ISTANBUL HEALTH AND TECHNOLOGY UNIVERSITY

Faculty of Engineering and Natural Sciences

Department of Computer Engineering

Database Management Project REPORT 1

Project Title: Campus Second-Hand Marketplace

Group Members:

AYŞENUR OTARAN 220611034

BÜŞRA DEMİREL 220611029

ŞEYMA AKIN 220611012

ÜMMÜGÜLSÜN TÜRKMEN 230611056

Table of Contents

1. Project Overview

The Campus Second-Hand Marketplace project is a campus-exclusive second-hand marketplace platform designed for university students and staff. This platform allows users to list items they no longer need, such as textbooks, course notes, or dormitory equipment. The system supports bidding on items and secure messaging between buyers and sellers.

The platform supports at least three distinct user roles:

- Student: Can create listings, place bids, and message other users
- Moderator: Approves listings and handles disputes
- Admin: Manages system categories and users

All user interactions are governed by a secure login/logout mechanism to ensure data integrity and user authentication.

2. Entity-Relationship (E-R) Diagram

The E-R diagram is presented above as an artifact. The diagram uses the standard notation taught in class:

Notation Explanation:

- **Rectangle**: Represents Entities
- **Ellipse**: Represents Attributes
- **Underlined Ellipse**: Represents Primary Keys
- **Diamond**: Represents Relationships
- **Lines**: Show connections between entities and attributes or relationships
- Labels '1', 'M', and 'N' on the connecting lines indicate the cardinality of the relationship .
- **Double Line**: Indicates Total Participation (mandatory participation)

In this E-R diagram, the standard Chen notation is applied. Entities are represented by rectangles, attributes by ellipses, and relationships by diamonds. The cardinality ratios (1:1, 1:N, M:N) are explicitly indicated by labels ('1', 'M', 'N') placed on the connecting lines.

For example, in the 'User Has_Role Role' relationship, the label **'1'** is placed on the Role side and **'N'** on the User side (indicating a 1:N relationship). Similarly, in the 'User Creates Product_Listing' relationship, **'1'** is on the User side and **'N'**is on the Product_Listing side.

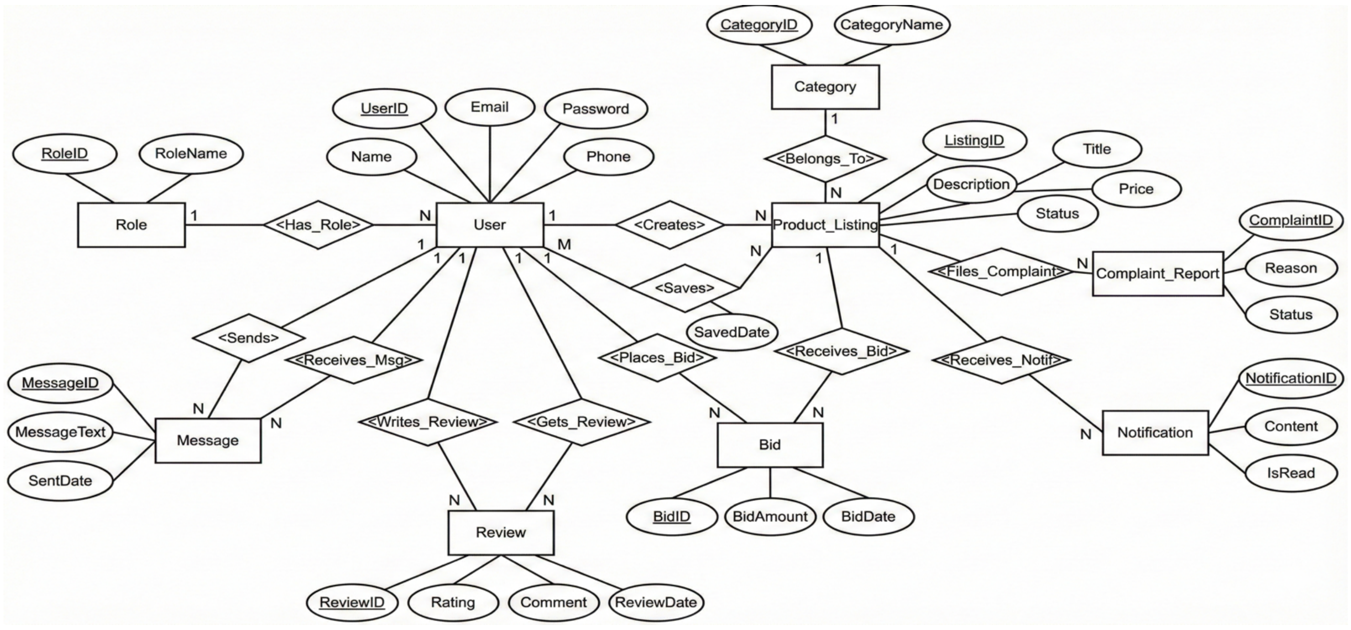

Figure 1: Detailed Entity-Relationship Diagram (Chen Notation)

3. Entities and Their Attributes

3.1 User

Description: The main entity representing all users in the system.

