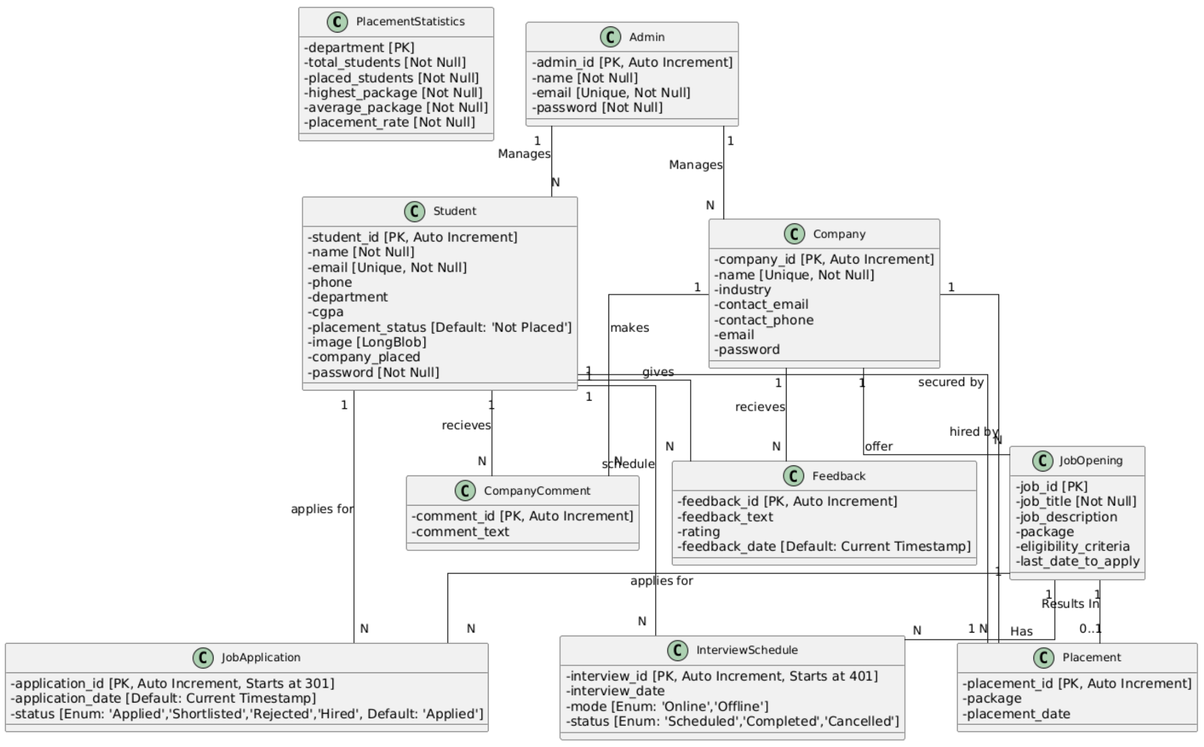
UML DIAGRAMS:

1. **CLASS DIAGRAM:**



PlacementStatistics
-department [PK]
-total_students [Not Null]
-placed_students [Not Null]
-highest_package [Not Null]
-average_package [Not Null]
-placement_rate [Not Null]
Admin
-admin_id [PK, Auto Increment]
-name [Not Null]
-email [Unique, Not Null]
-password [Not Null]
Manages
Manages
Student
-student_id [PK, Auto Increment]
-name [Not Null]
-email [Unique, Not Null]
-phone
-department
-cgpa
-placement_status [Default: 'Not Placed']
-image [LongBlob]
-company_placed
-password [Not Null]
Company
-company_id [PK, Auto Increment]
-name [Unique, Not Null]
-industry
-contact_email
-contact_phone
-email
-password
makes
gives
recieves
recieves
secured by
hired by
offer
schedule
applies for
applies for
CompanyComment
-comment_id [PK, Auto Increment]
-comment_text
Feedback
-feedback_id [PK, Auto Increment]
-feedback_text
-rating
-feedback_date [Default: Current Timestamp]
JobOpening
-job_id [PK]
-job_title [Not Null]
-job_description
-package
-eligibility_criteria
-last_date_to_apply
Results In
Has
JobApplication
-application_id [PK, Auto Increment, Starts at 301]
-application_date [Default: Current Timestamp]
-status [Enum: 'Applied','Shortlisted','Rejected','Hired', Default: 'Applied']
InterviewSchedule
-interview_id [PK, Auto Increment, Starts at 401]
-interview_date
-mode [Enum: 'Online','Offline']
-status [Enum: 'Scheduled','Completed','Cancelled']
Placement
-placement_id [PK, Auto Increment]
-package
-placement_date

Admin and Student (1:N): One admin manages multiple students.

Admin and Company (1:N): One admin manages multiple companies.

Company and JobOpening (1:N): A company can post multiple job openings.

Student and JobApplication (1:N): A student can submit multiple job applications.

Company and Placement (1:N): A company can have multiple placements.

JobOpening and Placement (1:N): A job opening many in a placement.

Student and Feedback (1:N): A student can submit multiple feedback entries.

JobOpening and JobApplication (1:N): A job opening can receive multiple applications from different students.

Student and Placement (1:0..1): A student may or may not get placed.

