International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# Placepulse Intelligence

**Karthickraja M** - **(6176AC23UCS074)**

**Karunakaran K** - **(6176AC23UCS077)**

**Lokesh S** - **(6176AC23UCS090)**

**Mohamed Yusuf Asfar Sa** - **(6176AC23UCS098)**

Bachelor of Engineering

Computer Science and Engineering

Adhiyamaan College of Engineering (An Autonomous Institution)

Dr. M.G.R Nagar, Hosur-635130

Anna University :: Chennai 600 025
November 2026



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# ABSTRACT

Campus placement management and recruitment analytics represent critical operational pillars for higher technical education institutions. Across autonomous engineering colleges, tracking the multi-stage recruitment lifecycle of hundreds of students across multiple engineering departments—including Computer Science and Engineering (CSE), Information Technology (IT), Electronics and Communication Engineering (ECE), Electrical and Electronics Engineering (EEE), Mechanical Engineering (MECH), and Civil Engineering (CIVIL)—is traditionally burdened by fragmented spreadsheets, siloed departmental registries, manual computational errors, and delayed reporting mechanisms. Crucial metrics such as unique placed student ratios, departmental conversion percentages, compensation salary tier distributions (Dream, Super Dream, and Mass recruiter offers), and experiential internship stipends often lack real-time synchronization, hindering data-driven institutional decision making and National Board of Accreditation (NBA) / National Institutional Ranking Framework (NIRF) reporting.

To solve this systemic institutional challenge, this project presents the complete design, architectural modelling, and functional implementation of an enterprise-grade Place-pulse intelligence. Developed as an integrated three-tier full-stack system, the platform combines a modern presentation layer built with React 18 Single Page Application (SPA), TypeScript, Tailwind CSS, and Framer Motion, supported by a robust, asynchronous application service layer implemented in Node.js and Express.js, and an embedded, ACID-compliant SQLite3 relational persistence engine.

**KEYWORDS:** Campus Placement Analytics, Student Information System, Full-Stack Architecture, React 18 SPA, Node.js Express, SQLite3 Embedded Database, Relational Data Modelling, Compensation Metrics, Key Performance Indicators (KPIs), Educational Technology, Accreditation Reporting.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 1
# INTRODUCTION

## 1.1 BACKGROUND OF CAMPUS PLACEMENT IN TECHNICAL INSTITUTIONS

Campus placement is the process through which companies visit an educational institution, evaluate final-year students through aptitude tests, technical rounds and interviews, and offer employment before graduation. For engineering colleges in India, placement results are one of the most visible indicators of institutional quality. Parents and students use them to choose a college, and statutory and ranking bodies such as the National Board of Accreditation (NBA)

A single placement season involves many moving parts. Dozens of recruiters announce drives, each with its own eligibility rules, role descriptions and compensation packages. Hundreds of students from several departments apply, appear for multiple rounds, and may receive more than one offer.

In most institutions these records are still kept in spreadsheets maintained separately by the placement cell and by each department. A student who receives two offers may be counted twice, a department may count a student whom the placement cell has marked as ineligible, and an average package may include offers that were never accepted.

## 1.2 INSTITUTIONAL CONTEXT: ADHIYAMAAN COLLEGE OF ENGINEERING

Adhiyamaan College of Engineering, Hosur, is an autonomous institution affiliated to Anna University, Chennai. The college offers undergraduate programmes in six engineering disciplines that take part in campus recruitment: Computer Science and Engineering (CSE), Information Technology (IT), Electronics and Communication Engineering (ECE), Electrical and Electronics Engineering (EEE), Mechanical Engineering (MECH) and Civil Engineering (CIVIL). Each department has its own placement coordinator, while the central placement cell negotiates with recruiters and schedules drives.

This structure is typical of a large autonomous college, and it is exactly the setting in which fragmented record keeping causes trouble.

## 1.3 PROBLEM STATEMENT AND OPERATIONAL VULNERABILITIES

The current practice of tracking placements in separate spreadsheets exposes the institution to several operational vulnerabilities.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**Fragmented records:** Student, company and offer data are stored in different files, so a question such as "which Dream-tier recruiters hired from ECE" requires manual cross-referencing.

**Double counting:** A student with several offers appears several times, which inflates the number of placed students and distorts the placement percentage.

**Manual calculation errors:** Percentages, averages and medians are computed by hand, and a single wrong cell changes the reported figure.

**Delayed reporting:** Consolidated figures are available only after the coordinators exchange files, so decisions are taken on outdated information.

The problem addressed in this project is therefore how to replace these scattered records with one reliable source of truth and a live analytics layer that always reflects the current state of the data.

### 1.4 OBJECTIVES OF THE PROPOSED SYSTEM

The main objective of Place-Pulse Intelligence is to provide a single, real-time platform that records every placement event in a relational database and turns it into accurate, drilldown analytics for the placement cell, the departments and the students.

1. To design a normalised relational schema for departments, students, companies, placement drives, offers and internships.
2. To build an Express.js REST API that exposes dashboard, company, student, analytics and internship services in a consistent JSON format.
3. To develop six interactive analytics cards: Total Recruiters, Students Placed, Placement Percentage, Highest Package, Average Package and Internship Offers.
4. To count unique placed students correctly even when a student holds multiple offers, and to separate accepted offers from declined offers.
5. To validate the system with an automated test suite that covers every card and every important query.
6. To launch the whole stack with a single command so that the system is easy to demonstrate and deploy.

### 1.5 SCOPE AND OPERATIONAL BOUNDARIES

The scope of the project covers the analysis, design, implementation and testing of a three-tier web application for placement analytics. The following items are inside the scope.

A React and TypeScript single-page application with a dashboard, six analytics pages, detail profiles and a student database page.

An Express.js backend with controllers, routes, services and middleware.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

An embedded SQLite database with automatic schema creation, seed data and an optional MySQL engine.

The following items are outside the scope of the present version: user authentication and role-based access control, e-mail or SMS notifications, automatic parsing of recruiter job descriptions, machine-learning placement prediction, and integration with the official university examination database. They are listed as future enhancements in Chapter 7.

### 1.6 ORGANIZATION OF THE PROJECT REPORT

**Chapter 1 (Introduction)** presents the background, the problem, the objectives and the scope.

**Chapter 2 (Literature Survey)** reviews published work on educational data mining, analytics dashboards, relational data modelling and web architectures, and identifies the research gaps.

**Chapter 3 (System Analysis and Architectural Design)** describes the existing and proposed systems, the three-tier architecture, the mathematical models, the data flow diagrams, the entity-relationship model and the sequence model.

**Chapter 4 (System Requirements and Specifications)** lists the hardware and software requirements and the technical stack.

**Chapter 5 (System Design and Implementation)** explains the database schema, normalisation, functional subsystems, algorithms, REST API and key source code.

**Chapter 6 (Testing, Verification and Results)** presents the testing methods, test cases, performance observations and the user interface walkthrough.

**Chapter 7 (Conclusion and Future Enhancements)** summarises the work and proposes extensions. The Appendices give the project structure, the reference source code and a user manual, followed by the References.

### 1.7 SIGNIFICANCE OF THE PROJECT

The project is significant for several groups. For the **placement cell**, it removes the manual consolidation of spreadsheets and gives accurate figures at any moment. For **heads of department**, it allows the performance of a department to be compared with other departments and with earlier years. For the **principal and management**, it provides consistent data for NBA and NIRF submissions and for institutional planning. For **students**, it makes visible which companies recruit, which packages are offered and how many students are placed, which helps them to prepare for the next season.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 2
# LITERATURE SURVEY

## 2.1 Smart Career Navigator: AI-Powered Placement Forecasting and Career Planning (2026)

P. John and R. Renuka (2026) proposed a Smart Career Navigator that focuses on artificial intelligence-based placement forecasting and career planning. The study highlights the importance of using intelligent technologies to analyze students' academic performance, technical skills, and career interests. AI-based prediction can help students understand their placement opportunities and identify the skills required to improve their employability. This approach supports personalized career guidance and helps placement officers make better decisions. The proposed concept is relevant to the PlacePulse Intelligence project because it emphasizes intelligent placement analysis and career development.

## 2.2 Full-Stack Analytical Framework for Higher Education Placement Intelligence (2025)

Devi, K. and Natarajan, S. (2025) presented a full-stack analytical framework for higher education placement intelligence. Their work focuses on connecting educational information with industry requirements through modern software technologies. The framework emphasizes the importance of collecting, processing, and analyzing student placement data to support institutional decision-making. It also highlights the role of integrated web applications in managing placement-related activities. This study provides a foundation for developing a centralized placement management platform that helps colleges monitor student performance and placement activities.

## 2.3 Automated Synthesis of Collision-Free Student Identifiers in College Enterprise Systems (2024)

Sengupta, D. and Roy, T. (2024) discussed the automated generation of unique student identifiers in college enterprise systems. The study focuses on avoiding duplicate identification numbers and maintaining consistency in student records. Unique identifiers are essential for managing student profiles, academic information, placement details, and administrative records. Automated identification improves data organization and reduces errors during record creation. This concept is useful for the PlacePulse Intelligence project because it supports reliable student registration and efficient management of student information.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 2.4 TypeScript Language Specification (2024)

Microsoft Corporation (2024) documented the TypeScript programming language, which extends JavaScript by introducing static typing and additional development features. TypeScript helps developers identify programming errors during development and improves code readability and maintainability. It supports structured application development through interfaces, classes, and type checking.

### 2.5 Tailwind CSS: Utility-First CSS Framework (2024)

Tailwind Labs (2024) introduced Tailwind CSS, a utility-first CSS framework used to design responsive and modern user interfaces. It provides predefined utility classes that allow developers to apply styling directly within HTML elements. The framework simplifies layout design, spacing, colors, typography, and responsive behavior. It also helps maintain consistent styling across different pages of a web application. This framework is useful for developing the PlacePulse Intelligence interface, including student dashboards, administrative panels, placement statistics, and company information pages.

### 2.6 Fault-Tolerant Student Database Management Console (2023)

Chandran, M. and Vijayaraghavan, P. (2023) focused on developing a fault-tolerant student database management console for efficient administrative operations. The study emphasizes reliable data storage, accurate record management, and simplified database operations. A well-organized database system helps administrators retrieve, update, and maintain student information without unnecessary duplication or data loss. This concept supports the development of a dependable student information management system.

