ChatGpt Low Level Design Document

Sorry for the little delay, I am building this project by my own without using any AI (for coding purposes) that why it take time,
I hope you understand me and like my project.

Task 0:

-> Read the code of conduct and learn how production grade code look like,
-> Use Best Practices of NextJs,

Task zero Done ✅

Task 1:
-> Pixel - perfect chatGPT UI -> Done from the chat functionality side only, not other features like search chat, library, Sora, because as I understood this assignment is more focused on chat functionality.✅

-> Full functional Chat using Vercel AI SDk -> Implement completely.✅ (used model -> m`models/gemini-2.0-flash-exp )`

-> Chat memory, file.image upload, message editing. -> Completed.✅

-> Backend with MongoDB, cloudinary integration -> completed. ✅

-> Deployed on vercel -> completed -> ✅
[(https://chat-gpt-gamma-five-82.vercel.app)](https://chat-gpt-gamma-five-82.vercel.app)

-> Complete README and environment setUP -> Completed -> ✅
-> well documented, maintainable, modular codebases,
I break things into multiple files and functions, they work independently on a single task and then I merge output from all these functions. (try to follow monorepo kind architecture)

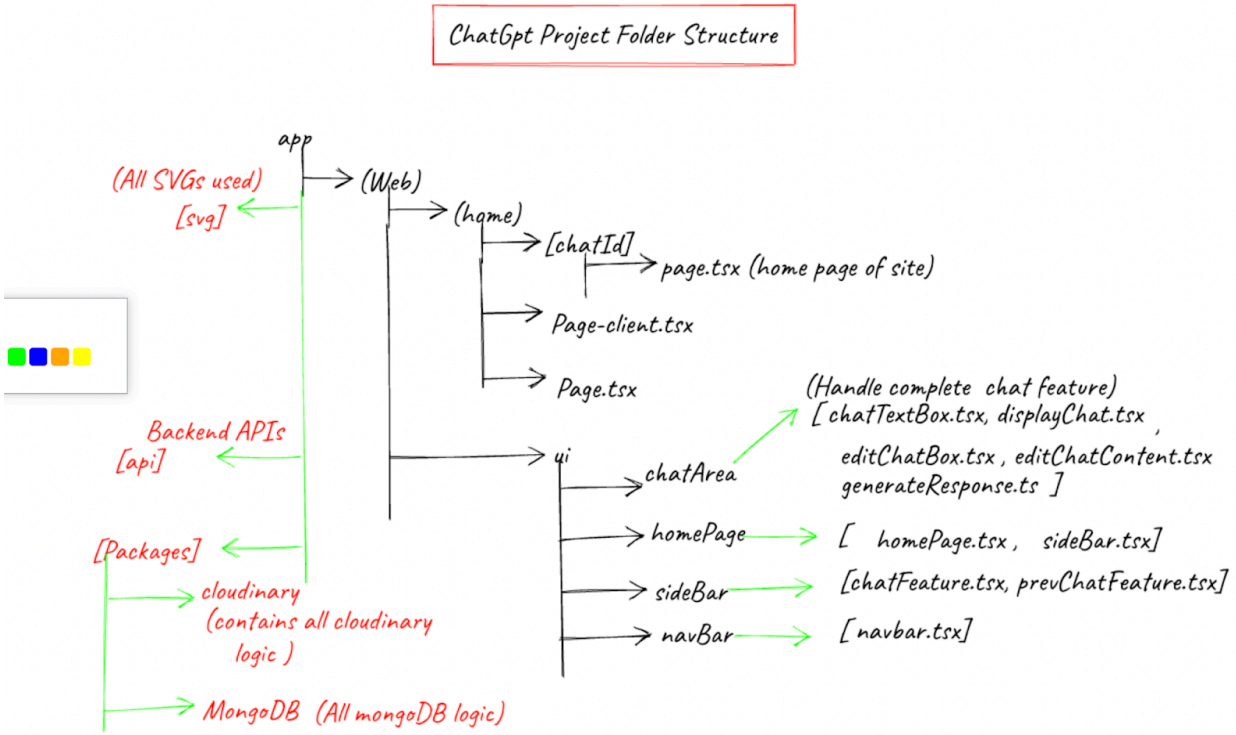
Technologies used:

Frontend -> NextJS, TailwindCSS (without any AI)
Backend -> ExpressJs, NodeJs
Auth -> Clerk,
DataBase -> MongoDB,
Multer, Cloudinary, lucide-react, Zod,

-> Auth -> Use clerk

Folder Structure :

(build on my whiteboard application
https://white-board-web-application.vercel.app pls visit this also 😄)



Backend routes:

/api/addChat -> To store new chat into the backend or update chatData if chat already exist .

