**🧾 SOFTWARE DESIGN SPECIFICATION (SDS)**

**ỨNG DỤNG DU LỊCH – TRAVEL APP**

**1. Mục tiêu tài liệu**

Mô tả chi tiết thiết kế hệ thống để đáp ứng SRS: kiến trúc, các thành phần, mô hình dữ liệu, API, UML diagrams, mô hình triển khai, bảo mật và phân quyền. Tài liệu phục vụ đội phát triển, tester và vận hành.

**2. Tổng quan kiến trúc**

**2.1 Mô hình tổng quát**

Hệ thống theo mô hình **client–server** 3 lớp:

* **Client (Frontend):** Ứng dụng di động và/hoặc web (React Native / React). Chịu trách nhiệm UI/UX, thu thập input người dùng, gọi API.
* **API Server (Backend):** Spring Boot cung cấp REST API, xử lý nghiệp vụ, xác thực, validation, transaction.
* **Database (Storage):** PostgreSQL lưu dữ liệu chính; có thể dùng Redis cho cache/session; S3-compatible storage cho ảnh.

Luồng tổng quát:  
Client → HTTPS → API Server → DB / External services (Payment, Maps, Email)

**2.2 Thành phần chính**

* **Frontend:** screens, components, state management (Context/Redux), network layer (API client), offline handling (cache).
* **Backend:** controllers, services, repositories, DTOs, authentication (JWT), exception handling, validation, scheduled jobs.
* **DB:** relational schema (users, destinations, bookings, reviews, images).
* **External:** Payment gateway (nếu cần), Maps API (Google/Mapbox), Storage (S3), Email service (SendGrid).

**3. Thiết kế chi tiết**

**3.1 UML — Use Case (mô tả)**

Use cases chính (từ SRS):

* UC01: Đăng ký / Đăng nhập
* UC02: Tìm kiếm tour
* UC03: Xem chi tiết tour
* UC04: Đặt tour
* UC05: Xem lịch sử đặt tour
* UC06: Đánh giá tour
* UC07: Quản lý tour (Admin)
* UC08: Quản lý người dùng (Admin)

*(Bạn có thể chuyển các use case này vào draw.io / Lucidchart để vẽ sơ đồ Use Case)*

**3.2 UML — Class Diagram (mô tả lớp backend chính)**

**Các entity chính (mô tả lớp):**

1. User

* id: UUID
* email: string
* passwordHash: string
* name: string
* phone: string
* role: enum {USER, ADMIN}
* avatarUrl: string
* createdAt: timestamp
* updatedAt: timestamp

1. Destination

* id: UUID
* name: string
* slug: string
* description: text
* location: string
* coordinates: geometry / point (optional)
* images: jsonb (array of urls)
* category: string
* priceFrom: numeric
* createdAt, updatedAt

1. Booking

* id: UUID
* userId: UUID (FK -> User)
* destId: UUID (FK -> Destination)
* dateFrom: date
* dateTo: date
* guests: int
* contactName: string
* contactPhone: string
* status: enum {PENDING, CONFIRMED, CANCELLED, COMPLETED}
* totalAmount: numeric (optional)
* notes: text
* createdAt, updatedAt

1. Review

* id: UUID
* userId: UUID
* destId: UUID
* rating: int (1-5)
* comment: text
* createdAt

1. Image (optional separate table)

* id: UUID
* entityType: string (destination/user/other)
* entityId: UUID
* url: string
* altText: string
* createdAt

**Relationships (summary):**

* User 1..\* Booking (1 user có nhiều booking)
* Destination 1..\* Booking (1 destination có nhiều booking)
* User 1..\* Review; Destination 1..\* Review

*(Bạn có thể vẽ Class Diagram dựa trên các lớp trên; thêm DTO/VO classes trên backend như UserDTO, BookingRequestDTO, etc.)*

**3.3 Sequence Diagram (ví dụ: luồng đặt tour)**

Mô tả bằng bước:

**Actors:** User (Client), Frontend, Backend (Controller → Service → Repo), Database, (Optional) Payment Service, Email Service.

1. User mở màn hình detail và nhấn "Đặt tour".
2. Frontend hiển thị form, user điền và submit.
3. Frontend gửi POST /api/bookings kèm JWT trong header Authorization: Bearer <token>.
4. Backend BookingController nhận request, kiểm tra JWT (AuthFilter), lấy userId.
5. Controller validate input → gọi BookingService.createBooking().
6. BookingService kiểm tra tính hợp lệ (destination tồn tại, chồng lịch, số khách hợp lệ).
7. Tạo đối tượng Booking (status=PENDING) và bookingRepository.save(booking).
8. Nếu có payment, gọi Payment Service (external) và chờ kết quả (synchronous hoặc async).
9. Khi payment success → update booking.status=CONFIRMED; nếu failure → status=CANCELLED hoặc PENDING.
10. Gửi email/Xác nhận push/notification cho user.
11. Backend trả response 201 Created với booking data.
12. Frontend hiển thị màn confirm cho user.

