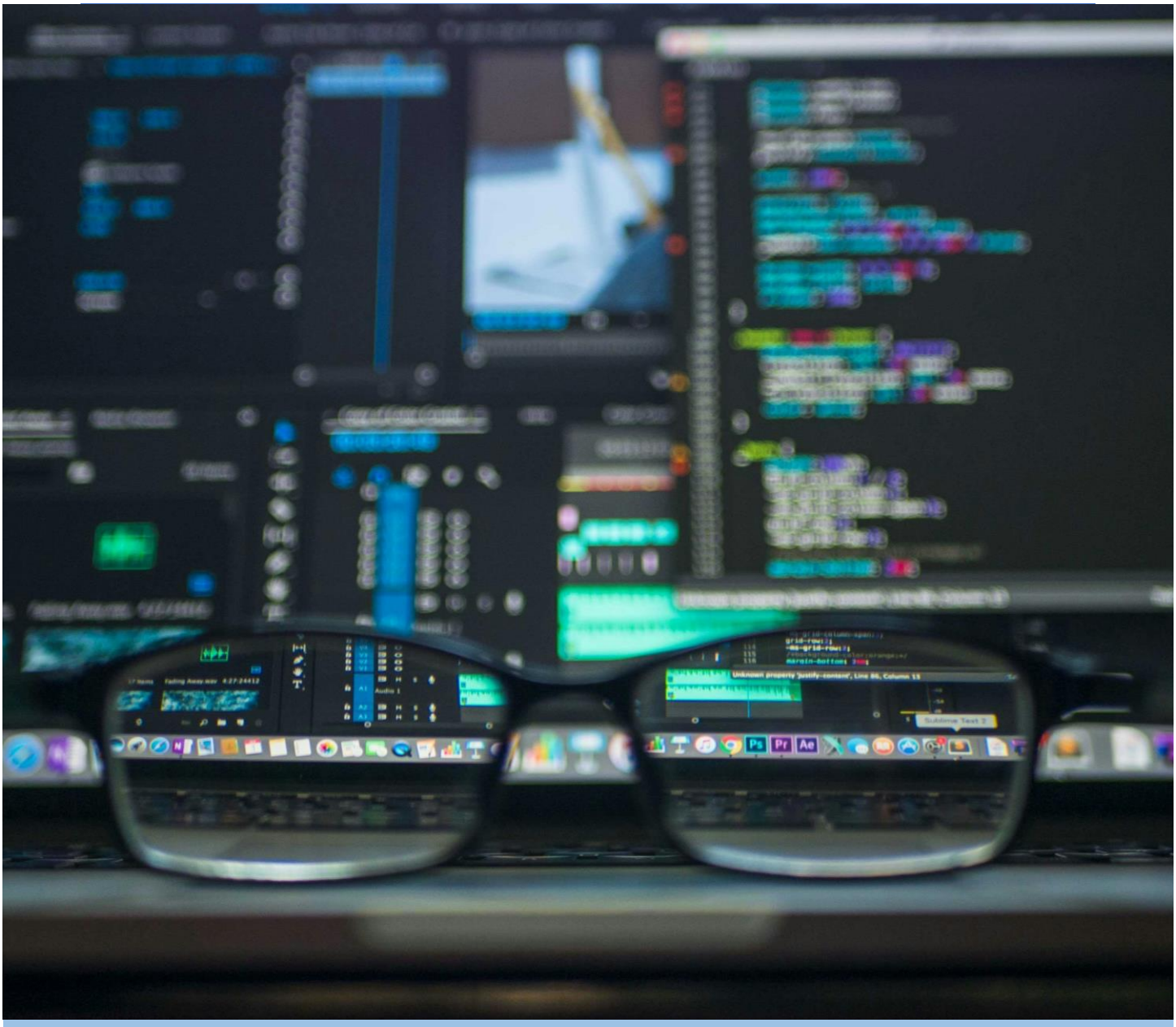


Documentations (For BookSafar.com)

Student 1: Muhammad Agha

Student 2: Jahangeer

December 17, 2025

University Of Sindh Laar Campus Badin

TABEL OF CONTENT

BookSafar

Project's overview

Abstract:

BookSafar is a JSP and Oracle-based web application designed to simplify travel planning and booking. It enables travelers to explore curated tour packages, reserve trips with secure online payments, manage their bookings (including cancellations within a defined time window), and share reviews after completing their journeys. On the administrative side, operators can create and update packages, oversee bookings and revenue, and manage user accounts and feedback.