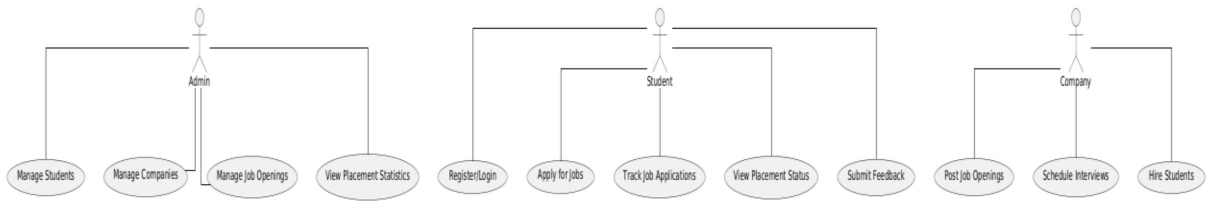
Student and CompanyComment(1:N): one student can give multiple companycomments

Student and interviewschedule(1:N): one student can give multiple interviews

Company and feedback(1:N):one company can get multiple feedback from different students

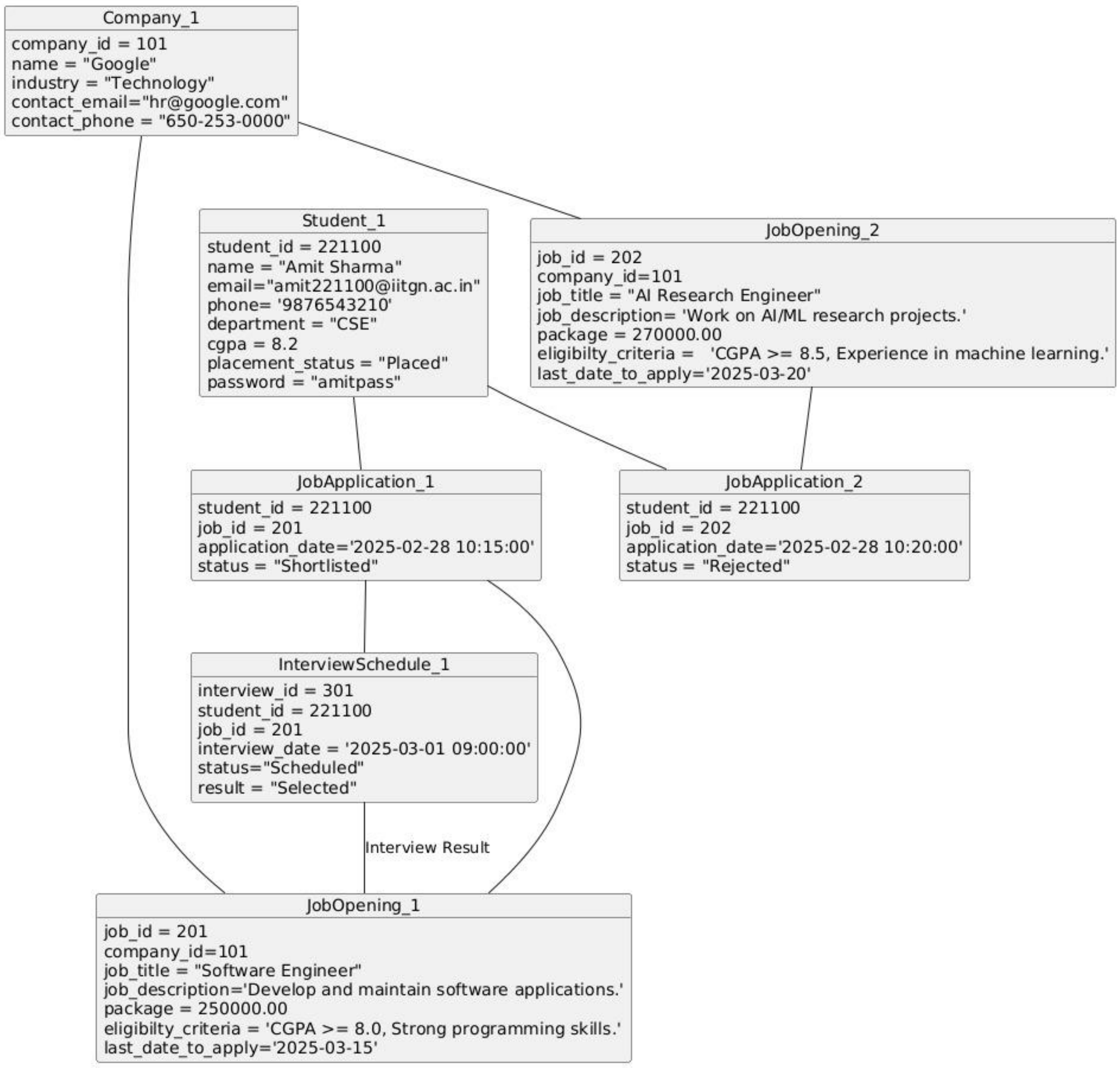
Company and CompanyComment(1:N): one company will take many interviews from that get many companycomments

2. Use Case Diagrams:



The Use Case Diagram illustrates the interactions between the Admin, Students, and companies in the Placement Management System. The admin manages students, companies, and job openings, while the student registers, applies for jobs, tracks applications, and submits feedback. The Company posts job openings, schedules interviews, and hires students.

