CS432 Databases
ScholarEase- Scholarship/Stipend Management System
Module 2 Task

23110103@iitgn.ac.in; 23110140@iitgn.ac.in; 23110192@iitgn.ac.in; 23110353@iitgn.ac.in

Name	Roll Number
Diya Nandan	23110103
Ishika Paresh Patel	23110140
Maitri Chaudhari	23110192
Vardayini Agrawal	23110353

Project Overview

Our project, ScholarEase, aims to streamline the process of student registration, scholarship application, eligibility verification, approval, and fund disbursement. Previously, in the Module 1 Task, we created the entire database for ScholarEase in MySQL, considering all the constraints for each table.

Module 2 Aim

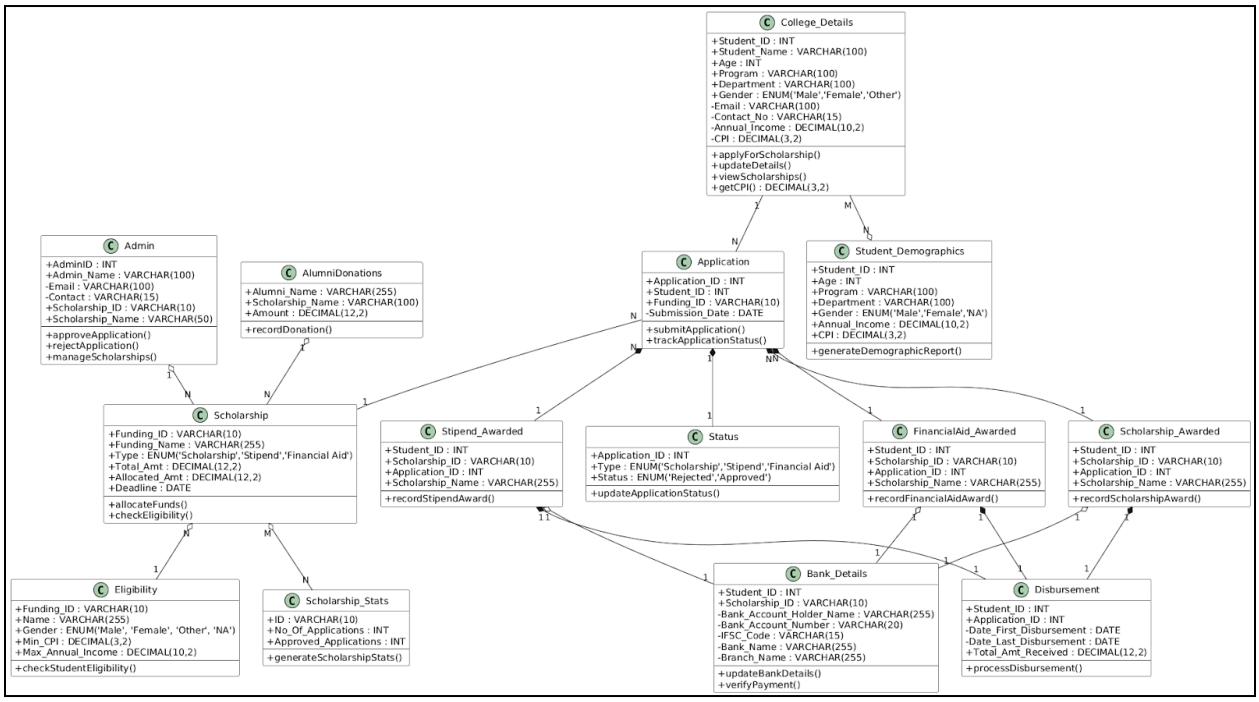
The goal is to visualize the structure and relationships of the database entities effectively. We create multiple UML diagrams, which are later transitioned into a single ER diagram. By doing so, we were able to reimage the project further.

UML Diagrams

Unified Modeling Language(UML) diagrams illustrate the structural and behavioral aspects of the system, ensuring a clear understanding of its functionality. They focus on object-oriented design, representing classes, attributes, methods, and relationships.

Here, we have made eight major UML diagrams relevant to our project.

1. Class Diagram



A Class Diagram is a type of UML diagram that represents the static structure of a system by showing its classes, attributes, and operations. It defines how different entities (classes) in the system relate to each other and interact.

Each rectangle represents a class, where:

- The top section contains the class name.
- The middle section contains attributes (fields) with their data types.
- The bottom section contains methods (functions) that the class can perform.

Example: Class Name: Admin

- Attributes: AdminID, Admin_Name, Email, Contact, Scholarship_ID, Scholarship_Name
- Methods: approveApplication(), rejectApplication(), manageScholarships()
- **Role:** Manages scholarship applications and approvals.

Cardinality:

- **1:1 (One-to-One)**
  Example**:** Scholarship_Awarded → Disbursement (1:1)
  For each scholarship awarded there is a particular disbursement.

