Document Service Portal

Content

- **Project Structure**
  - Directory and file organization
  - Critical directories and files
  - Main entry point of the application

- **Key Technologies and Libraries**
  - State management solution and overview
  - Routing mechanism
  - Styling approach
  - APIs and services used

- **Component Structure**
  - Reusable components and their usage

- **Important file and Directories**
  - Key configuration files and their purposes
  - Management and definition of environment variables

- **Deploy the web application**

1. **Project Structure:**
   - **Directory and File organization.**

Frontend (React.js)

“C:\ReactSite\Backend_backup\student-documents-project\src\main\frontend”

Directory Structure:

- **public** (or **build**): This directory contains the production-ready build of React application. It's typically generated by running a command like `npm run build` or `yarn build`. The contents of this directory will be served directly by web server (usually Spring Boot in this case). Common files include:
  - `index.html`: The main HTML file that serves as the entry point for React application.
  - `main.js` or `bundle.js`: The bundled JavaScript code containing React components and logic.
  - `favicon.ico` (optional): The favicon for application.
  - Asset files (optional): Images, fonts, CSS files, etc., that are used by React components. These can be placed within subfolders like `assets` or `images`.
- **src** (or **source**): This directory contains the source code for React application:
  - `App.js`: The main component of React application.
  - Component folders: Folders for organizing React components based on functionality or feature areas (e.g., `components`, `features`).
  - Utility folders (optional): Folders for utility functions, constants, or shared styles (e.g., `utils`, `constants`, `styles`).
  - `index.js`: The entry point for React application, typically rendering the `App` component.
  - Pages and Components folder contains code
- **package.json**: This file contains metadata about project, including dependencies, scripts (like `build` or `start`), and configuration options.

Backend (Spring Boot)

Directory Structure:

- **src/main/java** (or **main/java**): This directory contains the main source code for Spring Boot application:
  - `com.example.student_service_portal_application` (or similar package structure): Java packages containing application classes. This includes:
    - Application class: The main class that bootstraps Spring Boot application.
    - Controller classes: Classes that handle incoming HTTP requests and produce responses (e.g., `MyController.java`).
    - Service classes: Classes that encapsulate business logic and interact with data access layers (e.g., `MyService.java`).
    - Repository classes (optional): Classes that interact with database or data source (e.g., `MyRepository.java`).

- **src/main/resources** (or **resources**): This directory contains various resources used by Spring Boot application:
    - `application.properties`: The main configuration file for application, containing settings for database connections, server ports, logging, etc.
    - Other configuration files (optional): Additional configuration files specific to frameworks or libraries you're using.
    - Data files (optional): JSON, YAML, or other data files used by application.
- **pom.xml**: This file defines the project object model (POM) for Spring Boot application, specifying dependencies and build configuration.

2. **Key Technologies and Libraries:**

(Frontend reactJS)

- **State management solution and overview:** Used React JS state management. (useState)

  `useState` is a powerful built-in React hook for managing state in functional components. It's a good choice for many projects, especially for simpler applications or when you don't need the complexity of a global state management library like Redux.

- **Routing mechanism**: Used “react-router-dom” for routing.

React Router (Dominant Choice): A powerful library for creating dynamic and navigation-friendly React applications. It provides features like component nesting, URL parameters, history management, and programmatic navigation.

- **Styling approach**

CSS (Fundamental Approach): CSS remains the cornerstone of styling for web applications. You can employ methodologies like BEM, OOCSS, or SMACSS for better organization and maintainability.

- **APIs and services used**: Created REST APIs using Spring Boot. For payment gateway integration the payment API and service used.

  HTTP Libraries (Essentials): Axios are popular choices for making HTTP requests to backend APIs, retrieving data, and handling responses in an asynchronous manner.

Backend (Spring Boot):

Key Technologies and Libraries:

- **Spring MVC (RESTful APIs):** These Spring frameworks provide a robust foundation for building RESTful APIs.
- **Spring Data JPA (Database Access):** This library simplifies interacting with relational databases in Spring applications, handling object-relational mapping (ORM) and providing a layer of abstraction from the underlying database details.

- **Spring Security (Authentication and Authorization):** This framework provides comprehensive security features for user authentication, role-based authorization, and CSRF (Cross-Site Request Forgery) protection.

3. **Component Structure:**

   (Frontend reactJS)

   - **Reusable components and their usage:** Navbar Component, that is used in each and every page.

4. **Important Files and Code Sections:**
   - **Key configuration files and their purposes**: Don't have to configure anything, just open terminal and go to frontend folder. Then enter the command "npm install", then the project will automatically be configured by itself.

   Frontend (React.js):

   Key Configuration Files:

   - **package.json:** This file is essential for managing project dependencies, scripts, and configuration settings.
     - **dependencies:** Lists the external libraries and packages your React application relies on (e.g., `react`, `react-dom`, `axios`).
     - **scripts:** Defines commands for tasks like starting the development server (`npm start` or `yarn start`), building the production-ready version (`npm run build` or `yarn build`), and running tests (`npm test` or `yarn test`).
     - **browserslist:** (Optional) Specifies the target browsers your application should support, influencing build optimizations.

   Environment Variables:

   - **.env`(or similar):** This file (usually ignored by version control) stores sensitive information like API keys, database credentials, or feature flags. You can access these variables in your React code using environment variable libraries `like`dotenv`or`react-dotenv`. This keeps sensitive data out of your codebase and allows for easy configuration changes.