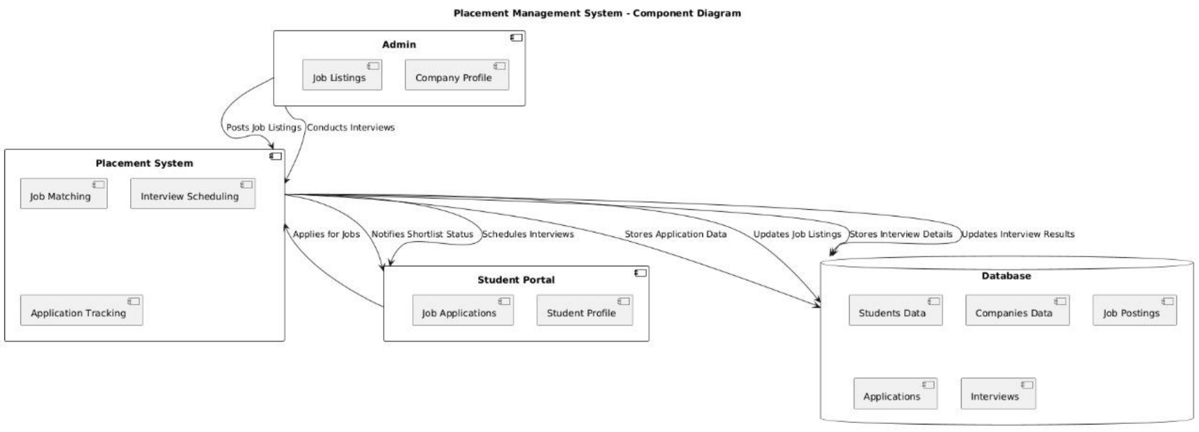
3. Object diagram:



Company_1
company_id = 101
name = "Google"
industry = "Technology"
contact_email="hr@google.com"
contact_phone = "650-253-0000"
Student_1
student_id = 221100
name = "Amit Sharma"
email="amit221100@iitgn.ac.in"
phone= '9876543210'
department = "CSE"
cgpa = 8.2
placement_status = "Placed"
password = "amitpass"
JobOpening_2
job_id = 202
company_id=101
job_title = "AI Research Engineer"
job_description= 'Work on AI/ML research projects.'
package = 270000.00
eligibilty_criteria = 'CGPA >= 8.5, Experience in machine learning.'
last_date_to_apply='2025-03-20'
JobApplication_1
student_id = 221100
job_id = 201
application_date='2025-02-28 10:15:00'
status = "Shortlisted"
JobApplication_2
student_id = 221100
job_id = 202
application_date='2025-02-28 10:20:00'
status = "Rejected"
InterviewSchedule_1
interview_id = 301
student_id = 221100
job_id = 201
interview_date = '2025-03-01 09:00:00'
status="Scheduled"
result = "Selected"
Interview Result
JobOpening_1
job_id = 201
company_id=101
job_title = "Software Engineer"
job_description='Develop and maintain software applications.'
package = 250000.00
eligibilty_criteria = 'CGPA >= 8.0, Strong programming skills.'
last_date_to_apply='2025-03-15'

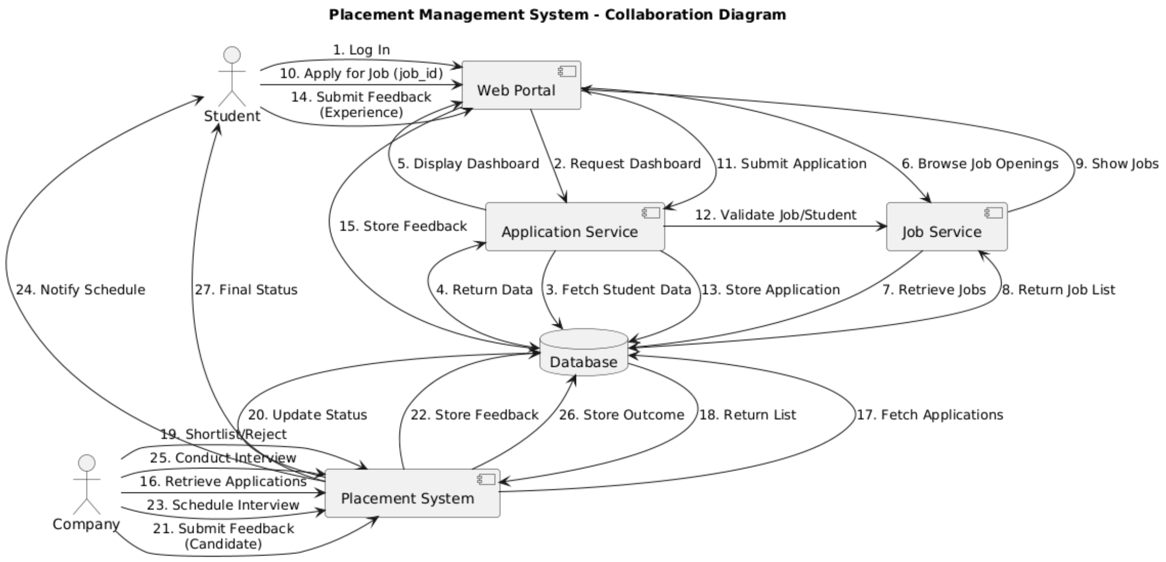
If you take Amit Sharma, he applied for two jobs. One is shortlisted, and the other one is rejected. He got an interview schedule, and the interview result is he got selected with a package of 250000.00 in Google.

