



# Reading Journal App

Jason Setyadi



Problem & Solution

...



Requirements

...



Design

...



Implementation

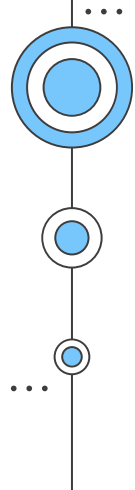
...



Testing

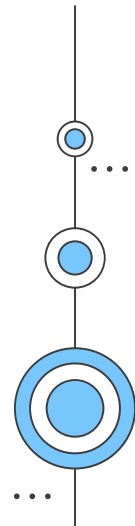
...

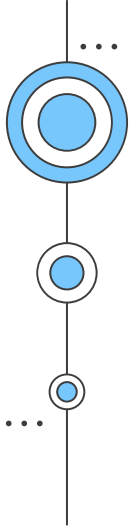
# Table of Contents



# 01

## Problem Statement



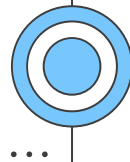
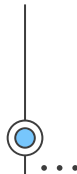



# The Problem.

We read so much on a day to day basis (emails, books, articles...), yet, we can only retain so little of it.

**So how do we keep track of what we read?**

...

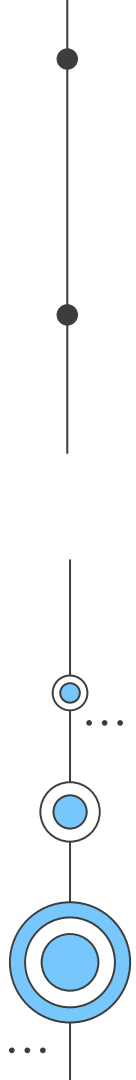




There are a lot of book tracking and note-taking/journaling apps, but there are not many that integrates both features.

## Proposed Solution:

An app where users can journal thoughts and summarise their selected reading material.



...



02

Requirements



# Requirements Definition

## Functional

Upload PDF files into the app or select a book using the **International Standard Book Number (ISBN)**

View the pdf uploaded by the user

View the book details (author, title, cover) based on ISBN

Create, Read, Update, Delete (**CRUD**) journals for the selected reading materials

CRUD sections for each journal that corresponds to the topics/chapters of the reading material

