# STG Tour Booking Platform – Detailed Design Document

## Executive Summary: Vision and Goals

Saudi Tourism Group (STG) aims to solve a key challenge for tourists in Saudi Arabia: the difficulty of finding and booking authentic, high-quality local tours. The current market is fragmented, often offline, and presents language barriers to international visitors. STG’s vision is to create a single, trusted, **multilingual digital platform** that connects a diverse range of travelers – from families to solo luxury adventurers – with **verified, unique, and safe Saudi experiences**. The platform’s core function will be to provide a **seamless, secure, and instant booking process**, becoming the go-to solution for discovering the true culture and beauty of the Kingdom. By curating authentic desert adventures, cultural excursions, and more, STG will position itself as the leading name for genuine Saudi experiences.

Our goal is to translate this powerful vision into a world-class **web-based application** that not only meets but exceeds the expectations of the modern tourist. The solution will initially be delivered as a responsive **mobile-friendly web app** (to avoid app store deployment overhead) while laying the groundwork for potential native apps in the future. This document presents a comprehensive blueprint covering the project’s phased implementation, core features, technical architecture, and future roadmap, enabling STG’s internal team (and AI-assisted development tools like Codex CLI) to efficiently bring this platform to life.

## Phase 1: Discovery, Strategy & Design

**Goal:** Establish a complete blueprint for the STG tour booking platform before development begins. This phase focuses on understanding requirements, researching the market, and designing an intuitive user experience aligned with Saudi culture and STG’s premium brand.

**Activities:**  
- **Stakeholder Workshops:** Collaborate with STG stakeholders to finalize the detailed feature list, user roles, and user flows. We will map out the end-to-end journey for tourists (from browsing to post-tour feedback) as well as the workflows for tour operators and administrators. Technical requirements (functional and non-functional) will be clarified during these sessions.  
- **Market Research:** Analyze competitor platforms (e.g., Viator, GetYourGuide, local tour websites) to identify best practices and opportunities for differentiation. We’ll pay special attention to features that resonate with users in the tours/activities sector, such as social proof (reviews), ease of booking, and trust signals (guide verification).  
- **Branding & UI/UX Design:** Create a visually stunning and user-friendly interface that reflects Saudi culture and STG’s premium brand identity. This includes defining **brand guidelines** (logo usage, colors, typography) and designing the complete user interface. We will produce wireframes for all key screens and then high-fidelity mockups or interactive prototypes. The design will emphasize intuitive navigation and culturally relevant imagery, ensuring users immediately feel the authenticity of Saudi experiences. The UX will cater to both English and Arabic (right-to-left) languages from the start, given the multilingual goal.

**Deliverables:** By the end of Phase 1, we will have:  
- **Brand Guidelines:** A style guide detailing the app’s visual identity and design principles.  
- **Complete App Wireframes:** Low-fidelity wireframes covering all screens (browse tours, tour detail, booking flow, user profile, etc., as well as admin/operator panels) to map out layout and navigation.  
- **Interactive Prototypes:** Clickable prototype (using tools like Figma or Adobe XD) allowing stakeholders to experience the user journey from discovery to booking to review. This will help in validating flow and usability before development.  
- **Detailed Technical Specification:** *This design document.* A comprehensive spec outlining the system architecture, data models, API design, and integration points. It will serve as a blueprint for the development team and can be provided to the Codex CLI (AI coding agent) to accelerate development.

*(Note: Phase 1 is a critical planning phase. Investing time here ensures that in Phase 2 we can develop the MVP efficiently with fewer costly changes. The technical spec will also help align the Codex AI development tool with our intended architecture.)*

## Phase 2: MVP Development (Minimum Viable Product)

**Goal:** Build and launch the core, essential version of the platform as a **web-based application** accessible on mobile and desktop browsers. The MVP will focus on the “must-have” features that provide immediate value to tourists and allow STG to start onboarding tours and users. By choosing a web-based approach (responsive design or Progressive Web App) for both the tourist-facing app and the admin/operator panel, we avoid initial app store deployment costs and delays, while ensuring broad accessibility[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts). The MVP will be developed for **deployment on a cloud platform** for scalability and reliability (e.g., AWS or Azure). It will include two main components: (1) the Tourist-Facing Web App, and (2) the Admin & Tour Operator Web Portal.

