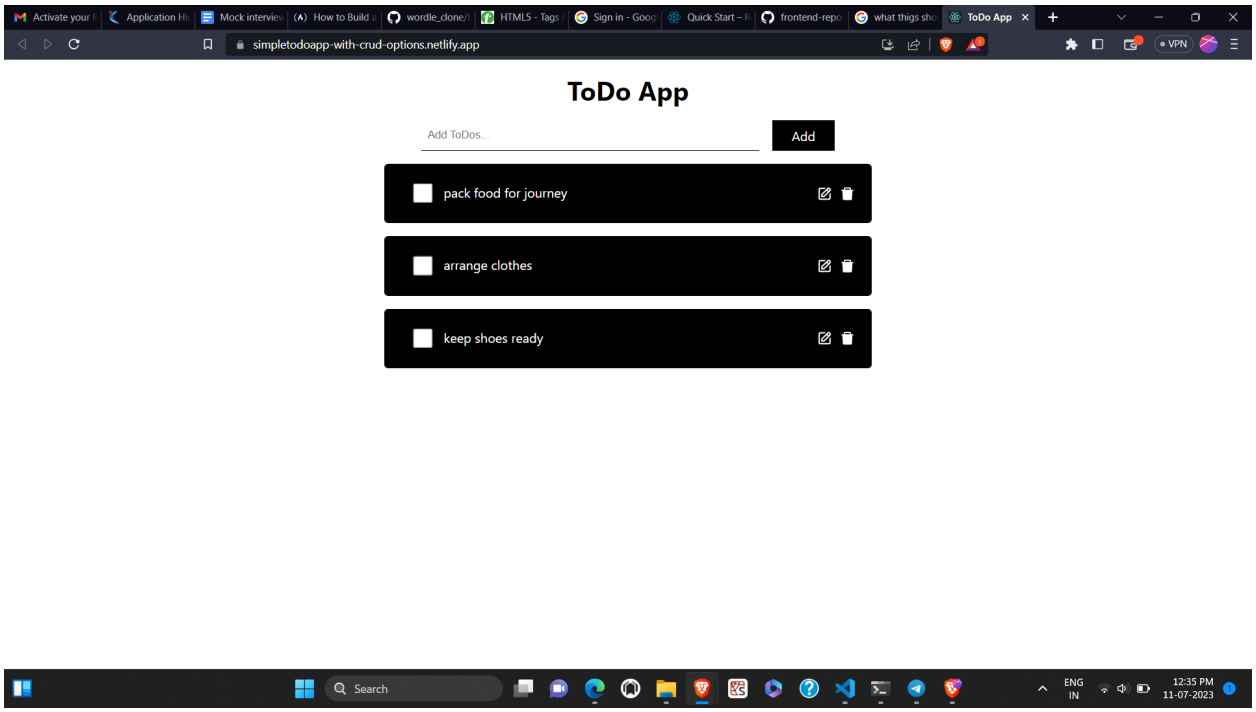
TODO APP DOCUMENTATION

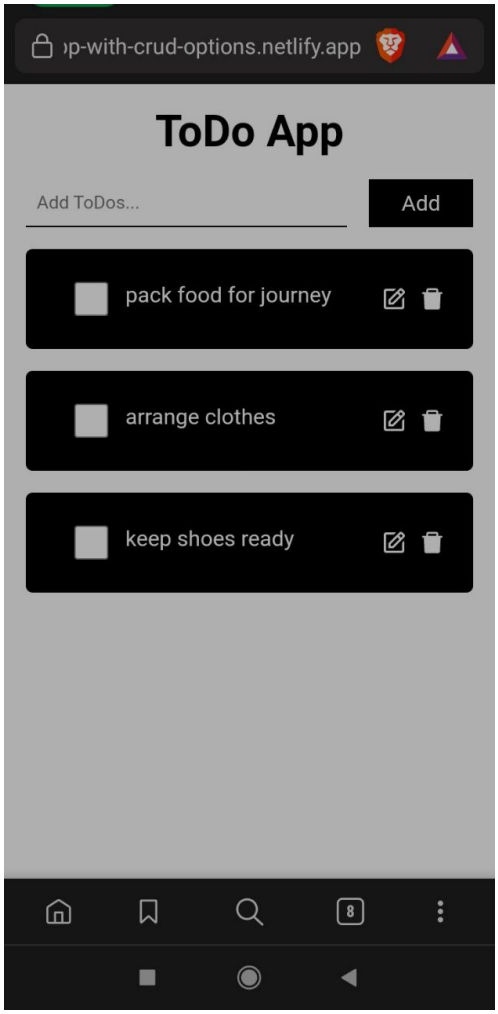
Scope of this app:

The Database is common across all the users.

