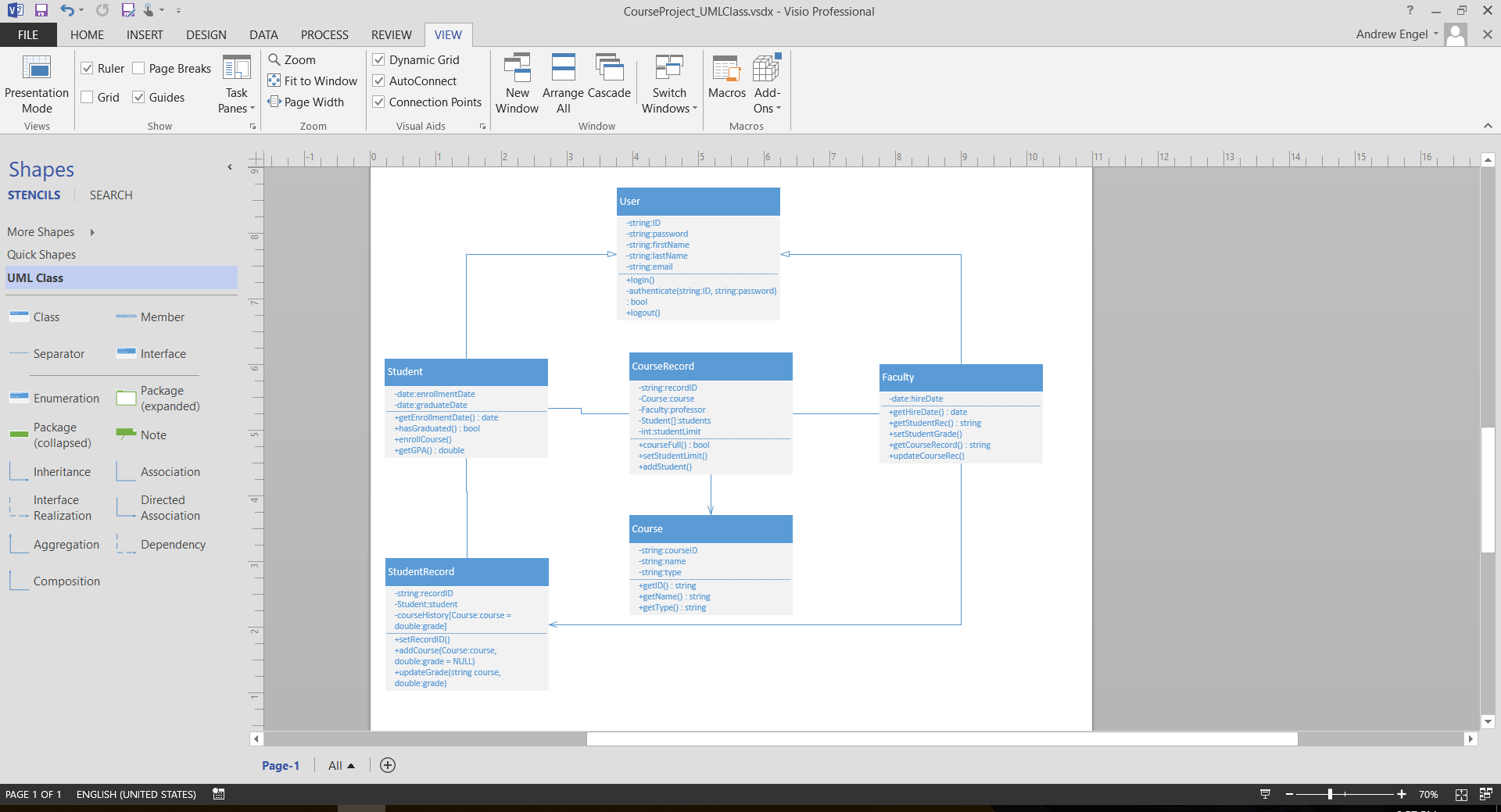
Exemplar Project

Andrew Engel

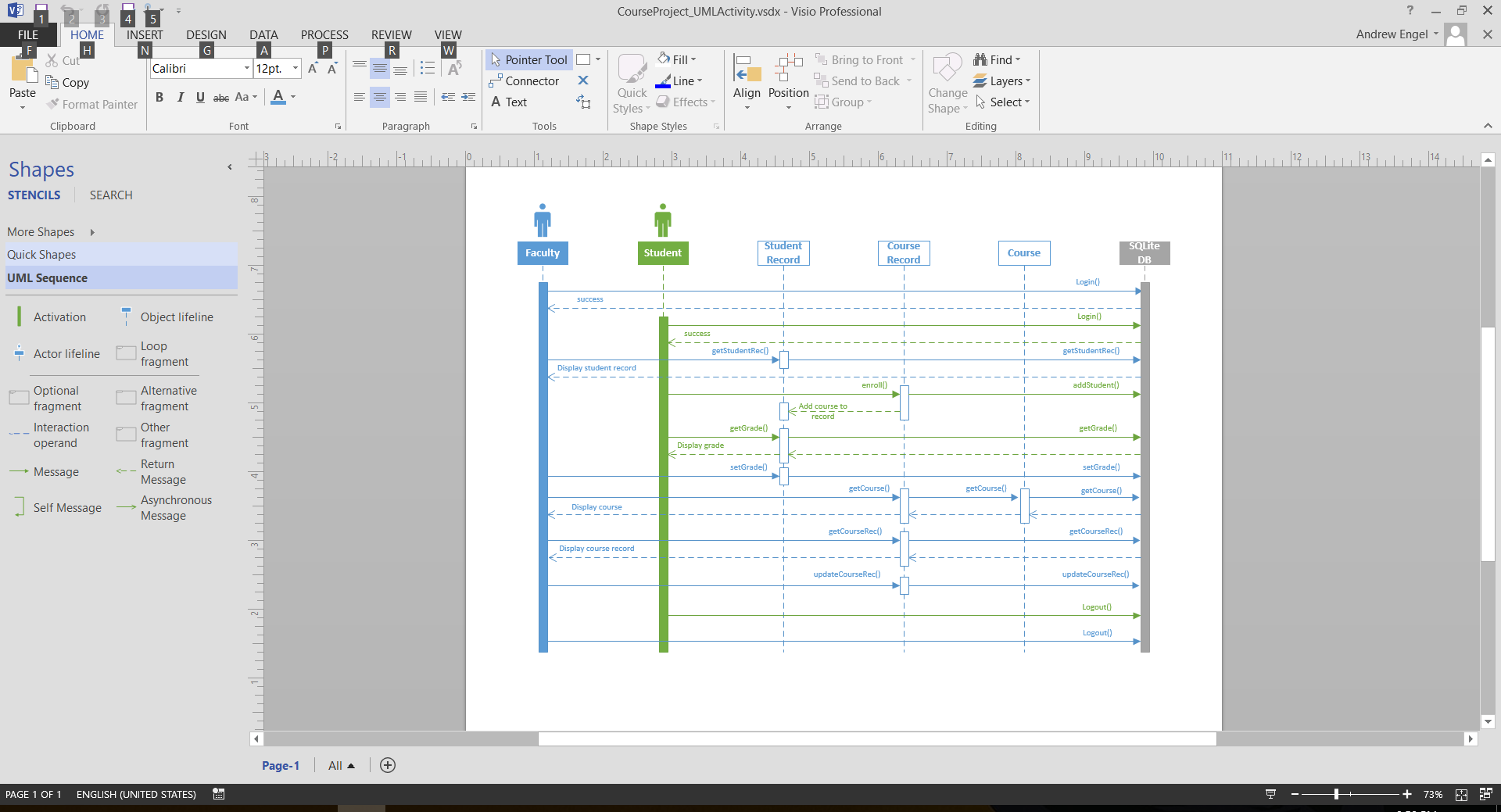
Rasmussen College

Class and Activity Diagrams

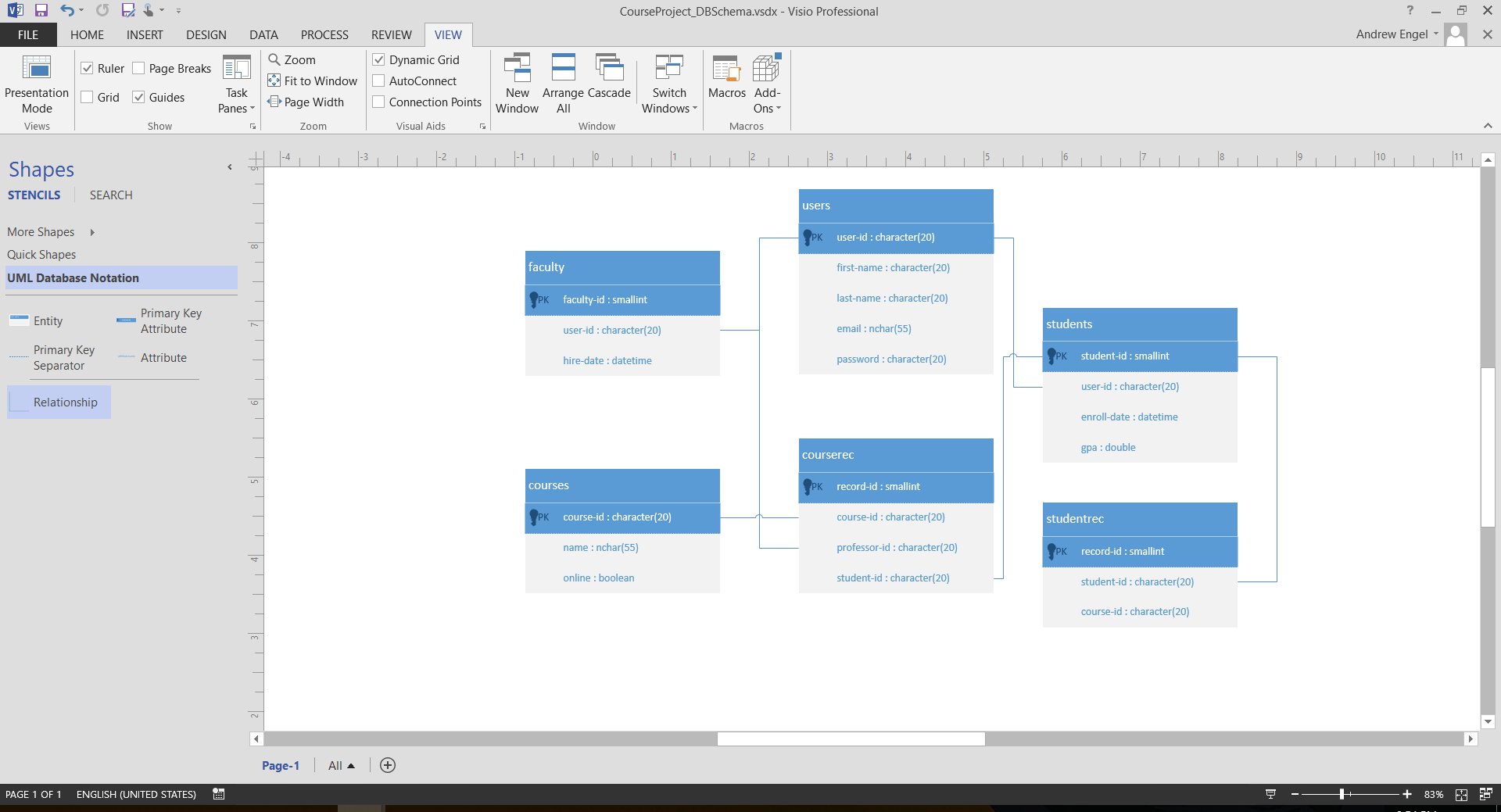
UML Class Diagram



UML Activity Diagram



SQLite Database Schema



SQLite Code

CREATE TABLE users(

User-id CHARACTER(20) PRIMARY KEY,

First-name CHARACTER(20),

Last-name CHARACTER(20),

Email NCHAR(55),

Password CHARACTER(20)

);

CREATE TABLE students(

Student-id SMALLINT PRIMARY KEY,

User-id CHARACTER(20),

Enroll-date DATETIME,

Gpa DOUBLE,

FOREIGN KEY(user-id) REFERENCES users(user-id)

);

CREATE TABLE faculty(

Faculty-id SMALLINT PRIMARY KEY,

User-id CHARACTER(20),

