**Problem Statement : Campus Placement & Internship Tracker**

**PHASE 1**

**Phase 1 : Problem Understanding & Industry Analysis**

* Current Issues (colleges):
  + Manual tracking of student applications in Excel sheets.
  + Recruiters receive CVs in bulk → difficult to filter the right students.
  + No automated communication for shortlisting, interviews, or offers.
  + Lack of placement analytics (branch-wise, recruiter-wise, student skill gaps).
* Solution must provide:
  + Centralized student database (skills, resumes, certifications).
  + Job posting portal for recruiters.
  + Smart student–job matching engine.
  + Application tracking workflow (Applied → Shortlisted → Interview → Selected).
  + Dashboards for TPO/Admin.
  + Automated reminders & notifications.

1. **Stakeholder Analysis**

* Recruiters = Need to find best-fit students quickly and communicate efficiently.
* Students = Want a clear way to apply, track progress, and improve their chances.
* Training & Placement Office (TPO/Admin) = Needs visibility into overall placement statistics, recruiter participation, and student performance.
* College Management = Wants reports on placement rates and recruiter engagement.

**3 Business Process Mapping (Current vs Proposed)**

**Current Flow (Manual):**

* Recruiter sends job description via email.
* TPO shares JD with students.
* Students apply via forms/emails.
* TPO shortlists manually & sends to recruiter.
* Interviews scheduled over phone/email.
* Placement report prepared manually.

**Proposed Flow with Career Hive 360:**

* Recruiter posts job directly in Salesforce (Job\_\_c).
* Students view/apply through portal → Application\_\_c auto-created.
* System assigns match score (Apex) → recruiter sees top Students.
* Flows trigger reminders (shortlist, interview invites).
* TPO monitors dashboards for real-time stats.
* Automated reports reduce manual effort.

**4 Industry-Specific Use Case Analysis**

* Education & Placement Industry Trends:
  + Rising number of students = scaling issues for placement cells.
  + Companies expect skill-based hiring (AI/ML-driven shortlisting).
  + Universities demand analytics for accreditation and ranking (NIRF, NAAC).
* Career Hive 360 Use Case:
  + Skill-based placement automation (unique differentiator).
  + Gamified student dashboards showing readiness score.
  + Branch-wise analytics → helps TPO & management with decisions**.**

**5 AppExchange Exploration**

* Look at similar apps on Salesforce AppExchange (e.g., Job Boards, Recruiting Apps).
* Findings: Most are generic HR solutions (Workday, Bullhorn, JobScience).
* Gap: None focus specifically on campus placements + student–recruiter ecosystem.
* Differentiator for Career Hive 360:
  + Tailored to colleges & universities.
  + Student-centric dashboards + recruiter-centric matching in a single app.
  + Lightweight and easy compared to enterprise HRMS.

**PHASE 2**

**Phase 2 : Org Setup & Configuration**

**Implemented**

* **User Setup & Licenses**
  + Created different users (Recruiters, Students, TPO/Admin) with appropriate Salesforce licenses.
  + Assigned relevant roles & permission sets.
* **Profiles**
  + Configured custom profiles for Recruiters, Students, and TPO/Admin.
  + Controlled object- and field-level access (CRUD/FLS).
* **Roles**
  + Designed a role hierarchy (TPO/Admin → Recruiter → Student) to control data visibility.
* **Permission Sets**
  + Built focused permission sets:
    - Manage Job Applications → gives Recruiters the ability to create and update applications.
    - Candidate Portal Access → allows Students to view jobs, apply, and edit only their profile/resume fields.
* **OWD (Organization-Wide Defaults)**
  + Set Student\_\_c to Private → ensures each student only sees/edits their own record.
  + Controlled sharing for Job Applications and Positions appropriately.
* **Sharing Rules**
  + Added rules so recruiters can view applications related to positions they own.
* **Login Access Policies / Security**
  + Tested Login IP Ranges for Recruiter profiles.
  + Ensures recruiters can only log in from approved networks (corporate/campus).
* **Dev Org Setup**
  + Project built and tested inside a Salesforce Developer Org.
  + Configured metadata like objects, fields, validation rules, and flows.

**PHASE 3**

**Phase 3 : Data Modeling & Relationships**

**Implemented**

* **Custom Objects**
  + Student\_\_c → stores student details (skills, resume link, contact info).
  + Job\_\_c → job postings by recruiters.
  + Application\_\_c → junction object .
* **Fields**
  + Student\_\_c → Skills, Resume\_Link, Email, Contact Number.
  + Application\_\_c → Application Status, Stage, Notes.
* **Page Layouts**
  + Separate layouts for Student, Position, and Job Application objects.
  + Layouts tailored for each role (Recruiters see more job details, Students see their own profile/application).
* **Relationships**
  + Lookup/Master-Detail Relationships configured:
    - Student ↔ Application ↔ Job.
  + Junction Object (Application\_\_c) connects many Students to many Positions.

**PHASE 4**

**Phase 4 : Process Automation (Admin)**

* **What was created:**
  + **Record-Triggered Flows** to:
    - Auto-assign applications to recruiters.
    - Fetch Owner details dynamically (Job Owner, Recruiter Owner).
  + Conditional logic for approvals and reminders.
* **Purpose & Functionality:**
  + Reduces manual intervention by automating repetitive tasks (assignment, notifications).
  + Guarantees a smoother placement cycle: Applied → Shortlisted → Interview → Selected.
  + Adds intelligence and speed to the system.