4. Component diagram:



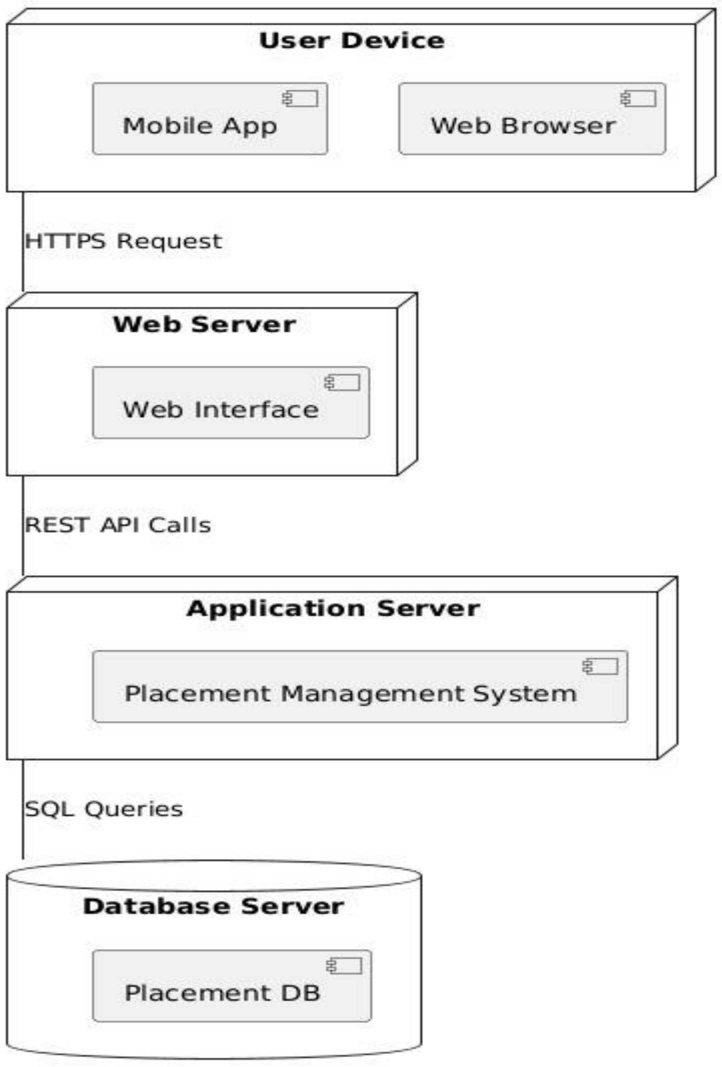
The Placement Management System comprises four key components: The Admin manages job listings and company profiles, facilitating job postings and interview processes. The Placement System enables job matching and interview scheduling to streamline student-company interactions. The Application Tracking component allows students to apply for jobs, view shortlist status, and manage interviews through the Student Portal and their profiles. Finally, the Database stores all essential data, including student records, company details, and job postings, while the Application component specifically handles interview-related operations.

5. Collaboration Diagrams:



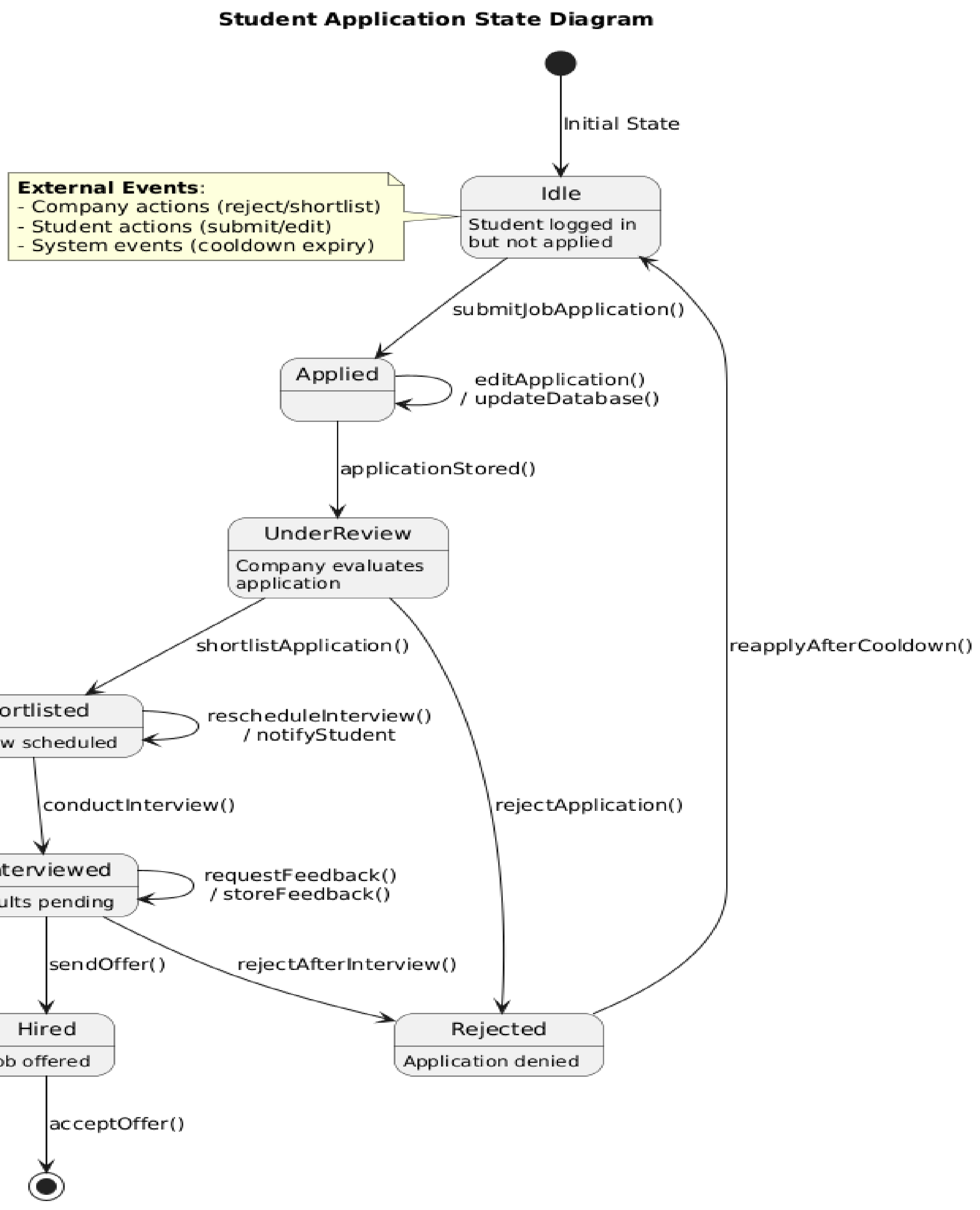
The student initiates the process by logging in, requesting their dashboard, browsing job openings, and applying for positions, which triggers the validation and storage of applications. The Application Service and Job Service handle core operations like validating applications, scheduling interviews, and managing feedback while interacting with the Database to store and retrieve data. The Web Portal serves as the interface for students to submit applications and feedback, with the system updating statuses and outcomes throughout the process. Finally, the Database stores all critical information, including applications, feedback, and interview outcomes, ensuring seamless data flow across the system.