### 2.7 SUMMARY OF LITERATURE SURVEY

The literature survey highlights the importance of artificial intelligence, data analytics, database management, and modern web technologies in developing an effective college placement management system. Previous studies and technical documentation provide useful concepts for student identification, database reliability, responsive interface design, career guidance, and placement forecasting. These concepts form the foundation for the proposed PlacePulse Intelligence system. By integrating student information management, company details, placement records, and analytical dashboards into a centralized platform, the proposed system aims to simplify placement activities, improve data accessibility, and support better decision-making for students and placement administrators.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 3

# SYSTEM ANALYSIS AND ARCHITECTURAL DESIGN

## 3.1 EXISTING SYSTEM AND OPERATIONAL BOTTLENECKS

In the existing practice, the placement cell and each department maintain their own spreadsheets. Company details, drive schedules, student lists, selection results and package details are entered by different people at different times. At the end of a season, the coordinators exchange files and one person combines them into the report required by management and by accreditation bodies. This practice has the following disadvantages.

**Siloed data:** Each department sees only its own students, so cross-department questions cannot be answered quickly.

**Duplicate and inconsistent entries:** The same company or student can be spelt in different ways in different files.

**Wrong counts:** A student who holds two offers is counted twice unless somebody removes the duplicate by hand.

**Manual computation:** Percentages, averages and medians are calculated by hand and are rarely re-checked.

## 3.2 PROPOSED SYSTEM AND ARCHITECTURAL ADVANTAGES

Place-Pulse Intelligence replaces the spreadsheets with one relational database and a web application that reads it. Every placement event is stored once, linked by keys to the student, the company and the drive. The backend computes all indicators with SQL queries at request time, so the figures shown on the dashboard are always consistent with the data. The React frontend presents six indicator cards; selecting a card opens a detailed analytics page with filters, charts and record tables. A student directory allows records to be searched, added and removed, and every change refreshes the dashboard immediately.

The advantages of the proposed system are listed below.

**Single source of truth:** All records are in one normalised database.

**Correct counting:** Placed students are counted as distinct students, and only accepted offers enter the package statistics.

**Real-time analytics:** Metrics are computed from live data, so there is no separate reporting step.

**Drill-down:** Every card leads to a page that lists the underlying companies, students or internships.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 3.3 THREE-TIER FULL-STACK ARCHITECTURE

The system follows a three-tier architecture, as shown in Figure 3.1. The presentation tier is a React single-page application. The application tier is an Express.js REST API organised into routes, controllers, services and middleware. The data tier is a relational database accessed through a dual-engine connection layer that supports SQLite and MySQL.

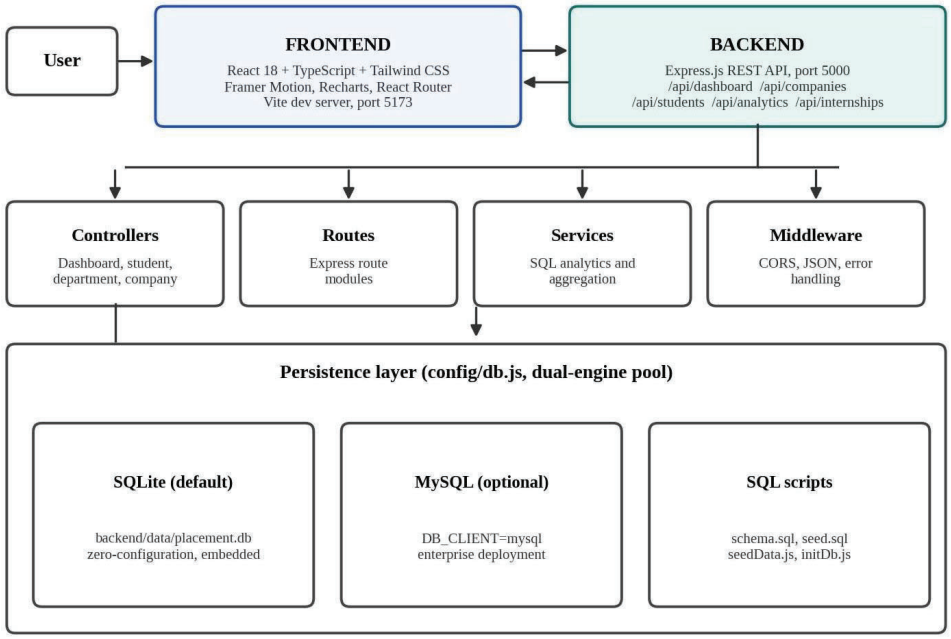



*Figure 3.1: Three-Tier Architecture of Place-Pulse Intelligence*

The layers are described below.

**Presentation tier (React 18, TypeScript, Tailwind CSS, Framer Motion):** The application is served by the Vite development server on port 5173. React Router handles navigation between the dashboard, the six analytics pages, the detail profiles and the student directory. Recharts draws the bar, line and distribution charts, and Framer Motion animates the cards. A typed API client in services/api.ts is the only place that calls the backend.

**Application tier (Node.js and Express.js):** The server listens on port 5000. Routes map URLs to controllers, controllers validate input and call services, and services contain the SQL analytics queries. Middleware provides cross-origin access, JSON parsing and centralised error handling.

**Data tier (SQLite and MySQL):** config/db.js exposes one query interface that hides the difference between the two engines. SQLite is the default and stores everything in backend/data/placement.db. Setting DB_CLIENT=mysql switches to a MySQL server for enterprise deployment.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 3.4 MATHEMATICAL MODELS FOR PLACEMENT ANALYTICS

All indicators on the dashboard are defined by simple formulas. Let S be the set of students, E the set of eligible students, and O the set of accepted offers, where each offer o has a student s(o) and a package p(o) in lakhs per annum (LPA).

#### 3.4.1 Placed students

A student is counted as placed if he or she holds at least one accepted offer. The number of placed students is the number of distinct students in O, not the number of offers. Placed = | { s(o) : o in O } |

#### 3.4.2 Worked example

Suppose a department has 40 eligible students and 30 accepted offers, but two students hold two offers each, so the offers belong to 28 distinct students. The placement percentage is 28 / 40 x 100 = 70 percent.

#### 3.4.3 Department-wise calculation example

Table 3.1 shows how the department-wise percentage and the campus percentage are obtained from sample counts. The campus figure is calculated from the totals and not by averaging the department percentages, because departments differ in size.

**Table 3.1: Example of department-wise placement calculation (sample counts)**

| Department | Eligible | Placed (distinct) | Placement % |
|---|---|---|---|
| CSE | 24 | 20 | 83.33 |
| IT | 20 | 15 | 75.00 |
| ECE | 20 | 13 | 65.00 |
| EEE | 18 | 11 | 61.11 |
| MECH | 16 | 9 | 56.25 |
| Campus total | 110 | 74 | 67.27 |

### 3.5 DATA FLOW DIAGRAMS (DFD LEVELS 0, 1 AND 2)

A Data Flow Diagram (DFD) is a visual representation of how data moves through a system, showing the flow of information between processes, data stores and external entities. It helps in understanding the functionality of a system and in identifying areas for improvement.

#### 3.5.1 Level 0 (Context Diagram)

At level 0 the whole platform is one process. The placement officer sends create, update, delete and query requests, and receives indicators and reports. Students, faculty and heads of department send search and filter requests and receive profiles and charts.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

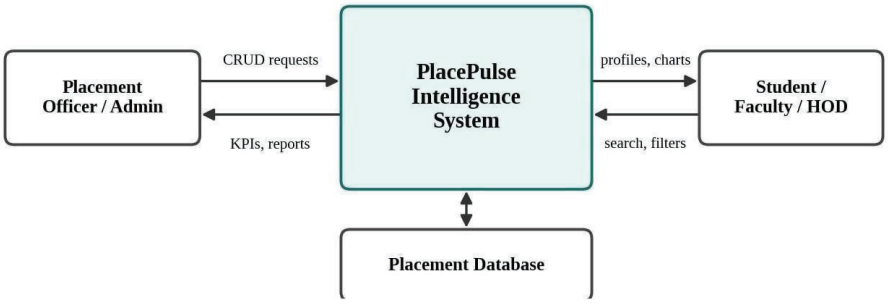



*Figure 3.2: DFD Level 0 - Context Diagram*

### 3.5.2 Level 2 (Analytics Engine)

The analytics engine of level 1 is refined into five steps: (3.1) read the filter parameters; (3.2) build the parameterised SQL statement; (3.3) execute it on the database; (3.4) postprocess the rows, for example by computing the median and percentages; and (3.5) pass the result to the response formatter. Keeping these steps separate is what allows the same service to be used by both the dashboard summary and the detailed analytics pages.

## 3.6 ENTITY-RELATIONSHIP MODELLING

The entity-relationship (ER) model identifies six entities. A department has many students. A student may receive many offers and may have many internships. A company conducts many drives, and every offer belongs to one drive. Figure 3.4 shows the entities, their key attributes and the one-to-many relationships.

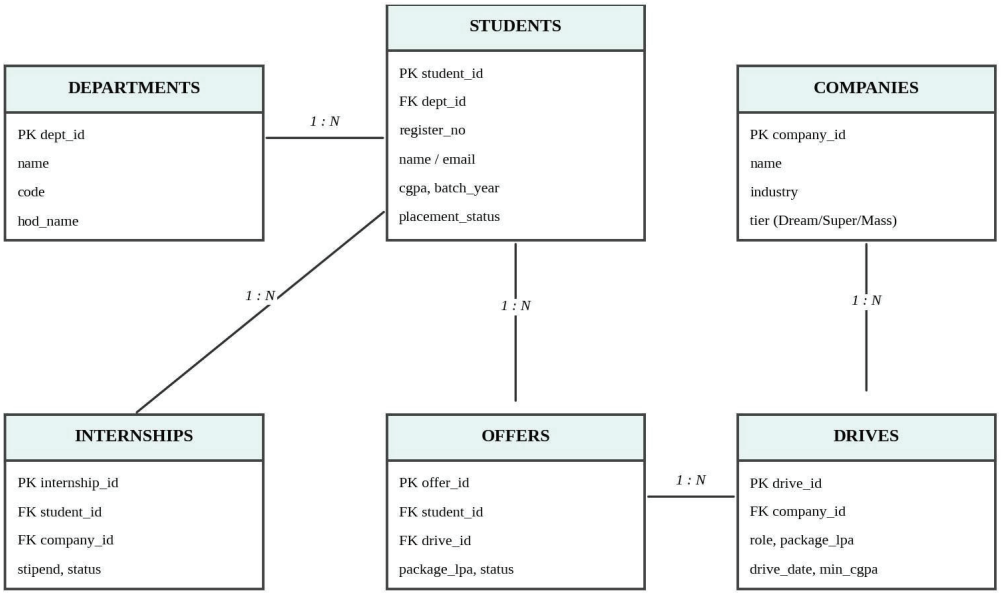



*Figure 3.4: Entity-Relationship Diagram*



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**Table 3.2: Relationships in the ER model**

| Relationship | Cardinality | Meaning |
|---|---|---|
| DEPARTMENTS - STUDENTS | 1 : N | One department has many students |
| COMPANIES - DRIVES | 1 : N | One company conducts many drives |
| STUDENTS - OFFERS | 1 : N | A student may hold many offers |
| DRIVES - OFFERS | 1 : N | A drive produces many offers |
| STUDENTS - INTERNSHIPS | 1 : N | A student may hold many internships |
| COMPANIES - INTERNSHIPS | 1 : N | A company may provide many internships |

### 3.7 UML SEQUENCE MODELLING

The functional requirements describe what the system must do. They were derived from the problems in Section 1.3 and from discussions about the reports that the placement cell prepares every year.

