INOTEBLOG

* **Overview**

INoteBlog is a social media platform that will contain authentication and authorization. Users can send friend requests to each other and follow each other. User can create, edit, and delete their notes and they can write comments on their friends’ notes.

* **Folder Structure**

The project contains two folders frontend and backend to separately manage front end and backend files.

i)**BACKEND**

The backend folder contains

FOLDERS-

Schemas -> contains database models

Middleware -> contains middleware functions

Routes -> contains API path according to role

FILES-

index.js -> the main entry point

db.js -> to connect to the database

package.json -> contains the names of libraries used and their versions

.env -> contains db string and JWT\_SECRET to hide it from users.

backend

|->routes

|->schemas

|->middleware

|->index.js

|->package.json

|->.env

ii)**FRONTEND**

The front-end folder contains

FOLDERS-

Node modules -> contains all the dependencies packages that are used to build and run your react project, the public folder

Src

Components

Context

frontend

|->public

|->src

|->Components

|->Context

|->index.js

|->Node Modules

|->package.json

* **Technologies Used**
* Frontend

React

* Backend

Express with NodeJS

Libraries Used- BcryptJS, JsonWebToken, Express-Validator

* DataBase

MongoDB

* **Database Design**

The database contains two Schemas

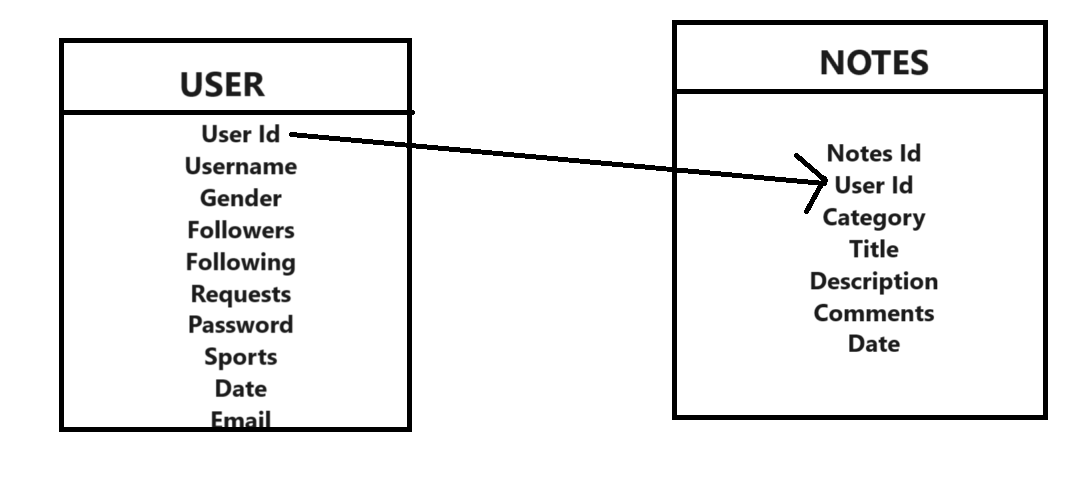
(i)Users

(ii)Notes

They are related to each other using a user ID.

User Schema contains username, gender, favorite sports, followers, following, friend requests

Notes Schema contains title, description, category, comments, and user (to interact with user Schema)



* **USER AUTHENTICATION AND AUTHORIZATION**

i)The project contains user authentication by checking passwords using the bcryptJS library. The API created for this is login, getUser which requires the first password to be verified.

ii)The project contains authorization to show only the notes to which a user is authorized after authentication. It is done using JSON WebToken.

* **API ENDPOINTS**

i)User API

Login User, Get User, Sign Up, Forgot Password, Send Friend Request, Accept Friend Request, Decline Friend Request, Remove Friend

ii)Notes API

Get Notes, Add Notes, Delete Notes, Edit Notes

iii)Friends API

Add Friend, Send Friend Request, Accept Friend Request, Decline Friend ……… Request

iv)Friend Notes API

Get Friend Notes, Add Comment, Delete Comment, Edit Comment

The response of these APIs will be in the form of JSON which will contain two fields first success whether true or false and a message and respective status code.

Example Success Response:

{

success: true,

data: {…..},

message: …..

}

Example Error Response:

{

success: false,

message: error message

}

* **FRONT END**

i)Some of the components used in the front end are Navbar, Side Panel, Home, Add Friend, View Followers, View Following, Friend Requests, My Notes, Friend’s Notes

ii) The router is used to navigate from one component to another.

iii)At every point message is shown in the form of an alert popup whether the API returns success or fails. For this two separate components Success Alert and Warning Alert have been used.

* **STATE MANAGEMENT**

The state is managed in the front end by Using React Hooks like use State, use Effect, Use Context.

* **ERROR HANDLING**

Errors are handled on both the front end and back end. HTTP status codes and descriptive error messages are provided in the API responses to assist in troubleshooting.