SOLUTION APPROACH DOCUMENT

FILE UPLOAD IN ANGULAR 13 AND SPRING BOOT JAVA

Report Submitted

by

LOGESHWARAN K S (VTU15366)

RAHUL KUMAR DAS (VTU15574)

RISHIKA KARUTURI (VTU18063)

VASANTHI DEVI P (VTU18041)

GAURAV VITRAG (VTU17236)

Under the guidance of

Dr. R. ARUNA (Associate Professor)

Ms. M. MEENALAKSHMI (Assistant Professor)

Mr. N. SIVA RAMA LINGHAM (Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN Dr. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

ABSTRACT

Recently the concept of file uploading has become very common in our environment, but it does not have further functions apart from accepting file and storing it in the local drive or the database. This file upload component is a library that makes file uploading much more flexible to all applications with slight code shifts. This component is built for re-useability and generic functionality. The angular framework is used to design the single dynamic page application accepts file which has sizes lesser than the limit and only of format CSV, XLS, XLSM and XLSX. It also gathers the module name from the user where it needs to be stored in a database. The file will transfer to the back-end spring boot service by converting it into a JSON file through the HTTP module using the secure POST method. Spring boot is a Standalone application with an embedded server. It read those file and check for constraints acceptance. After satisfying those constraints then the constraints acknowledgement is sent to the user. Then it processes the file for compatibility and stores the file MongoDB database as a record within the module.

Keywords: Angular, Spring Boot, HTTP module, POST method, MongoDB

LIST OF FIGURES

Fig 2.1	Component Design of File Upload	2
Fig 2.2	Class Diagram of File Upload	3
Fig 2.3	Sequence Diagram of File Upload	4
Fig 2.4	Low Level Diagram of File Upload	6

ABBREVIATIONS

CSV	Comma Separated Value
XLSM	Excel Macro-Enabled Workbook
XML	Extensible Markup Language
JSON	JavaScript Object Notation
SQL	Structured Query Language
NPM	Node Package Manager
HTTP	Hyper Text Transfer Protocol

TABLE OF CONTENT

ABSTRACT **ii**

LIST OF FIGURES **iii**

ABBREVIATIONS **iv**

1 INTRODUCTION

1.1 ABOUT THE DOCUMENT

1.1.1 PURPOSE & SCOPE 1

1.1.2 OVERVIEW 1

2 COMPONENT DESIGN

2.1 COMPONENT DESIGN DIAGRAM

2.1.1 OVERALL WORKFLOW 2

2.1.3 LOW LEVEL DESIGN 6

3 TECHNOLOGY AND FRAMEWORK

3.1 FRONT-END

3.1.1 ANGULAR 7

3.2 BACK-END

3.2.1 SPRING BOOT 8

3.2.2 MONGO DB 9

3.3 TOOLS USED

3.3.1 VISUAL STUDIO CODE 9

4 SOLUTION APPROACH

4.1 APPROACH DESCRIPTION 10

4.2 TEST 11

CHAPTER 1

INTRODUCTION

1.1 ABOUT THIS DOCUMENT

1.1.1 PURPOSE & SCOPE

- The main purpose of this component is to upload files based on the file size and file format like CSV, XLSM, XLSX and XML.
- It also stores those files in the database after removing unwanted columns, duplicate records, and excess columns, and verifying data types based on the config file.
- The scope of building this component is reusability and generic which can be later integrated into any application of the same type or cross-application with minor changes.

1.1.2 OVERVIEW

This file upload component is a generic and reusable library which makes it more flexible to all applications. The angular framework is used to design the single dynamic page application with login credentials used as a front end which serves as a UI component. It also gathers the module name and file. That file will be transferred to the service by converting it into a JSON file through the secure POST method. Sprint Boot is a Standalone application with an embedded server. It read those files and check for constraints. When the constraints are satisfied then the acknowledgement is sent to the user and the file will be stored in the database as a record within the module. This document serves as guide for the developer who looks to use the component without any difficulties.

