Chapter 1

INTRODUCTION

1.1 Preamble

As per 2015 there are more than two billion Personal Computers and Computing devices in use. The need for computers is constantly increasing. As the number of computers is increasing, there is an exponential rise in the number of files. These high number of files require proper management and organization. A small glitch in file management might leave the file system in a complete disarray. Looking up for a particular file in such large file systems is a tedious process. Furthermore there is a possibility of more than one copy of the same file existing in the file system leading to redundancy in the file system.

Upon our observation we have noticed that there are many PDF files that are downloaded from the internet (e.g. A technical paper from IEEE site)which might be obscurely named and keeping track of such files is nearly an impossible task.

These observed and faced drawbacks served as an incentive for us to develop an application which helps the user in various purposes (categorizing, tagging, renaming, etc.).

Our application logically aggregates files based on their extension into 6 main categories viz. Audio, Video, Pictures, Documents, Executables and Others. It helps to store information regarding file organization in storage media like internal and external HDDs etc. It reminds the user to backup important documents and photos. Movies and TV shows can be categorized into watched and unwatched so that users can delete watched ones to gain space.It suggests the user to rename vaguely named documents like PDFs files. Our application exhibits robust searches and these searches are implemented individually for each categories (documents, audio, etc.).

1.2 Motivation

The main driving force behind our application were two things; Firstly, on our observation many users have the tendency of keeping their file system extremely disorganized and searching for any file in such a chaotic file system may be a tiresome process. The second thing which we observed was the number of vaguely named PDFs were considerably high and what these PDFs were pertaining to was a big question mark?

So in order to address this problem we came up with this idea of developing the application which helps the user to maintain and organize his/her ever-growing file system.

1.3 Existing System

The existing system as of today has a typical built-in file explorer on any Windows OS. It is efficient but it lacks the speed and navigation in case of a huge file system. Also it is inefficient in keeping track of duplicate files present in the file system. Many a times we tend to lose track of the location of backed up files.

We are often required to manually rename the vaguely named PDFs files.

Many PCs and Laptops often get filled with watched movies and TV shows thus occupying memory space which can be utilized for storing other important documents.

Users often forget to back up photos and documents of importance.

1.4 Proposed System

This application will help aggregate photos, music, videos, document and executables. The user is given the option of renaming vaguely named PDF files. It will also store the file organization of backed up media.

TV shows and movies can be categorized into watched and unwatched. This application also provides the option for adding photos and documents to a category viz. “Remind for

Backup" so that the user can be reminded at regular intervals for backing-up the important documents.

Our application has also provided the feature of better and efficient search were searching is done in each individual categories, thus making the process searching faster.

1.5 Organization of Report

This section is intended to give the reader a brief overview of the structure of this document and the composition of each chapter.

- ❖ **Chapter 1 "INTRODUCTION"** deals with Preamble, Motivation/Relevance, Existing System, and Proposed Solution.
- ❖ **Chapter 2 "SYSTEM REQUIREMENTS"** deals with the software requirement specification, and software and hardware requirements.
- ❖ **Chapter 3 "SYSTEM DESIGN"** includes system design and various diagrams which represents the flow of the project.
- ❖ **Chapter 4 "IMPLEMENTATION ENVIRONMENT"** deals with the overview of NetBeans and Scene Builder and SQLite database manager.
- ❖ **Chapter 5 "ORGANIZATION INTO CATEGORIES"** deals with categorizing files and storing their references.
- ❖ **Chapter 6 "SEARCHING"** deals with implementing search within categories.
- ❖ **Chapter 7 "PDF RENAMING"** includes prompting the user to rename obscurely named PDFs.
- ❖ **Chapter 8 "TAGGING"** gives user the option to tag files appropriately.
- ❖ **Chapter 9 "SNAPSHOTS"** concludes the project and about future enhancement.
- ❖ **Chapter 10 "CONCLUSION"** concludes the project and about future enhancement.

Chapter 2

SYSTEM REQUIREMENTS

2.1 Software Requirement Specifications

Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform in identifying the variety of rice grains and determining its accuracy. The SRS is a requirements specification for a software system, is a description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Nonfunctional requirements impose constraints on the design and implementation. Software requirements specification establishes the basis for agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated.

2.1.1 Functional Requirement

The functional requirement of this project is described as a function which is set of inputs, the behavior, and outputs. The basic and the fundamental requirement for this application is to have complete access to the File system in order to perform basic operations and functions. As our application renames the PDF from within the application we need to have execute permission from the OS so that the above said operation can be performed. Tagging of files in our application requires access permission of various properties like date access time, date modified etc. When a file is created or deleted the native API inform the java API about the events, and accordingly these events are processed and updated in the application UI.

2.1.2 Non-Functional Requirement

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements". A quality, that is non-functional requirement, can be divided into two main categories:

1. Execution qualities, such as dynamic and usability, which are observable at run time.

2. Evolution qualities, such as testability, maintainability and robustness of the application data.

2.2 Software Requirements

Operating System is Windows 7/8/10/Vista machine.

To develop the backend of the application, we have used NetBeans. NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers.

To develop the front-end of the application we have used JavaFX Scene Builder. JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application’s logic.

We have also used SQLITE database, which is an embedded database which uses the sqlitejdbc library.
We have made use of Java JDK 8 which is an implementation of one of the Java SE. Our application has also made of iText Library in aiding the process of PDF renaming.
iText is an open source library for creating and manipulating PDF files in Java.

2.3 Hardware Requirements

- ❖ Computer System for execution of project, with minimum of 1GB of free hard disk space and 512MB of RAM or above.

2.4 Performance Requirement

This project satisfies the following performance requirements:

• Speed: The application performs quick search and produces correct results.

• Portability: The GUI of this application is user friendly, so it is very easy for the user to understand and respond to the same. Multiple User Interface is the evidence for the good portability of the system.

• Reliability: This system has high probability to deliver the required queries and the functionalities available in the application. This project must meet all the functional requirements. The visualization module makes the system more reliable.

• Scalability: The system can be extended to integrate the modifications done in the present application to improve the quality of the product. This is meant for the future works that needs to be done on the application.

•Dynamic: The application is dynamic because of the fact that whenever a user performs file manipulation operation, the application instantly reflects the change.

•Robust: The application is robust in nature because of the proper implementation of background services.

•Agile: The application is agile for we have adopted and used agile software development strategies.

•Efficient: The application is throughout efficient enough in producing desired results in form of various user queries.

Chapter 3

SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. We have made use of Model View Controller paradigm.

3.1 Model-View-Controller

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

3.1.1 Model

The central component of MVC, the model, captures the behavior of the application in terms of its problem domain, independent of the user interface. The model directly manages the data, logic and rules of the application. The following classes represent the models used in our application.

ScanFileSystem:

This class scans the file system and enters file references into database.

DontScan:

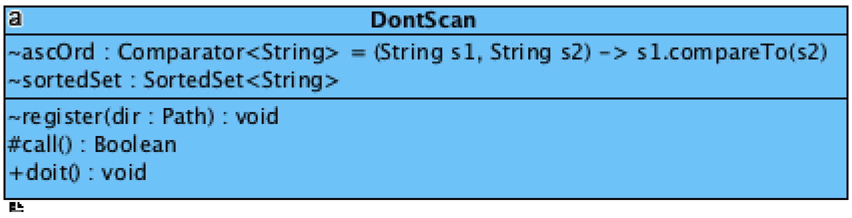
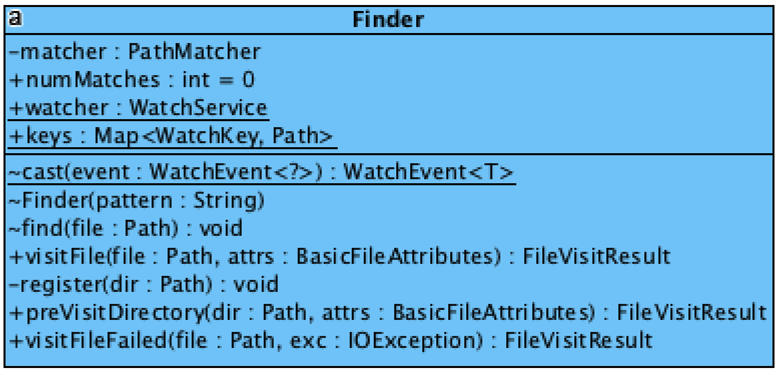

a
DontScan
~ascOrd : Comparator<String> = (String s1, String s2) -> s1.compareTo(s2)
~sortedSet : SortedSet<String>
~register(dir : Path) : void
#call() : Boolean
+doit() : void

Figure. 3.1: Diagram depicting organization of DontScan class

This class reads tables from database which contains file references and sets respective map variable.

Finder:

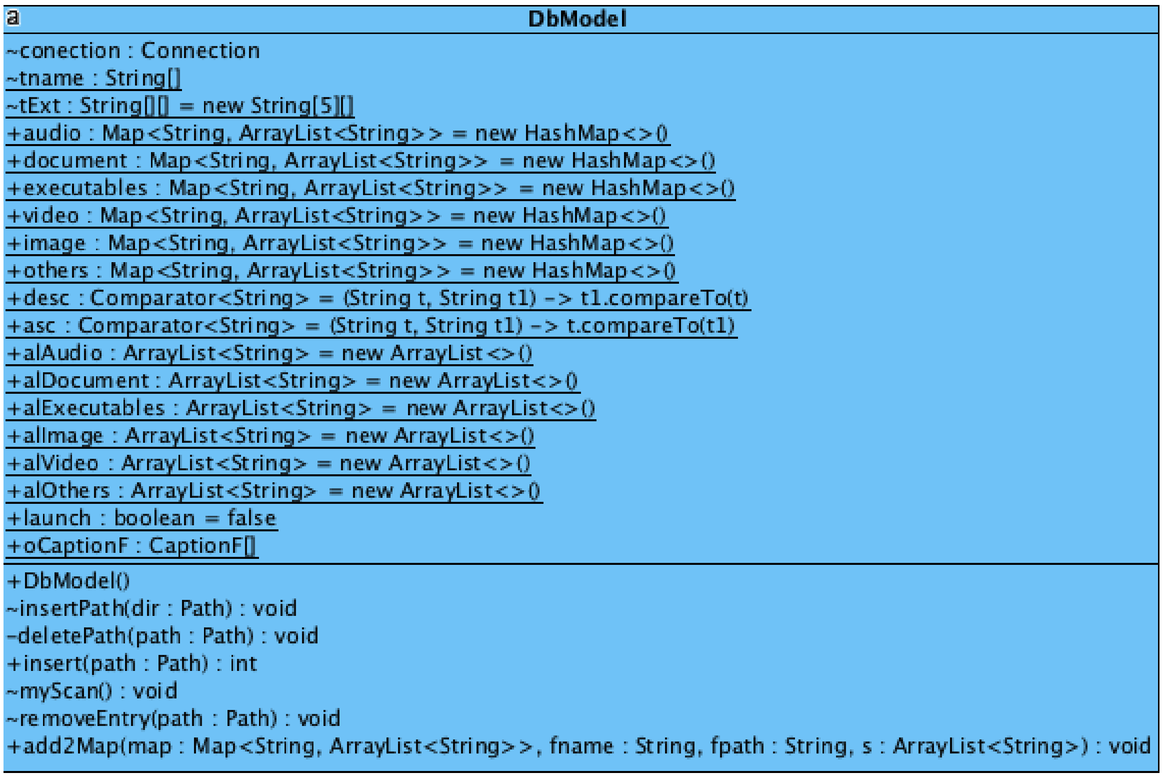


```
Finder
-matcher : PathMatcher
+numMatches : int = 0
+watcher : WatchService
+keys : Map<WatchKey, Path>
~cast(event : WatchEvent<?>) : WatchEvent<T>
~Finder(pattern : String)
~find(file : Path) : void
+visitFile(file : Path, attrs : BasicFileAttributes) : FileVisitResult
-register(dir : Path) : void
+preVisitDirectory(dir : Path, attrs : BasicFileAttributes) : FileVisitResult
+visitFileFailed(file : Path, exc : IOException) : FileVisitResult
```

Figure. 3.2: Diagram depicting organization of Finder class

ScanFileSystem uses finder to walk the file tree and do necessary operations.

