

# Slices

## Slice 1 — Foundation + Auth (JWT) + Roles

**Goal:** users can sign up/login securely, and the app knows who you are + what role you have.

### Includes

- Project setup (MERN structure, envs, scripts)
- User model: `name, email, password, role, isApproved`
- Password hashing (bcrypt) + JWT auth
- Auth endpoints (register/login/me/logout if you do it)
- Route protection middleware + role guards
- Basic UI scaffolding + auth screens (login/register) + protected routing
- Seed admin (or admin creation strategy)

**Exit check:** You can log in as **Admin / Provider / Customer** and protected routes work.

---

## Slice 2 — Provider Core: Service CRUD

**Goal:** providers can fully manage their services; customers can browse them.

### Includes

- Service model: `title, description, price, duration, providerId`
- Service APIs (CRUD, provider-scoped listing)
- Validation + error handling (clean API responses)
- Frontend:
  - Customer service browsing/list + details
  - Provider "My Services" dashboard section (create/edit/delete)

**Exit check:** Provider can create a service and customer can see it.

---

## Slice 3 — Booking System + Status Flow + Admin Approval Workflow

**Goal:** real “booking management” works end-to-end with approvals and statuses.

### Includes

- Booking model: `customerId, serviceId, providerId, date, status`
- Booking APIs:
  - customer creates booking
  - provider updates status (e.g., pending/accepted/completed/cancelled)
  - customer views bookings
  - provider views bookings for their services
- Admin module:
  - view users/providers
  - approve providers ( `isApproved=true` ) + restrict provider actions until approved
- Frontend dashboards:
  - Customer dashboard: my bookings + statuses
  - Provider dashboard: bookings to manage + status updates
  - Admin dashboard: approve providers + monitor basics

**Exit check:** Customer books → Provider sees it → Provider updates status → Customer sees status; Admin approval gates provider features.

---

## Slice 4 — Production Readiness: UX polish + Testing + Deployment + Docs

**Goal:** nothing breaks on push; app is presentable and shippable.

### Includes

- Responsive dashboards (mobile-friendly)
- API hardening: validation, consistent error format, security basics (env vars, no secrets committed)

- Testing flows with Postman + basic frontend flow testing
- Deployment (frontend + backend) + finalize envs
- Final deliverables: repo clean, live links, short final presentation

**Exit check:** CI passes, deployments stable, and you can demo all role flows confidently.