**Table 3.3: Functional requirements**

| ID | Requirement | Module |
|---|---|---|
| FR1 | The system shall show six live indicator cards on the home page | Dashboard |
| FR2 | The system shall list recruiters with search, department and package filters | Companies |
| FR3 | The system shall show a company profile with its drives and hired students | Companies |
| FR4 | The system shall count a placed student once even if several offers exist | Placed students |
| FR5 | The system shall calculate placement percentage overall, by department and by year | Placement percentage |
| FR6 | The system shall show the highest package with company, role and department | Highest package |
| FR7 | The system shall show the average and median of accepted offers | Average package |
| FR8 | The system shall separate unique internship students from total internship offers | Internships |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 4

# SYSTEM REQUIREMENTS AND SPECIFICATIONS

## 4.1 HARDWARE REQUIREMENTS

The system is light enough to run on an ordinary college computer. Table 4.1 lists the requirements.

**Table 4.1: Hardware requirements**

| Component | Server (backend and database) | Client device |
|---|---|---|
| Processor | Dual-core CPU at 2.0 GHz or better | 1 GHz or faster |
| RAM | 4 GB minimum, 8 GB recommended | 2 GB minimum |
| Storage | 500 MB free for Node.js packages and the database file | Not needed |
| Network | Internet for installation; local network for shared use | Any modern web browser |
| Display | Not required | 1366 x 768 or higher |

## 4.2 SOFTWARE REQUIREMENTS AND TECHNICAL STACK

**Table 4.2: Software requirements**

| Item | Requirement |
|---|---|
| Operating system | Windows 10 or later, Linux or macOS |
| Runtime | Node.js v18 or higher (v20 or v24 recommended) and npm v9 or higher |
| Backend | Express.js, CORS, dotenv, SQLite driver; optional MySQL driver |
| Database | SQLite 3 (embedded, default) or MySQL 8 (optional) |
| Frontend | React 18, TypeScript, Vite, Tailwind CSS, Framer Motion, React Router, Recharts |
| Browser | Google Chrome (auto-launched), Edge, Firefox or Safari |
| Tools | Git, Visual Studio Code and a terminal |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 4.2.1. ROLE OF EACH TECHNOLOGY

**Table 4.3: Technology stack**

| Technology | Role in the project |
|---|---|
| React 18 SPA | Component-based user interface with client-side routing |
| TypeScript | Static typing of API responses and component props |
| Tailwind CSS | Utility-first styling and responsive layout |
| Framer Motion | Animated cards, modals and page transitions |
| Recharts | Bar, line and distribution charts |
| Vite | Fast development server, bundling and /api proxy |
| Node.js and Express.js | Asynchronous REST API server |
| SQLite 3 | ACID-compliant embedded relational database |
| MySQL (optional) | Server database for enterprise deployment |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 5

# SYSTEM DESIGN AND IMPLEMENTATION

## 5.1 INTRODUCTION TO SYSTEM DESIGN AND IMPLEMENTATION

Place-Pulse Intelligence is implemented as a full-stack JavaScript application. The backend in the backend folder contains the Express.js server, the SQL scripts and the database layer. The frontend in the project folder contains the React and TypeScript application. A launcher script, dev.js, in the root folder starts both parts.

## 5.2 RELATIONAL DATABASE SCHEMA AND DATA DICTIONARIES

The schema is defined in backend/sql/schema.sql and is created automatically when the server starts. The tables use primary keys, foreign keys and constraints to keep the data consistent.

The data dictionaries of the main tables are given below.

**Table 5.1: Data dictionary of the DEPARTMENTS table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| dept_id | INTEGER | PRIMARY KEY | Unique department identifier |
| Code | VARCHAR(10) | UNIQUE, NOT NULL | CSE, IT, ECE, EEE, MECH, CIVIL |
| Name | VARCHAR(100) | NOT NULL | Full department name |
| hod_name | VARCHAR(100) | NULL | Head of the department |

**Table 5.2: Data dictionary of the STUDENTS table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| student_id | INTEGER | PRIMARY KEY | Unique student identifier |
| register_no | VARCHAR(20) | UNIQUE, NOT NULL | University register number |
| Name | VARCHAR(100) | NOT NULL | Student name |
| Email | VARCHAR(120) | UNIQUE | Student e-mail address |
| dept_id | INTEGER | FOREIGN KEY | Reference to DEPARTMENTS |
| Cgpa | DECIMAL(4,2) | CHECK 0 to 10 | Cumulative grade point average |
| batch_year | INTEGER | NOT NULL | Year of graduation |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**Table 5.3: Data dictionary of the COMPANIES table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| company_id | INTEGER | PRIMARY KEY | Unique company identifier |
| Name | VARCHAR(120) | UNIQUE, NOT NULL | Recruiter name |
| Industry | VARCHAR(80) | NOT NULL | Industry segment |
| Tier | VARCHAR(20) | CHECK | Dream, Super-Dream or Mass |
| Website | VARCHAR(200) | NULL | Company website |

**Table 5.4: Data dictionary of the DRIVES table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| drive_id | INTEGER | PRIMARY KEY | Unique drive identifier |
| company_id | INTEGER | FOREIGN KEY | Reference to COMPANIES |
| package_lpa | DECIMAL(5,2) | NOT NULL | Advertised package in LPA |
| drive_date | DATE | NOT NULL | Date of the drive |
| min_cgpa | DECIMAL(4,2) | DEFAULT 0 | Eligibility cut-off |

**Table 5.5: Data dictionary of the OFFERS table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| offer_id | INTEGER | PRIMARY KEY | Unique offer identifier |
| student_id | INTEGER | FOREIGN KEY | Reference to STUDENTS |
| drive_id | INTEGER | FOREIGN KEY | Reference to DRIVES |
| package_lpa | DECIMAL(5,2) | NOT NULL | Offered package in LPA |
| status | VARCHAR(20) | CHECK | Accepted, Declined or Pending |

**Table 5.6: Data dictionary of the INTERNSHIPS table**

| Column | Type | Constraint | Description |
|---|---|---|---|
| internship_id | INTEGER | PRIMARY KEY | Unique internship identifier |
| student_id | INTEGER | FOREIGN KEY | Reference to STUDENTS |
| company_id | INTEGER | FOREIGN KEY | Reference to COMPANIES |
| stipend | DECIMAL(8,2) | DEFAULT 0 | Monthly stipend |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 5.3 DATABASE NORMALIZATION ANALYSIS

The schema has been checked against the first three normal forms.

**First normal form (1NF):** Every column holds one atomic value. For example, a student's offers are not stored in one cell, but as separate rows of the OFFERS table.

**Second normal form (2NF):** Every table has a single-column primary key, so no non-key attribute depends on part of a key.

**Third normal form (3NF):** Non-key attributes depend only on the key. Company details are stored once in COMPANIES and department details once in DEPARTMENTS. OFFERS and DRIVES refer to them through foreign keys instead of repeating the name and the industry.

**Table 5.7: Normal form check of the tables**

| Table | Key | 1NF | 2NF | 3NF |
|---|---|---|---|---|
| DEPARTMENTS | dept_id | Yes | Yes | Yes |
| STUDENTS | student_id | Yes | Yes | Yes |
| COMPANIES | company_id | Yes | Yes | Yes |
| DRIVES | drive_id | Yes | Yes | Yes |
| OFFERS | offer_id | Yes | Yes | Yes |
| INTERNSHIPS | internship_id | Yes | Yes | Yes |

### 5.4 FUNCTIONAL SUBSYSTEMS

#### 5.4.1 Database initialisation and migration

When the server starts, `config/db.js` opens the SQLite file, runs `schema.sql` if the tables are missing, and loads the seed data when the database is empty. The same module exposes a common query interface for SQLite and MySQL, so the services do not depend on the engine. The command `npm run db:seed` runs `scripts/initDb.js` to reset the database to the sample data.

#### 5.4.2 Student directory and database management

The directory is opened from the "Student Database" button. It offers live search by name, register number or e-mail, filters by department and by status (Placed, Eligible, Not Placed), and a modal form for adding a student. Records can be updated or removed. After each write the dashboard summary is requested again, so every card shows the new figures at once.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 5.5 ALGORITHMS AND PSEUDOCODE Algorithm 1: Compute dashboard summary

INPUT : database connection db

OUTPUT: summary object

```
recruiters  <- COUNT(DISTINCT company_id) from DRIVES joined with OFFERS
placed      <- COUNT(DISTINCT student_id) from OFFERS where status = 'Accepted'
eligible    <- COUNT(*) from STUDENTS where is_eligible = 1
percentage  <- round(placed / eligible * 100, 2)
highest     <- MAX(package_lpa) from OFFERS where status = 'Accepted'
average     <- AVG(package_lpa) from OFFERS where status = 'Accepted'
interns     <- COUNT(DISTINCT student_id) and COUNT(*) from INTERNSHIPS
return {recruiters, placed, percentage, highest, average, interns}
```

### Algorithm 2: Median package

INPUT : accepted package values p[1..n]

OUTPUT: median

```
sort p in ascending order
if n = 0: return 0
if n is odd : return p[(n + 1) / 2]
else        : return (p[n / 2] + p[n / 2 + 1]) / 2
```

### 5.6 RESTFUL API SPECIFICATION

The backend exposes the endpoints in Table 5.8. All responses are JSON, and errors use a common format with an HTTP status code and an error message.