Purpose:

The purpose of BookSafar is to provide a seamless, end-to-end experience for travelers—from discovering packages and confirming reservations to managing changes and sharing feedback. At the same time, it equips administrators with efficient tools to publish offerings, update details, and monitor overall performance.

Introduction:

Objectives:

- Provide travelers with the ability to:
    - Explore available tour packages
    - Book trips and make secure payments
    - Manage or cancel bookings in line with policy
    - Submit reviews after completing their trips
- Provide administrators with tools to:
    - Publish and update travel packages
    - Monitor bookings and payment records
    - Moderate user reviews
- Maintain a basic audit trail of bookings and payments for each package.

Scope:

The system includes:

- JSP-based front-end pages
- Oracle-backed data management layer
- User and administrator authentication via session handling
- Package catalog browsing
- Booking and payment workflows
- Cancellation management
- Review submission functionality
- Administrative dashboards for managing packages, bookings, users, and reviews

The system does **not** include:

- Integration with external payment gateways
- Mobile applications
- Advanced security features (e.g., hashing, CSRF protection)
- Analytics beyond built-in counts and summaries

Audience:

- **Developers and testers** responsible for maintaining and extending the codebase
- **Administrators and operators** using the admin console to configure and manage the system
- **Stakeholders** requiring a functional understanding of the application's capabilities

---

System Architecture:

Flow:

The application follows a layered flow:

- **Browser → Tomcat JSP Web Tier → JDBC Data Access → Oracle Database**

Components:

- **Presentation & Logic:**
    - JSP pages for public users: browsing packages, booking, payment, and submitting reviews
    - JSP pages for administrators: dashboards to manage packages, bookings, users, and reviews
- **Data Access:**
    - Direct JDBC calls implemented through a shared connection helper defined in *config.jspf*
- **Persistence:**
    - Oracle RDBMS storing all core entities such as packages, bookings, payments, users, admins, and reviews

Technology Stack:

- **Server-side rendering:** Java/JSP
- **Database connectivity:** JDBC
- **Database:** Oracle Database (XE 21c edition)
- **Frontend styling:** HTML/CSS with custom styles (*style.css*, *active-link.css*)
- **Application server:** Tomcat

Database Documentation:

ER Entities & Relationships:

- **Users (1)–(M) Booking**: A user can make multiple bookings.
- **Package (1)–(M) Booking**: Each package can be booked by multiple users.
- **Booking (1)–(1) Payment**: Every booking has one corresponding payment record.
- **Booking (1)–(0..1) Review**: A booking may have zero or one review.
- **Users (1)–(M) Review**: A user can submit multiple reviews.
- **Package (1)–(M) Review**: Each package can have multiple reviews.

- **Admin (1)–(M) Package**: Packages are created by administrators.

Schema Overview (Oracle):