CRUD folders to store the journals

Login and register an account

## Non-Functional

Easy to navigate and intuitive to use

Data should only be viewable by authorised users

Reliable – no bugs that interfere with the use of the app



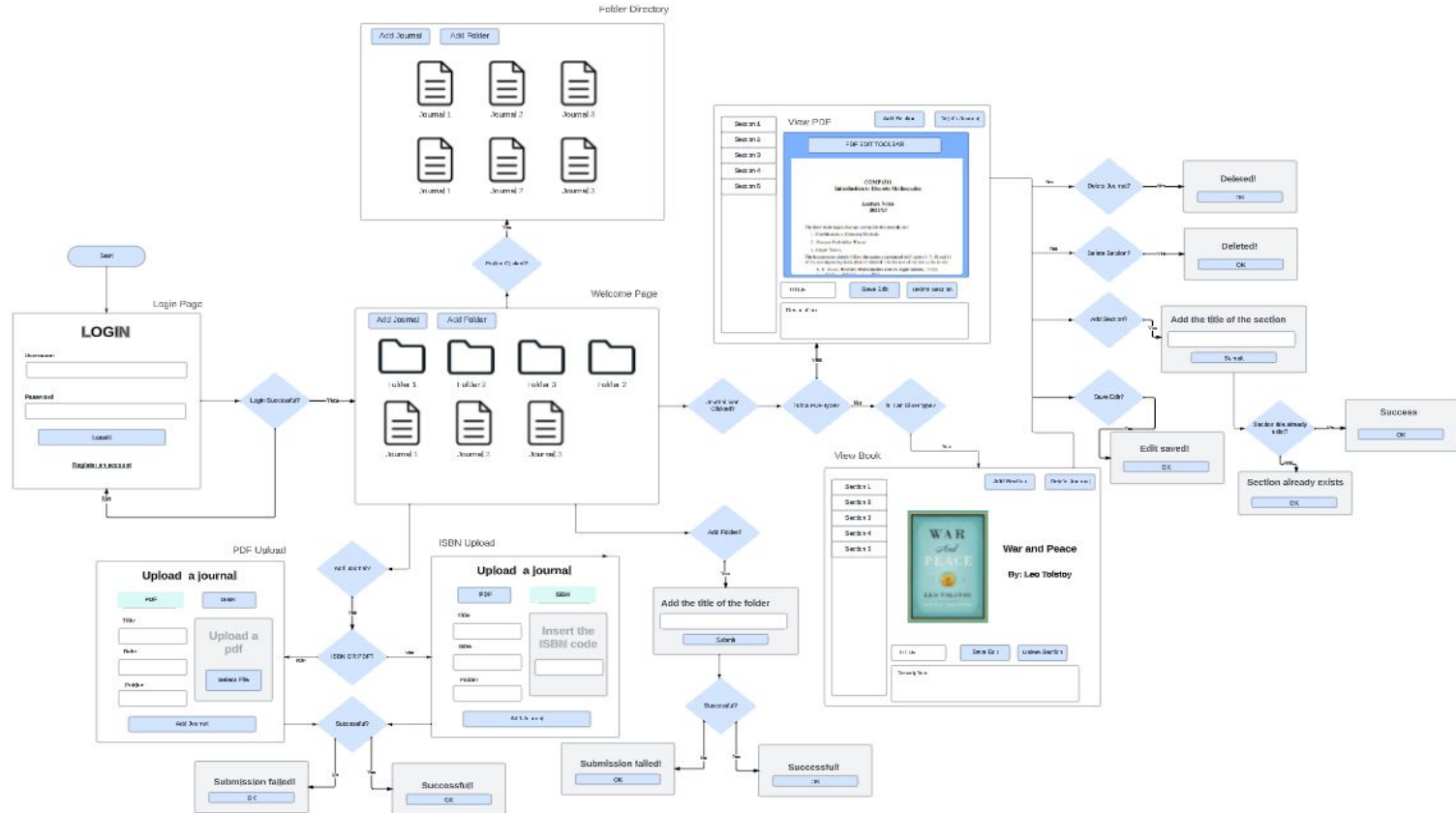
03

Design

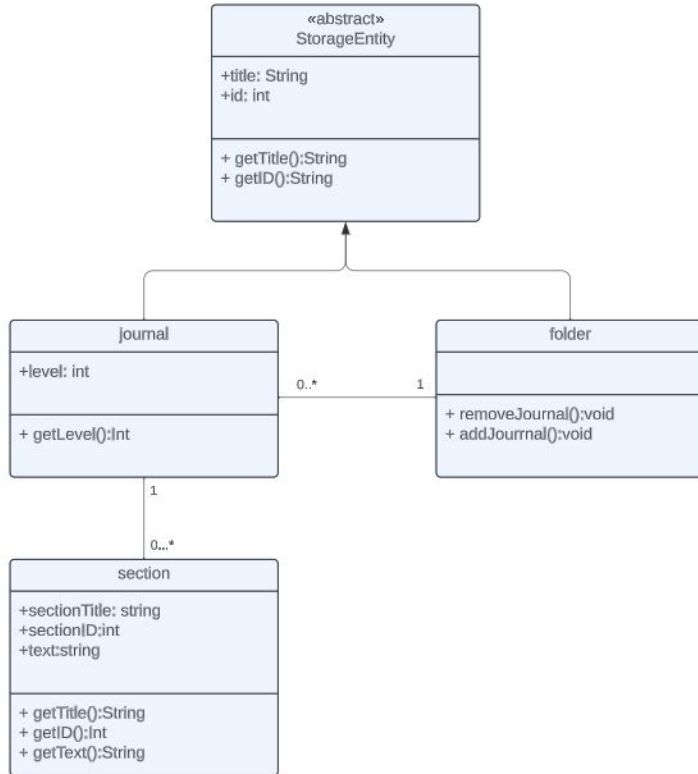




# User Interface and Flow Diagram...



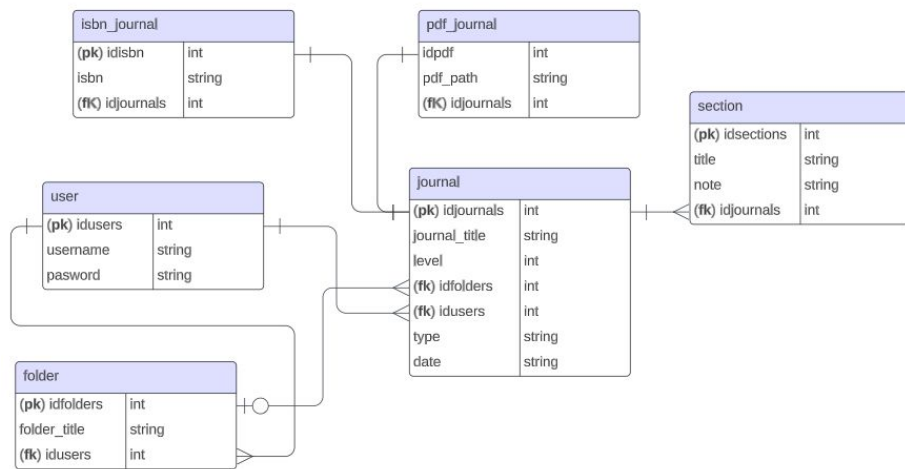
# UML Diagram



Represents the core entities within the journaling app and their interactions.

Does not represent all the classes in the final design.