**PHASE 5**

**Phase 5 : Apex Programming (Developer)**

**Implemented**

* Apex Classes & Objects
  + Created **MatchScoreHandler** class to compute the suitability score between a Student and a Job based on skill matches.
  + Designed **ApplicationPostProcessor** as a Queueable class to handle asynchronous post-processing after new Applications are submitted.
* Apex Trigger with Design Pattern
  + Developed **ApplicationTrigger on Application\_\_c** object, following the trigger design pattern:
    - *Before Insert/Update*: Calls MatchScoreHandler to compute Match\_Score\_\_c.
    - *After Insert*: Enqueues ApplicationPostProcessor for async execution.
* Control Statements
  + 1. Applied conditional checks and loops to compute skill overlap between Student and Job.
    2. Logic assigns 10 points per matched skill, producing a score range from 0 up to N×10 (where N = number of required skills).
* Asynchronous Processing
  + Implemented **Queueable Apex** (ApplicationPostProcessor) to handle operations that can run outside the transaction (e.g., logging, future notifications, or record sharing).
* Exception Handling
  + Incorporated try–catch structures and safe null checks in MatchScoreHandler to avoid runtime errors when skills are blank.
* Test Classes
  + Built MatchScoreHandlerTest to validate the logic:
    - Creates sample Student and Job records.
    - Inserts an Application and verifies that Match\_Score\_\_c is computed correctly.
    - Ensures test coverage >75%, meeting Salesforce deployment requirements.

**PHASE 6**

**Phase 6 : User Interface Development**

* **Lightning App Builder:**
  + Customized both Home Page (with Reports, List Views, Rich Text welcome message) and Record Page (Application Record Page).
  + Organized content for recruiters and students in a visually engaging way.
* **Record Pages:**
  + Built a custom **Application Record Page**.
  + Added tabs for Details, Related, Activities, and Skill Match Viewer (custom LWC).
* **Tabs:**
  + Configured record page tabs to group standard Salesforce components (Details, Related Lists, Activity Timeline) and custom ones (Skill Match Viewer).
* **Home Page Layouts:**
  + Embedded Student Application Status Overview report, Open Jobs List View, and a professional Rich Text introduction to CareerHive360.
* **LWC (Lightning Web Components):**
  + Developed a custom component **skillMatchViewer** that dynamically displays Student Skills, Job Requirements, and Match Score.
  + Added styling for better readability and branding.
* **Apex with LWC:**
  + Connected LWC with **ApplicationDataController** Apex class to fetch data using @AuraEnabled method.

**PHASE 7**

**Phase 7 : Integration & External Access**

Platform Event Creation:

* Created a Platform Event **ApplicationCreatedEvent\_\_e** with custom fields: ApplicationName\_\_c, StudentName\_\_c, and JobTitle\_\_c.
* An Apex Trigger publishes this event whenever a new Application\_\_c record is inserted.

Email Notification Setup:

* Email Template: **Lightning Email Template** named Application Notification. Merge fields are selected via the Merge Field Picker for StudentName\_\_c, JobTitle\_\_c, and ApplicationName\_\_c.
* Email Alert: Created Notify Recruiter on Application tied to Application\_\_c. Recipients include demo recruiter users (or own email), and sender is either default noreply@salesforce.com or an Org-Wide Email Address (careerhive360@college.edu).

Flow Integration:

* **Platform Event–Triggered Flow** listens for ApplicationCreatedEvent\_\_e.
* Flow Action calls the Email Alert to notify recruiters automatically.
* Flow cannot be run manually; it triggers only when the event is published.

Testing:

* Creating a new Application\_\_c record publishes the event.
* Flow detects the event and sends the email.
* Verified email is received in inbox, confirming successful notification setup.

**PHASE 8**

**Phase 8 : Data Management & Deployment**

**Duplicate Rules**

* **What was implemented:**
  + Created **Matching Rules** to check specific fields like Email\_\_c for Students
  + Built **Duplicate Rules** that use these matching rules to stop or warn users when trying to insert/update duplicate records.
* **Purpose & Functionality:**
  + Prevents multiple student records with the same email/phone → avoids confusion for recruiters.
  + Stops recruiters or admins from posting duplicate job positions → ensures a clean job listing experience.
  + Maintains a **single source of truth** for every student

**PHASE 9**

**Phase 9 : Reporting, Dashboards & Security Review**

* **At Profile Level (best for recruiters/admins)**
* Select the Profile (e.g., *Recruiter Profile*, *TPO Admin Profile*).
* **Login IP Ranges** → **Add Range**.
  + **Start IP Address** → e.g., 192.168.1.1
  + **End IP Address** → e.g., 192.168.1.255

This means recruiters can only log in from a **college campus network** or **corporate network only.**

* **Reports And Dashboards :** 
  + **Reports:**
    - *Job Applications by Status* (applied, shortlisted, selected).
    - *Applications per Position* (recruiter demand vs student interest).
    - *Recruiter Performance* (applications handled, offers made).
  + **Dashboard** plan: CareerHive Hiring Dashboard (Funnel).

**Purpose & Functionality:**

* + Gives TPO/Admin real-time visibility into placements.
  + Helps recruiters identify bottlenecks in hiring.
  + Provides colleges with branch-wise and recruiter-wise analytics for accreditation (NAAC, NIRF).

**PHASE 10**

**Phase 10: Final Presentation & Demo Day**