A PROJECT DOCUMENT ON

Placement Tracker Web (Dashboard)

BACHELOR OF TECHNOLOGY

In

COMPUTER ENGINNERING & ARTIFICIAL INTELIIGENCE Submitted By

##TeamMembers

|V V D Mahesh Peruri(24A35A4316) | Frontend & Backend developer, |

|Kadali Venu(24A35A4314) | Frontend Developer |

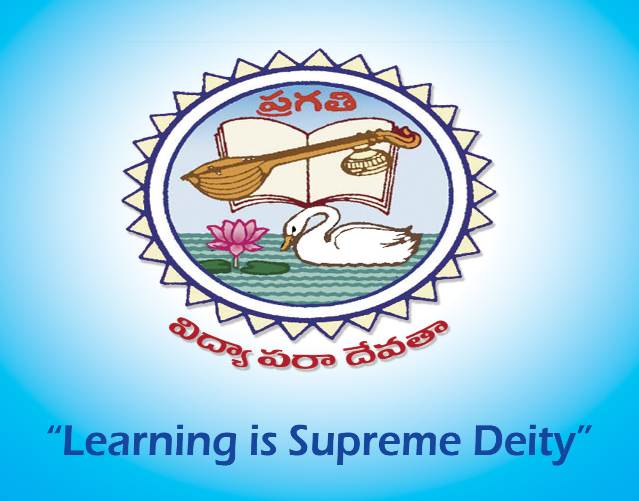
|Tirumareddi Durga Prasad(24A35A4318) | Backend Developer |

| Muralasetti Naga Chakra Satya Sai(24A35A4315) | Data Collector |

|Velanati Veera Venkata Sai(24A35A4319) | Document Creator|

PRAGATI ENGINERRING COLLEGE

JAN 2025



Placement Tracker Web (Dashboard)

Firebase

JavaScript

HTML

CSS

**“” The Placement Tracker Web is a web application designed to help students and administrators track placement-related activities, such as job applications, interviews, and offers. It is built using Firebase for backend services (Firestore, Authentication, Hosting) and HTML/CSS/JavaScript for the frontend**.””

Features:

User Authentication: Sign up, log in, and manage user profiles.

Placement Tracking: Add, update, and delete placement records.

Real-time Database: Store and retrieve data using Firebase Firestore.

Responsive Design: Works on all devices (desktop, tablet, mobile).

Technologies Used

Frontend: HTML, CSS, JavaScript

Backend: Firebase (Firestore, Authentication, Hosting)

Version Control: Git

Setup Instructions:

1. Prerequisites

Node.js (v16 or higher)

Firebase CLI (npm install -g firebase-tools)

A Firebase project (create one at Firebase Console)

2. Clone the Repository

bash

Copy

git clone https://github.com/your-username/placement-tracker-web.git

cd placement-tracker-web

3. Install Dependencies

npm install firebase

4. Set Up Firebase

Go to the Firebase Console.

Create a new project (e.g., placement-tracker-web).

Enable Firestore, Authentication, and Hosting.

Add your Firebase configuration to firebase.js:

// Import the functions you need from the SDKs you need

import { initializeApp } from "firebase/app";

import { getAnalytics } from "firebase/analytics";

// TODO: Add SDKs for Firebase products that you want to use

// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration

// For Firebase JS SDK v7.20.0 and later, measurementId is optional

const firebaseConfig = {

apiKey: "AIzaSyDYi0oWtn\_W7RFp-qSAt8LZP\_iqqVwxFrw",

authDomain: "placement-tracker-web.firebaseapp.com",

projectId: "placement-tracker-web",

storageBucket: "placement-tracker-web.firebasestorage.app",

messagingSenderId: "137204240039",

appId: "1:137204240039:web:9c0f54802adc4111fb5582",

measurementId: "G-884GW6G6KL"

};

// Initialize Firebase

const app = initializeApp(firebaseConfig);

const analytics = getAnalytics(app);5. Run the Project Locally

npm start

Open your browser and go to http://localhost:3000.

6. Deploy to Firebase Hosting

firebase login

firebase init

firebase deploy

Your app will be live at https://your-project-id.web.app.

Usage:

Sign Up: Create a new account using your email and password.

Log In: Access your dashboard after logging in.