# Entity Relationship Diagram

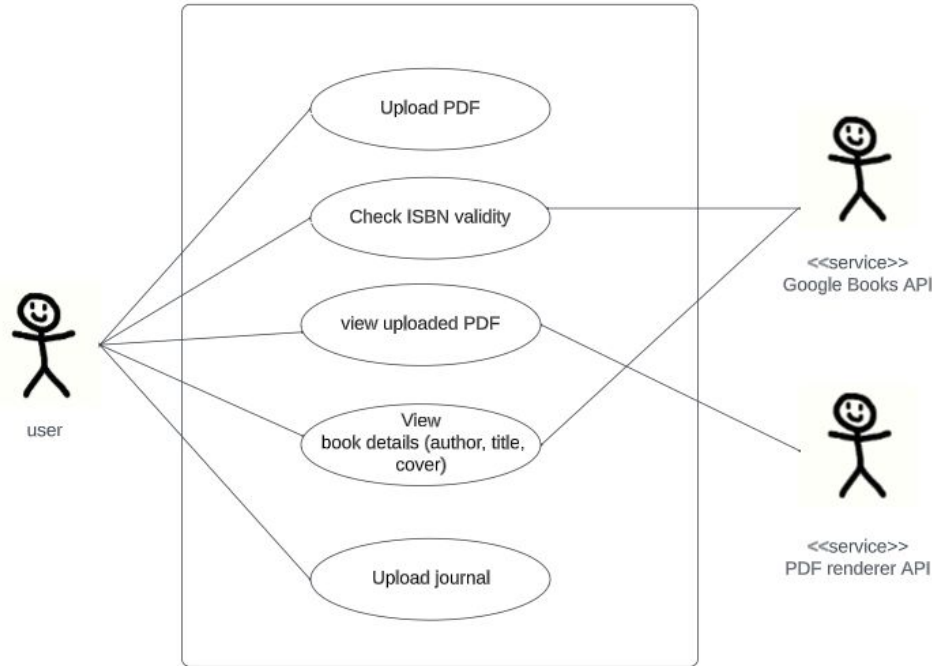


Detailing the relationships between the tables in the database.

- One user can have many folders and journals
- One journal can have many sections
- Journal can include isbn or pdf (but a journal must not be both at the same time)

# Use Case Diagram

Journaling App System



The journaling app interacts with some "actors" outside of system.

- Google books API - give the app access to the Google Books repository
- PDF renderer - allow users to view the pdf they uploaded within the app

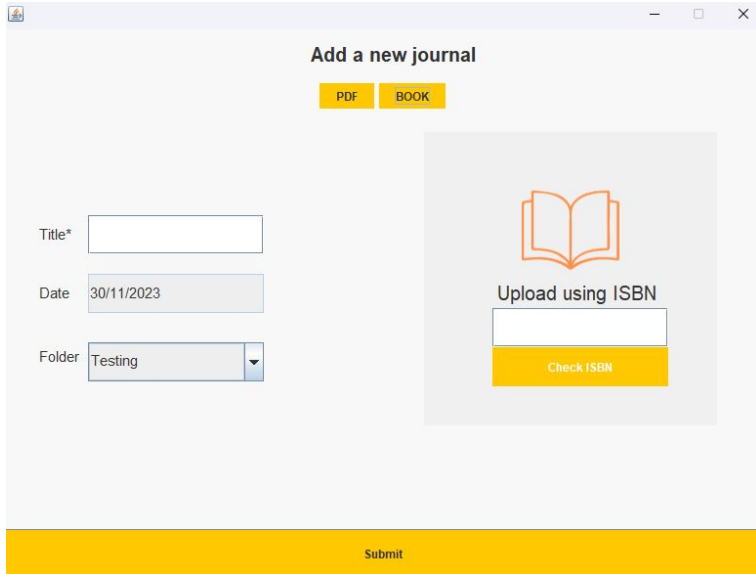


04

Implementation!



