Documentation of the Project

Internship Task

Project Overview:

The Book Recommendation System is a web application that provides book recommendations based on user input. The system uses a Neo4j knowledge graph database to provide personalized recommendations. The frontend is built with React.js, and the backend is powered by Express.js, which communicates with the Neo4j database.

Frontend Implementation:

The React Frontend was initialized using ‘create-react-app’. I structured the application into components to maintain the modularity and reusability.

Header.js: This displays the title and navigation links.

BookSearch.js: This gives an input field for users to enter search criteria.

BookTable.js: This displays the list of recommended books in a table format.

App.js: The main component that contains ‘Headers’, ‘BookSearch’, and ‘BookTable’.

Backend Implementation:

Setting up Express.js

The backend is developed using Express.js, where

index.js: This is the entry point of the backend application and it sets up the express server and routes.

bookRoutes.js: This handles the API endpoint for fetching book recommendations.

A ‘neo4jDriver.js’ module was created to manage the connection to the Neo4j database.

Data Flow:

The User enters a search query in the frontend and the query is sent to the backend API. Then the backend queries the Neo4j database using the search term. After this, the results are sent back to the frontend and it displays the recommendations in a table format.

Styling:

Styling is done via a CSS file (like ‘app.css’).

API Integration:

Axios is used for making API calls from the frontend to the backend.

Integration of Neo4j with the website:

Before integrating Neo4j with the system, I installed Neo4j and launched neo4j desktop. Then I created a new project and started a new database instance. Moreover, to access the database, I used the Neo4j Browser and logged in using the credentials (‘username’, ‘password’).

After logging in, I created the data model by defining nodes and relationships between them.

Nodes representing entities like ‘Book’, ‘Author’, and ‘Genre’.

Relationships represent connections between these nodes like ‘WRITTEN_BY’, ‘BELONGS_TO’. ‘RECOMMENDED_FOR’.

Connecting Neo4j from Node.js:

For this, I installed the Neo4j JavaScript driver using ‘npm install neo4j-driver’. Then I created a ‘neo4jDriver.js’ file in my backend project to handle the connection.

The backend interacts with Neo4j via API endpoints. The API fetches data from Neo4j and returns it to the frontend. It fetches books from the Database based on the user input and returns recommendations.

For frontend, I made API calls using ‘axios’, or ‘fetch’ to get data from the backend and display it to the user.