Campus Queue Management System – Project Knowledge Base

Purpose of this file
This document is your **single source of truth** for understanding, maintaining, updating, and explaining your project in the future.
Whenever you come back after weeks/months, read this file first.

---

1. Project Overview

Project Name: Campus Queue Management System
Type: Full-Stack Web Application (Production Deployed)
Core Idea: A real-time queue management system for campuses where: - Staff manages department queues - Students log in and join queues - Guests join queues via QR code - Live updates are sent to all users

---

2. Tech Stack (What & Why)

Frontend

- **React (Vite)** – Fast build, modern React
- **Tailwind CSS** – Utility-first styling
- **React Router (HashRouter)** – Required for Render SPA routing
- **Axios (centralized API)** – Clean API calls
- **Socket.io Client** – Real-time updates

Backend

- **Node.js + Express** – REST API
- **Socket.io** – Real-time communication
- **JWT** – Authentication
- **CORS + Cookies** – Secure cross-origin access

Database

- **MongoDB Atlas** – Cloud database (always running)

Deployment

- **Render** – Frontend + Backend hosting
- **GitHub** – Source control & deployment trigger

---

3. High-Level Architecture

```
Browser (User)
   ↓
Frontend (Render)
   ↓ API calls / Socket
Backend (Render)
   ↓
MongoDB Atlas
```

Key idea: **Nothing runs on your laptop after deployment**.

---

4. User Roles & Flows

4.1 Admin

- Logs in
- Creates departments
- Assigns staff

4.2 Staff

- Logs in
- Manages queue (open/close)
- Calls next ticket
- Completes ticket
- Generates QR code for guests

4.3 Student

- Logs in
- Joins department queue
- Sees ticket status

4.4 Guest (QR Based)

- Scans QR
- Joins queue without login
- Gets ticket
- Receives live updates

---

5. Routing Strategy (IMPORTANT)

Why HashRouter?

Render serves static files and doesn't know React routes.

HashRouter ensures:

```
/#/guest/join
```

Always loads `index.html`

Impact

- Query params live in `location.hash`
- NOT in `location.search`

This is why QR parsing required special logic.

---

6. Environment Variables (Critical)

Backend (.env on Render)

```
PORT=5000
MONGO_URI=your_mongodb_atlas_url
JWT_SECRET=your_secret
FRONTEND_BASE_URL=https://campus-queue-management-system-1.onrender.com
```

Frontend (.env)

```
VITE_API_URL=https://campus-queue-management-system.onrender.com/api
```

---

7. Centralized API Pattern

File: `frontend/src/services/api.js`

Why this exists: - Avoid localhost mistakes - Handle CORS & credentials - One place to change backend URL

All API calls MUST use this.

---

8. Socket.io Design

Rooms

```
department_<departmentId>
```

Events

- `join_department`
- `ticket_called`
- `ticket_completed`
- `queue_status_changed`

Rule

- Join room only ONCE (useRef guard)

---

9. QR Code Flow (Guest)

1. Staff generates QR
2. Backend creates URL:

```
/#/guest/join?departmentId=XXX
```

3. QR encodes this URL
4. Guest scans QR
5. GuestJoin parses `location.hash`
6. Guest joins queue

---

10. Guest Token System

- Guests don’t log in
- Backend issues `guestToken`
- Stored in `localStorage`
- Sent via header: `x-guest-token`

Used for: - Restore ticket - Cancel ticket

---

11. Common Bugs You Fixed (Remember These)

❌Localhost in production

✔️Fixed by centralized API

❌Mobile QR not working

✔️HashRouter query parsing

❌CORS issues

✔️Correct `FRONTEND_URL` + credentials

❌Socket duplicate joins

✔️ `useRef` guard

---

12. Deployment Flow (Mental Model)

```
Edit Code
→ git add .
→ git commit
→ git push
→ Render redeploys automatically
```

Your laptop is NOT involved after this.

---

13. How to Update in Future

1. Open this file
2. Identify layer:
3. UI
4. API
5. Socket
6. DB
7. Make change
8. Test locally
9. Push to GitHub
10. Redeploy on Render

---

14. How to Explain This Project (Interview / Demo)

> “This is a production-ready campus queue management system built with MERN and Socket.io. It supports staff-controlled queues, QR-based guest onboarding, and real-time updates across devices. The entire system is deployed on cloud infrastructure with no local dependency.”

---

15. Final Notes (For Future You)

- This project is **complete**
- Don’t underestimate it
- You solved real production problems
- This is resume-worthy

When in doubt → read this file again.