# Uploading Reading Materials

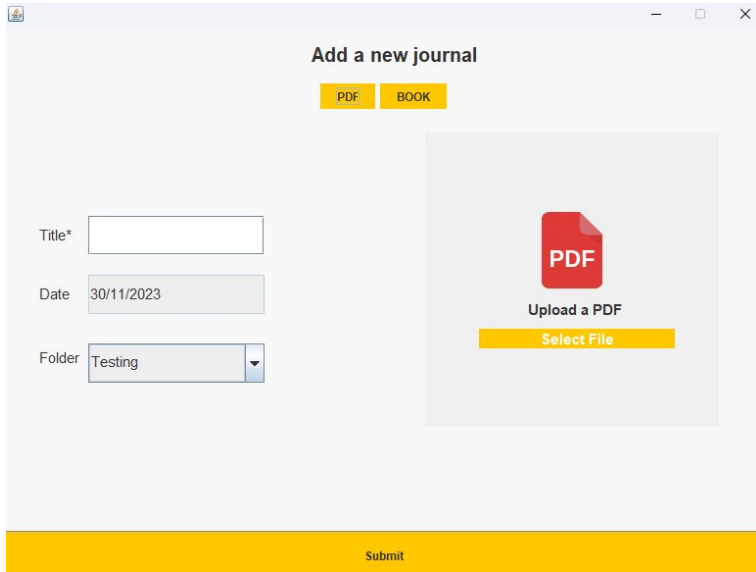


The screenshot shows a web application window titled "Add a new journal". At the top, there are two yellow buttons labeled "PDF" and "BOOK". Below these, there are three input fields: "Title\*" (a text box), "Date" (a date picker showing "30/11/2023"), and "Folder" (a dropdown menu showing "Testing"). To the right of these fields is a large gray box containing an orange book icon, the text "Upload using ISBN", an ISBN input field, and a yellow "Check ISBN" button. At the bottom of the window is a wide yellow "Submit" button.

```
public String searchBook(String ISBN) {  
    try {  
        //Setting up the query parameters  
        JsonFactory jsonFactory = GsonFactory.getDefaultInstance();  
  
        final Books books = new Books.Builder(  
            transport: GoogleNetHttpTransport.newTrustedTransport(),  
            jsonFactory,  
            tpRequestInitializer: null)  
            .setApplicationName(applicationName: "API key 1")  
            .setGoogleClientRequestInitializer(  
                new BooksRequestInitializer(key))  
            .build();  
  
        //Filtering Results  
        String query = ISBN;  
        Books.Volumes.List volumesList = books.volumes().list(q: query);  
        Volumes volumes = volumesList.execute();  
  
        // If there are no books found  
        if (volumes.getTotalItems() == 0) {  
            return "";  
        }  
  
        // Store the returned data value  
        for (Volume volume : volumes.getItems()) {  
            Volume.VolumeInfo volumeInfo = volume.getVolumeInfo();  
            authors = volumeInfo.getAuthors();  
            imageLink = volumeInfo.getImageLinks().getThumbnail();  
            title = volumeInfo.getTitle();  
            return "Success";  
        }  
    }  
    catch (Exception error) {  
        return "";  
    }  
}
```

Calls the API and searches the ISBN to determine whether it is valid.

