Term Project

CS 151: Object-Oriented Design

Technical-Spec – Final

Prof: Ahmad Yazdankhah

Team Number	26
Software Name	StudentSphere

#	Name	Major	Accomplishments
1	Samriddhi Matharu	Data Science	Completed components 1. Introduction (objective, references, acronyms and abbreviations) and 2. Software Overview (functional requirements)
2	Jeremy Greatorex	Software Engineering	Team management and data model, 1.2 Reference additions, 1.3 Acronyms
3	Ivan Rivera	Computer Science	UML Class Diagrams

1 Introduction

1.1 Objective

This document provides the technical specification for **StudentSphere**, a software system designed to serve as a faculty knowledgebase for managing student information. The system allows faculty members to create, edit, filter, search, and delete student profiles while maintaining accurate records of programming skills, academic and work status, evaluations, and service eligibility flags. This specification outlines the system architecture, data model, and component interactions intended for use by university faculty stakeholders, software architects, and developers involved in maintaining or extending the application.

1.2 References

The following resources, tools, and materials were used throughout the design and development of StudentSphere:

- **Project Problem Statement File**
  Provided the functional requirements, expected features, and constraints for the software system.
- **Visual Studio Code**
  Used as the primary development environment for writing, testing, and organizing Java source code.
- **IntelliJ**
  Additional integrated development environment used for writing and running StudentSphere Java application.

© Ahmad Yazdankhah

- **JavaFX Documentation (OpenJFX)**
  Referenced for building the graphical user interface, including TableViews, forms, event handling, and navigation components.
- **Git & GitHub**
  Used for version control, team collaboration, feature branching, and maintaining a shared repository of the project source code.
- **Lucidchart**
  Used to design UML Class Diagrams and Sequence Diagrams included in Section 3 of this document.
- **OpenCSV Library Documentation**
  Used to understand how to read/write CSV files for persistence of students, comments, and programming languages.
- **AI-assisted Tools (used minimally):**
  - **ChatGPT:** Leveraged only for formatting support, debugging guidance, and improving clarity in code comments. Final design and implementation decisions were made solely by the development team.

1.3 Acronyms and Abbreviations

CSV	Comma-Separated Value
PL	Programming Language
DB	Data Base
SFS	Student Service Flags
CB	Check Box

2 Software Overview

2.1 Functional Requirements

- Data Definition
  - Programming Language List: List defined by faculty of programming languages students can be proficient in
    - Name: Text, Required
  - Student Profile: Describes student, academic and professional experience, and comments
    - Personal Information:
      - Full Name: Text, Required
      - Year: Select [Freshman, Sophomore, Junior, Senior, Graduate], Required
      - Current Job Status: [Employed, Unemployed]
      - Job Details: Text, Present if Current Job Status is true
    - Experience and Interests: Professional expertise of student
      - Programming Languages Known: [Programming Language List Defined]
      - Databases Known: Select [MySQL, Postgres, MongoDB], Required
      - Preferred Professional Role: Select [Front-End, Back-End, Full-Stack, Data, Other], Required
    - Faculty Evaluation:
      - Comments/Journal Entries: Paragraph Text Entry
    - Future Services Flags
      - Whitelist: Boolean, optional (Eligible for recommendations)
      - Blacklist: Boolean, optional (Not Eligible for recommendations)
- Data Entry and Manipulation
  - Edit and delete a student profile
  - Search for students by name or any other parameter in their profile (Case insensitive, multi parameter search via comma-separated format

    Search fields include:
    - Name
    - Academic status
    - Job Status
    - Current Job

        - Programming languages
        - Databases
        - Preferred role
        - Future service flags
    - View and Add new comments to an existing student's profile (with date auto stamped)
- Reports
    - List all blacklisted students
    - List all whitelisted students
    - List all students with a specific profile parameter
    - Individual student report containing all details