Users { userid, name, email, password, phone, createdate }

Admin { adminid, name, email, password, phone, createdate }

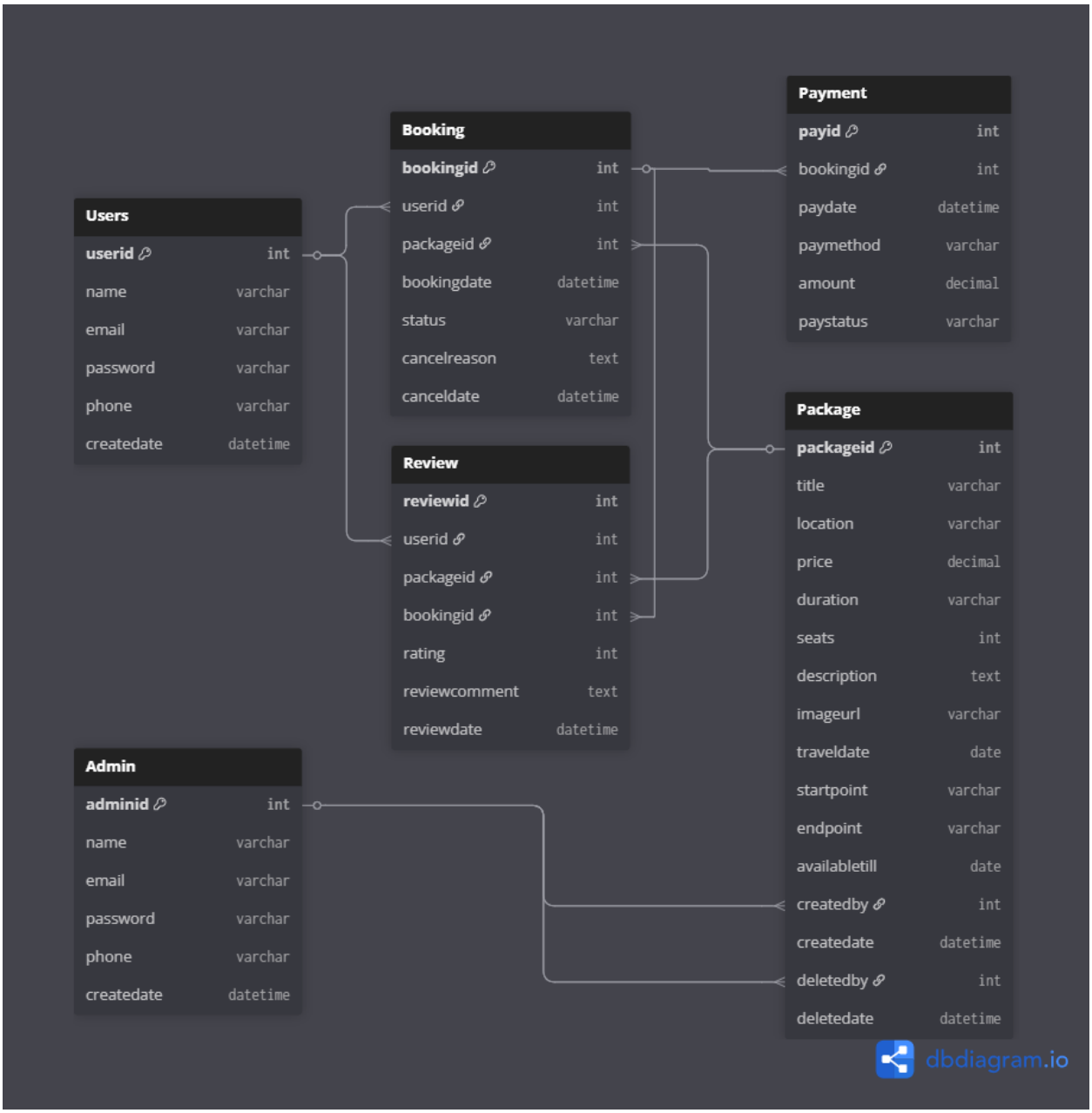
Package { packageid, title, location, price, duration, seats, description, imageurl, traveldate, startpoint, endpoint, availabletill, createdby, createdate, deletedby, deletedate }

Booking { bookingid, userid, packageid, bookingdate, status, cancelreason, canceldate }

Payment { payid, bookingid, paydate, paymethod, amount, paystatus }

Review { reviewid, userid, packageid, bookingid, rating, reviewcomment, reviewdate }

ER Diagram:


Users
userid int
name varchar
email varchar
password varchar
phone varchar
createdate datetime
Booking
bookingid int
userid int
packageid int
bookingdate datetime
status varchar
cancelreason text
canceldate datetime
Payment
payid int
bookingid int
paydate datetime
paymethod varchar
amount decimal
paystatus varchar
Review
reviewid int
userid int
packageid int
bookingid int
rating int
reviewcomment text
reviewdate datetime
Package
packageid int
title varchar
location varchar
price decimal
duration varchar
seats int
description text
imageurl varchar
traveldate date
startpoint varchar
endpoint varchar
availabletill date
createdby int
createdate datetime
deletedby int
deletedate datetime
Admin
adminid int
name varchar
email varchar
password varchar
phone varchar
createdate datetime
dbdiagram.io

Sample Queries (CRUD & Aggregate):

- **Create (Insert User):**

  INSERT INTO Users(name, email, password, phone)

  VALUES ('Ali', 'ali@example.com', 'pass123', '+1-555-1111');

- **Read (Browse Active Packages):**

  SELECT packageid, title, location, price, seats, traveldate, availabletill

  FROM Package

  WHERE deletedate IS NULL

  AND seats > 0

  AND availabletill >= TRUNC(SYSDATE);

- **Update (Soft-Delete Package):**

  UPDATE Package

  SET deletedby = adminId, deletedate = SYSTIMESTAMP

  WHERE packageid = pid

  AND deletedate IS NULL;

- **Update (Cancel Booking Within Policy):**

  UPDATE Booking

  SET status = 'Cancelled', cancelreason = reason, canceldate = SYSTIMESTAMP

  WHERE bookingid = bid AND status = 'Booked';

  UPDATE Payment

  SET paystatus = 'Refunded'

  WHERE bookingid = bid;

- **Update (Auto-Complete After 2 Days):**

  UPDATE Booking

  SET status = 'Completed'

```
WHERE status = 'Booked'
  AND bookingdate < (SYSTIMESTAMP - INTERVAL '2' DAY);
```