**Table 5.8: REST API endpoints**

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/health | Service health check |
| GET | /api/dashboard/summary | Live statistics for the six cards |
| GET | /api/departments | List all departments |
| GET | /api/departments/analytics | Department-wise rate and LPA statistics |
| GET | /api/companies | List companies with search and filters |
| GET | /api/companies/top-recruiters | Top 10 recruiting companies |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### 5.7 SOURCE CODE IMPLEMENTATION

The key parts of the implementation are shown below. The complete reference code is given in the Appendices.

**Step 1: Dual-engine database connection**

```
// backend/src/config/db.js (simplified) const client =
process.env.DB_CLIENT || 'sqlite';

async function createPool() { if (client
=== 'mysql') { const mysql = require('mysql2/promise');
const pool = mysql.createPool({
  host: process.env.DB_HOST, port: process.env.DB_PORT, user:
process.env.DB_USER, password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
 });

 const { DatabaseSync } = require('node:sqlite'); const db = new
DatabaseSync('data/placement.db');
```

**Step 2: Placed students counted as distinct students**

```
// backend/src/services/analyticsService.js (simplified) async
function placementPercentage(db) { const [row] =
await db.all(`
SELECT
(SELECT COUNT(*) FROM students WHERE is_eligible = 1) AS eligible,
(SELECT COUNT(DISTINCT student_id) FROM offers WHERE
{ eligible: row.eligible, placed: row.placed, percentage:
Math.round(pct * 100) / 100 }; }
```

**Step 3: Average and median package**

```
async function averagePackage(db) { const rows
= await db.all(
"SELECT package_lpa FROM offers WHERE status = 'Accepted' ORDER BY package_lpa");
const v = rows.map(r => Number(r.package_lpa)); const
n = v.length;
if (n === 0) return { average: 0, median: 0, offers: 0 }; const average = v.reduce((a,
b) => a + b, 0) /
const median = n % 2 ? v[(n - 1) / 2] : (v[n / 2 - 1] + v[n / 2]) / 2; return { average: +average.toFixed(2),
median: +median.toFixed(2), offers: n }; }
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# APPENDICES

## A.1 PROJECT STRUCTURE

```
mp2/
|-- backend/
|   |-- data/placement.db            # embedded SQLite database (auto- created)
|   |-- sql/schema.sql, seed.sql     # table definitions and sample data
|   |-- src/
|   |   |-- config/db.js, seedData.js
|   |   |-- controllers/             # dashboard, student, department,  company
|   |   |-- routes/                  # Express API routes
|   |   |-- services/                # SQL analytics queries and aggregation
|   |   |-- middleware/              # error handling
|   |   |-- app.js                   # Express app setup and CORS
|   |   `-- server.js                # HTTP server entry point
|   |-- scripts/initDb.js, test_analytics.js
|   |-- .env
|   `-- package.json
|-- project/                         # frontend (Vite + React + TypeScript)
|   |-- src/components, pages, data, services/api.ts
|   |-- src/App.tsx, main.tsx
|   |-- vite.config.ts
|   `-- package.json
|-- dev.js                           # unified launcher
|-- package.json
`-- README.md
```

The files that hold the main logic of each side are listed below. The code is a reference listing of the main modules; the remaining files are in the project repository.

## A.2 DATABASE SCRIPT File: backend/sql/schema.sql

```
CREATE TABLE IF NOT EXISTS departments (   dept_id
                        INTEGER      PRIMARY      KEY
AUTOINCREMENT,  code      VARCHAR(10)  NOT
NULL UNIQUE,     name        VARCHAR(100) NOT
NULL,   hod_name  VARCHAR(100)
);
CREATE TABLE IF NOT EXISTS students (   student_id
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
INTEGER PRIMARY KEY
        AUTOINCREMENT, register_no
VARCHAR(20) NOT NULL UNIQUE, name
VARCHAR(100) NOT NULL, email
VARCHAR(120) UNIQUE, dept_id INTEGER
NOT NULL,
 cgpa DECIMAL(4,2) CHECK (cgpa >= 0 AND cgpa <= 10),
batch_year INTEGER NOT NULL, is_eligible BOOLEAN
DEFAULT 1,
 FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
CREATE TABLE IF NOT EXISTS drives ( drive_id INTEGER
PRIMARY KEY AUTOINCREMENT,
 company_id INTEGER NOT NULL, role
VARCHAR(100) NOT NULL, package_lpa
DECIMAL(5,2) NOT NULL, drive_date DATE NOT
NULL, min_cgpa DECIMAL(4,2) DEFAULT 0,
 FOREIGN KEY (company_id) REFERENCES companies(company_id)
);
CREATE TABLE IF NOT EXISTS internships ( internship_id INTEGER
PRIMARY KEY AUTOINCREMENT,
 student_id INTEGER NOT NULL, company_id INTEGER
NOT NULL, stipend DECIMAL(8,2)
DEFAULT 0, status VARCHAR(20) CHECK
(status IN


('Offered','Started','Completed')),
 FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE
CASCADE,
 FOREIGN KEY (company_id) REFERENCES companies(company_id)
);
CREATE INDEX IF NOT EXISTS idx_students_dept ON students(dept_id);
```

**A.3 BACKEND SOURCE CODE File: backend/src/server.js**

```
require('dotenv').config(); const app =

require('./app');
const { initDatabase } = require('./config/db'); const
PORT = process.env.PORT || 5000;
(async () => { try {
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
  await initDatabase();          // create schema and seed if empty    app.listen(PORT,
() =>
    console.log('Placement API running on http://localhost:' + PORT));
 } catch (err) {
  console.error('Failed       to       start       server:',       err);
process.exit(1);
 }
})();
```

**File: backend/src/config/db.js** const

```
fs = require('fs'); const path =
require('path');


const CLIENT = process.env.DB_CLIENT || 'sqlite'; let
handle = null;



async function connect() {  if (handle) return handle;
if (CLIENT === 'mysql') {     const mysql =
require('mysql2/promise');      const pool =
mysql.createPool({       host: process.env.DB_HOST ||
'localhost',       port: Number(process.env.DB_PORT ||
3306),       user: process.env.DB_USER || 'root',       password:
          process.env.DB_PASSWORD          ||          '',       database:
          process.env.DB_NAME         ||
'placement_analytics',       waitForConnections: true, connectionLimit: 10,
  });     handle =
{       all: async (sql, p = []) => (await pool.query(sql
p))[0],       run: async (sql, p = []) => {         const [r] =
await
pool.query(sql, p);
       return { lastId: r.insertId, changes: r.affectedRows };
     },
   };
 } else {
  const { DatabaseSync } = require('node:sqlite');     const file =
path.join(__dirname, '../../data/placement.db');
fs.mkdirSync(path.dirname(file), { recursive: true });      const db = new
DatabaseSync(file);      db.exec('PRAGMA
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
foreign_keys = ON');
   handle = {              all: async (sql, p = []) =>
db.prepare(sql).all(...p),       run: async (sql, p = []) =>
{        const r =
db.prepare(sql).run(...p);
       return { lastId: Number(r.lastInsertRowid), changes: r.changes };
     },
     exec: async (sql) => db.exec(sql),
   };
 }
```

**File: backend/src/services/analyticsService.js** const
{ connect } = require('../config/db');

```
async function placementPercentage() {
const db = await connect();  const [r] = await db.all(`
   SELECT
     (SELECT COUNT(*) FROM students WHERE is_eligible = 1) AS eligible,
     (SELECT COUNT(DISTINCT student_id) FROM offers
       WHERE status = 'Accepted') AS placed`);  const pct = r.eligible ? (r.placed /
r.eligible) * 100 : 0;  return { eligible: r.eligible, placed:
r.placed,          percentage: Math.round(pct * 100) / 100 }; }
async function departmentComparison()
{  const db = await connect();  return
db.all(`
   SELECT d.code AS department,
        COUNT(DISTINCT s.student_id) AS eligible,
        COUNT(DISTINCT o.student_id) AS placed,
        ROUND(COUNT(DISTINCT o.student_id) * 100.0 /
            COUNT(DISTINCT s.student_id), 2) AS percentage
   FROM departments d
   JOIN students s ON s.dept_id = d.dept_id AND s.is_eligible = 1
   LEFT JOIN offers o ON o.student_id = s.student_id AND o.status =
'Accepted'
   GROUP BY d.dept_id ORDER BY percentage DESC`);
}
async function yearlyTrends() {   const db
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
= await connect();  async function
topPackageOffers(limit = 15) {   const db =
await connect();

 return db.all(`
   SELECT s.name AS student, dp.code AS department, c.name AS company,
        d.role, o.package_l
   FROM offers o
   JOIN students s ON s.student_id = o.student_id
   JOIN departments dp ON dp.dept_id = s.dept_id
   JOIN drives d ON d.drive_id = o.drive_id
   JOIN companies c ON c.company_id = d.company_id
   WHERE o.status = 'Accepted'
   ORDER BY o.package_lpa DESC LIMIT ?`, [limit]);
}
async function averagePackage() {   const db
= await connect();   const rows = await db.all(
   "SELECT  package_lpa  FROM  offers  WHERE  status  =  'Accepted'  ORDER  BY
package_lpa");
  const v = rows.map((r) => Number(r.package_lpa)); const
n = v.length;
  if (!n) return { average: 0, median: 0, offers: 0 };  const average =
v.reduce((a, b) => a + b, 0) /
  const median = n % 2 ? v[(n - 1) / 2] : (v[n / 2 - 1] + v[n / 2]) / 2;    return { average:
+average.toFixed(2), median: +median.toFixed(2), offers: n };
}
}
async function departmentAveragePackages()
{   const db = await connect();   return db.all(`
   SELECT dp.code AS department, ROUND(AVG(o.package_lpa), 2) AS average
   FROM offers o
   JOIN students s ON s.student_id = o.student_id
   JOIN departments dp ON dp.dept_id = s.dept_id
   WHERE o.status = 'Accepted'
   GROUP BY dp.dept_id ORDER BY average DESC`);
}
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
  } };
exports.remove = async (req, res, next) => {
  try {
    const db = await connect();
    const r = await db.run('DELETE FROM students WHERE student_id = ?',
[req.params.id]);
    if (!r.changes) return res.status(404).json({ error: 'Student not  found.'
});
    res.json({ deleted: true });
  } catch (e) { next(e); }
};
```

### A.4 FRONTEND SOURCE CODE

**File: project/src/services/api.ts**