3 Detailed Design

3.1 Data Model

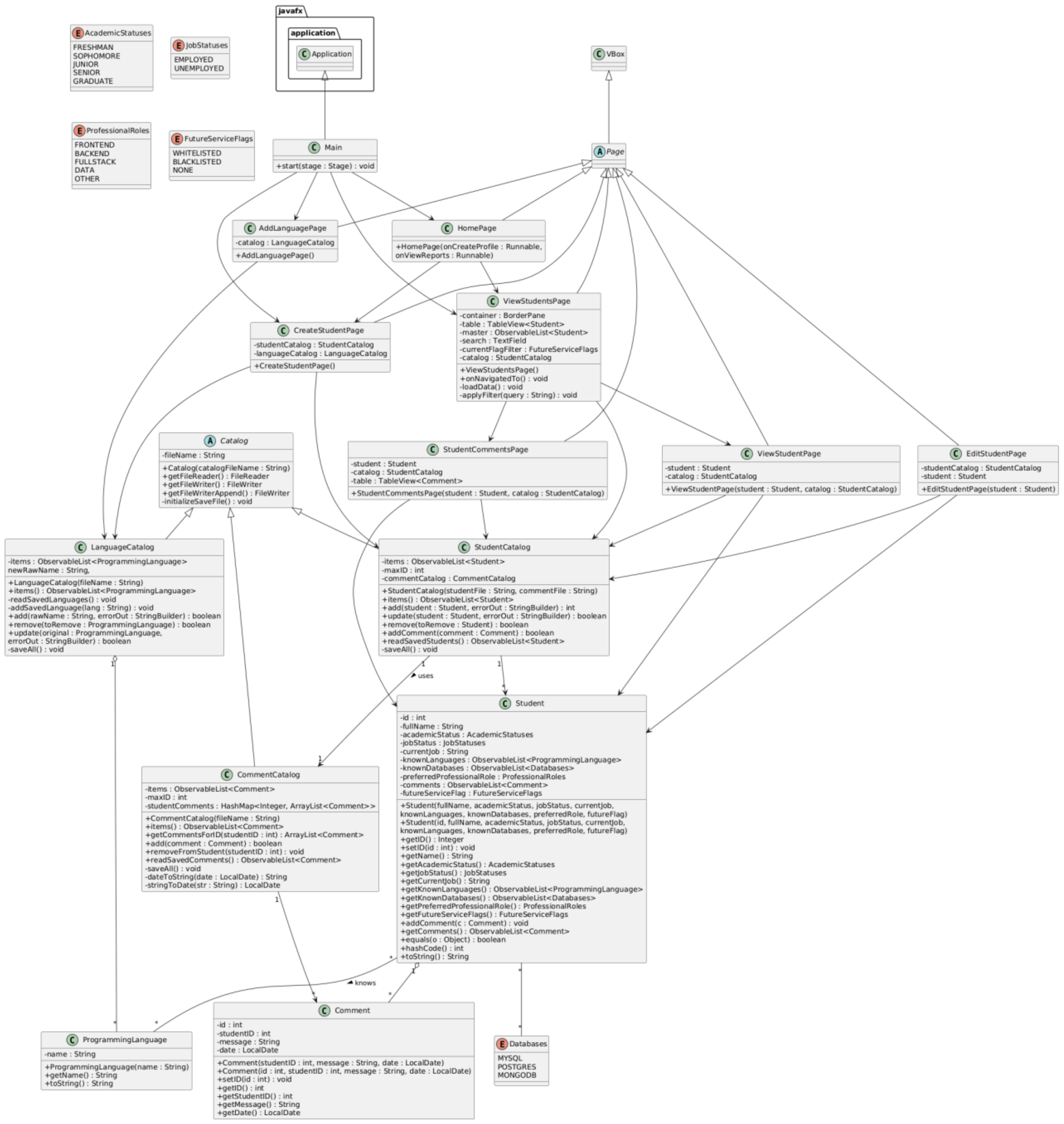
3.1.1 Data Storage Characteristics

Storage type	Flat Files (CSV)