Attributes:

- **UserID** (Primary Key): Unique user identification number (int, AUTO_INCREMENT)
- **Name**: User's full name (varchar(100), NOT NULL)
- **Email**: User's email address (varchar(100), NOT NULL, UNIQUE)
- **Password**: User's encrypted password (varchar(255), NOT NULL)
- **Phone**: User's phone number (varchar(20), NULL)

Relationships: Has_Role, Creates, Places_Bid, Sends, Receives_Msg, Writes_Review, Gets_Review, Files_Complaint, Receives_Notif, Saves

3.2 Role

Description: Entity that defines user roles in the system.

Attributes:

- **RoleID** (Primary Key): Unique role identification number (int, AUTO_INCREMENT)
- **RoleName**: Name of the role (varchar(50), NOT NULL) - "Student", "Moderator", "Admin"

Relationships: Has_Role

3.3 Product_Listing

Description: Represents product listings created by users for sale.

Attributes:

- **ListingID** (Primary Key): Unique listing identification number (int, AUTO_INCREMENT)
- **Title**: Title of the listing (varchar(200), NOT NULL)
- **Description**: Product description (text, NULL)
- **Price**: Product price (decimal(10,2), NOT NULL)
- **Status**: Status of the listing (varchar(20), NOT NULL) - "Active", "Sold", "Pending", "Removed"

Relationships: Creates, Belongs_To, Receives_Bid, Saved_Listing

3.4 Category

Description: Holds category information for classifying product listings.

Attributes:

- **CategoryID** (Primary Key): Unique category identification number (int, AUTO_INCREMENT)
- **CategoryName**: Name of the category (varchar(50), NOT NULL) - "Books", "Electronics", "Furniture", "Dorm Equipment"

Relationships: Belongs_To

3.5 Bid

Description: Represents bids placed by users on product listings.

Attributes:

- **BidID** (Primary Key): Unique bid identification number (int, AUTO_INCREMENT)
- **BidAmount**: The amount of the bid (decimal(10,2), NOT NULL)
- **BidDate**: Date and time when the bid was placed (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Places_Bid, Receives_Bid

3.6 Message

Description: Represents private messages exchanged between users.

Attributes:

- **MessageID** (Primary Key): Unique message identification number (int, AUTO_INCREMENT)
- **MessageText**: Content of the message (text, NOT NULL)
- **SentDate**: Date and time when the message was sent (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Sends, Receives_Msg

3.7 Review

Description: Reviews and ratings that users write about each other.

Attributes:

- **ReviewID** (Primary Key): Unique review identification number (int, AUTO_INCREMENT)
- **Rating**: Rating given (int, NOT NULL, CHECK(Rating BETWEEN 1 AND 5))
- **Comment**: Review comment (text, NULL)
- **ReviewDate**: Date when the review was made (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Writes_Review, Gets_Review

3.8 Complaint_Report

Description: Complaint reports filed by users about inappropriate listings or users.

Attributes:

- **ComplaintID** (Primary Key): Unique complaint identification number (int, AUTO_INCREMENT)
- **Reason**: Reason for the complaint (text, NOT NULL)
- **Status**: Status of the complaint (varchar(20), NOT NULL) - "Pending", "Reviewed", "Resolved"
- **ComplaintDate**: Date when the complaint was filed (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Files_Complaint

3.9 Notification

Description: System notifications sent to users (new bids, messages, status changes).

Attributes:

- **NotificationID** (Primary Key): Unique notification identification number (int, AUTO_INCREMENT)
- **Content**: Notification content (text, NOT NULL)
- **IsRead**: Whether the notification has been read (boolean, NOT NULL, DEFAULT FALSE)
- **CreatedDate**: Date when the notification was created (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Receives_Notif

3.10 Saved_Item

Description: Product listings saved by users for tracking (wishlist). This entity is the junction table for the M:N relationship between User and Product_Listing.

