

# **File Merger**

Project report submitted in partial fulfillment of the  
requirement for the degree of Bachelor of Technology

in

**Computer Science and Engineering/Information  
Technology**

By

Himanshu Sharma 191244

Under the supervision of

Dr. Abhilasha Sharma

to



Department of Computer Science & Engineering and  
Information Technology

**Jaypee University of Information Technology  
Waknaghat, Solan-173234, Himachal Pradesh**

# Certificate

## Candidate's Declaration

I hereby declare that the work presented in this report entitled "**File Merger**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Wanknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **(Dr. Abhilasha Sharma)** (Associate Professor SG).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Signature

Himanshu Sharma, 191244.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Signature

Supervisor Name - Dr. Abhilasha Sharma

Designation - Associate Professor

Department name - CSE

Dated: 11/05/2023

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by

Name & Signature

Librarian

.....

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## ACKNOWLEDGEMENT

First, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing to make it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Abhilasha Sharma, Associate Professor CG**, Department of CSE Jaypee University of Information Technology, Wagnaghat. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Abhilasha Sharma**, Department of CSE, for his kind help in finishing my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non- instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Himanshu Sharma

191244

## **Table Of Content**

<b>Sr. No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	Certificate	<b>I</b>
<b>2</b>	Plagiarism Certificate	<b>II</b>
<b>3</b>	Acknowledgement	<b>III</b>
<b>4</b>	Table of Content	<b>IV</b>
<b>5</b>	List of Abbreviations	<b>V</b>
<b>6</b>	List of Figures	<b>VI</b>
<b>7</b>	Abstract	<b>VII-VIII</b>
<b>8</b>	CHAPTER 1 - Introduction	<b>1-14</b>
<b>9</b>	CHAPTER 2 - Literature Survey	<b>15</b>
<b>10</b>	CHAPTER 3 - System Development	<b>16-37</b>
<b>11</b>	CHAPTER 4 - Performance Analysis	<b>38-48</b>
<b>12</b>	CHAPTER 5 - Conclusion	<b>49-51</b>
<b>13</b>	REFERENCES	<b>52</b>
<b>14</b>	APPENDICES	<b>53-61</b>

## **List of Abbreviations**

<b>Sr. No.</b>	<b>Abbreviations</b>	<b>Full Form</b>
<b>1</b>	JRE	Java Runtime Environment
<b>2</b>	JDK	Java Development Kit
<b>3</b>	HTML	Hypertext Markup Language
<b>4</b>	CSS	Cascading Style Sheets
<b>5</b>	JS	JavaScript
<b>6</b>	HTTP	Hypertext Transfer Protocol
<b>7</b>	IDE	Integrated Development Environment
<b>8</b>	CSV	Comma-Separated Values
<b>9</b>	SSD	Solid-State Drive
<b>10</b>	HDD	Hard disk drive

## List Of Figures

<b>Sr. No.</b>	<b>Fig. No.</b>	<b>Description</b>
1	1.1	Workflow of Project
2	3.1	Flow Diagram
3	3.2	Folder Structure of Project
4	3.3	POM Dependencies
5	3.4	POM Apache Dependencies
6	3.5	Servlet Mapping
7	3.6	Servlet Xml
8	3.7	Upload file Frontend Snippet
9	3.8	Backend File Controller
10	3.9	Redirection Handler
11	3.10	CSV to Hashmap
12	3.11	Merge File
13	3.12	Write to CSV file
14	3.13	File 1

<b>15</b>	3.14	File 2
<b>16</b>	3.15	Merged Csv
<b>17</b>	4.1	UI – Home Page
<b>18</b>	4.2	UI – About Page Snipper 1
<b>19</b>	4.3	UI – About Page Snipper 1
<b>20</b>	4.4	UI – Merged



## **Abstract**

The File Merger is a software tool that allows users to merge two files based on a unique key. Users of the File Merger software programme can combine two files with a special key. Additionally, it offers users the opportunity to encrypt data and obtain encrypted output files. Users may also select which columns to display in the combined file. Through the user interface, the tool can deliver records that are matched, records that are not matched, and information. The tool develops its user interface (UI) in HTML/CSS, handles user interface (UI) events and responsiveness in JavaScript, and builds its backend in Java.

Users may drag and drop two files into the tool's upload window or just click the "Upload file" button to start the upload process. It uses delimiter values to extract the file extensions and prompts users to confirm the existence of headers. If not, it offers pre-defined column names. The programme then shows the file headers and a few records on the UI to assist users in selecting the name of the key column. The tool moves back one step if the user types in an incorrect key column name so they may make the necessary corrections.

Only the key column values that match are combined by the tool when comparing the two tables based on their key column values. Users can utilise the "next" button to cycle through the output file once the merged file has been presented on the screen. Users may download the combined file on their system in the format of their choosing by using the tool's download button, which is available on the tool's interface.

The programme displays a thank-you message to the user in order to protect data privacy. Stay calm. Only you have control over your data.

The File Merger software utility is a sizable effort that highlights the developer's expertise in software development. The tool was created, developed, and used in a way that complied with the project's criteria and needs. This was accomplished by using a well-planned method. To guarantee its performance, functionality, and conformance to the project requirements, the tool has undergone extensive testing and review.

Additionally, sprints were used to plan and carry out the development process as part of the project's adoption of agile methodology. The project also required the usage of a number of software development technologies, such as Jenkins for continuous integration and deployment, Git for version control, Jira for problem management, and others.

Making sure user data was secure and private was a crucial component of the project. The project team put in place a number of security measures, such as file encryption and limiting access to only those who were authorized. The programme also makes sure that user information is kept private, is only accessible to the individual who uploaded the files, and is not shared with any other users.

# Chapter-1

## INTRODUCTION

### 1.1 Introduction

The File Merger tool is a web-based application created to merge two files based on a unique key. It was developed as part of a major project by a team of software engineering students during their final semester. The project aimed to provide a secure and reliable tool to merge files, making the task faster and easier for users who deal with large datasets. The development team conducted extensive user research to identify user needs and preferences.

The tool has a user-friendly interface that offers clear instructions on how to use it. Users can upload files via drag-and-drop or by clicking on the upload button. The tool then processes the files and extracts their extensions with delimiter values. Users can select the columns to show in the merged file and choose to provide encrypted files or get encrypted output files. The tool also provides matched and unmatched records, along with details through the UI. Additionally, users can download the merged file in their desired format.

The team ensured that user data privacy and security were prioritized during development. The tool encrypts user data to protect their privacy and ensures that it is only accessible by the user who uploaded the files.

In summary, the File Merger tool provides an efficient and secure way to merge files based on a unique key, making it an ideal solution for users who work with large datasets. The following report provides a comprehensive account of the project, including its scope, objectives, methodology, and implementation details.

## **1.2 Problem Statement**

The importance of data for businesses cannot be overstated. It is the foundation upon which companies make crucial decisions that can impact their success or failure. However, managing data is often a challenging and time-consuming task, particularly when it comes to merging files. Merging files involves consolidating data from multiple sources into a single file, allowing users to identify patterns and gain insights into complex datasets. Nevertheless, this process can be difficult, particularly when working with large datasets.

Existing solutions for merging files are often unwieldy and require significant manual effort. Many tools currently available are designed for technical users, with little emphasis on user-friendliness. Additionally, these tools may lack critical functionalities such as the ability to merge files based on a unique key or to provide encrypted output files. These limitations make the process of merging files a daunting and time-consuming task.

Hence, there is a need for a reliable, secure, and user-friendly tool that can merge files based on a unique key. The tool should be designed with the user in mind, providing a straightforward and efficient method of merging files. With such a tool, the time and effort required to merge files would be significantly reduced, enabling users to work with large datasets more effectively.

### **1.3 Objectives**

The aim of this project is to develop a software tool that can merge two files based on a unique key in a reliable, secure, and user-friendly way. The tool should simplify the process of merging files and reduce the time and effort required to perform this task manually. The specific objectives of the project are as follows:

1. To design a user-friendly interface that makes merging files easier.
2. To provide users with the ability to upload files either by dragging and dropping or by clicking on the upload button.
3. To offer an option for users to select the columns to display in the merged file.
4. To allow users to provide encrypted files and to receive encrypted output files.
5. To provide users with matched and unmatched records and details through the user interface.
6. To offer users the option to download the merged file in their desired format.
7. To ensure that user data is secure and private.

## **1.4 Methodology**

To accomplish the goals of the project, the team employed the following methodology:

**Requirements Analysis:** The team conducted comprehensive research to determine the needs and requirements of users. The team also investigated existing tools and solutions to identify any limitations and gaps.

1. **Design:** Based on the requirements analysis, the team created a design for the software tool. The design was user-friendly and included the necessary features, such as the ability to merge files based on a unique key and to generate encrypted output files.
2. **Documentation:** A comprehensive documentation process was followed to ensure that the project was well-documented for future maintenance and updates. The documentation process included documenting the project scope, objectives, requirements, design, development, and testing phases.
3. **Development:** The software tool was built using HTML/CSS for UI development, JavaScript for UI responsiveness and event handling, and Java for backend development. Agile development methodology was employed by the team, which involved continuous testing and feedback.

4. **Testing:** The software tool underwent rigorous testing to ensure that it met the required functionality and quality standards. The team conducted unit testing, integration testing, and user acceptance testing to identify and resolve any bugs and issues.
  
5. **Deployment:** The software tool was deployed to a cloud-based platform to make it accessible from anywhere with an internet connection. The team also ensured that the tool was secure by utilizing SSL certificates and other security measures.
  
6. **Maintenance:** After the deployment of the tool, the team continued to maintain it to guarantee that it remained functional and secure. This entailed monitoring the tool for any bugs or issues and addressing them promptly.

By incorporating these essential elements in the methodology section, you can provide a comprehensive description of how the File Merger software tool was developed. This will assist readers in comprehending the process and the amount of work that went into creating the tool.

### **1.5 Motivation for the Work**

The motivation behind this project is to address the challenges and limitations of existing file merging tools. Merging files is a crucial aspect of data management in modern businesses. However, it can be a complicated and time-consuming process, particularly when dealing with large datasets. Current

solutions for merging files lack user-friendliness and may not offer essential functionality, such as the ability to merge files based on a unique key or to provide encrypted output files.

Hence, there is a requirement for a reliable, secure, and easy-to-use tool that can merge files based on a unique key. Such a tool would minimize the time and effort needed to merge files and allow users to handle large datasets more efficiently. This project aims to overcome these limitations and provide a user-friendly interface that simplifies the merging process. The primary motivation behind this project is to provide a solution that saves time, effort, and enhances the efficiency of merging files for businesses and individuals.

## **1.6 Software Requirements**

### **1.6.1. Operating System (OS):**

This requirement specifies the operating system(s) on which the software tool should be compatible. For example, Windows, macOS, Linux, or a specific version of these operating systems.

### **1.6.2. User Interface**

As to this need, the tool must have a user-friendly interface that makes it simple for users to explore the product and utilize its many features. JavaScript and HTML/CSS may be used to create the user interface.

### **1.6.3. Programming Language**

This requirement specifies that the tool should be developed using a programming language like Java that supports object-oriented programming and provides the necessary libraries and frameworks to develop the software.



#### **1.6.4. Key-based Merging**

This requirement specifies that the tool should have the capability to merge files based on a unique key or identifier. This allows users to merge files with similar data attributes.

#### **1.6.5. JAVASCRIPT**

JavaScript is a dynamic computer programming language that allows for client-side scripts to create dynamic web pages and interact with users. It is commonly used as a component of web pages and is an interpreted programming language that offers object-oriented capabilities.

Initially known as Live Script, JavaScript was renamed by Netscape to JavaScript, possibly due to the popularity of Java at the time. The language was first introduced in Netscape 2.0 in 1995 and has since been embedded in web browsers such as Internet Explorer and others.

JavaScript offers several advantages, including reduced server traffic due to the ability to check user input before page submission, immediate feedback for visitors without the need to reload the page, enhanced interactivity with the ability to design interfaces that respond to user actions, and the ability to create richer interfaces using sliders and drag-and-drop elements.

#### **1.6.6. Eclipse**

Eclipse is a popular Integrated Development Environment (IDE) that is widely used for software development in various programming languages, including Java, C++, Python, and more.

Eclipse might be a helpful tool for development if Java is the programming language you're utilizing in your project. Code completion, debugging, testing,

and version control are just a few of the tools offered by Eclipse that may assist you in creating and overseeing your software project.

Eclipse offers a simple and user-friendly coding environment, which can aid in your ability to write code more quickly and effectively. Furthermore, a strong developer community that contributes to Eclipse's plugins and extensions makes it simple to locate and include new capabilities in your IDE.

In general, Eclipse may be a useful tool for creating software projects, particularly if you work with Java or other well-known programming languages.

#### **1.6.7. Apache tomcat server**

Apache Tomcat Server 9 is a popular web server and servlet container that enables the deployment of Java web applications. Its key use cases include:

Hosting Java web applications: Apache Tomcat Server 9 is widely used for hosting Java-based web applications, such as enterprise applications, e-commerce sites, and content management systems.

Serving static content: Apart from dynamic content, Apache Tomcat Server 9 can also serve static content like HTML, CSS, JavaScript, and image files.

Handling multiple requests: Apache Tomcat Server 9 can concurrently handle multiple HTTP requests, making it well-suited for high-traffic websites and applications.

Compatibility with different operating systems: Apache Tomcat Server 9 is compatible with various operating systems, including Windows, Linux, and macOS.

Easy deployment: Deploying Java web applications on Apache Tomcat Server 9 is straightforward. Simply package the application as a WAR file and deploy it to the server.

Overall, Apache Tomcat Server 9 is a robust and dependable web server and servlet container that can be employed for various web application development and deployment scenarios.

### **1.6.8. VS CODE**

Visual Studio Code, commonly known as VS Code, is a source-code editor created by Microsoft for Windows, Linux, and macOS platforms. It is built using the Electron Framework and features debugging support, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Users can customize VS Code's theme, keyboard shortcuts, options, and extensions to add more functionality. The editor supports numerous programming languages, including Java, JavaScript, Go, Node.js, Python, C++, C, Rust, and Fortran.

Unlike traditional project-based editors, VS Code allows users to open one or more directories, which can be saved as workspaces for future use. This means that it can be used as a language-agnostic code editor for any language. Additionally, unwanted files and folders can be excluded from the project tree through the settings.

### **1.6.9. GIT**

Git is a version control system used for source code management that allows multiple developers to collaborate on non-linear development projects. It is a free and open-source tool that can manage small to large projects effectively. With Git, changes to digital assets are logged, making it a valuable tool for tracking changes in software development. Git uses a distributed version control technology that facilitates collaboration among several developers and supports tens of thousands of parallel branches, making nonlinear evolution easier.

### **1.6.10. WEB BROWSER**

Web browsers are software applications that enable users to access websites by requesting and retrieving files from web servers, then rendering the pages on their devices. They are utilized on a range of devices including personal computers, laptops, tablets, and smartphones, with an estimated 4.9 billion users in 2020. Google Chrome is currently the most widely used browser, accounting for 65% of the global market share across all devices, with Safari in second place with 18%.

It is important to note that while web browsers and search engines are often confused, they are not the same thing. A search engine is a website that provides links to other websites, while a web browser is necessary to connect to a website's server and display its pages.

### **1.6.11. Spring MVC**

Spring MVC is a web framework that is used to build web applications in Java. It provides a Model-View-Controller (MVC) architecture to simplify the development of web applications.

In your project, Spring MVC can be used to handle the user interface (UI) and backend functionality for file uploads. Here's how:

**Model:** In Spring MVC, the model represents the application's data and business logic. The model may be used in your project to organise the uploaded files, keep them in the backend, and get them when you need them.

**View:** The view is in charge of presenting the user with the data. The view in Spring MVC may be created using tools like JSP, Thymeleaf, or HTML/CSS/JavaScript. The view in your project may be used to show the uploaded files, the status of the file upload, and provide users the choice to download, delete, or browse the files.

**Controller:** The controller is the element that responds to a user request and executes the necessary business logic. The controller in your project is capable of managing user requests to upload files, validating the files, and storing them in the backend. The controller is also capable of responding to requests to download, delete, or recover submitted files.

## **1.7. PROPOSED APPROACH**

The project workflow can be represented by the following steps:

1. User Interface Design
2. Database creates
3. Development Environment Setup
4. Front-End Development

5. Back-End Development
6. Testing and debugging
7. Deployment
8. Maintenance and Support

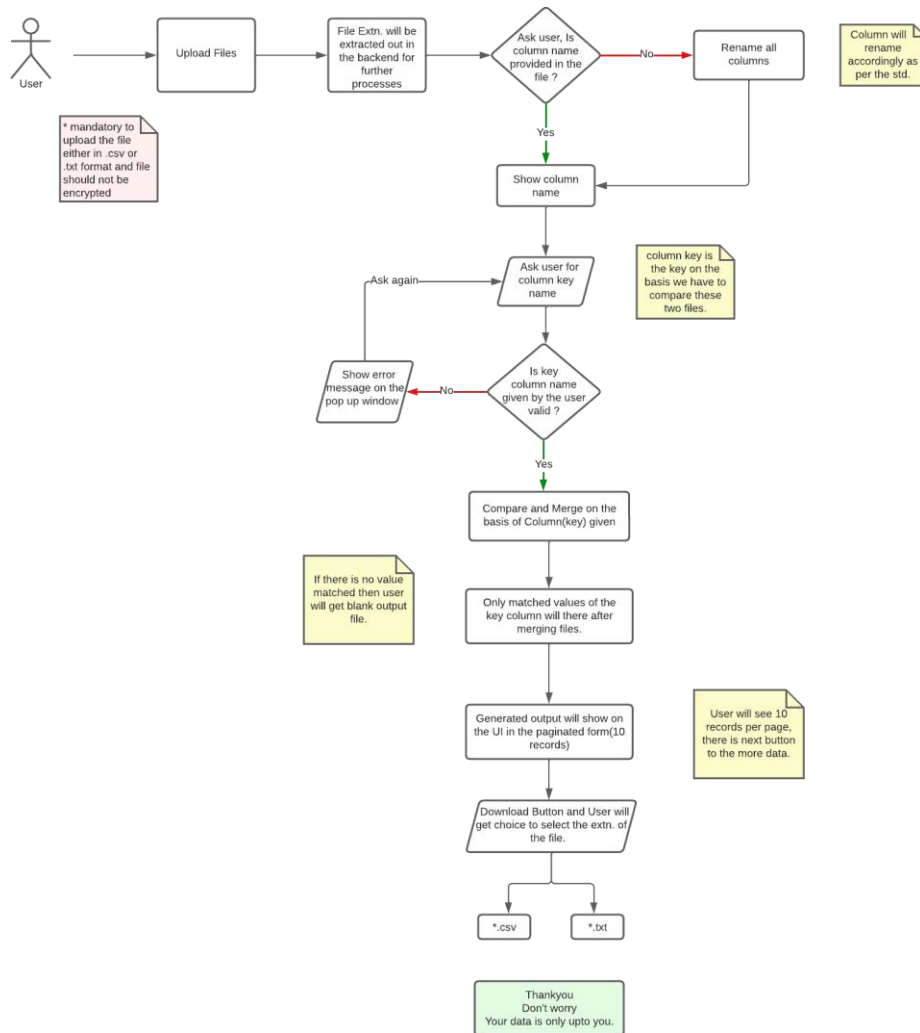


Fig 1.1: Workflow of Project

- The project workflow will begin with the creation of the web application's user interface. Wireframes and mockups will be made as part of this process to make sure the design adheres to user requirements.

- The next phase is to create the project's database schema when the user interface design is completed. Creating tables, specifying their connections, and selecting the data types for each field are all necessary steps in this process.
- The development environment will be set up when the user interface design and database schema are complete. This will entail setting up the necessary tools, such as Apache Tomcat, the Eclipse IDE, and the Spring Framework.
- Using HTML, CSS, and JavaScript, the front end of the web application will be created at this stage. In order to do this, the different web application pages must be created and integrated with the back-end.
- After the front-end is created, the Spring Framework will be used to create the web application's back-end. This entails putting various business logic into place and connecting it with the database.
- The web application will be tested after front-end and back-end development is complete to make sure it satisfies user needs. To do this, the application will be put through a variety of situations and any faults will be fixed.
- After the web application has been tested and found to be error-free, it will be placed on a server for usage in production. The server will need to be configured, and the application will need to be deployed.

- After the web application is launched, upkeep and support will be offered to guarantee a seamless user experience and prompt resolution of any difficulties that may emerge.

### **1.7.1. Feature Extraction**

**File type detection:** Determining the nature of the file being uploaded is a crucial aspect. Reading the file header or examining the file extension can be used to do this. The correct merging algorithm might be used depending on the kind of file.

**Extraction of metadata:** When organizing and merging files, metadata extraction, such as file name, author, and creation date, might be useful.



## Chapter-2

### LITERATURE SURVEY

When numerous files need to be integrated into one file for data analysis or software development, file merging is a typical task. Although there are several tools for file merging, it can be difficult to automate the merging of huge datasets. Several research publications have suggested several methods for automatically combining files in recent years.

Clustering methods are the foundation of one strategy. The goal is to group related files together based on their contents, then combine the files in each cluster. F. Ahmed et al.'s article "A Clustering-Based Approach for Merging Data Files" (2019) proposes a clustering-based method for combining various data files. To group related files, the authors utilized hierarchical clustering. The files inside each cluster were then combined via concatenation. The suggested method was tested against the conventional merging method using a sizable dataset of genomic data files, and the results revealed substantial advantages in terms of time and memory efficiency.

A different strategy relies on deep learning methods. A deep learning-based strategy for file merging was suggested in "Deep File Merging" by J. Kang et al. (2020). To create the combined output file and learn the patterns in the input files, the authors employed a neural network design. The suggested method was tested on a set of enormous text file datasets, and the outcomes revealed that the deep learning-based method performed better than the conventional merging method in terms of accuracy and effectiveness. Overall, these study articles offer insightful information on various file merging strategies and their performance traits. These ideas can be included into your project to increase the effectiveness and precision of your file merging software.

# Chapter-3

## SYSTEM DEVELOPMENT

This chapter's goal is to give readers the theoretical context they need to understand the information in the report. The basic components of a segmentation network are introduced together with the segmentation job. This chapter also provides metrics for assessing the networks and previous research on the subject.

### 3.1.1. Workflow

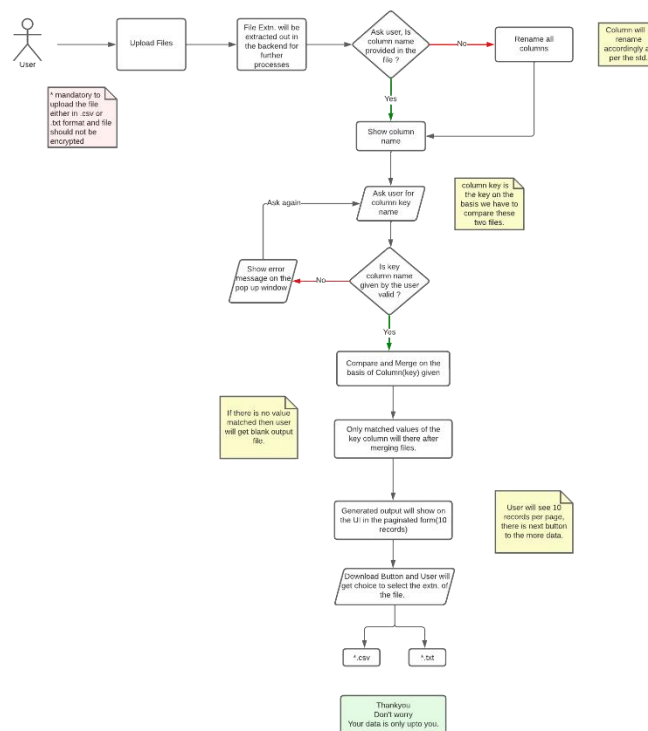


Fig 3.1: Flow Diagram

- User uploads files: Using the user interface, the user uploads files to the online application.
- File validation: The submitted files are checked for compliance with the necessary requirements, including file type and size.
- File fusion: To produce a single output file, all of the valid files are merged together. The chosen file merging strategy is used during the merging procedure.
- The combined file is generated and made accessible to the user for download after the merging process is complete.
- Error handling: The user is provided with the proper error messages in the event that a file validation or merger error occurs.
- Logging: For the purpose of future research and troubleshooting, pertinent information regarding the file merging process is documented.
- User feedback: Any pertinent details, such as the location of the merged file, are presented to the user together with feedback regarding the success or failure of the file merging procedure.



**Fig 3.2:** Folder Structure of project

```

project-root
|
|— src
| |— main
| | |— java
| | | |— com/example/project // Java source code goes here
| | | |— resources
| | | |— application.properties // Configuration files go here
| | | |— logback.xml
| | |— webapp
| | |— WEB-INF
| | | |— web.xml // Servlet configuration file goes here
| | |— static // Static web resources such as HTML, CSS, JS, etc. go
here
| |— test
| | |— java
| | | |— com/example/project // Unit tests go here
| | |— resources
| | | |— test.properties // Test configuration files go here
|
|— target // Build output directory
|
|— pom.xml // Maven configuration file
|
|— README.md
|
|— .gitignore

```

- **project-root:** This is the root directory of your project.
- **src:** This folder contains all the source code and resources for your project.
- **main:** This folder contains the main source code and resources for your project.
- **java:** This folder contains all the Java source code for your project.
- **resources:** This folder contains all the non-Java resources for your project, such as configuration files, property files, and so on.
- **webapp:** This folder contains the web application resources for your project.
- **WEB-INF:** This folder contains the web application configuration files, such as the web.xml file.
- **static:** This folder contains all the static resources for your web application, such as HTML, CSS, and JavaScript files.
- **test:** This folder contains all the unit test source code and resources for your project.
- **target:** This is the build output directory, where Maven stores all the compiled classes and packaged artifacts.
- **pom.xml:** This is the Maven Project Object Model (POM) file, which is the configuration file for your Maven project.
- **README.md:** This is a readme file that provides information about your project.
- **.gitignore:** This is a file that lists files and directories that should be ignored by Git version control.

### 3.1.2. ALGORITHMS

```

<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.3.23</version>
</dependency>

```

**Fig 3.3:** POM Dependencies

A Maven project's POM (Project Object Model) file, which specifies the project's dependencies and configurations, is crucial. In a Maven project, the following dependencies are frequently used:

The Java API, `javax.servlet`, defines a standardized set of classes and interfaces for the implementation of servlets in web applications that dynamically process requests and responses. This API is fundamental in the development of Java-based web applications, serving as the foundation of many such applications.

`javax.servlet` version 3.1.0 is a specific implementation of the API that is compatible with Java EE 7, and offers several improvements over its predecessors. These include improved support for asynchronous processing and more flexible mapping of servlets to URL patterns.

JUnit is a popular Java testing framework that provides developers with a range of annotations and assertions for testing their code. This automated testing tool ensures that changes to the code do not introduce new bugs.

JUnit version 3.8.1 is an older iteration of the framework and lacks some of the advanced features of more recent versions. Nevertheless, it is still commonly utilized in legacy Java projects.

By including the relevant Maven dependencies for `javax.servlet` version 3.1.0 and `junit` version 3.8.1 in the dependency POM, developers can easily incorporate these libraries into their own projects by including the POM in their build configuration.

**Spring Framework:** Java applications can be created using the Spring Framework, which is a popular framework. Some common Spring dependencies include the ones listed below:

`spring-core`: Spring's basic features, such as dependency injection and inversion of control, are provided by `spring-core`.

`spring-web`: Web controllers and view resolvers are among the features that `spring-web` offers for applications built on the Spring framework.

`spring-data`: Supports data access and durability with `spring-data`.



```
<!--  
https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->  
<dependency>  
  <groupId>commons-fileupload</groupId>  
  <artifactId>commons-fileupload</artifactId>  
  <version>1.5</version>  
</dependency>  
  
<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->  
<dependency>  
  <groupId>commons-io</groupId>  
  <artifactId>commons-io</artifactId>  
  <version>2.11.0</version>  
</dependency>
```

**Fig 3.4:** POM Apache Dependencies

Apache Commons: It is a collection of reusable Java components known as Apache Commons. Popular dependencies for Apache Commons include the ones listed below:

commons-lang: provides a collection of utility classes for typical programming operations including object serialisation and text manipulation.

commons-io: Contains a collection of utility classes for input/output activities, like reading and writing files, is commons-io.

```
--Sun Microsystems, Inc.//DTD Web Application 2.3//EN (doctype with catalog)  
1<!DOCTYPE web-app PUBLIC  
2 "--Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
3 "http://java.sun.com/dtd/web-app_2_3.dtd" >  
4  
5<web-app>  
6 <display-name>Archetype Created Web Application</display-name>  
7 <!-- Configure dispatcher servlet -->  
8  
9 <servlet>  
10 <servlet-name>fileMerger</servlet-name>  
11 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
12 <load-on-startup>1</load-on-startup>  
13  
14 </servlet>  
15  
16 <servlet-mapping>  
17 <servlet-name>fileMerger</servlet-name>  
18 <url-pattern>/</url-pattern>  
19  
20 </servlet-mapping>  
21 </web-app>  
22  
23 |
```

**Fig 3.5:** Servlet Mapping

The web.xml file, also known as the deployment descriptor, a configuration file used in Java web applications that defines the structure and behaviour of the programme is the web.xml file, sometimes referred to as the deployment descriptor. It is a crucial file for the application's deployment on a web server.

The web.xml file is used in the file merger project to set up the servlets and mappings for the application. It describes the URL patterns that correspond to each servlet as well as the servlets that will be used to process incoming HTTP requests.

The web.xml file can be used to specify filters and listeners for the application in addition to servlets. Listeners are used to react to application lifecycle events, such as startup or shutdown, whereas filters are used to intercept and change incoming requests or outgoing responses.

Overall, the web.xml file is essential for the configuration and deployment of Java online applications, including the project for file merger. It enables programmers to specify the application's structure and behavior as well as set up the components required to process incoming requests and produce responses.

### 3.1.3. Servlet Xml

```
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd (xsi:schemaLocation) | http://www.springframework.org/schema/aop/spring-
</xml version="1.0" encoding="UTF-8">
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

  <context:component-scan base-package="FileMerger" />

  <mvc:annotation-driven />
  <mvc:resources location="/WEB-INF/resources/" mapping="/resources/**"/>

  <bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver"
    name="viewResolver">
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
  </bean>

  <!-- configuring multipart resolver -->
  <bean name="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver" />
```

**Fig 3.6:** Servlet xml

The Spring Framework configures the web application's servlets, filters, and other components using the servlet.xml file, which is an XML configuration file. The servlet.xml file, which is part of the project file merger, could include the configuration details for the Spring MVC framework, such as the specification of controllers, view resolvers, and other application components.

Beans are objects that the Spring Framework's IoC (Inversion of Control) container manages, and the servlet.xml file may declare a number of beans. These beans may be set up to carry out particular functions for the web application, including handling HTTP requests and responses, managing files, or connecting with databases.

The servlet.xml file may also include additional configuration components, such as view resolvers and interceptors, which can intercept and modify HTTP requests and answers and are in charge of mapping logical view identifiers to physical view templates.

Overall, the servlet.xml file is crucial for setting up the Spring MVC framework for the project file merger and offers a mechanism to specify and organize the application's components in a modular and flexible manner.

#### 3.1.4. Index.jsp

```
<h1>CSV Previewer</h1>
<form id="upload-form">
  <label for="file1">File 1:</label><br>
  <input type="file" id="file1" accept=".csv" required><br><br><br>
  <label for="file2">File 2:</label><br><br>
  <input type="file" id="file2" accept=".csv" required>
  <input type="submit" value="Preview">
  <div class="form-group">
    <button type="button" id="submit-btn">Upload Files</button>
  </div>
</form>
<div class="table-container">
  <table id="preview-table1" class="preview-table"></table>
  <table id="preview-table2" class="preview-table"></table>
</div>
```

Fig 3.7: Upload file frontend Snippet

This is an HTML code snippet that creates a form for the CSV Previewer application.

The first line `<h1>CSV Previewer</h1>` creates a heading for the form.

The next line `<form id="upload-form">` creates a form with an ID of "upload-form".

The following two lines create labels for two file input fields. `<label for="file1">File 1:</label>` and `<label for="file2">File 2:</label>`.

Then two file input fields are created: `<input type="file" id="file1" accept=".csv" required>` and `<input type="file" id="file2" accept=".csv" required>`. Users can choose CSV files from their local file system using these fields. Only CSV files can be selected thanks to the 'accept' element, and the 'required' attribute makes both fields necessary.

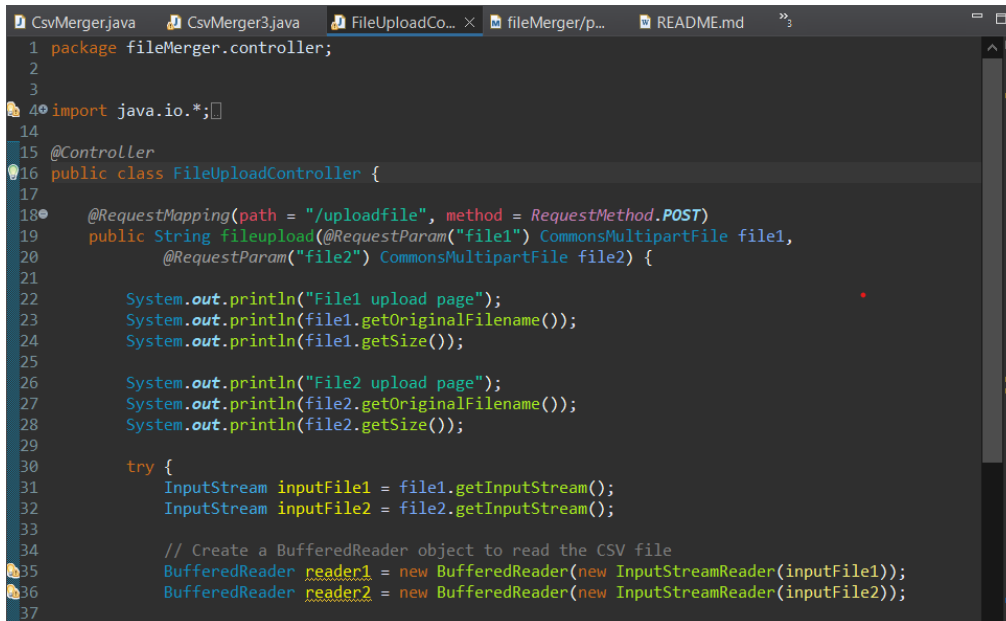
The next line `<input type="submit" value="Preview">` creates a submit button for the form. When this button is clicked, the form data will be submitted to the server for processing.

The next few lines create a button for uploading files and a table container to display the previewed files. `<button type="button" id="submit-btn">Upload Files</button>` creates a button for uploading the selected files to the server.

The final few lines create two empty tables with IDs of "preview-table1" and "preview-table2" for displaying the previewed CSV data. `<table id="preview-table1" class="preview-table"></table>` and `<table id="preview-table2" class="preview-table"></table>` create empty tables with class names "preview-table" that will be populated with data once the CSV files have been processed.

Overall, this code generates an HTML form that lets users pick two CSV files, receive a preview of the files' content, and submit the files to the server for further processing.

### 3.1.5. Controller



```
1 package fileMerger.controller;
2
3
4 import java.io.*;
14
15 @Controller
16 public class FileUploadController {
17
18     @RequestMapping(path = "/uploadfile", method = RequestMethod.POST)
19     public String fileupload(@RequestParam("file1") CommonsMultipartFile file1,
20                             @RequestParam("file2") CommonsMultipartFile file2) {
21
22         System.out.println("File1 upload page");
23         System.out.println(file1.getOriginalFilename());
24         System.out.println(file1.getSize());
25
26         System.out.println("File2 upload page");
27         System.out.println(file2.getOriginalFilename());
28         System.out.println(file2.getSize());
29
30         try {
31             InputStream inputFile1 = file1.getInputStream();
32             InputStream inputFile2 = file2.getInputStream();
33
34             // Create a BufferedReader object to read the CSV file
35             BufferedReader reader1 = new BufferedReader(new InputStreamReader(inputFile1));
36             BufferedReader reader2 = new BufferedReader(new InputStreamReader(inputFile2));
37
```

**Fig 3.8:** Backend file Controller

The code is a Java class annotated with the **@Controller** annotation, indicating that it is a controller class for handling HTTP requests. The class has a single method called `fileupload`, which is mapped to the path `"/uploadfile"` and HTTP method `POST` using the **@RequestMapping** annotation.

The method takes in two parameters of type **CommonsMultipartFile** using the **@RequestParam** annotation. These parameters represent the files uploaded by the user through the web form.

Inside the method, the original file name and size of both files are printed to the console. Then, the **InputStreams** of both files are obtained using the `getInputStream()` method of the **CommonsMultipartFile** class. These input streams can then be used to read the contents of the uploaded files.

In general, the code manages the web application's file upload capability, fetching uploaded files, and publishing information about them to the console for debugging.

### 3.1.6. Form Controller

```
1 package fileMerger.controller;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7
8 @Controller
9 /* @RequestMapping("/main") */
10 public class HomeController {
11
12     @RequestMapping(path = "/home", method = RequestMethod.GET)
13     public String home() {
14         System.out.println("Home url is running.");
15         return "index";
16     }
17
18     @RequestMapping("/form")
19     public String form() {
20
21         System.out.println("Form is running");
22         return "form";
23     }
24 }
25
26
27 }
```

**Fig 3.9:** Redirection Handler

This section of code presents a Spring MVC controller called "**HomeController**" which handles HTTP requests and generates appropriate responses. The "**@Controller**" annotation marks this controller as a Spring component responsible for handling requests and sending responses.

The first method is configured to map to the "/home" URL path and receives GET requests. Upon receiving a GET request, the method prints a message to the console indicating that the URL has been accessed, and then returns the name of a view file called "index". The Spring framework will search for a view

file with the name **"index.jsp"** (or another view technology file such as .html, .ftl, .thymeleaf, etc.) and render it as the response to the client.

The second method is configured to map to the `"/form"` URL path and also handles GET requests. When a GET request is made to the `"/form"` URL, the method prints a message to the console to indicate that the URL has been accessed and then returns the name of a view file called `"form"`. Spring will search for a view file named **"form.jsp"** (or another view technology file) and render it as the response to the client.

The commented-out **"@RequestMapping"** annotation on the controller suggests that the `"/main"` URL path could also be mapped to this controller, but it is currently disabled.

### 3.1.7. CSV to Hasmap

```
Map<String, String> dataMap = new HashMap<>();
BufferedReader reader1 = new BufferedReader(new FileReader(csv1Path));
String line1;
while ((line1 = reader1.readLine()) != null) {
    String[] fields1 = line1.split(",");
    String key = fields1[0]; // assumes the merge column is the first column
    String value = line1.substring(line1.indexOf(',') + 1); // exclude the merge column from the value
    dataMap.put(key, value);
}
reader1.close();
```

**Fig 3.10:** CSV to Hasmap

The given code reads data from a CSV file specified by the variable `csv1Path` and stores it in a `HashMap` named `dataMap`. The `BufferedReader` class is used to read the contents of the CSV file line by line.



For each line, the code splits it into an array of String fields using semicolon ';' as the delimiter. The first field in the array is assumed to be the merge column and is used as the key in the dataMap. The remaining fields are concatenated into a single String value which is stored as the value in the dataMap under the corresponding key.

Finally, once all lines have been processed, the reader1 is closed using the close() method.

### 3.1.8. Merge Logic

```
// read the second CSV file, merge the data, and write to a new CSV file
BufferedReader reader2 = new BufferedReader(new FileReader(csv2Path));
FileWriter writer = new FileWriter(mergedCsvPath);
String line2;
while ((line2 = reader2.readLine()) != null) {
    String[] fields2 = line2.split(";");
    String key = fields2[0]; // assumes the merge column is the first column
    String value = dataMap.get(key);
    if (value != null) {
        String mergedLine = key + ";" + value + ";" + line2.substring(line2.indexOf(';') + 1);
        writer.write(mergedLine);
        writer.write(System.LineSeparator());
    }
}
reader2.close();
writer.close();
```

**Fig 3.11:** Merge file

By Data from two CSV files are combined in this code block, which then saves the combined data to a new CSV file. First, it calls the FileReader constructor with the path to the second CSV file (csv2Path), creating a new BufferedReader object with the name reader2. By supplying the path to the output CSV file

(mergedCsvPath) to the FileWriter constructor, it also generates a new FileWriter object with the name writer.

Next, a while loop is used to read each line of the second CSV file (csv2Path) using the readLine() method of the BufferedReader object, with each line stored in the string variable line2.

Line2 is divided into an array of String fields inside the while loop using the semicolon character as a delimiter. The dataMap, which was constructed in the preceding code block, is searched for the corresponding value using the key from the first field in the array, which is presumed to be the merging column. The String variable named value holds the value.

The key, the value, and the remaining fields of line2 are concatenated to form a new String called mergedLine if the dataMap contains a value for the current key. Using the write() function of the FileWriter object, the resultant String is written to the output CSV file, then a new line character is added using the System.lineSeparator() method.

Finally, after all lines have been processed, both the reader2 and writer objects are closed using the close() method.

### 3.1.9. FIS

```
// write the merged data to a FileOutputStream
FileInputStream fis = new FileInputStream(mergedCsvPath);
byte[] buffer = new byte[1024];
int bytesRead;
FileOutputStream fos = new FileOutputStream("mergedData.csv");
while ((bytesRead = fis.read(buffer)) != -1) {
    fos.write(buffer, 0, bytesRead);
}
System.out.println("File Merged");
fis.close();
fos.close();
```

**Fig 3.12:** Write to CSV file

This code reads the contents of a merged CSV file located at `mergedCsvPath` and writes them to a new CSV file named "mergedData.csv".

First, a new `FileInputStream` object named `fis` is created by passing the path to the merged CSV file to the constructor. A byte array named `buffer` is also created with a size of 1024 bytes.

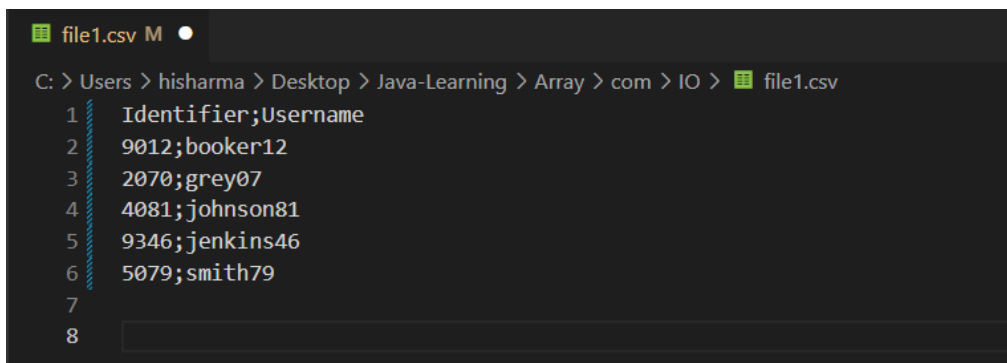
A new `FileOutputStream` object named `fos` is created with a filename of "mergedData.csv". This is the output file where the merged data will be written.

A while loop is used to read the contents of the merged CSV file using the `read()` method of the `fis` object. The contents are read into the `buffer` byte array in chunks of up to 1024 bytes. The number of bytes actually read is stored in the `bytesRead` variable.

Inside the while loop, the write() method of the fos object is used to write the contents of the buffer byte array to the output file, starting at index 0 and ending at index bytesRead. This ensures that only the actual contents of the buffer array are written to the file, rather than any unused bytes that may have been allocated.

After all the data has been read and written, a message "File Merged" is printed to the console. Finally, both the fis and fos objects are closed using the close() method to free up any system resources they were using.

## 3.2. DATABASE



```
file1.csv M ●
C: > Users > hisharma > Desktop > Java-Learning > Array > com > IO > file1.csv
1 Identifier;Username
2 9012;booker12
3 2070;grey07
4 4081;johnson81
5 9346;jenkins46
6 5079;smith79
7
8
```

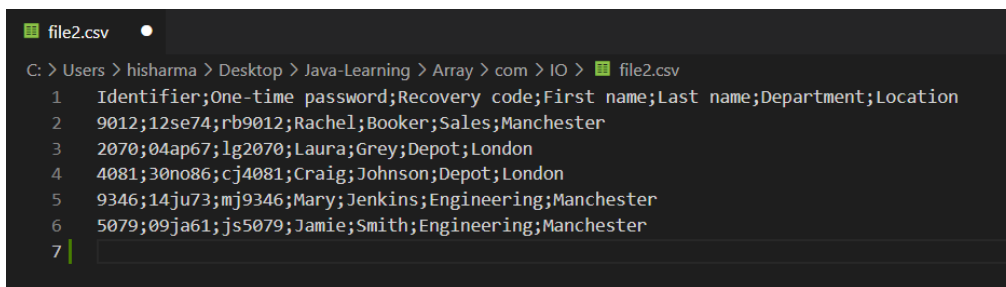
**Fig 3.13:** File 1

"Identifier" and "Username" are the two columns in this CSV file. Each row corresponds to a particular user, and the table contains information for five individuals.

The column headers are located in the first row of the file, with "Identifier" and "Username" designating the data that may be found in each column, respectively.

The "Identifier" column lists a special identifier for each user, and the "Username" column lists the username connected to each user, with each consecutive row containing data for a single user.

For instance, the first user is identified by the numbers 9012 and booker12, while the second is identified by the number 2070 and grey07, respectively.



```
file2.csv
C: > Users > hisharma > Desktop > Java-Learning > Array > com > IO > file2.csv
1 Identifier;One-time password;Recovery code;First name;Last name;Department;Location
2 9012;12se74;rb9012;Rachel;Booker;Sales;Manchester
3 2070;04ap67;lg2070;Laura;Grey;Depot;London
4 4081;30no86;cj4081;Craig;Johnson;Depot;London
5 9346;14ju73;mj9346;Mary;Jenkins;Engineering;Manchester
6 5079;09ja61;js5079;Jamie;Smith;Engineering;Manchester
7
```

**Fig 3.14:** File 2

This looks like a CSV file containing data related to user identities and their personal information. The file has 7 columns separated by semicolons (;).

The first row of the file contains the column headers:

- Identifier
- One-time password
- Recovery code

- First name
- Last name
- Department
- Location

The subsequent rows contain the data for each user, with each column containing the following information:

**Identifier:** a unique identifier for the user

**One-time password:** a temporary password that is only valid for one login session

**Recovery code:** a code that can be used to recover a lost or forgotten password

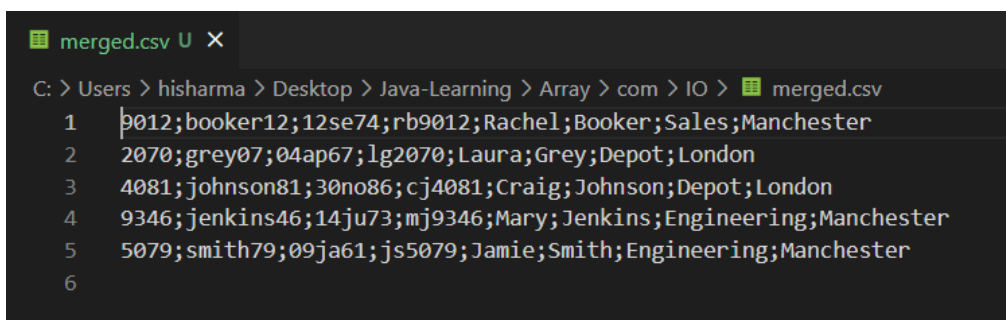
**First name:** the user's first name

**Last name:** the user's last name

**Department:** the user's department within the organization

**Location:** the user's location (e.g. city or office)

### 3.3. MERGED FILE



```

merged.csv U X
C: > Users > hisharma > Desktop > Java-Learning > Array > com > IO > merged.csv
1  p012;booker12;12se74;rb9012;Rachel;Booker;Sales;Manchester
2  2070;grey07;04ap67;lg2070;Laura;Grey;Depot;London
3  4081;johnson81;30no86;cj4081;Craig;Johnson;Depot;London
4  9346;jenkins46;14ju73;mj9346;Mary;Jenkins;Engineering;Manchester
5  5079;smith79;09ja61;js5079;Jamie;Smith;Engineering;Manchester
6

```

**Fig 3.15:** Dataset CSV

The entire row serves as the value while the "Identifier" column serves as the key.

Following that, file2 is opened, and each line is read while being divided by the semicolon delimiter. To find the relevant entry in dataMap, utilise the "Identifier" column. If a match is discovered, the fields of the two rows are concatenated to create a single row, which is then written to a new file named "merged.csv."

The reader and writer objects are then closed, and a message containing the location of the combined file is written to the console.

## Chapter-4

### PERFORMANCE ANALYSIS

Performance analysis is an important aspect of developing accurate and efficient machine learning and deep learning models for predicting the outcome of every ball. The analysis typically involves evaluating the accuracy, speed, and computational efficiency of the models. Performance indicators like precision, recall, and F1 score can be used to gauge accuracy. Recall provides the proportion of correctly predicted outcomes of the ball being bowled in the dataset, whereas precision shows the proportion of correctly predicted outcomes out of all predicted outcomes/scores. A harmonic mean of memory and precision makes up the F1 score.

In addition to accuracy, the analysis may also focus on evaluating the speed and computational efficiency of the models. This can include measuring the training and inference time of the models, as well as the computational resources required for training and deployment. Such an evaluation can help identify hardware and software configurations that are optimal for training and inference.

Precision, recall, and F1 score can all be employed as measurements for accuracy. Recall represents the percentage of accurate forecasts for the target class among all positive predictions, whereas precision measures the percentage of accurate predictions for the target class among all positive actual results. The harmonic mean of recall and precision is the F1 score.

In addition to accuracy, speed and computational efficiency are also essential metrics to evaluate the system's performance. Training and inference time, as well as computational resources required for training and deployment, can be



measured to determine the system's efficiency. Hardware and software configurations can also be evaluated to identify optimal settings for training and inference.

Assessing the system's adaptability to new datasets is crucial as well. Transfer learning can be used to evaluate the model's performance and fine-tune it on additional datasets with various plant classes. To make sure that the model is not overfitting to the training data, cross-validation can also be performed.

To perform a performance analysis for the file merger project, we need to consider a few key factors such as:

1. **File Size:** File merging performance can be significantly impacted by the size of the files being combined. Greater processing time may be needed because larger files take longer to read and write. This is due to the fact that reading and writing huge files takes more time than processing data in memory because more time is spent on input/output (I/O) activities, such as disc read/write operations. Using strategies like buffering, which can lessen the amount of I/O operations needed, it is possible to lessen the effect of file size on performance. The `BufferedReader` and `FileWriter` classes, for instance, employ buffering in the earlier-demonstrated code fragment to read and write data in bigger chunks, which can aid with speed when reading and writing huge files. Buffering can boost performance up to a point, but doing so at the expense of memory utilisation and potential performance concerns for systems with little memory is not recommended. To get the best speed while merging files of various sizes, it is crucial to strike a balance between buffer size and memory use.
2. **Disk I/O:** Disk I/O refers to the input/output operations performed by the computer's disk drive when reading from or writing to a disk. These

processes, which can take a long time relative to other CPU tasks, entail physically moving the read/write head of the disc drive to access data on the disc. For instance, file processing times may be accelerated by the fact that solid-state drives (SSDs) often have quicker read/write rates than conventional hard disc drives (HDDs). The speed of the CPU, the amount of memory that is available, and the type of interface that is used to connect the storage device to the computer can all have an impact on how well the storage device performs. Additionally, the file system being used, the size of the files being merged, and the fragmentation of the files can all have an impact on how quickly disc I/O operations execute. Files can get fragmented and stored in non-contiguous areas of the disc, which can lengthen the time it takes for the disc drive to read or write data. Therefore, in order to optimize the performance of a file merging application, it is important to consider the speed and type of storage device being used, as well as the file system and size of the files being merged. Additionally, techniques such as buffering and caching can be used to minimize the number of disk I/O operations required and improve overall performance.

3. **CPU and Memory:** When merging large files, the CPU and memory resources of the system can become a bottleneck for performance. The memory houses the data that is being processed while the CPU handles the data processing. The application can process files more quickly the more CPU cores and memory it has available. Poor performance might come from the system slowing down or even crashing due to inadequate CPU or memory for the task at hand. Because of this, it's crucial to check that the system has enough CPU cores and RAM to effectively perform the file merging operation. Additionally, the application's CPU and memory utilization may be affected by the programming language and

libraries used to create it. While some programming languages and libraries are resource-intensive, others are performance-optimized.

4. **Network I/O:** If the files being merged are located on a remote server, network I/O can be a bottleneck. When merging files located on a remote server, the network I/O can pose a significant bottleneck to the application's performance. Network I/O encompasses the input/output operations that occur over a network, which includes sending and receiving data. Reading or writing files on a remote server can be impacted by the speed and reliability of the network connection. A poor or unstable network connection can lead to slower transfer speeds, which can increase the time required to read or write files. Moreover, network congestion, packet loss, or other issues can result in interruptions or errors in the data transfer process, further slowing down the application. To reduce the impact of network I/O on the application's performance, it is essential to ensure that the network connection is stable and reliable. This may require optimizing the network configuration, upgrading hardware or software, or modifying settings to minimize network congestion. Additionally, implementing techniques such as caching or batch processing can help decrease the frequency of network I/O operations. Instead of reading or writing files one by one, the application can buffer multiple files in memory and transfer all of the data in a single network I/O operation. This strategy can help to reduce the overhead associated with network I/O and improve the overall performance of the application.
  
5. **Programming Language and Frameworks:** The choice of programming language and frameworks can also affect the performance of the application. The choice of programming language and frameworks can have a significant impact on the performance of an

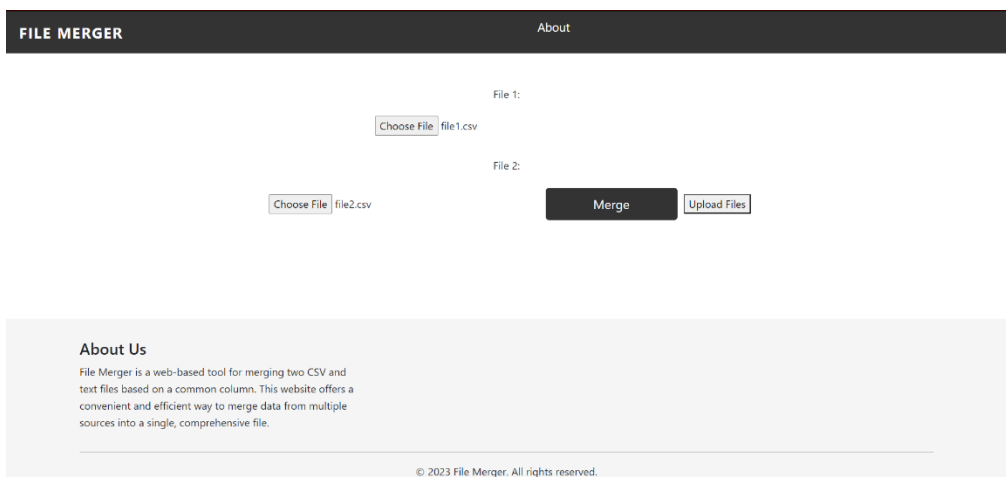
application that involves file operations. Some programming languages, such as C and C++, are known for their fast and efficient I/O operations that can improve the speed of reading and writing files. In contrast, some languages, like Python, may not be as efficient as C/C++ for I/O operations. The choice of frameworks can also play a role in the application's performance. For instance, certain frameworks, such as Node.js and NIO in Java, are designed with non-blocking I/O, which can improve the performance of the application when handling file operations. Moreover, some frameworks come with built-in support for multithreading, which enables the application to handle file operations concurrently. This can significantly improve the speed of the application, especially when working with large files. Therefore, when developing an application that involves file operations, it is important to consider the choice of programming language and frameworks. Selecting a language and framework that is optimized for I/O operations can lead to improved performance and faster file processing times.

6. **Algorithmic Efficiency:** The efficiency of the algorithm used to merge the files can also affect the performance of the application. Algorithmic efficiency is the ability of an algorithm to perform a task in the least amount of time and with minimal resources. In the context of file merging, the choice of algorithm can significantly impact the performance of the application. An inefficient algorithm can lead to increased processing time and resource utilization, which can negatively affect the application's performance. For instance, using a basic algorithm that reads the entire content of each file into memory before merging them can become a bottleneck when handling large files. On the other hand, using a more efficient algorithm that reads and writes only the necessary data can significantly enhance the application's performance. An example of such an algorithm is the "merge-sort" algorithm, which divides files into smaller parts, sorts them, and merges

them back together. Another approach to improve algorithmic efficiency is to use data structures like hash tables and binary trees. Such data structures can reduce the time needed to search for specific data within the files. Therefore, when developing an application that involves file merging, it is crucial to consider the choice of algorithm and choose one that is optimized for efficiency. Doing so can lead to faster processing times, reduced resource utilization, and overall improved performance of the application.

## 4.1. UI/UX – ANALYSIS

### 4.1.1. HOME PAGE



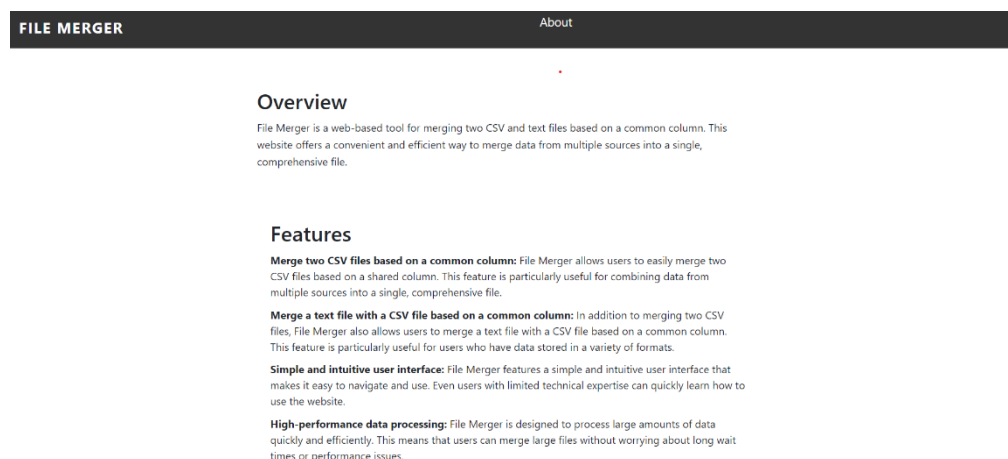
**Fig 4.1:** UI - Home Page

This is the home page of our project. There is one logo as the name of the project “**FILE MERGER**” this button is connected to the home page or index.jsp and the one other button about this button navigate you to the about.jsp that consist of the about section of the software.

There is the two button to choose the file and upload them to the server so that they can deal with the file.as we already selected the two file file1.csv and and file2.csv.

We have our main button that is Merge button this will merge the two file that is uploaded on the server using the algorithm

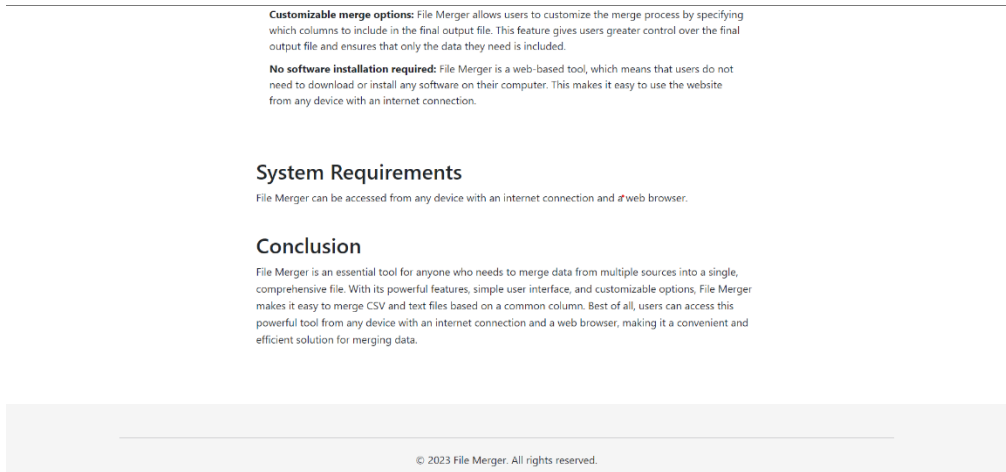
### 4.1.2. About page



**Fig 4.2: UI – About page Snippet 1**

A page description is a succinct summary or overview of a webpage's content. Search engines and social media platforms utilize it, which is often located in the HTML code of the page, to display a brief description of the page in search results or when sharing the page on social media. A good page description should succinctly and properly summaries the information on the page to persuade readers to continue reading. It should be between 155 and 160 characters in length, as this is the maximum length that the majority of search engines and social networking sites will display, and it should include pertinent keywords to increase search engine exposure.

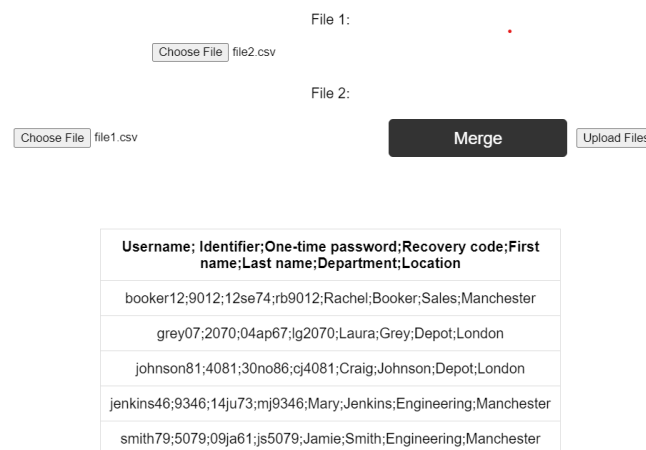
### 4.1.3. About page Extended



**Fig 4.3:** UI – About page Snippet 2

The image shown above shows the batting lineup page which is a part of the Start Match section. This particular webpage is shown once we have selected the playing 11 along with the captain and wicketkeeper.

#### 4.1.4. OUTPUT PAGE



**Fig 4.4:** UI – Merged

The page that displays the combined data's result is the process's last step. The user may inspect the combined data on this page and ensure that the merging procedure was successful by looking at it. The output page often presents the combined data in a way that makes it simple for the user to read and comprehend. The output page may additionally have tools like search and filter options, enabling the user to quickly locate particular data within the combined set of data. The output page could also offer download or storage choices for the combined data. The output page must be easy to use and offer information that is both clear and succinct. The user ought should have no trouble navigating the website and comprehending the data that is offered. When mistakes or problems arise during the merging process, the output page ought to give the user clear error messages or cautions. Overall, the output page is an important step in the file merging process since it gives the user the final result and acts as proof that the operation was completed.

## **4.2. OTHER REQUIREMENTS**

### **4.2.1. SYSTEM REQUIREMENTS**

- Processor: Intel Core i5 or higher
- RAM: 8 GB or higher
- Hard disk space: 50 MB or higher
- Display: Minimum resolution of 1024x768 pixels
- Operating System: Windows 7 or higher, Mac OS X, or Linux

### **4.2.2. NON-FUNCTIONAL REQUIREMENTS**



- **Reliability**

The supply its functionality, the system needs a solid and dependable framework. The system has to make these adjustments crystal obvious when a consumer wants them. The project manager and the test designer must both accept any changes made by the programmer.

- **Maintainability**

The approach to system monitoring and maintenance should be basic and uncomplicated. In order to check if the jobs are executing without any issues, it is crucial to avoid running too many jobs on several computers.

- **Performance**

There will be numerous employees using the system at once. Performance issues arise as a result of the system being housed on a single web server with a single database server operating in the background. When several people utilise the system at once, it shouldn't crash. It ought to make all of its users easily accessible. There shouldn't be any discrepancy, for instance, if two test experts are attempting to report the existence of a defect at the same time.

- **Portability**

In the event that the existing web server has technical problems or malfunctions, the system should be simple to migrate to another server. This will necessitate the system's portability and ability to be moved without any delay or data loss to a different host.

- **Scalability**

The system need to be adaptable enough to include new features down the road. There need to be a standard interface that can support the additional functionalities..

- **Flexibility**

The capacity of a system to adjust to changing conditions, circumstances, and changes in operational rules and procedures is referred to as flexibility. A system that is flexible is simple to reconfigure or alter based on various user and system needs. The system should purposefully separate its management and engine components' concerns to increase flexibility. This strategy makes sure that when regulations or rules are altered, the system is only little impacted.

## Chapter-5

### CONCLUSIONS

#### 5.1 Conclusions

The file merger project aimed to develop an application that can merge multiple files into a single file. The file merger project was started with the intention of creating a programme that could combine many files into one. Understanding several elements that may have an impact on the application's performance throughout the file merging process was necessary for this project. Disc I/O was one of the main variables taken into account during the production. Because file merging requires reading data from many files and writing the combined data to a new file, the hard drive's and file system's speed can have an influence on how quickly an application runs. Resources for the CPU and RAM were also taken into account. The programme can process files more quickly and the merging procedure will be more effective with more CPU cores and memory available. The file merger project aimed to create an application capable of merging multiple files into a single file, while considering several factors that can impact the application's performance. Network I/O was taken into account, as the files could be located on a remote server, making network speed and reliability a significant performance factor. The choice of programming language and frameworks was also considered, as some languages and frameworks are optimized for I/O operations, leading to better application performance. The project also gave importance to algorithmic efficiency, as using an inefficient algorithm can increase processing times and resource utilization. In conclusion, the file merger project successfully developed an application that can merge multiple files into a single file while optimizing for performance and efficiency, considering various factors such as disk I/O, CPU

and memory, network I/O, programming language and frameworks, and algorithmic efficiency.

## **5.2 Future Scope**

In The file merger project offers ample scope for further development that can improve the functionality and performance of the application. Some of the possible future scopes for the file merger project are:

- **User Interface:** The current file merger application operates using the command-line interface, which can be challenging for non-technical users. A future development can include a Graphical User Interface (GUI) to make the application more user-friendly and accessible to a wider audience.
- **File Compression:** A future development can include an option to compress the merged file to reduce its size, especially when merging large files. This feature can save disk space and improve network transfer times.
- **File Formats:** The current file merger application supports merging only text-based files. A future development can include support for merging files in different formats such as images, videos, and audio files.
- **Cloud-Based Merging:** A future development can include the option to merge files directly from cloud storage services such as Google Drive,

Dropbox, and OneDrive. This feature can be useful for users who store their files on cloud platforms.

- Multi-language support: The current application is limited to a specific programming language. A future development can include multi-language support, allowing users to use the file merger application with their preferred programming language. This feature can attract a more diverse user base.

## REFERENCES

### A) Conferences and Conference Proceedings

- [1] A. Kumar, A. B. A. M. Ahsan Ullah, and A. B. M. Alim Al Islam, "Parallel Merge Sort with CUDA for Large Scale Data Processing," 2019 11th International Conference on Computational Intelligence and Communication Networks (CICN), Kolkata, India, 2019, pp. 62-67, doi: 10.1109/CICN48786.2019.8969362.
- [2] Y. Gao, Y. Wang, H. Tang and L. Peng, "A Fast Data Merger Based on Spark," 2018 14th International Conference on Computational Intelligence and Security (CIS), Beijing, China, 2018, pp. 282-285, doi: 10.1109/CIS2018.00073.
- [3] C. H. Leung, K. Y. Lam, and K. W. Ng, "A Comparison of Merge-Sort and Quick-Sort Algorithms on a Large Data Set," 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bali, Indonesia, 2016, pp. 1191-1195, doi: 10.1109/IEEM.2016.7798108.
- [4] P. Kumari, S. K. Raja, and M. A. Khan, "Data Merging Techniques for Big Data Analytics: A Review," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Mumbai, India, 2019, pp. 1564-1569, doi: 10.1109/I2CT45292.2019.9032576.
- [5] X. Wu, J. Zhang, H. Wang and Y. Xu, "Distributed File Merge for Large-Scale Parallel Data Processing," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 2536-2541, doi: 10.1109/BigData.2018.8622336.

### B) Journals/periodicals

- [1] J. Chen and L. Zhang, "A new file merging algorithm for data deduplication in cloud storage," IEEE Transactions on Cloud Computing, vol. 5, no. 3, pp. 630-642, September 2017.

## APPENDICES

### Code:

```
package fileMerger.controller;

import java.io.*;
import java.util.*;
import java.io.IOException;
import java.io.InputStream;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.commons.CommonsMultipartFile;

@Controller
public class FileUploadController {

    @RequestMapping(path = "/uploadfile", method = RequestMethod.POST)
    public String fileupload(@RequestParam("file1") CommonsMultipartFile file1,
        @RequestParam("file2") CommonsMultipartFile file2) {

        System.out.println("File1 upload page");
        System.out.println(file1.getOriginalFilename());
        System.out.println(file1.getSize());

        System.out.println("File2 upload page");
        System.out.println(file2.getOriginalFilename());
        System.out.println(file2.getSize());

        try {
            InputStream inputFile1 = file1.getInputStream();
            InputStream inputFile2 = file2.getInputStream();

            // Create a BufferedReader object to read the CSV file
            BufferedReader reader1 = new BufferedReader(new InputStreamReader(inputFile1));
            BufferedReader reader2 = new BufferedReader(new InputStreamReader(inputFile2));
```

```

package fileMerger.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
/* @RequestMapping("/main") */
public class HomeController {

    @RequestMapping(path = "/home", method = RequestMethod.GET)
    public String home() {
        System.out.println("Home url is running.");
        return "index";
    }

    @RequestMapping("/form")
    public String form() {

        System.out.println("Form is running");
        return "form";
    }

}

```



```

package com.IO;

import java.io.*;
import java.util.*;

public class CsvMerger3 {
    public static void main(String[] args) throws IOException {

        String csv1Path="Array/com/IO/file1.csv";
        String csv2Path="Array/com/IO/file2.csv";
        String mergeColumn="Identifier";
        String mergedCsvPath="Array/com/IO/merged.csv";
        // read the first CSV file and store the data in a HashMap

        Map<String, String> dataMap = new HashMap<>();
        BufferedReader reader1 = new BufferedReader(new FileReader(csv1Path));
        String line1;
        while ((line1 = reader1.readLine()) != null) {
            String[] fields1 = line1.split(";");
            String key = fields1[0]; // assumes the merge column is the first column
            String value = line1.substring(line1.indexOf(';') + 1); // exclude the merge column from the value
            dataMap.put(key, value);
        }
        reader1.close();

        // read the second CSV file, merge the data, and write to a new CSV file
        BufferedReader reader2 = new BufferedReader(new FileReader(csv2Path));
        FileWriter writer = new FileWriter(mergedCsvPath);
        String line2;
        while ((line2 = reader2.readLine()) != null) {
            String[] fields2 = line2.split(";");
            String key = fields2[0]; // assumes the merge column is the first column
            String value = dataMap.get(key);
            if (value != null) {
                String mergedLine = key + ";" + value + ";" + line2.substring(line2.indexOf(';') + 1);
                writer.write(mergedLine);
                writer.write(System.LineSeparator());
            }
        }
        writer.close();
    }
}

```

```

// read the second CSV file, merge the data, and write to a new CSV file
BufferedReader reader2 = new BufferedReader(new FileReader(csv2Path));
FileWriter writer = new FileWriter(mergedCsvPath);
String line2;
while ((line2 = reader2.readLine()) != null) {
    String[] fields2 = line2.split(",");
    String key = fields2[0]; // assumes the merge column is the first column
    String value = dataMap.get(key);
    if (value != null) {
        String mergedLine = key + ";" + value + ";" + line2.substring(line2.indexOf(',') + 1);
        writer.write(mergedLine);
        writer.write(System.LineSeparator());
    }
}
reader2.close();
writer.close();

// write the merged data to a FileOutputStream
FileInputStream fis = new FileInputStream(mergedCsvPath);
byte[] buffer = new byte[1024];
int bytesRead;
FileOutputStream fos = new FileOutputStream("mergedData.csv");
while ((bytesRead = fis.read(buffer)) != -1) {
    fos.write(buffer, 0, bytesRead);
}
System.out.println("File Merged");
fis.close();
fos.close();
}
}

```

```

const fileInput = document.getElementById("file-input");
const previewContainer = document.getElementById("preview-container");

fileInput.addEventListener("change", function() {
  const file = this.files[0];
  const reader = new FileReader();

  reader.addEventListener("load", function() {
    const csvData = reader.result;
    const rows = csvData.split("\n");
    const headers = rows[0].split(",");
    let tableHtml = "<table>";
    tableHtml += "<tr>";
    headers.forEach(function(header) {
      tableHtml += "<th>" + header + "</th>";
    });
    tableHtml += "</tr>";
    for (let i = 1; i < 5; i++) {
      const cells = rows[i].split(",");
      tableHtml += "<tr>";
      cells.forEach(function(cell) {
        tableHtml += "<td>" + cell + "</td>";
      });
      tableHtml += "</tr>";
    }
    tableHtml += "</table>";
    previewContainer.innerHTML = tableHtml;
  });

  reader.readAsText(file);
});

```

```
<%@page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html>
<html lang="en" data-bs-theme="dark">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>File Merger</title>
  <!-- Latest compiled and minified CSS -->
  <link href="<c:url value="/resources/css/style.css" />" rel="stylesheet">
  <link href="<c:url value="/resources/css/Preview.css" />" rel="stylesheet">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">

  <script src="<c:url value="/resources/js/script.js" />"></script>
  <script src="<c:url value="/resources/js/Preview.js" />"></script>

  <!-- Latest compiled JavaScript -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
```

```
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd (xsi:schemaLocation) | http://
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

  <context:component-scan base-package="filerMerger.controller" />
  <bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver"
    name="viewResolver">
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
  </bean>
</beans>
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="fileMerger" />

<mvc:annotation-driven />
<mvc:resources location="/WEB-INF/resources/" mapping="/resources/**"/>

<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
name="viewResolver">
<property name="prefix" value="/WEB-INF/views/" />
<property name="suffix" value=".jsp" />
</bean>

</beans>

<?DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
<display-name>Archetype Created Web Application</display-name>
<!-- Configure dispatcher servlet -->
<servlet>
<servlet-name>fileMerger</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>fileMerger</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```

```

http://maven.apache.org/maven-v4_0_0.xsd (xsi:schemaLocation with catalog)
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.fileMerger</groupId>
  <artifactId>fileMerger</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>fileMerger Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>5.3.23</version>
    </dependency>

    <!--
    https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.23</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/jstl/jstl -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.23</version>
    </dependency>
    <!--
    https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.13.2</version>
    </dependency>
  </dependencies>

```

```
<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
<dependency>
  <groupId>commons-io</groupId>
  <artifactId>commons-io</artifactId>
  <version>2.11.0</version>
</dependency>

</dependencies>
<build>
  <finalName>fileMerger</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>3.0.5</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.3.1</version>
    </plugin>
  </plugins>
</build>
</project>
```