   Backend (Spring Boot):

   Key Configuration Files:

   - **application.properties`(or`application.yml`):** This file is the central configuration file for your Spring Boot application. It holds various settings like:

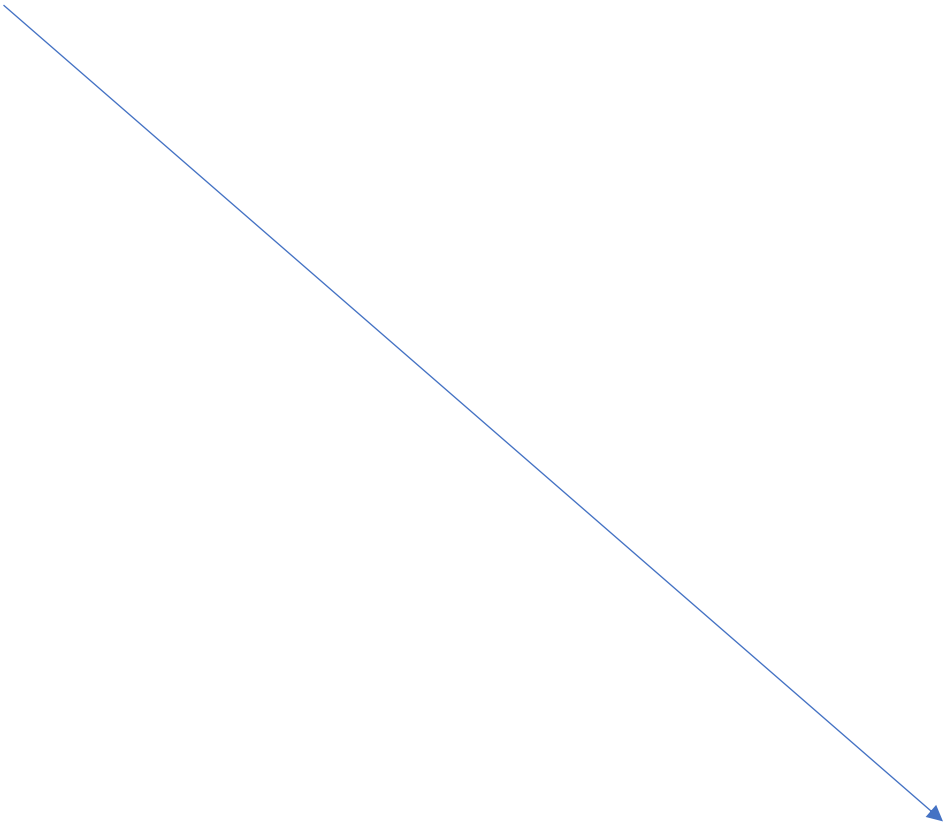
    - Server settings (port, context path)
    - Database connection details (URL, username, password)
    - Spring Security configurations (authentication, authorization)
    - Bean configurations (custom configurations for specific functionalities)

Environment Variables:

- **Similar approach to frontend:** Use a separate file (e.g., `.env`) to store sensitive information. Spring Boot provides mechanisms to access these variables during application startup (e.g., using `@Value` annotation).

Code Sections:

- **React (`App.js`):** This component is the starting point for your application and renders all other components within your hierarchy.
- **Functional Components:** These components are responsible for displaying UI elements and handling user interactions with props.
- **Container Components:** These components connect presentational components to your application's state or data, often fetching data from APIs and managing state using libraries like `useState` or Redux.
- **Spring Boot Controllers:** These classes handle incoming HTTP requests, process logic, and return responses.
- **Spring Boot Services:** These classes encapsulate business logic and interact with data access layers like JPA repositories for database interactions.