Add Placement Records: Use the form to add new placement records.

View Records: See all your placement records in the dashboard.

Edit/Delete Records: Update or delete records as needed.

Project Structure:

placement-tracker-web/(Firebase console)

├── public/              # Static files (HTML, CSS, images)

├── src/                 # JavaScript files

│   ├── firebase.js      # Firebase initialization

│   ├── app.js           # Main application logic

│   └── auth.js          # Authentication logic

├── firebase.json        # Firebase configuration

├── package.json         # Node.js dependencies

└── README.md            # Project documentation

/placement-tracker-dashboard (VSCODE)  
  ├── index.html                  # Landing Page  
  ├── login.html                  # Login Page  
  ├── signup.html                 # Signup Page  
  ├── dashboard.html              # Dashboard Page  
  ├── applications.html           # Application Tracker Page  
  ├── interviews.html             # Interview Schedule Page  
  ├── profile.html                # Profile Page  
  ├── assets/  
      ├── css/  
          ├── style.css           # Global CSS (shared styles)  
          ├── landing.css         # Specific CSS for Landing Page  
          ├── login.css           # Specific CSS for Login Page  
          ├── signup.css          # Specific CSS for Signup Page  
          ├── dashboard.css       # Specific CSS for Dashboard Page  
          ├── applications.css    # Specific CSS for Application Tracker Page  
          ├── interviews.css      # Specific CSS for Interview Schedule Page  
          ├── profile.css         # Specific CSS for Profile Page  
      ├── js/  
          ├── script.js           # Global JavaScript (shared functionality)  
          ├── login.js            # Specific JS for Login Page  
          ├── signup.js           # Specific JS for Signup Page  
          ├── dashboard.js        # Specific JS for Dashboard Page  
          ├── applications.js     # Specific JS for Application Tracker Page  
          ├── interviews.js       # Specific JS for Interview Schedule Page  
          ├── profile.js          # Specific JS for Profile Page  
      ├── images/                 # All images used in the website  
      ├── videos/                 # All videos used in the website  
      └── fonts/                  # Custom fonts (if any)

Future Features:

1. AI-Powered Resume Feedback

Feature: Integrate an AI tool (e.g., OpenAI API) to analyze the user’s resume and provide feedback.

How It Works: Users can upload their resume, and the AI provides suggestions (e.g., improve formatting, add skills, tailor for specific job roles).

Why It’s Unique: Helps students improve their resumes, increasing their chances of getting placements.

2. Job Recommendation System

Feature: Recommend job roles and companies based on the user’s profile and preferences.

How It Works: Users input their skills, interests, and preferred job roles, and the system recommends relevant job openings.

Why It’s Unique: Personalization makes the dashboard more user-friendly and practical.

3. Interview Preparation Tool

Feature: Provide a list of common interview questions and tips for specific job roles.

How It Works: Users select a job role, and the system displays common interview questions and sample answers. Include a mock interview feature where users can record themselves answering questions.

Why It’s Unique: Helps students prepare effectively for interviews.

4. Progress Analytics and Insights

Feature: Visualize placement progress with charts and graphs.

How It Works: Use a library like Chart.js or D3.js to create visualizations (e.g., pie charts for application status, bar graphs for interview performance).

Why It’s Unique: Data visualization makes the dashboard more engaging and informative.

5. Reminder and Notification System

Feature: Send reminders for upcoming interviews and application deadlines.

How It Works: Integrate with email or SMS APIs (e.g., Twilio, SendGrid) to send reminders.

Why It’s Unique: Keeps users organized and ensures they don’t miss important deadlines.

6. Skill Gap Analysis

Feature: Analyze the user’s skills and suggest areas for improvement.

How It Works: Users input their skills and desired job roles, and the system compares their skills with the requirements of the desired roles.

Why It’s Unique: Helps students identify and bridge skill gaps.

7. Collaboration and Networking

Feature: Allow users to connect with peers and share placement experiences.

How It Works: Add a forum or chat feature where users can discuss placement strategies, share interview experiences, and ask questions.

Why It’s Unique: Encourages collaboration and community building.

8. Gamification

Feature: Add gamified elements to motivate users.

How It Works: Award badges or points for completing tasks (e.g., “Applied to 5 companies”, “Cleared 3 interviews”).

