System Design Document — Sprint 1

Giuseppe D’Orazio

Ravneet Deol

Victoria Bolognese

Araz Manouchian

Table of Contents

1. Purpose and Scope

This document describes the system design of the ChromeNotes project for Sprint 1. It contains CRC cards for main classes/components, environment and dependencies, architecture overview, system decomposition, error and exception handling strategy, API contract (endpoints used in this sprint), data model.

2. High-level Description

The primary components of the backend are:

- **Express server** (server.js) — sets up middleware, CORS policy, JSON parsing, and routes.
- **Router** (routes/notesRoutes.js) — maps HTTP endpoints to controller functions.
- **Controller** (controllers/notesController.js) — implements business logic for notes operations.
- **Model** (models/Note.js) — Mongoose model and schema for Notes.
- **DB connector** (config/db.js) — connects to MongoDB using mongoose and MONGO_URI.

These components collaborate to handle incoming HTTP requests from the frontend and persist/retrieve data from MongoDB.

The primary components of the frontend are:

- Components folder - React components which are in their own files
- Pages - all the different pages that can be rendered

3. CRC Cards

Class Name: Server (server.js)
Responsibilities: - Configure Express app and middleware (JSON, CORS). - Initialize route mounting (/api/notes). - Launch HTTP server on a configured port. - Trigger database connection on startup.
Collaborators: - notesRoutes (router)

- connectDB (config/db.js)

Class Name: NotesRoutes (routes/notesRoutes.js)

Responsibilities:

- Expose REST endpoints for notes: GET /, POST /, DELETE /:id.
- Route requests to the appropriate controller functions.

Collaborators:

- notesController (controllers/notesController.js)

Class Name: NotesController (controllers/notesController.js)

Responsibilities:

- Implement business logic for creating, retrieving, and deleting notes.
- Validate input and IDs where appropriate.
- Return proper HTTP status codes and JSON responses.
- Log and handle internal errors.

Collaborators:

- Note model
- mongoose for ID validation

Class Name: Note (models/Note.js)

Responsibilities:

- Define the Note schema (title, content, timestamps).
- Provide persistence (save, find, findByIdAndDelete) via Mongoose.

Collaborators:

- NotesController

Class Name: DBConnector (config/db.js)

Responsibilities:

- Connect to MongoDB using the MONGO_URI environment variable.

- Provide connection lifecycle (connect, error handling, exit on fatal error).
Collaborators: - server.js - mongoose

Class Name: NotesList (components/NotesList.jsx)
Responsibilities: - Fetch notes from the backend and display all notes in a grid layout. - Trigger the deletion of notes and update on the UI.
Collaborators: - HomePage - axios.js

ClassName: HomePage
Responsibilities: - Render NotesList or NoteForm depending on user input
Collaborators: - NotesList - NoteForm

ClassName: NoteForm
Responsibilities: - Display a form which can be used to either add a note or edit a note. - Has text boxes for the title and the content of a note
Collaborators: - HomePage

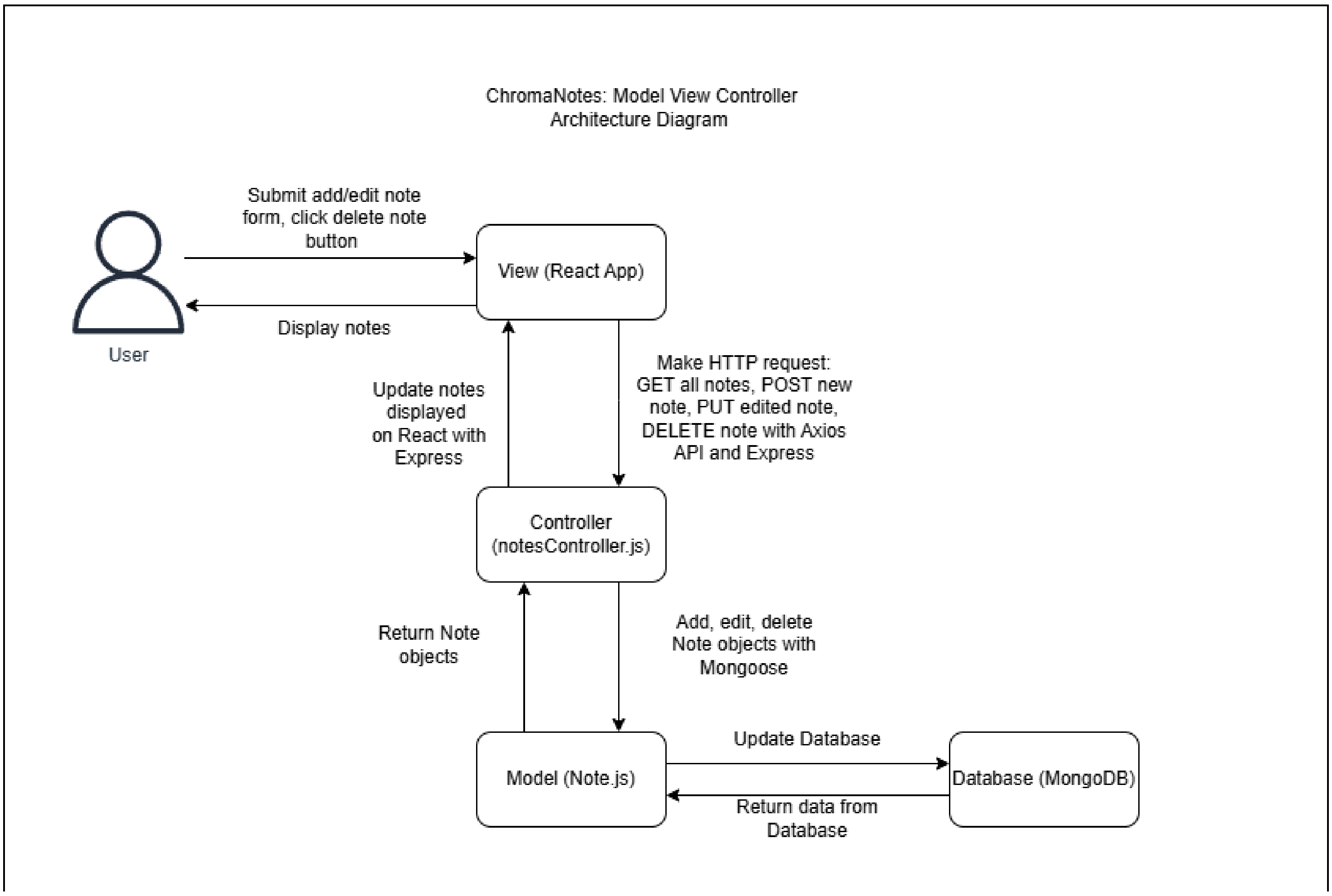
4. System Interaction with the Environment