# Uploading Reading Materials

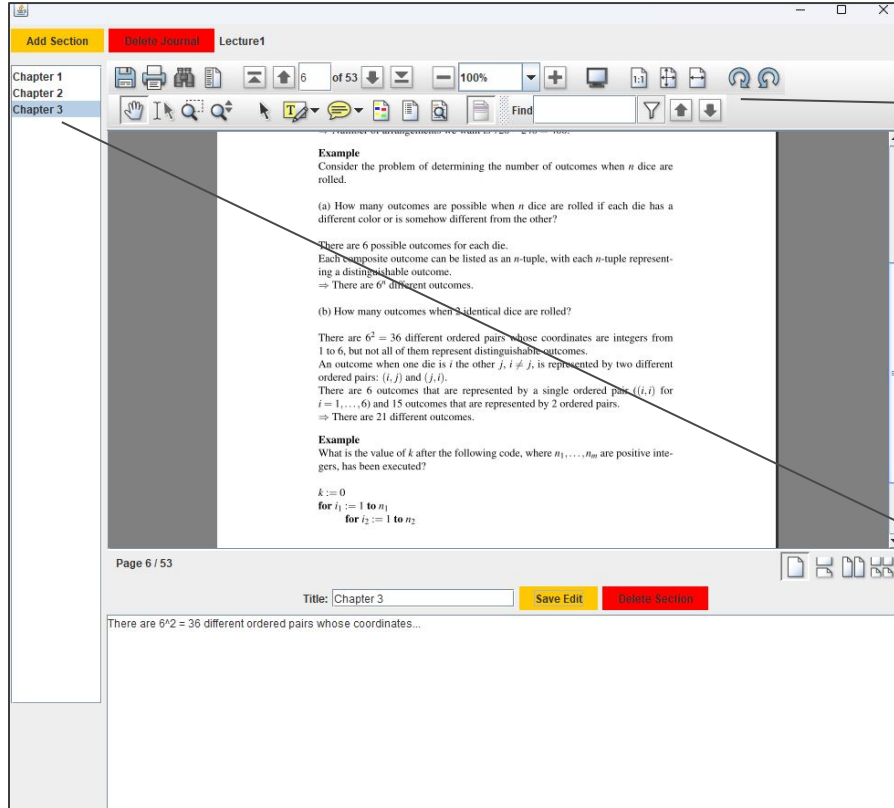


The screenshot shows a web form titled "Add a new journal". At the top, there are two yellow buttons labeled "PDF" and "BOOK". Below these, on the left, are three input fields: "Title\*" (a text box), "Date" (a text box containing "30/11/2023"), and "Folder" (a dropdown menu showing "Testing"). On the right side of the form, there is a large light gray box containing a red PDF icon and the text "Upload a PDF" above a yellow "Select File" button. At the bottom of the form is a wide yellow button labeled "Submit".

```
private File uploadPDF() throws FileNotFoundException{  
    JFileChooser fileChooser = new JFileChooser();  
  
    FileNameExtensionFilter filter = new FileNameExtensionFilter(  
        description: ".pdf",  
        extensions: "pdf");  
  
    fileChooser.setFileFilter(filter);  
    fileChooser.showOpenDialog(parent: null);  
  
    File file = new File(pathname: fileChooser  
        .getSelectedFile()  
        .getAbsolutePath());  
  
    return file;  
}
```

Select a PDF file and filter it for PDF files only.