- **1:N (One-to-Many)**
  Example: Admin → Scholarship (1:N)
  One Admin can manage multiple scholarships.

- **N:M (Many-to-Many)**
  Example: Scholarship → Scholarship_Stats (N: M)
  One scholarship can be included in more than one statistic, and one statistic can have more than one scholarship for analysis.

Arrow Notations

- **Association (Solid Line):** Represents a direct relationship between two classes.
  Example: College_Details —— Application

  A Student applies for a Scholarship.

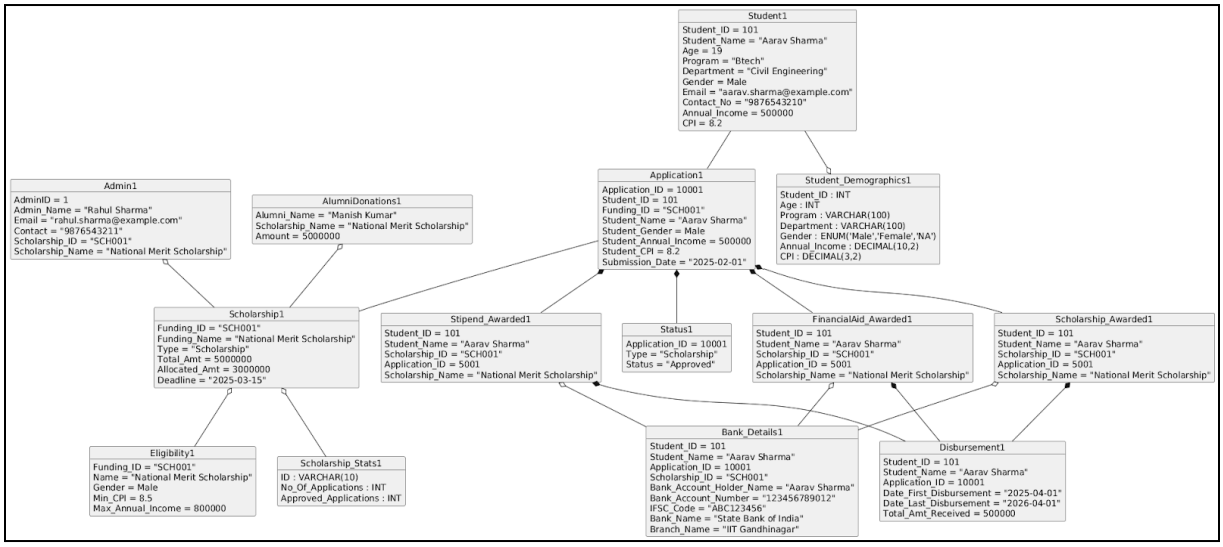
- **Aggregation (White Diamond):** Shows a whole-part relationship (e.g., a scholarship contains scholarship_stats entity within it).
  Example: Scholarship ◇—— Scholarship_Stats

  A scholarship has statistical records, but if the scholarship is removed, the stats can still be used in reports. Scholarship stats can exist independently.

- **Composition (Black Diamond):** Represents strong ownership. Indicates composition, meaning the lifetime of the contained object depends on the container.
  Example: Application ◆—— Status

  This means that the Status entity is entirely dependent on the existence of the Application entity.
  If the Application entity is deleted, then there is no point of the existence of the Status entity.

2. **Object Diagram**



An object diagram represents a snapshot of the system at a specific moment. It illustrates instances (objects) of classes, their attributes, and the links (relationships) between them.

Each rectangle represents a class, where:

- The top section contains the object name.
- The bottom section contains attributes (fields).

Example for an Object in an Object Diagram

Student1 Object (derived from College_Details)

Student_ID: 101
Student_Name: Aarav Sharma
Program: Btech
Age: 20
Department: Civil Engineering
Gender: Male
Email: aarav.sharma@example.com
Contact no.: 9876543210
Annual_Income: 500000
CPI: 8.2

Arrow Notations Used in the Object Diagram

- **Association (Solid Line):** Represents a direct relationship between two classes.
  Example: Student1 —— Application1

  A Student1 object applies for a Scholarship1 object from the Scholarship Entity.

- **Aggregation (White Diamond):** Shows a whole-part relationship (e.g., a scholarship contains scholarship_stats entity within it).
  Example: Scholarship1 ◇—— Scholarship_Stats1