CHAPTER 2

COMPONENT DESIGN

2.1 COMPONENT DESIGN DIAGRAM

This section describes about the component design and the specific way of designing.

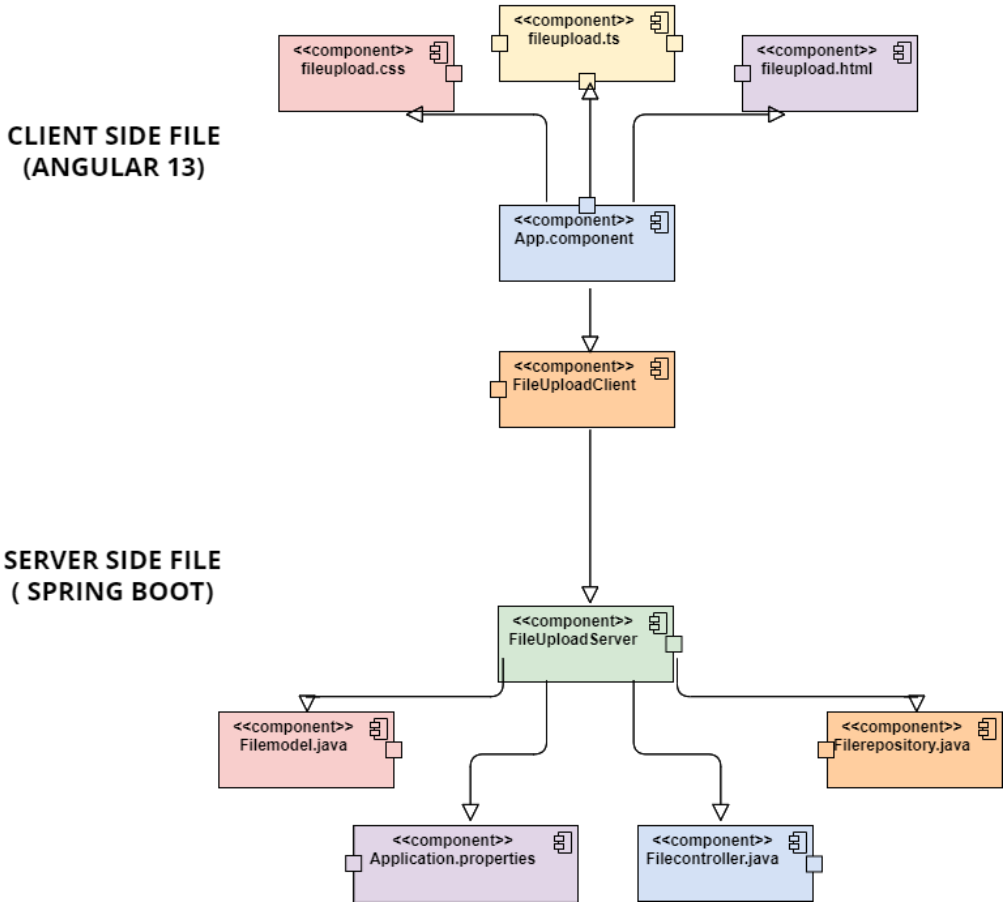

Fig 2.1: *Component Diagram of File Upload*

On client-side, the application uses the combination of angular component with html and CSS as user interface.

App.component : This is the main component from where it starts to implement. It calls the Fileupload component.

FileUpload.html : The html page by which the static page of the UI interface created.

FileUpload.css : The CSS stylesheet helps design the element which are available in the FileUpload component.

FileUpload.ts : The main part the component from where the connection, validation and transferring of file to server is done.

On server-side, the application uses combination of spring boot to extract file and store in database.

Application.properties : This is location where all property of the rest control application is stored and used for needs like running URL, database username and password,

Filecontroller.java :It create the controller which can gather the file from the URL by mapping and return the acknowledgement.

Filemodel.java : Model file contains the data of the specific application, the data is json object.

Filerepository.java : Repository file provide the mechanism of storage, retrieval, search and database operation.

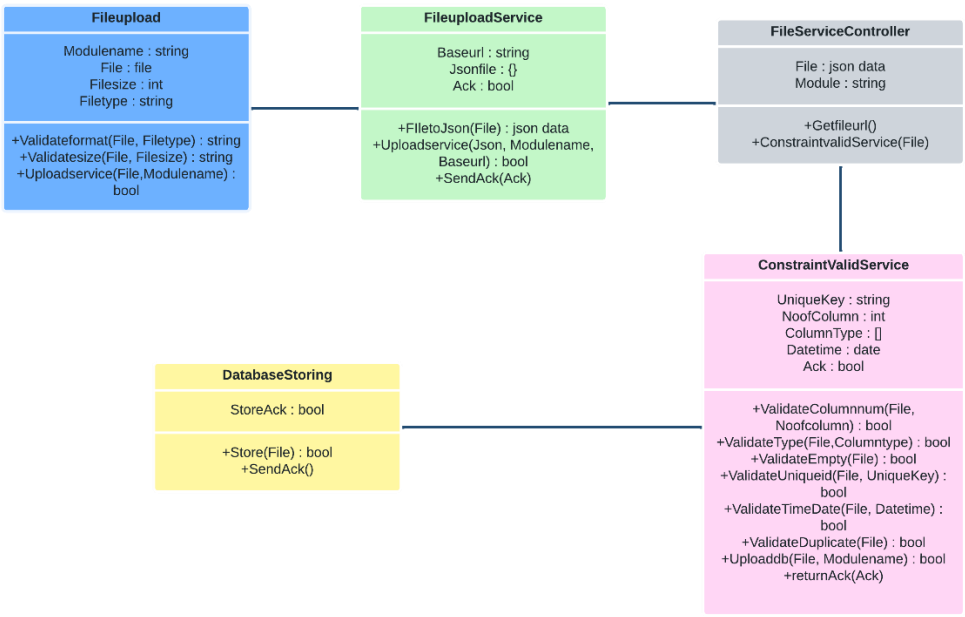

Fig 2.2: *Class Diagram of File Upload*

The front-end processes in this case, which include the file size and file type, are handled by the file upload. This verifies the file size and format. The file is converted to JSON in the file upload service using papa parser, which offers the upload service and gives the user an acknowledgment. Finally, in the Validate and store () function, we are required to use the File Upload Controller object to submit a POST request. This object belongs to the main class library object that has been built, which is File Upload Service. Every class method will be verified in this. Next, we updated the File Upload Controller Class with the success/failure message. Using the same data() function of the db_context class, we store the message if it is successful. the message follows

2.1.1 OVERALL WORKFLOW

This component workflow diagram describes the complete working process of the interdependent component. Initially, the user gets to authenticate using the login credentials. Then user uploads the file which gets validated and converted into a JSON file. It passes to the Spring Boot service with the POST method through the HTTP module. Service will receive the file and check for constraints and store it in the database.

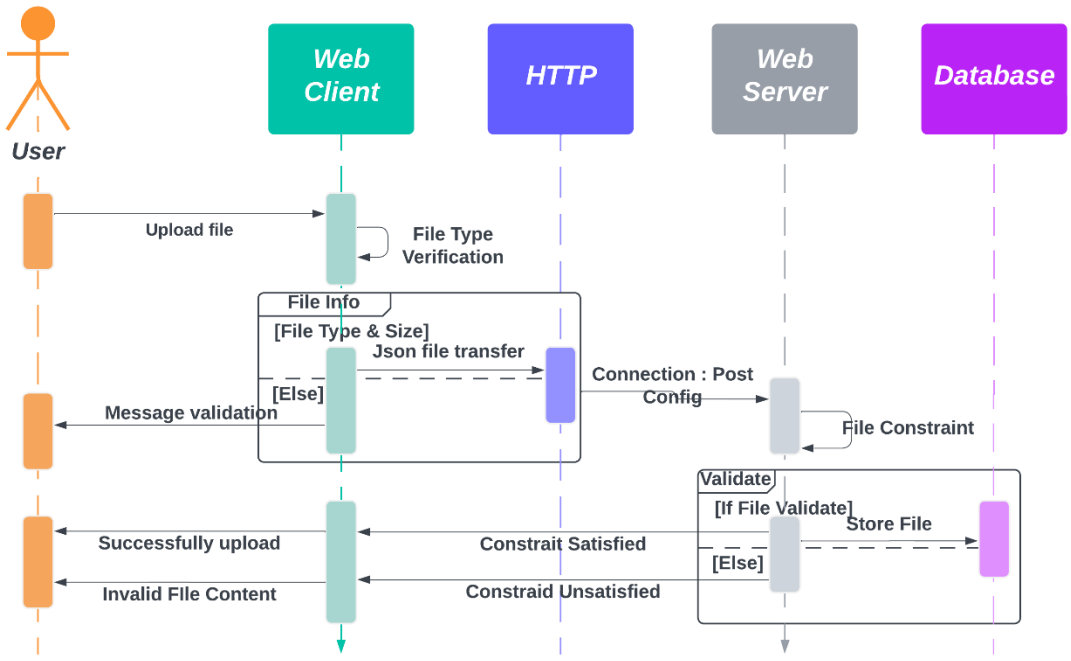

Fig 2.3: *Sequence Diagram of File Upload*

● The sequence diagram shows how to upload a file from a web client to a web server and store the information in a database. The web client, web server, and database are all elements of this process.

● The online client will ask the user to upload a file at the start of the procedure, which can be in Excel, CSV, or XML format. The file is sent from the client to the web server when the user picks it.

● At this stage, a third-party library, such as Papa Parse for CSV files, is used to parse the file and convert it into a JSON object. The database is probably more effective at storing and retrieving data in JSON format, thus this modification is required.

● The web server does validation tests on the file once it has been converted to JSON to make sure the data is correct. For instance, the server could confirm that the file size is within a predetermined range, the file format is accurate, and that the necessary fields are present.

● If the validation checks fail, the server informs the web client that the file upload was unsuccessful by issuing a failure message. In the event that the validation checks are successful, the server stores the JSON data in the database.

● The server then notifies the web client that the file upload was successful by returning a success message. This message may be used to alert the user that the data has been successfully uploaded and saved in the database and to refresh the user interface.

● In summary, the sequence diagram depicts the procedures involved in uploading a file, converting it to JSON, verifying the data, and saving it in a database. This procedure is required for many web applications that need users to submit data, such as social networking platforms, e-commerce websites, and data analytics tools.

2.1.2 LOW LEVEL DESIGN

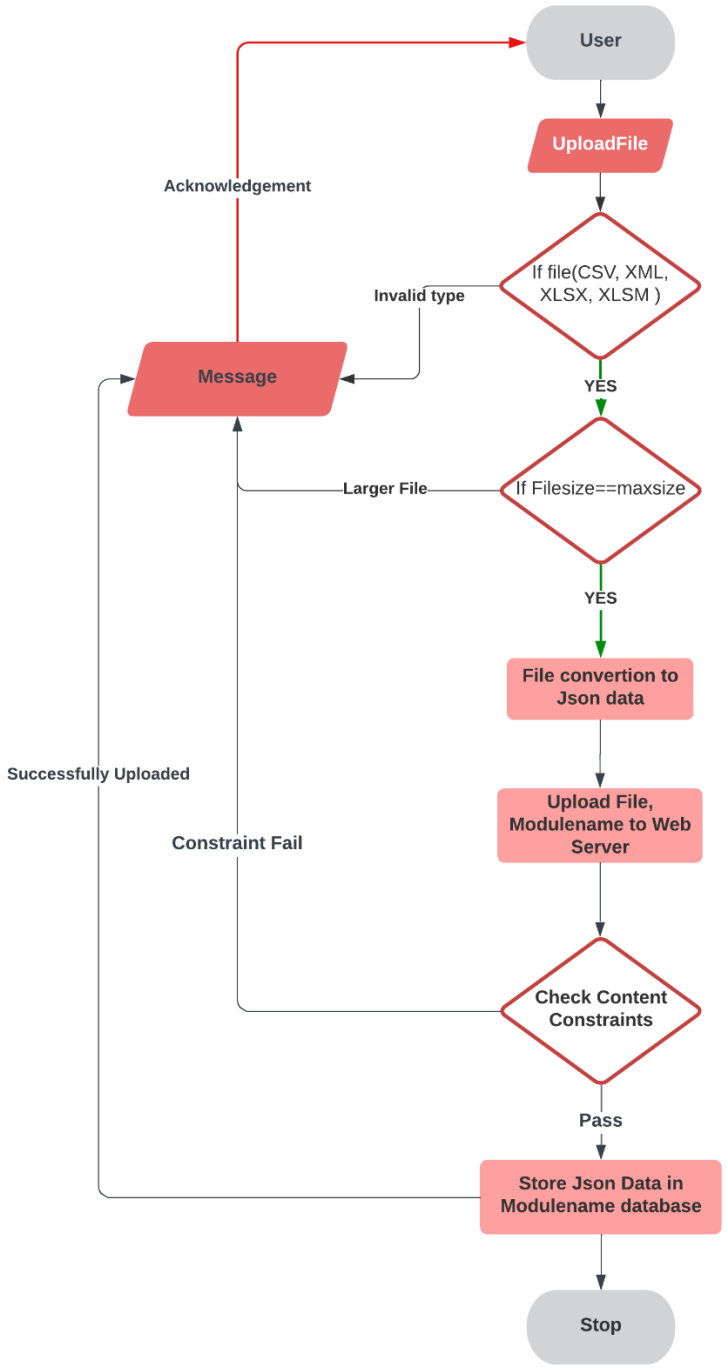

Fig 2.4: *Low Level Diagram of File Upload*

CHAPTER 3

TECHNOLOGY & FRAMEWORK

3.1 FRONT-END

3.1.1 ANGULAR

Version 13

- Angular is an application-design framework and development platform for creating efficient and sophisticated single-page apps.
- It is a component-based framework for building scalable web applications using typescript.
- A suite of developer tools to help you develop, build, test, and update your code.

Purpose

- To create a dynamic page for authentication and file upload. It validates the file format and size and returns the acknowledgement to the user.
- It passes the file to the service by converting it into a JSON file with the module name.

Prerequisite

Node js

Version 19

- Node js is the open-source JavaScript environment used to load all the dependencies for angular applications and it acts as a web server.

Npm

Version 8

- Node Package Manager is installed along with node js. Npm is a default package manager for JavaScript.
- Npm manages the JavaScript runtime.

3.2 BACK-END

3.2.1 SPRING BOOT

Version 3

- Spring Boot helps to create apps that are not platform specific and that can run locally on a device without an internet connection or other installed services to be functional.
- It is a standalone application with an embedded server.

Purpose

- Spring Boot service will gather the JSON file along with the module name from Angular through the POST method.
- It will check for constraints and processes them before storing them in the database.

Prerequisite

Maven

Version 3.9

- The Spring Boot Maven plugin provides support for Spring Boot in Apache Maven.
- It packages all executable jar and war archives and helps to run the Spring Boot application.
- It generates build information and boots the Spring Boot application before running tests.

Tomcat

Version 10

- Tomcat is a web container that manages multiple applications and avoids each separate application setup.
- It makes the application run on the web server.

3.2.2 MONGO DB

Version 6.1

- MongoDB is a source-available cross-platform document-oriented database program.
- Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

Purpose

As it is NoSQL and it can also accept JSON files, which makes us store the file easier in the database.

3.3 TOOLS USED

3.3.1 VISUAL STUDIO CODE

- Visual studio code is a code editor with all language support functionality. It supports development processes like debugging, version control, and testing.
- No separate framework or packages are needed, it can automatically plugin all libraries and frameworks.

CHAPTER 4

SOLUTION APPROACH

4.1 APPROACH DESCRIPTION

This section provides a detailed view of how the components are partitioned and assigned to the functionalities.

1. The user selects a file of types of Excel, CSV, or XML from their local file system and clicks on the upload button. The file content is read and converted to JSON data.. We can convert the uploaded file to JSON format using Papa Parse.
2. After the user clicks the upload button, the JSON data is passed as a parameter to the File Upload Service. To send HTTP requests to the backend, Angular's HTTP Client module can be utilized. The Spring Boot backend receives the JSON data, and the File Upload Service triggers the file upload library/component to validate the received File Upload JSON data.
3. In case all validations pass, the data is uploaded into the database, and a Success message is returned to the front-end. However, if any validation fails, the corresponding validation message is sent back to the front-end. The creation of RESTful services for handling HTTP requests sent from the AngularJS front-end can be achieved using Spring Boot. Service methods can be implemented using Spring libraries, such as Spring Data JPA and Spring MVC, which are responsible for handling file upload and JSON data conversion.
4. Validations - The file upload functionality has the following constraints:
    - Files must not exceed 10 MB.
    - File should support a maximum of 20 columns data.
    - The allowed file format should be XLSX, XLSM, CSV, and XML.
    - The system should ignore blank columns that are between the columns having the data.

- The system should be able to check whether the columns are mandatory or not based on the mandatory configuration done for each column in the config file.
- Unique Key column, which is defined in the config file, should not be duplicated.
- The system should be able to eliminate exact duplicate records before bulk insert or update. It undergoes unit and functional testing. Then detect the issue caused and fixed.
- The system should validate the column length, which is defined in the config file.
- The system should validate the data type of each column, which is defined in the config file.
- The system should be able to validate whether the date/date-time column adheres to a specific date/date-time format (i.e., MM/DD/YYYY or DD/MM/YYYY, etc.), which is configured in the config file.
- Spring libraries such as Spring Data JPA can be used to validate data and handle exceptions such as incorrect format or data type, duplicate records, etc. based on the defined constraints.

5. The application should handle exceptions caused by timeouts or other failures by logging them in the application logs. If the exception is due to incorrect data or format, the application should send back user-friendly messages to the UI. Spring Framework libraries such as Exception and Spring Boot Logging can be used for exception handling, while Jackson can be used for serializing and deserializing JSON data.
6. Security considerations: For security considerations, Spring Security can be used to ensure that only authenticated users are allowed to upload files. Additionally, the uploaded files can be scanned for viruses using an anti-virus software or a cloud-based service.

4.2 TEST CASE SCENARIO

- Test Case 1: Validating file upload for correct file type Input: Upload a file with correct file type (XLSX, XLSM, CSV or XML) Expected Output: Success message with the uploaded file details.

- Test Case 2: Validating file upload for incorrect file type Input: Upload a file with an incorrect file type (e.g. TXT) Expected Output: Error message indicating invalid file type
- Test Case 3: Validating file upload for file size limit Input: Upload a file that exceeds the maximum allowed file size of 10 MB Expected Output: Error message indicating file size limit exceeded.
- Test Case 4: Validating file upload for maximum number of columns Input: Upload a file with more than the maximum allowed number of columns (20) Expected Output: Error message indicating maximum number of columns exceeded.
- Test Case 5: Validating file upload for mandatory columns Input: Upload a file with missing mandatory columns Expected Output: Error message indicating missing mandatory columns.
- Test Case 6: Validating file upload for unique key constraint Input: Upload a file with duplicate records in the unique key column Expected Output: Error message indicating violation of unique key constraint.
- Test Case 7: Validating file upload for column length constraint Input: Upload a file with data that exceeds the maximum allowed length for a column Expected Output: Error message indicating column length exceeded.
- Test Case 8: Validating file upload for data type constraint Input: Upload a file with data that does not match the expected data type for a column Expected Output: Error message indicating invalid data type for column.