# Viewing PDF uploaded



```
// Adding the controller
controller = new SwingController();
SwingViewBuilder factory = new SwingViewBuilder(sc: controller);
JPanel viewerComponentPanel = (JPanel) factory.buildViewerPanel();
controller.setToolBarVisible(btn:true);

ComponentKeyBinding.install(ctrl: controller, jc: viewerComponentPanel);

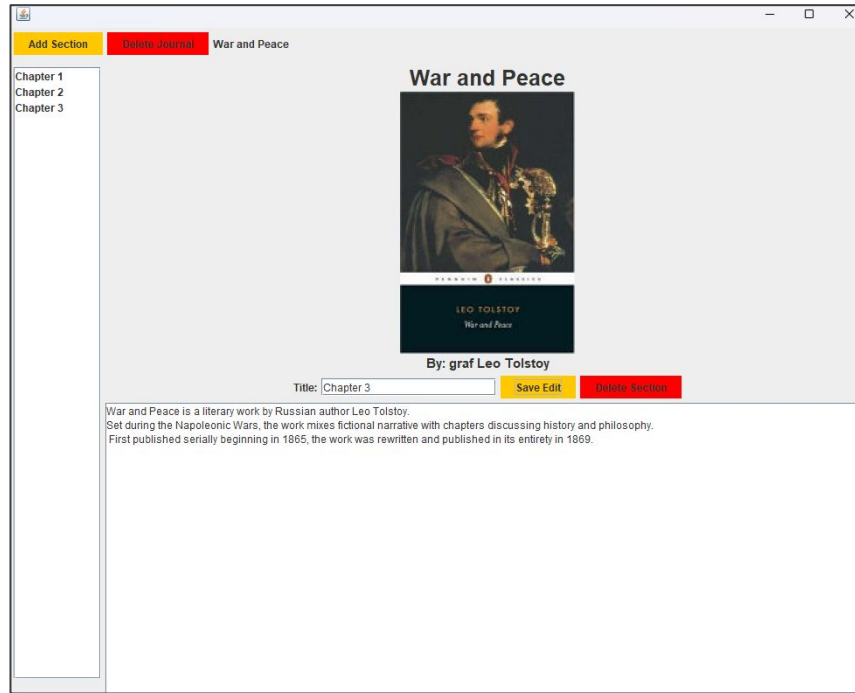
// Configuring the controller
controller.getDocumentViewController().setAnnotationCallback(
    new org.icepdf.ri.common.MyAnnotationCallback(
        dvc: controller.getDocumentViewController()));
```

```
private void loadSections(Journal journal) throws Exception {
    // Setting the query
    PreparedStatement statement = conn.prepareStatement(
        sql:"SELECT * FROM sections WHERE idjournals = ?" );
    statement.setInt(parameterIndex: 1, n: journal.getID());
    // Execute the query
    ResultSet result = statement.executeQuery();

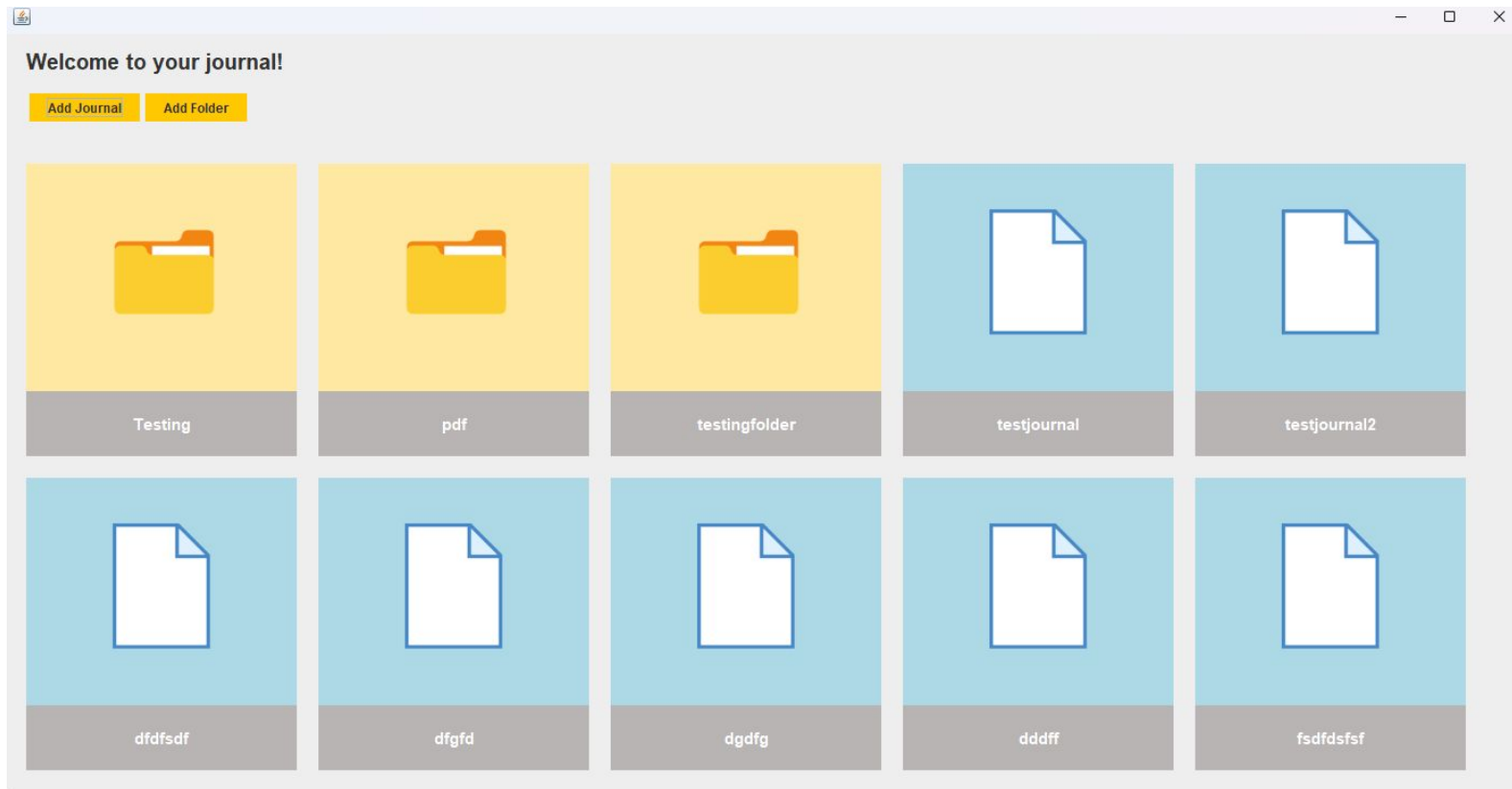
    // Iterating through the results and adding it to a list
    while (result.next()) {
        section newSection = new section(
            s: result.getString(columnLabel:"title"),
            n: result.getString(columnLabel:"note"),
            s: result.getInt(columnLabel:"idsections")
        );
        sectionList.add(e: newSection);
    }
}
```