Hire-date DATETIME,

FOREIGN KEY(user-id) REFERENCES users(user-id)

);

CREATE TABLE courses(

Course-id CHARACTER(20) PRIMARY KEY,

Name NCHAR(55),

Online BOOLEAN

);

CREATE TABLE courserec(

Record-id SMALLINT PRIMARY KEY,

Course-id CHARACTER(20),

Professor-id CHARACTER(20),

Student-id CHARACTER(20),

FOREIGN KEY(course-id) REFERENCES courses(course-id),

FOREIGN KEY(professor-id) REFERENCES faculty(faculty-id),

FOREIGN KEY(student-id) REFERENCES students(student-id)

);

CREATE TABLE studentrec(

Record-id SMALLINT PRIMARY KEY,

Student-id CHARACTER(20),

Course-id CHARACTER(20),

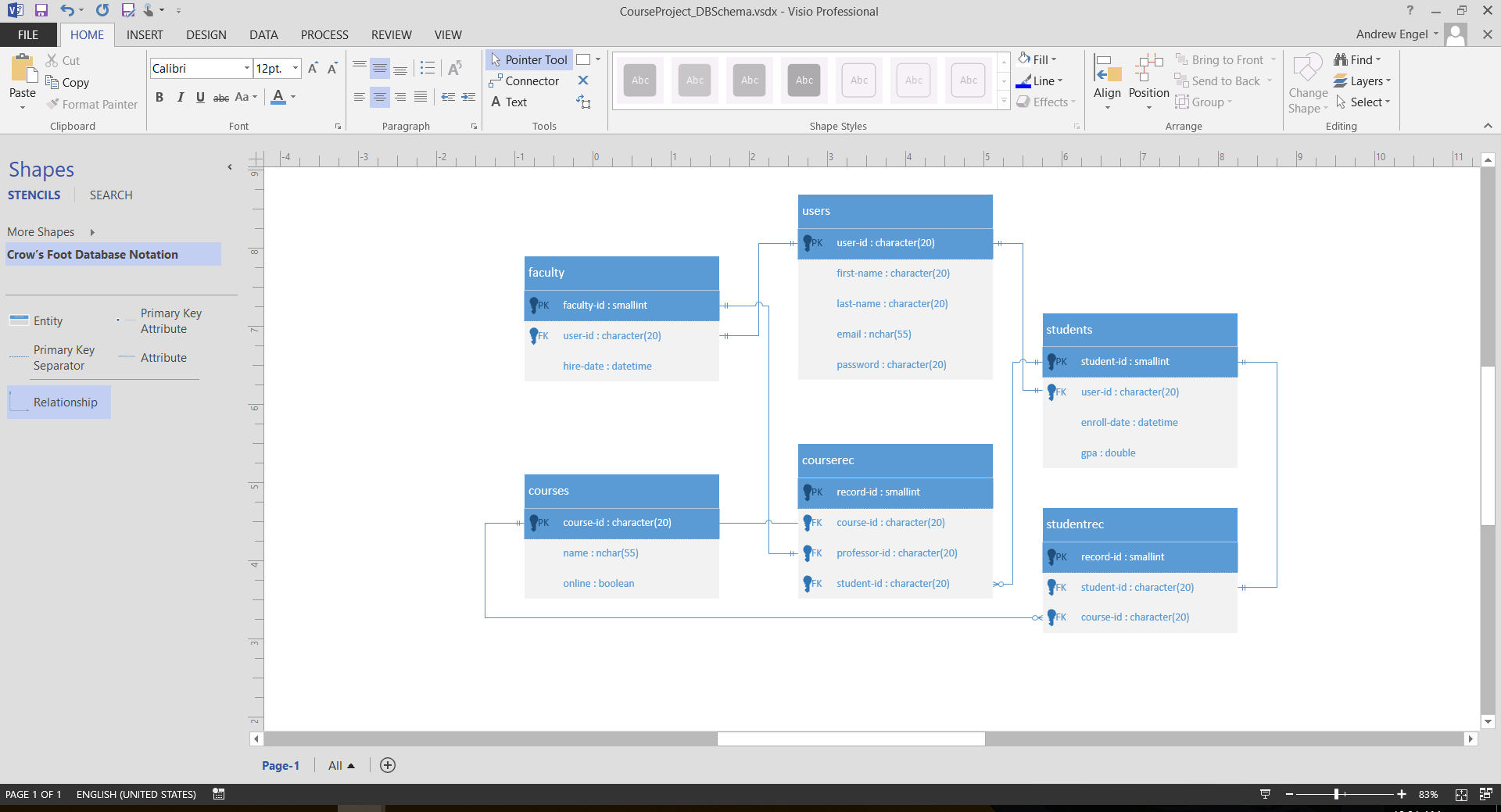
FOREIGN KEY(student-id) REFERENCES students(student-id),