DbModel:



```
DbModel
~conection : Connection
~tname : String[]
~tExt : String[][] = new String[5][]
+audio : Map<String, ArrayList<String>> = new HashMap<>()
+document : Map<String, ArrayList<String>> = new HashMap<>()
+executables : Map<String, ArrayList<String>> = new HashMap<>()
+video : Map<String, ArrayList<String>> = new HashMap<>()
+image : Map<String, ArrayList<String>> = new HashMap<>()
+others : Map<String, ArrayList<String>> = new HashMap<>()
+desc : Comparator<String> = (String t, String t1) -> t1.compareTo(t)
+asc : Comparator<String> = (String t, String t1) -> t.compareTo(t1)
+alAudio : ArrayList<String> = new ArrayList<>()
+alDocument : ArrayList<String> = new ArrayList<>()
+alExecutables : ArrayList<String> = new ArrayList<>()
+alImage : ArrayList<String> = new ArrayList<>()
+alVideo : ArrayList<String> = new ArrayList<>()
+alOthers : ArrayList<String> = new ArrayList<>()
+launch : boolean = false
+oCaptionF : CaptionF[]
+DbModel()
~insertPath(dir : Path) : void
-deletePath(path : Path) : void
+insert(path : Path) : int
~myScan() : void
~removeEntry(path : Path) : void
+add2Map(map : Map<String, ArrayList<String>>, fname : String, fpath : String, s : ArrayList<String>) : void
```

Figure. 3.3: Diagram depicting organization of DbModel class

This class is used to retrieve extensions of different categories. It also inserts file attributes to the data base. It also helps removing a file reference if the file is deleted from the file system.

WatchDir:

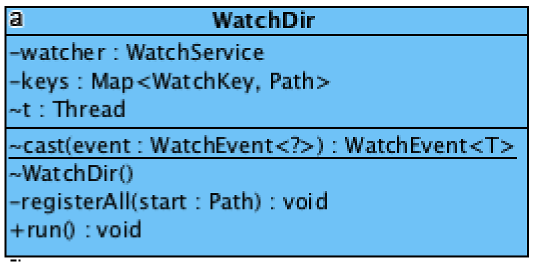

a
WatchDir
-watcher : WatchService
-keys : Map<WatchKey, Path>
~t : Thread
~cast(event : WatchEvent<?>) : WatchEvent<T>
~WatchDir()
-registerAll(start : Path) : void
+run() : void

Figure. 3.4: Diagram depicting organization of WatchDir class

WatchDir class uses built-in java class called watch service which in turn uses native (OS specific) support for getting notifications regarding file creation and deletion.

VagueFileDiscovery:

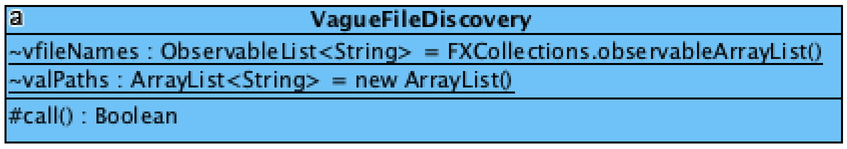

a
VagueFileDiscovery
~vfileNames : ObservableList<String> = FXCollections.observableArrayList()
~valPaths : ArrayList<String> = new ArrayList()
#call() : Boolean

Figure. 3.5: Diagram depicting organization of VagueFileDiscovery class

It identifies the vaguely named PDF files. We are using a database containing list of legitimate words extracted words extracted from Wiktionary.

PDFTitleExtraction:

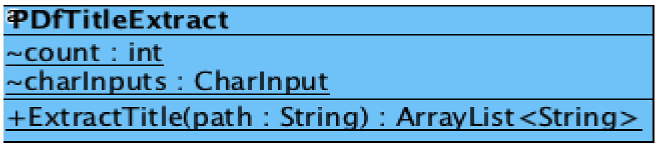


Figure. 3.6: Diagram depicting organization of VagueFileDiscovery class

We extract a list of possible titles (maximum 6 suggestion) and which is used to rename the PDF.

3.1.2 View

Welcome:

Welcome greets the user based on Time API's and login name. It provides the option to the user to scan or not scan the user.

Home Page:

Home page displays files in a categorized manner (in different panes). It provides options to view tagged files.

Search Window

This window is reused by many modules. It provides a text field to search for a file. It displays file information in a tabular way (filename and extension) which has inherent capability of ordering and grouping (based on extension).

PDF renaming window

It provides a list of probably vaguely named PDFs. A user can double click on a filename and rename the PDF based on suggestions provided.

3.1.3 Controller

Welcome Screen controller

It initializes the welcome screen that is greets the user based on time and login name and receives the input for scanning/not scanning.

Home Screen Controller

It displays files based on categories in different panes. It handles user events.

Search window controller

It displays search results to the user.

PDF renaming controller

It displays vaguely named PDFs and retrieves user input from combo box for renaming.

Chapter 4

IMPLEMENTATION ENVIRONMENT

4.1 NetBeans

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers. The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5.

NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactoring's, version control (supporting CVS, Subversion, Git, Mercurial and Clear case).Modularity: All the functions of the IDE are provided by modules. Each module provides a well-defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN.

NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the NetBeans IDE.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
  - NetBeans Visual Library
  - Integrated development tools

4.2 JavaFX Scene Builder

- **Java FX:-** JavaFX is a software platform for creating and delivering desktop applications, as well as rich internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and Mac OS X.

  Before version 2.0 of JavaFX, developers used a statically typed, declarative language called JavaFX Script to build JavaFX applications. Because JavaFX Script was compiled to Java bytecode, programmers could also use Java code instead. JavaFX applications could run on any desktop that could run Java SE, on any browser that could run Java EE, or on any mobile phone that could run Java ME. JavaFX 2.0 and later is implemented as a native Java library, and applications using JavaFX are written in native Java code.

  JavaFX 2.x platform includes the following components:
  1). The JavaFX SDK: runtime tools. Graphics, media web services, and rich text libraries. Java FX 1.x also included JavaFX compiler, which is now obsolete as JavaFX user code is written in Java.
  2). NetBeans IDE for JavaFX: NetBeans with drag-and-drop palette to add objects with transformations, effects and animations plus a set of samples and best practices. For JavaFX 2 support you need at least NetBeans 7.1.1. For Eclipse users there is a community-supported plugin hosted on Project Kenai.

- **Java FX Scene Builder**: JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application's logic.The various features of Scene Builder are-:

- UI Layout Tool- Scene Builder allows you to easily layout JavaFX UI controls, charts, shapes, and containers, so that you can quickly prototype user interfaces.

Animations and effects can be applied seamlessly for more sophisticated UIs.

- FXML Visual Editor- Scene Builder generates FXML, an XML-based markup language that enables users to define an application's user interface, separately from the application logic. You can also open and edit existing FXML files authored by other users.
- Integrated Developer Workflow- Scene Builder can be used in combination with any Java IDE, but is more tightly integrated with NetBeans IDE. You can bind the UI to the source code that will handle the events and actions taken on each element through a simple process, run your application in NetBeans, and any changes to FXML in NetBeans will also reflect in your Scene Builder project.
- Preview Your Work- At any time during the creation of your project, you can preview what the user interface will really look like when deployed, unencumbered by the tool's menus and palettes.
- Cross Platform, Self-Contained- Scene Builder is written as a JavaFX application, supported on Windows, Mac OS X and Linux. It is the perfect example of a full-fledge JavaFX desktop application. Scene Builder is packaged as a self-contained application, which means it comes bundled with its own private copy of the JRE.
- CSS Support- You can apply the look and feel of your choice to your GUI layout by using style sheets. It's as easy as selecting a GUI component, and pointing to the CSS file of your choice from the Properties Panel. The CSS analyzer allows you to understand how specific CSS rules can affect aspects of a JavaFX component.

❖ **SQLite Database**: SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite has bindings to many programming languages.

Chapter 5

ORGANIZATION INTO CATEGORIES

5.1 Overview

Our Application aims to provide a very comprehensible view of the file system to the user. This is done by categorizing the files into widely accessed categories namely document, audio, video, images and executables. We have also include a category known as other where files which do not fall into the above mentioned categories, these files are mostly system files like dll, BAT, ini, inif etc. which are rarely accessed by the user.

We have designed this application in such a way that we extract the extension of the file system and depending these extensions we insert these files into the respective table in the database. The database consists of six tables namely audio (with extensions such as .mp3 etc.), video (with extensions such as .mp4, .avi etc.), document (with extensions such as .pdf, .docx etc.), executables (with extensions such as .exe, .msi, .apk etc.), image (with extensions such as .gif, .png, .jpg etc.) and others for storing files belonging to different categories.

Extensions belonging to the above mentioned file categories are stored in five tables respectively. For instance consider the case of document categories wherein which we have a table named document_ext and its field are extensions and information. The extension field stores all the extension of the files and the information field stored a short description pertaining to the corresponding extension.

We provide to options namely scan the file system and don’t scan the file system. The scan option walks the file tree. While walking the file tree we store file information like name, extension, path etc. in the respective tables. We create in-memory hashmaps for every category mentioned, the key of the hashmaps is the filename and value is the list of corresponding paths (A filename can have multiple paths if multiple copies of the file is present) .Don’t scan option retrieves the filenames from the database, sets the corresponding hashmaps and starts the application.

Hashmaps will be used to facilitate the searching of files. Since there are hashmaps corresponding to different categories, searching will be very intuitive.

5.2 Pseudocode

Algorithm: Organization of files into categories

Input: Table of extensions and table containing file references

Output: Categorized hashmaps

Start

```
Step1: Extract extension from database.
```

Step2a: If user wants to scan.

I. Walk the file tree
II. Insert into respective tables and hashmaps based on extension.

Step2b: If user does not want to scan.

I. Extract file information (name and path) from respective category tables.
II. Set respective hashmaps.

Step3: Display files pertaining to mentioned categories in appropriate GUI elements respectively.