- **Delete (Hard-Delete Review):**

```
DELETE FROM Review
WHERE reviewid = rid;
```

- **Aggregate (Admin Revenue):**

```
SELECT NVL(SUM(p.amount), 0)
FROM Payment p
JOIN Booking b ON p.bookingid = b.bookingid
JOIN Package pk ON b.packageid = pk.packageid
WHERE pk.createdby = adminId
  AND p.paystatus = 'Completed';
```

Web Application Documentation:

Modules & Features:

- **User Authentication**
  - Login: *login.jsp*
  - Registration: *register.jsp*
  - Session-based access control
  - Logout: *logout.jsp*
- **Package Discovery**
  - Package listing with filter and sort by price options: *index.jsp*
  - Detailed package view with reviews and booking option: *packageDetails.jsp*
- **Booking & Payment**
  - Booking initiation: *book.jsp*

    - Payment capture and seat decrement: *processPayment.jsp*
    - Booking history with auto-complete aging: *myBookings.jsp*
    - Booking cancellation with refund and seat restoration: *cancelBooking.jsp*
- **Reviews**
    - Post-trip review submission: *addReview.jsp*
    - Review display on package details: *packageDetails.jsp*
    - Admin moderation of reviews: *reviews.jsp*
- **Admin Panel**
    - Authentication: *login.jsp*
    - Dashboard with active package table: *dashboard.jsp*
    - Package management (CRUD and soft-delete): *packages.jsp*, *addPackage.jsp*, *editpackage.jsp*
    - Booking management with auto-complete aging: *bookings.jsp*
    - User list scoped to admin's packages: *users.jsp*
    - Admin creation: *addAdmin.jsp*
    - Logout: *logout.jsp*

Use Cases & Flows:

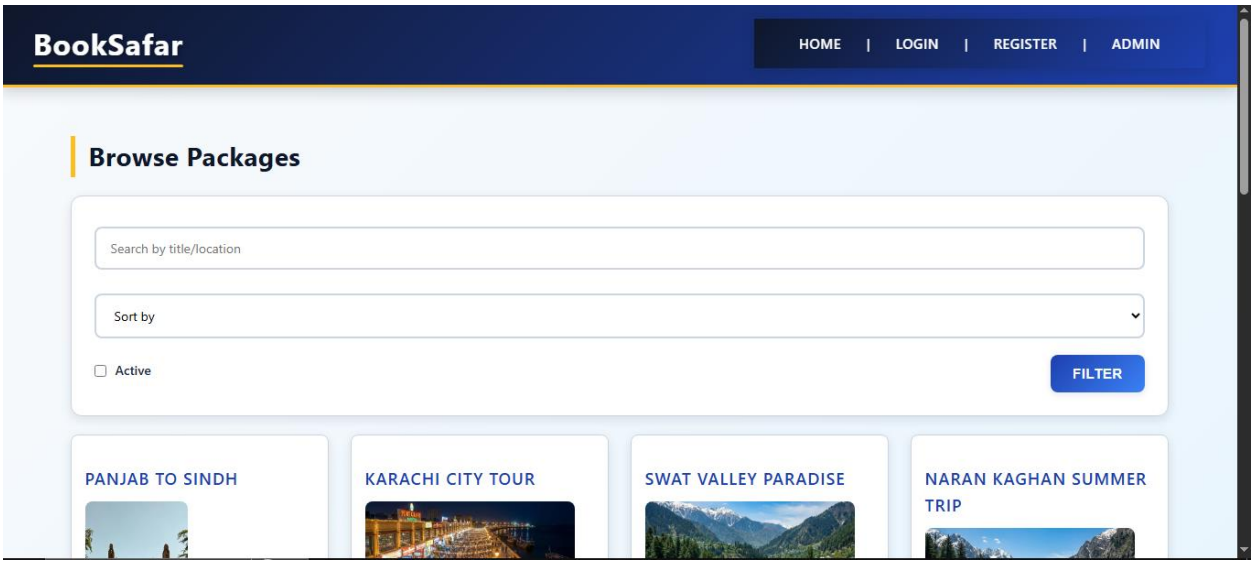
- **Browse & Book**
  Visitor searches or filters packages → opens package details → if seats and dates are valid and user is logged in, clicks *Book* → confirms and proceeds to payment → system records booking and payment, decrements available seats → user views booking in *My Bookings*.
- **Cancel Booking (User)**
  From *My Bookings*, if within 48-hour window and status is *Booked* → user selects *Cancel* → submits reason → booking marked *Cancelled*, payment marked *Refunded*, seat count incremented.
- **Submit Review**
  After travel date or once status is *Completed* → user opens *Add Review* → enters

booking ID, rating, and comment → review saved and displayed on package details → admin can delete reviews via the reviews module.

