**Introduction**

The **Student Past Paper Review System** is a Java-based desktop application designed to help students access and manage past examination papers efficiently. With a simple and user-friendly graphical interface, the application enables students to upload, view, and search past papers based on metadata like course name and year. This project aims to simplify the process of finding relevant past papers for academic study, ensuring that students can easily review the materials needed for their coursework and exams.

The system supports three key functionalities:

1. **Upload Past Paper**: Students can upload past papers by selecting a file from their local storage and entering relevant metadata such as the course name and year.
2. **View Past Papers**: Users can browse a list of all uploaded past papers, which are displayed with their metadata (e.g., course name and year).
3. **Search Past Papers**: Students can search past papers by course name or year, allowing them to quickly filter and access relevant documents.

By organizing past papers and providing easy access to them, this system aims to streamline the study process for students, saving them time and effort in locating the necessary study materials.

Additionally, the project is designed to be intuitive, with clear navigation options and straightforward functionality. Whether you are looking to upload new papers, review old ones, or find papers for specific courses or years, this system offers a simple yet effective solution.

In the next sections, we will delve deeper into the features of the application, describe the user interface, and discuss the challenges faced during development, along with potential solutions and improvements for the future.

**Features Implemented:**

1. **Main Menu**:
   * The main interface consists of a simple window with three buttons:
     + **Upload Past Paper**: Allows the user to upload a past paper file, enter metadata (course name and year), and save it.
     + **View Past Papers**: Displays all uploaded past papers in a new window with clickable buttons for each past paper.
     + **Search Past Papers**: Allows the user to search past papers by course name or year. It filters and displays matching papers.
2. **Upload Past Paper**:
   * The user can choose a file through the JFileChooser dialog, after which metadata (course name and year) is entered using input dialogs.
   * The past paper is stored in an ArrayList as a PastPaper object.
3. **View Past Papers**:
   * Displays a list of past papers, each with a button. When a button is clicked, the corresponding file is opened using the Desktop.getDesktop().open() method.
4. **Search Past Papers**:
   * Prompts the user for a search term (course name or year).
   * It filters and displays matching past papers in a new window, each with a clickable button to open the file.
5. **Open File**:
   * The Desktop.getDesktop().open() method is used to open the selected file on the system's default application (e.g., PDF reader).

**Screenshot of the Project Interface:**

Currently, I cannot generate actual screenshots. However, I can describe the layout:

1. **Main Menu Window**:
   * This is the initial window that contains three buttons stacked vertically:
     + **Upload Past Paper**
     + **View Past Papers**
     + **Search Past Papers**
2. **Upload Window**:
   * This is a file chooser dialog that lets the user select a past paper file from their computer.
   * After selecting a file, input fields prompt the user to enter the course name and year.
3. **View Past Papers Window**:
   * A new window shows a vertical list of buttons. Each button represents a past paper, displaying the file name, course name, and year.
   * Clicking on a button opens the corresponding file.
4. **Search Results Window**:
   * Similar to the "View Past Papers" window, but only displaying past papers that match the search term (course name or year).

**Challenges Faced During Development:**

Here are specific challenges students face when reviewing past exam papers and their corresponding software-based solutions:

1. **Difficulty Finding Relevant Past Papers.**

Students struggle to find past papers relevant to their specific course, topic, or year. Papers are often scattered across different platforms or stored in disorganized formats.

Solution: Implement a Searchable Repository

Develop a search and filter system in the app to allow students to find past papers based on keywords (e.g., course name, year, topic). Use a database to store file metadata (e.g., course name, year, file path). Allow users to search using text input.

1. **Inability to View and Annotate Past Papers**

Many students cannot review past papers directly within the application or add personal notes/highlights for better understanding.

Solution: Build an Integrated Viewer with Annotation Features Integrate a PDF viewer using libraries like Apache PDFBox or JavaFX. Enable users to highlight text, add comments, and save annotations to review later.

1. **Lack of Feedback Mechanism**

Students review past papers without knowing if their answers are correct. This limits learning and preparation.

Solution: Include an Interactive Quiz Mode Extract questions from past papers (or allow students to input them). Enable a quiz system where students answer questions and get instant feedback.

Store quiz results to track progress.

1. **Scattered Notes and Review Materials**

Students often keep notes and materials for revision separately from past papers, making it hard to correlate them.

Solution: Add a Note-Taking Feature

Allow students to write and save notes linked to specific past papers.

Include a note retrieval system to access notes quickly when reviewing.

1. **Lack of Progress Tracking**

Students don’t have an overview of how many past papers they’ve reviewed or their improvement over time.

Solution: Create a Progress Tracker Track and display the number of past papers reviewed, quizzes taken, and scores. Provide visual progress indicators, such as bar charts or completion percentages. Software-Based Solution: An Integrated Java Application

**Possible Improvements:**

* Implement a database or use file-based storage to handle large collections of past papers more efficiently.
* Add sorting and filtering options to the view and search windows.
* Implement better error handling and input validation to improve the robustness of the system.
* Enhance the graphical user interface (GUI) for a more modern and user-friendly experience.

If you need any help implementing these features or tackling the challenges, feel free to ask!