Attributes:

- **SavedItemID** (Primary Key): Unique saved item identification number (int, AUTO_INCREMENT)
- **SavedDate**: Date when the item was saved (datetime, NOT NULL, DEFAULT CURRENT_TIMESTAMP)

Relationships: Saves, Saved_Listing

4. Relationships and Cardinalities

4.1 One-to-One (1:1) Relationships

Relationship 1: User Has_Role Role

Cardinality: 1:1

Explanation: Each user has exactly one role, and each role instance is assigned to exactly one user. In the system, a user cannot be both a Student and a Moderator simultaneously. Each role assignment is unique.

Parent-Child Relationship:

- **Parent:** Role
- **Child:** User

Why this cardinality:

- A user cannot have multiple roles (according to the system's security and authorization logic)
- A role assignment belongs to a single user
- Therefore, the relationship is 1:1

Foreign Key Location: RoleID will be added as a foreign key to the User table.

Participation:

- User entity's participation is **TOTAL** (shown with double line) - Every user must have a role
- Role entity's participation is PARTIAL - A role may not yet be assigned to any user

4.2 One-to-Many (1:N) Relationships

Relationship 2: User Creates Product_Listing

Cardinality: 1:N

Explanation: A user (with Student role) can create multiple product listings, but each listing is created by only one user (seller).

Parent-Child Relationship:

- **Parent:** User (1 side)
- **Child:** Product_Listing (N side)

Why this cardinality:

- A seller (user) can list multiple products (e.g., a student can sell both books and electronics)
- Each product listing belongs to only one seller (a listing cannot be created by multiple sellers)
- Therefore, the relationship is 1:N

Foreign Key Location: SellerID (referencing User.UserID) will be added as a foreign key to the Product_Listing table.

Participation:

- User entity's participation is PARTIAL - A user may not have any listings yet
- Product_Listing entity's participation is **TOTAL** - Every listing must be created by a user

Relationship 3: Category Belongs_To Product_Listing

Cardinality: 1:N (in the direction Category → Product_Listing)

Explanation: A category can contain many product listings, but each listing belongs to only one category.

Parent-Child Relationship:

- **Parent:** Category (1 side)
- **Child:** Product_Listing (N side)

Why this cardinality:

- A category can contain multiple products (e.g., the "Books" category can have dozens of book listings)
- Each product listing belongs to only one category (a product cannot be in both "Books" and "Electronics" categories)
- Therefore, the relationship is 1:N

Foreign Key Location: CategoryID (referencing Category.CategoryID) will be added as a foreign key to the Product_Listing table.

Participation:

- Category entity's participation is PARTIAL - A newly created category may not have any listings yet

- Product_Listing entity's participation is **TOTAL** - Every listing must belong to a category

Relationship 4: User Places_Bid Bid

Cardinality: 1:N

Explanation: A user can place multiple bids (on different products or on the same product at different times), but each bid is placed by only one user (buyer).

Parent-Child Relationship:

- **Parent:** User (1 side)
- **Child:** Bid (N side)

Why this cardinality:

- A buyer can bid on many different products or update bids on the same product multiple times
- Each bid belongs to only one user (a bid cannot be placed by multiple users)
- Therefore, the relationship is 1:N

Foreign Key Location: BuyerID (referencing User.UserID) will be added as a foreign key to the Bid table.

Participation:

- User entity's participation is PARTIAL - A user may not have placed any bids (could be only a seller)
- Bid entity's participation is **TOTAL** - Every bid must be placed by a user

Relationship 5: Product_Listing Receives_Bid Bid

Cardinality: 1:N

Explanation: A product listing can receive multiple bids (many users can place different bids on the same product), but each bid is for only one product listing.

Parent-Child Relationship:

- **Parent:** Product_Listing (1 side)
- **Child:** Bid (N side)

Why this cardinality:

- A product listing can receive bids from many different buyers (a popular product can have dozens of bids)
- Each bid is for only one product listing (a bid cannot be for multiple products)
- Therefore, the relationship is 1:N

Foreign Key Location: ListingID (referencing Product_Listing.ListingID) will be added as a foreign key to the Bid table.

Participation:

- Product_Listing entity's participation is PARTIAL - A listing may not have received any bids yet
- Bid entity's participation is **TOTAL** - Every bid must be for a product listing

Relationship 6: User Sends Message

Cardinality: 1:N

Explanation: A user can send multiple messages, but each message is sent by only one sender.

