**Project Horizon**

**“Broadening Horizons, Shaping Your Future”**

A logo with a tree and red leaves

Description automatically generated

**Group Members**

Huzaifa Nasir 22i-1053

Maaz Ali 22i-1042

Kaf Abbas 22i-0763

Table of Contents

[Abstract 4](#_Toc183364208)

[Introduction 4](#_Toc183364209)

[System Features 4](#_Toc183364210)

[Administrator Features 4](#_Toc183364211)

[Student Features 4](#_Toc183364212)

[Guest Features 5](#_Toc183364213)

[Technologies Used 5](#_Toc183364214)

[Project Structure 5](#_Toc183364215)

[Directory Overview 6](#_Toc183364216)

[Database Schema 6](#_Toc183364217)

[Design Patterns 10](#_Toc183364218)

[GoF Design Patterns 10](#_Toc183364219)

[Singleton Pattern 10](#_Toc183364220)

[Factory Method Pattern 10](#_Toc183364221)

[Observer Pattern 10](#_Toc183364222)

[Command Pattern 11](#_Toc183364223)

[GRASP Patterns 11](#_Toc183364224)

[Controller 11](#_Toc183364225)

[Information Expert 11](#_Toc183364226)

[Creator 11](#_Toc183364227)

[High Cohesion 12](#_Toc183364228)

[Low Coupling 12](#_Toc183364229)

[Polymorphism 12](#_Toc183364230)

[Workflow Overview 12](#_Toc183364231)

[Admin Workflow 12](#_Toc183364232)

[Student Workflow 12](#_Toc183364233)

[Guest Workflow 13](#_Toc183364234)

[Design Enhancements 13](#_Toc183364235)

[Consistent Styling: 13](#_Toc183364236)

[Improved Scholarship Details View: 13](#_Toc183364237)

[Responsive Design: 13](#_Toc183364238)

[Images of the Application UI 14](#_Toc183364239)

[Login 14](#_Toc183364240)

[Signup 15](#_Toc183364241)

[Admin Dashboard 16](#_Toc183364242)

[Student Dashboard 22](#_Toc183364243)

[Guest Scholarship View 34](#_Toc183364244)

[Conclusion 34](#_Toc183364245)

# Abstract

The Scholarship Management System is a robust JavaFX-based desktop application designed to streamline the management and application process for scholarships. It serves three distinct user groups: administrators, students, and guests. Administrators manage scholarships, review applications, and generate reports. Students can view scholarships, apply for them, track their application status, and provide feedback, while guests can only view available scholarships. The system is built following best software design practices, utilizing GRASP and GoF Design Patterns to ensure modularity, scalability, and maintainability. This report discusses the system's architecture, workflows, and implementation of design patterns in detail.

# Introduction

The Scholarship Management System solves the challenges of manual scholarship management by digitizing the process. With this system, users can efficiently handle tasks such as:

* Administrators managing scholarship data, student applications, and generating reports.
* Students tracking scholarship statuses, receiving notifications, and providing feedback.
* Guests accessing scholarship information without needing to log in.

The project utilizes a layered architecture for better separation of concerns and implements several design patterns to enhance code quality and usability.

# System Features

## Administrator Features

1. **Login**: Secure login system to authenticate administrators.
2. **Dashboard**: Overview of system statistics, including available scholarships, applications, and approvals.
3. **Manage Scholarships**: Add, edit, and delete scholarship records.
4. **Approve/Reject Applications**: Review and update application statuses.
5. **Generate Reports**: Create detailed reports on system usage and scholarship data.
6. **Manage Account**: Update personal admin details.

## Student Features

1. **Login**: Secure authentication for students.
2. **Dashboard**: Personalized view with quick statistics and access to:
   * Available scholarships.
   * Application history and status tracking.
   * Notifications about new scholarships.
3. **Scholarships**: View details and apply for scholarships.
4. **Notifications**: Receive real-time updates on scholarships and system announcements.
5. **Feedback**: Submit feedback on scholarships or the system.

## Guest Features

1. **Scholarship Viewer**: Browse available scholarships without requiring login credentials.
2. **Scholarship Details**: View detailed information about scholarships.

