University Canteen Smart Queue (Prototype)

README.md content

```
# University Canteen Smart Queue (Prototype)

This is the prototype implementation of the **University Canteen Smart Queue & Token
Management System** using the **MERN stack**.
It is designed to handle ~100 concurrent users on free-tier services and can be scaled to
production-ready architecture for 100k+ users.

## Tech Stack
- **Backend:** Node.js + Express + MongoDB Atlas (Free tier)
- **Frontend:** React (Vite or CRA), hosted on Vercel/Netlify
- **Realtime:** Socket.IO (rooms per vendor), Redis adapter (for scaling)
- **Auth/OTP:** OTP via Outlook SMTP (NodeMailer)
- **Queueing & Jobs:** BullMQ (Redis-based job queue, optional in prototype)
- **CI/CD:** GitHub Actions (free)
- **Monitoring:** Basic logging (winston), Sentry (free tier)

## Quick Start
1. Clone repo & install dependencies
2. Configure environment variables (`.env`):
- `MONGO_URI` (MongoDB Atlas free cluster)
- `OUTLOOK_USER` + `OUTLOOK_PASS` (Outlook SMTP credentials)
- `JWT_SECRET` (for token signing)
3. Run backend: `npm run server`
4. Run frontend: `npm run client`

## Features (Prototype)
- Student: Signup/login with OTP, book/cancel tokens, realtime queue updates
- Vendor: Serve next, cancel tokens, see live queue
- Admin: Add vendors, view audit logs (basic prototype)

---
```

ROADMAP.md content

```
# Roadmap: Smart Queue System (MERN)

## Phase 1 — Student backend
- Express API: signup/login (OTP), token create/cancel/status
- MongoDB collections: users, tokens, vendors, logs

## Phase 2 — OTP via Outlook & realtime
- NodeMailer + Outlook SMTP for OTP
- Socket.IO for live queue updates (vendor-based rooms)

## Phase 3 — Student frontend
- React app with OTP login, token booking, live queue view
- Deploy to Vercel/Netlify

## Phase 4 — Vendor backend
```

- Vendor auth & APIs for queue management
- Emit realtime updates to student clients

## Phase 5 — Vendor frontend
- Vendor dashboard: serve next, cancel, daily counts

## Phase 6 — Admin backend
- CRUD for vendors, audit logs, daily reports

## Phase 7 — Admin frontend
- Admin portal: manage users/vendors, view logs

## Phase 8 — Hardening (small production)
- Cloudflare for CDN/protection
- Mongo indexes + rate limiting
- Logging (winston), Sentry free

## Phase 9 — Load testing & optimization
- k6/Locust load tests
- Add caching & background jobs

## Phase 10 — Production scale plan (100k+ users)
- Migrate DB to larger cluster / self-hosted Mongo
- Redis for pub/sub scaling (Socket.IO adapter)
- Replace Outlook free SMTP with university SMTP or paid transactional provider
- Observability (Prometheus/Grafana)