Operating System: Any OS that supports Node.js (Linux, macOS, Windows). Development tested on Linux/WSL and Windows.
Runtime / Language: Node.js. Uses ECMAScript modules (type: module in package.json or .mjs usage).
Package Manager: npm
Frameworks / Libraries:

- Express
- Mongoose
- dotenv
- Cors
- Axios
- Hot-toast
- React-router
- Tailwind

Database: MongoDB. MONGO_URI is required in environment variables.
Frontend (caller): React dev server running on http://localhost:5173 (CORS origin configured for development).
Network: Server expects the MongoDB server to be reachable from the host running the backend. If using MongoDB Atlas, ensure IP whitelist or proper credentials.
Ports: Backend default port is 5001. Frontend on 5173 (Vite) used during development.

5. Architecture


ChromaNotes: Model View Controller Architecture Diagram
User
Submit add/edit note form, click delete note button
Display notes
View (React App)
Make HTTP request: GET all notes, POST new note, PUT edited note, DELETE note with Axios API and Express
Update notes displayed on React with Express
Controller (notesController.js)
Return Note objects
Add, edit, delete Note objects with Mongoose
Model (Note.js)
Update Database
Return data from Database
Database (MongoDB)

6. System Decomposition

Backend

server.js — Express app setup, middleware, and server bootstrap.

routes/notesRoutes.js — HTTP route mapping for notes endpoints.

controllers/notesController.js — Controller logic for GET, POST, DELETE.

models/Note.js — Mongoose schema and model definition.

config/db.js — MongoDB connection helper.

Frontend

src/components/NotesList.jsx - React component that displays all created notes

pages/HomePage.jsx - React component for the main page of ChromaNotes, has a header with Add Note button

App.jsx - Main component that gets rendered

Index.css - Css file, only used to import Tailwind
Main.jsx - File that gets ran by index.html, renders app.jsx
Index.html - root html file

7. API Contract

Base path: /api/notes

1. **GET /api/notes/**
    - Description: Return all notes, sorted by creation timestamp (server can implement sort later).
    - Response: 200 OK JSON array of notes.
2. **POST /api/notes/**
    - Description: Create a new note.
    - Request JSON body: { "title": "string", "content": "string" } (both required)
    - Responses:
        - 201 Created — returns the newly-created note object.
        - 400 Bad Request — when required fields are missing (to be implemented).
        - 500 Internal Server Error — server error.
3. **PUT /api/notes/:id**
    - Description: Update an existing note by its ID
    - Request JSON body: { "title": "string", "content": "string" } (at least one required)
    - Responses:
        - 200 OK — returns the updated note project
        - 404 Not Found — when the note with the given ID does not exist
        - 500 Internal Server Error — server error
4. **DELETE /api/notes/:id**
    - Description: Delete a note by MongoDB ObjectId.
    - Responses:
        - 200 OK — on successful deletion, returns deleted note and message.
        - 400 Bad Request — invalid note id format.
        - 404 Not Found — note not found with given id.
        - 500 Internal Server Error — server error.

8. Data Model

Note

- title: String (required)
- content: String (required)
- createdAt and updatedAt (timestamps auto-managed)

9. Error & Exceptional Case Strategy

Missing or invalid environment variables (e.g., MONGO_URI):

- Detection: connectDB() will throw on a failed connection.
- Response: server.js should log an explicit error. Process can exit(1) on fatal start-up errors (current code calls process.exit(1) on connect failure).

Invalid JSON payload / missing fields for POST:

- Detection: Validate req.body for title and content in createNote() before creating model.
- Response: Return 400 Bad Request with message indicating missing fields.

Invalid MongoDB ObjectId on DELETE:

- Response: If invalid, return 400 Bad Request with a helpful message.

Note not found when deleting:

- Response: Return 404 Not Found.
- Notification: Toast notification “Failed to delete note”

Fail to connect to server

- Notification: Toast notification “Failed to load notes”