# Technologies Used

1. **JavaFX**: For building the graphical user interface (GUI) of the application.
2. **Java**: Core programming language for implementing business logic and backend functionality.
3. **SQL Server**: Relational database for storing and managing application data such as users, scholarships, applications, notifications, and feedback.
4. **JDBC (Java Database Connectivity)**: For connecting the application to the SQL Server database.
5. **FXML**: Used for defining the layout and structure of the application's GUI.

# Project Structure

The system adheres to a well-defined structure using a layered architecture:

1. **Presentation Layer**: JavaFX views (FXML and CSS) for user interaction.
2. **Controller Layer**: Handles user input and coordinates between the view and model layers.
3. **Model Layer**: Represents the core business logic and data entities.
4. **Data Access Layer**: Manages database interactions.

# Directory Overview

* **Controllers**: Manage interactions between the UI and business logic.
  + Examples: AdminDashboardController, ScholarshipController, LoginController.
* **Models**: Define core data entities.
  + Examples: Application, Scholarship, Feedback.
* **Utilities**: Provide reusable components like DatabaseUtil for database operations and SessionManager for session management.
* **Views**: FXML and CSS files for user interfaces.

# Database Schema

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY(1,1),

UserName VARCHAR(255) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Role VARCHAR(255) NOT NULL CHECK (Role IN ('Student', 'Admin'))

);

CREATE TABLE Scholarships (

ScholarshipID INT PRIMARY KEY IDENTITY(1,1),

ScholarshipName VARCHAR(255) NOT NULL,

ProviderName VARCHAR(255) NOT NULL,

Description TEXT NOT NULL,

EligibilityCriteria TEXT,

ApplicationDeadline DATETIME NOT NULL,

ApplicationRequirements TEXT,

Level VARCHAR(255) NOT NULL CHECK (Level IN ('BS', 'MS', 'PhD')),

Country VARCHAR(255) NOT NULL,

ViewCount INT DEFAULT 0,

ApplicationCount INT DEFAULT 0

);

CREATE TABLE Applications (

ApplicationID INT PRIMARY KEY IDENTITY(1,1),

ScholarshipID INT NOT NULL,

UserID INT NOT NULL,

FullName VARCHAR(255) NOT NULL,

FatherName VARCHAR(255),

Phone VARCHAR(255),

Address VARCHAR(255),

DateOfBirth DATETIME,

University VARCHAR(255),

DegreeProgram VARCHAR(255) NOT NULL CHECK (DegreeProgram IN ('BS', 'MS', 'PhD')),

CGPA FLOAT,

InterMarks FLOAT,

MatricMarks FLOAT,

AcademicAchievements TEXT,

StatementOfPurpose TEXT,

Extracurricular TEXT,

CNIC VARCHAR(255) NOT NULL,

Status VARCHAR(255) NOT NULL CHECK (Status IN ('Submitted', 'Under Review', 'Approved', 'Rejected')),

IsApproved Bit DEFAULT 0,

SubmissionDate DATETIME NOT NULL,

ReviewDate DATETIME,

StatusUpdateDate DATETIME,

ReviewerComments TEXT,

FOREIGN KEY (ScholarshipID) REFERENCES Scholarships(ScholarshipID) ON DELETE CASCADE,

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE

);

CREATE TABLE Notifications (

NotificationID INT PRIMARY KEY IDENTITY(1,1),

UserID INT NOT NULL,

Event TEXT NOT NULL,

Message TEXT NOT NULL,

SentAt DATETIME NOT NULL,

IsRead Bit DEFAULT 0,

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE

);

CREATE TABLE Feedback (

FeedbackID INT PRIMARY KEY IDENTITY(1,1),

UserID INT NOT NULL,

FeedbackType VARCHAR(255) NOT NULL,

FeedbackText TEXT NOT NULL,

SubmittedAt DATETIME NOT NULL,

AdminResponse TEXT,

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE

);

-- Add ScholarshipName column

ALTER TABLE Feedback

ADD ScholarshipName VARCHAR(255);