Parent-Child Relationship:

- **Parent:** User (Sender - 1 side)
- **Child:** Message (N side)

Why this cardinality:

- A user can send messages to many different users
- Each message is sent by only one sender
- Therefore, the relationship is 1:N

Foreign Key Location: SenderID (referencing User.UserID) will be added as a foreign key to the Message table.

Participation:

- User entity's participation is PARTIAL - A user may not have sent any messages
- Message entity's participation is **TOTAL** - Every message must be sent by a sender

Relationship 7: User Receives_Msg Message

Cardinality: 1:N

Explanation: A user can receive multiple messages, but each message is sent to only one receiver.

Parent-Child Relationship:

- **Parent:** User (Receiver - 1 side)
- **Child:** Message (N side)

Why this cardinality:

- A user can receive messages from many different users
- Each message is sent to only one receiver (the system currently does not support group messaging)
- Therefore, the relationship is 1:N

Foreign Key Location: ReceiverID (referencing User.UserID) will be added as a foreign key to the Message table.

Participation:

- User entity's participation is PARTIAL - A user may not have received any messages yet
- Message entity's participation is **TOTAL** - Every message must be sent to a receiver

Relationship 8: User Writes_Review Review

Cardinality: 1:N

Explanation: A user can write multiple reviews (about different users), but each review is written by only one user (reviewer).

Parent-Child Relationship:

- **Parent:** User (Reviewer - 1 side)
- **Child:** Review (N side)

Why this cardinality:

- A user can write reviews about many users after different transactions
- Each review is written by only one user

- Therefore, the relationship is 1:N

Foreign Key Location: ReviewerID (referencing User.UserID) will be added as a foreign key to the Review table.

Participation:

- User entity's participation is PARTIAL - A user may not have written any reviews
- Review entity's participation is **TOTAL** - Every review must be written by a user

Relationship 9: User Gets_Review Review

Cardinality: 1:N

Explanation: Multiple reviews can be written about a user, but each review is about only one user (reviewee).

Parent-Child Relationship:

- **Parent:** User (Reviewee - 1 side)
- **Child:** Review (N side)

Why this cardinality:

- Reviews can be written about a user by many different users
- Each review is about only one user
- Therefore, the relationship is 1:N

Foreign Key Location: RevieweeID (referencing User.UserID) will be added as a foreign key to the Review table.

Participation:

- User entity's participation is PARTIAL - A user may not have received any reviews yet (new user)
- Review entity's participation is **TOTAL** - Every review must be about a user

Relationship 10: User Files_Complaint Complaint_Report

Cardinality: 1:N

Explanation: A user can file multiple complaint reports (about different listings or users), but each complaint report is filed by only one user (reporter).

Parent-Child Relationship:

- **Parent:** User (Reporter - 1 side)
- **Child:** Complaint_Report (N side)

Why this cardinality:

- A user can report many complaints in different situations
- Each complaint is reported by only one user
- Therefore, the relationship is 1:N

Foreign Key Location: ReporterID (referencing User.UserID) will be added as a foreign key to the Complaint_Report table.

Participation:

- User entity's participation is PARTIAL - A user may not have filed any complaints
- Complaint_Report entity's participation is **TOTAL** - Every complaint must be filed by a user

Relationship 11: User Receives_Notif Notification

Cardinality: 1:N

Explanation: A user can receive multiple notifications (for new bids, messages, listing status changes), but each notification is sent to only one user.

Parent-Child Relationship:

- **Parent:** User (1 side)
- **Child:** Notification (N side)

Why this cardinality:

- A user can receive many notifications for different events in the system
- Each notification is specific to one user
- Therefore, the relationship is 1:N

Foreign Key Location: UserID (referencing User.UserID) will be added as a foreign key to the Notification table.

Participation:

- User entity's participation is PARTIAL - A user may not have received any notifications yet

- Notification entity's participation is **TOTAL** - Every notification must be sent to a user

4.3 Many-to-Many (M:N) Relationships

Relationship 12: User Saves Product_Listing (via Saved_Item)

Cardinality: M:N

Explanation: A user can save multiple product listings (as wishlist/favorites), and a product listing can be saved by multiple users. This is a typical M:N relationship.

