ScholarEase – Report (Module B)

Team – Data Traffickers

1. Introduction

ScholarEase is a database-based system that helps academic institutions manage scholarships and stipends more easily.

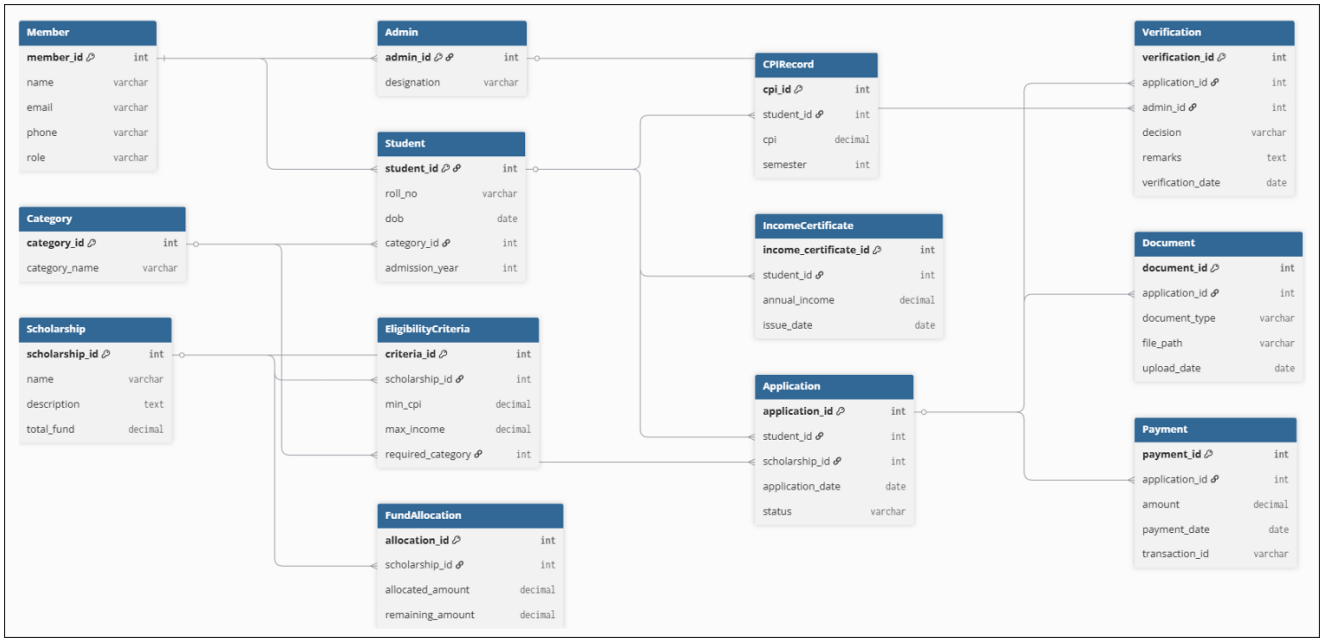
The system keeps all information in one place including student records, scholarship details, eligibility rules, applications, verification steps, payments, and required documents.

This report includes the UML Class Diagram, the ER Diagram, an explanation of how the UML was converted to ER, the reasons for each relationship, and all constraints used in the system, as required for Module B.

2. Table Descriptions

Table Name	Table Description
Member	Stores user accounts for both admins and students.
Student	Stores student-specific data including roll number and category.
Admin	Stores admin staff responsible for verification.
Category	Represents caste/category classification.
Scholarship	Holds scholarship information.
EligibilityCriteria	Defines CPI, income, and category-based rules.
Application	Tracks all scholarship applications.
Verification	Stores admin decision and remarks.
Document	Stores document uploads linked to applications.
Payment	Tracks scholarship disbursements.
FundAllocation	Tracks fund allocated to scholarships.
CPIRecord	Stores semester-wise CPI for students.
IncomeCertificate	Stores annual income information uploaded by students.