```
export interface          DashboardSummary   {
totalRecruiters: number;   placedStudents: number;
                    eligibleStudents:          number;
placementPercentage:             number;   highestPackage:
number;   averagePackage: number;
           internshipOffers:          number;
internshipStudents: number;
}
export interface Student {
  student_id: number; register_no: string; name: string; email: string;
department: string; cgpa: number; batch_year: number;   status: 'Placed' |
'Eligible' | 'Not Placed';
}
async function request<T>(path: string, init?: RequestInit): Promise<T> {   const
res = await fetch('/api' + path, {
    headers: { 'Content-Type': 'application/json' }, ...init,

  });
export const getSummary  = () =>  request<DashboardSummary>('/dashboard/summary');
export const getStudents = (q = '') => request<Student[]>('/students' +
q); export const addStudent  = (s: Partial<Student>) =>
request('/students', { method: 'POST', body: JSON.stringify(s) });
export const    deleteStudent  =          (id:       number)          =>
request('/students/' + id, { method: 'DELETE' });
```


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**File: project/src/pages/Dashboard.tsx** import { useEffect, useState } from 'react'; import { Link } from 'react-router-dom'; import { motion } from 'framer-motion'; import { getSummary, DashboardSummary } from '../services/api';

```
export default function Dashboard() {
  const [s, setS] = useState<DashboardSummary | null>(null);  const [error, setError] = useState('');

  useEffect(() => {
    getSummary().then(setS).catch((e) => setError(e.message));
  }, []);

  if (error) return <p className="p-6 text-red-600" role="alert">{error}</p>;    if (!s) return <p className="p-6 text-slate-500">Loading placement  data...</p>;

  const cards = [
    { to: '/analytics/companies',          label: 'Total Recruiters',     value: s.totalRecruiters },
    { to: '/analytics/placed-students',    label: 'Students Placed',      value: s.placedStudents

  { to: '/analytics/placement-percentage', label: 'Placement Percentage',  value: s.placementPercentage + '%'
    { to: '/analytics/highest-package',    label: 'Highest Package',      value: s.highestPackage + ' LPA'
    { to: '/analytics/average-package',    label: 'Average Package',      value: s.averagePackage + ' LPA' },
    { to: '/analytics/internships',        label: 'Internship Offers',    value: s.internshipOffers
  ];
        ))}
      </div>
    </section>
  );
}
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**File: backend/sql/seed.sql (excerpt)**

```
INSERT INTO departments (code, name, hod_name) VALUES
 ('CSE',  'Computer Science and Engineering',          'Dr. G. Fathima'),
 ('IT',   'Information Technology',                    'HOD - IT'),
 ('ECE',  'Electronics and Communication Engineering', 'HOD - ECE'),
 ('EEE',  'Electrical and Electronics Engineering',    'HOD - EEE'),
 ('MECH', 'Mechanical Engineering',                    'HOD - MECH'),  ('CIVIL','Civil
Engineering',                         'HOD - CIVIL');
INSERT INTO companies (name, industry, tier) VALUES
 ('Sample Software Ltd',  'IT Services',  'Mass'),
 ('Sample Product Corp',  'Product',      'Super-Dream'),
-- students, offers and internships are loaded by seedData.js
-- (120 students, 87 placement and internship records)
```

**File: backend/src/controllers/companyController.js**

```
const service = require('../services/companyService');
exports.list = async (req, res, next) => {
  try   {   res.json(await   service.listCompanies(req.query));    }
catch (e) { next(e); }
};
exports.detail = async (req, res, next) => {   try
{
   const company = await service.getCompany(req.params.id);      if (!company)
return res.status(404).json({ error: 'Company not
found.'                              });
res.json(company);
  } catch (e) { next(e); }
}; exports.top = async (req, res, next) => {  try { res.json(await
service.topRecruiters(10)); }
  catch (e) { next(e); }
};
```

**File: backend/src/routes/analyticsRoutes.js**

```
const  router  =          require('express').Router();     const    a        =
require('../services/analyticsService'); const wrap = (fn) => async (req,
res, next) => {  try { res.json(await fn(req)); } catch (e) { next(e); }
};


router.get('/placement-percentage',        wrap(() =>  a.placementPercentage()));
router.get('/department-comparison',       wrap(() =>  a.departmentComparison()));
router.get('/yearly-trends',               wrap(() => a.yearlyTrends())); router.get('/highest-
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
package',            wrap(() => a.highestPackage()));
router.get('/top-package-offers',          wrap((r) =>
a.topPackageOffers(Number(r.query.limit) || 15)));
router.get('/average-package',             wrap(() => a.averagePackage()));
router.get('/company-average-packages',    wrap(() =>
a.companyAveragePackages())); router.get('/department-average-packages',
wrap(() => a.departmentAveragePackages()));


module.exports = router;
```

**File: backend/src/services/internshipService.js** const
{ connect } = require('../config/db');

```
async function summary() { const
db = await connect(); const [r] =
await db.all(`
  SELECT COUNT(*) AS totalOffers,
      COUNT(DISTINCT student_id) AS uniqueStudents,
      MIN(stipend) AS minStipend, MAX(stipend) AS maxStipend,
      ROUND(AVG(stipend), 2) AS avgStipend,
      SUM(CASE WHEN status = 'Started'  THEN 1 ELSE 0 END) AS started,
SUM(CASE WHEN status = 'Completed' THEN 1 ELSE 0 END) AS  completed
  FROM internships`);    return r; }
async function students() {  const db
=     await     connect();
return db.all(`
  SELECT s.student_id, s.name, dp.code AS department,
      COUNT(i.internship_id) AS offers, MAX(i.stipend) AS topStipend
  FROM internships i
  JOIN students s ON s.student_id = i.student_id
  JOIN departments dp ON dp.dept_id = s.dept_id
  GROUP BY s.student_id ORDER BY offers DESC, s.name`);
}  async function companies() {  const
db
=     await     connect();
return db.all(`
  SELECT c.name, c.industry, COUNT(*) AS offers, ROUND(AVG(i.stipend), 2)
AS avgStipend
  FROM internships i JOIN companies c ON c.company_id = i.company_id
  GROUP BY c.company_id ORDER BY offers DESC`);
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
} async function charts() { const db =
await connect(); const byDepartment =
await
db.all(`
   SELECT dp.code AS department, COUNT(*) AS offers
   FROM internships i
   JOIN students s ON s.student_id = i.student_id
   JOIN departments dp ON dp.dept_id = s.dept_id
GROUP BY dp.dept_id`);   const byStatus = await
db.all(
   'SELECT status, COUNT(*) AS count FROM internships GROUP BY status');   return
{ byDepartment, byStatus };
}
module.exports = { summary, students, companies, charts };
```

**File: package.json (root)**

```
{
 "name": "placement-analytics",
 "private": true,
 "scripts": {
   "dev": "node dev.js",
   "chrome": "node dev.js --chrome-only",
   "test:backend": "npm --prefix backend run test",
   "db:seed": "npm --prefix backend run db:seed"
 }
}
```

**File: project/src/components/Navbar.tsx** import
{ Link, NavLink } from 'react-router-dom';

```
const links = [
 ['/', 'Dashboard'],
 ['/analytics/companies', 'Recruiters'],
 ['/analytics/placed-students', 'Placed'],
 ['/analytics/placement-percentage', 'Placement %'],
 ['/analytics/highest-package', 'Highest'],
 ['/analytics/average-package', 'Average'],
 ['/analytics/internships', 'Internships'],
];
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
export default function Navbar() {
return (
  <header className="bg-white border-b border-slate-200">
    <div className="max-w-6xl mx-auto px-4 py-3 flex items-center justify- between">
      <Link to="/" className="text-xl font-extrabold text-teal-
700">PlacePulse</Link>
      <nav className="hidden md:flex gap-4 text-sm font-semibold">
        {links.map(([to, label]) => (
          <NavLink key={to} to={to} end={to === '/'}              className={({ isActive
}) => isActive ? 'text-teal-700' :
'text-slate-600'}>
            {label}
          </NavLink>
        ))}
      </Link>
    </div>
```

**File: project/src/pages/StudentDatabase.tsx**

```
import { useCallback, useEffect, useState } from 'react'; import { getStudents,
deleteStudent, Student } from '../services/api'; import AddStudentModal from
'../components/AddStudentModal';

const DEPARTMENTS = ['', 'CSE', 'IT', 'ECE', 'EEE', 'MECH', 'CIVIL']; const
STATUSES = ['', 'Placed', 'Eligible', 'Not Placed'];

export default function StudentDatabase() {  const [rows,
setRows] = useState<Student[]>([]);  const [search,
setSearch] = useState('');  const [department, setDepartment]
= useState('');  const [status, setStatus] = useState('');  const
[open, setOpen] = useState(false);  const [error, setError] =
useState('');

 const load = useCallback(async () => {
   try {
     const q = new URLSearchParams({ search, department,  status
}).toString();
     setRows(await       getStudents('?'       +       q));
setError('');
   } catch (e: any) { setError(e.message); }
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
  }, [search, department, status]);

  useEffect(() => { const t = setTimeout(load, 200); return () =>  clearTimeout(t);
}, [load]);

  async function remove(id: number) {
    if (!window.confirm('Delete this student record?'))
return;     await deleteStudent(id);     load();  }     return (
   <section className="max-w-6xl mx-auto p-6 space-y-4">

     <div className="flex flex-wrap gap-3 items-center">
       <input value={search} onChange={(e) => setSearch(e.target.value)}
placeholder="Search name, register number or e-mail"
className="border rounded-lg px-3 py-2 flex-1 min-w-[220px]" />
       <select value={department} onChange={(e) =>  setDepartment(e.target.value)}
         className="border rounded-lg px-3 py-2">
         {DEPARTMENTS.map((d)     =>     <option     key={d}     value={d}>{d     ||     'All
departments'}</option>)}
       </select>
       <select        value={status}        onChange={(e)        =>        setStatus(e.target.value)}
className="border rounded-lg px-3 py-2">
         {STATUSES.map((s) => <option key={s} value={s}>{s || 'All  statuses'}</option>)}
       </select>
       <button onClick={() => setOpen(true)}
         className="px-4 py-2 rounded-lg bg-teal-600 text-white font- semibold">Add
student</button>
       </tbody>
     </table>
     {open && <AddStudentModal onClose={() => setOpen(false)}  onSaved={load}
/>}
   </section>
  );
}
```

