Pedal – A Cycling Unit

Database Specification : Purpose,
Business Problems Addressed,
Business Rules, Design Decisions

Team 18:
Abhishek Satbhai, Chinmayi Shaligram,
Gayatri Degaonkar, Pavan Sai Pati,
Mohammed Umair Ul Hasan, Krishna Sai Pallela

Database Purpose:
The purpose of this project is to develop a centralized cycling system for the city of Bangalore. It will be used by students, professionals, and other citizens to facilitate functionalities like booking, sharing, and renting bicycles by improving upon the current system thus providing a cheaper mode of transportation.

Business Problems Addressed:
- Provide a centralised system for different users to leverage bicycle renting facility for various purposes at subsidized rates.
- To promote bicycles as a mode of transportation and introduce the eco-friendly way of transportation in the metropolitan area.
- Allow bicycle owners to contribute to the organisation by providing their own bicycle for renting thereby providing them with a way to earn money as well.
- Allow users to track the history of their renting activities and money transactions with organisation.
- Provide users with an additive advantage of getting various discounts applied.
- Make the facility available to users for extended period of time including weekends.
- Provide users with flexibility for start and end stations of their trips.

Business Rules:
- Bookings can be cancelled by the user at any time, by paying a specific cancellation penalty. Booking will be automatically cancelled if a user does not start his trip on time.
- Pre Booking can be done only 3 hours prior to the ride.
- For booking, a customer has to make sure if he has a specific minimum balance in his wallet.
- If trip time goes above 24 hours, users will be charged according to daily rental charges and not hourly rental charges.
- Discount will be applied according to membership type of customer and availability of discount.
- Users need to create an unique Username and Password to connect to the system.
- A User is considered as a premium member when he completes 15 or more rides every month.
- Users cannot choose between user rented bikes and organisation rented bikes.

Design Requirements (Credit to Professor Simon Wang):

- Use Crow's Foot Notation.
- Specify the primary key fields in each table by specifying PK beside the fields.
- Draw a line between the fields of each table to show the relationships between each table. This line should be pointed directly to the fields in each table that are used to form the relationship.
- Specify which table is on the one side of the relationship by placing a one next to the field where the line starts.
- Specify which table is on the many side of the relationship by placing a crow's feet symbol next to the field where the line ends.

Design Decisions:

Sr No.	Entity Name	Entity Description	Relationships With Other Entities
1.	UserMemberships	UserMemberships has information about a user's membership. It works as an associative entity for Users and Membership entities. This holds attributes: UserID, MembershipID, StartDate and EndDate of membership.	In this entity, a composite key (UserID and MembershipID) is used that uniquely identifies a row. UserMemberships has a mandatory one to one relationship with the Users table and many to one relationship with Membership entity.
2.	Users	Users need to register in order to rent a bike from the organisation or supply a bike to the organisation. User entity records all the basic details of the user like FirstName, LastName, Contact number, DOB, etc. It also has attributes like Username and Password which contain credentials for the user's account. Each User is uniquely identified using AadharID which acts like a natural key.	Users entity has mandatory one to one relationships with Address, UserMemberships and Wallet. It has optional one to many relationships with Transactions, Billing, Bookings entities and mandatory one to many relationships with UserGroups.
3.	Membership	The Membership entity is used to record the membership details of each user. Based on the user's membership (Regular / Premium), the discount offered while booking a ride varies. It has attributes like MembershipID and MembershipType.	It has a one to many relationships with UserMemberShips and Discount entities. A user with a particular Membership can avail multiple discounts.