3. UML Class Diagram



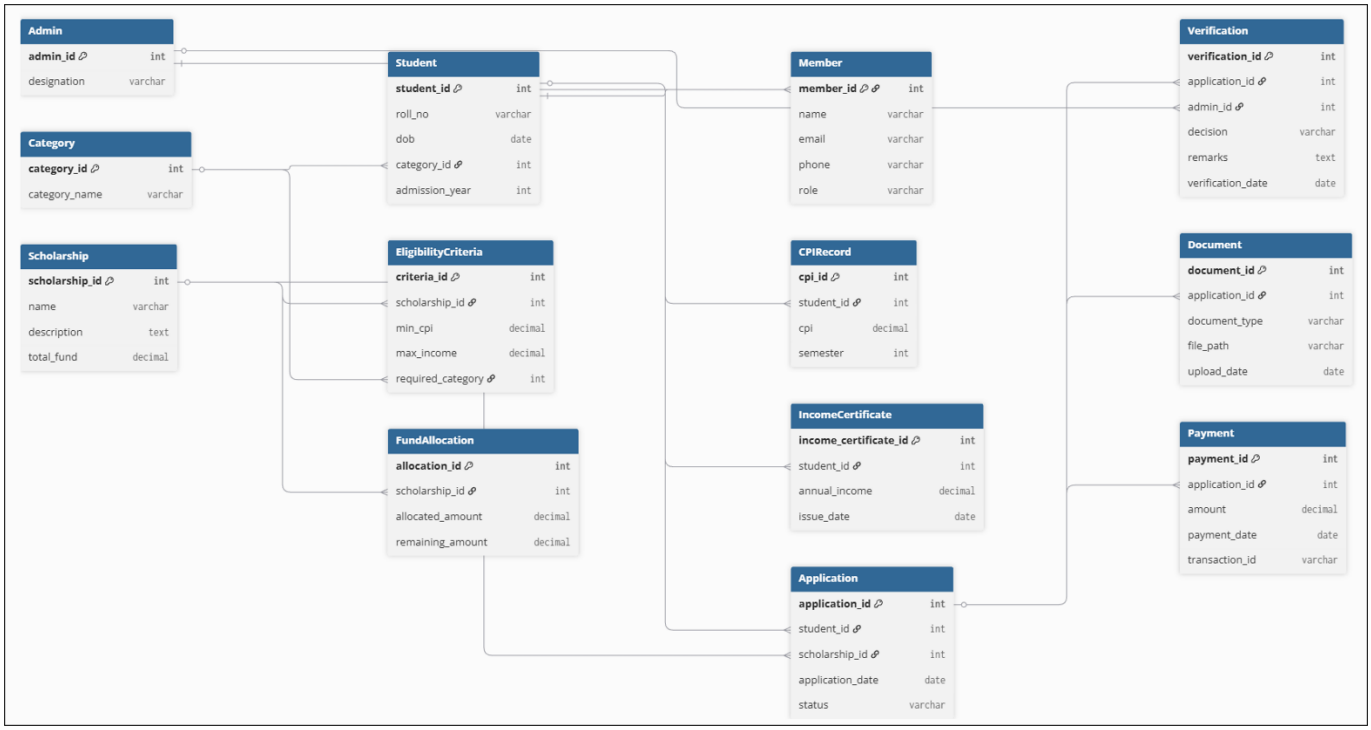
- Primary Keys (PK)
- Foreign Keys (FK)

Multiplicities

1. Member → Student	**1..1 → 0..1** One Member can be linked to zero or one Student (only students have student profiles).
2. Member → Admin	**1..1 → 0..1** One Member can be linked to zero or one Admin (only admins have admin profiles).
3. Category → Student	**1 → * (1..*)** One Category has many Students. One Student belongs to exactly one Category.
4. Student → CPIRecord	**1 → * (1..*)** One Student has many CPI records. Each CPIRecord belongs to exactly one Student.
5. Student → IncomeCertificate	**1 → * (1..*)** One Student can upload many income certificates. Each certificate belongs to exactly one Student.
6. Scholarship → EligibilityCriteria	**1 → * (1..*)** One Scholarship may have multiple eligibility rules. Each rule belongs to exactly one Scholarship.