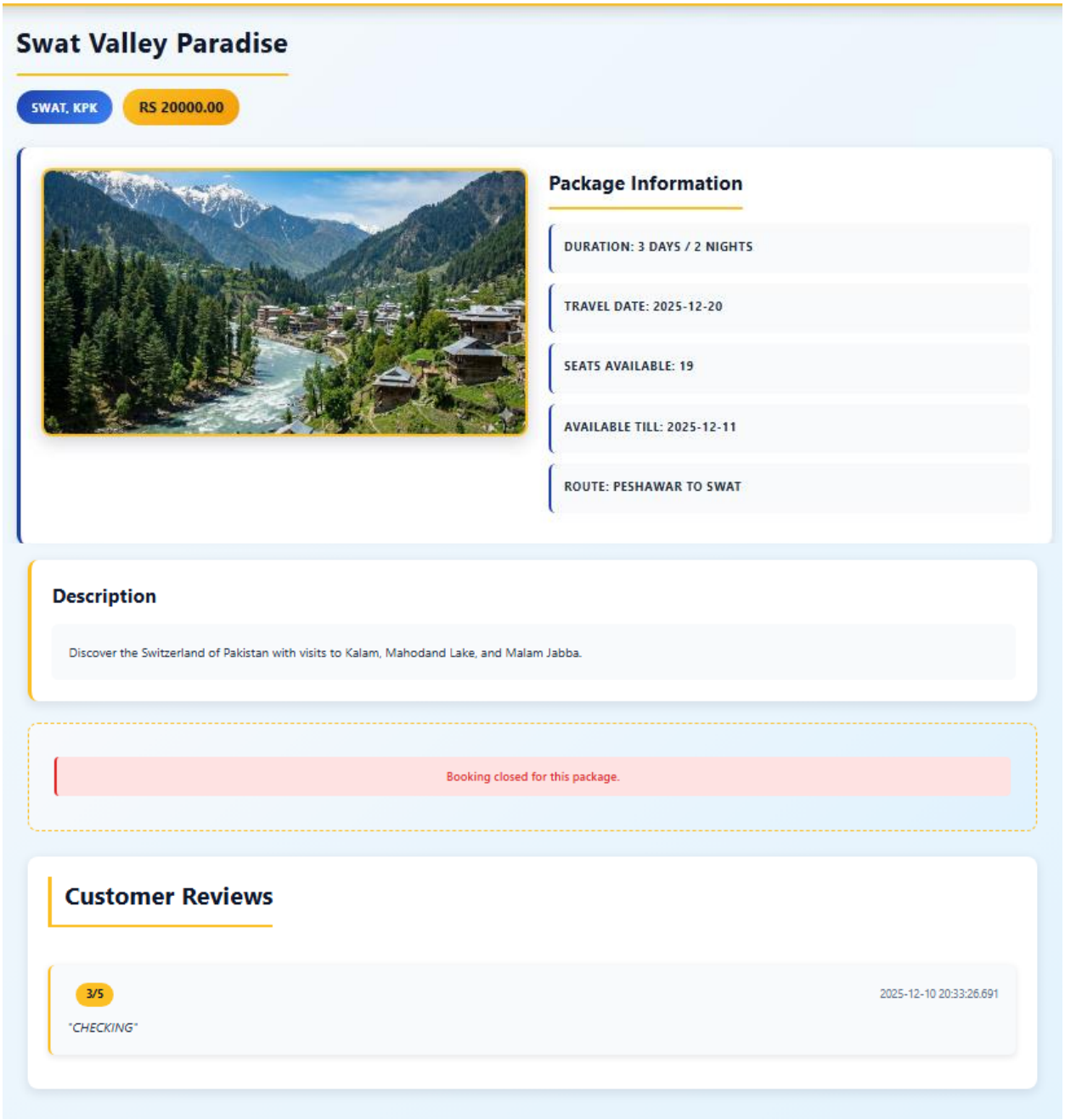
- **Admin Manage Packages**
  Admin logs in → accesses dashboard → adds or edits packages (travel dates, availability, seats, price, route, image) → packages appear in public catalog → admin can soft-delete packages when needed.
- **Admin Monitor Bookings, Users, and Reviews**
  Admin views bookings for their packages (including payment status and auto-complete aging) → sees users who booked their packages → deletes reviews associated with their packages.