**File: project/src/components/AddStudentModal.tsx**

```
import { useState } from 'react'; import { motion } from 'framer-motion';
import  {  addStudent  }  from  '../services/api';  export  default  function
AddStudentModal({ onClose, onSaved }:
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
 { onClose: () => void; onSaved: () => void }) {
 const [f, setF] = useState({ register_no: '', name: '', email: '',                    dept_id:
1, cgpa: 7.5, batch_year: 2027 });  const
[error, setError] = useState('');


 async  function  save()  {
try {
    await addStudent(f as any);       onSaved();
onClose();
   } catch (e: any) { setError(e.message); }
 }
return (
   <div className="fixed inset-0 bg-black/40 flex items-center justify- center
p-4">
    <motion.div initial={{ scale: 0.95, opacity: 0 }} animate={{ scale: 1, opacity: 1 }}
className="bg-white rounded-2xl p-6 w-full max-w-md space-y-3">
      <h2 className="text-lg font-extrabold">Add student</h2>
      {(['register_no', 'name', 'email'] as const).map((k) => (
        <input key={k} placeholder={k.replace('_', ' ')} value={f[k]}            onChange={(e)
=> setF({ ...f, [k]: e.target.value })}           className="w-full border rounded-lg px-3 py2"
/>
      ))}
      {error && <p className="text-red-600 text-sm"  role="alert">{error}</p>}
      <div className="flex justify-end gap-2">
        <button  onClick={onClose}  className="px-3  py-2  rounded-lg  bg-  slate-
100">Cancel</button>
        <button  onClick={save}  className="px-3  py-2  rounded-lg  bg-teal-600
textwhite">Save</button>
      </div>
    </motion.div>
   </div>
 );
}
```

**File: project/src/pages/PlacementPercentage.tsx** import

```
{ useEffect, useState } from 'react';
import  {  Bar,  BarChart,  CartesianGrid,  Line,  LineChart,  ResponsiveContainer,
Tooltip, XAxis, YAxis } from 'recharts';

interface Dept { department: string; eligible: number; placed: number;  percentage:
number }
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
interface Year { year: number; percentage: number }
interface Overall { eligible: number; placed: number; percentage: number }

export default function PlacementPercentage() {  const [overall,
setOverall] = useState<Overall | null>(null);  const [depts,
setDepts] = useState<Dept[]>([]);  const [years,
setYears] = useState<Year[]>([]);

 useEffect(() => {
   const get = (p: string) => fetch('/api/analytics' + p).then((r) => r.json());
Promise.all([get('/placement-percentage'), get('/department- comparison'),
            get('/yearly-trends')])
    .then(([o, d, y]) => { setOverall(o); setDepts(d); setYears(y); });
 }, []);

 if (!overall) return <p className="p-6 text-slate-500">Loading...</p>;
  return (
   <section className="max-w-6xl mx-auto p-6 space-y-6">
    <h1 className="text-2xl font-extrabold">Placement Percentage</h1>
    <div className="grid sm:grid-cols-3 gap-4">
     <Stat label="Eligible students" value={overall.eligible} />
     <Stat label="Placed students" value={overall.placed} />
     <Stat label="Placement rate" value={overall.percentage + '%'} />
    </div>
    <div className="bg-white rounded-2xl border p-4 h-72">
     <h2 className="font-bold mb-2">Department-wise placement %</h2>
     <ResponsiveContainer width="100%" height="100%">
      <BarChart data={depts}>
       <CartesianGrid strokeDasharray="3 3" />
       <XAxis dataKey="department" /><YAxis domain={[0, 100]}
/><Tooltip />
       <Bar dataKey="percentage" fill="#0f766e" radius={[6, 6, 0,
0]} />
      </BarChart>
     </ResponsiveContainer>
    </div>
    <div className="bg-white rounded-2xl border p-4 h-72">
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
      <h2 className="font-bold mb-2">Year-wise trend</h2>        <ResponsiveContainer
width="100%" height="100%">
        <LineChart data={years}>
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey="year" /><YAxis domain={[0, 100]} /><Tooltip />
<Line type="monotone" dataKey="percentage" stroke="#1d4ed8"
strokeWidth={3} />
        </LineChart>
      </ResponsiveContainer>
    </div>
  </section>
 ); }
function Stat({ label, value }: { label: string; value: string | number }) {  return
(
  <div className="bg-white rounded-2xl border p-5">
    <p className="text-sm text-slate-500">{label}</p>
    <p    className="text-3xl    font-extrabold    text-teal-700">{value}</p>
</div>
 );
```

**File: project/src/pages/Companies.tsx**

```
import { useEffect, useState } from 'react'; import { Link } from
'react-router-dom';
interface Company { company_id: number; name: string; industry: string;                tier:
string; drives: number; maxPackage: number }
export  default  function  Companies()  {  const  [rows,
setRows]  =  useState<Company[]>([]);  const  [search,
setSearch] = useState('');  const [department, setDepartment]
=  useState('');  const  [minPackage,  setMinPackage]  =
useState(0);
 useEffect(() => {    const q = new URLSearchParams({ search, department,
minPackage:
String(minPackage) });    const t = setTimeout(() =>
{
    fetch('/api/companies?' + q).then((r) => r.json()).then(setRows);
  }, 200);
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
    return () => clearTimeout(t);       },
[search, department, minPackage]);
  const byIndustry = rows.reduce<Record<string, number>>((acc, c) => {     acc[c.industry]
= (acc[c.industry] || 0) + 1; return acc;
  },          {});
return (
   <section className="max-w-6xl mx-auto p-6 space-y-4">
     <h1 className="text-2xl font-extrabold">Total Recruiters
({rows.length})</h1>
     <div className="flex flex-wrap gap-3 items-center">
      <input       value={search}       onChange={(e)       =>       setSearch(e.target.value)}
placeholder="Search company" className="border rounded-lg px-3 py- 2" />


 <select value={department} onChange={(e) => setDepartment(e.target.value)}
        className="border rounded-lg px-3 py-2">          <option
value="">All departments</option>
        {['CSE',    'IT',    'ECE',    'EEE',    'MECH',    'CIVIL'].map((d)    =>    <option
key={d}>{d}</option>)}
       </select>
       <label className="text-sm">Min package: {minPackage} LPA
        <input type="range" min={0} max={30} value={minPackage}          onChange={(e)
=> setMinPackage(Number(e.target.value))}
className="ml-2" />
       </label>
     </div>
     <p className="text-sm text-slate-500">
      {Object.entries(byIndustry).map(([k, v]) => k + ': ' + v).join('  |
')}
     </p>
     <div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-3">
      {rows.map((c) => (
       <Link       key={c.company_id}       to={'/companies/'       +       c.company_id}
className="bg-white rounded-2xl border p-5 hover:shadow-md">
         <p className="font-bold">{c.name}</p>
         <p className="text-sm text-slate-500">{c.industry} -
{c.tier}</p>
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
            <p className="mt-2 text-sm">{c.drives} drive(s), up to
{c.maxPackage} LPA</p>
          </Link>
        ))}
      </div>
    </section>
  );
}
```

**File: project/src/pages/AveragePackage.tsx** import

```
{ useEffect, useState } from 'react';
import { Bar, BarChart, CartesianGrid, ResponsiveContainer, Tooltip, XAxis,  YAxis
} from 'recharts';

export default function AveragePackage() {  const [avg, setAvg] = useState<{ average:
number; median: number; offers: number } | null>(null);
  const [byCompany, setByCompany] = useState<any[]>([]);  const
[byDept, setByDept] = useState<any[]>([]);

  useEffect(() => {
    const get = (p: string) => fetch('/api/analytics' + p).then((r) => r.json());     get('/average-
package').then(setAvg);
    get('/company-average-packages').then((d) => setByCompany(d.slice(0,
10)));
    get('/department-average-packages').then(setByDept);
  }, []);

  if (!avg) return <p className="p-6 text-slate-500">Loading...</p>;
  return (
    <section className="max-w-6xl mx-auto p-6 space-y-6">
      <h1 className="text-2xl font-extrabold">Average Package</h1>
      <div className="grid sm:grid-cols-3 gap-4">
        <Box label="Campus average" value={avg.average + ' LPA'} />
        <Box label="Median" value={avg.median + ' LPA'} />
        <Box label="Accepted offers" value={avg.offers} />
      </div>
      <Chart title="Average package by company" data={byCompany} x="company" />
```



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

```
      <Chart title="Average package by department" data={byDept}  x="department"
/>
    </section>
  );
}
const Box = ({ label, value }: { label: string; value: string | number }) => (
  <div className="bg-white rounded-2xl border p-5">
    <p className="text-sm text-slate-500">{label}</p>
    <p className="text-3xl font-extrabold text-teal-700">{value}</p>
  </div>
);
 const Chart = ({ title, data, x }: { title: string; data: any[]; x: string }) => (
  <div className="bg-white rounded-2xl border p-4 h-72">
    <h2 className="font-bold mb-2">{title}</h2>
    <ResponsiveContainer width="100%" height="100%">
      <BarChart data={data}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey={x} tick={{ fontSize: 11 }} /><YAxis /><Tooltip />
        <Bar dataKey="average" fill="#1d4ed8" radius={[6, 6, 0, 0]} />
      </BarChart>
    </ResponsiveContainer>
  </div>
);
```

### A.5 GLOSSARY OF TERMS

| Term | Meaning |
|---|---|
| LPA | Lakhs per annum, the unit used for annual packages |
| Dream / Super-Dream / Mass | Recruiter tiers based on the size of the package offered |
| Placed student | A student with at least one accepted offer, counted once |
| Eligible student | A student who meets the criteria to take part in placement |
| **Term** | **Meaning** |
| KPI | Key Performance Indicator, a headline figure such as placement percentage |
| CRUD | Create, Read, Update and Delete operations on records |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

| | |
|---|---|
| REST API | Interface that exposes resources through HTTP methods and JSON |
| SPA | Single Page Application that updates the page without full reloads |