**3.4 ERD — Mô hình cơ sở dữ liệu (bảng & khóa)**

Định nghĩa bảng (chi tiết):

**users**

* id UUID PK
* email varchar(255) UNIQUE NOT NULL
* password\_hash varchar(255) NOT NULL
* name varchar(255)
* phone varchar(50)
* role varchar(20) DEFAULT 'USER'
* avatar\_url text
* created\_at timestamptz
* updated\_at timestamptz

**destinations**

* id UUID PK
* name varchar(255) NOT NULL
* slug varchar(255) UNIQUE
* description text
* location varchar(255)
* coordinates GEOGRAPHY(Point) NULL
* images jsonb NULL
* category varchar(100)
* price\_from numeric
* created\_at, updated\_at

**bookings**

* id UUID PK
* user\_id UUID FK -> users(id)
* dest\_id UUID FK -> destinations(id)
* date\_from date
* date\_to date
* guests int
* contact\_name varchar(255)
* contact\_phone varchar(50)
* status varchar(20) default 'PENDING'
* total\_amount numeric
* notes text
* created\_at, updated\_at

**reviews**

* id UUID PK
* user\_id UUID FK
* dest\_id UUID FK
* rating smallint
* comment text
* created\_at timestamptz

**indexes**

* idx\_users\_email (unique)
* idx\_destinations\_name (gin to\_tsvector for fulltext search)
* idx\_bookings\_user\_id
* idx\_bookings\_status
* idx\_reviews\_dest\_id

*(ERD sơ bộ: Users ↔ Bookings ↔ Destinations; Users ↔ Reviews ↔ Destinations)*

**4. Thiết kế API (RESTful)**

**Nguyên tắc chung**

* Giao thức: HTTPS (TLS)
* Định dạng: JSON cho request/response
* Xác thực: JWT Bearer token cho endpoints private
* Pagination: page & size hoặc cursor-based cho danh sách
* Trả lỗi: cấu trúc JSON { "timestamp": "...", "status": 400, "error": "Bad Request", "message": "Chi tiết", "path": "/api/..." }

**Danh sách endpoint chính (chi tiết)**

**Auth**

* POST /api/auth/register
  + Body: { "email": "...", "password": "...", "name": "..." }
  + Response 201: { "userId": "...", "token": "..." }
  + Errors: 400 (invalid), 409 (email exists)
* POST /api/auth/login
  + Body: { "email": "...", "password": "..." }
  + Response 200: { "token": "...", "user": { ... } }
  + Errors: 401 (invalid credentials)

**Users**

* GET /api/users/me (Auth) → 200 { user }
* PUT /api/users/me (Auth) → body { name, phone, avatarUrl } → 200

**Destinations**

* GET /api/destinations
  + Query: q, category, minPrice, maxPrice, page, size, sort
  + Response 200: { data: [ {id,name,slug,priceFrom,images,...} ], meta: { page, size, total } }
* GET /api/destinations/{id}
  + Response 200: destination detail incl. avgRating, reviews
* POST /api/destinations (Admin)
  + Body: destination payload → 201
* PUT /api/destinations/{id} (Admin) → 200
* DELETE /api/destinations/{id} (Admin) → 204

**Bookings**

* POST /api/bookings (Auth)
  + Body: { destId, dateFrom, dateTo, guests, contactName, contactPhone, notes }
  + Response 201: booking object
* GET /api/bookings (Auth)
  + Query: user=true default → returns bookings for user
  + Admin: can pass ?all=true
* GET /api/bookings/{id} (Auth) → 200 (if owner or admin)
* PATCH /api/bookings/{id} (Auth/Admin) → update status (cancel/confirm)

**Reviews**

* POST /api/destinations/{id}/reviews (Auth)
  + Body: { rating, comment } → 201
* GET /api/destinations/{id}/reviews → paginated list

**Admin / Analytics**

* GET /api/admin/reports/bookings?from=YYYY-MM-DD&to=YYYY-MM-DD (Admin) → stats

**Mẫu lỗi/response**

* Success: 200 OK / 201 Created
* Client error: 400 Bad Request (validation), 401 Unauthorized, 403 Forbidden, 404 Not Found
* Server error: 500 Internal Server Error

**5. Giao diện (UI) & Prototype**

* **Figma:** (chèn link prototype) — screens: Home, Search Results, Destination Detail, Booking Form, Profile, Login/Register, Admin Dashboard.
* **Style system:** màu chủ đạo, typography, spacing tokens (đã chuẩn bị từ design system SRS).
* **Component library:** Button, Card, List, Modal, DatePicker, Rating component.