FOREIGN KEY(course-id) REFERENCES courses(course-id)

);

Entity Relationship Modeling

ER Diagram



Source Code – Python / SQLite

import sqlite3

db = sqlite3.connect('db.py')

cur = db.cursor()

#create users table

print('create users table')

cur.execute("DROP TABLE IF EXISTS users")

cur.execute("""

CREATE TABLE users (

userid CHARACTER(20) PRIMARY KEY,

firstname CHARACTER(20),

lastname CHARACTER(20),

email NCHAR(50),

password CHARACTER(20),

accesslvl CHARACTER(20)

)

""")

#create faculty table

print('create faculty table')

cur.execute("DROP TABLE IF EXISTS faculty")

cur.execute("""

CREATE TABLE faculty (

facultyid SMALLINT PRIMARY KEY,

userid CHARACTER(20),

hiredate DATETIME,

FOREIGN KEY(userid) REFERENCES users(userid)

)

""")

#create students table

print('create students table')

cur.execute('DROP TABLE IF EXISTS students')

cur.execute("""

CREATE TABLE students (

studentid SMALLINT PRIMARY KEY,

userid CHARACTER(20),

enrolldate DATETIME,

gpa DOUBLE,

FOREIGN KEY(userid) REFERENCES users(userid)

)

""")

#create courses table

print('create courses table')

cur.execute('DROP TABLE IF EXISTS courses')

cur.execute("""

CREATE TABLE courses (

courseid CHARACTER(20) PRIMARY KEY,

name NCHAR(50),

online BOOLEAN

)

""")

#create courserec table

print('create courserec table')

cur.execute('DROP TABLE IF EXISTS courserec')

cur.execute("""

CREATE TABLE courserec(

recordid SMALLINT PRIMARY KEY,

courseid CHARACTER(20),

professorid CHARACTER(20),

studentid CHARACTER(20),

FOREIGN KEY (courseid) REFERENCES courses(courseid),

FOREIGN KEY (professorid) REFERENCES faculty(userid),

FOREIGN KEY (studentid) REFERENCES students(userid)

)

""")

#create studentrec table

print('create studentrec table')

cur.execute('DROP TABLE IF EXISTS studentrec')

cur.execute("""

CREATE TABLE studentrec (

recordid SMALLINT PRIMARY KEY,

studentid CHARACTER(20),

courseid CHARACTER(20),

FOREIGN KEY (studentid) REFERENCES students(studentid)

FOREIGN KEY (courseid) REFERENCES courses(courseid)

)

""")

print('insert rows')

#insert into users

cur.execute("""

INSERT INTO users (userid, firstname, lastname, email, password, accesslvl)

VALUES ('0001', 'admin', 'admin', 'admin@admin.com', 'password', 'root'),

('0002', 'Jim', 'Hinkins', 'jim.henkins@college.edu', 'jmaster', 'faculty'),

('0003', 'Billy', 'Matthews', 'billy.matthews@colledge.edu', 'bmaster', 'student')

""")

#insert into students

cur.execute("""

INSERT INTO students (studentid, userid, enrolldate, gpa)

VALUES ('0001', '0003', 05/01/18, 0.0)

""")

#insert into faculty

cur.execute("""

INSERT INTO faculty (facultyid, userid, hiredate)

VALUES ('0001', '0002', 04/01/16)

""")

#insert into courses

cur.execute("""

INSERT INTO courses (courseid, name, online)

VALUES ('0001', 'Geometry', 1),

('0002', 'Chemistry', 1),

('0003', 'Software Security', 1),

('0004', 'Python Programming', 0)

""")

print('commit')

db.commit()

print('read from db')

for row in cur.execute('SELECT \* FROM users'):

print(row)

print('close')

db.close()

Detailed Design/Pseudocode Development

The application I am creating allows students and faculty to view and manage school-related records. It is a console application which requires users to login using a valid user ID and password before accessing the system. Once logged in, students and faculty are presented with different menu options, preventing them from accessing areas of the system that are irrelevant or off-limits.

All records in the system are inserted by the administrator (who is currently the programmer), with the exception of students being able to enroll into a course. Students have the options to view their student record, view all courses, and enroll into a course. Faculty have the options to view a student record of their choice, view their course records, and give a student a grade.

The application is written in Python and connects to an SQLite database. The UML diagrams below showcase the classes, database structure, activities within the system, and use cases. The class diagram displays each class and module, their data members/methods, and the dependencies between them.

The database schema shows each table and their fields, and the relationships between them. The activity diagram shows the actors and objects in the system, how data is passed between them, and their life span. Lastly, the use case diagrams show who/what interacts with each of the primary functions of the system.

Working prototype code

The following is code that demonstrates the main function’s login interface. It begins by initializing the database, pre-filling it with records. In addition, a menu object is created. Then, a while loop begins with the condition that the authorized variable must be true before it will exit the loop. It calls the menu object’s method, getID(), which gets and returns the user’s input. Then it calls the menu object’s getPassword() method, which also gets and returns the user’s input. After that, the user ID and password are passed to the controller’s authenticate() function. If the database returns a match, the authorized variable is set to true, and the loop stops, allowing the user to proceed into the application.

def main():

controller.createDB()

menu1 = menu.Menu()

id = ""

pw = ""

userRecord = None

authorized = False

print("School Records Application")

while not authorized:

#get id and password from user

id = menu1.getID()

if id == "":

print("Username cannot be blank.")

return

pw = menu1.getPassword()

if pw == "":

print("Password cannot be blank")

return

#authenticate

userRecord = controller.authenticate(id, pw)

if userRecord != None:

authorized = True

else:

print("Invalid login - Access denied")

#end while

Pseudocode

viewStudentRecord(id)

Declare record as tuple

Declare courses as tuple

Call controller module’s function to request student record from database using id and assign to record variable