### Tourist-Facing Web App (User Platform) – Core Features

* **Tour Discovery & Search:** An intuitive browsing experience for tours. Users can discover tours through a home page showcasing featured experiences and can search with filters (location, date, category, price, etc.). High-quality images, engaging descriptions, and guide profiles will be displayed for each tour listing to entice users. The search results page will allow sorting and filtering, helping users find the perfect tour. (For example, a user could filter by “Desert Safari” or sort by highest rated tours.)
* **Detailed Tour Pages:** Each tour will have a dedicated page with comprehensive details: description of the experience, itinerary highlights, duration, included amenities (e.g., meals, transportation), photos/gallery, location map, guide information (with bio and languages spoken), and available dates or schedule. Importantly, **real-time availability** will be shown – a calendar or list of upcoming dates with open slots. If a tour has limited capacity, the system will display how many spots are left for a given date to create urgency.
* **Real-Time Booking & Scheduling:** A seamless booking flow is at the heart of the MVP. Users can select a date and the number of people, then instantly book if slots are available. The system will have a calendar that shows live availability updated by operators. Once a date is chosen, that slot’s availability is **locked** during checkout to prevent double-booking conflicts. Upon booking confirmation, the slot count decreases immediately in the database. (In future iterations, we may introduce a short hold on inventory during payment to further prevent conflicts.)
* **Secure Instant Payment:** Integration with a trusted payment gateway is essential for instant confirmation. We will integrate **Stripe** for global payments and plan for **HyperPay** for local payment methods in KSA (especially to support Mada debit cards and regional methods). Stripe will cover credit card payments (Visa, MasterCard, Amex) and support multiple currencies for international tourists[[2]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=%2A%20Cloud,io%2C%20Skylight)[[3]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=For%20instance%2C%20TravelBid%E2%80%99s%20tech%20stack,included). HyperPay (a leading MENA payment provider) can be introduced to accept local cards (e.g., Mada) and alternative methods like STC Pay, ensuring Saudi users have a familiar, convenient payment option. All transactions will be processed via these PCI-compliant gateways – **no sensitive card data will be stored on our servers**. Payment receipts/invoices will be emailed to users automatically.
* **User Accounts & Profiles:** Tourists will be able to create a personal account (sign up via email, and potentially social login in future). In their profile, they can view and edit personal information, see **booking history** (past tours with dates and status, as well as upcoming scheduled tours), and access any digital assets from their tours (e.g. the complimentary photos/videos provided by tour operators). Account creation will be optional for browsing, but required for booking so that we can maintain a record and offer a personalized experience. Users can also save favorite tours to a wishlist for future reference.
* **Ratings & Reviews:** After a tour is completed, users can rate the experience (e.g., 5-star scale) and leave a review. This system will build trust for new customers by highlighting authentic feedback. The app will display average ratings and review comments on tour pages. We will implement a simple but robust review moderation process – e.g., reviews might only be allowed from users who actually booked that tour (verified reviews). In the MVP, this can be basic, with manual admin oversight for any inappropriate content. Over time, a more refined moderation or incentive system (like badges for top reviewers) can be added.

*(The above features constitute the core user-facing functionality required to search, book, and review tours – essentially the entire customer journey from discovery to feedback. Together, they provide immediate utility and a strong foundation to grow upon.)*

### Admin & Tour Operator Web Portal – Core Features

The platform will include a secure web-based portal for **STG administrators** and **verified tour operators**. This will likely be a single web application with role-based access control – where STG internal staff have admin privileges across the system, and tour operators (guides or agencies partnering with STG) have access only to their own listings and bookings. This **back-office** system is critical for managing content and operations[[4]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=User,items%2C%20and%20pay%20for%20them). Core features include:

* **Tour Management Dashboard:** Tour operators can create and manage their tour listings through an easy interface. This includes adding tour details (title, descriptions in multiple languages, photos, pricing, capacity, duration, etc.). They can also specify the schedule or recurring availability (e.g., a tour happens every Friday, or specific date ranges). Operators will be able to update a tour’s availability calendar – for example, blocking off dates that are fully booked or unavailable. Admins can oversee all listings, editing or approving content to ensure quality and consistency (especially for brand alignment and safety compliance).
* **Booking Management:** A module to view and manage bookings in real-time. Tour operators can see bookings for their tours (upcoming and past) including customer details and any special requests. They should be able to confirm that a booking is acknowledged, mark a tour session as completed, or cancel/reschedule if needed (with notifications to the customer handled by the system). Administrators will have a global view of all bookings across the platform and can step in to assist or reassign if an operator has an issue. This dashboard will also help STG monitor overall booking volume and detect any issues (e.g., a tour that gets overbooked or cancellations).
* **Customer Management & Support Tools:** Admin users will have access to a customer database and tools to support users. This includes viewing user profiles, past bookings, and the ability to act on behalf of a user if needed (for example, to cancel a booking per a phone request). Integrations with customer support channels (email ticketing or chat) can be considered so that any support issues can be tracked. (In the MVP, this could be as simple as an admin adding notes to a booking or marking a refund issued.)
* **Ratings/Reviews Moderation:** Admins will have an interface to monitor the reviews coming in. They can flag or remove inappropriate reviews and see overall tour ratings. This ensures the ratings system remains trustworthy. In future, this could be augmented with automation or AI to detect spam or offensive content.
* **Financial Reports & Earnings:** Tour operators should be able to view their earnings from the platform – a breakdown of completed bookings, their revenue share, and payout status. The portal can show simple reports (e.g., monthly bookings, total revenue earned). For MVP, this might be a basic table export, while future versions could include charts and analytics. STG Admins will have more comprehensive financial reporting to track platform revenue, commissions, and to facilitate payouts to operators.
* **Administration Settings:** Admin users may have additional controls such as managing platform-wide settings (supported languages, featured tours on the homepage, promotional codes or discounts, etc.). They will also handle **tour operator onboarding** – e.g., verifying a new operator’s identity and tours before approving them on the platform. For MVP, these processes might occur off-platform (via direct communication), with admin manually adding new operators into the system once vetted.

**Deliverables for Phase 2:** By the end of MVP development, we will have a **fully functional, tested, and deployed web application** comprising the user-facing site and the admin/operator portal. The system will be deployed to a production environment (e.g., an AWS or Azure cloud server) with proper domain setup (possibly a main website for users and a subdomain for the portal, or a single app that detects role on login). We will prepare user documentation (basic how-to for operators) and internal technical documentation (API docs, etc.). The MVP should be ready for STG to launch to a pilot group of users and begin real transactions. Launch on mobile app stores is **not required in this phase**, as the mobile-optimized web app will be the primary channel (users can “Add to Home Screen” if we leverage PWA features, giving an app-like experience without app store installation[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts)). This strategy saves on app store fees and bypasses the lengthy review processes, allowing rapid iteration and direct deployment of updates[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts).

## Phase 3: Post-Launch Enhancements & “Wow Factor” Features

**Goal:** Once the MVP is live and we gather real user feedback, Phase 3 focuses on iterating to improve the user experience and introducing the standout features that will differentiate STG’s platform. These are the “nice-to-have” or innovative features that were envisioned, which can significantly enhance engagement and delight users. We will prioritize these based on user demand and business impact, rolling them out in stages. Some key enhancements under consideration:

* **Personalized Itineraries & AI Recommendations:** Implement an AI-driven recommendation engine that suggests tours and builds custom itineraries for users. By analyzing user preferences (e.g., if a user frequently books cultural tours or indicates interests during sign-up) and past behavior, the system can recommend relevant experiences (for example, suggest a historical city tour to someone who enjoyed a museum tour)[[5]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=One%20of%20Onix%E2%80%99s%20clients%2C%20a,the%20solutions%20they%20use%20are). Over time, as data grows, this could evolve into automatically curating multi-day itineraries – stringing together complementary tours and activities into a coherent trip plan. Personalized suggestions increase engagement and conversion by showing travelers the experiences most likely to interest them.
* **In-App Messaging & Chat:** Integrate a messaging feature for direct communication between tourists and tour operators (or customer support). This could be a chat interface in the app where, after booking, a user can message the guide for coordination (e.g., to ask “What should I wear?” or notify about dietary restrictions). In the long term, a chatbot could also be implemented to handle frequently asked questions and provide 24/7 assistance[[6]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=Instant%20messages%20can%20inform%20users,engagement%20and%20winning%20customer%20loyalty). For MVP we rely on email/phone communications, but adding in-app chat will keep users engaged on the platform and provide a paper trail of communications. We might leverage a service like **Twilio** or **SendBird** for chat infrastructure, ensuring notifications are sent to users’ devices.
* **Exclusive/Premium Experiences Section:** A special section of the app highlighting unique tours **available only on STG’s platform**. This could involve partnerships for rare experiences (e.g., a private dinner in a desert camp under the stars, or an exclusive tour of a heritage site not open to general public tours). Technically, this feature is more about content curation and marketing, but we would create a distinct UI section for “STG Exclusives” to showcase these. It reinforces STG’s brand as the go-to place for one-of-a-kind Saudi experiences.
* **Integrated Photography & Video Memories:** One of the value-adds mentioned is providing complimentary trip photos/videos to customers. In the app, we can have a feature where after a tour, the tour operator uploads the photos or a short video taken during the trip. The app then notifies the customer that their **memory package** is ready. Users can view these photos/videos in their profile or a dedicated “My Tour Memories” section and share them on social media. For implementation, this requires having storage for media (we can use cloud storage like AWS S3) and an interface for upload (operator side) and viewing (user side). Proper permissions and expiries might be set (e.g., media is available for download for a certain period). This feature not only delights users but also encourages them to share their experiences, indirectly marketing the platform.
* **Multi-Language Expansion:** While the MVP will launch bilingual (English/Arabic), Phase 3 can expand to additional languages to cater to more tourists (such as French, Chinese, etc., depending on target demographics). The architecture will have been built with internationalization in mind, so adding a new language would involve translating the interface strings and possibly key tour content. We will use standard i18n libraries/practices so the app can seamlessly switch languages and support **right-to-left** layouts for Arabic. Offering content in users’ native languages greatly improves user experience and conversion – studies show 89% of users prefer content in their native language[[7]](https://www.bacancytechnology.com/blog/react-internationalization#:~:text=the%20core%20logic). Thus, as STG’s user base grows internationally, expanding language support will be important.

Beyond these, ongoing **data-driven improvements** will be made: we will analyze usage analytics and user feedback post-launch to refine UI elements, improve performance, and add any high-demand features (for example, a “Wish List/Save for later” if not in MVP, or social sharing functions so users can share tour links with friends). Phase 3 is also about scaling the system – if traffic increases, we might need to introduce load balancers, caching mechanisms, or even microservice architecture for certain components. Security enhancements (like advanced fraud detection for payments, or two-factor authentication for users) could also be part of this phase as the platform matures.

**Deliverables for Phase 3:** Regular app updates that include the above features, with each major addition accompanied by updated documentation and possibly user training materials (for operators if features affect them). By the end of this phase, STG’s platform should have a true “wow” factor that sets it apart from competitors, and a stable, scalable infrastructure ready for national and international growth.

## System Architecture and Technology Stack

*Illustration: High-level 3-tier architecture for the web application (Presentation → Application → Data). The STG platform will follow this model, separating concerns into front-end client, back-end server, and database/storage*[*[8]*](https://www.softkraft.co/web-application-architecture/#:~:text=Modern%20web%20application%20architecture%20is,tier%20structure)*.*

At a high level, the STG tour booking platform is designed as a classic **three-tier web application**[[8]](https://www.softkraft.co/web-application-architecture/#:~:text=Modern%20web%20application%20architecture%20is,tier%20structure), comprising: (1) the presentation layer (client-side web UI), (2) the application layer (server-side logic and APIs), and (3) the data layer (databases and storage). This separation of concerns improves scalability, maintainability, and security[[9]](https://www.softkraft.co/web-application-architecture/#:~:text=,depending%20on%20the%20exact%20architecture). Below is an overview of the architecture and chosen tech stack for each component:

* **Front-End (Presentation Layer):** This is the client-facing part of the system – essentially the web application that runs in users’ browsers (both tourists and operators/admins, as both will access via web). We recommend using a modern JavaScript framework such as **React** (with possibly Next.js for server-side rendering and SEO benefits) or **Angular/Vue**. React is a strong choice given its popularity and rich ecosystem, and it will allow building a responsive single-page application (SPA) that provides an app-like experience. The front-end will be responsible for the UI/UX – rendering tour lists, forms, etc. – and will communicate with the back-end via HTTP APIs (likely RESTful endpoints or GraphQL queries). It will also handle client-side input validation and some state management (for example, storing the user’s selections before booking). The site will be made fully **responsive** to work on various mobile devices and desktops. We will also leverage PWA (Progressive Web App) features: users can load the site **directly with a URL**, without any installation, and optionally “install” it to their home screen; we can also enable offline support (caching) for certain assets and push notifications for updates/offers[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts). Multi-language support on the front-end will be handled via an internationalization library (e.g., **i18next** for React or Angular’s i18n module), allowing us to easily swap out text and format dates/currencies based on the selected locale. We’ll ensure the UI accommodates **RTL** for Arabic seamlessly.
* **Back-End Application (Business Logic Layer):** The server side will be implemented with a robust framework to handle all core business logic and integrations. We propose using **Node.js** with an Express or NestJS framework (given Node’s ability to handle a large number of concurrent requests, which suits real-time booking operations). Node.js is also advantageous as our team can use one language (JavaScript/TypeScript) across front and back-end, and OpenAI’s Codex has strong familiarity with it for code generation. Alternatively, a Python (Django/Flask) or PHP (Laravel) approach could be used – but Node.js aligns well with real-time needs and modern stacks[[10]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=Jest%2C%20Kendo%20%2A%20Back,GCP%29%2C%20Heroku)[[11]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=%2A%20Back,NET%2C%20Micronaut%2C%20gRPC%2C%20RxJava%2C%20CloudCMS). The back-end will expose **RESTful API endpoints** that the front-end calls (for example: GET /api/tours for listing tours, POST /api/booking to create a booking, POST /api/login for authentication, etc.). We will also implement role-based access control in the API – ensuring, for instance, that operators can only access their own tours’ data. Key responsibilities of the back-end include: user authentication (likely JWT-based token auth for the SPA), input validation and business rules (e.g., ensuring a booking doesn’t exceed available capacity), interfacing with the payment gateways (Stripe/HyperPay SDK or API calls), sending notification emails, and maintaining audit logs. The back-end will also have scheduled jobs (cron tasks) as needed – for example, to send reminders to users a day before their tour, or to release any bookings that are pending payment for too long. Given the need for real-time updates (such as instantly reflecting a new booking), we may use WebSockets or a real-time library (Node has **Socket.io** available) especially for features like in-app chat or live availability refresh. (However, for MVP, simple polling or refresh upon actions might suffice to reduce complexity.) The back-end will be deployed on a scalable environment – e.g., an AWS EC2 instance or AWS Elastic Beanstalk for Node, behind a load balancer to allow scaling to multiple servers as traffic grows. We will enforce HTTPS for all client-server communication to secure data in transit.
* **Database and Storage (Data Layer):** We will use a **relational database** as the primary data store – likely **PostgreSQL** or **MySQL** (PostgreSQL is preferred for its reliability and rich feature set). The database will store structured data for the platform: user accounts, tour listings, bookings, reviews, etc., with relationships between them. An Entity-Relationship model will be designed. For example: a User table (with fields: id, name, email, password\_hash, role, etc.), a Tour table (id, title, description, price, guide\_id, etc.), a Booking table (id, tour\_id, user\_id, date, status, payment\_status, etc.), Review table (id, tour\_id, user\_id, rating, comment, date), and so forth. Using a SQL database ensures we can use transactions (important for the booking process, to avoid race conditions on inventory) and perform complex queries (e.g., find all tours in a certain location with average rating above 4). We will also employ caching for frequently read data that doesn’t change often (for example, tour listings or home page features) – using an in-memory store like **Redis** to speed up responses and reduce DB load. For storing photos/videos (tour images and the post-tour media), we will use cloud object storage. For instance, **Amazon S3** can be used to store all media files; the URLs can be stored in the database or generated on the fly. This allows efficient delivery via CDN and reduces load on our web server for serving large files[[12]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=,s%3A%20PayPal%2C%20Braintree%2C%20Stripe%2C%20Adyen).
* **External Integrations:** The architecture includes integration with several external services:
* **Payment Gateways:** As mentioned, Stripe and HyperPay will be integrated. Our back-end will communicate with these via their secure APIs. For example, when a user submits a booking with payment, the front-end might obtain a payment token (via Stripe Elements or similar) which is sent to our server, then the server uses Stripe’s API to charge the card. On success, our server marks the booking as paid in the database. HyperPay integration will be similar (likely using their API or hosted payment page). Using these services ensures compliance with payment regulations – e.g., Stripe and HyperPay are PCI-DSS compliant, and HyperPay specifically is optimized for the MENA region transactions.
* **Email/SMS Notifications:** To automate user communications, we’ll integrate an email service such as **SendGrid** or AWS SES for transactional emails (sign-up confirmations, booking confirmations, reminders, password resets, etc.). This is important for user experience (immediate confirmations) and doesn’t rely on manual actions. For SMS (e.g., sending a day-of-tour SMS reminder or verification codes), a service like **Twilio** can be used. These integrations can be added gradually; MVP might start with email only, then add SMS for critical alerts.
* **Maps and Location Services:** Since tours are location-based, we will integrate a mapping service to show tour locations (Google Maps or Mapbox). On the tour detail page, an embedded map can show the meeting point or area of the tour. This requires using the map provider’s API (Google Maps JS API, for example) on the front-end, and possibly geocoding addresses to coordinates on the back-end if we allow searching by location.
* **Analytics and Monitoring:** To monitor the health of the application and user behavior, we will include tools like Google Analytics (for user behavior tracking) and server monitoring (New Relic or Sentry for error tracking[[13]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=,io%2C%20Skylight)). These help in quickly identifying issues and understanding how users are using the app, guiding future improvements.

