**1. System Architecture**

| **Layer** | **Technology Suggestion** |
| --- | --- |
| Frontend (Web) | React.js / Angular / Vue.js |
| Backend (API) | Node.js with Express / Spring Boot (Java) / Django (Python) |
| Database | PostgreSQL / MySQL for relational data |
| Authentication | JWT with OAuth2 support |
| Cloud Hosting | AWS (EC2, RDS, S3) / GCP / Azure |
| Payment Integration | Razorpay / Stripe |
| PDF Generation | jsPDF / wkhtmltopdf / WeasyPrint |
| Geolocation | Google Maps API |
| Media Upload | AWS S3 or Cloudinary |

**2. Customer Portal Implementation**

**🔍 Tour Search**

* **Features**:
  + ElasticSearch integration for fast filtering
  + Auto-complete location using Google Places API

**📦 Booking Flow**

* **Step 1**: Token amount booking stored with PENDING\_CONFIRMATION status.
* **Step 2**: Operator confirmation triggers email/SMS.
* **Step 3**: If declined, refund token via payment gateway API.
* **Step 4**: Full payment option enabled after confirmation.

**🧾 PDF Handling**

* Use a microservice to:
  + Generate booking receipt/invoice (PDF)
  + Generate itinerary PDF (HTML-to-PDF)

**📂 Booking Management**

* DB Tables: bookings, users, tour\_status, payments
* Statuses: PENDING, CONFIRMED, CANCELLED, REFUNDED

**❌ Cancellation and Refund**

* UI: Cancel button visible if within policy limits
* Backend: Trigger refund API if is\_refundable = true
* Update status to REFUNDED post success

**3. Operator Extranet Portal Implementation**

**🏞 Tour Builder Module**

* Multi-step form:
  1. Basic Info (title, description, media)
  2. Locations (Google Maps integration)
  3. Itinerary builder (day-wise)
  4. Meals, Hotels, Policies

**🏨 Accommodation & Halt Setup**

* DB: accommodations, halts, meals
* Validation:
  + All locations used must exist in operator's list

**4. Security & Scalability Guidelines**

* **Security**:
  + Input validation (backend and frontend)
  + HTTPS & CORS policy setup
  + Role-based access control (RBAC)
  + Store sensitive data with encryption (e.g., passwords, payment tokens)
* **Scalability**:
  + Use Redis/Memcached for caching heavy reads (tour listings)
  + Use CDN (Cloudflare, AWS CloudFront) for static content
  + Paginate all large lists (bookings, tours)

**5. DevOps & CI/CD**

* CI/CD Tools: GitHub Actions / GitLab CI / Jenkins
* Dockerize the app
* Environments:
  + dev → staging → prod
* Use monitoring tools like Grafana + Prometheus or ELK stack