Screenshots:

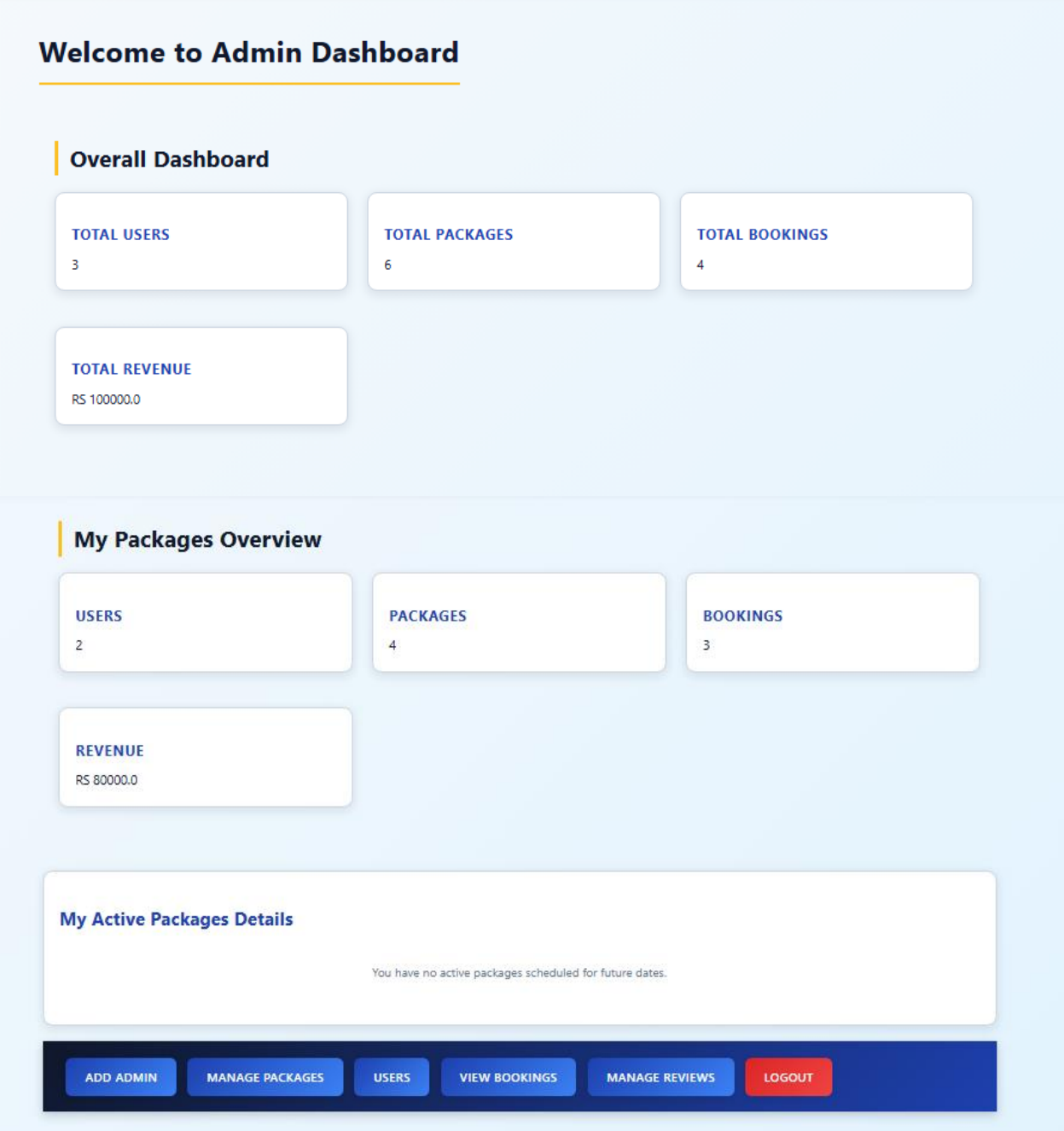
Img1:



Img 2:


Swat Valley Paradise
SWAT, KPK
RS 20000.00
Package Information
DURATION: 3 DAYS / 2 NIGHTS
TRAVEL DATE: 2025-12-20
SEATS AVAILABLE: 19
AVAILABLE TILL: 2025-12-11
ROUTE: PESHAWAR TO SWAT
Description
Discover the Switzerland of Pakistan with visits to Kalam, Mahodand Lake, and Malam Jabba.
Booking closed for this package.
Customer Reviews
3/5
2025-12-10 20:33:26.691
"CHECKING"

Img 3:


Welcome to Admin Dashboard
Overall Dashboard
TOTAL USERS
3
TOTAL PACKAGES
6
TOTAL BOOKINGS
4
TOTAL REVENUE
RS 100000.0
My Packages Overview
USERS
2
PACKAGES
4
BOOKINGS
3
REVENUE
RS 80000.0
My Active Packages Details
You have no active packages scheduled for future dates.
ADD ADMIN
MANAGE PACKAGES
USERS
VIEW BOOKINGS
MANAGE REVIEWS
LOGOUT

Integration (Web ↔ Database):

Connection Layer

- All JSP pages include *config.jspf* to establish Oracle JDBC connections via getConnection().
- Each page uses PreparedStatement with parameterized queries and ensures proper closure of resources.