Why this cardinality:

- A user can add many products to their favorites list
- A product listing can be in many different users' favorites lists
- There is a "many" relationship in both directions, therefore the cardinality is M:N

Junction Table Requirement: This M:N relationship **must have a junction table (bridge table)**. Because:

1. We cannot add ListingID to the User table because a user can have many saved products (violation of 1st Normal Form - cannot write multiple values in one column)
2. We cannot add UserID to the Product_Listing table because a product can be saved by many users (violation of 1st Normal Form)
3. Therefore, we must create a new junction table to represent this M:N relationship

Junction Table: Saved_Item

Junction Table Structure:

- **SavedItemID** (Primary Key): Unique saved item identification number (int, AUTO_INCREMENT)
    - Alternative: Composite primary key (UserID, ListingID) can also be used
- **Foreign Key 1:** UserID → User(UserID) - Indicates which user saved the item
- **Foreign Key 2:** ListingID → Product_Listing(ListingID) - Indicates which product was saved
- **Descriptive Attribute:** SavedDate (datetime) - Indicates when the product was saved (this is the descriptive attribute of this relationship)

Unique Constraint: The pair (UserID, ListingID) must be UNIQUE. A user cannot save the same product multiple times.

Relationship Representation:

- There is a 1:N relationship between User and Saved_Item (User **Saves** Saved_Item)
- There is a 1:N relationship between Product_Listing and Saved_Item (Product_Listing **Saved_Listing** Saved_Item)
- These two 1:N relationships combine to represent the M:N relationship between User and Product_Listing

Participation:

- User entity's participation is PARTIAL - A user may not have any saved products
- Product_Listing entity's participation is PARTIAL - A product may not be saved by any users yet
- Saved_Item entity's participation in both relationships is **TOTAL** - Every saved item record must be connected to both a user and a product listing

5. Participation Constraints

5.1 Total Participation (Mandatory Participation)

The following entities participate in their related relationships with total participation (shown with double lines in the E-R diagram):

1. User → Has_Role: Every user must have a role
2. Product_Listing → Creates: Every listing must be created by a user
3. Product_Listing → Belongs_To: Every listing must belong to a category
4. Bid → Places_Bid: Every bid must be placed by a user
5. Bid → Receives_Bid: Every bid must be for a product listing
6. Message → Sends: Every message must be sent by a sender
7. Message → Receives_Msg: Every message must be sent to a receiver
8. Review → Writes_Review: Every review must be written by a user
9. Review → Gets_Review: Every review must be about a user
10. Complaint_Report → Files_Complaint: Every complaint must be filed by a user
11. Notification → Receives_Notif: Every notification must be sent to a user
12. Saved_Item → Saves and Saved_Listing: Every saved item record must be connected to both a user and a product

5.2 Partial Participation (Optional Participation)

All other participations are partial (shown with single lines in the E-R diagram).

6. Database Schema

Role(RoleID, RoleName)

User(UserID, Name, Email, Password, Phone, *RoleID*)

- **Foreign Key:** RoleID → Role(RoleID)

Category(CategoryID, CategoryName)

Product_Listing(ListingID, Title, Description, Price, Status, *SellerID*, *CategoryID*)

- **Foreign Key:** SellerID → User(UserID)
- **Foreign Key:** CategoryID → Category(CategoryID)

Bid(**BidID**, BidAmount, BidDate, *BuyerID*, *ListingID*)

- **Foreign Key:** BuyerID → User(UserID)
- **Foreign Key:** ListingID → Product_Listing(ListingID)

Message(MessageID, MessageText, SentDate, *SenderID*, *ReceiverID*)

- **Foreign Key:** SenderID → User(UserID)
- **Foreign Key:** ReceiverID → User(UserID)

Review(ReviewID, Rating, Comment, ReviewDate, *ReviewerID*, *RevieweeID*)

- **Foreign Key:** ReviewerID → User(UserID)
- **Foreign Key:** RevieweeID → User(UserID)
- Complaint_Report(ComplaintID, Reason, Status, ComplaintDate, *ReporterID*)
- **Foreign Key:** ReporterID → User(UserID)
- Notification(NotificationID, Content, IsRead, CreatedDate, *UserID*)
- **Foreign Key:** UserID → User(UserID)
- Saved_Item(SavedItemID, SavedDate, *UserID*, *ListingID*)
- **Foreign Key:** UserID → User(UserID)
- **Foreign Key:** ListingID → Product_Listing(ListingID)
- **UNIQUE Constraint:** (UserID, ListingID)