*(Bạn chèn link Figma vào vị trí này hoặc upload file thiết kế.)*

**6. Mô hình triển khai (Deployment Diagram)**

**6.1 Mô tả triển khai**

* **Client:** Mobile App (Expo/React Native) hoặc Web (React) — cài đặt trên device.
* **API Server:** Spring Boot ứng dụng chạy trong Docker container.
* **DB:** PostgreSQL (container hoặc managed RDS).
* **Object Storage:** S3 (ảnh).
* **Reverse Proxy / Load Balancer:** nginx / cloud LB.
* **CI/CD:** GitHub Actions → build, test, push Docker image → deploy (Render / GCloud Run / AWS ECS).
* **Monitoring:** Prometheus + Grafana (metrics) & Sentry (errors).
* **Backup:** scheduled DB backups, snapshots.

**6.2 Ví dụ flow deploy**

1. Commit → Pull Request → CI runs tests.
2. Merge → GitHub Action builds image, pushes to registry.
3. Deployment pipeline deploys to staging → smoke tests → promote to production.

**7. Bảo mật & phân quyền**

**7.1 Xác thực & ủy quyền**

* **Xác thực:** JWT (Bearer token) hoặc OAuth2. Token được cấp sau login/register và kèm expiry.
* **Phân quyền:** role-based (USER, ADMIN). Endpoints admin protected bằng role check.

**7.2 Bảo mật dữ liệu**

* Luôn dùng **HTTPS/TLS**.
* **Mật khẩu:** lưu dưới dạng hash (bcrypt/argon2).
* **Dữ liệu nhạy cảm:** mã hóa at-rest cho các trường cần thiết (PII).
* **Secrets:** lưu ở secret manager / environment variables (KHÔNG commit vào repo).
* **Validation & Sanitization:** validate input, chống SQL injection (ORM + param binding), XSS trên dữ liệu hiển thị.
* **Rate limiting & Throttling:** bảo vệ endpoints nhạy cảm (login, register) để chống brute-force.
* **Audit logging:** giữ log hoạt động quan trọng (đăng nhập, thay đổi đặt tour, admin actions).

**7.3 Tuân thủ & privacy**

* Tuân thủ quy định bảo vệ dữ liệu cá nhân (VD: GDPR/PDPA nếu áp dụng).
* Cung cấp chính sách bảo mật & điều khoản sử dụng.

**8. Tối ưu & mở rộng (Non-functional design notes)**

* **Caching:** cache danh sách destinations/popular queries (Redis) để giảm tải DB.
* **Search:** dùng full-text search (Postgres tsvector) hoặc Elasticsearch cho tìm kiếm nâng cao.
* **Pagination:** cursor-based cho danh sách lớn.
* **Asynchronous tasks:** sử dụng job queue (Redis + worker) cho email, thumbnail generation, báo cáo.
* **Internationalization (i18n):** resource files cho VN/EN.

**9. Testability & QA**

* **Unit tests:** services, utils, validators.
* **Integration tests:** controllers + in-memory DB (H2) hoặc Testcontainers.
* **E2E tests:** simulate flows (register → login → booking) bằng Espresso/Detox/Playwright.
* **Performance tests:** JMeter / k6 để stress endpoints.
* **Security tests:** OWASP ZAP scan.

**10. Deliverables từ SDS**

* File SDS hoàn chỉnh (docx/pdf).
* UML diagrams: Use Case, Class, Sequence, ERD (file .drawio hoặc .svg).
* OpenAPI / Swagger spec (YAML/JSON) cho API.
* Figma prototype link.
* Deployment scripts: docker-compose.yml, Dockerfile, GitHub Actions workflow.

**11. Phụ lục — Mẫu OpenAPI (snippet)**

openapi: 3.0.0

info:

title: TravelApp API

version: 1.0.0

paths:

/api/auth/login:

post:

summary: Login user

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

email:

type: string

password:

type: string

responses:

'200':

description: ok

content:

application/json:

schema:

type: object

properties:

token:

type: string

user:

$ref: '#/components/schemas/User'

components:

schemas:

User:

type: object

properties:

id:

type: string

email:

type: string

name:

type: string

**12. Ghi chú triển khai nhanh (checklist)**

* Tạo repo: /frontend, /backend, /docs
* Thêm CI (maven/npm lint/test)
* Thiết lập environment: DEV/PROD config, secret store
* Tạo baseline DB migration (Flyway)
* Triển khai staging & smoke test trước production