Student Attendance System — Design & Deployment Documentation

1. Project Overview

The Student Attendance S is a full-stack web application enabling educators to record, visualize, and manage student attendance in real time. Key features include:
- Daily attendance entry via a modal interface
- Dashboard with charts and statistics to monitor trends
- Persistent storage of attendance records
- Responsive design for desktop and mobile use

2. Technology Stack

Layer	Technology
Frontend	React, TypeScript, Tailwind CSS
Backend	Node.js, TypeScript, Vite
Database	SQLite (via Drizzle ORM)
Build	Vite, PostCSS
Version Control	Git

3. System Architecture

[Browser/Client] ←(HTTP/JSON)→ [Node/Vite Server] ←(Drizzle ORM)→ [SQLite Database]

1. Frontend: Sends/receives JSON to backend API and renders charts, tables, modals.
2. Backend: Exposes RESTful endpoints and handles business logic, DB I/O.
3. Database: Stores attendance records and metadata; schema centralized in shared code.

4. Frontend Design

- Entry Points: index.html, main.tsx, App.tsx
- Key Component Groups:
  1. Dashboard Widgets (src/components/dashboard/): AttendanceChart, StatsCards, RecentClasses
  2. Attendance Tools (src/components/attendance/): TakeAttendanceModal
  3. Shared UI Kit (src/components/ui/): Button, Alert, Badge, Avatar
- Routing & State: React Router or Vite file-based routing; local component state + data fetch hooks.

5. Backend Design

- Server Entry: server/index.ts
- Routing: server/routes.ts (GET /attendance, POST /attendance, GET /stats)
- Data Layer: server/storage.ts with Drizzle ORM; shared schema in shared/schema.ts
- Hot Reloading: Vite dev server proxies API and refreshes on change.

6. Database Schema

Located in shared/schema.ts (Drizzle + TypeScript):

```
export const attendance = pgTable("attendance", {
  id: serial("id").primaryKey(),
  studentId: integer("student_id").notNull(),
  date: date("date").notNull(),
  status: text("status").notNull(),  // "present" | "absent"
  note: text("note")              // optional remarks
});
```

7. Core Workflow & Logic

1. Recording Attendance:
   - Open Take Attendance modal, select student statuses, submit via POST.
2. Viewing Insights:
   - Dashboard GETs from /stats and /attendance; charts and cards compute trends.

8. Deployment & Platform Justification

Development & Build Commands:

```bash
npm install
npm run dev
npm run build
```

Replit-Only Compatibility:

1. Simplified environment: pre-configured IDE with TypeScript, Node.js, Vite, Tailwind, SQLite.
2. Persistent cloud hosting: live URL sharing without VPS setup.
3. Custom Replit config: uses .replit and live preview settings not native to local IDEs.

Note: The codebase is platform-agnostic; porting requires updating connection strings and scripts.

9. Future Enhancements

- Authentication & Roles (teacher/admin)
- Report Export (PDF, Excel)

- Push Notifications for unexcused absences
- Mobile App wrapper (React Native or PWA)