CSC 440 Fall 2021

Individual Project

Micah Meadows: Registrar Database Management System

# Intro

## Problem Statement

EKU’s Registrar Office needs to update student course records after receiving student grades from all of the faculty at the end of each semester. Each faculty member provides the office an Excel file with letter grades (A, B, C, D, or F) of the students who took course(s) with them. The office is asking you to develop a system to manage the data for them. They need you to develop a software system to do the following tasks:

* Add new grades of courses for each student to a database: The office can gather all the Excel files from the faculty, and put them in a folder. The formats of the names for the folder and Excel files are:

Folder name: “**Grades [Year] [Semester]**”

(Ex., “**Grades 2021 Spring**”, “**Grades 2023 Fall**”)

File name: “**[Course Prefix] [Number] [Year] [Semester]**”

(Ex., “CSC 440 2021 Fall”, “MAT 234 2027 Spring”)

* Edit a grade for a student: There might be some mistakes in the Excel files provided by the faculty. The system shall allow the office to edit the grade of a course for a student. Editing a grade means changing the grade, deleting the grade, and/or adding a grade to the database.
* Print a report card (or transcript) for a student: The system shall let them print a report card (or transcript) for a selected student. The report card (or transcript) should list student’s name, id, overall gpa, and a lists of courses with grades that he/she has taken before.

## Proposal

* To solve the needs of EKU’s Registrar Office I will develop a system that will allow for the management of the database through a user-friendly interface and validating the choices made to avoid possible errors

# System Description

## Framework

The grade management system is built using Windows Presentation Foundation (WPF) and following the MVVM design pattern to deliver a separation of concerns in regards to the presentation views, view interaction, and business logic.

## Database

The grade management system is built to not rely on a specific database implementation but instead rely on abstractions to follow the Dependency Inversion principle. However, for the instance of the EKUs registrar an SQL database implementation is included as well as a local memory testing solution. In the instance that EKU would swap their database to a solution such as a NoSQL database it would take minimal effort to swap as all of the system processes are abstracted.

# System Requirements

## Functional Requirements

1. The system shall allow the user to log in to the system
   1. The system shall display a login form to the user
   2. The user shall enter their username and password and submit the login form
      1. If the username and password are valid submit the user
      2. If the username or password are invalid do not submit the user
   3. The login system will navigate to the main system
2. The system shall allow for new grades to be added for each course into the database from a formatted file
   1. The user shall select “Upload Records” from the navigation menu on the GUI
   2. The system shall display the Upload Records page
   3. The system should allow a user to import files from his computer
      1. If a file is incorrect format the system will not allow it and display an error message
      2. If a file is recognized as correct it will be added to the database
3. The system shall allow for the searching of students by id, and or name
   1. The user shall select ‘Student Search’ From the navigation menu on the GUI
   2. The user shall be displayed with a list all students
   3. The user shall input information into the filter to narrow down the student selection
      1. The system will output a list of students that apply to the entered information
   4. The user shall be able to select a student to display their course records
      1. If a user has no course records the list will be empty
      2. If a user has course records the list will be populated with them
4. The system shall allow for manual input of course Records
   1. The user will select a student by following R3
   2. The user will select the option to add a course record manually to a student
   3. The system shall display a form popup allowing for the input of information pertaining to the added course record
   4. The system will attempt to import the new course record into the database
      1. If the database already holds the record return an error
5. The system shall allow for the editing of course records
   1. The user shall select a student by following R3
   2. The user will have the option to edit a course record manually by selecting the course record after selecting a student
   3. The system shall display a form popup allowing for the edit of information pertaining to the selected course record
      1. If the database already holds the record return an error telling the user that the data already exists
6. The system shall allow for the deletion of course records
   1. The user shall select a student and then course by following R3
   2. The user shall be presented with the option to delete a course
   3. The user shall click the option to delete a course
   4. The system shall delete the course from the database
7. The system shall allow for the creation of a Transcript or Report Card
   1. The user shall select a student by following R3
   2. The user shall have the option to generate a report
   3. The user shall select the generate report button
   4. The system will attempt to generate a report
      1. If the report has failed to generate the system will display an error to the user
      2. If the report is created it will tell the user it has succeeded and save the file to the computer
8. The user shall have the option to view the application settings
   1. The user shall select ‘Settings’ from the navigation menu
   2. The user will be presented with page of settings
   3. The user will change any settings available

## Nonfunctional Requirements

* Courses must be loaded into database by an administrator in order to determine how many credit hours a course has for that semester
* Students must be loaded into database by an administered in order to prevent a student that should not exist being added

# Use Case Diagram



# Data Flow Diagrams

## Level 0

### Context



## Level 1

### From P0



### From P1



### From P2



## Level 2

### From P3



### From P4



### From P5



### From P6



## Level 3

### From P5.2



# Structure Chart

## 

# Database Design

## ER Diagram

A picture containing text, iPod

Description automatically generated

## Table Schema

Student (ID, Name, GPA)

Course (Prefix, Number, Year, Semester, Hours)

Grade (Prefix, Number, Year, Semester, Grade)

# Algorithm Design

### Calculate GPA algorithm design



# Conclusion

The system we have described will provide an easy way for the EKU registrar to manage their database system for student grades. It also has the capabilities to be scalable and extensible if the functional requirements change over time. The system will not need any major overhauling if requirements such as student filter options, or database storage location change.

# Data Dictionary

IStudentRepository: an interface for defining a student database

SqlStudentRepository: sql implementation of student database

ICourseRepository: an interface for defining a course database

SqlCourseRepository: sql implementation of course database

IGradeRepository: an interface for defining a grade database

SqlGradeRepository: sql implementation of grade database

IAccountRepository: an interface for defining an account database

SqlAccountRepository: sql implementation of account database

IReportGenerator: an interface for defining a report generator

MessageBoxReportGenerator: implementation of report generator that will print to a MessageBox

PdfReportGenerator: implementation of report generator that will save to a pdf

Account: a class model that represents a user account

Course: a class model that represents course information

CourseGrade: a class model that represents a students course result

Student: a class model that represents a student

StudentCourseResult: a class model that represents a students course result with hours

ViewModelBase: an abstract class that allows ViewModels to exist

MainViewModel: viewmodel that handles system navigation

LoginViewModel: viewmodel that handles log in screen interaction

TabsViewModel: viewmodel that handles the tabs screen

EditCourseGradeViewModel: viewmodel that handles course edit popup interaction

PopupWindowViewModel: viewmodel that handles popup window interaction

SettingsViewModel: viewmodel that handles settings page interaction

StudentSearchViewMode: viewmodel that handles search page interaction

UploadRecordsViewMode: viewmodel that handles record upload page interaction

UserControl: a class that allows for custom controls to be used

LoginView: a class that handles the GUI for login page

TabsView: a class that handles the GUI for the tabs page

SettingsView: a class that handles the GUI for the settings tab

StudentSearchView: a class that handles the GUI for the student search tab

UploadRecordsView: a class that handles the GUI for the upload records tab

EditCourseGrade: a class that handles the GUI for the edit course popup

PopupWindow: a class that handles the GUI for popup windows

RepositorySingleton: a singleton class that handles storing the repositories and authenticated users

AddGradeException: a class that represents an exception with adding a grade

CourseNotExistException: a class that represents an exception where a course does not exist

DeleteCourseException: a class that represents an exception where a course could not be deleted

GetCourseException: a class that represents an exception with getting a course record