3.1.2 Entities Names

1	Students.csv
2	ProgrammingLangauges.csv
3	Comments.csv

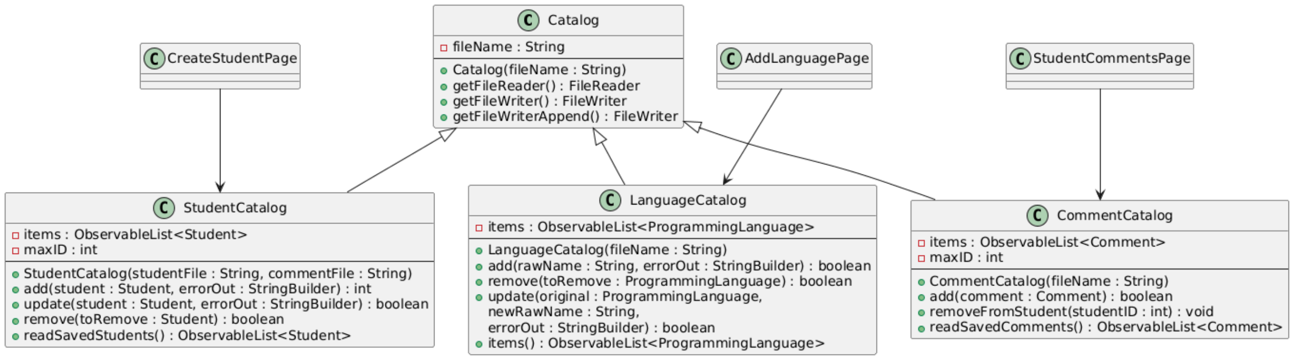
3.2 UML Class Diagrams



AcademicStatuses
FRESHMAN
SOPHOMORE
JUNIOR
SENIOR
GRADUATE
JobStatuses
EMPLOYED
UNEMPLOYED
javafx
application
Application
VBox
ProfessionalRoles
FRONTEND
BACKEND
FULLSTACK
DATA
OTHER
FutureServiceFlags
WHITELISTED
BLACKLISTED
NONE
Main
+start(stage : Stage) : void
Page
AddLanguagePage
-catalog : LanguageCatalog
+AddLanguagePage()
HomePage
+HomePage(onCreateProfile : Runnable, onViewReports : Runnable)
ViewStudentsPage
-container : BorderPane
-table : TableView<Student>
-master : ObservableList<Student>
-search : TextField
-currentFlagFilter : FutureServiceFlags
-catalog : StudentCatalog
+ViewStudentsPage()
+onNavigatedTo() : void
-loadData() : void
-applyFilter(query : String) : void
CreateStudentPage
-studentCatalog : StudentCatalog
-languageCatalog : LanguageCatalog
+CreateStudentPage()
Catalog
-fileName : String
+Catalog(catalogFileName : String)
+getFileReader() : FileReader
+getFileWriter() : FileWriter
+getFileWriterAppend() : FileWriter
-initializeSaveFile() : void
StudentCommentsPage
-student : Student
-catalog : StudentCatalog
-table : TableView<Comment>
+StudentCommentsPage(student : Student, catalog : StudentCatalog)
ViewStudentPage
-student : Student
-catalog : StudentCatalog
+ViewStudentPage(student : Student, catalog : StudentCatalog)
EditStudentPage
-studentCatalog : StudentCatalog
-student : Student
+EditStudentPage(student : Student)
LanguageCatalog
-items : ObservableList<ProgrammingLanguage>
newRawName : String,
+LanguageCatalog(fileName : String)
+items() : ObservableList<ProgrammingLanguage>
-readSavedLanguages() : void
-addSavedLanguage(lang : String) : void
+add(rawName : String, errorOut : StringBuilder) : boolean
+remove(toRemove : ProgrammingLanguage) : boolean
+update(original : ProgrammingLanguage, errorOut : StringBuilder) : boolean
-saveAll() : void
StudentCatalog
-items : ObservableList<Student>
-maxID : int
-commentCatalog : CommentCatalog
+StudentCatalog(studentFile : String, commentFile : String)
+items() : ObservableList<Student>
+add(student : Student, errorOut : StringBuilder) : int
+update(student : Student, errorOut : StringBuilder) : boolean
+remove(toRemove : Student) : boolean
+addComment(comment : Comment) : boolean
+readSavedStudents() : ObservableList<Student>
-saveAll() : void
1
uses
1
1
*
Student
-id : int
-fullName : String
-academicStatus : AcademicStatuses
-jobStatus : JobStatuses
-currentJob : String
-knownLanguages : ObservableList<ProgrammingLanguage>
-knownDatabases : ObservableList<Databases>
-preferredProfessionalRole : ProfessionalRoles
-comments : ObservableList<Comment>
-futureServiceFlag : FutureServiceFlags
+Student(fullName, academicStatus, jobStatus, currentJob, knownLanguages, knownDatabases, preferredRole, futureFlag)
+Student(id, fullName, academicStatus, jobStatus, currentJob, knownLanguages, knownDatabases, preferredRole, futureFlag)
+getID() : Integer
+setID(id : int) : void
+getName() : String
+getAcademicStatus() : AcademicStatuses
+getJobStatus() : JobStatuses
+getCurrentJob() : String
+getKnownLanguages() : ObservableList<ProgrammingLanguage>
+getKnownDatabases() : ObservableList<Databases>
+getPreferredProfessionalRole() : ProfessionalRoles
+getFutureServiceFlags() : FutureServiceFlags
+addComment(c : Comment) : void
+getComments() : ObservableList<Comment>
+equals(o : Object) : boolean
+hashCode() : int
+toString() : String
CommentCatalog
-items : ObservableList<Comment>
-maxID : int
-studentComments : HashMap<Integer, ArrayList<Comment>>
+CommentCatalog(fileName : String)
+items() : ObservableList<Comment>
+getCommentsForID(studentID : int) : ArrayList<Comment>
+add(comment : Comment) : boolean
+removeFromStudent(studentID : int) : void
+readSavedComments() : ObservableList<Comment>
-saveAll() : void
-dateToString(date : LocalDate) : String
-stringToDate(str : String) : LocalDate
1
1
*
knows
*
1
*
*
*
ProgrammingLanguage
-name : String
+ProgrammingLanguage(name : String)
+getName() : String
+toString() : String
Comment
-id : int
-studentID : int
-message : String
-date : LocalDate
+Comment(studentID : int, message : String, date : LocalDate)
+Comment(id : int, studentID : int, message : String, date : LocalDate)
+setID(id : int) : void
+getID() : int
+getStudentID() : int
+getMessage() : String
+getDate() : LocalDate
Databases
MYSQL
POSTGRES
MONGODB