-- Add Rating column

ALTER TABLE Feedback

ADD Rating FLOAT CHECK (Rating BETWEEN 1 AND 5); -- Ensure rating is between 1 and 5

CREATE TABLE Documentation (

DocumentationID INT PRIMARY KEY IDENTITY(1,1),

UserID INT NOT NULL,

FileName VARCHAR(255) NOT NULL,

FilePath VARCHAR(255) NOT NULL,

ActionTime DATETIME NOT NULL,

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE

);

CREATE TABLE Logs (

LogID INT PRIMARY KEY IDENTITY(1,1),

UserID INT NOT NULL,

Action VARCHAR(255) NOT NULL,

ActionTime DATETIME NOT NULL,

FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE

);

CREATE TABLE Bookmarks (

BookmarkId INT PRIMARY KEY IDENTITY(1,1),

UserId INT NOT NULL,

ScholarshipId INT NOT NULL,

CreatedAt DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (UserId) REFERENCES Users(UserId) ON DELETE CASCADE,

FOREIGN KEY (ScholarshipId) REFERENCES Scholarships(ScholarshipId) ON DELETE CASCADE

);

# Design Patterns

The system leverages GoF (Gang of Four) and GRASP (General Responsibility Assignment Software Patterns) principles for modular and maintainable code.

## GoF Design Patterns

### Singleton Pattern

* + **Used In**:
    - DatabaseUtil:
      * Ensures only one database connection is active at a time.
      * Reduces resource consumption and improves performance.
  + **Benefits**:
    - Centralized resource management.
    - Improved consistency across the application.

### Strategy Method Pattern

* + **Used In**:
    - GenerateReportsController
    - ScholarshipsOverviewReportStrategy: Generates the "Scholarships Overview" report.
    - StudentParticipationReportStrategy: Generates the "Student Participation" report.
    - SystemUsageStatisticsReportStrategy: Generates the "System Usage Statistics" report.
  + **Benefits**:
    - Simplifies object creation logic.
    - Promotes loose coupling by abstracting the creation process.

### Observer Pattern

* + **Used In**:
    - Notification System:
      * Observes updates in scholarship data and notifies students in real time.
  + **Benefits**:
    - Ensures students stay informed about new scholarships and application updates.

### Facade Pattern

* + **Used In**:
* UserFacade Class:
  + - * Acts as the facade, centralizing and simplifying access to user-related operations like login and signup.
      * Hides the complexity of UserService and its underlying logic.
      * LoginController Class:
      * Instead of interacting directly with UserService, it calls methods on UserFacade for operations like login and signup.
      * Keeps the controller code cleaner and more focused on user interaction
      * **Benefits**:
    - Encapsulates actions into reusable commands, simplifying testing and extensibility.

## GRASP Patterns

### Controller

* + Examples:
    - LoginController, ScholarshipController, and AdminDashboardController.
  + Role:
    - Centralize control and mediate between the UI and business logic.

### Information Expert

* + Examples:
    - Application: Holds application data, including statuses and associated user information.
    - Scholarship: Encapsulates scholarship-specific data, such as eligibility and requirements.

### Creator

* + Examples:
    - DatabaseUtil: Creates objects like Scholarship, Application, and Notification when retrieving data from the database.

### High Cohesion

* + Each class has a single, focused responsibility:
    - Feedback focuses solely on feedback management.
    - Notification deals exclusively with notifications.

### Low Coupling

* + Views and models are loosely coupled through controllers, enhancing maintainability.

### Polymorphism

* + Used In:
    - ScholarshipController and ScholarshipGuestController:
      1. Both controllers handle viewing scholarships but differ in functionality.
      2. Students can apply for scholarships, while guests cannot.

# Workflow Overview

## Admin Workflow

1. Login as admin via LoginController.
2. Access the admin dashboard via AdminDashboardController:
   * View system statistics.
   * Manage scholarships using ManageScholarshipsController.
   * Approve/reject applications through ApproveRejectApplicationsController.
   * Generate reports using GenerateReportsController.

## Student Workflow

1. Login as student via LoginController.
2. Access the student dashboard via DashboardController:
   * View scholarships using ScholarshipController.
   * Track application status using TrackApplicationStatusController.
   * Receive notifications via NotificationsController.
   * Submit feedback using ProvideFeedbackController.

