

Placement Web Portal project using Django web framework.

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

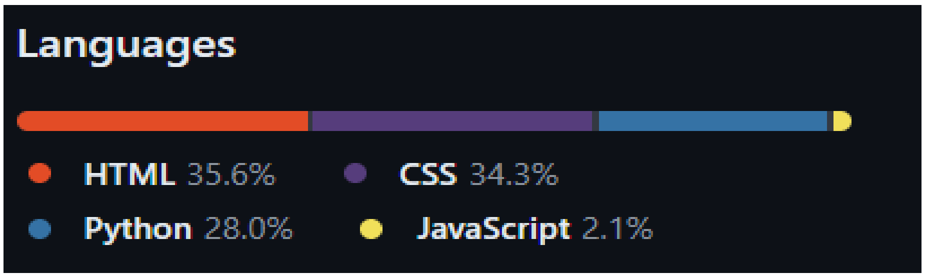
No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 2

Singaram-117
shanthanugg Shanthanu



Languages

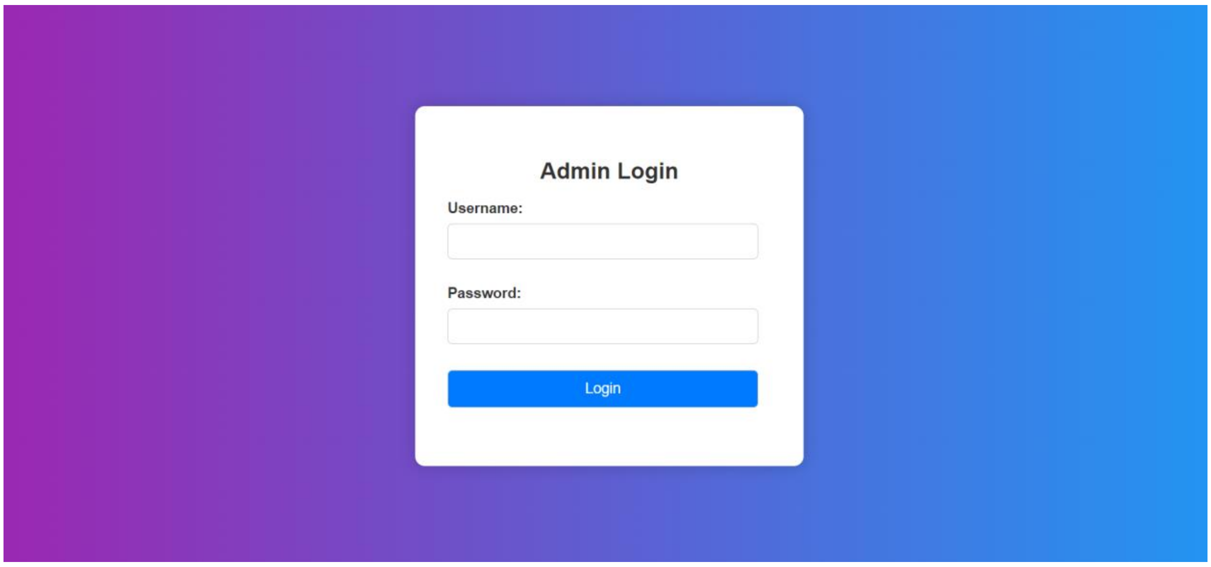
● HTML 35.6% ● CSS 34.3%
● Python 28.0% ● JavaScript 2.1%