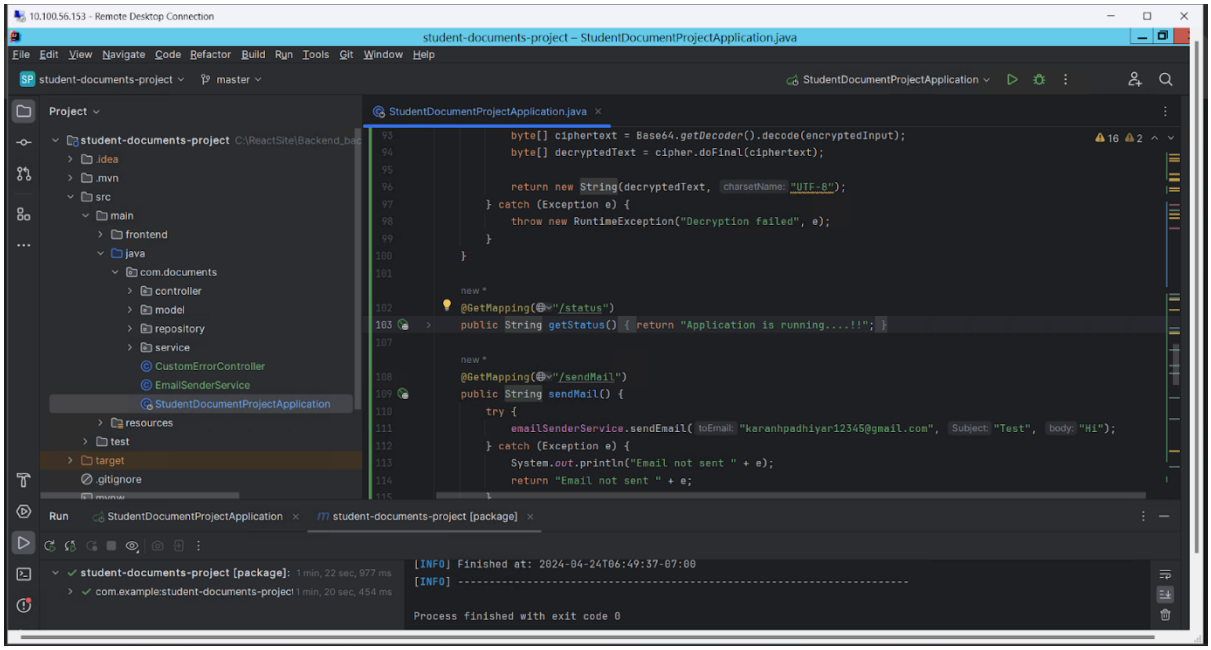


5. Deploy the web application:

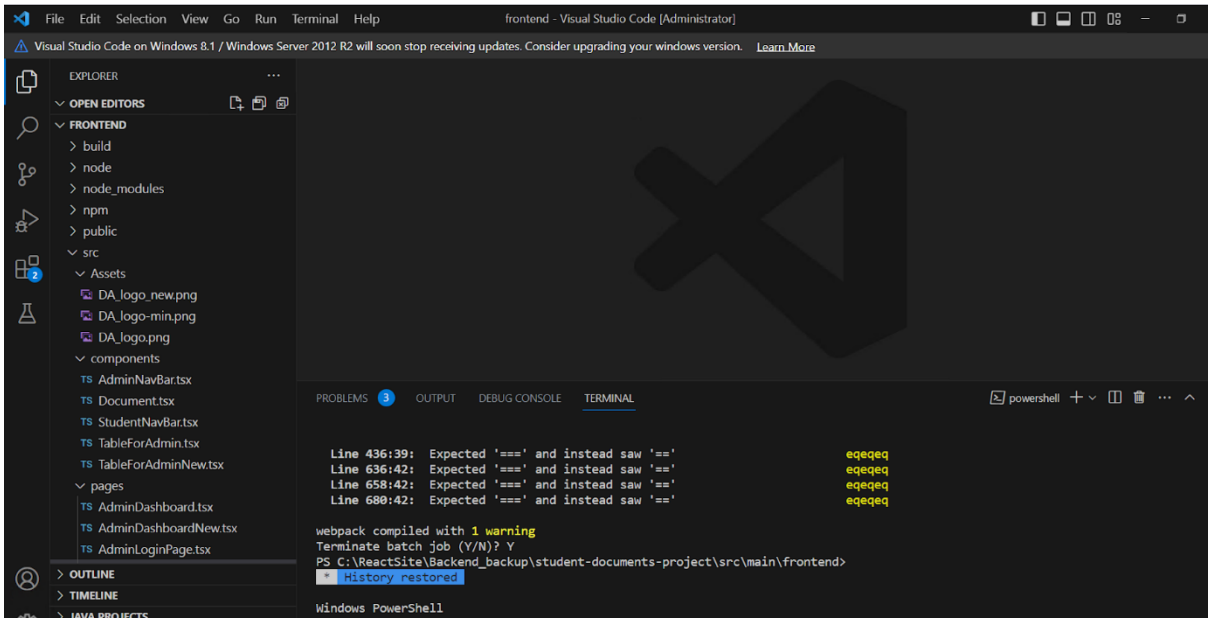
You need VS code (Recommended) for frontend and IntelliJ Idea for backend (Required).

Open **Remote Desktop Connection** and connect to the given **Windows server 2012 R2.**

Open the folder “C:\ReactSite\Backend_backup\student-documents-project” in IntelliJ Idea.



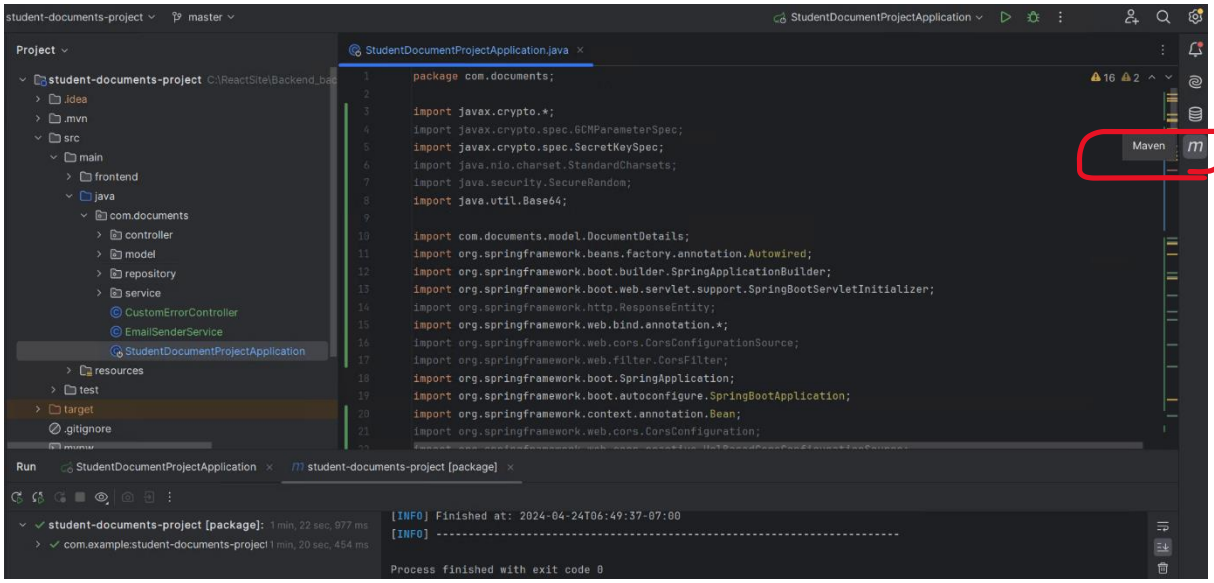
Now the Frontend folder which contains the reactJS code is located at “student-document-project/src/main/frontend” you can open this frontend folder to the individual VS code.



