REPORT

Travel Agency Booking System Report

1. Introduction

This document provides an overview of the Travel Agency Booking System, covering its purpose, key actors, main features, architectural components (class diagram), database schema, and typical user scenarios. The system enables clients to browse and book pre-defined trip packages and allows agency administrators to manage all aspects of hotels, airlines, contracts, reservations, and reporting.

2. Key Actors & Use Cases

Actors:

- **Client**: A user who registers, logs in, browses trips, makes bookings, and submits feedback.
- **Admin**: Agency staff who log in with administrator privileges to manage partners, agreements, packages, client accounts, and generate business reports.

2.1 Use Case Summary

ID	Actor	Use Case	Details / Sub-tasks
U1	Client	Creates a new account	Includes: Fill registration form
U2	Client, Admin	Logs in to the account	Shared authentication
U3	Client	Browses trips	Includes: Filter by destination, filter by date
U4	Client	Views trip details	—

U5	Client	Books a trip	Includes: Select number of people, confirm booking
U6	Client	Views reservations	—
U7	Client	Submits feedback	Includes: Write review, rate hotel and airline
U8	Client	Updates personal information	Extends: Update name, update address, update phone
A1	Admin	Manages hotels	Includes: Add; extends: Update details, Delete
A2	Admin	Manages airlines	Includes: Add; extends: Update details, Delete
A3	Admin	Manages agreements	Includes: Create agreement; extends: Update terms
A4	Admin	Manages trip packages	—
A5	Admin	Manages client reservations	—
A6	Admin	Generates reports	—
A7	Admin	Manages client accounts	—
A8	Admin	Views client feedback	—
A9	Admin	Manages client categories	—

A detailed use-case diagram illustrates these interactions and their include/extend relationships.

---

3. System Architecture (Class Diagram)

The core domain model comprises the following main classes and associations:

- **User** (abstract base): `userId`, `username`, `passwordHash`, `role` (Client or Admin).
  - **Client** inherits from User.
  - **Admin** inherits from User.

- **Profile** (1:1 with User): Stores `fullName`, `address`, `phone`.
- **ClientCategory**: Allows categorizing clients (e.g., Individual, Corporate).
- **Hotel** and **Airline**: Partner entities with basic attributes.
- **Agreement**: Contracts between the agency (Admin) and hotels/airlines; attributes include `agreementNumber`, dates, `terms`, and `prices`.
- **TripPackage**: Bundles a single hotel and airline under an (optional) agreement, plus trip details like dates, price, duration, amenities, type, and availability.
- **Reservation**: Links a Client to a TripPackage, with booking details (`numberOfPeople`, `status`, `bookingDate`, `travelDate`).
- **Feedback**: Captures post-trip ratings and reviews per Reservation; one feedback per reservation.

Associations reflect foreign-key relationships: 1-to-1 between User and Profile; 1-to-many between Admin→Agreement, Hotel→Agreement, Airline→Agreement; 1-to-many between TripPackage→Reservation; 1-to-many between User (Client)→Reservation and Reservation→Feedback.

---

4. Database Schema

The MySQL schema implements the above classes as tables:

- **User**: Single-table inheritance via `role` ENUM, with `userId` PK.
- **ClientCategory**: `categoryId` PK, `name` unique.
- **Hotel**, **Airline**: Basic partner tables.
- **Profile**: FK to `User(userId)`.
- **Agreement**: FK to `User(adminId)` and optional FKs to `Hotel(hotelId)` and `Airline(airlineId)`.
- **TripPackage**: FKs to `Agreement`, `Hotel`, and `Airline`.
- **Reservation**: FKs to `User(clientId)` and `TripPackage(packageId)`.
- **Feedback**: FK to `Reservation(reservationId)` (unique) and `User(clientId)`.

All tables use InnoDB, with appropriate primary keys, unique constraints, foreign keys, and cascading rules.

---

5. Usage Scenarios

Scenario 1: New Client Registration & Booking

1. Sarah (new user) opens the site and selects **creates a new account** (U1).
2. She **fills registration form** (U1a) and submits; the system creates a User + Profile.
3. She **logs in** (U2) and **browses trips** (U3), applying destination/date filters (U3a/U3b).
4. Clicking a package, she **views trip details** (U4), then **books a trip** (U5), selecting number of people (U5a) and confirming (U5b).
5. Sarah checks **views reservations** (U6) to confirm her booking.

Scenario 2: Client Submits Feedback

1. After travel, Sarah logs in (U2) and navigates to her completed booking under **views reservations** (U6).
2. She selects **submits feedback** (U7), then **writes review** (U7a) and **rates hotel and airline** (U7b).
3. The system stores her feedback and links it to her reservation.

Scenario 3: Admin Manages Partners & Packages

1. Mr. Lee (Admin) logs in (U2) with administrator credentials.
2. He **manages hotels** (A1): **adds a hotel** (A1a), then later **updates hotel details** (A1b).
3. He **manages agreements** (A3): **creates an agreement** (A3a) tying the hotel to agency terms.
4. He **manages trip packages** (A4): creates a new TripPackage linking the hotel, an airline, and an agreement.
5. He **generates reports** (A6) to see top-selling packages and loyal clients.

---

6. Conclusion

This Travel Agency Booking System provides a clear separation of client and administration concerns, a robust relational model, and a straightforward domain architecture. Each use case is directly supported by domain classes and database tables, ensuring traceable requirements coverage and a solid foundation for implementation in a PHP/MySQL/XAMPP environment.