

**Homework 3: Technology enablers**

**Design document**

**by**: Fatima El kabir

**supervised by**: Dr. Omar Iraqi

**Table of Contents**

1. **Protocol Used**
2. **Consumer side technology**
3. **Server side technology**
4. **API**
5. **Protocol Used**

The communication between the client and server relies on the utilization of the HTTP protocol. This HTTP protocol is characterized by its statelessness, functioning according to a request-response model. In this model, each request is treated as a distinct and independent entity, separate from any other requests. This design ensures that the interactions between the client and server are self-contained, enabling them to operate without retaining information about previous requests, making the system more flexible and scalable.

1. **Consumer side technology:**

**JavaScript** serves as the cornerstone of the development process for the consumer-side application. It's the language of choice for crafting the functionality that end-users interact with. In parallel, the **Fetch** **API** plays a pivotal role by providing the means to initiate **HTTP** requests to a remote server, facilitating a diverse set of actions such as renaming, deleting, downloading, and uploading files. This versatile API acts as the bridge between the client-side application and the server, enabling these essential interactions.

Within the realm of JavaScript, **asynchronous** **programming** takes the forefront. It adopts a promise-based approach, allowing operations to be executed without hindering the overall program's execution. This is particularly advantageous when dealing with tasks that are not directly related to the CPU, ensuring that the application remains responsive and efficient.

These technology enablers, **JavaScript** and the **Fetch API**, synergize cohesively to construct a powerful command-line application. This application is tailored to seamlessly interact with the server, and it excels at performing a lot of file-related operations. From simple tasks like renaming and deleting files to more complex ones like downloading and uploading files, this application streamlines the process and offers a robust solution for server-side interactions.

1. **Server side technology:**

The foundational pillars of this server-side project encompass several key technological components, each playing a pivotal role in its successful operation:

* **Java**: At the core of this project lies Java, the chosen programming language for developing the server-side application. Java's versatility and robustness make it a reliable foundation for building and executing server-side operations.
* **Spring Boot**: an innovative framework built on top of the Spring Framework, is instrumental in simplifying the development and deployment of Java-based web applications. In this specific project, Spring Boot takes center stage as it is harnessed to define the RESTful API for the server-side application. This empowers developers with a streamlined approach to building and maintaining the application, reducing complexity and accelerating development.
* **Jackson**: The Jackson library finds its role as a specialist in handling JSON data within Java applications. Within this project, Jackson is harnessed for the crucial tasks of serializing and deserializing JSON objects in the RESTful API. This ensures efficient data exchange and processing, vital for any modern web-based application.
* **RESTful API**: a set of design conventions for creating networked applications that communicate over the HTTP protocol, forms the backbone of this project. Employing the Spring Boot framework, a RESTful API is meticulously implemented, offering a standardized interface for file management services. These services include file browsing, renaming, deletion, downloading, and uploading.
* **MultipartFile:** Serving as a representation of an uploaded file within a multipart HTTP request, MultipartFile takes on a critical role in handling file uploads to the server in this project. It ensures a seamless and efficient process for users to transmit files to the server.

These technology enablers synergistically collaborate, coming together to give life to a robust server-side application. This application, equipped with a RESTful API, offers a comprehensive suite of file management services. From the client-side perspective, this server-side application acts as a cornerstone for executing various file-related operations. Users can seamlessly interact with the application, taking advantage of its capabilities for tasks such as browsing, renaming, deleting, downloading, and uploading files, thereby enhancing their experience and productivity.

1. **API**

The Directory API provides users with the capability to manage files using a range of operations, including uploading, renaming, downloading, listing, and deleting files. It is built upon the OpenAPI Specification v3.0.1 and operates from a base URL of **"http://localhost:8080".** This API is structured around five distinct endpoints:

* **/upload (POST):** This endpoint facilitates the uploading of a file, with an optional query parameter 'path' to specify the destination
* **/rename (GET):** For renaming a file, users can utilize this endpoint, which requires query parameters 'oldPath' and 'newname'.
* **/download/{filename} (GET):** To download a file, users access this endpoint, providing the required 'filename' path parameter and an optional 'path' query parameter for specifying the location of the file.
* **/browse (GET):** This endpoint enables users to retrieve a list of files located within a specified 'path'.
* **/delete (DELETE):** For file deletion, this endpoint is employed, and users provide the file path within the request body.

In summary, the directory API empowers users to efficiently handle files with a comprehensive set of operations, making file management a seamless and straightforward process. The API adheres to the OpenAPI Specification and operates from the base URL **"http://localhost:8080"** providing users with a reliable and versatile solution for their file-related needs.