Call controller module’s function to request student courses from database using id and assign to courses variable

Print out student information and courses from record and courses tuples

viewAllCourses()

Declare record as tuple

Call controller module’s function to request all course records from database and assign to record variable

Iterate through each record, displaying course ID, name and whether it is online

//database function

enroll(studentID, courseID)

create database connection object

create cursor object

Declare listID variable

Using courseID, get the studentListID from courserec table from the database and assign it to the listID variable

Insert student into the courserecstudents table using the listID and studentID variables

Insert course into the studentrec table using the studentID and courseID variables

viewCourseRecord(facultyID)

Declare record as tuple

Call controller module’s function to request course record from database and assign to record variable

Iterate through each record, displaying course ID, name and each student’s first and last name

//database function

gradeStudent(courseID, studentID, grade)

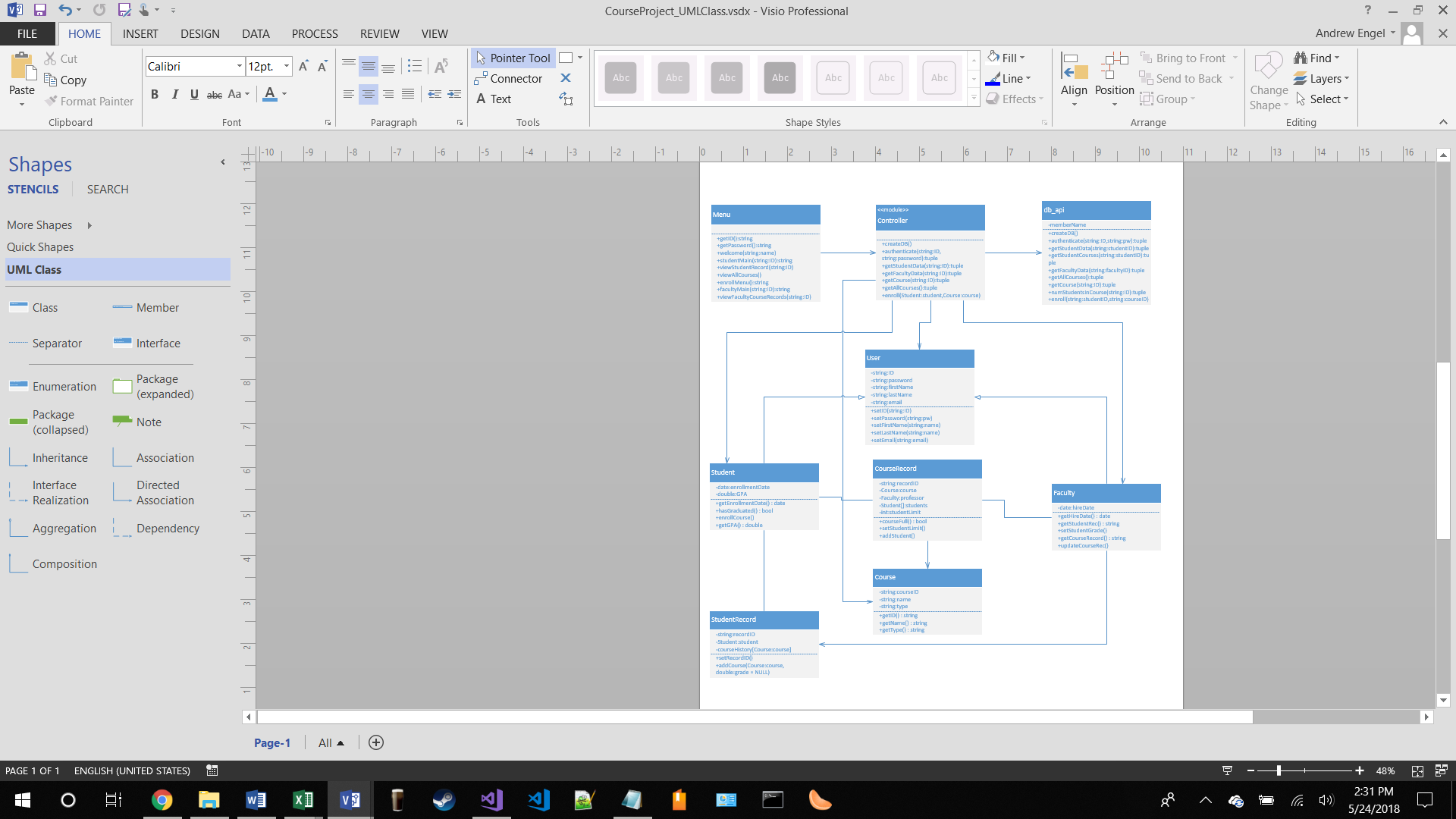
create database connection object

create cursor object

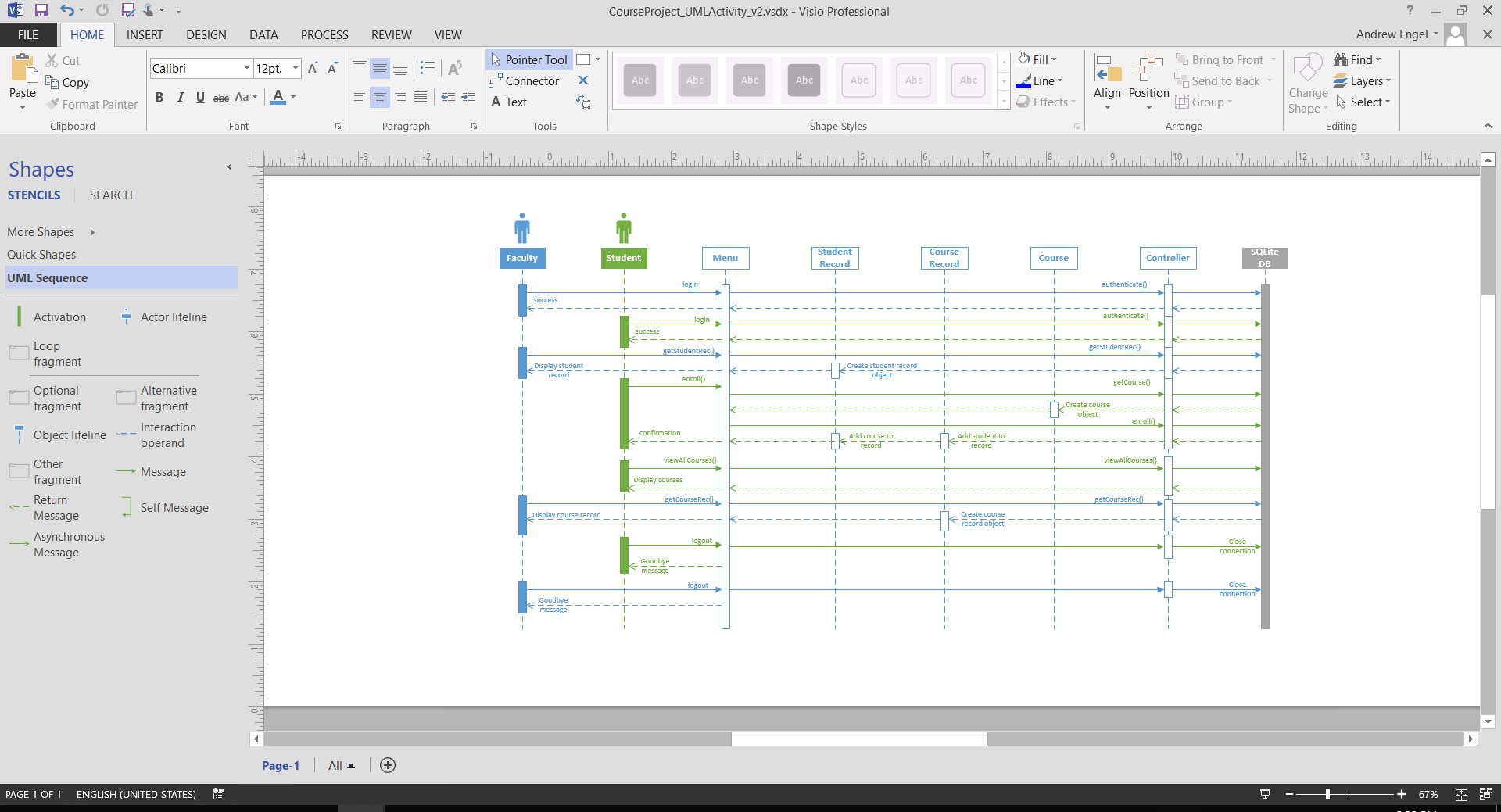
Find studentlistID in courserec table using courseID variable