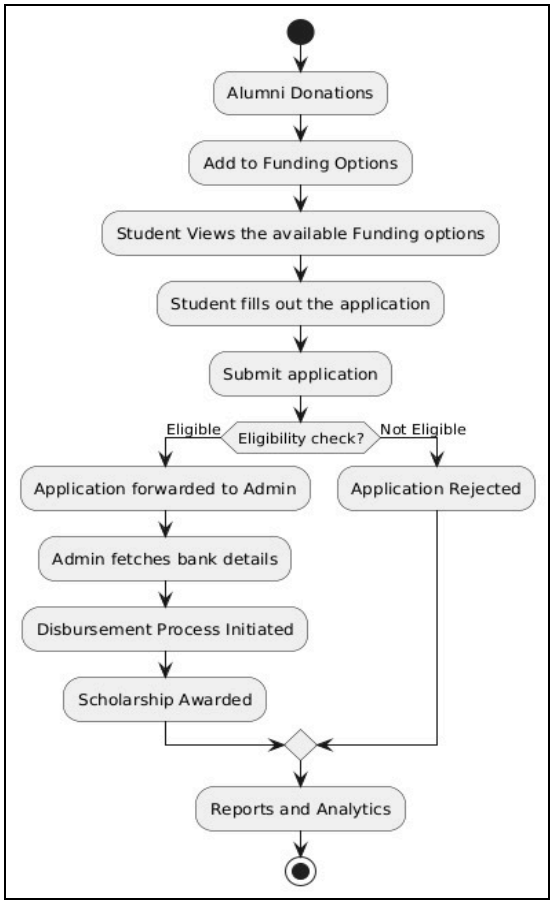
  A scholarship object has statistical records of a particular scholarship, but if the scholarship is removed, the stats can still be used in reports. Scholarship_stats1 object can exist independently.

- **Composition (Black Diamond):** Represents strong ownership. Indicates composition, meaning the lifetime of the contained object depends on the container.
  Example: Application1 ◆—— Status1

  This means that the Status1 object is entirely dependent on the existence of the Application1 object.
  If the Application1 object is deleted, then there is no point of the existence of the Status1 object.

3. Activity Diagram

Activity diagrams depict the flow of tasks between various components of a system. Here, it summarises the entire process from alumni donating to scholarships, students applying, eligibility being checked, funds being disbursed, and finally, reports being generated.

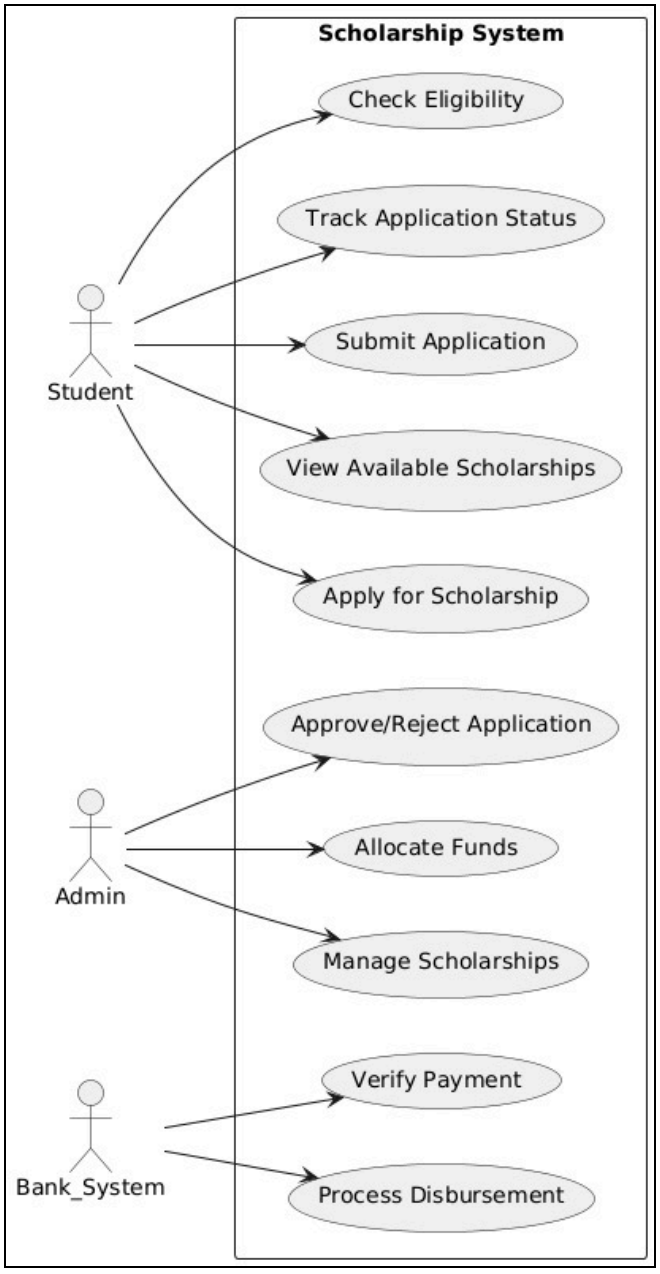


Activities

1. **Alumni Donations**: The first activity is alumni making donations to fund scholarships.
2. **Add to Funding Options**: These donations are processed and added to the available funding options in the system.
3. **Student Views Available Funding Options**: Students can browse and see what scholarship opportunities are available.
4. **Student Fills Out the Application**: Interested students complete application forms for their chosen scholarships.
5. **Submit Application**: Students formally submit their completed applications.
6. **Eligibility Check**: A decision node determines if the student meets the eligibility criteria:
If "Eligible" → Application is forwarded to Admin
If "Not" → Application is rejected
7. **Application Forwarded to Admin**: Eligible applications proceed to administrator review.
8. **Admin Fetches Bank Details**: For approved applications, administrators retrieve the student's banking information.
9. **Disbursement Process Initiated**: The system begins the process of transferring funds.
10. **Scholarship Awarded**: The scholarship is officially granted to the student.
11. **Reports and Analytics**: Both approved and rejected applications feed into a merge node (diamond shape), which leads to generating reports and analytics.