Stop

5.3 Code Snippet

```
public int insert(Path path) {

        int resultSet = 0, tIndex = 0;

        boolean flag = false;

        String fname;

        fname = path.getFileName().toString();

        String fpath = path.toString();



        String extension = "";

        int i = fname.lastIndexOf('.');

        if (i > 0) {

            extension = fname.substring(i + 1);

        }

        String root = fpath.substring(0, 1);
```

```
label:
for (int j = 0; j < 5; j++) {
    for (String s : tExt[j]) {
        if (extension.compareToIgnoreCase(s) == 0) {
            tIndex = j;
            flag = true;
            break label;
        }
    }
}
if (!flag) {
    try {
        PreparedStatement preparedStatement = null;
        String query = "INSERT INTO OTHERS VALUES (?,?, ?, ?,?,?);";
        preparedStatement = conection.prepareStatement(query);
        preparedStatement.setString(2, fname);
        preparedStatement.setString(3, extension);
        preparedStatement.setString(4, fpath);
        preparedStatement.setString(5, root);
        preparedStatement.setString(6, "ORD");
        resultSet = preparedStatement.executeUpdate();
        preparedStatement.close();
    } catch (SQLException ex) {
        Logger.getLogger(DbModel.class.getName()).log(Level.SEVERE, null, ex);
    }
    add2Map(others, fname, fpath, alOthers);
}
if (flag) {
    //Add fname to map
    switch (tname[tIndex]) {
        case "audio":
            add2Map(audio, fname, fpath, alAudio);
            break;
        case "document":
            add2Map(document, fname, fpath, alDocument);
            break;
        case "executables":
            add2Map(executables, fname, fpath, alExecutables);
            break;
        case "image":
            add2Map(image, fname, fpath, alImage);
            break;
        case "video":
            add2Map(video, fname, fpath, alVideo);
            break;
        default:
            break;

        }

        //Database Insertion

        try {

            PreparedStatement preparedStatement = null;

            String query = "INSERT INTO " + tname[tIndex] + " VALUES (?,?, ?, ?,?,?);";

            preparedStatement = conection.prepareStatement(query);

            preparedStatement.setString(2, fname);

            preparedStatement.setString(3, extension);

            preparedStatement.setString(4, fpath);

            preparedStatement.setString(5, root);

            preparedStatement.setString(6, "ORD");

            resultSet = preparedStatement.executeUpdate();

            preparedStatement.close();

        } catch (SQLException ex) {

            Logger.getLogger(DbModel.class.getName()).log(Level.SEVERE, null, ex);

        }

    }

    return resultSet;

}
```

5.4 Flow Diagram

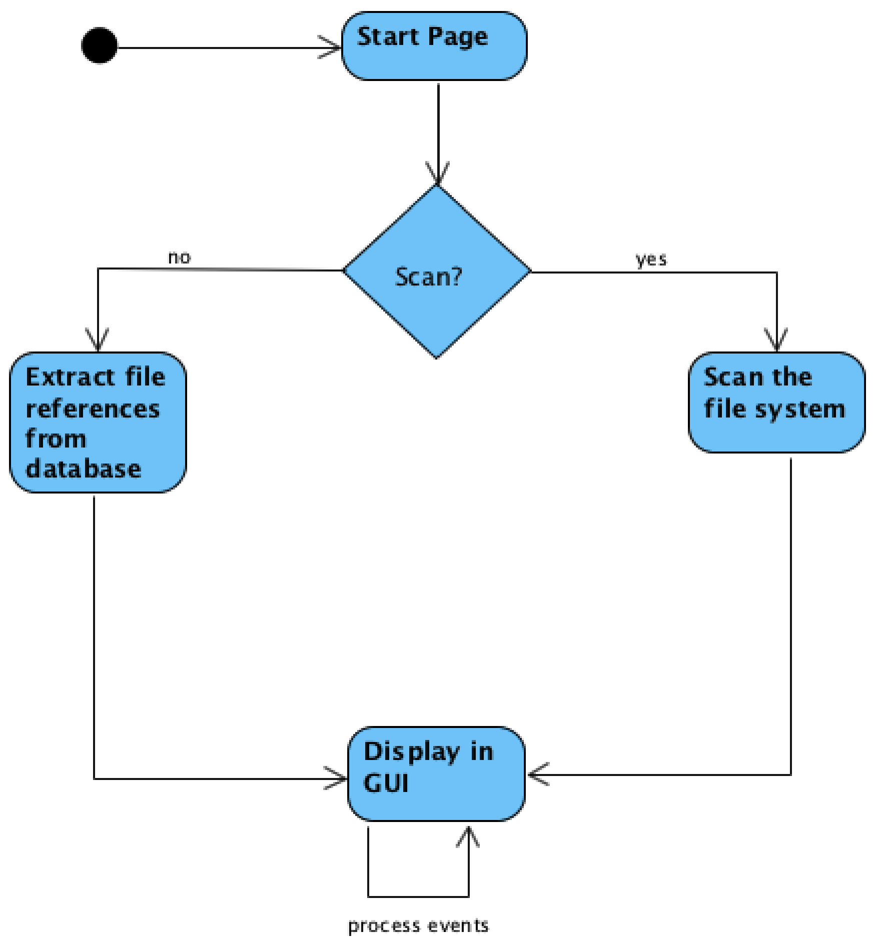


Figure. 5.1: Diagram depicting organization into categories

Chapter 6

SEARCHING

6.1 Overview

Searching which is implemented in this application is substring based. We are obtaining the filenames from the keyset of hashmaps and since we are performing substring search on memory variables it is extremely fast. The working of searching is extremely simple.

The user enter whatever he wants to search in the provided text field the code snippet performs a substring search and displays the result back to the same tableview.

6.2 Pseudocode

```
Algorithm:Searching Files displayed in list view

Input:List view Items

Output: Search results displayed in table view

Start

Step1: For all String s in tableItemsDo

Step2a: Convert s to lowercase

Step2b: Convert search textfield’s string(ts) to lowercase

Step3: If s contains ts? Goto step 4 Otherwise Goto step1

Step4: Add s to temporary list; Goto step1

Step5: Set tableview to temporary list and display

Stop
```

6.3 Code Snippet

```
public void displaySearchResults() {

         ObservableList<String> items = FXCollections.observableArrayList();

        for (String s : sPopContSearchList) {

             boolean contains = s.toLowerCase().contains(searchF.getText().toLowerCase());

             if (contains)        items.add(s);
        }

      table.setItems(getDetails(items)); //Set table columns

   }
```

6.4 Flow Diagram

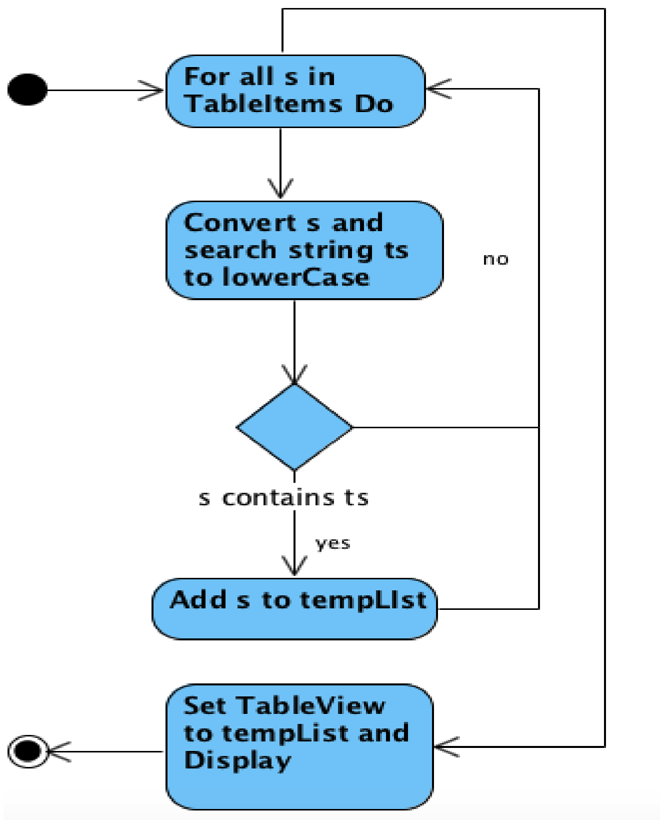

For all s in TableItems Do
Convert s and search string ts to lowerCase
no
s contains ts
yes
Add s to tempLIst
Set TableView to tempList and Display

Figure. 6.1: Diagram depicting search operation

Chapter 7

PDF RENAMING

7.1 Overview

In this module we discover vaguely named PDFs and suggest tittles to the user in a combobox. Algorithm 1 is used to discover vaguely named PDFs and algorithm 2 is used to extract and suggest titles to the user.

7.2Pseudocode

Algorithm: Displaying vaguely named PDF

Input: Document files

Output: Vaguely named PDFS

```
Start
Step1: Extract Document files from hashmaps
Step2: Filter PDF files and store in FileList
Step3: For file f in FileList Do
Step4:Split filename (f) around spaces, punctuations and numbers and store in Words (List)
Step5: For Word w in words Do
Step6: If Dictionary contains w goto Step 3 else goto Step 5
Step7: Add file f to vague FileList
Stop
```

7.3 Code Snippet

7.3.1 For Displaying Vaguely named PDFs

```
protected Boolean call(){

    vfileNames.clear();

    valPaths.clear();

    //all filenames and paths
```

```
//filenames

ObservableList<String> fileNames = FXCollections.observableArrayList();

//paths

ArrayList<String> alPaths = new ArrayList();

boolean vague = true;

Statement stmt = null;

HashSet<String> dwords = new HashSet<>();

String[] fwords;

//Step1: Extract Words From Database

Connection conn = SqliteController.Connector("jdbc:sqlite:D:wiktionary.sqlite");

String sql = "SELECT words FROM list";

ResultSet rs = null;

try {

    stmt = conn.createStatement();

    rs = stmt.executeQuery(sql);

    while (rs.next()) {

        String word = rs.getString("words").toLowerCase();

        Pattern pattern = Pattern.compile("q+|w+|e+|r+|t+|y+|u+|i+|o+|p+|a+|s+|d+"

                + "|f+|g+|h+|j+|k+|l+|z+|x+|c+|v+|b+|n+|m+|[0-9]");

        Matcher matcher = pattern.matcher(word);

        if(!word.equals("pdf") && !matcher.matches())

      dwords.add(word);

        //add interruption check
```

```
            if (isCancelled()) {
                success = false;
                return success;
            }
        }
    } catch (SQLException ex) {
                Logger.getLogger(VagueFileDiscovery.class.getName()).log(Level.SEVERE,
null, ex);
        }
        //Step2: Extract PDF Filenames And PAths
        Map<String, ArrayList<String>> map = DbModel.document;
//add to items and alPaths
for (String s : map.keySet()) {
            String extension = "";
            int i = s.lastIndexOf('.');
            if (i > 0) {
                extension = s.substring(i + 1);
  }
            //get paths
            ArrayList<String> al = map.get(s);
            if (extension.compareTo("pdf") == 0) {
                for (String path : al) {
                    fileNames.add(s);
                    alPaths.add(path);

                }

            }

            //add interruption check

            if (isCancelled()) {

                success = false;

                return success;

            }

        }

        //Step 3:Obtain Vague pdfs

for (int i = 0; i < fileNames.size(); i++) {

            fwords = fileNames.get(i).split("\\s+|(\\p{Punct})+|[0-9]+")

            for (String s : fwords) {

                if (dwords.contains(s.toLowerCase())) {

                    vague = false;

                    break;

                }

            }

            if (vague) {

                vfileNames.add(fileNames.get(i));

  valPaths.add(alPaths.get(i));

            }

            vague = true;
            //addd interrruptioon checck
            if (isCancelled()) {
               success = false;
                vfileNames.clear();
                valPaths.clear();
                return success;
            }
        }
        return success;
    }
}
//Step4:Open PDF renamind Dialog
        fileNamesLv.setItems(VagueFileDiscovery.vfileNames);
        ArrayList<String> vPaths = VagueFileDiscovery.valPaths;
        fileNamesLv.setOnMouseClicked((MouseEvent event) -> {
            if (event.getClickCount() > 1) {



                int lvIndex = fileNamesLv.getSelectionModel().getSelectedIndex();
                try {
                    Stage renameDialog = new Stage();
                    renameDialog.initModality(Modality.APPLICATION_MODAL);
                    VBox      renameDialogRoot      =
                       FXMLLoader.load(getClass().getResource("renameDialog.fxml"));

                    renameDialog.setScene(new Scene(renameDialogRoot));

                    renameDialog.setResizable(false);

                    //Set Path to label

                    Label lb = (Label) renameDialogRoot.getChildren().get(1);

                    lb.setText(vPaths.get(lvIndex));

                    //set Label used for displaying unsuccessful message

                    Label msgLabel = (Label) renameDialogRoot.getChildren().get(4);

                    //retrieve combobox

                ComboBox<String>coB=(ComboBox<String>)renameDialogRoot.getChildren()
.get(2);

                    coB.setOnAction(e -> {

                       newFName = coB.getSelectionModel().getSelectedItem();

                    });

                    //setting dummy items

                    coB.getItems().addAll(PDfTitleExtract.ExtractTitle(vPaths.get(lvIndex)));

                     HBox hb = (HBox) renameDialogRoot.getChildren().get(3);

                    Button okB = (Button) hb.getChildren().get(0);

                    Button cancelB = (Button) hb.getChildren().get(1);

                    cancelB.setOnAction(e -> renameDialog.close());

                    okB.setOnAction(e -> {

            //Get Name from combobox

                       //pattern for files starting with whitespaces
```

```
                Pattern pattern = Pattern.compile("(^\\s+)(.*)");

                Matcher matcher = pattern.matcher(newFName);

                  boolean check = newFName == null || newFName.compareTo("") == 0 || matcher.matches()

                                  || newFName.contains("/") || newFName.contains("\\") || newFName.contains(":")

                                  || newFName.contains("*") || newFName.contains("?") || newFName.contains("\"")

                                  || newFName.contains("<") || newFName.contains(">") || newFName.contains("|");

                if (check) {

                   //User enters nothing

                   msgLabel.setText("Please Give A Valid Name");

                } else {

                   //Renaming from user input

                   //File oldName = new File(vPaths.get(lvIndex));

Path oldPath = Paths.get(vPaths.get(lvIndex));

                   Path newPath = oldPath.resolveSibling(newFName + ".pdf");
```

