System Design Document — Sprint 2

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. 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.

New additions for Sprint 1 include:

- Tagging system (Tag model + tag CRUD + tag-to-note linking)
- Pin/unpin notes
- Custom note styling (color, font style, font size)
- New routes and controller logic to support these features

2. High-level Description

The primary components of the backend are:

Express server (server.js)
Sets up Express app, JSON parsing, CORS, tag and note routes, and database connection.
Notes Router (routes/notesRoutes.js)
Handles note CRUD + tag-related note endpoints + pinned notes.

Tags Router (routes/tagsRoutes.js)
Handles CRUD operations for tags.

NotesController (controllers/notesController.js)
Implements business logic for notes:

- create, read, update, delete
- add/remove tags
- search notes by tag

- toggle pin status
- get pinned notes

TagsController (controllers/tagsController.js)
Implements logic for tag creation, listing, and deletion.

Models (models/Note.js, models/Tag.js)
Mongoose schemas for notes and tags.

DBConnector (config/db.js)
Connects backend to MongoDB via Mongoose.

The primary components of the frontend are:

- Components folder - React components which are in their own files
    - EditTagsForm.jsx - Component for users to add or delete tags for their notes. Shows when the “edit tags” button in the sidebar is clicked.
    - Header.jsx - Component that shows the title of the website (ChromaNotes), and the Add Note button.
    - NoteForm.jsx - Component for users to fill out information when adding a new note, or editing an existing note. Users can write a title and description for the note in different fonts, select a colour for the note, and assign tags to the note.
    - NotesList.jsx - Component that displays all notes that user has added. The user can sort the notes using the Sort button by date created or date modified, and they can pin certain notes so that they are always listed first.
    - SideBar.jsx - Component that has a Pinned button for viewing pinned notes only, an Edit Tags button to add and delete tags, and a button for each existing tag, to only see those notes which contain the tag.
    - StyleControl.jsx - Component to handle the colour choosing feature within NoteForm.
- Pages folder - all the different pages that can be rendered
    - HomePage.jsx - the main page of the app that controls the api requests, and includes which components are rendered

3. CRC Cards

Class Name: Server (server.js)

Responsibilities: - Configure Express app, JSON parsing, and CORS. - Mount routes: /api/notes, /api/tags. - Initialize MongoDB connection. - Launch server on port 5001
Collaborators: - notesRoutes - tagsRoutes - connectDB

Class Name: NotesRoutes (routes/notesRoutes.js)
Responsibilities: - Map all note-related API endpoints: - GET, POST, PUT, DELETE - Add/remove tags - Get notes by tag - Pin/unpin notes - Get pinned notes
Collaborators: - notesController

Class Name: TagsRoutes (routes/tagsRoutes.js)
Responsibilities: - Provide CRUD endpoints for tags.
Collaborators: - TagsController

Class Name: NotesController (controllers/notesController.js)

Responsibilities:

- Implement create, update, delete, and retrieval of notes.
- Manage tags on notes (add/remove/filter).
- Implement pinned note functionality.
- Handle custom note styling (color, fontStyle, fontSize).
- Input validation and error handling

Collaborators:

- Note model
- Tag model
- Mongoose

Class Name: TagsController (controllers/tagsController.js)

Responsibilities:

- Get all tags
- Create tags
- Delete tags
- Cascade removal of tag references from notes

Collaborators:

- Tag model
- Note model
- Mongoose

Class Name: Note (models/Note.js)

Responsibilities:

- Define note schema fields:
    - title
    - content
    - tags (ObjectId[])
    - pinned
    - color

• fontStyle • fontSize • timestamps
Collaborators: - NotesController

Class Name: Tag (models/Tag.js)
Responsibilities: - Define tag schema (name, unique).
Collaborators: - TagsController - 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

ClassName: HomePage (frontend/src/pages/HomePage.jsx)
Responsibilities: - Render NotesList or NoteForm depending on user input, display the Header and Sidebar components, display the EditTagsForm modal component - Handle all API requests
Collaborators:

- NotesList
- NoteForm
- EditTagsForm
- Header
- Sidebar
- axios.js

Class Name: NotesList (frontend/src/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.
- Clicking the edit button on any note displays the NoteForm component to edit that note.

Collaborators:

- HomePage

ClassName: NoteForm (frontend/src/components/NoteForm.jsx)

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

ClassName: EditTagsForm (frontend/src/components/EditTagsForm.jsx)

Responsibilities:

- Display a form which can be used to either add or delete tags.
- Has a text box for the new tag name and a list of all tags with a delete button next to each one

Collaborators: - HomePage

ClassName: Header (frontend/src/components/Header.jsx)
Responsibilities: - Display the title of the website and the add note button. - Clicking the add note button displays the NoteForm component.
Collaborators: - HomePage

ClassName: Sidebar (frontend/src/components/Sidebar.jsx)
Responsibilities: - Display buttons to only show pinned notes, only show specific tagged notes, or show the EditTagsForm component.
Collaborators: - HomePage

ClassName: StyleControl (frontend/src/components/StyleControl.jsx)
Responsibilities: - Display buttons for various colours that the user can set for their note.
Collaborators: - NoteForm