# Viewing A Book



# Main Page



# After clicking the folder...



Welcome to your journal!

Add Journal

Add Folder

Delete Folder

Back



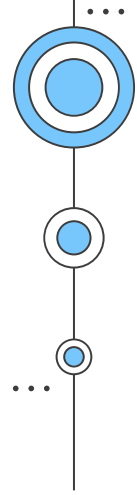
fjsdfklksdjf



dfhfh

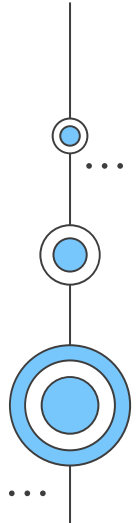


dfgdfgdfg



# 05

## Testing





# Some Test Cases

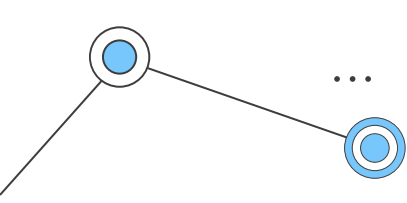


Requirement	Test Input	Expected Output
Upload PDF files into the app	A random PDF file	Input Success!
Select a book using the <b>International Standard Book Number (ISBN)</b> when adding a journal	9780140447934	Input Success!
Select a book using the <b>International Standard Book Number (ISBN)</b> when adding a journal	534594308540985093485904 859823490584950348590834 0598	Submission Failed!
Select a book using the <b>International Standard Book Number (ISBN)</b> when adding a journal	null	Submission Failed!
View the book details (author, title, cover) based on ISBN	Open a journal of book-type	The book cover, author name and book title
Create Journals for the selected reading material	Title: "Lecture 1" Folder: "none" PDF: "any pdf"	Input Success!
Create Journals for the selected reading material	Title: null Folder: "none" PDF: "any pdf"	Creation Failed!
Create Journals for the selected reading material	Title: "Lecture 1" (already exist) Folder: "none" PDF: "any pdf"	No duplicate titles
Read journal	Open a journal	<ul style="list-style-type: none"><li>• (if PDF) -&gt; pdf should be viewable</li><li>• If (ISBN) -&gt; book title, author, and bookcover visible</li><li>• Sections list contains all the sections</li></ul>

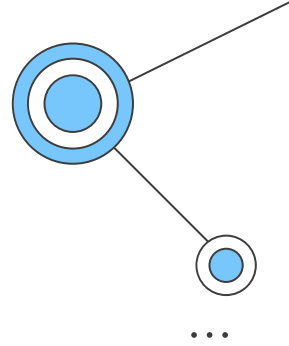
Ensure that the requirements are met.

Types:

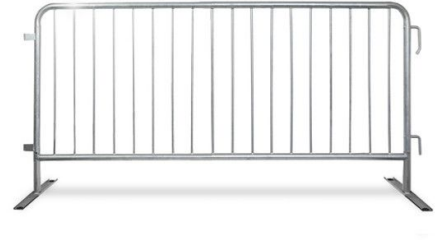
- Boundary values
- Empty values
- Functional values
- Negative values **x**



# Challenges Faced



- Integrating the PDF renderer and Google Books API
- Creating the user interface
- Refreshing data and managing windows





# Limitations of the App



Is the app perfect? No.

- System vulnerabilities
- An integrated toolbar for the text editor – storing formatted texts (bullet points, bold texts, etc..)
- Concurrency management / scalability