7.3.2 For extracting Title

```
public static ArrayList<String> ExtractTitle(String path) {

     count = 0;

     charInputs = new ArrayList<>();

     PdfReader reader = null;

     try {
         reader = new PdfReader(path);

         CharSizeExtraction charSizeExtObj = new CharSizeExtraction();

                     PdfTextExtractor.getTextFromPage(reader, 1, (TextExtractionStrategy)
charSizeExtObj);

     } catch (IOException ex) {

             Logger.getLogger(PDfTitleExtract.class.getName()).log(Level.SEVERE, null,
ex);

     }

     reader.close();

     Comparator<Integer> descOrd

             = (Integer e1, Integer e2) -> e2.compareTo(e1);

     //set of fonts (among extracted characters) in desc order

     SortedSet<Integer> setOfSizes = new TreeSet<>(descOrd);

     for (CharInput c : charInputs) {

         setOfSizes.add(c.fSize);

     }


     StringBuilder sb = new StringBuilder();

     // size ->[arraylist of words belonging to that size]

     TreeMap<Integer, ArrayList<String>> map = new TreeMap<>(descOrd);

     int tcount = 0;

     boolean endflag = false;

     ArrayList<String> tAl = null;

     CharInput c;
```

```
int m;

//Obtain Map

for (Integer tempSize : setOfSizes) {

    //for every size in setOfSizes loop through all characters

    for (m = 0, endflag = false; m < charInputs.size(); m++) {

        c = charInputs.get(m);

        if (c.fSize.intValue() == tempSize.intValue()) {

            //characters that belong to tempsize

            if (tcount < 100) {

                //no of character sequence appended to sb restriced to 100 characters

                sb.append(c.character);

                tcount++;

            }

            endflag = true;

        } else {

            //if a sequence (of characters that belong to tempsize) ends

            if (endflag) {

                if (map.containsKey(tempSize)) {

                    tAl = map.get(tempSize);

                    tAl.add(sb.toString());

                    map.put(tempSize, tAl);

                } else {

                    tAl = new ArrayList<>();
                    tAl.add(sb.toString());

                    map.put(tempSize, tAl);

                }

                endflag = false;

                //empty sb to make room for new sequence of characters

                sb.delete(0, sb.length());

                tcount = 0;

            }

        }

    }

}

int ctitles = 0;

final int maxSuggestion = 6;

//title that map to particular size(temporary variable)

ArrayList<String> tempTitles = new ArrayList();


//arraylist of titles based on maxSuggestion

ArrayList<String> titles = new ArrayList();

//Get maxSuggestion titles from keyset

for (int k : map.keySet()) {

    tempTitles = map.get(k);

    titles.addAll(tempTitles);

    ctitles += tempTitles.size();       //ctitles is essentially titles.size()
```

```
            if (ctitles > maxSuggestion) {

                break;

            }

        }

        //Array of titles

        ArrayList<String> cutTitles = new ArrayList<>();

        for (int i = 0; i < Math.min(maxSuggestion, titles.size()); i++) {

            cutTitles.add(titles.get(i));

        }

 return cutTitles;

    }
```

7.4 Flow Diagram

7.4.1 Flow diagram for displaying vaguely named PDF

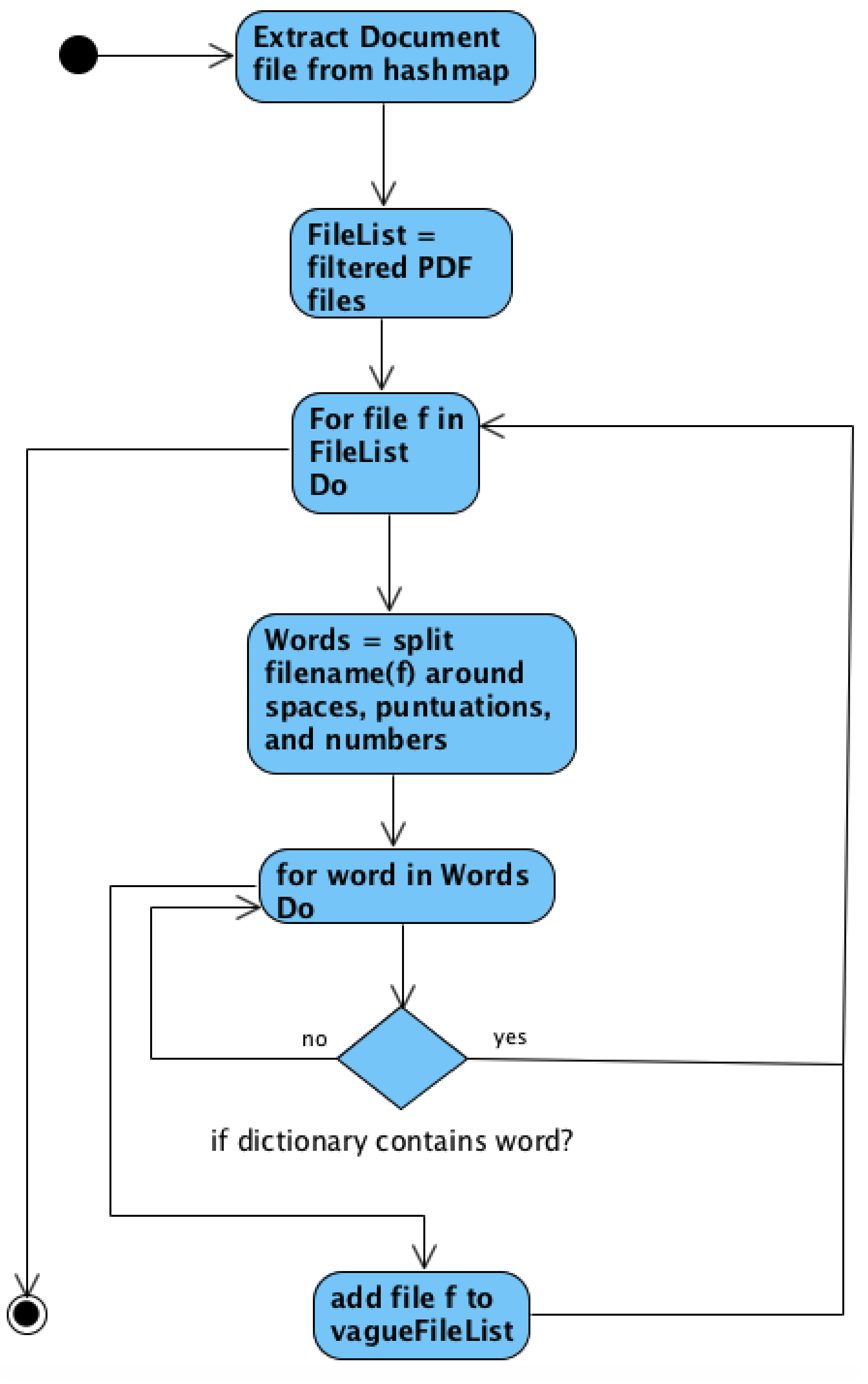

Extract Document file from hashmap
FileList = filtered PDF files
For file f in FileList Do
Words = split filename(f) around spaces, puntuations, and numbers
for word in Words Do
no
yes
if dictionary contains word?
add file f to vagueFileList

Figure. 7.1: Diagram depicting vaguely named PDF

7.4.2 Flow diagram for Title Extraction

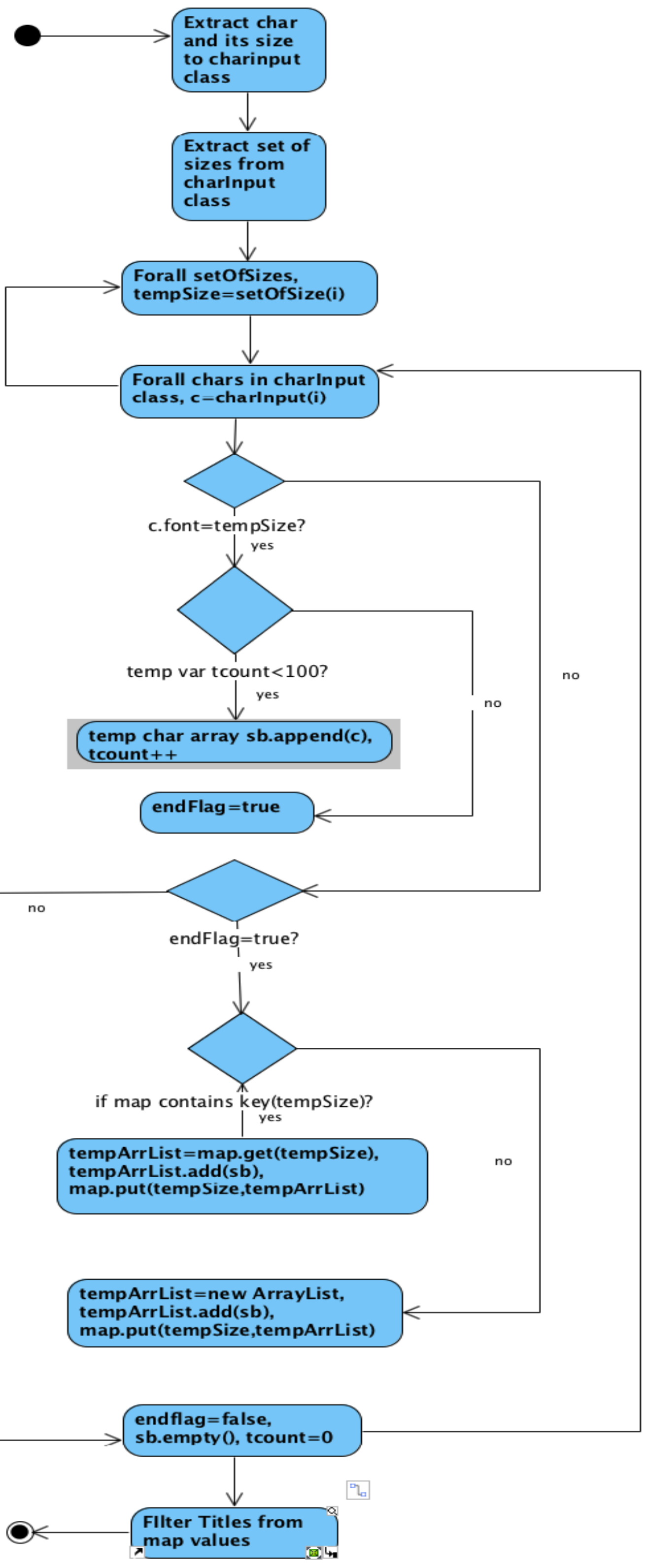

Extract char and its size to charinput class
Extract set of sizes from charInput class
Forall setOfSizes, tempSize=setOfSize(i)
Forall chars in charInput class, c=charInput(i)
c.font=tempSize?
yes
temp var tcount<100?
yes
temp char array sb.append(c), tcount++
no
no
endFlag=true
no
endFlag=true?
yes
if map contains key(tempSize)?
yes
tempArrList=map.get(tempSize), tempArrList.add(sb), map.put(tempSize,tempArrList)
no
tempArrList=new ArrayList, tempArrList.add(sb), map.put(tempSize,tempArrList)
endflag=false, sb.empty(), tcount=0
FIlter Titles from map values

Figure. 7.2: Diagram depicting Title Extraction

Chapter 8

TAGGING

8.1 Overview

In this module we have defined two tags namely Starred and Backup. In the Backup tag a user can add the filenames which he/she would like to backup and in the future, the user can refer to this tag while taking the system backup. Similarly, a user can also add important files to the starred tags. Both the tags makes use of tableview to display the list of files.

