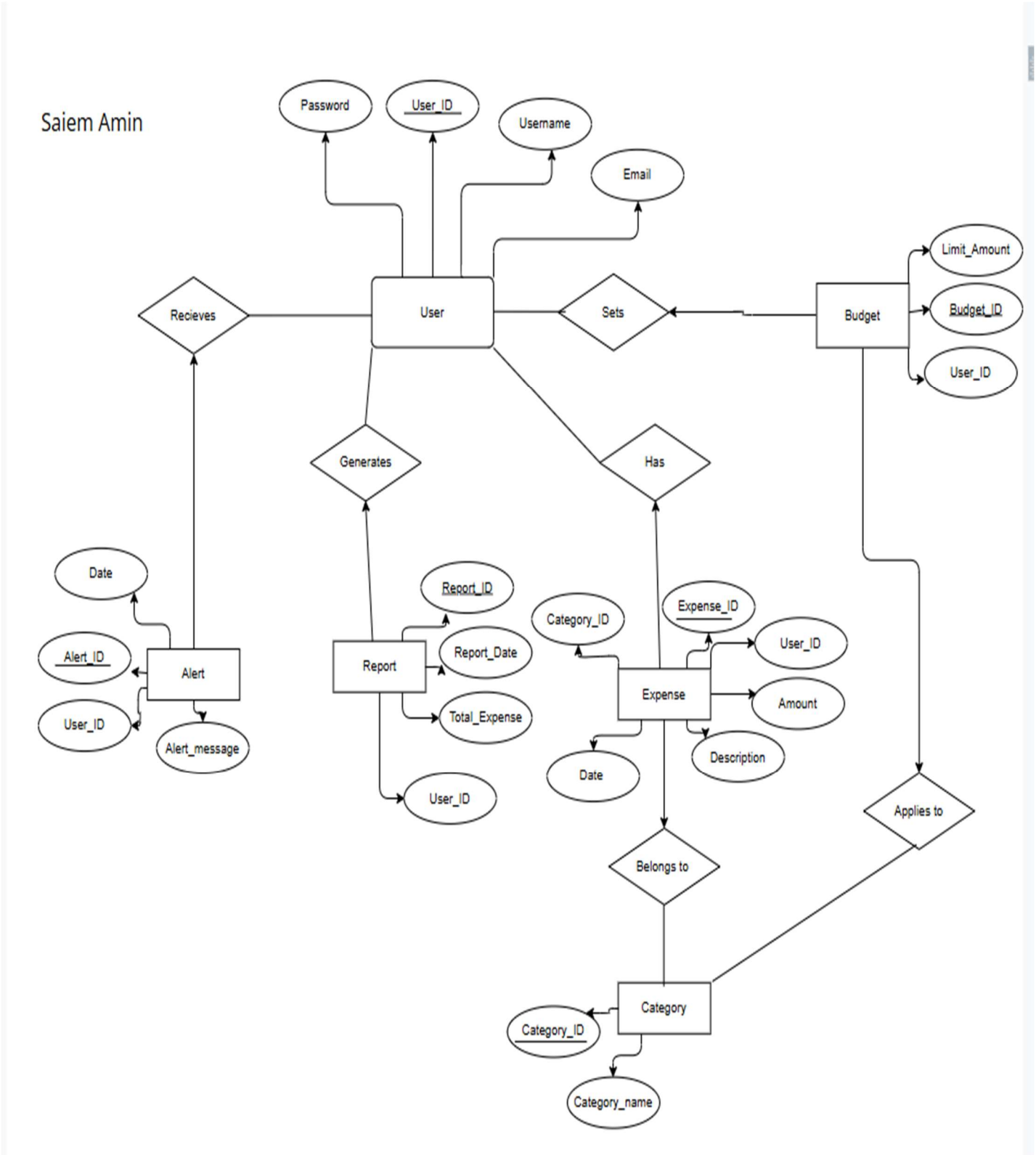
Stage II: Database Design

1. ER-Modeling Procedures

- **Entities:**
    1. **User:** Manages user information
    2. **Category:** Categorizes each expense for user (e.g., groceries, entertainment)
    3. **Expense:** expense details for each user
    4. **Budget:** Stores budget limits for each user
    5. **Alert:** Tracks alerts when a user spending exceeds a budget.
    6. **Report:** Summarizes monthly expenses by category.

- **Relationships:**
    1. **User – Expense:** One user can have multiple expenses **(1:M)**
    2. **User - Budget:** One user can set multiple budgets **(1:M)**
    3. **Expense - Category:** Each expense belongs to one category **(M:1)**
    4. **Budget - Category:** Each budget is associated with a specific category **(M:1).**
    5. **User - Alert:** One user can receive multiple alerts **(1:M)**.
    6. **User - Report:** One user can have multiple monthly reports. **(1:M).**

2. ER-Diagram:


Saiem Amin
Password
User_ID
Username
Email
Recieves
User
Sets
Budget
Limit_Amount
Budget_ID
User_ID
Generates
Has
Date
Alert_ID
Alert
User_ID
Alert_message
Report_ID
Report
Report_Date
Total_Expense
User_ID
Category_ID
Expense_ID
User_ID
Expense
Amount
Date
Description
Applies to
Belongs to
Category
Category_ID
Category_name