## Guest Workflow

1. Access scholarships via ScholarshipGuestController.
2. View scholarship details without applying.

# Design Enhancements

## Consistent Styling:

* + Dynamic styling applied to buttons, grids, and text for improved readability.

## Improved Scholarship Details View:

* + Enhanced modal design for displaying scholarship details (e.g., showScholarshipDetails method).
  + Organized layout with proper spacing, borders, and colors.

## Responsive Design:

* + Scrollable content (ScrollPane) for managing dynamic data.
  + Grid-based layouts for scalability across screen sizes.

# Images of the Application UI

## Login

A screenshot of a login screen

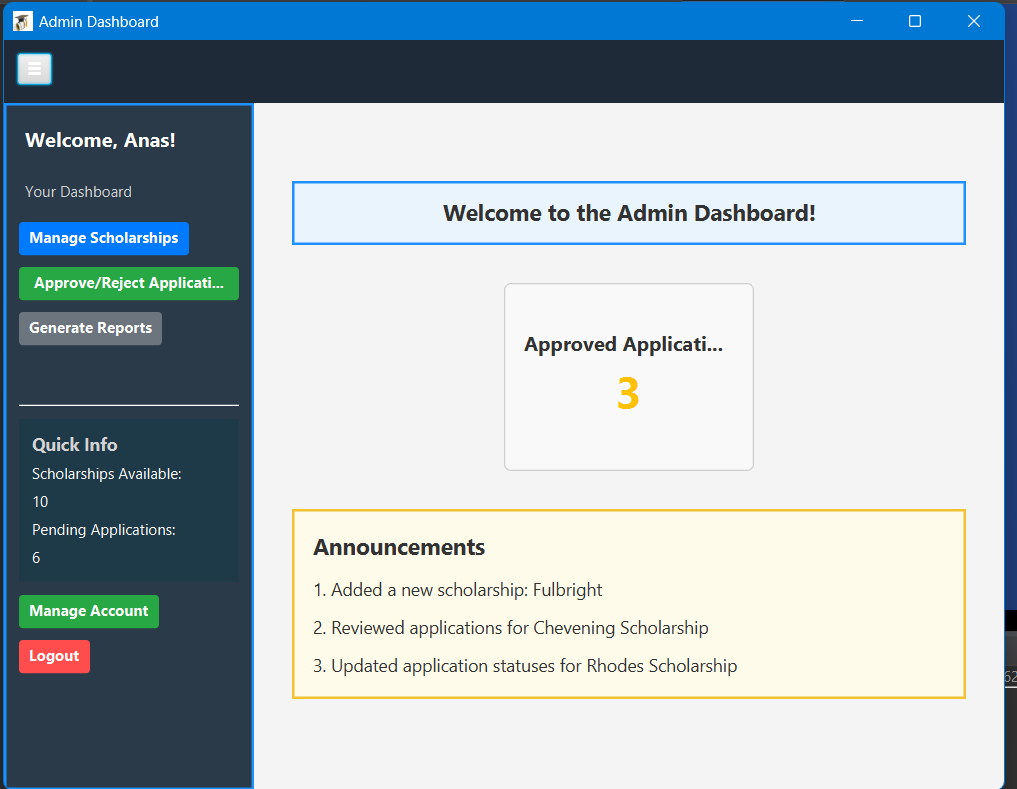
Description automatically generated

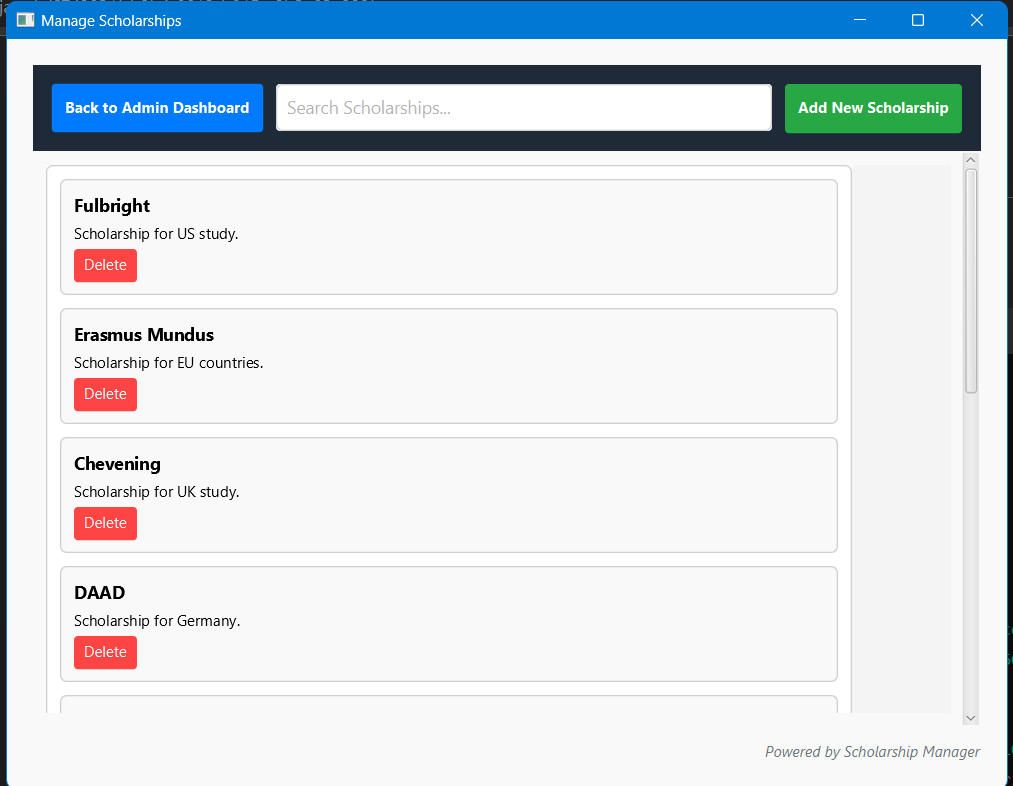
## Signup

A screenshot of a login form

Description automatically generated

## Admin Dashboard





A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

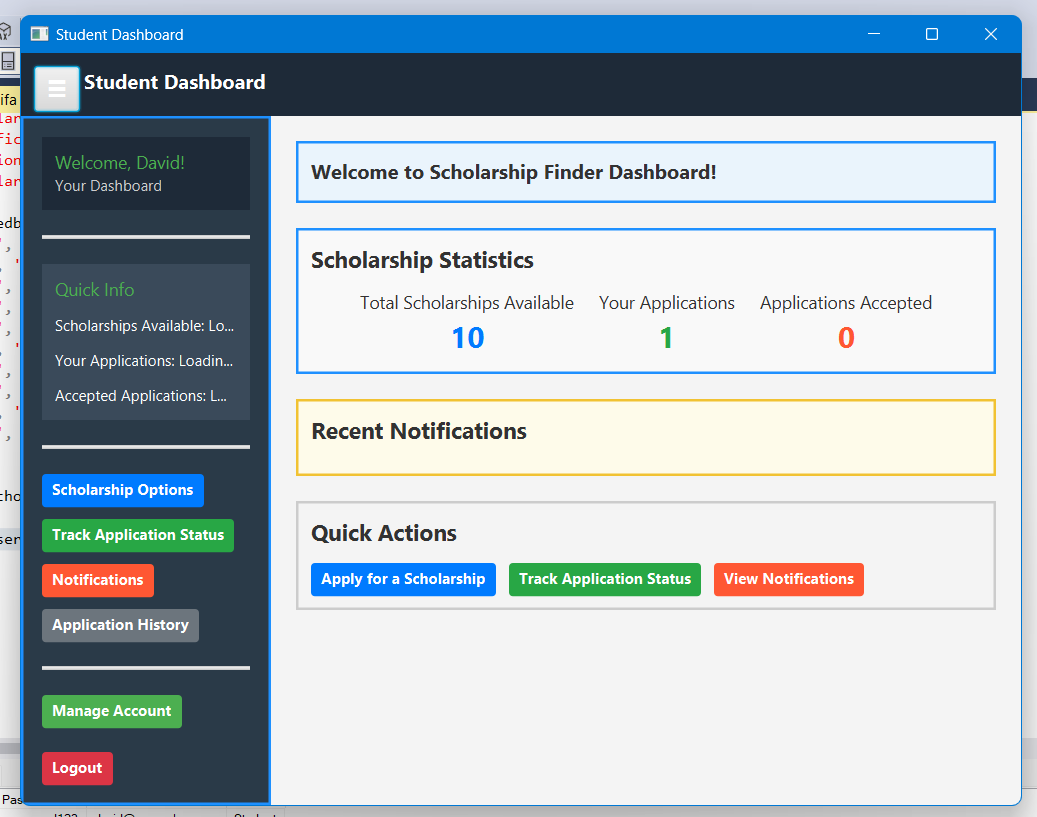