8.2 Pseudocode

```
Algorithm 1: Tagging of files
Input: User Click (Left Mouse Click)
Output: Tagged Files
Start
Step1: Find the file that has been clicked
Step2: Find the row entry for the clicked File in the database
Step3: Update the TAG column for the row found fin Step2
Stop
```

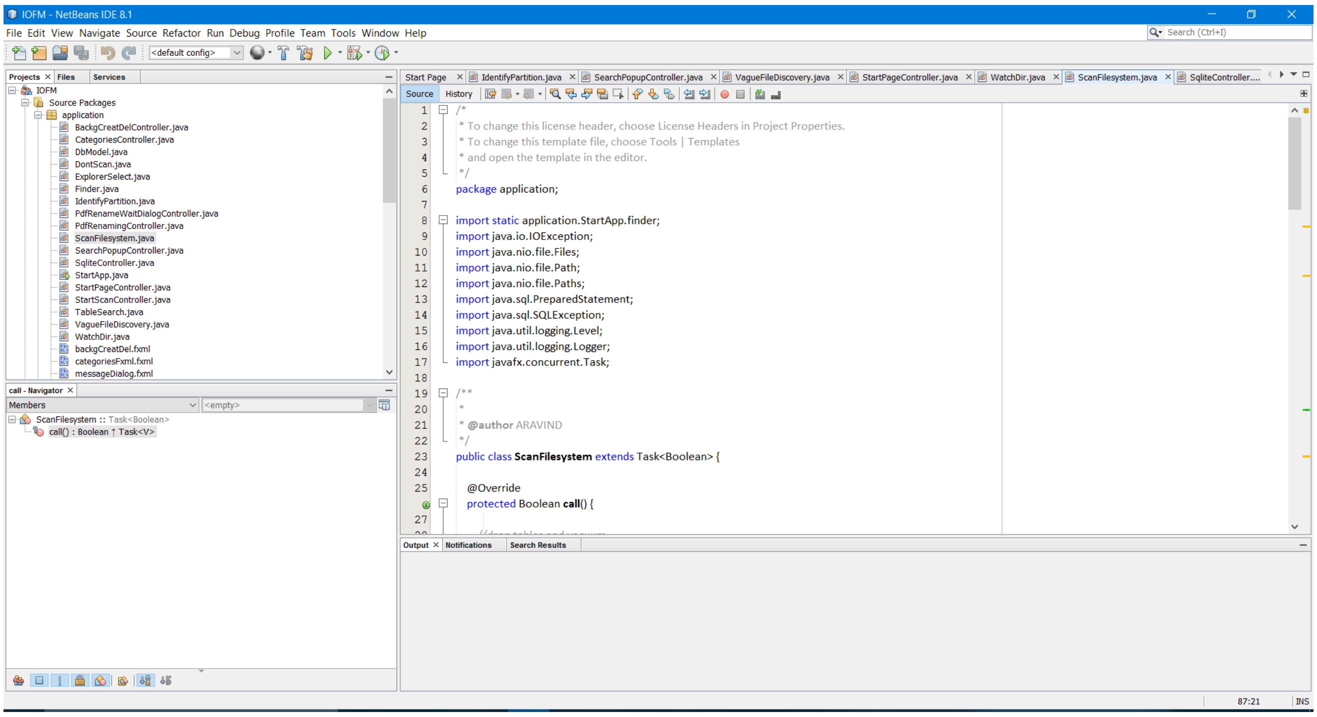
8.3 Code Snippet

```
// FOR TAGS
    Menu tagMenu = new Menu("TAGS");
    MenuItem starMenu = new MenuItem("STARRED");
    MenuItem backupMenu = new MenuItem("Backup");
```

```
        tagMenu.getItems().addAll(starMenu, backupMenu);

        starMenu.setOnAction(e -> {

            try {

                PreparedStatement ps = StartApp.demodbModel.conection.prepareStatement(

                        "UPDATE " + tableName + " SET TAG = ? WHERE PATH = ?");

                // set the preparedstatement parameters

                ps.setString(1, "STAR");

                ps.setString(2, al.get(0));

                // call executeUpdate to execute our sql update statement

                ps.executeUpdate();

                ps.close();

            } catch (SQLException ex) {

                Logger.getLogger(CategoriesController.class.getName()).log(Level.SEVERE,
null, ex);

            }

        });

        backupMenu.setOnAction(e -> {

            try {

                PreparedStatement ps = StartApp.demodbModel.conection.prepareStatement(

                        "UPDATE " + tableName + " SET TAG = ? WHERE PATH = ?");

                // set the preparedstatement parameters

                ps.setString(1, "BACKUP");

                ps.setString(2, al.get(0));
                // call executeUpdate to execute our sql update statement

                ps.executeUpdate();

                ps.close();

            } catch (SQLException ex) {

                Logger.getLogger(CategoriesController.class.getName()).log(Level.SEVERE,
null, ex);

            }

        });
```

CHAPTER 9

SNAPSHOTS


IOFM - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
Start Page IdentifyPartition.java SearchPopupController.java VagueFileDiscovery.java StartPageController.java WatchDir.java ScanFilesystem.java SqliteController....
Source History
Output Notifications Search Results
87:21 INS

```
IOFM
  Source Packages
    application
      BackgCreatDelController.java
      CategoriesController.java
      DbModel.java
      DontScan.java
      ExplorerSelect.java
      Finder.java
      IdentifyPartition.java
      PdfRenameWaitDialogController.java
      PdfRenamingController.java
      ScanFilesystem.java
      SearchPopupController.java
      SqliteController.java
      StartApp.java
      StartPageController.java
      StartScanController.java
      TableSearch.java
      VagueFileDiscovery.java
      WatchDir.java
      backgCreatDel.fxml
      categoriesFxml.fxml
      messageDialog.fxml
```

```
ScanFilesystem :: Task<Boolean>
  call() : Boolean ↑ Task<V>
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package application;

import static application.StartApp.finder;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.concurrent.Task;

/**
 *
 * @author ARAVIND
 */
public class ScanFilesystem extends Task<Boolean> {

    @Override
    protected Boolean call() {
```