4. Use-Case Diagram

Use case diagrams show the interaction between users and the system, in particular, the steps of tasks that users perform. Here, it depicts the interaction of the three actors: student, admin and bank.



Actors and Their Use Cases

1. Student

- **View Available Scholarships**: Browse funding opportunities
- **Apply for Scholarship**: Submit applications for specific scholarships
- **Submit Application**: Complete and formally enter applications into the system
- **Check Eligibility**: Verify if they are eligible for the scholarship
- **Track Application Status**: Monitor where their application is.

2. Admin

- **Approve/Reject Application**: Review applications and make decisions
- **Allocate Funds**: Assign scholarship money to approved students
- **Manage Scholarships**: Create, modify, and overview scholarship programs

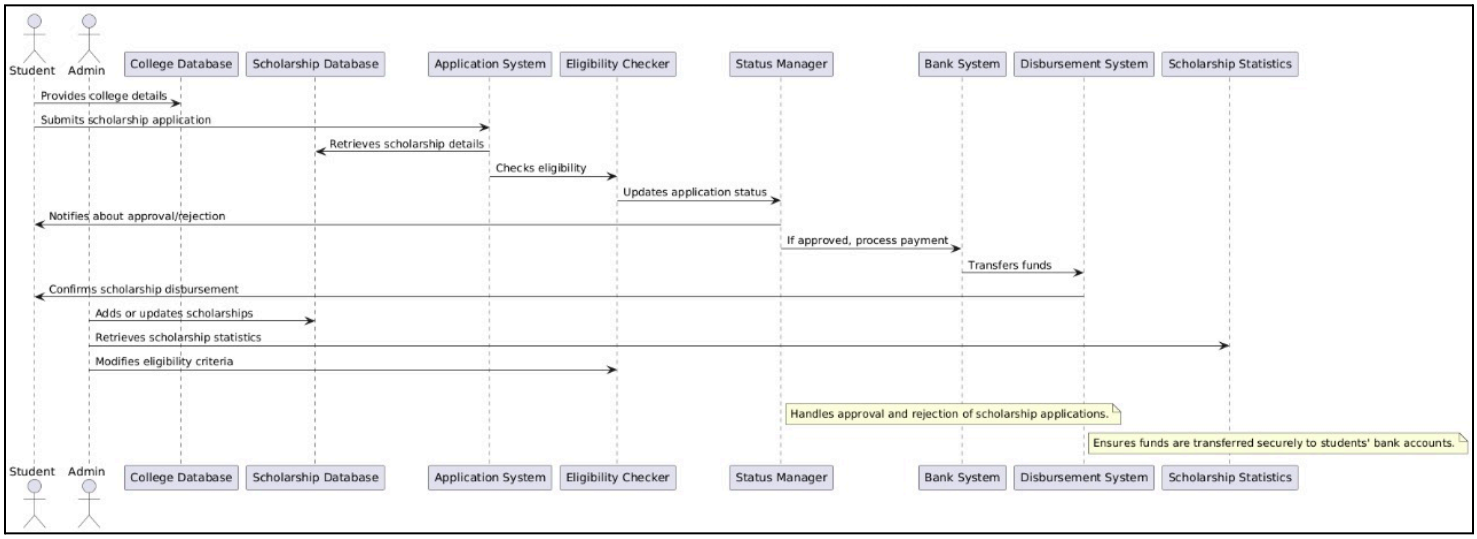
3. Bank_System

- **Verify Payment**: Confirm payment details are valid
- **Process Disbursement**: Handle the actual transfer of funds to students

System Boundary: The rectangle labeled "Scholarship System" defines the boundary of the software system, distinguishing between:

- Internal functionality (use cases within the boundary)
- External entities that interact with the system (actors outside the boundary