6. Deployment Diagram:



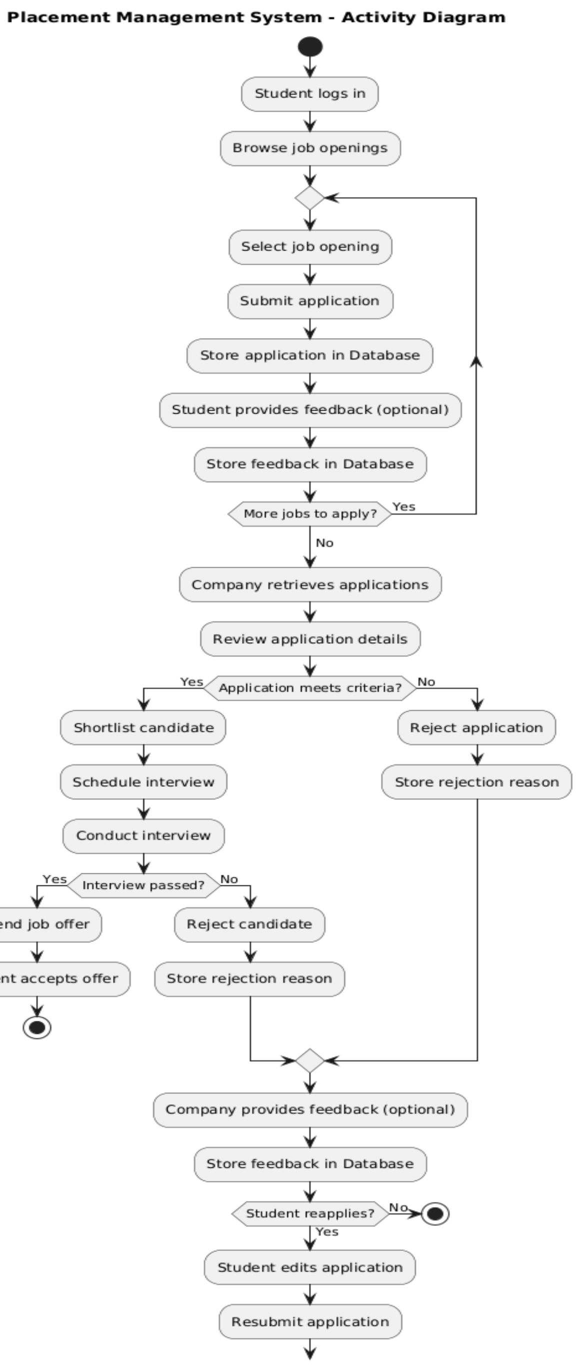
User Devices (Mobile Apps and Web Browsers) initiate interactions by sending HTTPS requests to the Web Server, which hosts the Web Interface. The Web Server communicates with the Application Server (hosting the Placement Management System) through REST API calls to process business logic. The Application Server then executes SQL queries to interact with the Database Server, where all data is stored in the Placement DB. This tiered structure ensures secure, scalable, and efficient data flow from user interfaces to backend storage.