7. Scholarship → FundAllocation	**1 → 1** One Scholarship has exactly one FundAllocation entry.
8. Student → Application	**1 → * (1..*)** One Student can submit many Applications. Each Application belongs to exactly one Student.
9. Scholarship → Application	**1 → * (1..*)** One Scholarship can receive many Applications. Each Application applies to exactly one Scholarship.
10. Application → Document	**1 → * (1..*)** One Application can have many Documents. Each Document belongs to exactly one Application.
11. Application → Verification	**1 → 1** One Application has exactly one Verification entry. One Verification belongs to one Application.
12. Admin → Verification	**1 → * (1..*)** One Admin can verify many Applications. Each Verification is performed by exactly one Admin.
13. Application → Payment	**1 → * (1..*)** One Application can have many Payments (e.g., installments). Each Payment is attached to exactly one Application.

4. ER Diagram Creation



The ER diagram converts UML classes into database tables.
It shows:

-  Primary Keys (PK)
-  Foreign Keys (FK)
- 1-to-many relationships
- Table attributes
- Structural layout for actual SQL implementation

5. How ER is Derived from UML

Class → Entity

Each UML Class became an ER entity

Associations → Relationships

If UML shows:

Student 1 ---- * Application

ER shows:

- PK: student_id in Student
- FK: student_id in Application

That FK enforces the 1-to-many relationship.

Multiplicity → Cardinality

UML multiplicity tells us how many objects can be linked together in a relationship.

- **1..1 → one**
  Means exactly one item must be present.
- ***1.. → one-to-many****
  Means one item on the left can be connected to many items on the right.
- ***** → many-to-many****
  Means many items on both sides can connect with each other.

In our project, none of the relationships needed many-to-many.
So when converting UML → ER diagram, every relationship became 1-to-many (1:N).

Examples:

- **Scholarship → EligibilityCriteria = 1:N**
  One scholarship can have many eligibility rules.
- **Student → CPIRecord = 1:N**
  One student has many CPI entries (one per semester).
- **Application → Payment = 1:N**
  One application can receive many payments.

6. Relationship Justification

Table Name	Table Description
Student – Category (N:1)	A student belongs to one category (General/OBC/SC/ST), but a category has many students.
Student – Application (1:N)	A student can submit multiple scholarship applications.
Scholarship – EligibilityCriteria (1:N)	A scholarship may have multiple eligibility rules.
Scholarship – FundAllocation (1:1 or 1:N)	One scholarship has one allocation entry (in our design).
Application – Verification (1:1)	Each application is verified once by an admin.
Application – Payment (1:N)	Multiple payments can be linked to a single application (installments, corrections).
Application – Document (1:N)	An application can have many required documents.
Student – CPIRecord (1:N)	Each semester, a student has a new CPI entry.
Student – IncomeCertificate (1:N)	A student can upload multiple income certificates over time.

6. Additional constraints

Primary Keys

Every table has a unique PK:

- student_id
- scholarship_id
- verification_id
- application_id
- criteria_id
- etc.

Foreign Keys

- application.student_id → Student
- eligibilitycriteria.scholarship_id → Scholarship
- verification.admin_id → Admin
- payment.application_id → Application

NOT NULL Constraints

All essential columns (name, date, status, etc.) are NOT NULL.

UNIQUE Constraints

- email (Member)
- transaction_id (Payment)
- roll_no (Student)
- category_name (Category)

CHECK Constraints

- CPI must be between 0 and 10
- Income must be non-negative

Referential Integrity

ON DELETE and ON UPDATE rules maintain database consistency.

Example:

Application → Student

FOREIGN KEY (student_id) REFERENCES Student(student_id) ON DELETE CASCADE

Meaning If a student is deleted then all their applications are deleted automatically.

8. Team Member Contributions

Name	Roll No.	Contribution
Vansh Kumar	23110351	Entire Project
Aditya Jain	23110016	
Satchit Velankar	23110291	
Yash Yeole	23110367	

9. Conclusion

This report presents the complete conceptual design of the ScholarEase system.

The UML and ER diagrams clearly show the main tables, their relationships, and all required constraints.

The design fully follows the Module A and Module B requirements and provides a strong foundation for building the ScholarEase database.