5. Sequence Diagram



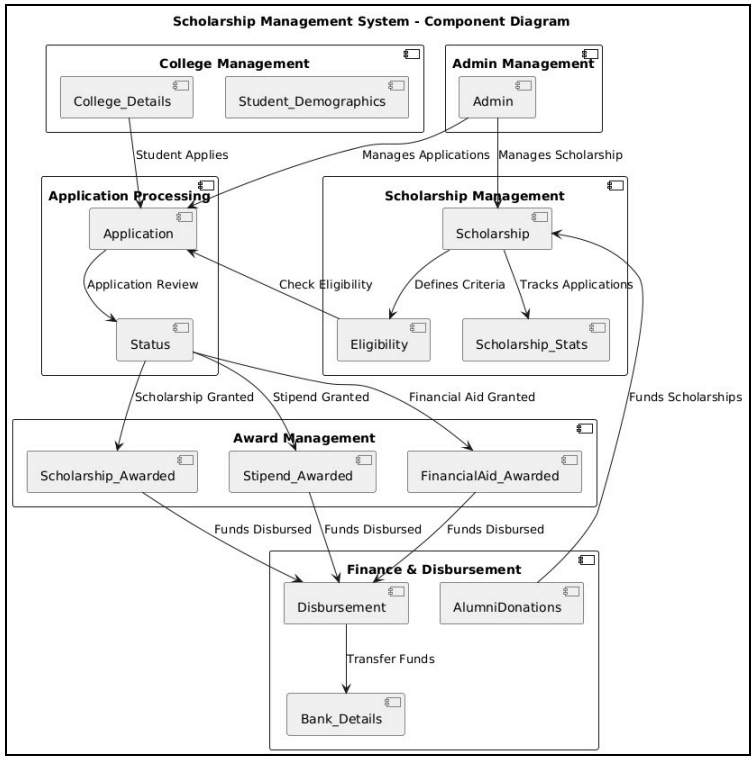
Sequence Diagrams are interaction diagrams that detail how operations are carried out. It shows the sequence of interactions between objects over time. Here the entire operational flow involving the major entities has been depicted. Here, the student and admin are the actors.

Entities & Relationships:

1. **Students** submit scholarship applications.
2. **Admin** manages scholarships, college details, and retrieves statistics.
3. **Application System** processes applications and interacts with other components.
4. **Eligibility Checker** verifies if the student meets the scholarship criteria.
5. **Status Manager** updates the application approval or rejection.
6. **The Bank System** processes payments for approved scholarships.
7. **Disbursement System** securely transfers funds to student bank accounts.
8. **Scholarship Database** stores and retrieves scholarship-related data.
9. **College Database** maintains student academic details.
10. **Scholarship Statistics** stores historical data on scholarship distribution.

6. Component diagram

A component diagram shows how different parts (components) of the system interact with each other and how they are connected. In this case, our Scholarship/Stipend Management System has several key components.



Key Features of Component Diagrams:

1. **Components:** Represent modular, replaceable parts of a system.
Ex: **Application Processing Component**
Handles submission and processing of applications
Updates the application status

2. **Interfaces:** Define how components interact using provided and required interfaces.

3. **Dependencies:** Show how components rely on each other (e.g., a web app depending on a database).

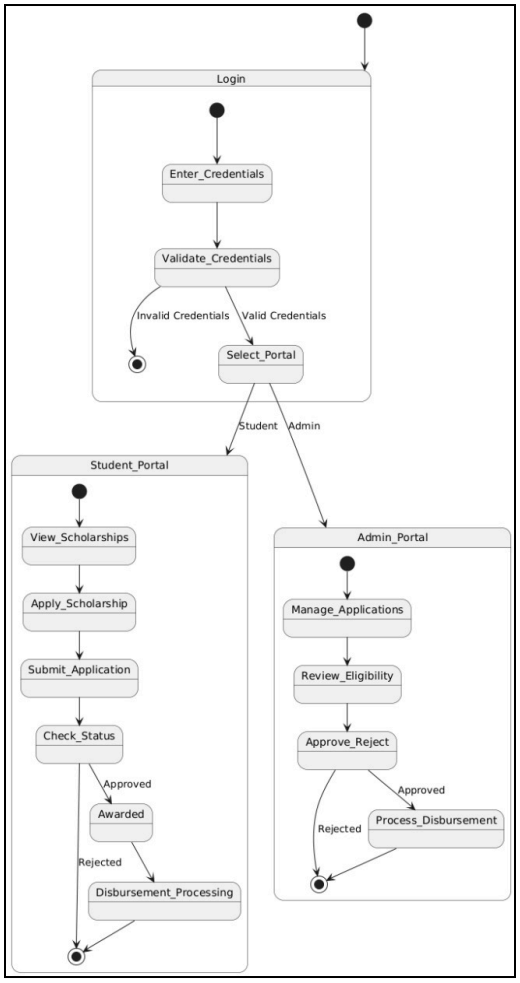
   Example: Application Processing → Status Management

   Once an application is submitted, the status component tracks updates.

4. **Relationships between components:**
   College Details is linked to Application because each student applies for a scholarship, and their details are necessary for application processing.

7. Statechart Diagram

This describes how an object's state changes in response to external events. A rectangle with rounded corners indicates the current nature of an object.



Entities Identified: Student, Admin, Application

Relationships Derived:

1) Students interact with the Student Portal and apply for scholarships.

2)Admin manages applications and verifies approval.

3)Applications transition through different states (submitted, approved, rejected, processed).

4) Money gets disbursed.