Why It’s Unique: Makes the dashboard more engaging and fun to use.

9. Integration with Job Portals

Feature: Integrate with job portals (e.g., LinkedIn, Indeed) to fetch job openings directly.

How It Works: Use APIs to fetch job listings based on user preferences.

Why It’s Unique: Saves time and provides a seamless experience.

10. Dark Mode and Custom Themes

Feature: Allow users to customize the dashboard’s appearance.

How It Works: Add a toggle for dark mode and allow users to choose from different color themes.

Why It’s Unique: Improves user experience and makes the dashboard visually appealing.

11. Export Data

Feature: Allow users to export their placement data (e.g., applications, interviews) as a PDF or CSV file.

How It Works: Add a button to export data for offline use or sharing.

Why It’s Unique: Adds convenience and flexibility for users.

12. Multi-Language Support

Feature: Support multiple languages to cater to a wider audience.

How It Works: Add a language switcher to change the dashboard’s language (e.g., English, Hindi, Spanish).

Why It’s Unique: Makes the dashboard accessible to non-English speakers.

13. Offline Mode

Feature: Allow users to access their data offline.

How It Works: Use localStorage or IndexedDB to store data locally and sync data when the user reconnects to the internet.

Why It’s Unique: Ensures users can access their data even without an internet connection.

14. Voice Commands

Feature: Allow users to interact with the dashboard using voice commands.

How It Works: Use the Web Speech API to enable voice-based navigation and actions (e.g., “Show my applications”, “Add a new interview”).

Why It’s Unique: Adds a futuristic and innovative touch.

15. AI-Powered Interview Simulator

Feature: Simulate real interviews using AI.

How It Works: Users can practice answering interview questions, and the AI evaluates their responses and provides feedback.

Why It’s Unique: Provides a realistic interview practice experience.

Future Steps for Upgrading the Website:

1. Improve UI/UX

Modern Design: Redesign the website with a modern, user-friendly interface.

Responsive Layout: Ensure the website works seamlessly on all devices (desktop, tablet, mobile).

2. Performance Optimization

Lazy Loading: Implement lazy loading for images and data to improve page load times.

Caching: Use browser caching and server-side caching to reduce load times.

3. Enhance Security

Firebase Security Rules: Tighten Firestore and Storage security rules for production

App Check: Enable Firebase App Check to prevent abuse and unauthorized access.

4. Add More Firebase Services

Cloud Functions: Use Firebase Cloud Functions for backend logic (e.g., sending emails, processing data).

Cloud Messaging: Implement Firebase Cloud Messaging (FCM) for push notifications.

5. Integrate Third-Party APIs

Job Portals: Integrate with job portals like LinkedIn, Naukri, or Indeed to fetch job postings.

Calendar Integration: Sync placement schedules with Google Calendar or Outlook.

6. Multi-Language Support

Localization: Add support for multiple languages to make the platform accessible to a wider audience.

Creating a Mobile Application:

1. Choose a Framework

React Native: A popular framework for building cross-platform mobile apps using JavaScript.

Flutter: A framework by Google for building natively compiled apps for mobile, web, and desktop from a single codebase.

2. Set Up Firebase for Mobile

Add Firebase to Your Mobile App: Follow Firebase’s documentation to integrate Firebase services (Firestore, Authentication, etc.) into your mobile app.

Enable Mobile Authentication: Add support for phone number authentication, Google Sign-In, and other methods.

3. Reuse Existing Code

Shared Logic: Reuse the Firebase logic (e.g., Firestore queries, authentication) from your web app.

API Integration: Use the same backend APIs for both the web and mobile apps.

4. Design Mobile-Specific Features

Offline Support: Use Firestore’s offline persistence to allow users to access data without an internet connection.

Push Notifications: Implement push notifications using Firebase Cloud Messaging (FCM).

Camera Integration: Allow users to upload documents or photos (e.g., resumes, certificates) directly from their phones.

5. Test and Deploy

Testing: Test the app on both Android and iOS devices.

Deployment: Publish the app on the Google Play Store and Apple App Store.

License:

This project is licensed under the MIT License. See the LICENSE file for details.

Acknowledgments:

Firebase for providing an excellent backend-as-a-service.

MDN Web Docs for JavaScript and web development resources.