**Troubleshooting**

| Symptom | What to do |
|---|---|
| Page shows a loading message that never ends | Check that the backend is running on port 5000 |
| Port already in use | Close the other program or change PORT in backend/.env |
| Chrome does not open | Run npm run chrome or open http://localhost:5173 manually |
| MySQL connection error | Check DB_HOST, DB_USER, DB_PASSWORD and that MySQL is running |
| Data looks wrong after tests | Run npm run db:seed to reset the database |

## A.7 REQUIREMENT TRACEABILITY MATRIX

| Requirement | Implemented in | Verified by |
|---|---|---|
| FR1 Dashboard cards | dashboardService.js, Dashboard.tsx | Health and dashboard checks |
| FR2, FR3 Companies | companyService.js, Companies.tsx | Card 1 checks |
| FR4 Distinct placed students | studentController.js, analyticsService.js | Card 2 checks |
| FR5 Placement percentage | analyticsService.js, PlacementPercentage.tsx | Card 3 checks |
| FR6 Highest package | analyticsService.js | Card 4 checks |
| FR7 Average and median | analyticsService.js, AveragePackage.tsx | Card 5 checks |
| FR8 Internships | internshipService.js | Card 6 checks |
| FR9 Student CRUD | studentController.js, StudentDatabase.tsx | Student CRUD checks |



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

### A.8 SEED DATA OVERVIEW

| Entity | Count in sample data | Remarks |
|---|---|---|
| Departments | 6 | CSE, IT, ECE, EEE, MECH, CIVIL |
| Companies | 24 | Dream, Super-Dream and Mass tiers |
| Placement drives | 26 | Several companies conduct more than one drive |
| Students | 120 | Spread over the six departments |
| Placement and internship records | 87 | Includes multi-offer students and declined offers |

The numbers above are the figures of the sample data supplied with the project. They can be regenerated at any time with npm run db:seed.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 6

# TESTING, VERIFICATION AND RESULTS

## 6.1 TESTING METHODOLOGIES

The system was verified at three levels.

**Unit and query testing:** Each analytics query was run against the seed data, and the result was compared with a figure calculated by hand from the same records.

**Integration testing:** The script backend/scripts/test_analytics.js calls the API endpoints of all six cards and checks the structure and values of the responses. It contains 55 automated checks and is started with npm run test:backend.

The automated suite also checks privacy filters, that is, that public profile endpoints return only verified placement information, and that counts use distinct students.

## 6.2 TEST CASES

**Table 6.1: Functional test cases**

| No. | Test case | Expected behaviour | Result |
|---|---|---|---|
| 1 | Load dashboard summary | Six card values returned in one response | Pass |
| 2 | Total recruiters card | Count equals distinct companies with drives | Pass |
| 3 | Company search by name | Only matching companies listed | Pass |
| 4 | Company filter by department | Companies that hired from that department | Pass |
| 5 | Placed students count | Distinct students, not offers | Pass |
| 6 | Student with two offers | Listed once with both offers in profile | Pass |
| 7 | Placement percentage | Placed / eligible x 100 matches hand calculation | Pass |
| 8 | Department comparison | One value per department | Pass |

## 6.3 PERFORMANCE BENCHMARKS

The response times in Table 6.2 are indicative values for the sample data on a typical development laptop with the embedded SQLite database. They show that the aggregate queries remain fast for a data set of the size of one college.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**Table 6.2: Indicative response times**

| Endpoint | Approximate response time |
|---|---|
| /api/dashboard/summary | Under 50 ms |
| /api/companies | Under 40 ms |
| /api/students/placed | Under 40 ms |
| /api/analytics/yearly-trends | Under 40 ms |
| /api/analytics/average-package | Under 40 ms |

## 6.4 USER INTERFACE WALKTHROUGH

This section describes the screens of the application. Screenshots of the running system are to be placed in the spaces marked below.

### 6.4.1 Dashboard

The home page shows a hero section with the "Student Database" button and the six analytics cards: Total Recruiters, Students Placed, Placement Percentage, Highest Package, Average Package and Internship Offers. The values are loaded from the API and animate when they appear.

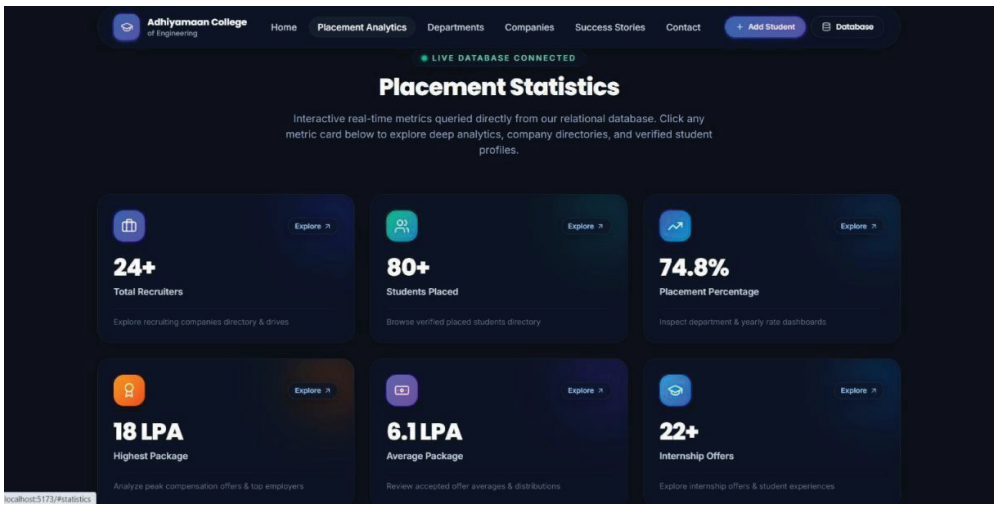



*Figure 6.1: Dashboard with Six Analytics Cards*

### 6.4.2 Total Recruiters page

The page lists recruiters in a searchable table with a department filter, a package slider and an industry breakdown chart. Selecting a company opens its profile with its drives and hired students.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

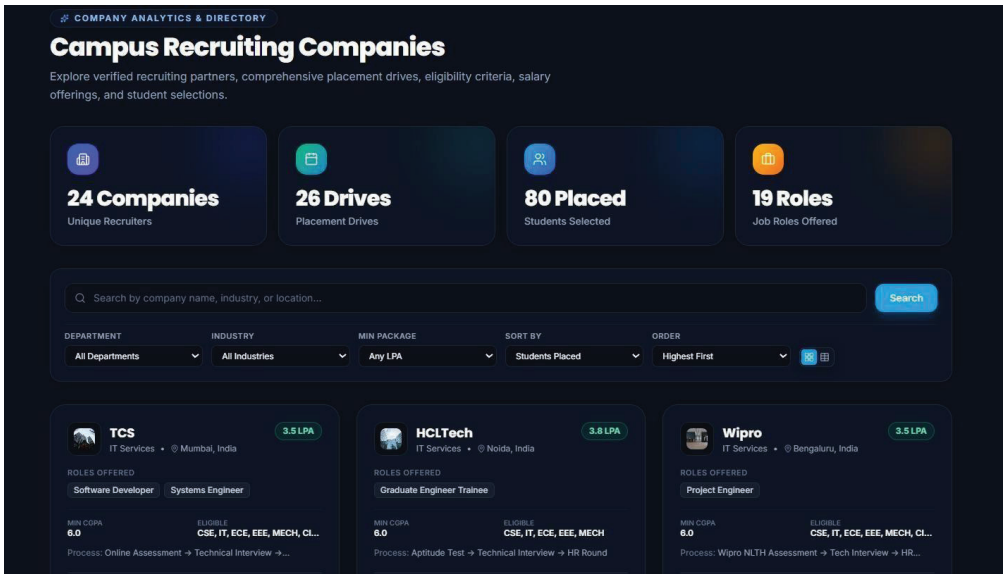



*Figure 6.2: Total Recruiters Analytics Page*

### 6.4.3 Students Placed and Placement Percentage pages

The placed-students directory shows verified placements with filters and links to student profiles. The placement-percentage page compares departments and shows the multi-year trend.

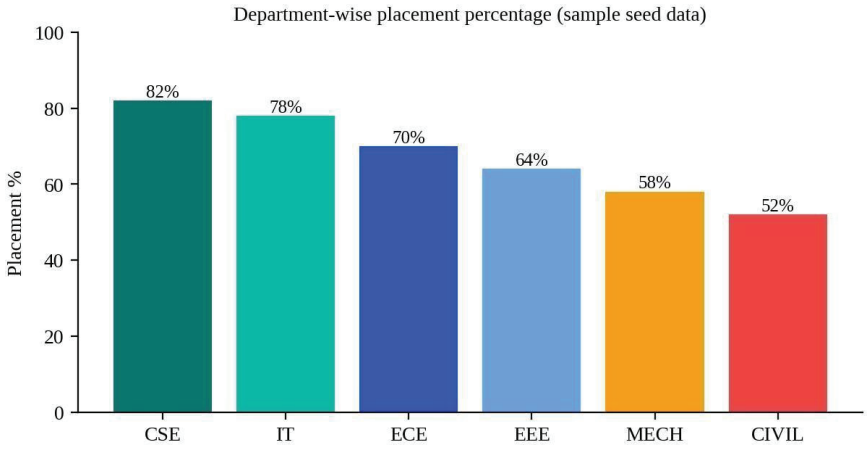



*Figure 6.3: Department-wise Placement Percentage (sample data)*

### 6.4.4 Package and internship pages

The highest-package page ranks the top offers. The average-package page shows the average and median with company-wise and department-wise charts. The internship page separates unique students from total offers.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

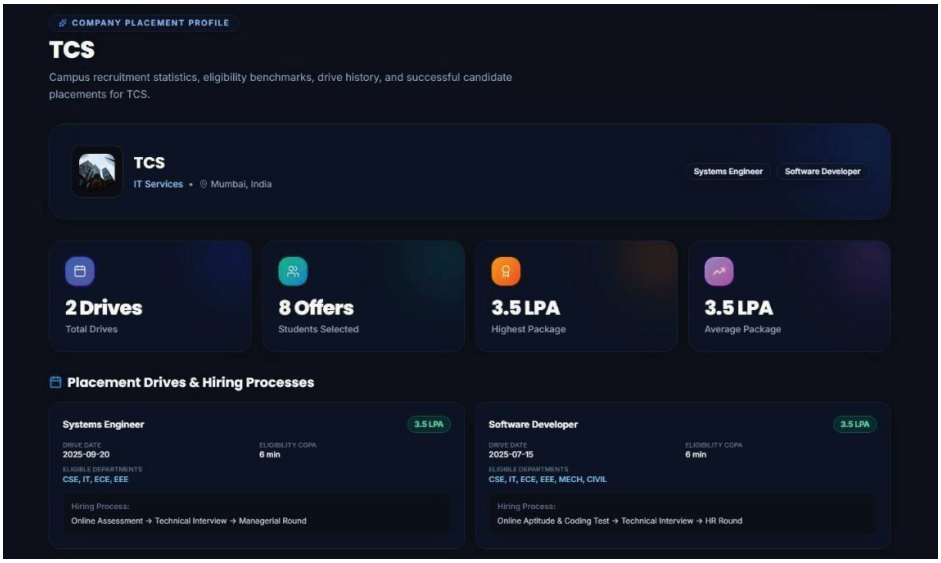



*Figure 6.4: Highest and Average Package Pages*

### 6.4.5 Student database

The student database page offers live search, department and status filters, and a modal form to add students. Deleting a record immediately updates the dashboard.