7. Statechart Diagram:



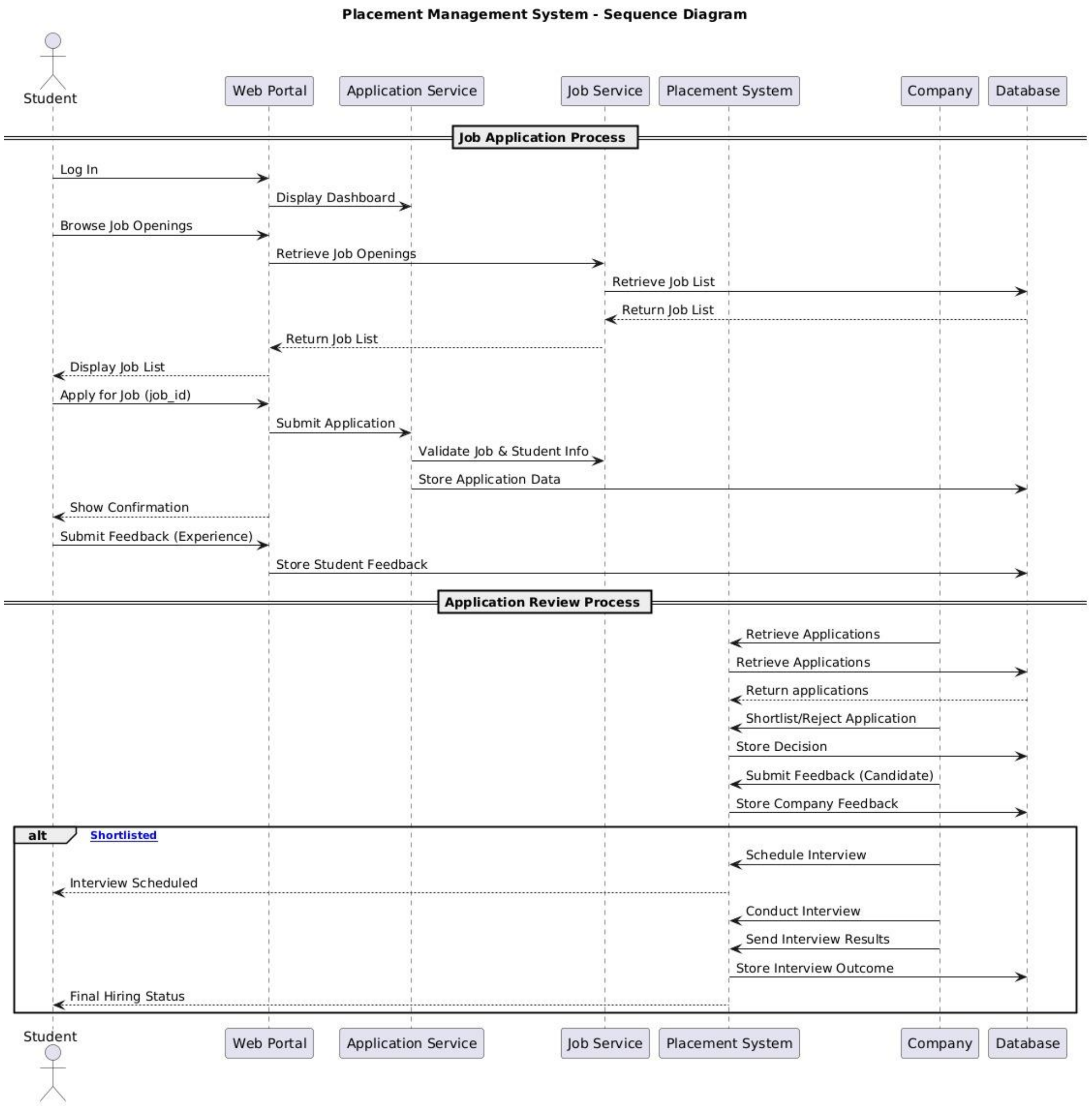
The state diagram illustrates the various stages a student's job application goes through, starting from the "Idle" state where the student is logged in but hasn't applied yet. Upon applying, the system transitions to the "UnderReview" state, where the company evaluates the application. If shortlisted, the application moves to the "Shortlisted" state, leading to interview scheduling and execution. After the interview, the "Interviewed" state awaits results, with possible outcomes like receiving an offer or rejection. Finally, the "Hired" state is reached if the student accepts the job offer, and completes the process.