User Operations

- **Login (login.jsp):**
  Executes:

  SELECT userid, name FROM Users WHERE email=? AND password=?;

  On success, sets user session; otherwise, displays error.
- **Registration (register.jsp):**
  Checks for existing email → inserts new user record → prompts user to log in.
- **Package Listing/Search (index.jsp):**
  Builds filtered SELECT queries on *Package* (title/location filters, active seats/date validation, sorting by price) → renders package cards.
- **Package Details (packageDetails.jsp):**
  Fetches package by packageid (excluding deleted packages) → retrieves associated reviews → displays package details and reviews.
- **Booking Start (book.jsp):**
  Loads package availability → on POST, stores pending package info in session → redirects to payment.
- **Payment & Booking Creation (processPayment.jsp):**
  - Locks package row FOR UPDATE
  - Validates seats and travel date
  - Inserts booking record
  - Updates package seats (seats = seats - 1)
  - Inserts payment record with status *Completed*
  - Commits transaction and clears pending session data

- **My Bookings (myBookings.jsp):**
    - Auto-updates bookings older than 2 days to *Completed*
    - Joins *Booking*, *Package*, and *Payment* for current user
    - Displays booking status and payment details
    - Provides cancellation option within 48 hours
- **Cancel Booking (cancelBooking.jsp):**
    - Verifies ownership, status, and time window
    - On POST:
    - UPDATE Booking SET status='Cancelled', cancelreason=reason, canceldate=CURRENT_TIMESTAMP WHERE bookingid=uid;
    - UPDATE Payment SET paystatus='Refunded' WHERE bookingid=uid;
    - UPDATE Package SET seats=seats+1 WHERE packageid=pid;
- **Reviews (addReview.jsp):**
  Validates booking ownership, travel completion/status *Completed*, and absence of prior review → inserts new review record.

Admin Operations

- **Login (login.jsp):**
  Executes:

  SELECT adminid, name FROM Admin WHERE email=? AND password=?;

  On success, sets admin session.
- **Packages (packages.jsp):**
    - Lists active packages
    - Displays per-package booking counts via grouped query
    - Supports soft-delete action
    - *addPackage.jsp* inserts new packages
    - *editpackage.jsp* updates existing packages

- **Bookings (bookings.jsp):**
    - Auto-completes aging bookings
    - Selects bookings and payments for packages created by current admin
- **Users (users.jsp):**
  Joins *Users*, *Booking*, and *Package* to list users who booked the admin's packages.
- **Reviews (reviews.jsp):**
  Lists reviews for admin's packages → allows deletion via:
- DELETE FROM Review WHERE reviewid=?;
- **Dashboard (dashboard.jsp):**
  Executes multiple aggregate queries for:
    - Total users, packages, bookings, and revenue
    - Admin-specific users, packages, bookings, and revenue
    - Active packages
      Displays upcoming packages with booking counts in tabular format.

Testing & Validation:

Functional Test Cases

- **User Login**
    - Valid credentials → redirect to homepage with session established
    - Invalid credentials → error message displayed on *login.jsp*
- **Registration**
    - Missing required fields or short password → error message displayed
    - Unique email enforced (query check) → successful insert and success banner on *register.jsp*
- **Package Search/List**

  - Keyword search and Active filter on *index.jsp* → only matching, non-deleted, valid packages with seats > 0 appear
  - Sorting toggles between ascending/descending price

- **Package Details**
  - Valid, non-deleted package → details and reviews displayed on *packageDetails.jsp*
  - Invalid or soft-deleted package ID → "Package not found" message

- **Booking Flow**
  - From package details → *book.jsp* shows availability
  - Proceed to *processPayment.jsp* → booking created, seats decremented, payment recorded, success message displayed

- **Cancellation**
  - From *myBookings.jsp*, cancel a *Booked* booking within 48 hours → status updated to *Cancelled*, payment marked *Refunded*, seat incremented
  - After 48 hours → error message "Cancellation window passed"