Find student in courserecstudents by using studentID variable and insert grade using grade variable

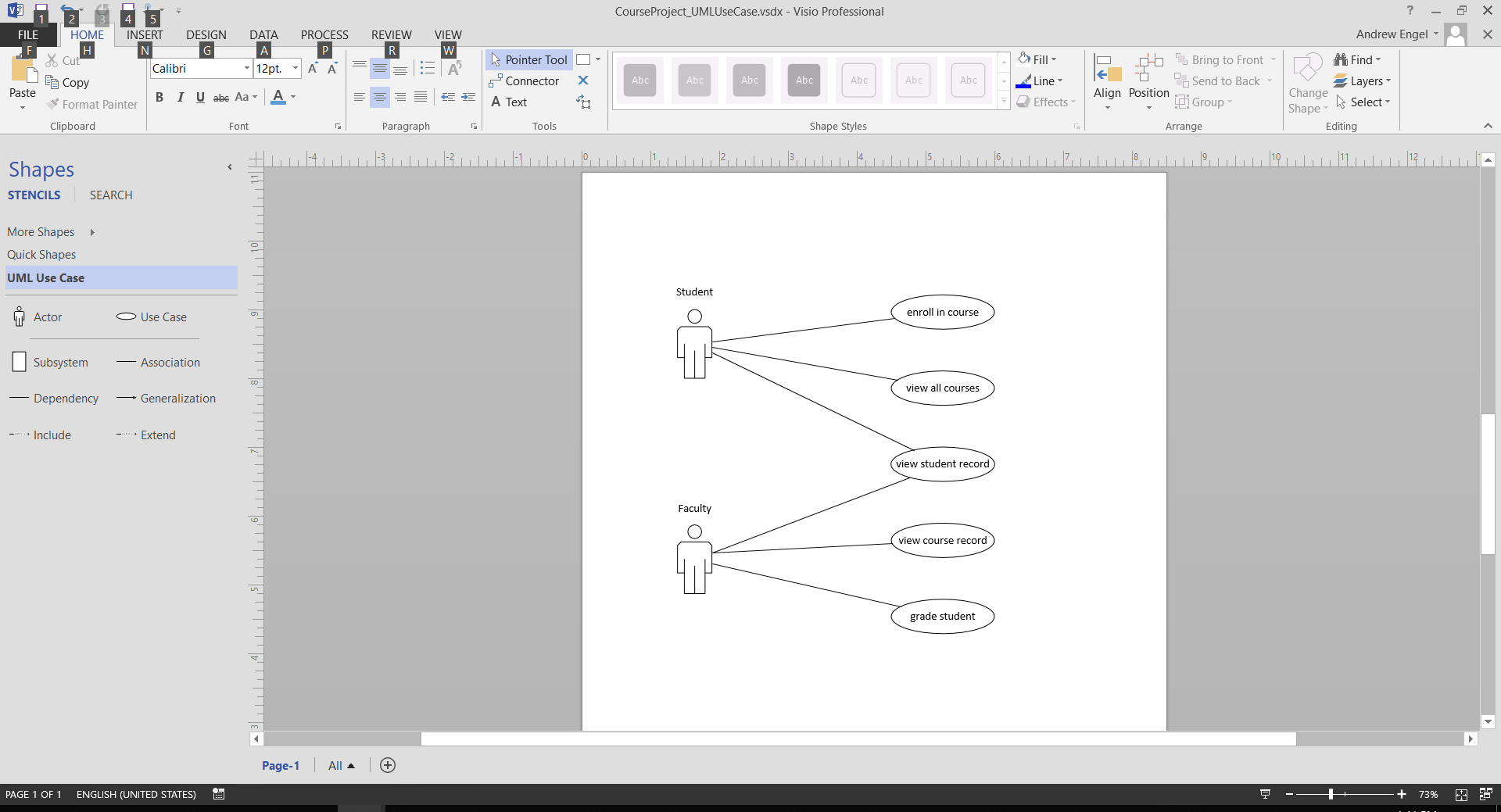
Class Diagram



Sequence Diagram



Use-case Diagram



Establishing a Security Plan

Establishing a security plan is critical to the success of the Rasputin College Application. In any environment, there are internal and external security threats that must be accounted for and prevented. The following security plan addresses potential security threats, a contingency plan, and how the access policy, privileges and data integrity are designed to help secure the application.

Access Policy

The access policy maintains that, at this time, only administrators can add new students, faculty members, courses, and course records. Only allowing administrators to access this part of the system prevents faculty members from creating students and adding themselves to course records. This type of access restriction allows the college’s processes of student enrollment and course management to be left to the college directors to handle. That way, we mitigate the possibility of additions being made to the database before the proper approvals are made.

In addition, the main application can only be accessed by students and faculty. Currently, faculty consists of professors only, but can be extended in the future to include other faculty members as well. This access constraint prevents anyone outside of the college to gain access to the system.

Privileges

Privileges in the system are maintained by the application’s menu system. Each user has a user ID and password they must enter before they are granted access to the application. Depending on if they are a student or faculty member, they will be taken to a different menu that displays options only available to them. This removes the option of students accessing faculty-only functions and faculty accessing student-only functions.

Data Integrity

Data integrity is maintained through data validation and database constraints. Data validation is used throughout the system on multiple layers. The application is designed with an MVC model, separating the view, business logic, and database into separate components. Each layer has validation so that if one layer fails to catch invalid data, the others are there as fail-safe mechanisms.

In addition, constraints have been placed in the creation of the database to maintain data integrity. The database currently uses primary key, unique, not null and default values. Primary keys ensure all records are uniquely identifiable, unique values ensure every record is different, not null ensures a column does not have a null value, and default inserts a default value if no value is provided for a new record.

Potential Security Threats

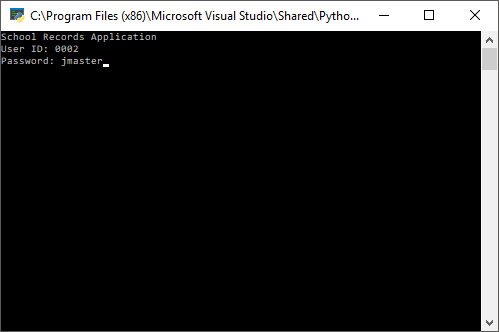
One security threat is having unauthorized users entering new students, faculty members, courses, or course records. This could be done if someone who was not authorized logged in as an administrator. This is why the access policy and privileges were designed so that only admins have can add these records. This way, any additions to the system are funneled through one or two users, minimizing security threats and making attacks easier to manage.