8. Activity Diagrams:



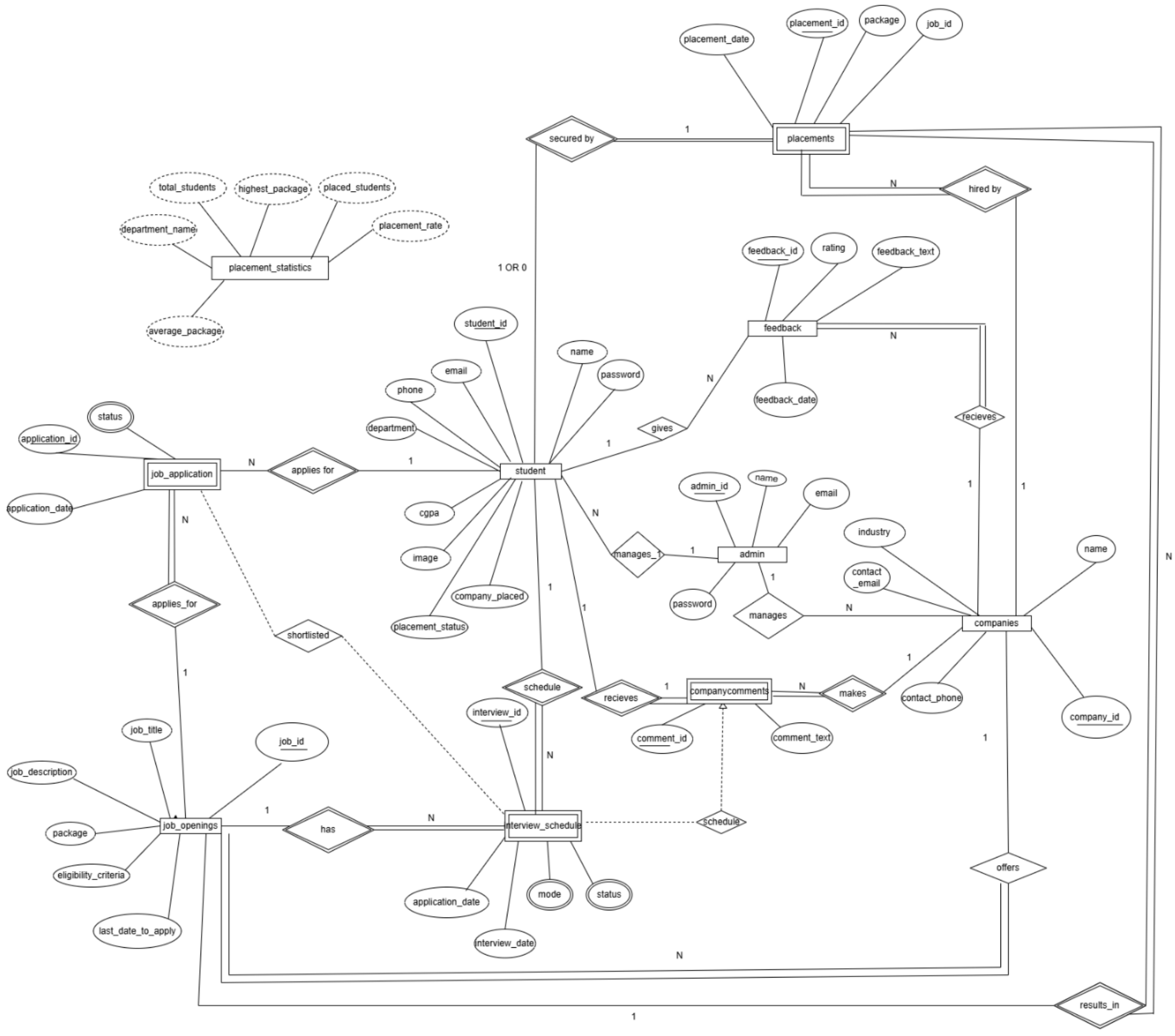
The process begins with the student logging in, browsing job openings, and submitting applications, which are stored in the database. The company then retrieves and reviews these applications, shortlisting candidates who meet the criteria or rejecting others with stored reasons. Shortlisted candidates proceed to interviews, and based on performance, they either receive a job offer or a rejection. If the student accepts the offer, the process concludes; otherwise, they may reapply or edit their application for future opportunities. Optional feedback from both parties is stored for improvement. The loop allows students to apply for multiple jobs until they secure a position.

9. Sequence Diagram:



This sequence diagram illustrates the step-by-step interactions between the student, web portal, services, and database in a Placement Management System. The process begins with the student logging in, browsing job openings, and applying for a job, which triggers the system to store the application and confirm submission. The company then retrieves applications, reviews them, and makes shortlisting or rejection decisions, storing feedback for both parties. If shortlisted, the student proceeds to interviews, and the final hiring status is communicated back. The diagram highlights key actions like job retrieval, application submission, feedback storage, and interview outcomes, ensuring a clear flow of operations between all system components

ER DIAGRAM:



Important points regarding the ER diagram:

1. Indirect Relationships (Dashed Lines in ER Diagram)

In some cases, an entity influences another entity's existence or behavior without having a direct foreign key reference. Instead of a direct relationship, we represent this using a dashed line to indicate dependency.

Example: Job Applications & Interview Schedule

Interviews are only scheduled if a student is shortlisted in Job Applications.

There's no direct foreign key, but the status ("Shortlisted") in Job Applications triggers Interview Scheduling.

Solution: Use a dashed relationship line labeled "Shortlisted" between Job Applications and Interview Schedules.

Example: Company Comments & Interview Schedule

Company Comments are provided based on a student's performance in interviews.

There's no foreign key linking Interview Schedules and Company Comments, but comments happen after interviews.

Solution: Use a dashed line labeled "Feedback On" between Interview Schedules and Company Comments.

2. Total Participation (Double Lines in ER Diagram)

Total participation means that an entity must be associated with another entity, meaning every instance of this entity must participate in the relationship. To represent this, we use double lines in the ER diagram.

Example: Placements & Students

If a student is placed, they must have a corresponding placement record.

Solution: Use double lines between Placements and Students.

Example: Job Applications & Students

Every job application must be submitted by a student.

Solution: Use double lines between Job Applications and Students.

Example: Job Applications & Job Openings

Every Job Application must reference a Job Opening.

Solution: Use double lines between Job Applications and Job Openings

Mappings:

Admin and Student (1:N): One admin manages multiple students.

One admin (e.g., Admin One) can manage multiple students (e.g., Amit Sharma, Neha Verma, and Rahul Singh) in the placement system, establishing a one-to-many (1:N) relationship.

Admin and Company (1:N): One admin manages multiple companies.

One admin (e.g., Admin One) can manage multiple companies (e.g., Google, Microsoft, and Amazon) in the placement system, creating a one-to-many (1:N) relationship.

Company and JobOpening (1:N): A company can post multiple job openings.

One company (e.g., Google) can post multiple job openings (e.g., Software Engineer, AI Research Engineer) in the placement system, forming a one-to-many (1:N) relationship

Student and JobApplication (1:N): A student can submit multiple job applications.

One student (e.g., Amit Sharma) can submit multiple job applications (e.g., for Software Engineer at Google and AI Research at Microsoft), creating a one-to-many relationship where each application belongs to only one student

Company and Placement (1:N): A company can have multiple placements.