Figure 9.1: NetBeans Execution Environment

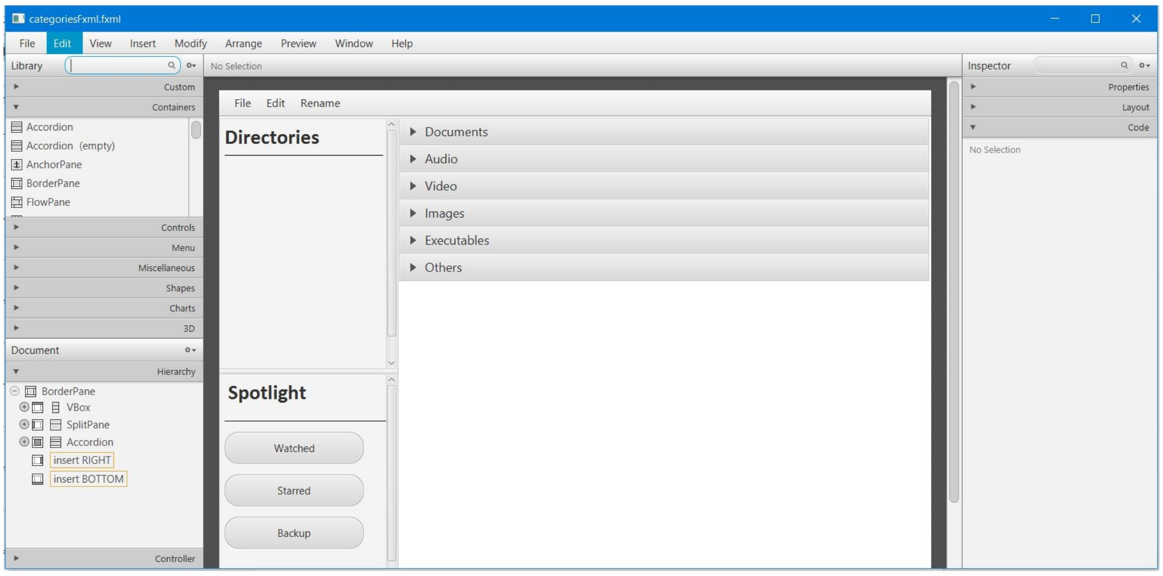


Figure 9.2: JavaFX Scene Builder

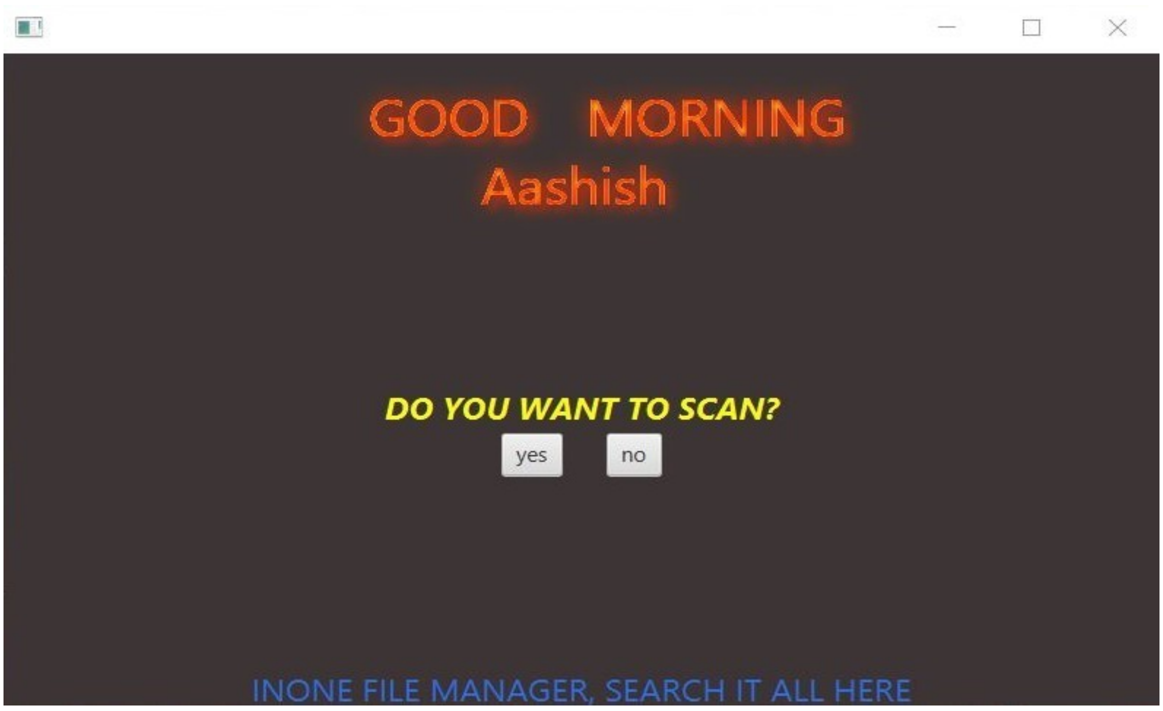


Figure 9.3: Welcome Screen

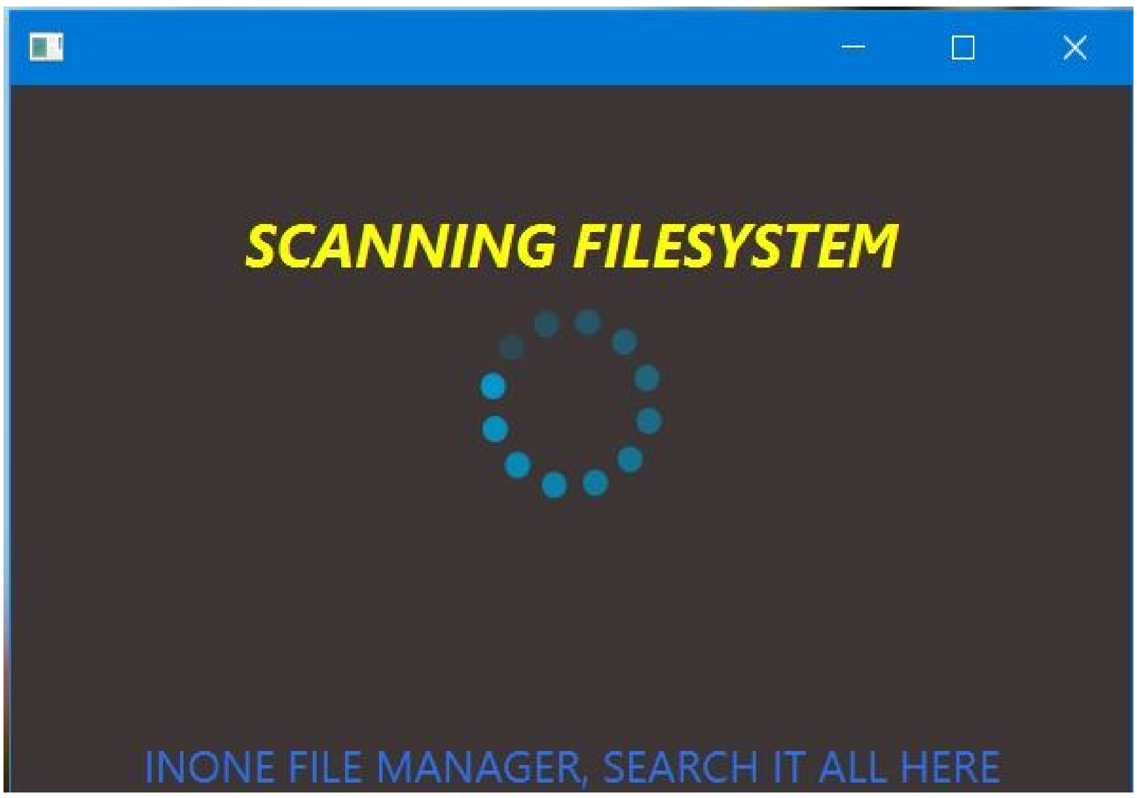


Figure 9.4: Scanning Filesystem

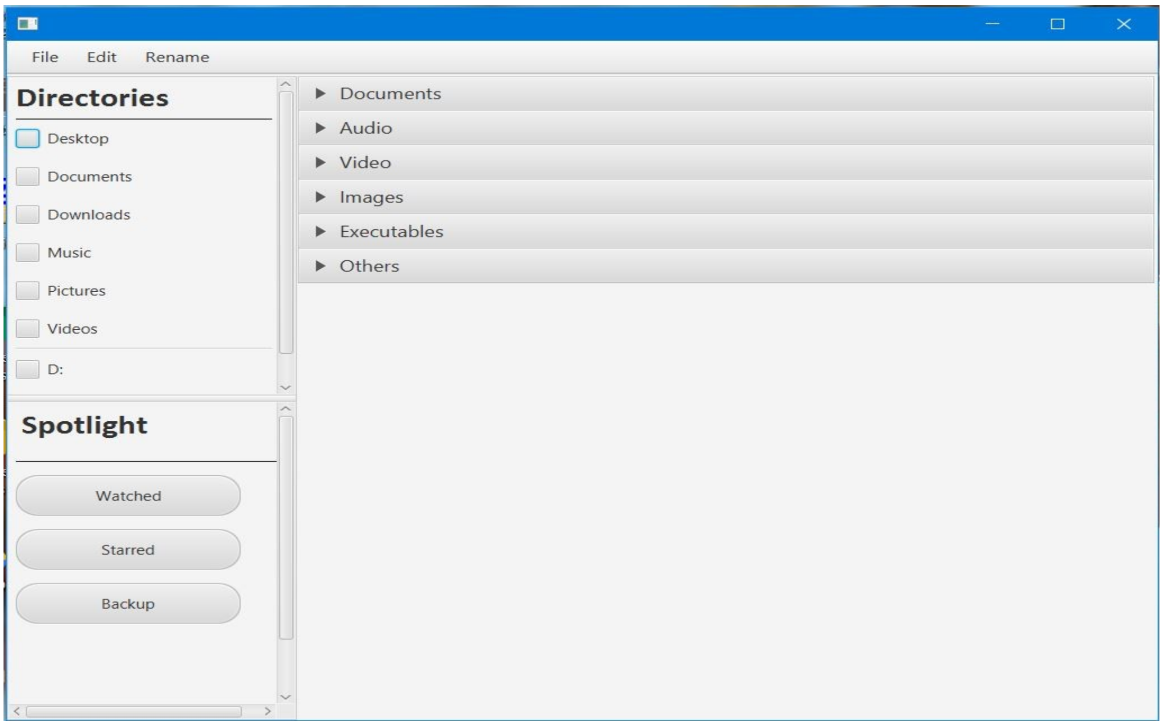
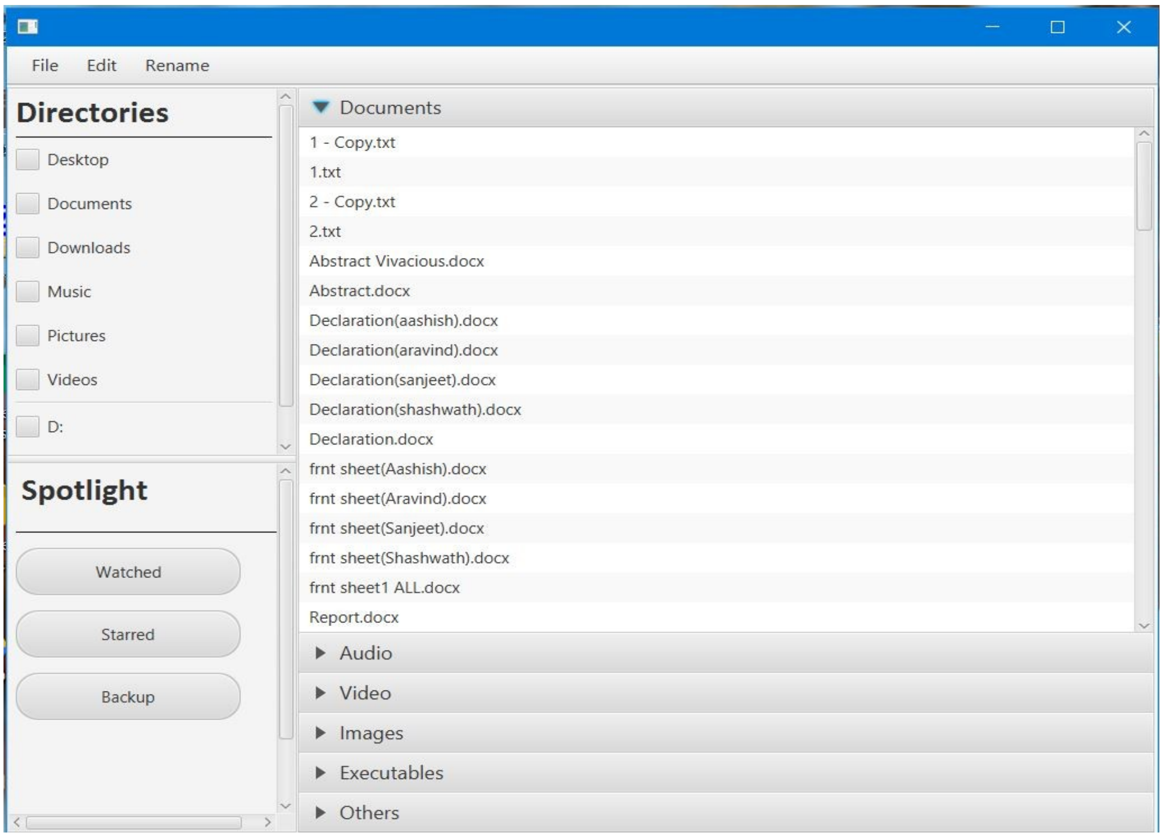
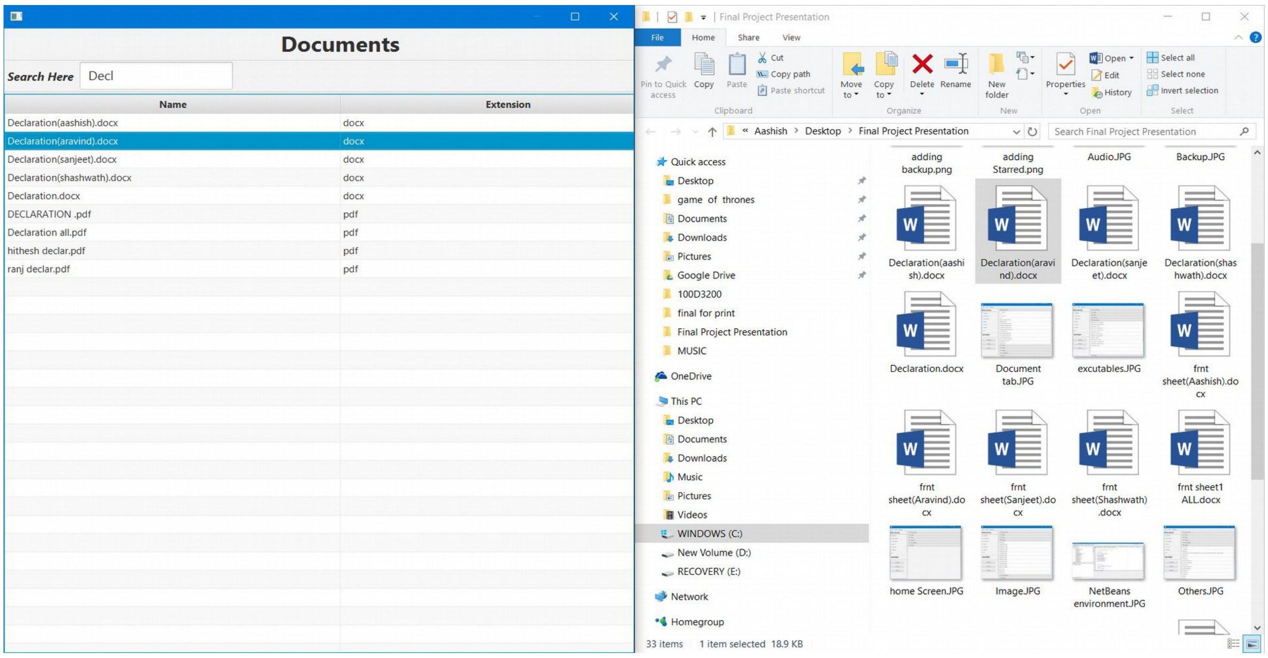


