**Database Design (Oracle SQL):**

**Tables:**

Stations: StationID (Primary Key), StationName

Trains: TrainID (Primary Key), TrainName

WeeklySchedules: ScheduleID (Primary Key), TrainID (Foreign Key), StartStationID (Foreign Key), EndStationID (Foreign Key), DepartureTime, ArrivalTime

TicketAvailability: TicketID (Primary Key), TrainID (Foreign Key), ScheduleID (Foreign Key), AvailableSeats, Cost

Users: UserID (Primary Key), Username, Password

**Relationships:**

Trains and Stations have a many-to-many relationship through WeeklySchedules.

WeeklySchedules has foreign keys pointing to Trains, Stations (StartStation and EndStation).

TicketAvailability has foreign keys pointing to Trains and WeeklySchedules.

**Backend Design (Django):**

**Models:**

Define Django models for Stations, Trains, WeeklySchedules, TicketAvailability, and Users. Use Django's ORM to establish relationships.

**Views:**

Create views to handle user requests such as train schedules, ticket availability, and user authentication.

**Controllers (Views in Django):**

Implement controllers to handle user requests, query the database using ORM, and return appropriate responses.

**Authentication:**

Use Django's built-in authentication system for basic user authentication.

**Reports:**

Create Django views or functions to generate small-scale reports based on your requirements, such as the number of passengers in one station or on one train.

**Frontend Design:**

**User Interface:**

Use HTML, CSS, and JavaScript to design a simple and user-friendly interface for users to interact with the system.

**Forms:**

Create forms for adding trains, stations, and users.

**Views:**

Connect frontend views to Django backend views to display information and handle user inputs.

**Authentication:**

Implement basic authentication features in the frontend to allow users to log in and access relevant information.

**Additional Considerations:**

**Database Connectivity:**

Use the cx\_Oracle package for Oracle database connectivity in your Django project.

**Django Admin:**

Leverage Django Admin for easy management of database entries during development.

**Error Handling:**

Implement error handling in both backend and frontend to provide a smooth user experience.

**Testing:**

Perform thorough testing of the system to ensure functionality and identify and fix any potential issues.