3. Translation to relational tables, written in DDL:

```
CREATE TABLE User (
  User_ID INT PRIMARY KEY,
  Username VARCHAR(50) NOT NULL,
  Email VARCHAR(100) UNIQUE,
  Password VARCHAR(50) NOT NULL
);
  CREATE TABLE Category (
  Category_ID INT PRIMARY KEY,
  Category_Name VARCHAR(50) UNIQUE
);

  CREATE TABLE Expense (
  Expense_ID INT PRIMARY KEY,
  Amount DECIMAL(10, 2) NOT NULL,
  Date DATE NOT NULL,
  Description TEXT,
  User_ID INT,
  Category_ID INT,
  FOREIGN KEY (User_ID) REFERENCES User(User_ID),
  FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)
);
  CREATE TABLE Budget (
  Budget_ID INT PRIMARY KEY,
  Limit_Amount DECIMAL(10, 2) NOT NULL,
  User_ID INT,
  Category_ID INT,
```

```
    FOREIGN KEY (User_ID) REFERENCES User(User_ID),
    FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)
);

    CREATE TABLE Alert (
    Alert_ID INT PRIMARY KEY,
    Alert_Message TEXT,
    Date DATE NOT NULL,
    User_ID INT,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

    CREATE TABLE Report (
    Report_ID INT PRIMARY KEY,
    Report_Date DATE NOT NULL,
    Total_Expense DECIMAL(10, 2),
    User_ID INT,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);
```

4. Schema refinement to BCNF:

- **User Table:**
  Attributes: User_ID (PK), Username, Email, Password
  Functional Dependencies: User_ID → Username, Email, Password

  User_ID is a primary key and a superkey. All non-key attributes depend only on User_ID; therefore, the table is **BCNF.**

- **Category Table:**
  Attributes: Category_ID (PK), Category_Name
  Functional Dependencies: Category_ID → Category_Name

  Category_ID is a primary key and a superkey. All non-key attributes depend only on Category_ID; therefore, making the table **BCNF.**

- **Expense Table:**
  Attributes: Expense_ID (PK), Amount, Date, Description, User_ID (FK), Category_ID (FK)
  Functional Dependencies: Expense_ID → Amount, Date, Description, User_ID, Category_ID.

  Expense_ID is a primary key and a superkey. All non-key attributes depend only on Expense_ID; therefore, making the table **BCNF.**

- **Budget Table:**
  Attributes: Budget_ID (PK), Limit_Amount, User_ID (FK), Category_ID (FK)
  Functional Dependencies: Budget_ID → Limit_Amount, User_ID, Category_ID

  Budget_ID is a primary key and a superkey. All non-key attributes depend only on Budget_ID; therefore, making the table **BCNF.**

- **Alert Table:**
  Attributes: Alert_ID (PK), Alert_Message, Date, User_ID (FK)
  Functional Dependencies: Alert_ID → Alert_Message, Date, User_ID

  Alert_ID is a primary key and a superkey. All non-key attributes depend only on Alert_ID; therefore, making the table **BCNF.**

- **Report Table:**
  Attributes: Report_ID (PK), Report_Date, Total_Expense, User_ID (FK)
  Functional Dependencies: Report_ID → Report_Date, Total_Expense, User_ID

  Report_ID is a primary key and a superkey. All non-key attributes depend only on Report_ID; therefore, making the table **BCNF.**

  As you can see all tables are satisfying the BCNF form. There are no transitive or partial dependencies.

5. Supporting SQL queries for all proposed application functionalities:

- **Insert a new User:**

```
INSERT INTO User (User_ID, Username, Email, Password)
VALUES (1, 'john_doe', 'saiem@psu.edu, 'password123');
```

- **Log a new Expense:**

```
INSERT INTO Expense (Expense_ID, Amount, Date, Description,
User_ID, Category_ID)
VALUES (1, 50.00, '2024-11-01', 'Groceries', 1, 1);
```

- **Updating an expense record:**

```
UPDATE Expense
SET Amount = 55.00
WHERE Expense_ID = 1;
```

- **Setting a budget limit for a Category:**

```
INSERT INTO Budget (Budget_ID, Limit_Amount, User_ID,
Category_id)
VALUES (1, 200.00, 1, 1);
```

- **Searching for a particular expense:**

```
SELECT * FROM Expense
WHERE Description LIKE '%grocery%' AND User_ID = ‘1234’;
```

- **Generating a monthly report:**

```
SELECT Category_name, SUM(AMOUNT)  AS TOTAL_AMOUNT
FROM EXPENSE
JOIN Category ON  Expense.Category_ID = Category.Category_ID
WHERE Expense.user_ID = ‘1234’
        AND DATE BETWEEN ‘10/8/24’ AND ‘11/8/24’
GROUP BY Category.Category_name
```

- **Transaction Queries that involves a rollback:**

1. START TRANSACTION;
   UPDATE Expense
   SET Amount = 100.00
   WHERE Expense_ID = 1;
   ROLLBACK;

2. START TRANSACTION;

   INSERT INTO Expense (Expense_ID, Amount, Date, Description, User_ID, Category_ID)
   VALUES (2, 500.00, '2024-11-10', 'Electronics Purchase', 1, 1);

   ROLLBACK;

- **Retrieving the top 3 highest Expenses**

  SELECT Expense_ID, Amount, Date, Description
  FROM EXPENSE
  WHERE User_ID = 1
  ORDER BY AMOUNT DESC
  LIMIT 3;

- **User Updating their email**

  UPDATE
  SET email = ‘new_email@gmail.com’
  WHERE User_id = 1;