Figure 9.5: Home Screen



1 - Copy.txt
1.txt
2 - Copy.txt
2.txt
Abstract Vivacious.docx
Abstract.docx
Declaration(aashish).docx
Declaration(aravind).docx
Declaration(sanjeet).docx
Declaration(shashwath).docx
Declaration.docx
frnt sheet(Aashish).docx
frnt sheet(Aravind).docx
frnt sheet(Sanjeet).docx
frnt sheet(Shashwath).docx
frnt sheet1 ALL.docx
Report.docx

Figure 9.6: Home Screen with Document Tab

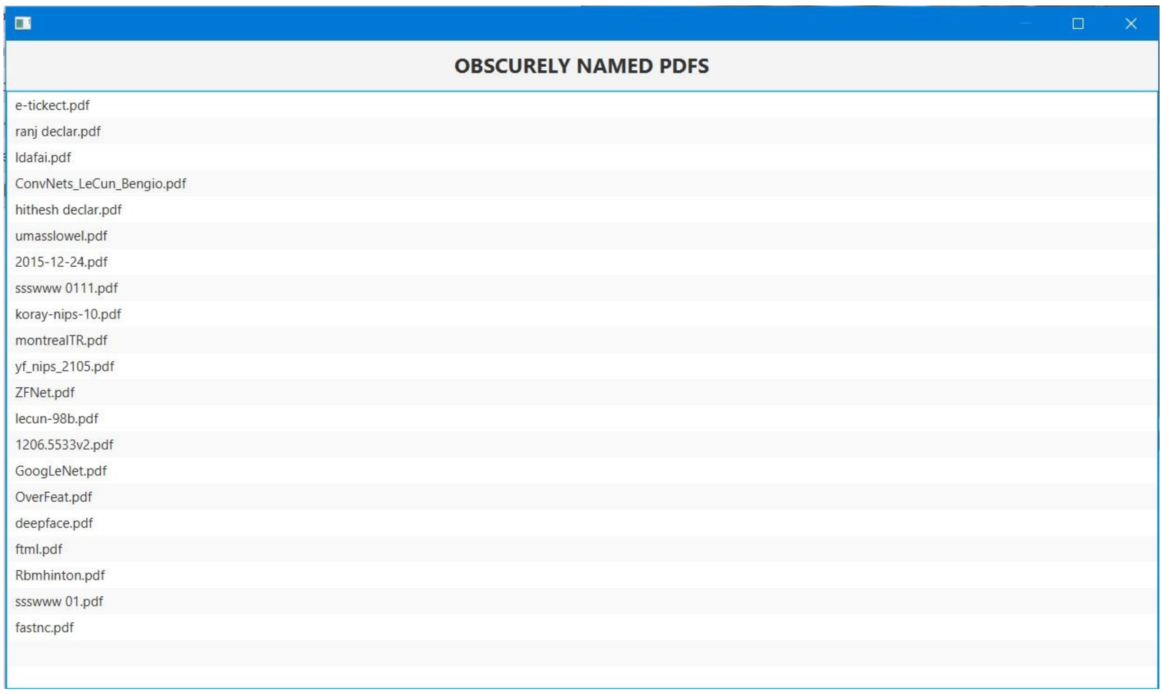


Documents

Search Here: Decl

Name	Extension
Declaration(aashish).docx	docx
Declaration(aravind).docx	docx
Declaration(sanjeet).docx	docx
Declaration(shashwath).docx	docx
Declaration.docx	docx
DECLARATION .pdf	pdf
Declaration all.pdf	pdf
hithesh declar.pdf	pdf
ranj declar.pdf	pdf

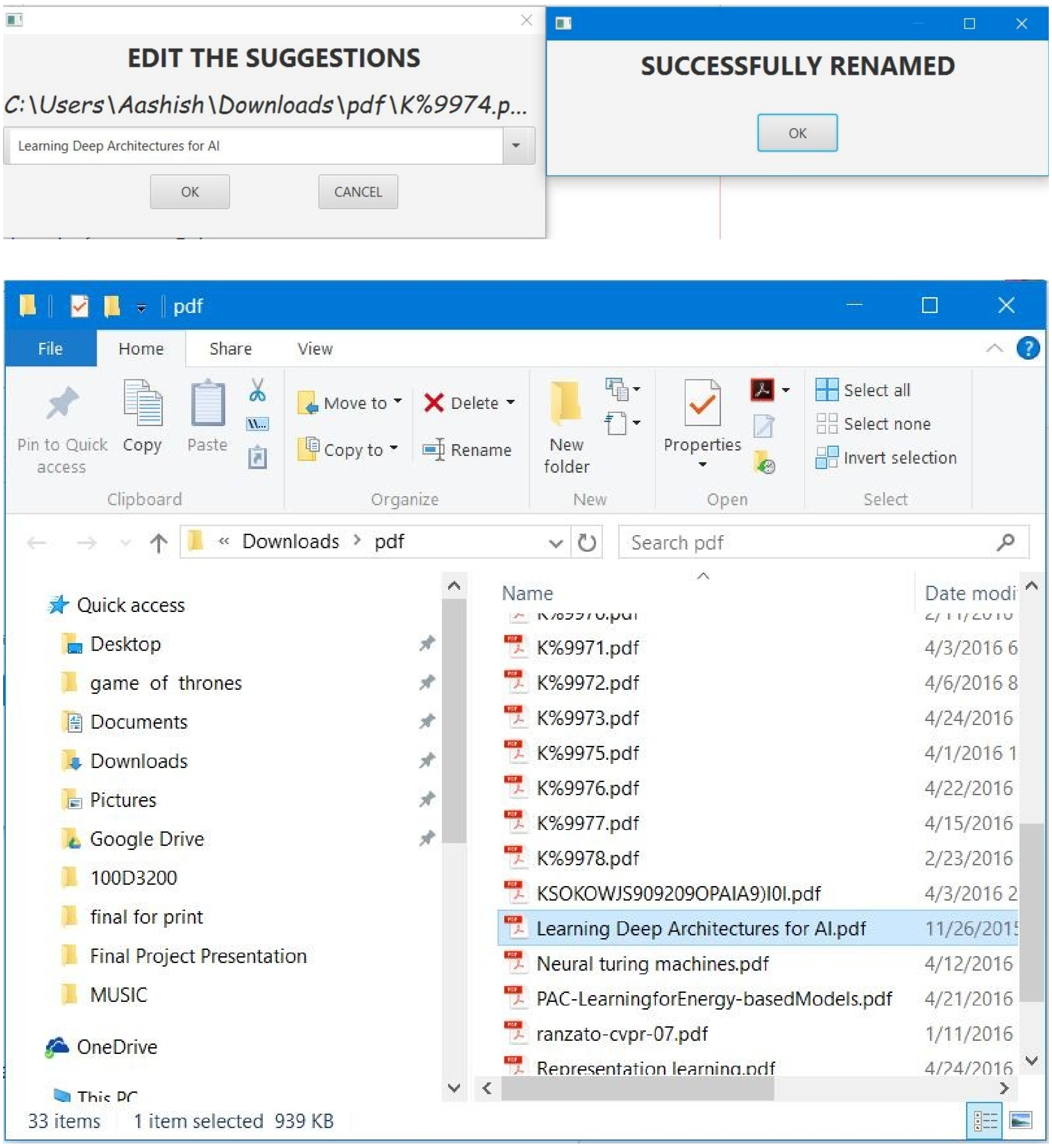
Figure 9.7: Search within categories



OBSCURELY NAMED PDFS

e-tickect.pdf
ranj declar.pdf
ldafai.pdf
ConvNets_LeCun_Bengio.pdf
hithesh declar.pdf
umasslowel.pdf
2015-12-24.pdf
ssswww 0111.pdf
koray-nips-10.pdf
montrealTR.pdf
yf_nips_2105.pdf
ZFNet.pdf
lecun-98b.pdf
1206.5533v2.pdf
GoogLeNet.pdf
OverFeat.pdf
deepface.pdf
ftml.pdf
Rbmhinton.pdf
ssswww 01.pdf
fastnc.pdf

Figure 9.8: PDF renaming window



EDIT THE SUGGESTIONS

C:\Users\Aashish\Downloads\pdf\K%9974.p...

Learning Deep Architectures for AI

SUCCESSFULLY RENAMED

Downloads › pdf

Name	Date modi
K%9971.pdf	4/3/2016 6
K%9972.pdf	4/6/2016 8
K%9973.pdf	4/24/2016
K%9975.pdf	4/1/2016 1
K%9976.pdf	4/22/2016
K%9977.pdf	4/15/2016
K%9978.pdf	2/23/2016
KSOKOWJS909209OPAIA9)I0I.pdf	4/3/2016 2
Learning Deep Architectures for AI.pdf	11/26/2015
Neural turing machines.pdf	4/12/2016
PAC-LearningforEnergy-basedModels.pdf	4/21/2016
ranzato-cvpr-07.pdf	1/11/2016
Representation learning.pdf	4/24/2016

