**GROUP-C**

**BIKE-RENTAL DATABASE MANAGEMENT SYSTEM**

PURPOSE & SCOPE OF OUR SYSTEM (Database Planning and System definition):

1. **Purpose:**

The purpose of our system is to effectively manage the operations of a bike rental business. This will include:

* Facilitating the rental and return of bicycles.
* Tracking the availability and location of all bikes in the inventory.
* Managing customer information, including rental history and payment details.
* Providing reporting for management to monitor business performance.

1. **Main Users:**

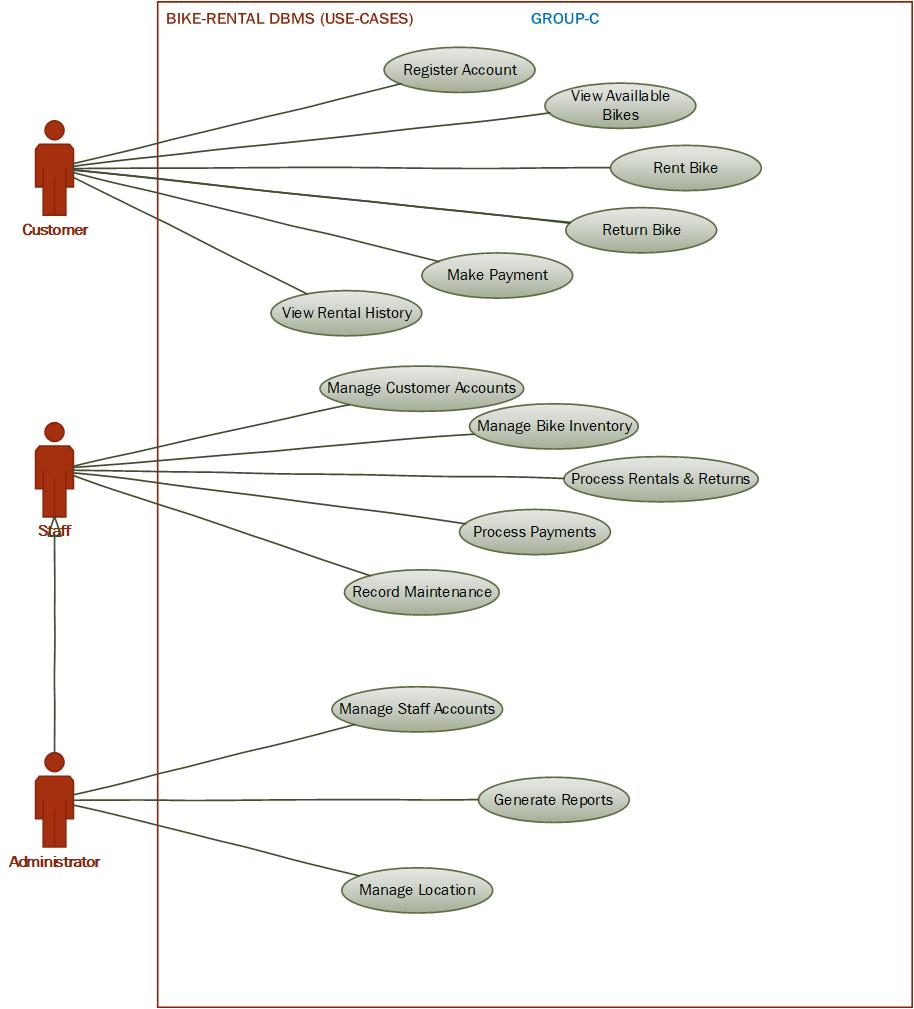
The system will be used by three main groups of people, each with a different set of needs and permissions.

* **Customers:** The end-users who interact with the system primarily to rent bikes, make payments, view availability, and check their rental history. They will need to be able to:
* View available bikes and their locations.
* Register for an account.
* Rent for bikes.
* View their rental history and current rental status.
* Make Payments.
* **Staff:** Employees who work at the rental locations. They will need to be able to:
* Assist customers with rentals and returns.
* Add, update, and remove bike information in the inventory.
* Check bike availability and condition.
* Process payments and issue receipts.
* Manage customer accounts.
* Indicate another staff member if needed to log bike transfers between locations.
* Be available whenever finalizing of a rental process is required.
* **Specific Roles:**
* **Agent/Operator:** Handles core transactional duties like processing rentals, and payments at a location. Also responsible for logging bike transfers between locations. Also assists in customer management tasks like registering new customers and updating their accounts.
* **Mechanic:** Responsible for bike maintenance; performs service, updates the bike’s status in the inventory, and logs repairs in the Maintenance logs.
* **Administrator:** The owner or manager of this business will require:
* Full access to all system data.
* The ability to generate comprehension reports on rentals, revenue, and bike usage.
* Permissions to manage staff accounts and system settings.
* The ability to add or remove rental locations.