**Technology Stack Summary:** In summary, the MVP will likely utilize **React** (JavaScript/TypeScript) on the front-end, **Node.js/Express** on the back-end, **PostgreSQL** for the database, and cloud services (AWS preferred) for hosting and storage. This stack is modern, widely supported, and can meet the needs of a booking platform. For instance, a similar travel platform “TravelBid” successfully used Node.js + Vue.js and AWS with Stripe for payments[[3]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=For%20instance%2C%20TravelBid%E2%80%99s%20tech%20stack,included), validating this approach. Should STG later decide to build native mobile apps, the back-end API can be reused, or we could opt for cross-platform tools like React Native or Flutter[[14]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=If%20you%20intend%20to%20enrich,111%20or%20React%20Native) to expedite development – but thanks to the mobile-first responsive web design, the immediate need for native apps is minimized. The entire architecture is designed to be **scalable**: we can handle increasing load by vertical scaling (bigger server/DB instance) or horizontal scaling (multiple app servers behind a load balancer, read replicas for the database, etc.) as needed.

**Security Considerations:** From day one, we will enforce secure coding practices. All passwords will be stored hashed (using bcrypt or similar algorithms). The website will use HTTPS everywhere. We’ll implement proper input sanitization to prevent SQL injection or XSS attacks. User data, especially personal info and payment data, will be protected according to relevant regulations (for example, compliance with Saudi data protection laws and GDPR if Europeans use the service). Since our payments go through third parties, we avoid storing credit card details, reducing our security scope. Regular backups of the database will be scheduled and access to servers will be restricted and logged. These measures will ensure the platform maintains user trust and data integrity.

## Data Model Overview

To facilitate the implementation (especially if using Codex CLI to generate code), here is a high-level overview of the core data entities and their relationships in the system:

* **User:** represents a platform user. Attributes: user\_id (PK), name, email, password\_hash, role (e.g., “tourist”, “operator”, “admin”), language\_preference, etc. We may also store contact info (phone) and for operators, some verification status or profile description. A user can have many bookings (one-to-many). Operators (who are users with role=operator) can have many tours they offer.
* **Tour:** represents a tour/experience. Attributes: tour\_id (PK), title, description\_en, description\_ar (and potentially other languages in future, or a separate table for localized content), location (could be a city or GPS coordinates), price\_per\_person, duration, capacity (max people per session), photos (perhaps stored as URLs in a related table), category (e.g., cultural, adventure), operator\_id (FK reference to User who is the guide/operator), status (active, disabled). A Tour has many sessions/instances (dates it runs) – this can be modeled in two ways for MVP: (a) as a recurring rule plus exceptions, or (b) explicitly list each available date/time as a separate **TourSession** record. We might choose a simple **TourSession** table: with session\_id, tour\_id, date, available\_spots (initially equal to capacity, decrements on bookings). This makes real-time availability checks straightforward.
* **Booking:** represents a booking made by a user for a specific tour session. Attributes: booking\_id (PK), tour\_id (FK), user\_id (FK), session\_id (FK if sessions are separate), num\_people, booking\_date (timestamp when booked), status (confirmed, canceled, completed), payment\_status (paid, pending, refunded), amount\_paid, payment\_transaction\_id (if returned by Stripe/HyperPay for reconciliation). Bookings link users and tours. We ensure that for each session, the total num\_people booked does not exceed tour.capacity – this is enforced in the booking logic (e.g., by an atomic update or DB constraint).
* **Review:** represents a review left by a user for a tour. Attributes: review\_id (PK), tour\_id (FK), user\_id (FK), rating (1-5), comment, date\_posted. We might also store booking\_id to ensure the user actually had a booking for that tour (review gating). Reviews will be displayed on tour pages and can be moderated by admin if needed (e.g., an additional field is\_approved).
* **Payment/Payout Records:** although payment transactions might be recorded in the booking or separately. We could have a **Payment** table storing each transaction: payment\_id, booking\_id, amount, currency, status, gateway (Stripe/HyperPay), transaction\_ref. This helps in reconciliation and refunds. For payouts to operators, we might track those in an **OperatorEarnings** table (accumulating commissions), but initially this can be done via reports.
* **Miscellaneous:** We will likely have other tables such as TourPhotos (tour\_id FK and photo URL or path), TourCategories, maybe a Messages table if chat is implemented (storing message text, sender\_id, receiver\_id, timestamp). If we implement promotional codes or referral programs in future, there could be tables for those too. For Phase 1’s scope, the above core entities suffice.

This relational design can be translated into actual schema during implementation. Using an ORM (Object-Relational Mapping) in our chosen back-end framework (like Sequelize for Node or TypeORM for TypeScript, or Django ORM if Python) will speed up development and ensure consistency. The Codex CLI can use these definitions to scaffold models and CRUD operations, given the clarity of relationships.

## Implementation Approach and Timeline

With the design and specifications laid out, the development will proceed iteratively. We plan to utilize **AI-assisted development** tools (such as OpenAI’s Codex CLI) to expedite boilerplate coding – for example, generating React components or API route handlers from the described specifications. The internal development team will validate and refine the AI-generated code to ensure it meets our quality standards. The timeline roughly aligns with the phases:

* *Phase 1 (Design)* – Approximately 4-6 weeks for research, design, and prototyping. (This phase produces the current document and associated designs).
* *Phase 2 (MVP Development)* – Approximately 10-12 weeks for implementation and testing. In the first half, focus on back-end foundation and front-end of user-facing features. In the second half, build the admin portal and integrate payments, followed by testing. Before launch, we’ll conduct a **beta test** with internal users to gather feedback and fix any critical issues.
* *Phase 3 (Post-Launch enhancements)* – Begins immediately after MVP launch and continues in an agile, ongoing manner. We expect the initial “wow features” to roll out over 3-6 months post-launch, in multiple app update releases. This phase also includes scaling infrastructure as needed and optimizing based on real usage.

Throughout development, we will maintain close feedback loops with STG stakeholders. We will also ensure compliance with any regulatory requirements (for example, tourism regulations in Saudi Arabia, data residency if required to store data in-country, etc.). The deployment process will include setting up a staging environment for testing and a production environment for the live app. Using cloud DevOps tools (CI/CD pipelines), every code change can be tested and deployed seamlessly, which is especially important as we iterate quickly with the help of AI coding tools.

## Conclusion and Next Steps

In conclusion, this detailed design document provides a comprehensive plan for building STG’s tour booking platform as a web-based application. We have outlined the vision, core features, system architecture, and a roadmap for phased development. By focusing first on a browser-based solution, STG can rapidly enter the market and start generating bookings without the overhead of native app distribution – yet delivering an **app-like experience** to users via a responsive design[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts). The platform will connect travelers with authentic Saudi tours in a way that is seamless, trustworthy, and scalable.