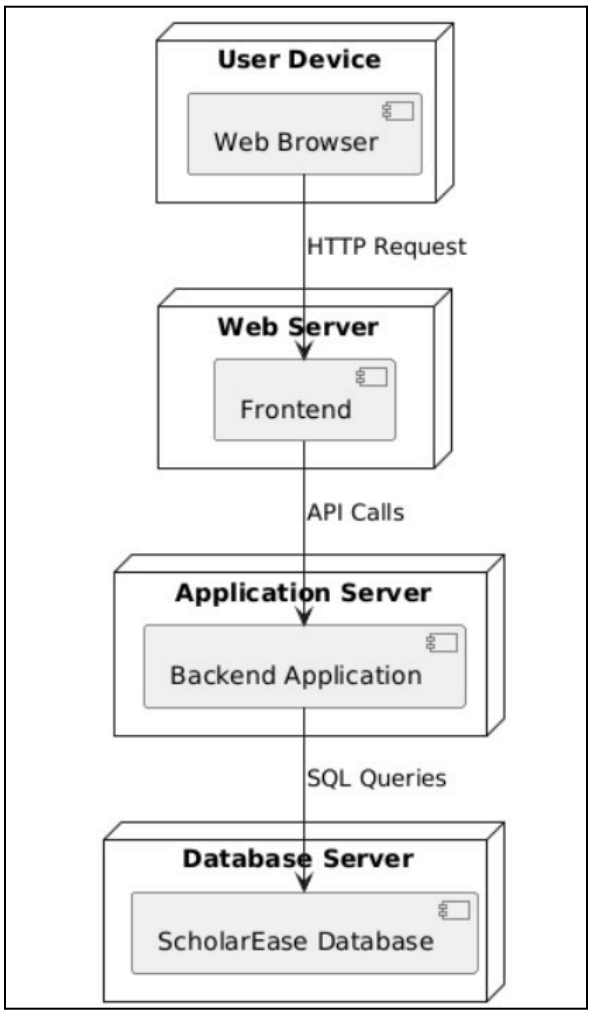
◉ : A circle with a dot in it indicates that a process is terminated

Arrows: Show transitions between states.

Rectangles (rounded): Represent different states

8. Deployment Diagram

This UML deployment diagram illustrates the physical architecture of the scholarship management system, showing how the software components are distributed across hardware nodes.



Deployment Nodes and Components:

1. **User Device**

Contains the Web Browser component.
Represents the client-side of the application.

2. **Web Server**

Contains the Frontend component.
Serves HTML, CSS, and JavaScript to the user's browser.

3. **Application Server**

Contains the Backend Application component.
Implements the classes and relationships defined in the class diagram.

4. **Database Server**

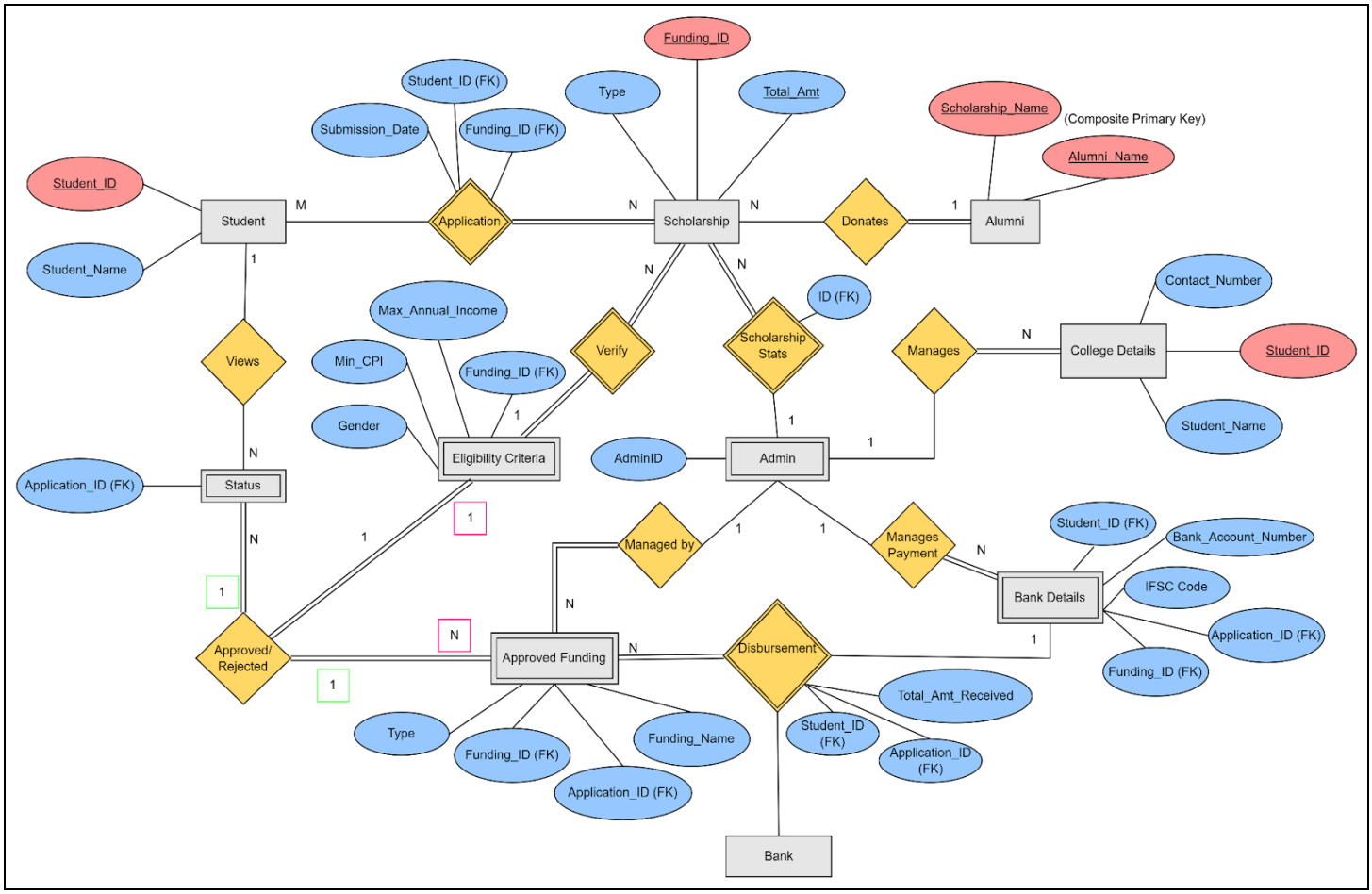
Contains the ScholarEase Database component and stores all persistent data for the system.