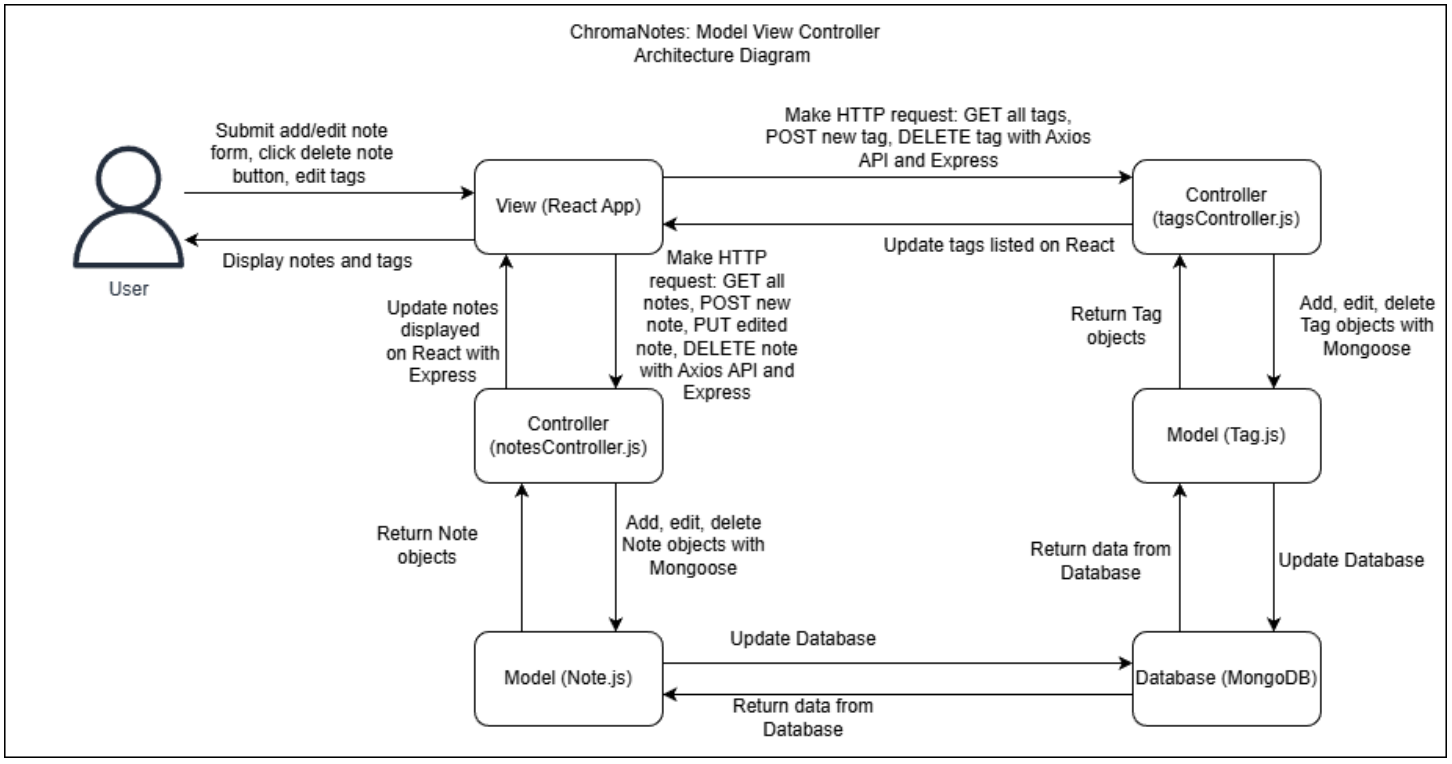
4. System Interaction with the Environment

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



6. System Decomposition

Backend

server.js — Express setup, CORS, routes, DB connection.
routes/notesRoutes.js — Note CRUD + pin + tag operations.
routes/tagsRoutes.js — Tag CRUD routes.
controllers/notesController.js — Logic for notes, including tags & pinning.
controllers/tagsController.js — Logic for tag creation and deletion.
models/Note.js — Note schema with tags + pinned + styling.
models/Tag.js — Tag schema.
config/db.js — MongoDB connector.

Frontend

src/components/NotesList.jsx - Displays all created notes, user can edit or delete notes
src/components/EditTagsForm - User can add or delete tags
src/components/Header - User can click add note button
src/components/Sidebar - User can click edit tags button
src/components/NoteForm - User fills out information for new or edited note
src/components/StyleControl - User picks a colour for their note
pages/HomePage.jsx - Main page of ChromaNotes, displays all components and handles all API requests to backend
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/**
    - Returns all notes.
2. **GET /api/notes/pinned**
    - Returns all pinned notes.

3. **POST /api/notes/**
    - Create a note.
4. **PUT /api/notes/:id**
    - Update note.
5. **DELETE /api/notes/:id**
    - Delete note.
6. **PATCH /api/notes/:id/pin**
    - Toggle the pinned state of a note.
7. **GET /api/notes/tag/:tagId**
    - Get all notes containing a tag.
8. **POST /api/notes/:id/tags**
    - Add a tag to a note.
9. **DELETE /api/notes/:id/tags/:tagId**
    - Remove a tag from a note.

Base path: /api/tags

1. **GET /api/tags/**
    - Return all tags.
2. **POST /api/tags/**
    - Create a tag.
3. **DELETE /api/tags/:id**
    - Delete a tag and remove it from all associated notes.

8. Data Model

Note

Field	Type	Required	Default
title	String	Yes	–
content	String	Yes	–
tags	ObjectId[]	Np	[]
pinned	Boolean	No	false
color	String	No	“#FFFFFF”

fontStyle	String	No	“Ariel
fontSize	Number	No	14
createdAt	Date	Auto	–
updatedAt	Date	Auto	–

Tag

Field	Type	Required
name	string	Yes, unique

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:

- POST /notes validates required fields
- Response: Return 400 Bad Request with message indicating missing fields.

Invalid MongoDB ObjectId on DELETE:

- For note or tag endpoints
- Return 400 with helpful message

Note or Tag not found when deleting:

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

Tag Deletion Cascade:

When deleting a tag:

- Tag removed from DB

- Tag removed from all notes using $pull

Fail to connect to server

- Notification: Toast notification “Failed to load notes”