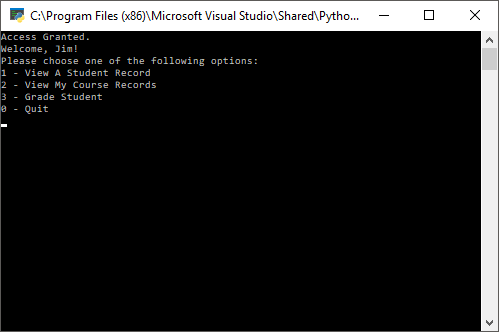
Another security threat is SQL injection. There are instances in the application where the user will be asked for an ID to look up in the system, which is then sent to the database to find the record. This is where data integrity helps mitigate these issues. All input is validated, and IDs must be exactly 4 characters long before they’re sent to the database, leaving hackers the opportunity to come up with exactly a 4 character SQL injection. As of right now, there doesn’t appear to be any way to successfully attack the system with SQL injection.

Contingency Plan

The contingency plan is to maintain nightly system backups. We will monitor logs on a weekly basis to ensure system integrity is maintained and free from successful attackers. If any corruption is found, we will restore the system, using the most recent, breach free backup. From there, we will identify how the breach was successful, and patch those areas of the system to avoid similar future attacks.

User Interface





User Interface Code

def main():

controller.createDB()

menu1 = menu.Menu()

id = ""

pw = ""

userRecord = None

os.system('cls' if os.name == 'nt' else 'clear')

authorized = False

print("School Records Application")

while not authorized:

#get id and password from user

id = menu1.getID()

if id == "":

print("Username cannot be blank.")

return

pw = menu1.getPassword()

if pw == "":

print("Password cannot be blank")

return

#authenticate

userRecord = controller.authenticate(id, pw)

if userRecord != None:

authorized = True

else:

print("Invalid login - Access denied")

#end while

id = userRecord[0]

privilege = userRecord[5]

choice = ""

if privilege == 'student':

data = controller.getStudentData(id)

st = student.Student(data[0], data[1], data[2], data[3], data[4], data[5], data[6])

os.system('cls' if os.name == 'nt' else 'clear')

menu1.welcome(st.firstName)

while choice != "0":

choice = menu1.studentMain(st.id)

if choice == '1':

menu1.viewStudentRecord(st.id)

elif choice == '2':

menu1.viewAllCourses()

elif choice == '3':

course1 = menu1.enrollMenu() #returns Course object

numStudents = controller.numStudentsInCourse(course1.id)

if numStudents[0] >= course1.studentLimit:

print("This class is full. Enrollment Aborted.")

return

else:

controller.enroll(st.id, course1.id)

print("You are now enrolled.")

elif choice == '0':

print("Goodbye!")

return

else:

print("Invalid entry.\n")

#pause and clear console

input("Press Enter to continue ...")

os.system('cls' if os.name == 'nt' else 'clear')

#End while

elif privilege == 'faculty':

data = controller.getFacultyData(id)

fac = faculty.Faculty(data[0], data[1], data[2], data[3], data[4], data[5])

os.system('cls' if os.name == 'nt' else 'clear')

menu1.welcome(fac.firstName)

while choice != '0':

choice = menu1.facultyMain(fac.id)

if choice == '1':

menu1.facViewStudentRecord()

elif choice == '2':

menu1.viewCourseRecords(fac.id)

elif choice == '3':

menu1.gradeStudent(fac.id)

elif choice == '0':

print("Goodbye!")

else:

print("Invalid entry.\n")

#pause and clear console

input("Press Enter to continue ...")

os.system('cls' if os.name == 'nt' else 'clear')

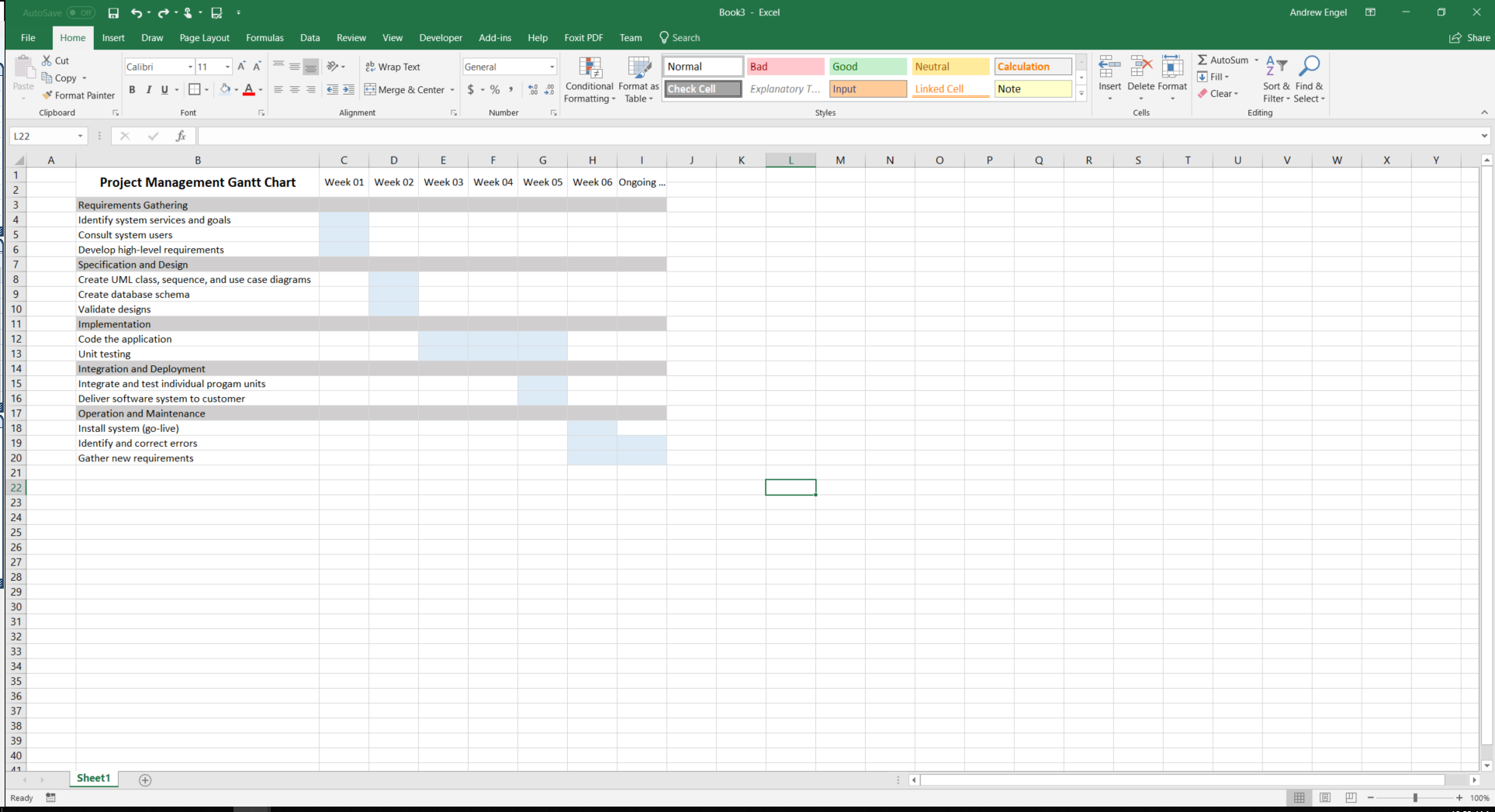
if \_\_name\_\_ == '\_\_main\_\_':

main()