A screenshot of a computer

Description automatically generated

A screenshot of a computer

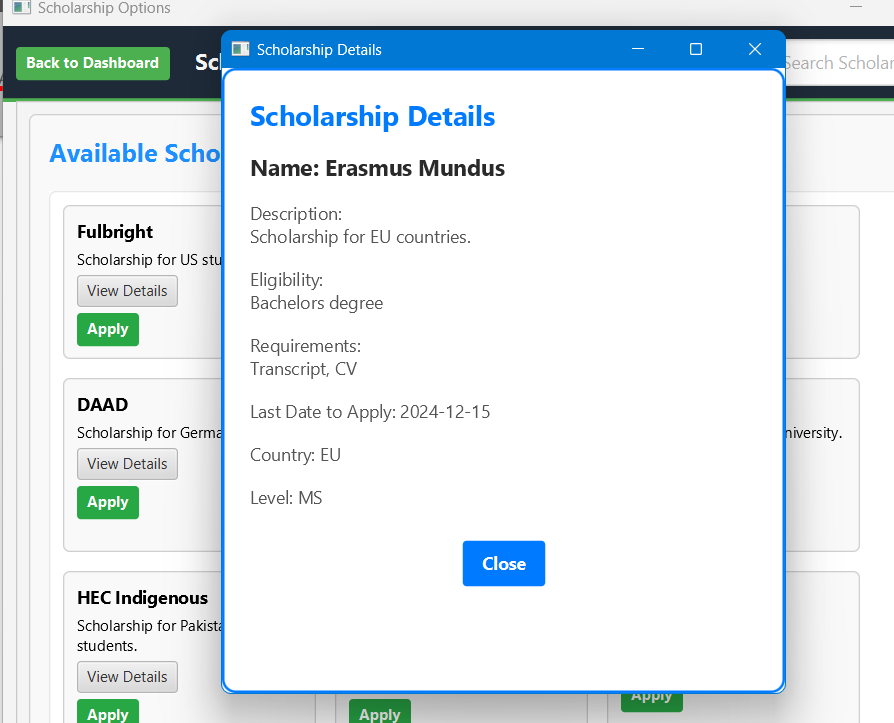
Description automatically generated

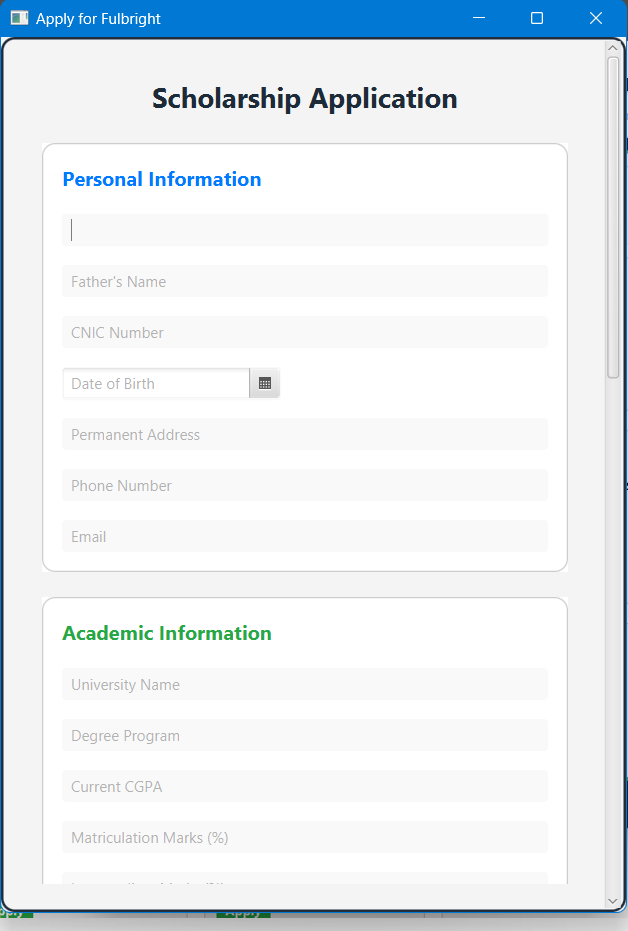
## Student Dashboard



A screenshot of a computer

Description automatically generated





A screenshot of a computer

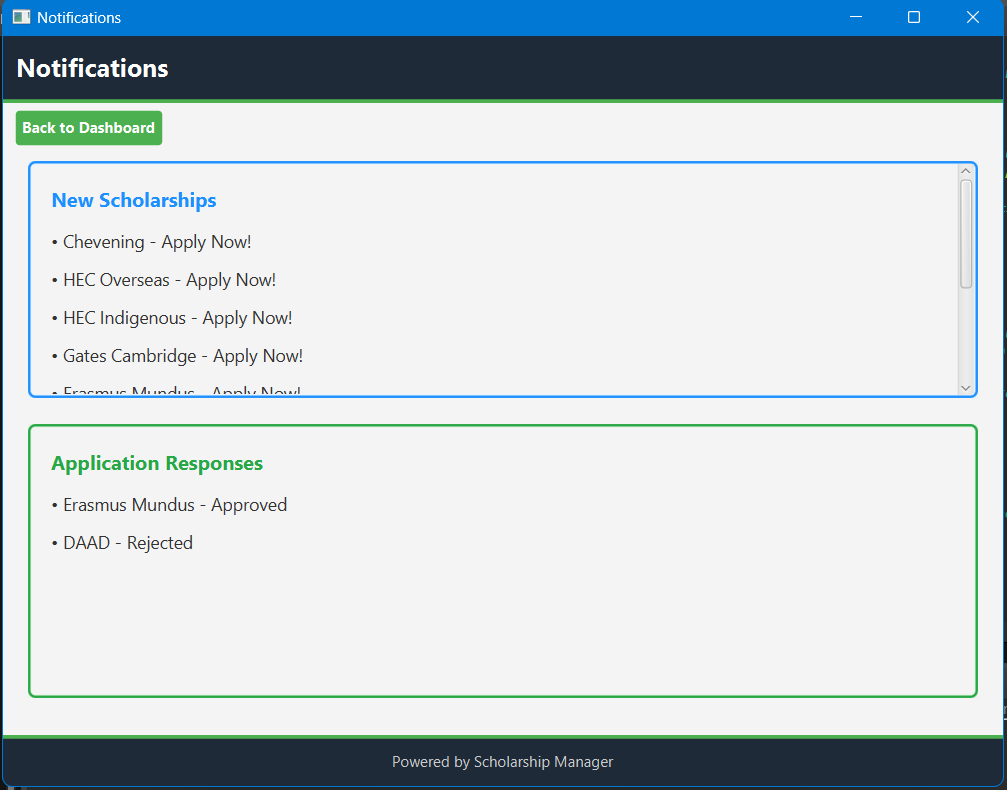
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Guest Scholarship View

A screenshot of a computer

Description automatically generated

# Conclusion

The Scholarship Management System demonstrates a well-structured implementation of software design principles. GoF Design Patterns like Singleton, Factory Method, and Observer ensure code reusability and modularity, while GRASP Patterns like Controller and Information Expert maintain a clean separation of concerns. The system is user-friendly, scalable, and maintainable, offering a strong foundation for future enhancements, such as integrating analytics or advanced user role management.