One company (e.g., Google) can have multiple placement records (e.g., hiring Amit Sharma and Priya Patel) for these placed students, establishing a one-to-many relationship where each placement is associated with only one company.

JobOpening and Placement (1:N): A job opening One-to-many in a placement.

One job opening (e.g., Software Engineer at Google) can result in multiple placements (e.g., hiring Amit Sharma and Rahul Singh), forming a one-to-many relationship where each placement corresponds to only one job opening.

Student and Feedback (1:N): A student can submit multiple feedback entries.

One student (e.g., Amit Sharma) can receive multiple feedback entries (e.g., from Google and Microsoft interviews), creating a one-to-many relationship where each feedback is about only one student.

JobOpening and JobApplication (1:N): A job opening can receive multiple applications from different students.

One job opening (e.g., Software Engineer at Google) can receive multiple applications (e.g., from Amit Sharma and Neha Verma), forming a one-to-many relationship where each application targets only one job opening.

Student and Placement (1:0..1): A student may or may not get placed.

One student (e.g., Amit Sharma) can have only one placement record (e.g., at Google as a Software Engineer), creating a one-to-one relationship where each placement corresponds to exactly one student. Or if any student which doesn't placed it is 1 to 0. like Vivek Bhat

Student and Feedback (1:N): A student can submit multiple feedback entries.

One student (e.g., Amit Sharma) can receive multiple feedback entries (e.g., from Google and Microsoft interviews), forming a one-to-many relationship where each feedback is tied to only one student.

JobOpening and JobApplication (1:N): A job opening can receive multiple applications from different students.

One job opening (e.g., Software Engineer at Google) can receive multiple applications (e.g., from Amit Sharma and Neha Verma), creating a one-to-many relationship where each application applies to only one job opening.

Student and Placement (1:0..1): A student may or may not get placed.

One student (e.g., Amit Sharma) can have at most one placement record (e.g., at Google as a Software Engineer), forming a zero-or-one-to-one relationship where a student may or may not be placed.

Student and CompanyComment(1:N): one student can give multiple company comments.

One student (e.g., Amit Sharma) can have multiple comments from companies (e.g., feedback from Google and Microsoft recruiters), establishing a one-to-many relationship where each comment references only one student.

Student and interview schedule (1:N): one student can give multiple interviews.

One student (e.g., Amit Sharma) can have multiple scheduled interviews (e.g., with Google and Microsoft), creating a one-to-many relationship where each interview slot is assigned to only one student.

Company and feedback(1:N): one company can get multiple feedback from different students.

One company (e.g., Google) can provide multiple feedback entries (e.g., for Amit Sharma and Priya Patel), forming a one-to-many relationship where each feedback is given by only one company.

Company and CompanyComment(1:N): one company will take many interviews that get many companycomments.

One company (e.g., Google) can post multiple comments (e.g., about different students), establishing a one-to-many relationship where each comment originates from only one company.

Weak entities:

A JobApplication is a weak entity because it depends on two strong entities: Student and JobOpening. It cannot exist independently as each application must be associated with a specific student applying for a specific job.

InterviewSchedule is a weak entity because it depends on the JobApplication entity. An interview is scheduled only if a student applies for a job, meaning it cannot exist without a corresponding job application.

CompanyComment is a weak entity because it relies on the Student entity. A comment is written by a student about a company, meaning it cannot exist independently without being associated with a specific student.

Placement is a weak entity because it depends on both Student and Company entities. A placement record only exists if a student is placed in a company for a specific job, meaning it cannot exist independently without these relationships.

Explanation of the UML diagrams, how entities and relationships are derived:

UML (Unified Modeling Language) diagrams are widely used in software engineering to model systems, while ER (Entity-Relationship) diagrams are specifically designed to represent database structures. When transitioning from UML to an ER diagram, we primarily use two types of UML diagrams: Class Diagram → Represents entities (tables) and their attributes. Use Case Diagram → Helps in identifying the relationships between entities.

UML Class Diagram to ER Diagram Transition

A UML Class Diagram consists of:

1. Classes (Entities in ERD) - Represent database tables.
2. Attributes - Become table columns.
3. Methods (if applicable) - Converted into database operations (Stored Procedures or Constraints).
4. Associations (Relationships in ERD) - Define foreign key constraints between entities.

UML Use Case Diagram to ER Diagram Transition

A UML Use Case Diagram helps understand how users interact with the system. This helps in Identifying key entities (who interact with the system) and Deriving relationships (how they interact).

Justification for the transition from UML to ER:

ER diagrams explicitly show NOT NULL, UNIQUE, and ON DELETE rules that UML doesn't express. UML Limitation: Doesn't distinguish weak/strong entities clearly. But ER shows the weak entities/relationships. when operating from many-to-many relations the UML directly connects without any junction table but er will have an associative table in between for the many-to-many relation.

Team Contribution:

1.Raj Kamal: I have studied and researched ER diagrams and have drawn the ER diagram from scratch in draw.io by understanding UML diagrams and helped in writing explanations with my other teammates.

2. Avinash: I worked on ER diagram mapping, including weak entities, and created UML diagrams with assistance from Raj Kamal, while some were drawn by Manohar. Additionally, I helped with mapping examples and wrote the theoretical part with support from my teammates.

3. Manohar: Created UML diagrams and provided detailed explanations. Additionally, I collected information for preparing ER diagrams and compiled everything into a PDF.