1) Users will be able to write to-do tasks and can
    a) Add the to-do
    b) Update to-dos that are already existing.
    c) Can mark if its done or not
    d) Can edit the to-do
    e) Can delete to-do task



This is the view of the todo app in a desktop and the view in a mobile is given below.



HOW TO USE??

i)Just type out the to-do task in add todos space. Click on Add button to add the new to-do task.

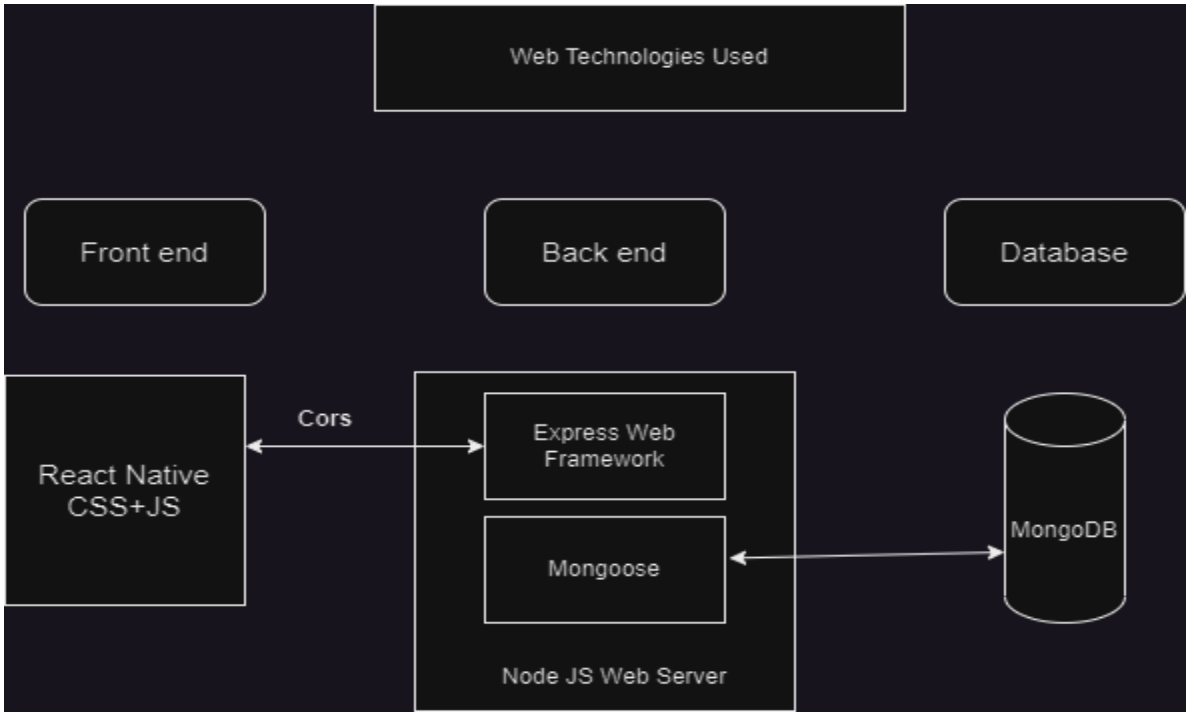
ii)User can click on the edit button to update the task and then click on the Update button which will come in place of add button.

iii)Click the delete button to delete the to-do task.

iv) Click on a checkbox to mark it as done. If user needs to mark it as undone the click again.

Technical Documentation:

This app is built with react native for front-end , nodejs web server with expressJS and mongoose for back-end, mongoDB as database.



Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be accessed from another domain outside the domain from which the first resource was served.

Mongoose is an Object Data Modeling (ODM) library for MongoDB. It defines a strongly-typed-schema, with default values and schema validations which are later mapped to a MongoDB document.

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

All commands used in backend, frontend and database:

1)Backend:

i)npm init – to create a Node. js project. The npm init command will create a package where the project files will be stored. All the modules you download will be stored in the package. Along with creating node modules it creates a file called package.json in which dependencies are listed.

Need to change the scripts field to “start” : “node Server.js” which runs the nodeJS file named server.

ii)**npm i express cors dotenv mongoose** —- this installs these libraries.

Create a .env file where we can define sensitive information which we won't upload to git but set as environment variables while deploying.

iii)**npm start** is to run the backend server which starts executing the server.

Prior to deployment, to view if things are going correctly or not, we can view in http://localhost:8000 where 8000 is a random port number. Our app needs to listen from port 8000 just like HTTP uses 80.

FRONTEND:

i)**npx create-react-app frontend** (where frontend id the folder name) to create react app which includes the package.json which has requirements, and node_modules.

ii)**npm start** – to start the app which we can view in a browser.

This app shows react logo rotating. We can remove these things by deleting the files logo.svg, App.test.js, App.css, and in App.js we need to remove the import statements of these files and remove the HTML section where react symbol and link to learn react.

iii)**npm i react-icons** to use react icons of edit and delete where these are defined in react-icons/bi and /ai respectively which we need to import

iv)**npm i axios** —---- installs axios library where Axios is a popular JavaScript library that allows us to make HTTP requests from our Node.js applications. It simplifies the process of sending HTTP requests and handling responses by providing an intuitive and powerful API.

DATABASE:

Create an account in mongodb.com which provides 512MB of free space. In the databases section -> connect section used to connect to my application where they provide the link to connect to the MongoDB server, need to update the <password> field with the actual password and append the database name that we want to create. This should be kept in a .env file.

Eg:

mongodb+srv://bnsk1998:<password>@cluster0.clunieg.mongodb.net/ToDoApp?retryWrites=true&w=majority

Now the database is ready to take data in.

In the Backend, we used REST API to communicate with the database, specifically get and put functions

Folder structure for the backend:

—----------

Server.js is the main file that has actual code where we define how to connect to the database.
.env file has mongodb url where we define the url with our userid and password which is sensitive(To use this, we installed dotenv library.)
Routes/ToDoRoute.js is where we define the rest API. Here we use expressjs feature Router. The Router object is a feature provided by Express that allows you to create modular and reusable routes for handling HTTP requests. After defining the API, we export it using module.exports = router where the router is an object of Router. In this, the actions of API are not defined here. It is defined in a separate folder.
controllers/ToDocontroller.js is where we define the action of API using "/' for get function. "/save" for saving "/update" for updating "/delete" for deleting. And export all these functions. Every API function definition follows this procedure first action followed by then (response to the user) and catch (for error handling.)
models/ToDoModels.js is where we define the schema of what we want to insert into MongoDB. We use this file name to access and update the field of data in the controller file.

We used **Postman** application to check if we are able to modify, add, delete, and update the database.

Folder structure for front-end:

—----------

public contains the details of the icon, logo, and html code for the app name.
Node_modules contains the required files to run the react app we don't upload this into Git Hub, based on the requirement(from package.json) it installs when we build it in deploying stage.
App.js is the main file that provides the actual look of the app. useState is a hook that gets the state of the database using which we can update the data fields.
Index.css is where we define the styling of all the components of the app.
components/ToDo.js Since the data displayed is dynamic, we do it using JS that we define in this file.
utils/HandleApi.js is the file where we define the actions of buttons that are displayed in front end which communicates to the database based on the REST API functions that we defined in the backend.

Hosting:

Create an account in render.com, to host the code, code must be there in github.

i) click on new -> web services -> give a name to identify the app -> Set Environment to Node -> branch as main -> build command as npm install -> start command as npm start -> go to advanced and set the environment variables defined in .env and create web services. This will take a little time to install and comeup.

Render.com provides a link to access the backend code. Copy this link and update the frontend HandleApi.js where we defined the link to communicate to the database.

Create an account in netlify.com for frontend

Click new -> pick a repo from Git Hub -> set build command as npm run build -> deploy site. Netlify provides you with a link for your cite which you can edit(to not taken web site name) which serves the users.