4.	Groups	The Groups entity is used to distinguish between users whether the User is a Owner or Customer. Such information is contained in the attribute GroupName. It has two rows, one for the renters group and other for the owners group.	Groups entity has a one to many relationship with UserGroups entity which is its associative child entity.
5.	UserGroups	The UserGroups entity is an associative entity between User and Groups entities. It has the following attributes : UserID, GroupID, IsActive. To avoid many to many relationship and data redundancy between Users and Groups this entity is created.	UserGroups has two many to one identifying relationships with Users and Groups entities. It has a composite primary key containing UserID, GroupID and an optional one to many relationship with Bikes entity.
6.	Address	Address is common for both Stations and Users entities. So, adding Address entity will reduce the repetition of address attributes in both tables. It contains the attributes AddressID, Area, Pincode, City, State, UserID.	It establishes a many to one and non-identifying relationship with Users entity. It also establishes a one to one non-identifying relationship with Stations entity.
7.	Discount	Discount entity keeps the records of the discounts a user could avail with particular membership per booking. Discount entity has the following attributes : DiscountID, CouponName, Percentage,StartDate, EndDate, IsActive, MembershipID	Discount has many to one relationship with Membership entity. MembershipID is the foreign key here. A user with a particular MembershipType will have zero or multiple discounts available at a given point of time and if multiple discounts are available the user has to choose a single discount according to their choice. Discount entity has one to many relationship with Bookings entity.
8.	Bookings	Bookings entity consists of information about each booking made by the User. It mainly contains attributes like current status of the booking, booking date and ride time.	Bookings entity has a one to one relationship with the Trips entity, it has optional participation in Trips, incase user books a ride and later cancels without starting the trip. It has many to one relationship with Discount and User entities.

9.	Trips	Trips entity records all the information about each trip made by the User. This consists of TripStatus, StartTime and EndTime of trip, BikeStation from which Trip was started and ended as well as TripType i.e trip by customer or by owner.	Trips has mandatory one to one relationship with Bookings. StartBikeStationID and EndBikeStationID are used to record start and end points for each trip and have one to many relationship with Trips.
10.	Transactions	Transactions entity keeps records of users' transactions. Amount of money being credited or debited to/from the user's wallet is recorded along with TransactionStatus for respective users like Successful, Failed. Transaction Type like Debit or Credit.	In Transactions entity, TransactionID is the Unique key used to track details of transactions. TransactionBy plays as a foreign key in this table which establishes a many to one relation with Users.
11.	Billing	Billing table generates an invoice of user's trips. It contains information like status of the bill (Successful, Failed), Total amount according to the time for which the user has booked a bike and Billing type like Debit or Credit.	For each trip there is an associated invoice. TripID helps to get the information of trip to generate a bill. So, Billing has a mandatory one to one relationship with the Trip entity. It has many to one relationship with Users, RentalRates. We get rental rates from RentalRates table using RentalID.
12.	Wallet	Wallet entity is like a customer's bank account, a customer can add money to his wallet anytime, and after he completes his ride, the billing amount will be deducted from his wallet, thus eliminating the complexities involved in traditional payment methods.	Wallet has a mandatory one to one relationship with the Users entity, having TransactionBy as the foreign key referenced from Users entity.
13.	RentalRates	RentalRates holds hourly rates and daily rates for customers and renters.	RentalRates has one to many relationship with Billing entity, that means multiple bills are generated with a specific rental rate.
14.	Stations	Station entity stores the details of stations built for the bike renting	Station has one to many relationship with the BikeStations entity. It has a

		model. It has a StationID to trace each station and an AddressID to trace the address of the station.	mandatory one to one relationship with the address entity where the station address is stored.
15.	Bikes	Bikes entity stores the details of bikes, such as the bike model and serial number. isActive attribute is used to check if the bike is available for the rent or not. CategoryID tells about the type of bike. OwnerID gives details about the owner of the bike.	Bikes entity has a unique ID to trace details of bikes at the bike-station and it has zero to many relationship with the BikeStation .OwnerID is used to trace the information of the owner. So, It establishes many to one relationship with UserGroups. Bikes has many to one relationship with BikeCategory to trace the type of bike.
16.	BikeStations	BikeStation connects the bikes with each station. BikeID and StationID together form a composite primary key. Bikes status (Available/ Not Available) are recorded here with respect to the station, and it also records the time at which bike is available at that station.	BikeStation entity has many to one relationship with bikes and stations entities. It has one to many relationship with Trips entity.
17.	BikeCategory	BikeCategory is used to store the type of bike. i.e., gear bike or non-gear bike.	BikeCategory has one to many relationship with Bikes to trace the type of bike available using BikeCategoryID