Communication Pathways

1. **HTTP Request** - Web browsers communicate with the server using standard HTTP protocol. Enables users to access the scholarship system interface
2. **API Calls** -The frontend communicates with the backend through defined API endpoints
3. **SQL Queries** -The backend application interacts with the database using SQL. It handles data retrieval, storage, and manipulation.

Entity Relationship (ER) diagram

ER Diagrams are used for database design, representing entities, attributes, relationships, and constraints for structured data storage. They focus on database design, ensuring a clear understanding of data organization and relationships, and helping to model real-world scenarios effectively.



Tool used to draw ER diagram: Draw.io
Link for the same:
https://drive.google.com/file/d/1mFykWuHhsMZr2Q7iBs6CD9T4WXuA3rDZ/view?usp=sharing

Note

From the database made for the module 1 submission, we have made a few changes:

- Integrated the three tables Scholarship_Awarded, Stipend_Awarded, and FinancialAid_Awarded into a single entity Approved Funding
- Removed the primary key constraint from tables Admin, Eligibility, Approved Funding, Bank Details, Status, Application, and Disbursement as we realized that they could become weak entities in the ER model.

Notations used
Regular Rectangle - Strong Entity
Doubly lined rectangle - Weak Entity
Regular Diamonds - Strong relationships
Doubly lined diamonds - Weak relationships
Red Ovals (also underlined) - Primary key attributes
Blue ovals - Foreign key and other major attributes
Double line between components - Total participation
Figures on the lines - Cardinality

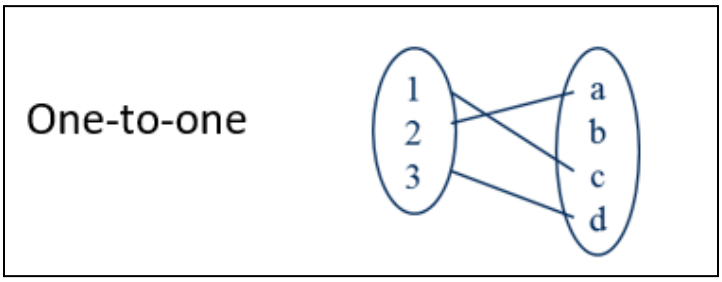
Transition from UML to ER Diagram

The transition from UML to ER is necessary for the following reasons:

1. **Database Design:** ER diagrams give a very structured representation of the database, and we can directly convert ER diagrams easily to a relational database.
2. **Data-Centric Focus:** UML diagrams emphasize system workflows and interactions between different components of the system. However, ER diagrams are entity-based rather than object-based in UML. It is better suited for representing the underlying structure required to support these workflows.
3. **Cardinality and Constraints:** ER diagrams focus significantly on the relationships between entities, showing participation figures. These relationships are derived from interactions shown in UML diagrams.
4. **Understanding the redundancies and loss of information:** In ER diagrams, we know if any data is getting repeated in any relations and if data is getting lost somewhere.