- **Auto-Complete Aging**
  - Bookings older than 2 days automatically marked *Completed* in *myBookings.jsp* and *bookings.jsp*
  - Verified by status update

- **Reviews**
  - After travel date or *Completed* status → submit review in *addReview.jsp* with rating (1–5) and booking ID → "Review submitted" message
  - Duplicate review for same booking → error message
  - Cancelled or unfinished trips → error message

- **Admin Authentication**
  - Valid admin credentials → redirect to dashboard
  - Invalid credentials → error message displayed

- **Admin Package CRUD**

- Add package in *addPackage.jsp* → appears in public list
- Edit package in *editpackage.jsp* → updates values successfully
- Delete package in *packages.jsp* → soft-delete applied, package no longer listed

- **Admin Reviews/Bookings/Users**
  - Delete review in *reviews.jsp* → review removed
  - Bookings list → correct status and payment details displayed
  - Users list → shows only users tied to admin's packages

Error Handling Behavior:

- **Input Validation**
  - Required-field checks on registration and package add/edit
  - Rating bounds enforced (1–5)
  - Numeric/date parsing wrapped in try/catch with user-friendly error messages
- **Authentication Guards**
  - Pages requiring session (user/admin) redirect to respective login when session attributes are missing
- **Availability Guards**
  - Booking and payment re-validate seats and travel/availability dates
  - Unavailable packages trigger error and block booking
- **Transactions**
  - Payment and cancellation flows wrap multi-step database updates in transactions
  - Rollback applied on exception to prevent partial writes
- **Soft Delete**
  - Packages flagged with *deletedate/deletedby* excluded from public queries
  - Edits restricted to the owning admin
- **Graceful Resource Closure**

- Result sets, statements, and connections closed in *finally* blocks
- Exceptions logged and safely ignored

Deployment & Maintenance:

Setup Instructions

- **Prerequisites:**
  - Apache Tomcat (compatible with the required JSP/Servlet version)
  - Oracle XE/Oracle Database with a user account matching the credentials in *config.jspf*
  - Oracle JDBC driver (*ojdbc*) placed in Tomcat's lib directory
- **Database Setup:**
  - Create schema and tables as defined in the ERD
  - Load seed data from *Sample Database.txt* after adjusting admin/user IDs to align with primary keys
  - Ensure appropriate grants for CONNECT, CREATE SESSION, and DML operations
- **Configuration:**
  - Verify database URL and credentials in *config.jspf*
  - Update host, port, SID, or service name as required
  - Restart Tomcat after configuration changes
- **Deployment:**
  - Place the project under Tomcat's webapps/BookSafar directory
  - Start Tomcat
  - Access the application via:
    - Public interface: http://localhost:8080/BookSafarAi/
    - Admin interface: http://localhost:8080/BookSafarAi/admin/
- **Testing:**

- Create at least one admin record
- Verify admin login functionality
- Add a package and perform a booking/payment flow
- Confirm that seats decrement and payments are recorded correctly

Future Enhancements:

- **Security:**
  - Implement password hashing and salting
  - Strengthen input validation and output escaping
  - Introduce role-based authorization for admin actions
- **Payments:**
  - Integrate with a real payment gateway
  - Support webhook-based confirmations and refunds
  - Record transaction references for audit purposes
- **User Experience & Data:**
  - Add pagination and advanced filtering for package listings
  - Enable image upload and storage instead of static path references
  - Enhance review display with average ratings and review counts
- **Operations:**
  - Implement logging and monitoring
  - Support environment-based configuration (development/production)
  - Automate database migrations and seed scripts
- **Reliability:**
  - Ensure concurrency-safe seat management with explicit transaction isolation
  - Add retry mechanisms and alerts for payment failure scenarios

Conclusion:

Summary

BookSafar provides a complete solution for travel package management, covering package discovery, booking, secure payment capture, cancellations, and post-trip reviews. It also includes an administrative console for publishing and monitoring packages, bookings, users, and feedback—all built on a unified JSP/Oracle technology stack.

Limitations

- Passwords stored in plaintext
- No CSRF protection or strong input validation/escaping
- Lack of integration with real payment gateways
- Minimal error handling and user experience for failures
- No pagination or advanced search features
- Limited auditing and logging capabilities

Future Improvements

- Implement secure credential storage with hashing and salting
- Add CSRF protection and strengthen input validation/sanitization
- Integrate with real payment providers using webhook-based reconciliation and refund handling
- Enhance user experience with pagination, advanced filters, and media uploads
- Improve system observability through logging, metrics, and monitoring tools
- Introduce automated migration scripts and environment-based configuration for smoother operations