1. **Key Functions of the System:**

The system will be able to support the following core processes:

* **Bike Management:**
* Adding new bikes to the inventory.
* Removing bikes that are damaged or sold.
* Updating bike status (e.g., “Available”, “Rented”, Under Maintenance”).
* Tracking the location of each bike.
* **Customer Management:**
* Registering new customers and creating their profiles.
* Updating customer information.
* Storing customer rental history.
* **Rental/Return Process:**
* Taking Record whenever a bike is rented, including the start time, location, and customer.
* Taking Record whenever a bike is returned, including the end time, return location, and final cost.
* Calculating the rental fee based on duration.
* **Payment and Billing:**
* Processing payments for rentals.
* Handling different payment methods (e.g., credit card, mobile money).
* Generating and storing transaction records.
* **Reporting:**
* Generating reports on daily/monthly revenue.
* Tracking the most popular bikes and rental locations.
* Monitoring bike usage to identify high-traffic areas.



**REQUIREMENT GATHERING & ANALYSIS:**

**Entities & Attributes:** Based on our project requirements, we can say that below are our expanded and critically considered list of entities and their attributes for the BIKE-RENTAL DATABASE MANAGEMENT SYSTEM.

1. **Customer:** This entity represents the individuals who rent the bikes.

* **CustomerID (PK, INT):** A unique identifier for each customer.
* **FirstName (VARCHAR (50)):** The customer’s first name.
* **LastName (VARCHAR (50)):** The customer’s Last name.
* **Email (VARCHAR (100), UNIQUE):** The customer’s email address, used for communication and as a unique identifier for user accounts.
* **PhoneNumber (VARCHAR (20)):** The customer’s contact number.
* **DateOfBirth (DATE):** The customer’s date of birth (useful for age-based restrictions or promotions).
* **RegistrationDate (DATETIME):** The date and time the customer registered in the system.
* **PasswordHash (VARCHAR (255)):** A secure, hashed version of the customer’s password.

1. **Bike:** This entity represents each individual bicycle available for rent.

* **BikeID (PK, INT):** A unique identifier for each bike.
* **BikeSerialNumber (VARCHAR (50), UNIQUE):** The unique manufacturer-assigned serial number of the bike.
* **Model (VARCHAR (100)):** The bike’s model name (e.g., “Trek Verve 2”).
* **BikeType (VARCHAR (50)):** The type of the bike (e.g., “Mountain Bike,” “Road Bike,” “Electric Bike”).
* **CurrentStatus (VARCHAR (20)):** The current condition and availability of the bike (e.g., “Available,” “Rented,” “Under Maintenance,” “Retired”).
* **LastMaintenanceDate (DATE):** The date of the last service performed on the bike.
* **RentalRatePerMinute (DECIMAL (5,2)):** The cost to rent this specific bike per minute.
* **LocationID (FK, INT):** The current location of the bike, referencing the **Location** entity.

1. **Rental:** This entity records each instance of a bike being rented by a customer. This is a core transactional entity.

* **RentalID (PK, INT):** A unique identifier for each rental transaction.
* **CustomerID (FK, INT):** The customer who initiated the rental.
* **BikeID (FK, INT):** The bike that was rented.
* **RentalStartDate (DATETIME):** The date and time the rental began.
* **RentalEndDate (DATETIME):** The date and time the rental ended.
* **StartLocationID (FK, INT):** The location where the bike was picked up.
* **EndLocationID (FK, INT):** The location where the bike was returned.
* **TotalCost (DECIMAL (10, 2)):** The final calculated cost of the rental.
* **PaymentStatus (VARCHAR (20)):** The status of the payment (e.g., “Paid,” “Pending,” “Failed”).

1. **Location:** This entity represents the physical stations or hubs where bikes can be rented and returned.

* **LocationID (PK, INT):** A unique identifier for each rental station.
* **LocationName (VARCHAR (100)):** The name of the location (e.g., “Makerere Main Gate Station”).
* **Address (VARCHAR (255)):** The physical address of the location.
* **Longitude (DECIMAL (9, 6)):** The geographical longitude of the location.
* **Capacity (INT):** The total number of the bikes the location can hold.

1. **Staff:** This entity represents the employees who manage the rental locations and operations.