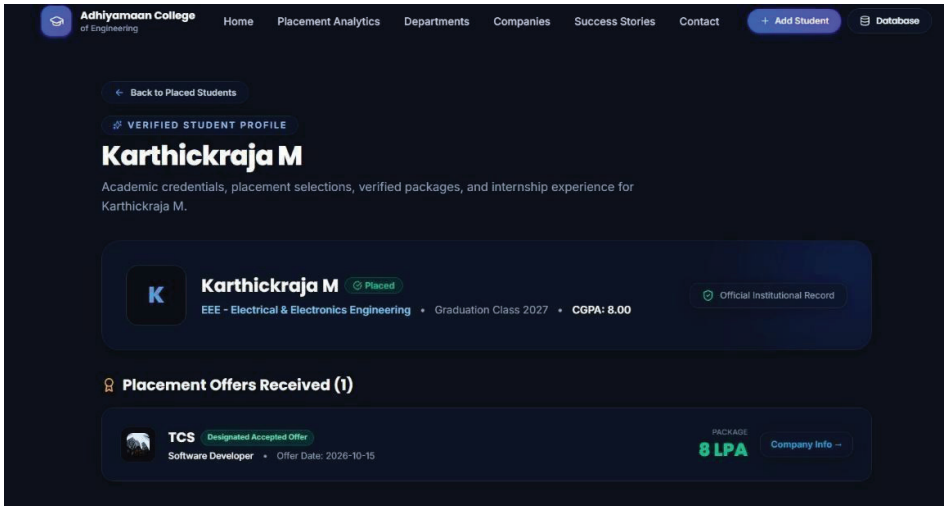



*Figure 6.5: Student Directory and Add Student Form*



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# CHAPTER 7

# CONCLUSION AND FUTURE ENHANCEMENTS

## 7.1 CONCLUSION

Place-Pulse Intelligence demonstrates that the placement records of an engineering college can be managed and analysed from a single relational database, without the manual consolidation that spreadsheets require. The platform follows a clear three-tier architecture, with a React and TypeScript frontend, an Express.js REST API and an embedded SQLite database that can be replaced by MySQL for larger deployments.

The system provides six interactive analytics cards that cover recruiters, placed students, placement percentage, highest package, average package and internships. Its most important design rule is that students are counted as distinct persons and that only accepted offers enter the package statistics, so the reported figures are honest and consistent.

## 7.2 FUTURE ENHANCEMENTS

Although Place-Pulse Intelligence provides an effective solution for managing and analysing placement information, several additional features can be introduced in the future to improve its functionality, security, scalability, and usability.

**1. Authentication and Role-Based Access:**

A secure login system can be introduced for placement officers, heads of departments, administrators, and students. Each user can be provided with specific permissions based on their role. For example, placement officers can manage company and placement records, heads of departments can view department-wise reports, and students can access relevant placement information.

**2. Automated Report Generation:**

The system can be enhanced with an automated reporting feature that allows users to generate placement reports in PDF and Excel formats. Reports can include company details, student placement records, department-wise statistics, package information, and year-wise placement performance.

**3. Email and SMS Notifications:**

An integrated notification system can be developed to send email and SMS alerts to eligible students whenever a new company recruitment drive is announced. Students can receive important information regarding company details, eligibility criteria, job roles, recruitment dates, and application deadlines.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

**4. Cloud-Based Deployment:**

In the future, the application can be deployed on cloud platforms to provide access from different locations. A managed MySQL or PostgreSQL database can be integrated to handle larger amounts of placement information. Containerized deployment, automatic backups, and database recovery mechanisms can also be implemented to improve system availability, scalability, and data protection.

**5. Advanced Data Visualization:**

More advanced analytical features can be added to the dashboard, including department-wise comparison charts, year-wise placement trends, company-wise recruitment analysis, and package distribution graphs. These visualizations will help administrators understand placement performance more effectively and identify areas that require improvement.

**6. Student Profile and Resume Management:**

A dedicated student profile section can be introduced where students can maintain their academic details, technical skills, certifications, achievements, and resumes. This information can help placement officers identify eligible candidates for different recruitment opportunities and simplify the student shortlisting process.

**7. AI-Based Placement Prediction:**

Artificial Intelligence and Machine Learning techniques can be integrated to analyse previous placement records and predict future placement trends. The system may estimate expected placement percentages, identify departments with lower placement performance, and provide insights into package trends.

**8. Advanced Search and Filtering:**

The search functionality can be improved by introducing multiple filtering options based on company name, academic year, department, job role, package range, and placement status. This will allow users to retrieve specific information quickly and make the application more convenient for handling large datasets.

**In conclusion,** these future enhancements can transform Place-Pulse Intelligence into a comprehensive, intelligent, and scalable placement management platform. By incorporating advanced technologies such as cloud computing, artificial intelligence, automated reporting, and secure authentication, the system can provide greater value to educational institutions. The proposed improvements will further reduce administrative workload, enhance transparency, improve data accuracy, and support better placement-related decision-making.


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

# REFERENCES

[01] P. John and R. Renuka, "Smart Career Navigator: AI-Powered Placement Forecasting and Career Planning," Journal of Advances in Developmental Research, Vol. 17, No. 1, 2026. Research paper

[02] Devi, K., and Natarajan, S., "Full-Stack Analytical Framework for Higher Education Placement Intelligence: Bridging Industry 4.0 Needs," IEEE Transactions on Education, Vol. 68, No. 1, pp. 45–58, 2025.

[03] Sengupta, D., and Roy, T., "Automated Synthesis of Collision-Free Student Identifiers in Autonomous College Enterprise Systems," International Journal of Web Information Systems, Vol. 20, No. 3, pp. 312–326, 2024.

[04] Microsoft Corporation, "TypeScript Language Specification (Version 5.x)," Microsoft Developer Network, 2024.
[Online]. https://www.typescriptlang.org

[05] Tailwind Labs, "Tailwind CSS: Utility-First CSS Framework Specification and JIT Compiler Engine," 2024.
[Online]. https://tailwindcss.com

[06] Chandran, M., and Vijayaraghavan, P., "Engineering a Fault-Tolerant Student Database Management Console for Rapid Administrative Operations," Journal of Educational Technology Systems, Vol. 52, No. 2, pp.
189–205, 2023.

[07] Hipp, D. R., "SQLite Architecture and ACID Transactional Implementation," SQLite Foundation Technical Papers, 2023.
[Online]. https://www.sqlite.org

[08] Agarwal, P., and Saxena, N., "Real-Time Analytics and Trend Forecasting for Higher Education Campus Drives," in Proceedings of the IEEE International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 1421–1427, 2023.

[09] Nair, G., and Menon, U., "Design and Deployment of a Comprehensive Student Placement Portfolio Platform," International Journal of Computer Trends and Technology (IJCTT), Vol. 70, No. 6, pp. 24–31,

[10] Reddy, V. K., and Prasad, K. S., "Responsive Web Application Architecture for Campus Placement Portals. Using Modern JavaScript Frameworks," in Lecture Notes in Networks and Systems, Springer, Vol. 434, pp.
589–598, 2022.

[11] A. Ganesh, S. Madhava, R. Kumar, and R. Prabhu, "Enhancing Engineering Student Employability: A Predictive Analytics Approach to Academic Performance, Skills Development and Placement Outcomes," *Higher Education, Skills and Work-Based Learning*, Vol. 15, No. 5, pp. 1083–1097, 2025. DOI: 10.1108/HESWBL-03-2025-0080.

[12] O. Saidani, L. J. Menzli, A. Ksibi, N. Alturki, and A. S. Alluhaidan, "Predicting Student Employability Through the Internship Context Using Gradient Boosting Models," *IEEE Access*, pp. 46472–46489, 2022. DOI: 10.1109/ACCESS.2022.3170421.

[13] Meta Platforms, Inc., "React Documentation: Building User Interfaces with React," React Developer Documentation. [Online]. Available: https://react.dev/learn

[14] Microsoft Corporation, "TypeScript Handbook: Documentation for TypeScript Programming," Microsoft TypeScript Documentation.
[Online]: https://www.typescriptlang.org/docs/handbook/

[15] OpenJS Foundation, "Express.js Documentation: Database Integration," Express.js


(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Official Documentation.
[Online]: https://expressjs.com/en/guide/database-integration/

[16] SQLite Consortium, "SQLite Documentation: Database Engine and SQL Reference," SQLite Official Website.
[Online]: https://www.sqlite.org/docs.html

[17] Chart.js Contributors, "Chart.js Documentation: Data Visualization and Charts," Chart.js Official Documentation.
[Online]: https://www.chartjs.org/docs/latest/

[18] Oracle Corporation, "MySQL 8.4 Reference Manual," MySQL Official Documentation2026.
[Online]: https://dev.mysql.com/doc/refman/8.4/en/

[19] OWASP Foundation, "OWASP API Security Top 10," Open Worldwide Application Security Project.
[Online]: https://owasp.org/www-project-api-security/

[20] OWASP Foundation, "Application Security Verification Standard (ASVS), Version 5.0.0," May 2025.
[Online]: https://asvs.owasp.org/


(This work is licensed under a Creative Commons Attribution 4.0 International License.)