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={F
IREBASE_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)
```

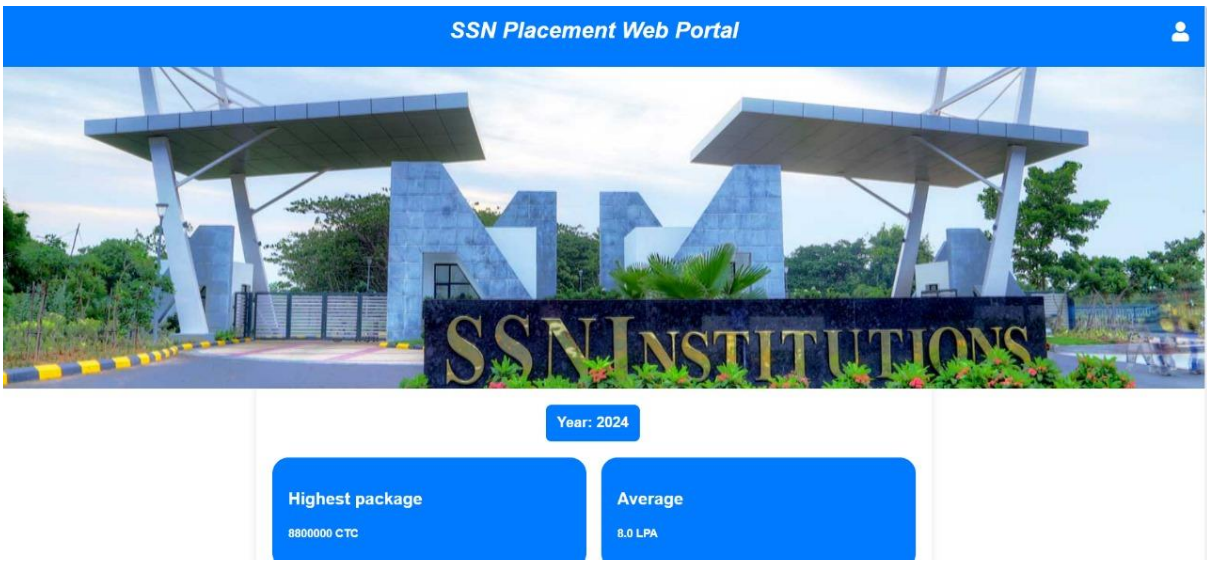
Results:


Admin Login

Username:

Password:

Login


SSN Placement Web Portal

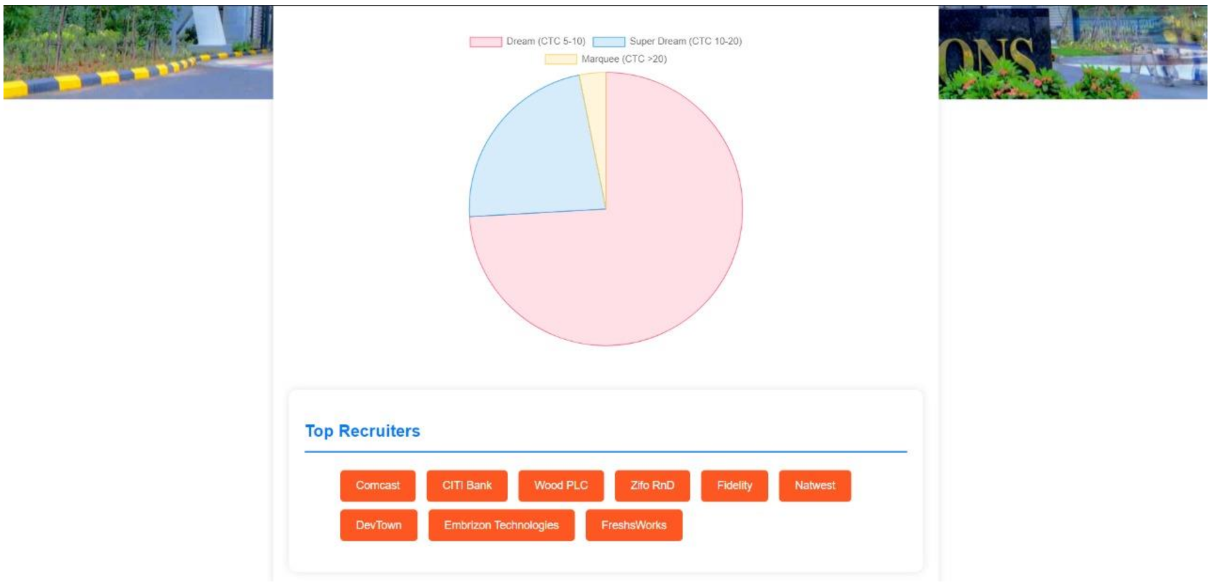
Year: 2024

Highest package

8800000 CTC

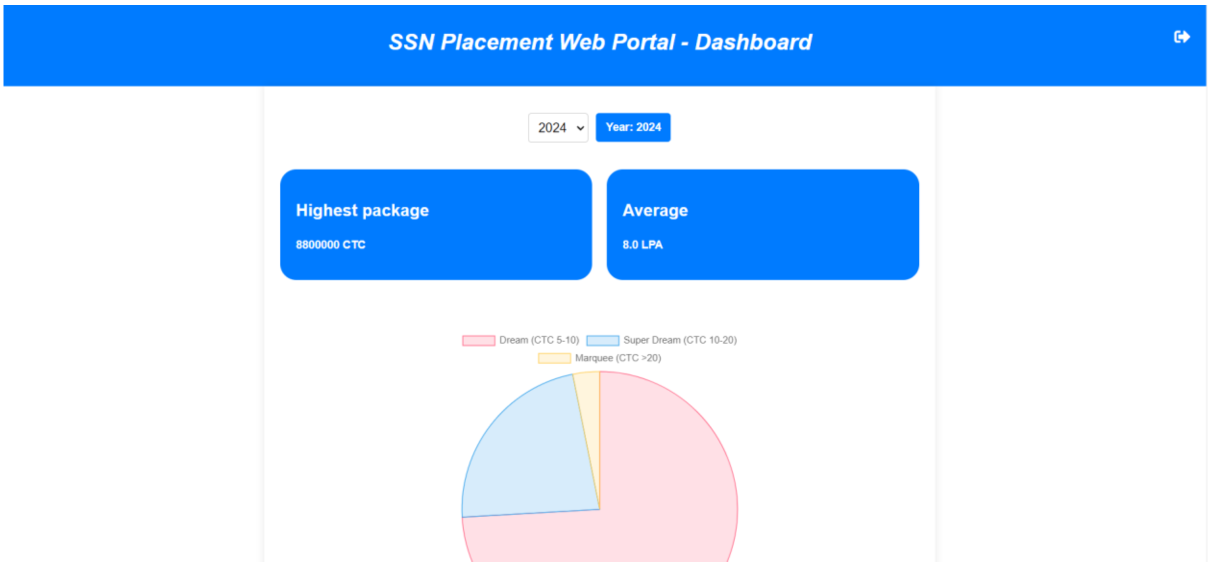
Average

8.0 LPA



Top Recruiters

Comcast
CITI Bank
Wood PLC
Zifo RnD
Fidelity
Natwest
DevTown
Embrizon Technologies
FreshsWorks

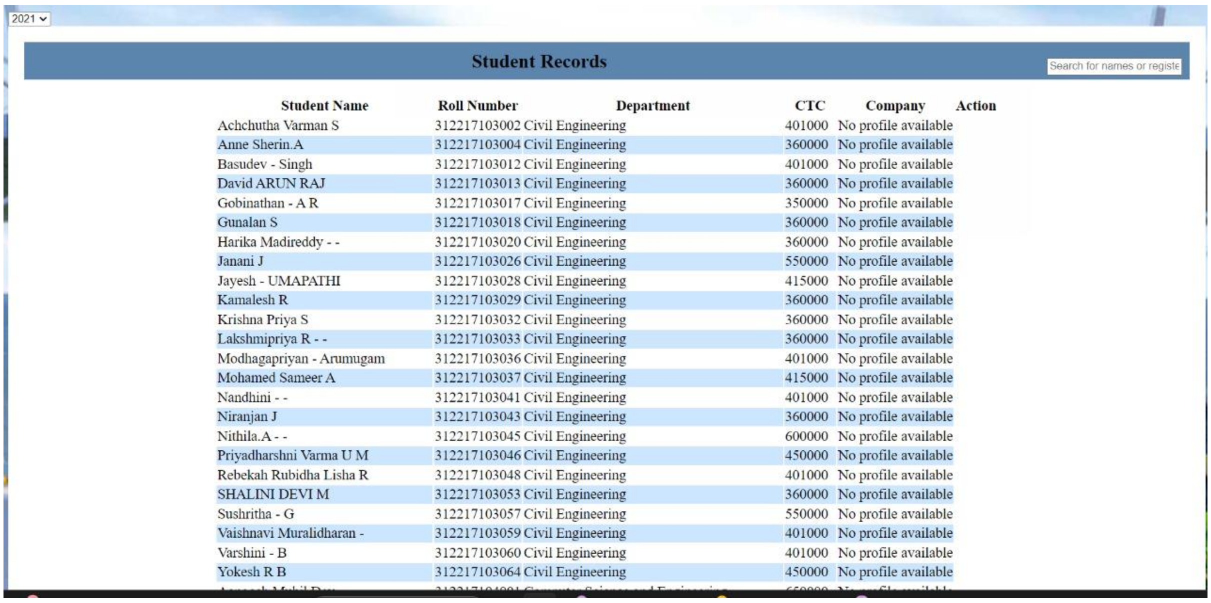

SSN Placement Web Portal - Dashboard

Highest package

8800000 CTC

Average

8.0 LPA



Student Records

Student Name	Roll Number	Department	CTC	Company	Action
Achchutha Varman S	312217103002	Civil Engineering	401000	No profile available	
Anne Sherin.A	312217103004	Civil Engineering	360000	No profile available	
Basudev - Singh	312217103012	Civil Engineering	401000	No profile available	
David ARUN RAJ	312217103013	Civil Engineering	360000	No profile available	
Gobinathan - A R	312217103017	Civil Engineering	350000	No profile available	