Project Management Plan

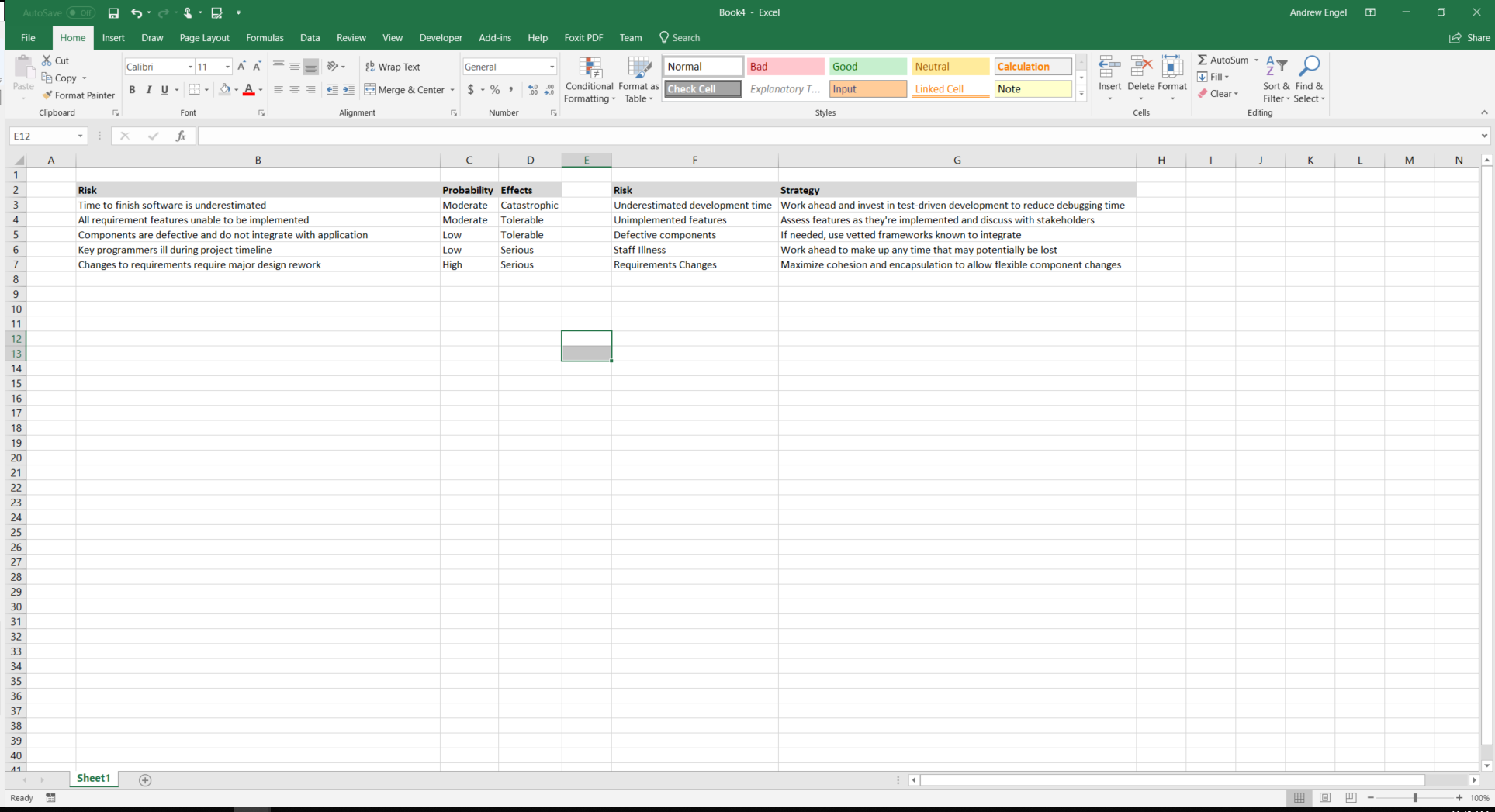
The following project management plan discusses the school record application’s design and implementation. It includes an activity chart, detailing timeline and deliverables, a risk analysis, project plan tracking, and processes to ensure quality in the software.

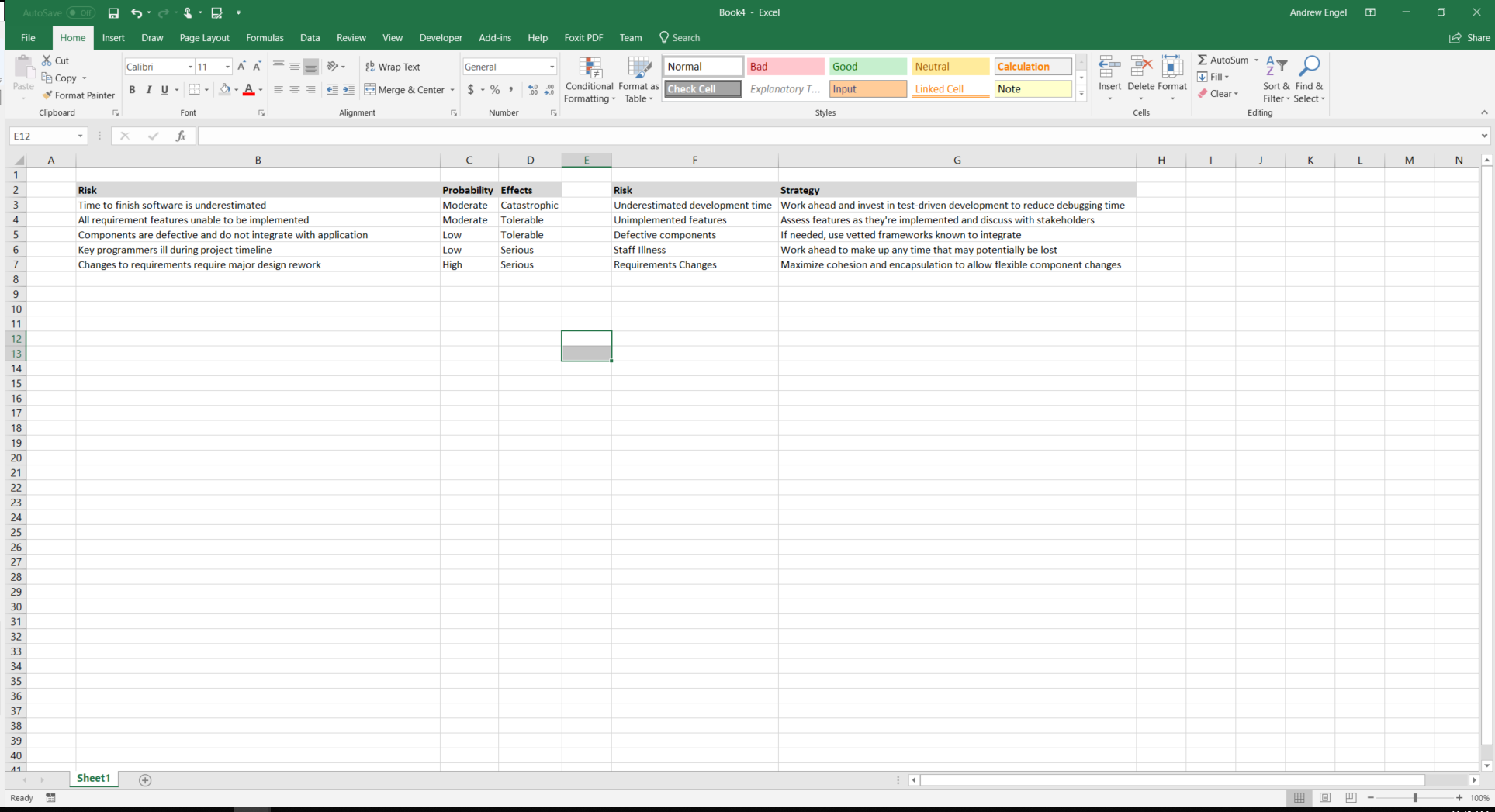
Activity Chart



Risk Analysis

A risk analysis is conducted in the beginning of the project to assist in foreseeing potential issues that may arise, their probability of occurring, and their impact on the project deliverables. Once identified and measured, each risk is then analyzed to determine a strategy for circumventing the risk. The following displays both the risk analysis and strategies for each of the risks.