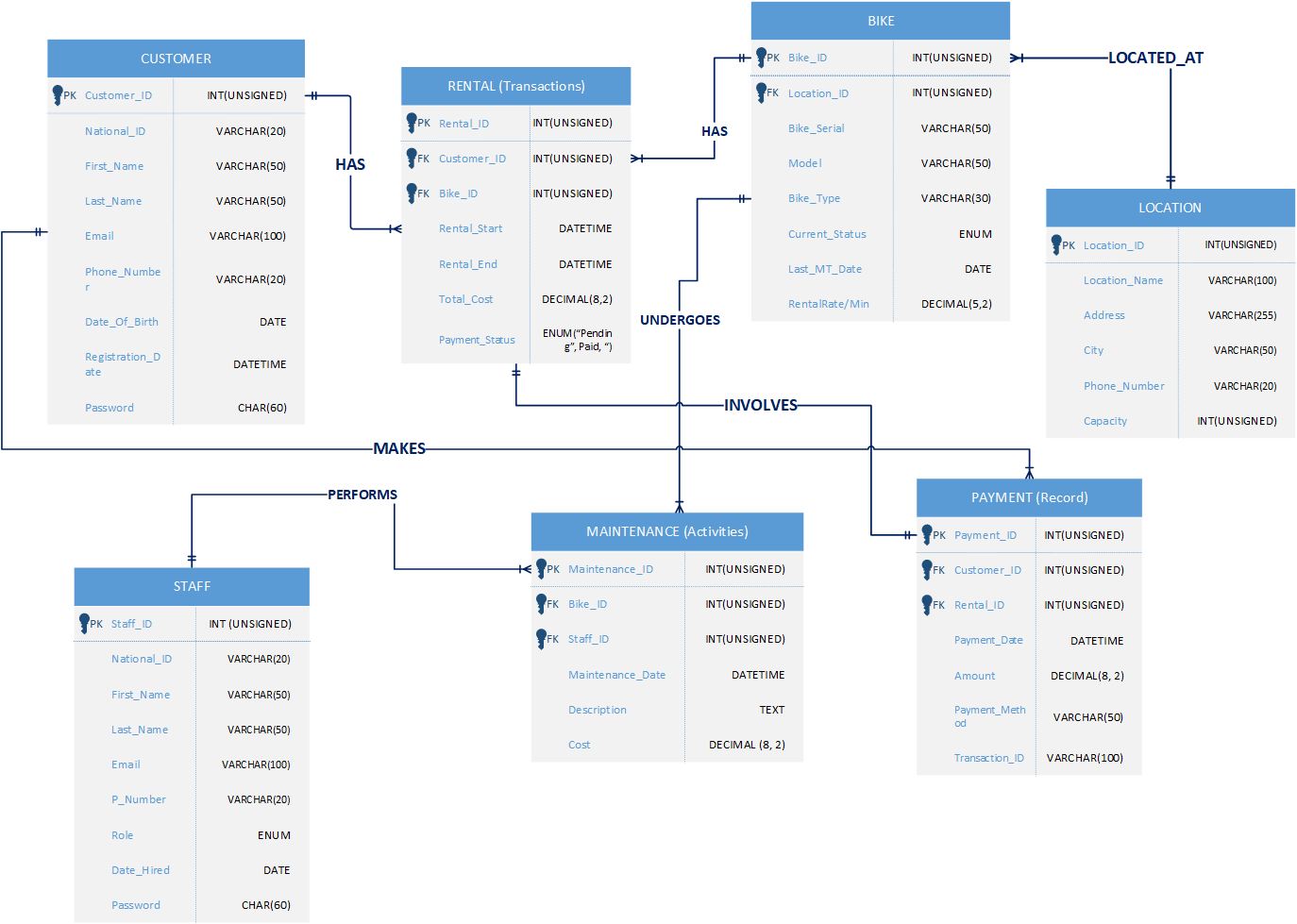
* **StaffID (PK, INT):** A unique identifier for each staff member.
* **FirstName (VARCHAR (50)):** The staff member’s first name.
* **Lastname (VARCHAR (50)):** The staff member’s Last name.
* **Role (VARCHAR (50)):** The role of the staff member (e.g., “Manager,” “Operator,” “Administrator”).
* **LocationID (FK, INT):** The primary location where the staff member works.
* **Email (VARCHAR (100), UNIQUE):** The staff member’s email address.

1. **Payment:** This entity records all payment transactions, providing a detailed history for billing and reporting.

* **PaymentID (PK, INT):** A unique identifier for each payment transaction.
* **RentalID (FK, INT):** The rental transaction this payment corresponds to.
* **Amount (DECIMAL (10, 2)):** The amount paid
* **PaymentDate (DATETIME):** The date and time the payment was made.
* **PaymentMethod (VARCHAR (50)):** The method of payment (e.g., “Credit Card,” “Mobile Money,” “Cash”).
* **TransactionID (VARCHAR (100), UNIQUE):** The unique transaction identifier from the payment gateway.

1. **Maintenance Log:** This entity tracks the maintenance and repair history of each bike.

* **LogID (PK, INT)):** A unique identifier for each maintenance entry.
* **BikeID (FK, INT):** The bike that was serviced.
* **MaintenanceDate (DATETIME):** The date and time of the service.
* **Description (TEXT):** A detailed description of the maintenance performed (e.g., “Replaced front tire,” “Brake adjustment”).
* **Cost (DECIMAL (10, 2)):** The cost of the maintenance.
* **StaffID (FK, INT):** The staff member who performed the maintenance.



Data Dictionary