33 items 1 item selected 939 KB

Figure 9.9: Successful renaming of PDF

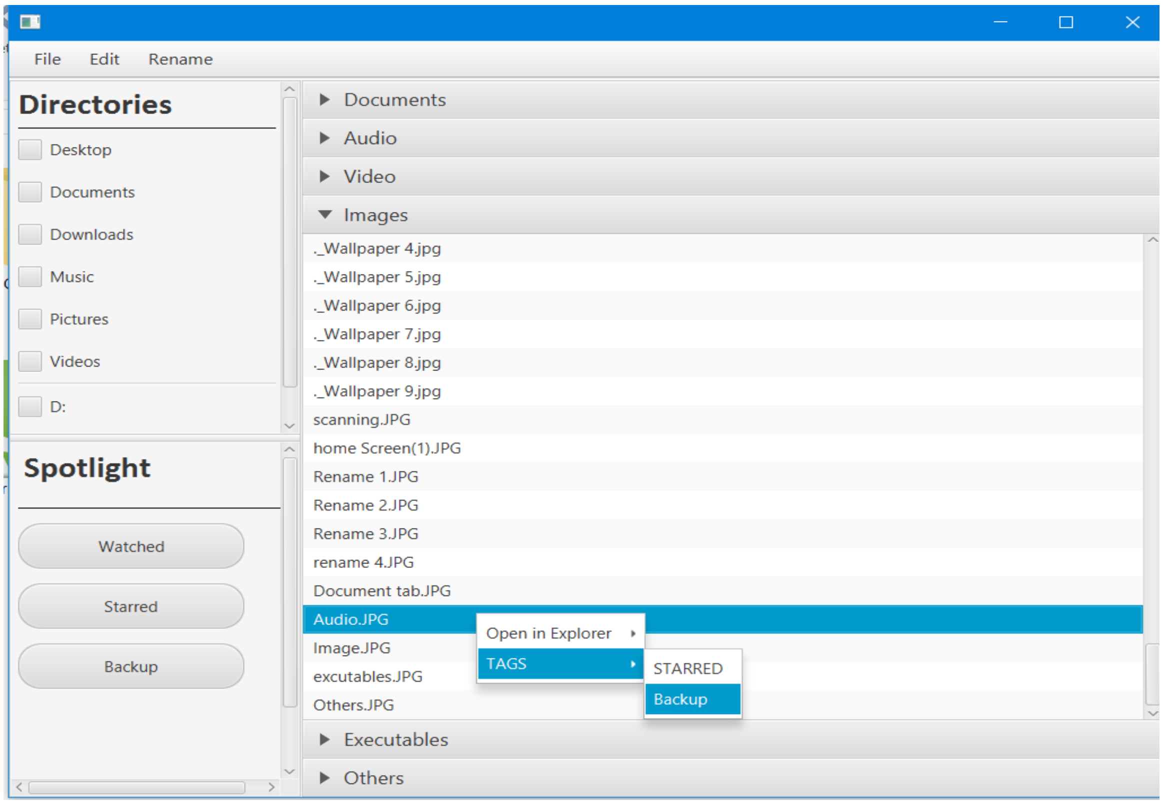
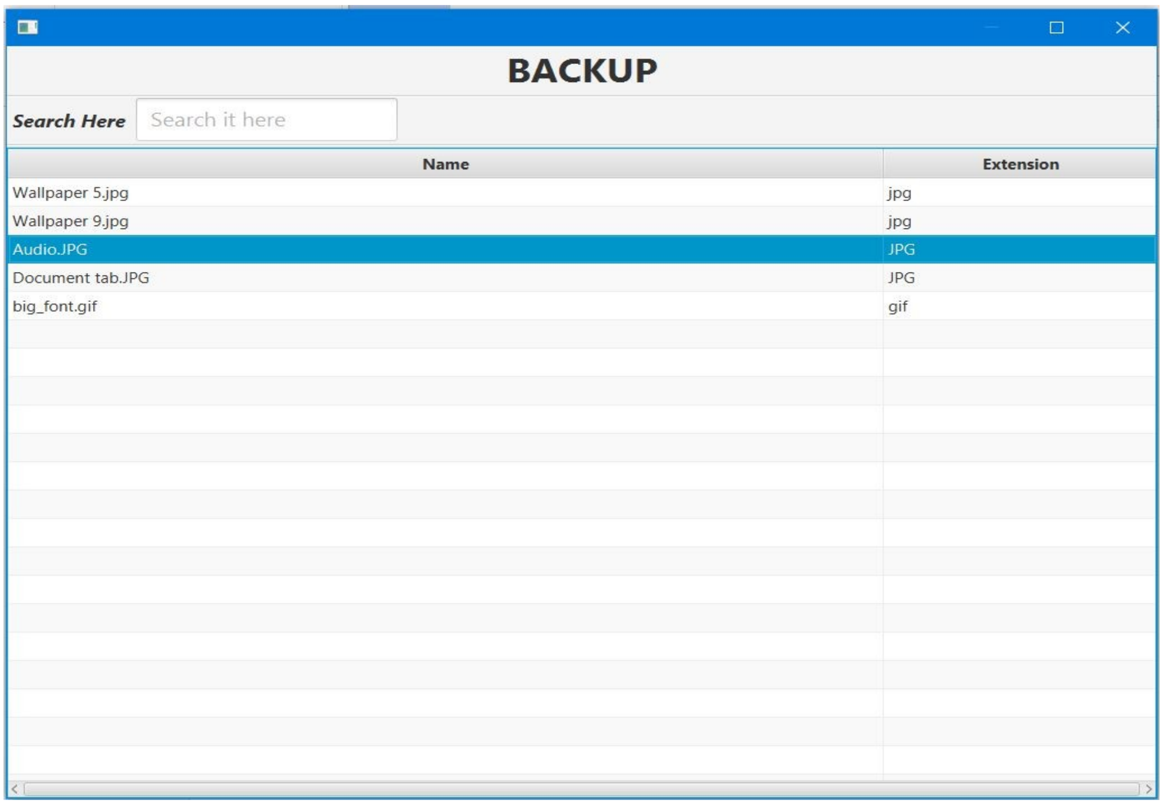


Figure 9.10: Adding to the Backup Tags

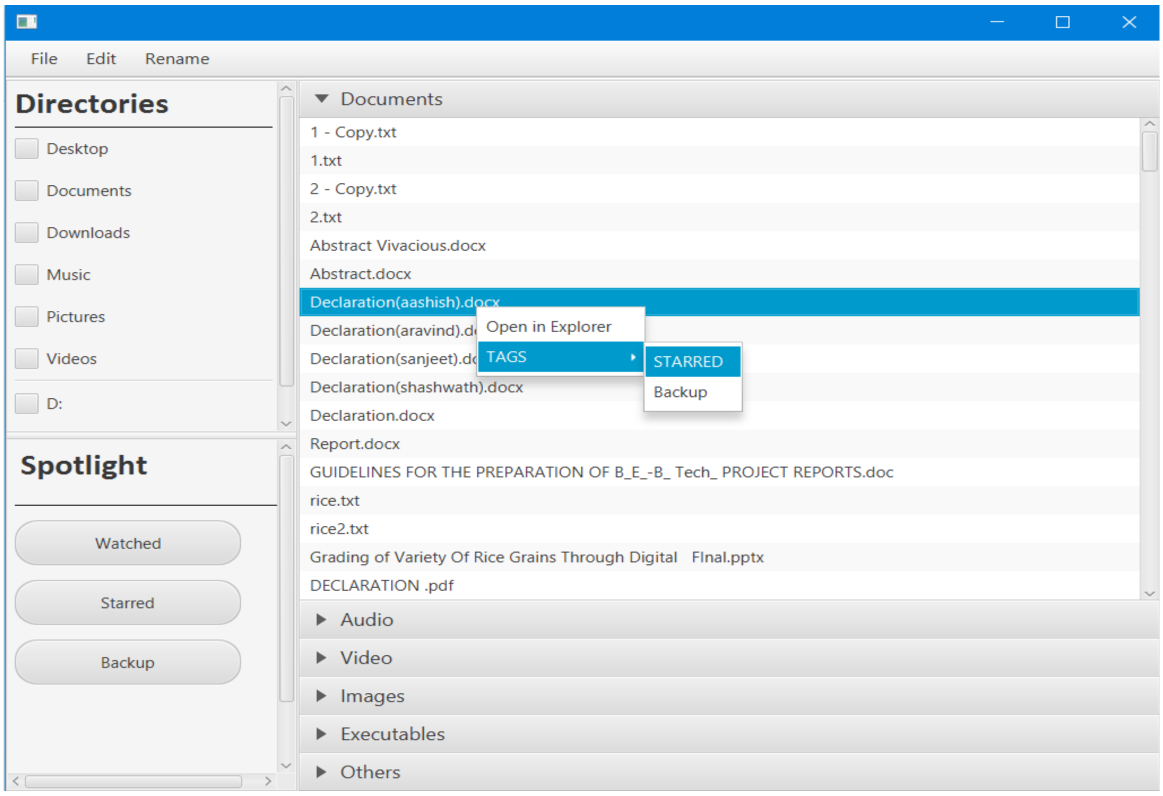


BACKUP

Search Here [Search it here]

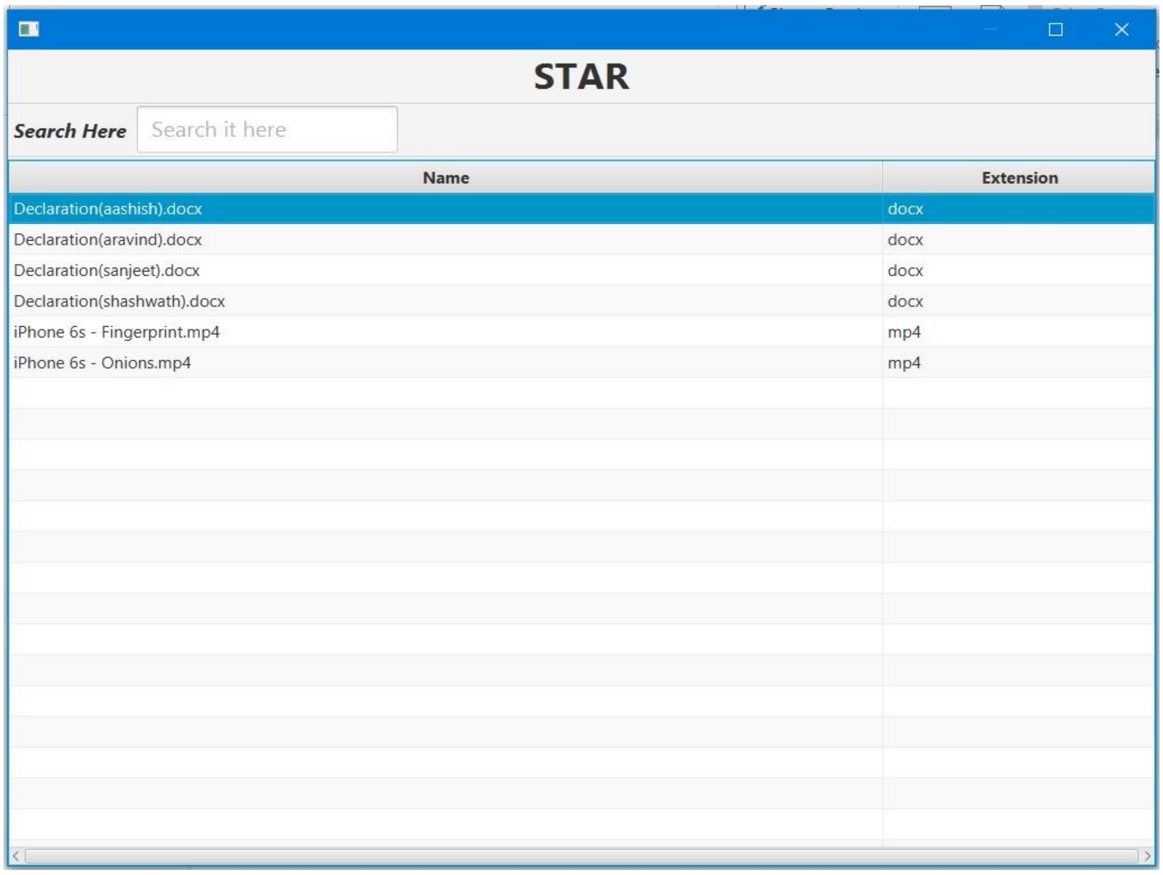
Name	Extension
Wallpaper 5.jpg	jpg
Wallpaper 9.jpg	jpg
Audio.JPG	JPG
Document tab.JPG	JPG
big_font.gif	gif

Figure 9.11: Backup Window



1 - Copy.txt
1.txt
2 - Copy.txt
2.txt
Abstract Vivacious.docx
Abstract.docx
Declaration(aashish).docx
Declaration(aravind).d
Declaration(sanjeet).d
Declaration(shashwath).docx
Declaration.docx
Report.docx
GUIDELINES FOR THE PREPARATION OF B_E_-B_ Tech_ PROJECT REPORTS.doc
rice.txt
rice2.txt
Grading of Variety Of Rice Grains Through Digital FInal.pptx
DECLARATION .pdf

Figure 9.12: Adding to the Starred Tags



Name	Extension
Declaration(aashish).docx	docx
Declaration(aravind).docx	docx
Declaration(sanjeet).docx	docx
Declaration(shashwath).docx	docx
iPhone 6s - Fingerprint.mp4	mp4
iPhone 6s - Onions.mp4	mp4

Figure 9.13: Starred Window

Chapter 10

CONCLUSION

To conclude the whole point of view, we have designed an application that presents the file in organized, categorized & appealing manner. A user need not put extra effort to do this (which many OS provides through their native file explorer). Searching of files is very fast because we are storing file and their attributes in database. It is done based on substring search of filenames stored in memory. A few additional features include listing of watched video files, duplicate files and manual tagging of files (important and backup). Listing of vaguely named pdfs and title suggestion of these pdfs is significant feature that we have included. Thus this project automates the trivial manual tasks that the user can do using native GUI file explorer provided by various OS.