Gunalan S	312217103018	Civil Engineering	360000	No profile available	
Harika Madireddy - -	312217103020	Civil Engineering	360000	No profile available	
Janani J	312217103026	Civil Engineering	550000	No profile available	
Jayesh - UMAPATHI	312217103028	Civil Engineering	415000	No profile available	
Kamalesh R	312217103029	Civil Engineering	360000	No profile available	
Krishna Priya S	312217103032	Civil Engineering	360000	No profile available	
Lakshmipriya R - -	312217103033	Civil Engineering	360000	No profile available	
Modhagapriyan - Arumugam	312217103036	Civil Engineering	401000	No profile available	
Mohamed Sameer A	312217103037	Civil Engineering	415000	No profile available	
Nandhini - -	312217103041	Civil Engineering	401000	No profile available	
Niranjan J	312217103043	Civil Engineering	360000	No profile available	
Nithila.A - -	312217103045	Civil Engineering	600000	No profile available	
Priyadharshni Varma U M	312217103046	Civil Engineering	450000	No profile available	
Rebekah Rubidha Lisha R	312217103048	Civil Engineering	401000	No profile available	
SHALINI DEVI M	312217103053	Civil Engineering	360000	No profile available	
Sushritha - G	312217103057	Civil Engineering	550000	No profile available	
Vaishnavi Muralidharan -	312217103059	Civil Engineering	401000	No profile available	
Varshini - B	312217103060	Civil Engineering	401000	No profile available	
Yokesh R B	312217103064	Civil Engineering	450000	No profile available	

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 by freeCodeCamp.org
   - CSS Crash Course For Absolute Beginners by Traversy Media
2. **Websites**
   - MDN Web Docs: HTML
   - MDN Web Docs: CSS

Python Django

1. **YouTube**
   - Django Crash Course by Traversy Media
   - Django for Beginners by The Net Ninja
2. **Websites**
   - Django Official Documentation
3. **GitHub**
   - Django Official Repository
   - Django Rest Framework

Firebase

1. **YouTube**
   - Firebase Firestore Tutorial by Academind
   - Firebase Authentication Tutorial by The Net Ninja

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: