Certainly! Here are more detailed explanations of the functional requirements for an online project management system:

1. User Management:

- User registration and authentication: Users should be able to create accounts and log in securely to access the system.

- User roles and permissions management: Differentiate between user roles (e.g., project managers, team members, administrators) and assign appropriate permissions to control access to features and data.

- User profile management: Allow users to update their profiles, including personal information, profile pictures, and contact details.

2. Project Creation and Management:

- Create new projects and assign project details: Users should be able to create projects and provide essential information such as project title, description, start and end dates, and associated stakeholders.

- Assign project managers and team members: Designate project managers responsible for overseeing project activities and assign team members to collaborate on tasks.

- Track project progress and status: Provide a dashboard or overview page to monitor the overall progress and status of each project, including key milestones, task completion, and project timeline.

- Add, update, and delete project tasks: Enable the creation, modification, and deletion of tasks within a project to define the work breakdown structure.

- Set task dependencies and milestones: Allow users to establish task dependencies and define milestones to track progress and ensure sequential execution of tasks.

- Generate reports on project status and progress: Generate project-specific reports, such as task completion rates, resource utilization, and project timeline, to assist in monitoring and decision-making.

3. Task Management:

- Create, update, and delete tasks within a project: Enable the creation, modification, and removal of individual tasks, including task details (title, description, due dates, etc.).

- Assign tasks to team members: Assign tasks to specific team members responsible for their completion.

- Track task status, priority, and deadlines: Track the status of each task (e.g., not started, in progress, completed), set task priorities, and establish deadlines for timely completion.

- Set task dependencies and milestones: Specify dependencies between tasks to ensure proper task sequencing and identify critical path activities. Define milestones to mark significant achievements within the project.

- Assign task resources and estimated effort: Allocate resources (e.g., team members, equipment) to tasks and estimate the effort required for task completion.

- Allow task comments and discussions: Facilitate collaboration by enabling task-specific discussions and comments among team members.

4. Collaboration and Communication:

- Provide real-time collaboration features: Offer real-time collaboration capabilities, such as simultaneous editing of documents or task updates, to enable effective teamwork.

- Support file sharing and document management: Allow users to upload, store, and manage project-related documents and files in a centralized repository accessible to authorized team members.

- Facilitate communication through project-specific discussions or chats: Provide communication channels, such as project discussions or chat rooms, to foster effective communication and information sharing among team members.

- Send notifications and reminders: Send automated notifications and reminders to users for task assignments, upcoming deadlines, or important updates to keep them informed and engaged.

5. Time Tracking and Reporting:

- Allow team members to track time spent on tasks: Enable team members to log and track the time they spend on individual tasks to monitor progress and evaluate productivity.

- Generate reports on task and project-level time tracking: Generate reports summarizing the time spent on tasks or at the project level. These reports can provide insights into resource allocation, identify bottlenecks, and aid in project planning.

- Provide insights into resource utilization and workload distribution: Analyze resource allocation and workload distribution across tasks and projects to ensure optimal resource utilization and balanced work assignments.

6. Document Management:

- Store and manage project-related documents and files

Certainly! Here are some common business entities that could be associated with an online project management system:

1. User: Represents an individual who accesses the system and performs various actions within the project management application. It includes attributes such as username, password, email address, role, and contact information.

2. Project: Represents a specific endeavor with defined goals, timelines, and deliverables. It includes attributes such as project ID, title, description, start and end dates, and associated stakeholders.

3. Task: Represents a unit of work that needs to be completed as part of a project. It includes attributes such as task ID, title, description, status, priority, due date, and assigned team members.

4. Project Manager: Represents a user who is responsible for overseeing and managing a project. Project managers have additional privileges and responsibilities compared to regular team members.

5. Team Member: Represents a user who is part of a project team and is assigned tasks and responsibilities within the project.

6. Milestone: Represents significant achievements or stages within a project. Milestones are used to track progress and mark key points in the project timeline.

7. Comment/Discussion: Represents a communication or discussion related to a project or task. Users can post comments, ask questions, provide updates, and engage in discussions within the project management system.

8. Document/File: Represents project-related files or documents, such as project plans, requirements documents, design specifications, or meeting minutes. Documents can be uploaded, stored, and shared within the project management system.

9. Notification: Represents automated messages or alerts sent to users to notify them of important updates, task assignments, upcoming deadlines, or changes within the project.

These business entities provide a foundation for modeling the data and functionality of an online project management system. They represent the core entities involved in managing projects, tasks, users, and communication within the system.

-- Create the User table

CREATE TABLE User (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL,

role ENUM('Project Manager', 'Team Member') NOT NULL,

-- Additional user attributes

-- ...

);

-- Create the Project table

CREATE TABLE Project (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

description VARCHAR(255),

start\_date DATE,

end\_date DATE,

manager\_id INT,

FOREIGN KEY (manager\_id) REFERENCES User(id),

-- Additional project attributes

-- ...

);

-- Create the Task table

CREATE TABLE Task (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

description VARCHAR(255),

status ENUM('Not Started', 'In Progress', 'Completed') DEFAULT 'Not Started',

priority ENUM('Low', 'Medium', 'High') DEFAULT 'Low',

due\_date DATE,

project\_id INT,

assigned\_to INT,

FOREIGN KEY (project\_id) REFERENCES Project(id),

FOREIGN KEY (assigned\_to) REFERENCES User(id),

-- Additional task attributes

-- ...

);

-- Create the Milestone table

CREATE TABLE Milestone (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

description VARCHAR(255),

due\_date DATE,

project\_id INT,

FOREIGN KEY (project\_id) REFERENCES Project(id),

-- Additional milestone attributes

-- ...

);

-- Create the Comment table

CREATE TABLE Comment (

id INT PRIMARY KEY AUTO\_INCREMENT,

content VARCHAR(255) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

task\_id INT,

user\_id INT,

FOREIGN KEY (task\_id) REFERENCES Task(id),

FOREIGN KEY (user\_id) REFERENCES User(id),

-- Additional comment attributes

-- ...

);

-- Create the Document table

CREATE TABLE Document (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

file\_path VARCHAR(255) NOT NULL,

uploaded\_by INT,

project\_id INT,

FOREIGN KEY (uploaded\_by) REFERENCES User(id),

FOREIGN KEY (project\_id) REFERENCES Project(id),

-- Additional document attributes

-- ...

);

-- Create the Notification table

CREATE TABLE Notification (

id INT PRIMARY KEY AUTO\_INCREMENT,

content VARCHAR(255) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES User(id),

-- Additional notification attributes

-- ...

);

-- Additional tables and relationships can be added based on specific requirements

Functional Requirements for an Employee Recruitment System:

1. User Management:

- User registration and authentication: Users can create accounts and log in securely to access the system.

- User roles and permissions: Differentiate between user roles (e.g., administrators, recruiters, hiring managers) and assign appropriate permissions to control access to features and data.

- User profile management: Allow users to update their profiles, including personal information, contact details, and professional qualifications.

2. Job Posting and Management:

- Job creation: Enable administrators or authorized users to create job postings with details such as job title, description, requirements, and application deadline.

- Job search and filtering: Allow job seekers to search for job openings based on criteria such as location, industry, job type, and keywords.

- Application submission: Provide an interface for candidates to submit their applications, including uploading resumes, cover letters, and other required documents.

- Application tracking: Enable recruiters and hiring managers to track the status of received applications, review candidate profiles, and manage the hiring process.

3. Candidate Management:

- Candidate profiles: Maintain comprehensive profiles for each candidate, including personal details, contact information, work experience, education, skills, and references.

- Candidate search and filtering: Allow recruiters and hiring managers to search and filter candidates based on specific criteria (e.g., skills, experience, qualifications) to identify potential matches for job openings.

- Candidate assessment and evaluation: Provide tools to evaluate and assess candidates, including interview scheduling, interview feedback, and rating systems.

- Communication with candidates: Facilitate communication between recruiters/hiring managers and candidates, including email notifications, interview invitations, and status updates.

4. Interview and Assessment Management:

- Interview scheduling: Enable recruiters and hiring managers to schedule and manage candidate interviews, including date, time, location, and interviewers.

- Assessment tools: Provide assessment features such as online tests, questionnaires, and evaluations to assess candidate skills, knowledge, and suitability for the position.

- Interview feedback and rating: Allow interviewers to provide feedback and ratings for each candidate, which can be used in the decision-making process.

- Interview progress tracking: Track the progress of each candidate through the interview stages, including scheduling, completion, and outcomes.

5. Offer and Onboarding:

- Offer generation: Allow hiring managers to generate job offers for selected candidates, including details such as salary, benefits, start date, and employment terms.

- Offer acceptance and rejection: Capture candidate responses to job offers and update their status accordingly.

- Onboarding process: Support the onboarding process for hired candidates, including the collection of necessary documents, new hire paperwork, and orientation activities.

6. Reporting and Analytics:

- Recruitment analytics: Generate reports and analytics related to the recruitment process, such as time-to-hire, source of candidates, applicant demographics, and recruitment effectiveness.

- Data visualization: Present recruitment data and statistics using charts, graphs, and visual representations to facilitate data-driven decision-making.

- Compliance reporting: Provide reports and documentation to ensure compliance with legal and regulatory requirements, such as Equal Employment Opportunity (EEO) reporting.

7. Integration and APIs:

- Integration with external systems: Allow integration with external systems such as job boards, career websites, and HRIS (Human Resource Information System) for seamless data exchange and synchronization.

- APIs and data access: Provide APIs or data access mechanisms to allow integration with other HR or recruitment-related systems used by the organization.

These functional requirements form the basis for developing an Employee Recruitment System. They define the core features and capabilities required to manage the recruitment process effectively, streamline candidate selection, and facilitate efficient hiring decisions.

-- Create the Candidate table

CREATE TABLE Candidate (

candidate\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone VARCHAR(20),

resume TEXT,

-- Additional candidate attributes

-- ...

);

-- Create the Job table

CREATE TABLE Job (

job\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

description TEXT,

requirements TEXT,

application\_deadline DATE,

-- Additional job attributes

-- ...

);

-- Create the Application table

CREATE TABLE Application (

application\_id INT PRIMARY KEY AUTO\_INCREMENT,

candidate\_id INT,

job\_id INT,

status ENUM('Submitted', 'Under Review', 'Rejected', 'Selected'),

FOREIGN KEY (candidate\_id) REFERENCES Candidate(candidate\_id),

FOREIGN KEY (job\_id) REFERENCES Job(job\_id),

-- Additional application attributes

-- ...

);

-- Create the Interview table

CREATE TABLE Interview (

interview\_id INT PRIMARY KEY AUTO\_INCREMENT,

date DATE,

time TIME,

location VARCHAR(100),

Business Entities for an Employee Recruitment System:

1. Candidate:

- Attributes: Candidate ID, Name, Email, Phone, Resume, etc.

- Responsibilities: Apply for jobs, provide required information, update profile, etc.

2. Job:

- Attributes: Job ID, Title, Description, Requirements, Application Deadline, etc.

- Responsibilities: Create job postings, manage job details, track application status, etc.

3. Application:

- Attributes: Application ID, Candidate ID, Job ID, Status, etc.

- Responsibilities: Submit applications, track application status, update application details, etc.

4. Interview:

- Attributes: Interview ID, Date, Time, Location, Application ID, etc.

- Responsibilities: Schedule interviews, conduct interviews, update interview details, etc.

These business entities represent the main actors or objects involved in the employee recruitment system. Each entity has its own attributes and responsibilities that are essential for managing the recruitment process effectively.

Functional Requirements for Krushi Kendra (Agricultural Center):

1. User Management:

- User registration and authentication: Users can create accounts and log in securely to access the system.

- User roles and permissions: Differentiate between user roles (e.g., farmers, agricultural experts, administrators) and assign appropriate permissions to control access to features and data.

- User profile management: Allow users to update their profiles, including personal information, contact details, and farming preferences.

2. Crop Information and Advisory:

- Crop database: Maintain a comprehensive database of crops with details such as crop types, varieties, cultivation practices, and pest management.

- Crop search and recommendation: Enable farmers to search for crops based on factors like climate, soil type, water availability, and market demand. Provide recommendations for suitable crops based on the farmer's location and preferences.

- Advisory services: Provide agricultural advice and best practices for crop cultivation, irrigation, fertilization, disease prevention, and pest control. Allow experts to share their knowledge and recommendations with farmers.

3. Soil Testing and Analysis:

- Soil testing requests: Allow farmers to request soil testing for their land to determine nutrient levels, pH balance, and other relevant factors.

- Soil sample collection: Provide guidelines for collecting soil samples and facilitate their submission to the testing laboratory.

- Soil analysis reports: Generate detailed soil analysis reports that include recommendations for soil amendments, fertilizers, and other necessary actions to optimize crop productivity.

4. Weather Information:

- Real-time weather data: Integrate with weather APIs to provide accurate and up-to-date weather information, including temperature, rainfall, humidity, and wind speed.

- Weather alerts and forecasts: Send timely alerts to farmers about severe weather conditions, such as heavy rainfall, drought, or storms, to help them make informed decisions and take necessary precautions.

5. Market Information:

- Crop prices: Provide farmers with access to real-time market prices for various crops in different markets. Help farmers make informed decisions regarding crop selection and selling.

- Market trends and analysis: Analyze historical market data to identify trends and patterns. Provide insights into market demand, supply, and price fluctuations to assist farmers in planning their production and marketing strategies.

6. Farmer Community and Collaboration:

- Discussion forums: Facilitate online forums where farmers can interact, ask questions, share experiences, and seek advice from each other and agricultural experts.

- Knowledge sharing: Allow farmers and experts to share articles, success stories, research findings, and other relevant information to foster learning and collaboration within the farming community.

- Events and workshops: Promote and organize agricultural events, workshops, and training programs to enhance farmers' skills and knowledge.

7. Farm Management:

- Farm records and documentation: Enable farmers to maintain digital records of their farm activities, including crop planting dates, fertilization schedules, irrigation logs, and pest control measures.

- Task management: Provide tools for farmers to plan and track their farming tasks, set reminders, and monitor progress.

- Expense tracking: Assist farmers in tracking their farm-related expenses, including input costs, equipment maintenance, labor charges, and other financial aspects.

8. Reports and Analytics:

- Farm performance reports: Generate reports and analytics related to farm productivity, crop yield, resource utilization, and profitability to help farmers assess their performance and make data-driven decisions.

- Crop-specific analytics: Provide insights and analytics for individual crops, such as growth patterns, yield potential, and recommended practices, to assist farmers in optimizing their crop cultivation.

These functional requirements define the core features and capabilities of the Krushi Kendra system. They aim to empower farmers with valuable information, advisory services, and tools to enhance their agricultural practices, improve productivity, and make informed decisions throughout the farming lifecycle.

-- Table: Users

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

role ENUM('Farmer', 'Agricultural Expert', 'Administrator') NOT NULL

);

-- Table: Crops

CREATE TABLE Crops (

crop\_id INT PRIMARY KEY AUTO\_INCREMENT,

crop\_name VARCHAR(100) NOT NULL,

crop\_type VARCHAR(50) NOT NULL,

description TEXT,

-- Additional crop attributes

-- ...

);

-- Table: Farmers

CREATE TABLE Farmers (

farmer\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

contact\_number VARCHAR(20),

address VARCHAR(100),

-- Additional farmer attributes

-- ...

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- Table: Experts

CREATE TABLE Experts (

expert\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

contact\_number VARCHAR(20),

expertise\_area VARCHAR(100),

-- Additional expert attributes

-- ...

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- Table: SoilTests

CREATE TABLE SoilTests (

test\_id INT PRIMARY KEY AUTO\_INCREMENT,

farmer\_id INT NOT NULL,

test\_date DATE,

soil\_type VARCHAR(50),

pH\_level DECIMAL(4, 2),

nutrient\_levels TEXT,

-- Additional soil test attributes

-- ...

FOREIGN KEY (farmer\_id) REFERENCES Farmers(farmer\_id)

);

-- Table: Weather

CREATE TABLE Weather (

weather\_id INT PRIMARY KEY AUTO\_INCREMENT,

location VARCHAR(100),

date DATE,

temperature DECIMAL(5, 2),

rainfall DECIMAL(6, 2),

humidity DECIMAL(5, 2),

wind\_speed DECIMAL(5, 2),

-- Additional weather attributes

-- ...

);

-- Table: MarketPrices

CREATE TABLE MarketPrices (

price\_id INT PRIMARY KEY AUTO\_INCREMENT,

crop\_id INT NOT NULL,

market\_name VARCHAR(100),

price DECIMAL(8, 2),

price\_date DATE,

-- Additional market price attributes

-- ...

FOREIGN KEY (crop\_id) REFERENCES Crops(crop\_id)

);

-- Table: Discussions

CREATE TABLE Discussions (

discussion\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

topic VARCHAR(100) NOT NULL,

message TEXT,

post\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

-- Additional discussion attributes

-- ...

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- Table: Events

CREATE TABLE Events (

event\_id INT PRIMARY KEY AUTO\_INCREMENT,

event\_name VARCHAR(100) NOT NULL,

event\_date DATE,

location VARCHAR(100),

description TEXT,

-- Additional event attributes

-- ...

);

-- Table: FarmRecords

CREATE TABLE FarmRecords (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

farmer\_id INT NOT NULL,

crop\_id INT NOT NULL,

planting\_date DATE,

harvest\_date DATE,

yield DECIMAL(10,

Business Entities for Krushi Kendra:

1. User:

- Attributes: ID, Username, Password, Email, Role, Contact Details, etc.

- Responsibilities: Register and authenticate users, manage user profiles, access system features based on roles, etc.

2. Farmer:

- Attributes: ID, User ID, First Name, Last Name, Contact Details, Address, Farming Preferences, etc.

- Responsibilities: Access crop information and advisory, request soil testing, receive weather information and alerts, participate in the farmer community, manage farm records, etc.

3. Agricultural Expert:

- Attributes: ID, User ID, First Name, Last Name, Contact Details, Expertise Area, etc.

- Responsibilities: Provide crop advisory services, share knowledge and best practices, participate in discussions and forums, conduct workshops and training programs, etc.

4. Crop:

- Attributes: ID, Crop Name, Crop Type, Description, Cultivation Practices, Pest Management, etc.

- Responsibilities: Store and retrieve crop information, provide recommendations, support crop selection, etc.

5. Soil Test:

- Attributes: ID, Farmer ID, Test Date, Soil Type, pH Level, Nutrient Levels, etc.

- Responsibilities: Receive soil testing requests, conduct soil analysis, generate reports, provide recommendations for soil amendments, etc.

6. Weather Information:

- Attributes: ID, Location, Date, Temperature, Rainfall, Humidity, Wind Speed, etc.

- Responsibilities: Retrieve and display real-time weather data, send weather alerts and forecasts, assist in decision-making related to farming activities, etc.

7. Market Information:

- Attributes: ID, Crop ID, Market Name, Price, Price Date, etc.

- Responsibilities: Provide market prices for crops, analyze market trends, assist farmers in making informed decisions about crop selling, etc.

8. Discussion Forum:

- Attributes: ID, User ID, Topic, Message, Post Date, etc.

- Responsibilities: Facilitate discussions among users, enable knowledge sharing, provide a platform for farmers and experts to interact, etc.

These business entities represent the key actors and objects involved in the Krushi Kendra system. Each entity has its own attributes that capture relevant information, and they are responsible for specific actions and interactions within the system to fulfill the functional requirements outlined in the SRS.

Functional Requirements for Online Ticket Booking System:

1. User Registration:

- Users should be able to create new accounts by providing necessary details such as name, email, and password.

2. User Login:

- Registered users should be able to log in to their accounts using their credentials.

3. Search and Browse:

- Users should be able to search for available tickets based on criteria such as destination, date, time, and number of passengers.

- The system should provide a user-friendly interface to browse through the available options and view ticket details.

4. Seat Selection:

- Users should be able to select seats or choose seat preferences during the booking process.

- The system should ensure that seat availability is updated in real-time to avoid double bookings.

5. Booking and Payment:

- Users should be able to book tickets by providing passenger details and making payment using a secure payment gateway.

- The system should generate a booking confirmation and send it to the user via email or SMS.

6. Ticket Cancellation and Refund:

- Users should have the option to cancel their booked tickets within a specified time frame.

- The system should process ticket cancellations and initiate refunds according to the cancellation policy.

7. Ticket Modification:

- Users should be able to modify their ticket details, such as passenger names or travel dates, subject to any applicable restrictions and policies.

8. Booking History and Profile Management:

- Users should have access to their booking history, including past and upcoming trips.

- Users should be able to manage their profile information, update contact details, and modify preferences.

9. Notifications and Alerts:

- The system should send notifications and alerts to users regarding booking confirmation, changes in schedule, or any other relevant updates.

10. Customer Support:

- Users should have access to customer support channels, such as chat, email, or phone, to address their queries or concerns related to bookings.

11. Admin Dashboard:

- An admin dashboard should be provided to manage user accounts, ticket inventory, and handle system configurations.

- Admins should have the ability to add, update, or remove routes, fares, and other relevant information.

12. Reporting and Analytics:

- The system should generate reports and provide analytics on ticket sales, popular routes, user preferences, etc., to help in decision-making and business analysis.

These functional requirements outline the key features and capabilities that an online ticket booking system should provide to its users and administrators.

Business Entities for Online Ticket Booking System:

1. User:

- Attributes: ID, Name, Email, Password, Contact Details, etc.

- Responsibilities: Register and authenticate users, manage user profiles, track booking history, etc.

2. Ticket:

- Attributes: ID, Source, Destination, Date, Time, Seat Number, Fare, etc.

- Responsibilities: Represent individual tickets with their details, manage seat availability, handle ticket bookings, cancellations, and modifications.

3. Payment:

- Attributes: ID, Ticket ID, Amount, Payment Method, Transaction Details, etc.

- Responsibilities: Process ticket payments securely, handle payment verification and confirmation, generate receipts.

4. Route:

- Attributes: ID, Source, Destination, Distance, Duration, etc.

- Responsibilities: Define travel routes, provide information on source and destination locations, calculate distance and duration.

5. Schedule:

- Attributes: ID, Route ID, Departure Time, Arrival Time, Availability, etc.

- Responsibilities: Define departure and arrival times for specific routes, manage seat availability for each schedule.

6. Booking:

- Attributes: ID, User ID, Ticket ID, Booking Date, Status, etc.

- Responsibilities: Capture booking details, associate users with booked tickets, track booking status, handle cancellations and modifications.

7. Notification:

- Attributes: ID, User ID, Message, Timestamp, etc.

- Responsibilities: Send notifications to users regarding booking confirmation, changes in schedule, payment status, etc.

8. Customer Support:

- Attributes: ID, Name, Contact Details, Department, etc.

- Responsibilities: Provide customer support services, handle user inquiries, resolve issues related to bookings.

9. Admin:

- Attributes: ID, Name, Email, Password, Role, etc.

- Responsibilities: Manage the overall system, handle user accounts, configure routes and schedules, generate reports, etc.

These business entities represent the key actors and objects involved in the online ticket booking system. Each entity has its own attributes that capture relevant information and responsibilities that outline their roles and actions within the system.

Here's an example of an SQL file that creates tables for the entities mentioned in the Online Ticket Booking System:

```sql

-- Create User table

CREATE TABLE User (

ID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Email VARCHAR(50) NOT NULL UNIQUE,

Password VARCHAR(50) NOT NULL,

ContactDetails VARCHAR(100)

);

-- Create Ticket table

CREATE TABLE Ticket (

ID INT PRIMARY KEY,

Source VARCHAR(50) NOT NULL,

Destination VARCHAR(50) NOT NULL,

Date DATE NOT NULL,

Time TIME NOT NULL,

SeatNumber INT NOT NULL,

Fare DECIMAL(8, 2) NOT NULL

);

-- Create Payment table

CREATE TABLE Payment (

ID INT PRIMARY KEY,

TicketID INT NOT NULL,

Amount DECIMAL(8, 2) NOT NULL,

PaymentMethod VARCHAR(50) NOT NULL,

TransactionDetails VARCHAR(100)

);

-- Create Route table

CREATE TABLE Route (

ID INT PRIMARY KEY,

Source VARCHAR(50) NOT NULL,

Destination VARCHAR(50) NOT NULL,

Distance INT NOT NULL,

Duration TIME NOT NULL

);

-- Create Schedule table

CREATE TABLE Schedule (

ID INT PRIMARY KEY,

RouteID INT NOT NULL,

DepartureTime TIME NOT NULL,

ArrivalTime TIME NOT NULL,

Availability INT NOT NULL

);

-- Create Booking table

CREATE TABLE Booking (

ID INT PRIMARY KEY,

UserID INT NOT NULL,

TicketID INT NOT NULL,

BookingDate DATE NOT NULL,

Status VARCHAR(50) NOT NULL

);

-- Create Notification table

CREATE TABLE Notification (

ID INT PRIMARY KEY,

UserID INT NOT NULL,

Message VARCHAR(200) NOT NULL,

Timestamp DATETIME NOT NULL

);

-- Create CustomerSupport table

CREATE TABLE CustomerSupport (

ID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

ContactDetails VARCHAR(100),

Department VARCHAR(50)

);

-- Create Admin table

CREATE TABLE Admin (

ID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Email VARCHAR(50) NOT NULL UNIQUE,

Password VARCHAR(50) NOT NULL,

Role VARCHAR(50) NOT NULL

);

-- Add foreign key constraints

ALTER TABLE Ticket

ADD CONSTRAINT FK\_Ticket\_User FOREIGN KEY (UserID) REFERENCES User(ID);

ALTER TABLE Payment

ADD CONSTRAINT FK\_Payment\_Ticket FOREIGN KEY (TicketID) REFERENCES Ticket(ID);

ALTER TABLE Schedule

ADD CONSTRAINT FK\_Schedule\_Route FOREIGN KEY (RouteID) REFERENCES Route(ID);

ALTER TABLE Booking

ADD CONSTRAINT FK\_Booking\_User FOREIGN KEY (UserID) REFERENCES User(ID),

ADD CONSTRAINT FK\_Booking\_Ticket FOREIGN KEY (TicketID) REFERENCES Ticket(ID);

ALTER TABLE Notification

ADD CONSTRAINT FK\_Notification\_User FOREIGN KEY (UserID) REFERENCES User(ID);

```

Please note that this is just a sample SQL file for creating the necessary tables for the Online Ticket Booking System. The actual SQL implementation may vary based on specific requirements, additional constraints, and relationships between entities.