Payload ->
{`title, userId, question, chatId, answer,fileUrls` }

Title -> title of chat,
userId -> comes from Clerk,
Question -> user input ,
chatId -> Id of current chat, (If this is new chat, then chatId -> new)
Answer -> answer -> response generated by vercel SDK,
fileUrls -> cloudinary URL (If user uploads file input)

/api/chat/[userId] => Fetch chat by userId,

Payload -> userId on params.

/api/editChat -> To update exist chats.

Payload -> { **`chatId, messageId, question, answer, title}`**

chatId -> Id of that particular chat (see on browser URL)
messageId -> Id of message in this chat.
Question -> updated question from user.
Answer -> updated answer for updated question.
Title -> updated title of that chat.

/api/getChatByChatId -> return chat by chatId

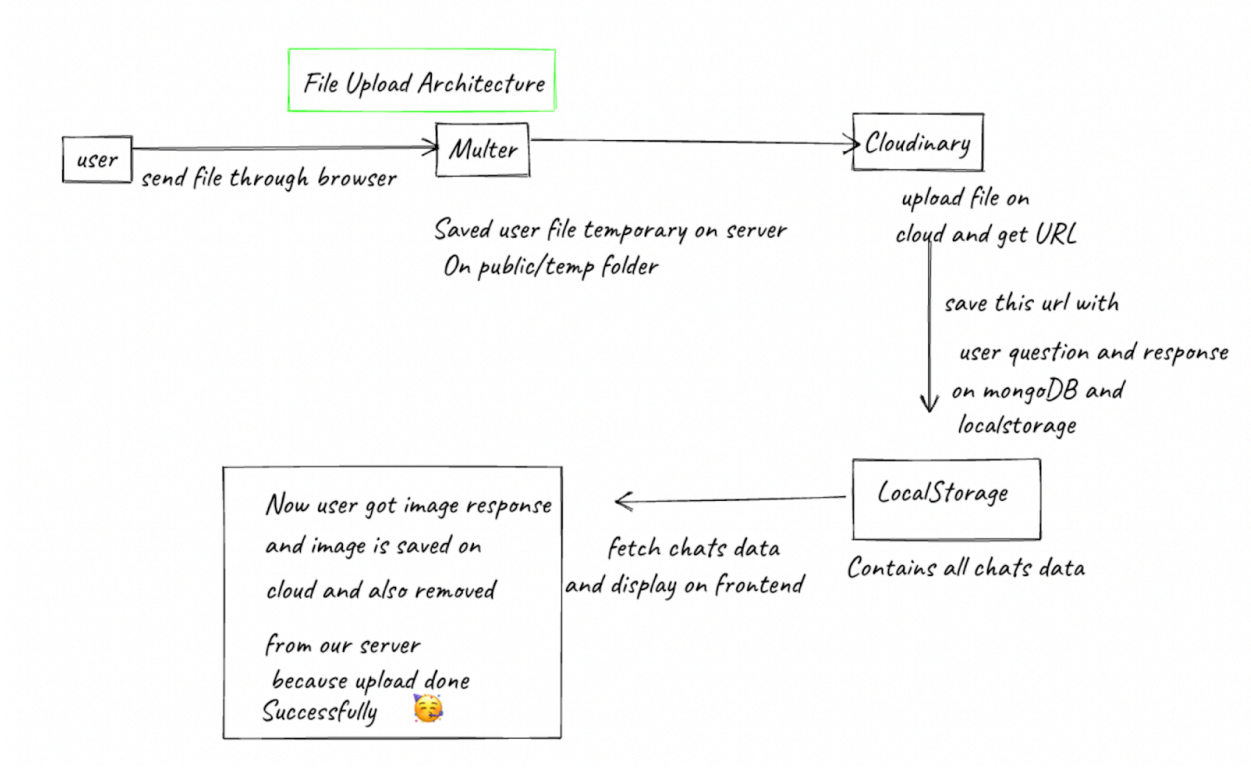
Payload -> chatId -> in params.

/api/upload -> Upload file on cloudinary

Payload -> file/image

Production grad architecture to handle file upload

Return -> file cloudinary url



MongoDB Schema

One document -> Chat

Fields ->

```
const ChatsSchema: Schema<Chats> = new mongoose.Schema({
    title:{
        type: String,
        required: [true, "Title is required"],
        default: "Chat"
    },
    userId: {
        type: String,
        required: [true, "UserId is required"]
    },
    chatData: [{
        question: {
            type: String,
            required: [true, "Question is required"],
        },
        answer: {
            type: String,
            required: [false, "Statement of question is required"],
        },
        fileUrls: {
            type: String,
            default: '',
        }
    }],
    createdAt: {
        type: Date,
        default: Date.now,
    },
},
    {
        timestamps: true,
    });


const ChatsModel = (mongoose.models.Problem as mongoose.Model<Chats>) || mongoose.model<Chats>("Chat", ChatsSchema);

export default ChatsModel;
```