Cardinality of a relationship

One-to-One (1:1) Relationship

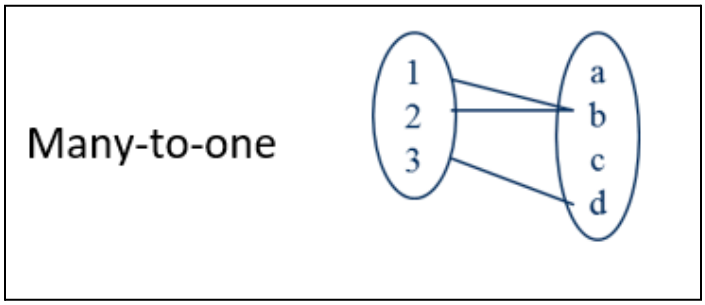


A one-to-one relationship means that a single instance of Entity A is associated with a single instance of Entity B.

Example:

- An Approved Funding entity has a one-to-one relationship with a Status entity, as each status can have only one approved funding, and each approved funding can have only one status.

One-to-Many (1:N) Relationship



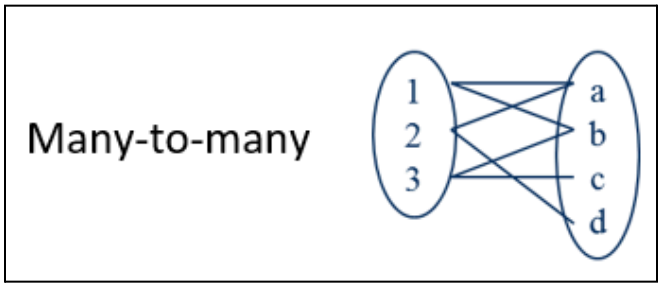
A one-to-many relationship means that a single instance of Entity A is related to multiple instances of Entity B.

Example:

- A Student entity has a one-to-many relationship with a Status entity, as each student can have multiple applications whose status can be viewed, but each status can be viewed by only one student.
- An Admin entity has a one-to-many relationship with an Approved Funding entity, as each approved application is managed by an admin but all admins do not manage all the approved applications (There are total 4 admins so each admin manages a set of scholarships) .
- An Admin entity has a one-to-many relationship with College Details entity, as each student's detail is managed by an admin but all admins do not manage all the student's college details
- An Admin entity has a one-to-many relationship with a Bank Details entity, as each Bank detail is managed by an admin but all admins do not manage all the bank details. .
- An Eligibility Criteria entity has a one-to-many relationship with a Scholarship entity, as each scholarship will have only one eligibility criteria but the same eligibility criteria can apply for multiple scholarships.
- A Scholarship entity has a one-to-many relationship with a Alumni entity, as each Alumni in the database stored can donate multiple scholarships but each scholarship will be donated by only one alumni. (Note: We have considered only alumni who are donating in our database)

- An Approved Funding entity has a one-to-many relationship with a Bank Details entity, as each Bank detail is related to one Approved Funding, but more than one approved funding can have the same bank details (in case a student receives multiple scholarships).
- An Eligibility entity has a one-to-many relationship with a Approved Funding entity, as each eligibility criteria is for one approved funding but more than one approved funding can have the same eligibility criteria.
- An Eligibility entity has a one-to-many relationship with a Status entity, as each eligibility criteria corresponds to one status (i.e the application) but more than one (application) statuses can have the same eligibility criteria.

Many-to-Many (M:N) Relationship



A many-to-many relationship means that multiple instances of Entity A can be related to multiple instances of Entity B.

Example:

- A Student entity has a many-to-many relationship with a Scholarship entity, as each Student can apply for multiple Scholarships, and each Scholarship can have multiple Students.

Additional Constraints or Considerations

In an ER diagram, total participation refers to a constraint where every entity in an entity set must participate in at least one relationship given. This is represented by a double line connecting the entity set to the relationship.

Total Participation (Double Line): Every entity must be involved in at least one relationship instance.
Partial Participation (Single Line): Some entities may not participate in the relationship.

Example:

In our **ER diagram**:

- Entity1: Approved Funding
- Entity2: Admin
- Relationship: Managed by

If we say, "Every Approved Funding must be managed by at least one admin," then the Approved Funding entity has total participation in the Managed by relationship.

Example:

In our **ER diagram**:

- Entity1: Alumni
- Entity2: Scholarship
- Relationship: Donates

If we say, "Every Alumni must donate to at least one scholarship," then the Alumni entity has total participation in the Donates relationship.

Example of Partial Participation:

In our **ER diagram**:

- Entity1: Student
- Entity2: Scholarship
- Relationship: Application

Every student won’t apply for a scholarship, hence the partial participation of the student entity in the application relationship.

Work Contribution

Name	Roll Number	Contributions
Diya Nandan	23110103	Use-Case, Sequence, Activity, Statechart UML diagrams
Ishika Paresh Patel	23110140	Class, Object, Component, Sequence UML diagrams
Maitri Chaudhari	23110192	Class, Object, Component, Sequence UML diagrams
Vardayini Agrawal	23110353	Use-Case, Sequence, Activity, Statechart UML diagrams

All members contributed equally to planning the entire project, making the ER Diagram and the final report.