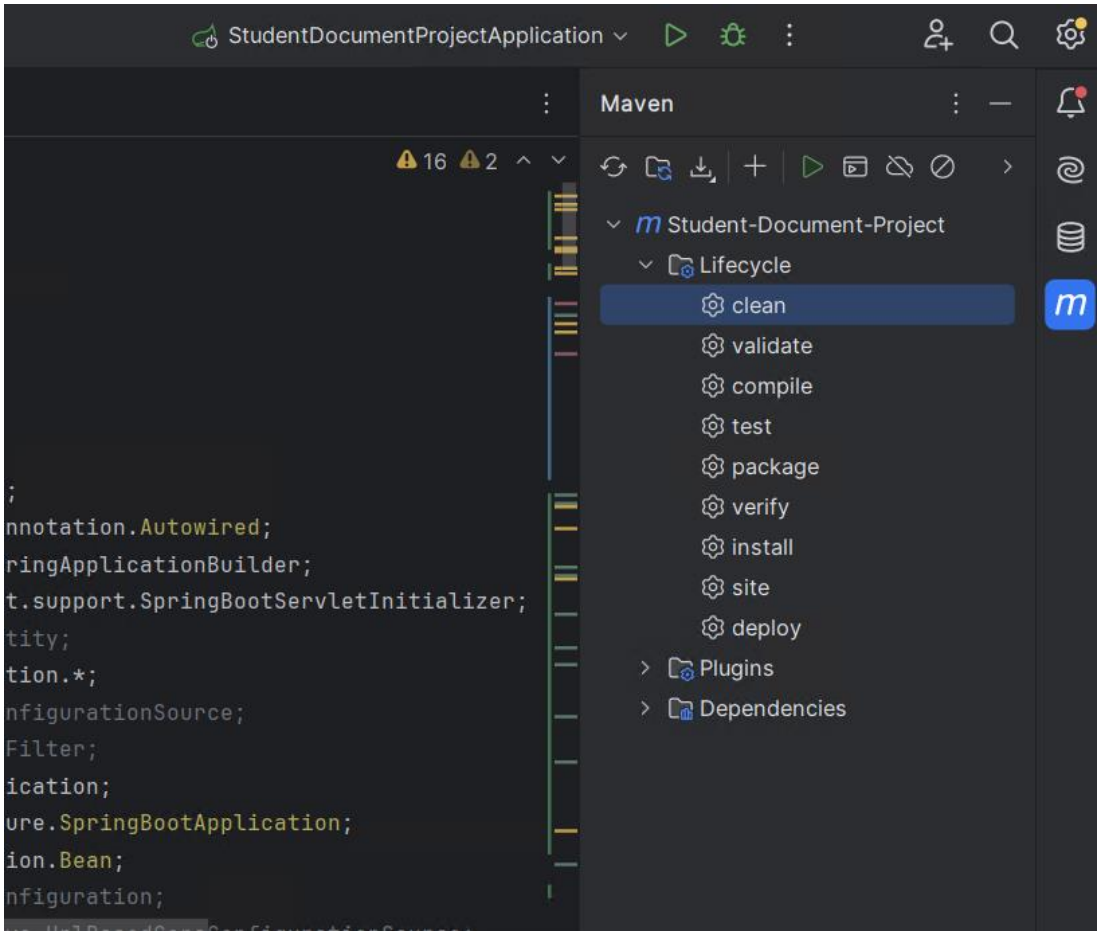
Then you can edit the react code.

Note: Above already written the structure about the react and spring boot.

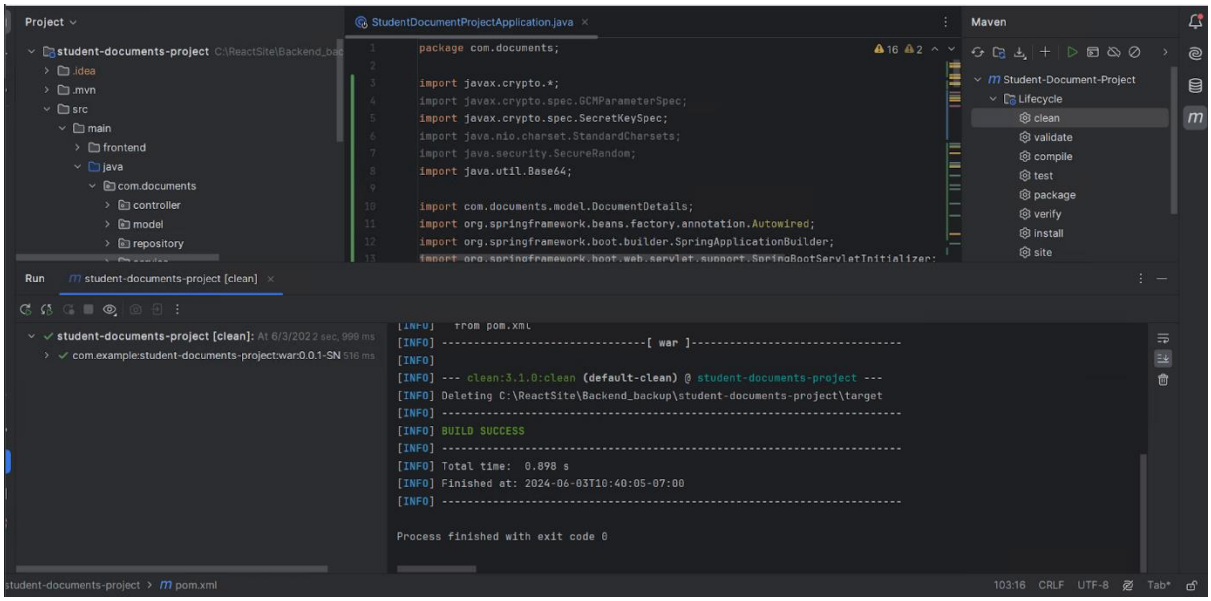
When all the changes made, then open the IntelliJ Idea and click “m” symbol from the right most panel



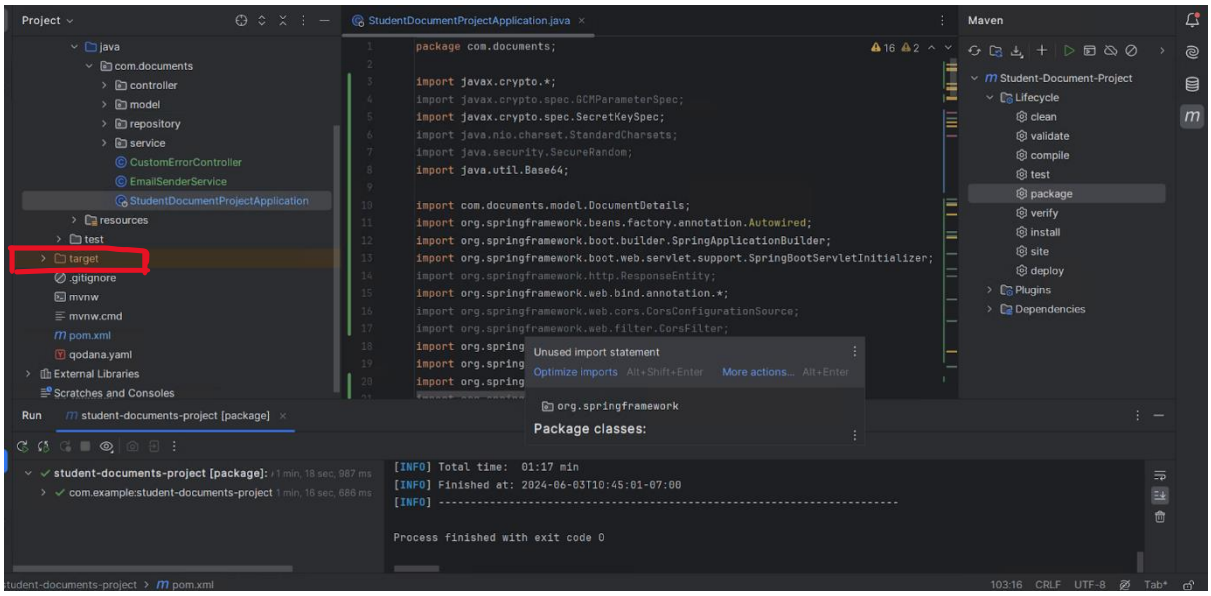
After clicking the m (maven) Icon it shows some options like:



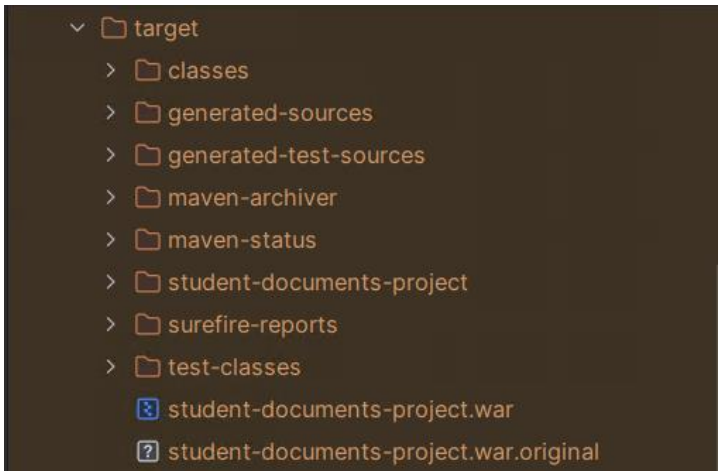
Then, double Click on **clean** option. It will delete the “Target” folder in the current directory.



Then, double click on **packge** option (It will take 2 to 3 minutes to execute the command). It will build the frontend and backend build in one go**.** And the Target folder will appear again.



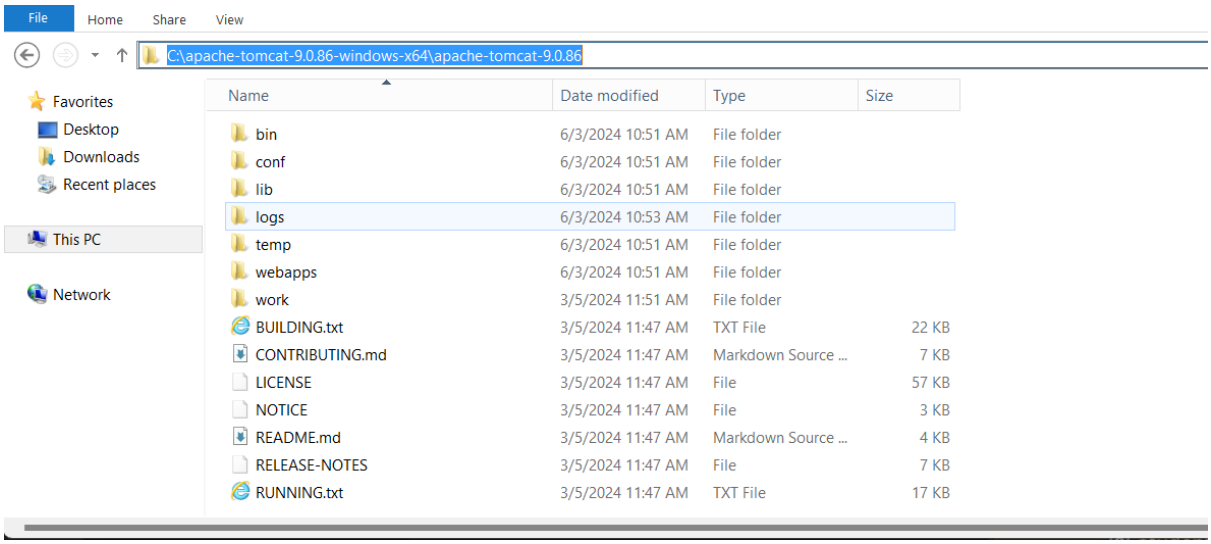
Target folder contains:



Here the **student-documents-project.war** is very important, we will use it to deploy the web application.

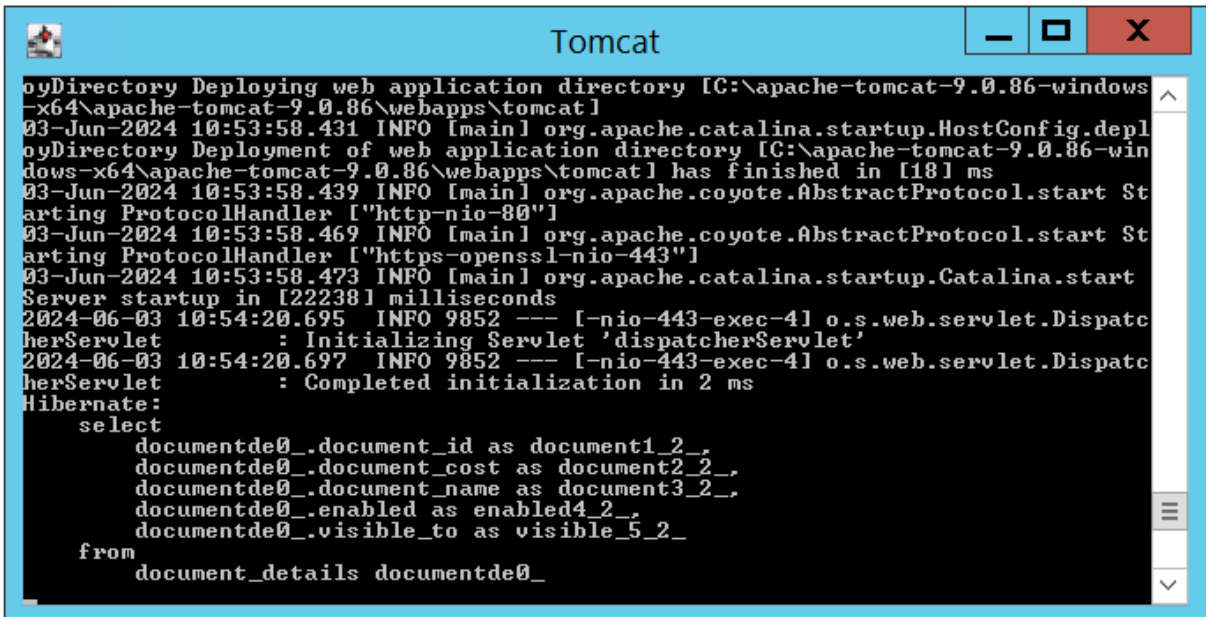
Click on the student-documents-project.war file and copy the file (ctrl + c or right click and copy).

Now go to “C:\apache-tomcat-9.0.86-windows-x64\apache-tomcat-9.0.86”, this is tomcat server configuration folder.



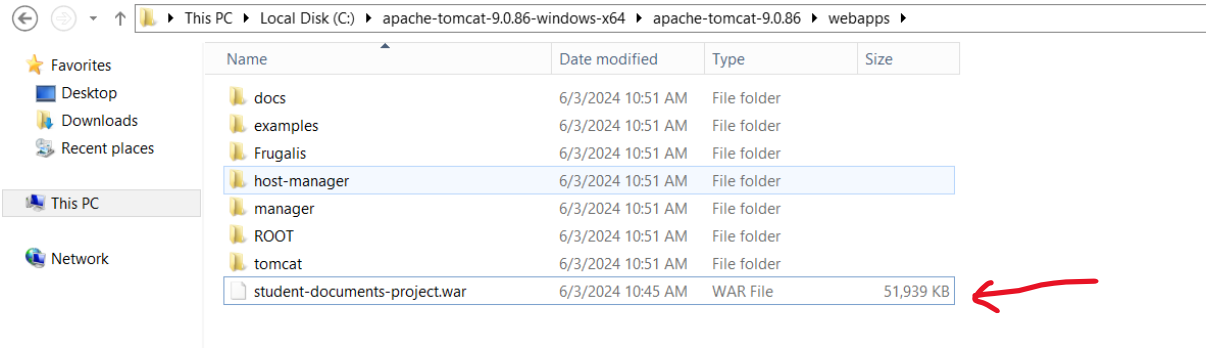
If the tomcat server is already running then you have to stop it. (If the icon is visible in the taskbar)

Then open the tomcat server window, and press (ctrl + c) in the terminal to stop the server.

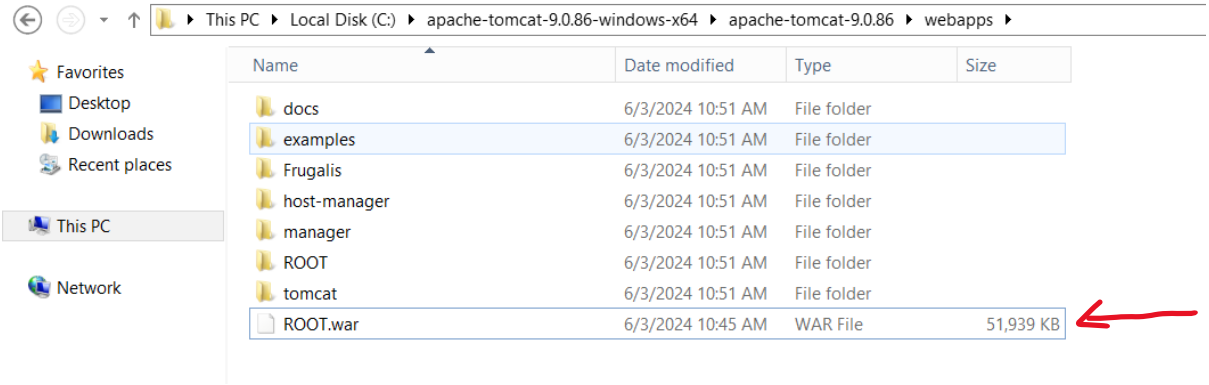


Then, open “C:\apache-tomcat-9.0.86-windows-x64\apache-tomcat-9.0.86\webapps” in the file explorer.

And paste the student-documents-project.war file in this folder.

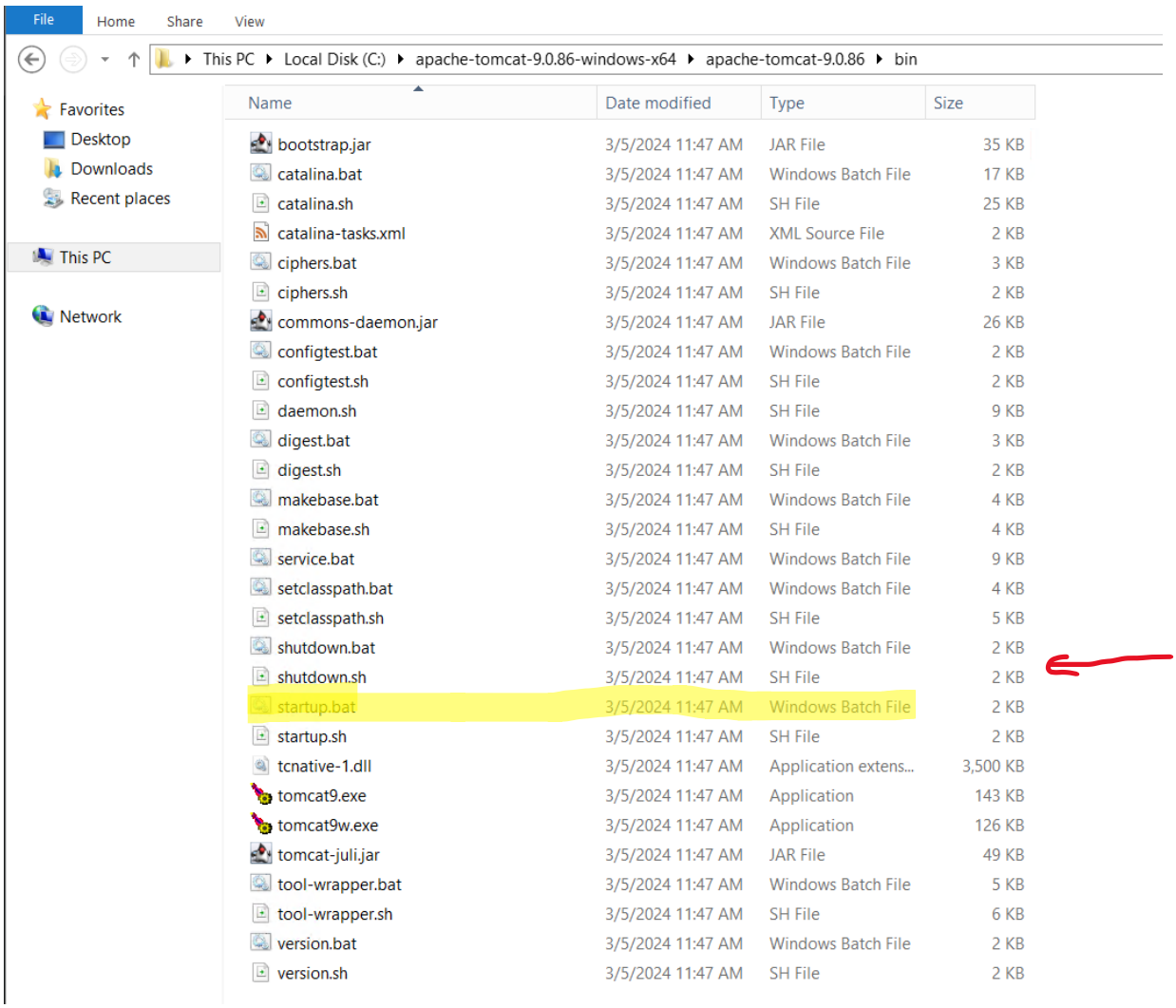


Now, you have to rename the war file to **ROOT.war.**

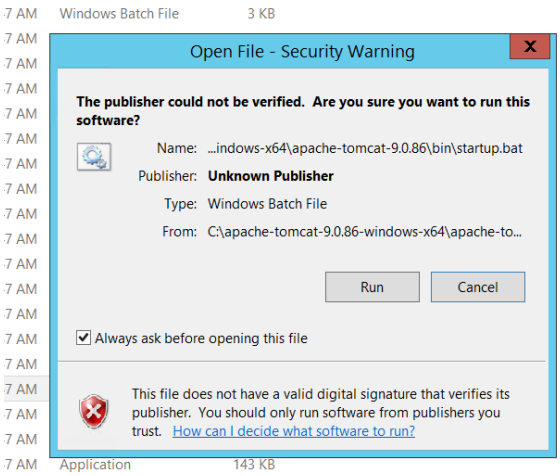


Then go to “C:\apache-tomcat-9.0.86-windows-x64\apache-tomcat-9.0.86\bin” folder to start the tomcat server.

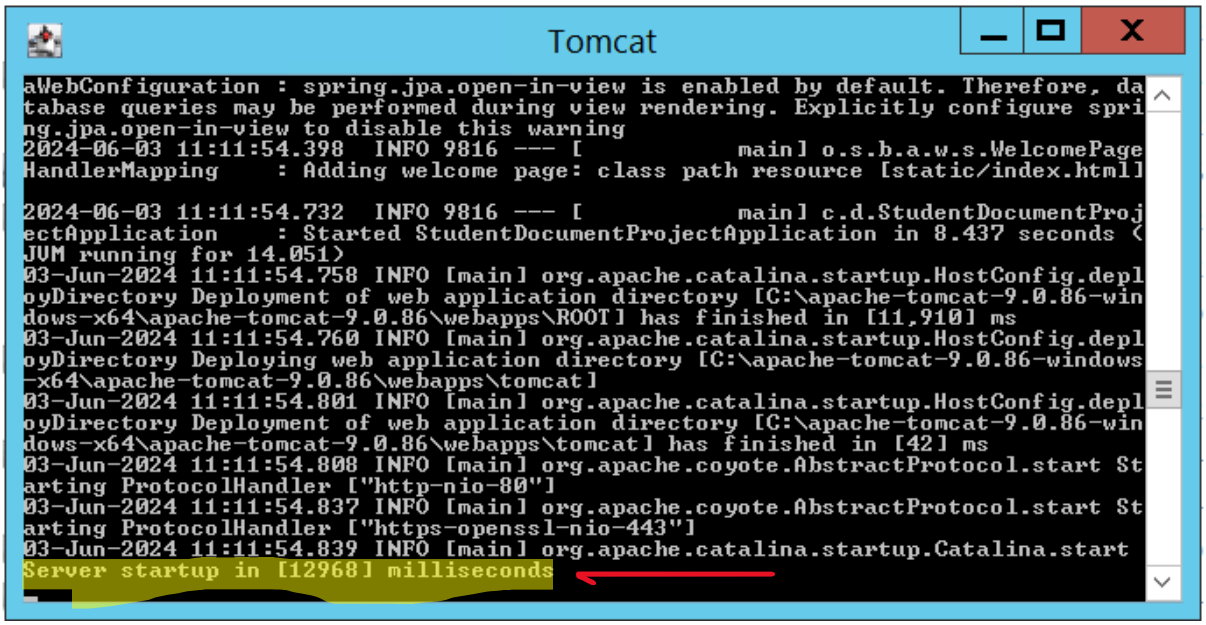
-



Open the **startup.bat** file and it will ask for admin permission to run the application. Then click on Run button.



Then the Tomcat Server terminal will launch and it will take 3 to 5 minutes to start.



When the text highlighted above is appear that means the server is running.