Tracking

Tracking the project plan will be integral to the success of the project. Undoubtedly, challenges and unforeseen issues will arise, and tracking/monitoring will allow us to solve the problem and course-correct the project, if needed. The project plan will also be used as a reference guide when making decisions about the design. The team will meet daily to discuss progress, issues, and answer questions so we can maintain momentum, stay within scope of the project, and identify and solve issues as soon as possible.

Quality

Quality is of the utmost importance to this project, and there are processes in place to ensure quality standards are met. One of the ways to ensure quality is through test driven development. Test driven development creates tests for functions before the function is written, requiring the programmer to design the function to pass the test. This ensures that every function and component is fully tested before integrated with the rest of the module/system. While this may sound like it will extend the amount of time needed for implementation, it vastly reduces the amount of debugging and ensures the highest quality product.

In addition, each software component will be designed with high cohesion and low coupling. Cohesion will ensure that each module, class and function serve a narrowly defined purpose. This makes design choices simpler and it makes correcting coding issues much easier. Low coupling means that there are minimal dependencies between software components. Having fewer dependencies allows the software to be flexible to changes, helping us stay on our timeline when challenges arise.

Software Quality Assurance Plan

Key to the success of school records application is the quality assurance plan. As such, the entire application, top to bottom, will be designed with quality up front. With having such high quality expectations, the following document will describe the cost of quality, threats to quality, how quality can be compromised, and the implementation of quality throughout all stages of the project life cycle.

Cost of Quality

The cost of quality are the expenditures that arise in support of quality. One expenditure is quality planning. From the start, a plan will be devised to integrate quality processes and procedures throughout each stage of the software development life cycle. This extensive planning phase will take roughly 120 combined staff hours to complete, which is both a monetary cost and a timeline cost. However, planning for quality will provide us with the best possible quality design, and a streamlined approach to handling post-deployment issues and requests.

Testing will be another cost of quality, as it will be integrated throughout the project. Not only will programmers be employing test driven development, but they will also be participating in partnered code reviews and weekly formal technical reviews. In addition, after each component iteration, we will deploy those components to non-technical testers to simulate actual users to help us stay aligned with user expectations. The costs will include time away from development and expenses for user/tester time.

Thirdly, training will be a cost of quality. All software developers who are not currently fluent in test driven development practices will be trained. It is expected that each developer would require approximately 8 hours of training on the topic. Another training cost is from the end-user perspective. They will need to be trained on how to use the software and what communication paths should be followed in the event of software issues. It is expected that each user will need to attend a two-day training, costing 16 hours per user.

Threats to Quality

There are many possible threats to quality, so it is important to identify and plan to prevent them. One of those threats is poor coding practices. Poor coding practices can include, but is not limited to, tight coupling, low cohesion, poor use of encapsulation, lack of notes, inconsistent formatting (such as indentation), and poor variable naming convention. These issues will be addressed in two ways. The first is code reviews, as mentioned above. Code reviews are expected to mitigate issues related to coupling, cohesion, encapsulation and overall object-oriented coding practices. The second is through defined company coding standards. We will train and provide reference materials on company coding expectations regarding spacing, variable naming conventions, notes, order of declarations and code blocks within classes/modules, etc.

Another threat to quality is poor communication from both the developer team and users. Poor communication can cause overlapping work, misconceptions of scope of work, improper component integrations, and unresolved issues. To prevent these issues, the developer team will establish an open flow of communication by meeting every morning to discuss progress, issues, and answer questions. And as mentioned above, users will be trained on proper communication paths for identifying software quality issues.

The final threat to quality is lack of testing. Lack of testing can leave existing bugs in the final software product and introduce new bugs when merging branches into the live system. For these reasons, each developer will be utilizing test driven development, participating in code reviews, employing automated testing. In addition, each pull request will require at least to project leaders to approve before committed to the master branch.

Implementation

Quality will be implemented in each stage of the software development process. From the beginning, quality will be integrated into the requirements gathering stages of the process. As requirements are gathered, they will be validated between the development team and all project stakeholders. This validation will take place for each requirement and will be repeated for any new requirements added later in the project.

In the software design stage, multiple stages of technical reviews will take place with the development team to ensure all system, software and database components are designed in a quality manner. The development team will create class diagrams, sequence diagrams, use-case diagrams, and activity diagrams to communicate design and scope in a visual manner. Stakeholders will also be brought in to validate scope and answer any potential questions that have arisen.

During implementation, one of the ways to ensure quality is through test driven development. Test driven development creates tests for functions before the function is written, requiring the programmer to design the function to pass the test. This ensures that every function and component is fully tested before integrated with the rest of the module/system. While this may sound like it will extend the amount of time needed for implementation, it vastly reduces the amount of debugging and ensures the highest quality product.

In addition, each software component will be designed with high cohesion and low coupling. Cohesion will ensure that each module, class and function serve a narrowly defined purpose. This makes design choices simpler and it makes correcting coding issues much easier. Low coupling means that there are minimal dependencies between software components. Having fewer dependencies allows the software to be flexible to changes, helping us stay on our timeline when challenges arise.

These testing practices will carry over into the integration and system testing phase of the project. Integration testing will be conducted to ensure all software components work together as a complete system and that requirements have been met. System testing will be conducted to ensure the program functions as expected with other hardware and software systems.

Finally, quality will be built into the operation and maintenance stage of the software lifecycle. Once the project is deployed, all stakeholders will review and document the project’s strengths and weaknesses to measure current quality and what could be improved. As issues are communicated to the development team and additional requirements are identified, they will go through each of the software development steps that were just covered. In addition, ongoing analysis of software/database performance will be conducted on a regular basis to help identify areas of improvement.

Transferable Skills Self Reflection

Throughout the Software Application Development degree, I have improved my understanding of software and how it is developed. I have gained knowledge and skills that will help me in my chosen career path. These knowledge and skills are considered transferable skills. The following will outline which skills I’ve gained, how I’ll incorporate them into my chosen career path, and my future goals.

The transferable skills I have gained are planning, critical thinking and communication. After every class and project, I have increasingly recognized the importance of designing applications and planning upfront. Every application I have written I have seen that dedicating more time to the design phase would have increased my productivity. Second, critical thinking has played a major role throughout my classes. I have found that no matter how complex a problem may appear, it can be broken down into smaller and smaller components, making it manageable. And lastly, I have gained the skill of communication. Communicating application design, costs, ideas, deadlines, and many other aspects is critical to being successful in an organization.

I will incorporate these skills into my chosen field of being a software developer. As I’ve experienced throughout these courses, I will be required to help plan projects, use critical thinking to solve problems, and communicate aspects of the project in a clear and concise manner. The software engineering classes and all language specific classes will assist me in being successful

Based on these transferable skills, my goals for the future are clear. I want to acquire a position in software development where I can take part in the development process, help problem solve, and express my ideas to others. I’m hoping to continue learning new technologies and provide the best solutions possible for my organization. Given my management experience and Business Management Bachelor’s Degree, I would eventually like to move into a software engineer or software management roll to help lead projects and manage teams.

In conclusion, I have gained the transferable skills of planning, critical thinking, and communication throughout this Software Application Development degree. I will utilize these skills moving forward as I look to acquire a software development position, and I’m looking forward to using them in all future endeavors.