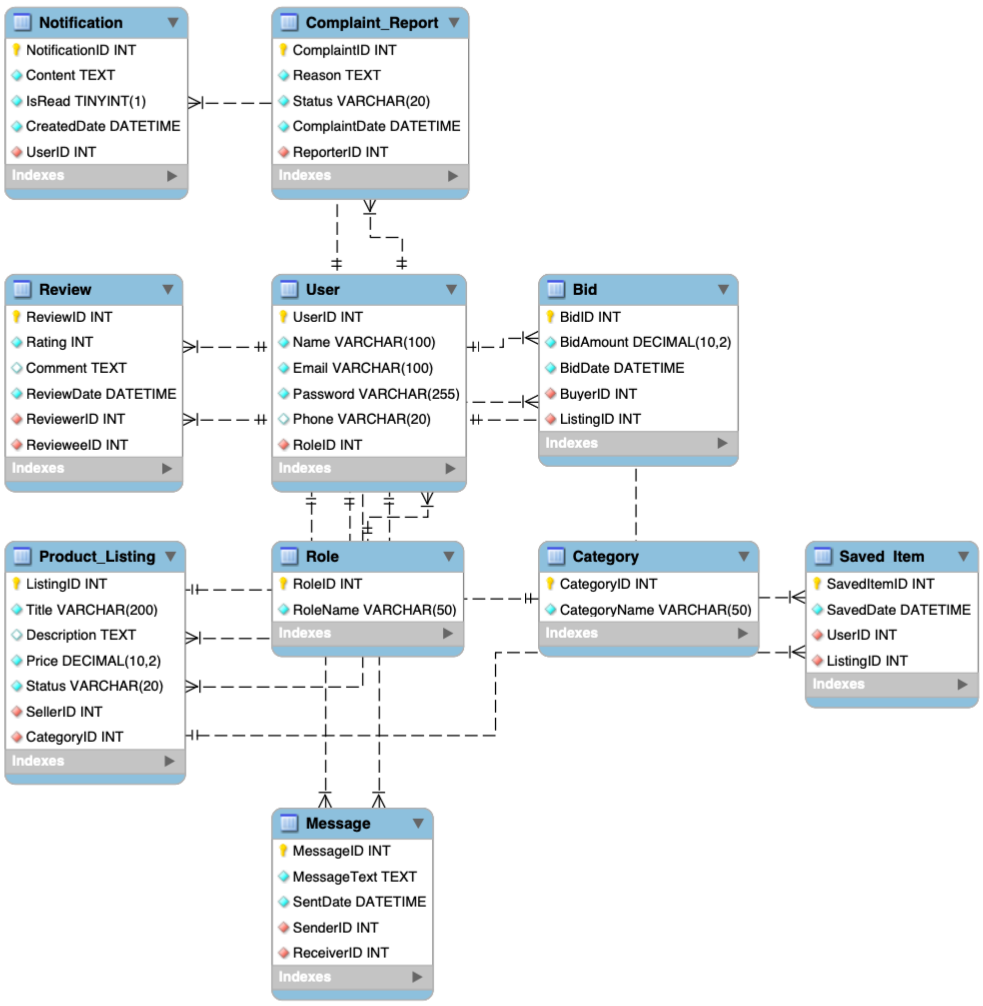

Figure 2: Physical Database Schema in Crow's Foot Notation

Justification for using Crow's Foot Notation: As permitted in the course guidelines, this supplementary diagram uses **Crow's Foot Notation** (Information Engineering style) to visualize the **Physical Data Model**. Unlike the conceptual E-R diagram (Chen notation), this notation is superior for representing the actual database structure because it explicitly displays **Foreign Key placements** and the direction of **1:N relationships** (where the 'crow's foot' symbol denotes the 'Many' side), ensuring a precise mapping of the logic to the actual tables.

7. Conclusion

This report represents the first phase of the Campus Second-Hand Marketplace project. The project requirements have been met as follows:

• Ten entities have been defined (User, Role, Product_Listing, Category, Bid, Message, Review, Complaint_Report, Notification, Saved_Item).
• Twelve relationships have been defined (eleven 1:N, and one M:N).
• The E-R diagram has been drawn using the standard notation taught in class.
• All cardinalities have been explained in detail.
• Parent–child relationships have been determined for each relationship.
• It has been explained whether each relationship requires a new table.
• All primary keys and foreign keys have been defined in detail.
• For each foreign key, the referenced table and column have been clearly stated.
• The database schema has been created.
• A junction table (Saved_Item) has been created for the M:N relationship.

The SQL CREATE statements will be provided as a separate file and are not included in this report.