**Next Steps:** With this blueprint in hand, the immediate next step is to kick off Phase 2 (MVP development). This involves setting up the development environment and repository, configuring the database and server infrastructure, and then coding the features as per the specifications. The internal team will leverage this document alongside the Codex CLI to generate initial code modules (for instance, create the data models from the Data Model section, scaffold API routes, and perhaps even generate basic UI components from the described wireframes). Regular check-ins will be scheduled to review progress against the design.

Once the MVP is built and internally tested, we will prepare for a soft launch – perhaps targeting a specific region or a subset of tours – to gather real-world feedback. Marketing efforts can simultaneously start to onboard tour operators and populate the platform with compelling experiences ahead of the public launch. All the while, we’ll remain adaptable: if any new insights or requirements emerge (either from stakeholders or changing market conditions), we can adjust the plan. The design as presented is robust yet flexible, providing a solid foundation to achieve STG’s vision of becoming the leading digital gateway to Saudi Arabia’s cultural and adventure experiences.

By adhering to this design and development plan, STG is well positioned to launch a successful product that delights users and scales with the growing tourism opportunities in the Kingdom. We are excited to proceed with building this platform and turning the vision into reality, one line of code at a time.

**References (for Implementation & Best Practices):**

* Onix-Systems – *How to Build a Travel Booking Website Successfully in 2025* – insights on travel platform features and tech stack[[4]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=User,items%2C%20and%20pay%20for%20them)[[3]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=For%20instance%2C%20TravelBid%E2%80%99s%20tech%20stack,included).
* Movement.so – *PWA vs Native App* – advantages of web apps (no store fees, full control over updates)[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts).
* Bacancy Tech – *React Internationalization Guide* – importance of multilingual support (89% users prefer native language)[[7]](https://www.bacancytechnology.com/blog/react-internationalization#:~:text=the%20core%20logic).
* SoftKraft – *Web Application Architecture Guide* – 3-tier architecture separation and benefits[[8]](https://www.softkraft.co/web-application-architecture/#:~:text=Modern%20web%20application%20architecture%20is,tier%20structure)[[9]](https://www.softkraft.co/web-application-architecture/#:~:text=,depending%20on%20the%20exact%20architecture).

[[1]](https://movement.so/app-stores-vs-pwa#:~:text=As%20creators%2C%20brands%2C%20and%20businesses,store%20restrictions%20or%20revenue%20cuts) You Don't Need the App Stores

<https://movement.so/app-stores-vs-pwa>

[[2]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=%2A%20Cloud,io%2C%20Skylight) [[3]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=For%20instance%2C%20TravelBid%E2%80%99s%20tech%20stack,included) [[4]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=User,items%2C%20and%20pay%20for%20them) [[5]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=One%20of%20Onix%E2%80%99s%20clients%2C%20a,the%20solutions%20they%20use%20are) [[6]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=Instant%20messages%20can%20inform%20users,engagement%20and%20winning%20customer%20loyalty) [[10]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=Jest%2C%20Kendo%20%2A%20Back,GCP%29%2C%20Heroku) [[11]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=%2A%20Back,NET%2C%20Micronaut%2C%20gRPC%2C%20RxJava%2C%20CloudCMS) [[12]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=,s%3A%20PayPal%2C%20Braintree%2C%20Stripe%2C%20Adyen) [[13]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=,io%2C%20Skylight) [[14]](https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business#:~:text=If%20you%20intend%20to%20enrich,111%20or%20React%20Native) How to Build a Booking Website for a Travel Business in 2025

<https://onix-systems.com/blog/how-to-make-a-booking-website-for-a-tourism-business>

[[7]](https://www.bacancytechnology.com/blog/react-internationalization#:~:text=the%20core%20logic) React Internationalization: Guide for Multilingual Apps

<https://www.bacancytechnology.com/blog/react-internationalization>

[[8]](https://www.softkraft.co/web-application-architecture/#:~:text=Modern%20web%20application%20architecture%20is,tier%20structure) [[9]](https://www.softkraft.co/web-application-architecture/#:~:text=,depending%20on%20the%20exact%20architecture) Web Application Architecture [Complete Guide & Diagrams]

<https://www.softkraft.co/web-application-architecture/>