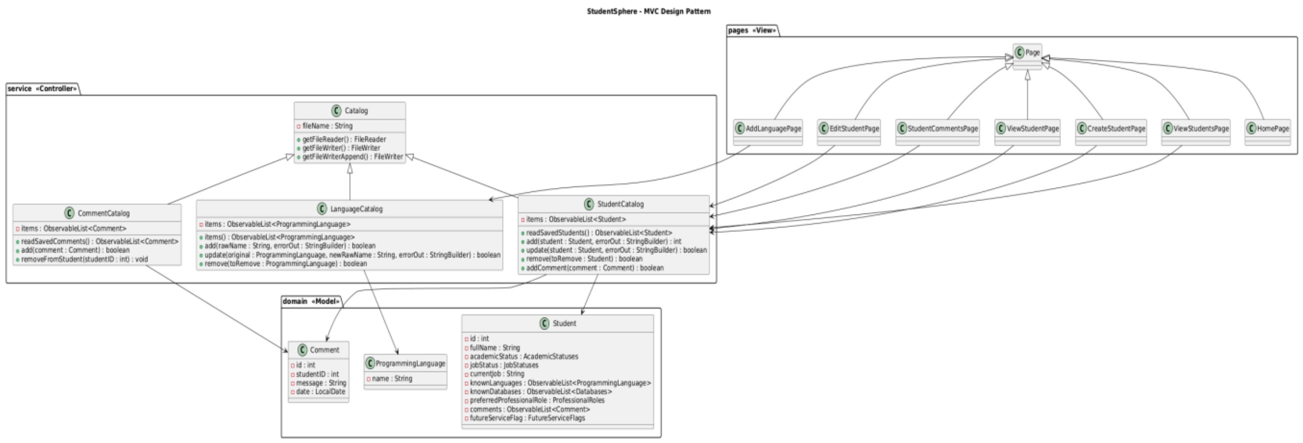
3.2.1 Polymorphism UML Class Diagram

Polymorphism is used via the Catalog base class and its subclasses StudentCatalog, LanguageCatalog, and CommentCatalog, which all share common file I/O behavior while specializing it for different domain entities.


Catalog
- fileName : String
+ Catalog(fileName : String)
+ getFileReader() : FileReader
+ getFileWriter() : FileWriter
+ getFileWriterAppend() : FileWriter
CreateStudentPage
AddLanguagePage
StudentCommentsPage
StudentCatalog
- items : ObservableList<Student>
- maxID : int
+ StudentCatalog(studentFile : String, commentFile : String)
+ add(student : Student, errorOut : StringBuilder) : int
+ update(student : Student, errorOut : StringBuilder) : boolean
+ remove(toRemove : Student) : boolean
+ readSavedStudents() : ObservableList<Student>
LanguageCatalog
- items : ObservableList<ProgrammingLanguage>
+ LanguageCatalog(fileName : String)
+ add(rawName : String, errorOut : StringBuilder) : boolean
+ remove(toRemove : ProgrammingLanguage) : boolean
+ update(original : ProgrammingLanguage, newRawName : String, errorOut : StringBuilder) : boolean
+ items() : ObservableList<ProgrammingLanguage>
CommentCatalog
- items : ObservableList<Comment>
- maxID : int
+ CommentCatalog(fileName : String)
+ add(comment : Comment) : boolean
+ removeFromStudent(studentID : int) : void
+ readSavedComments() : ObservableList<Comment>

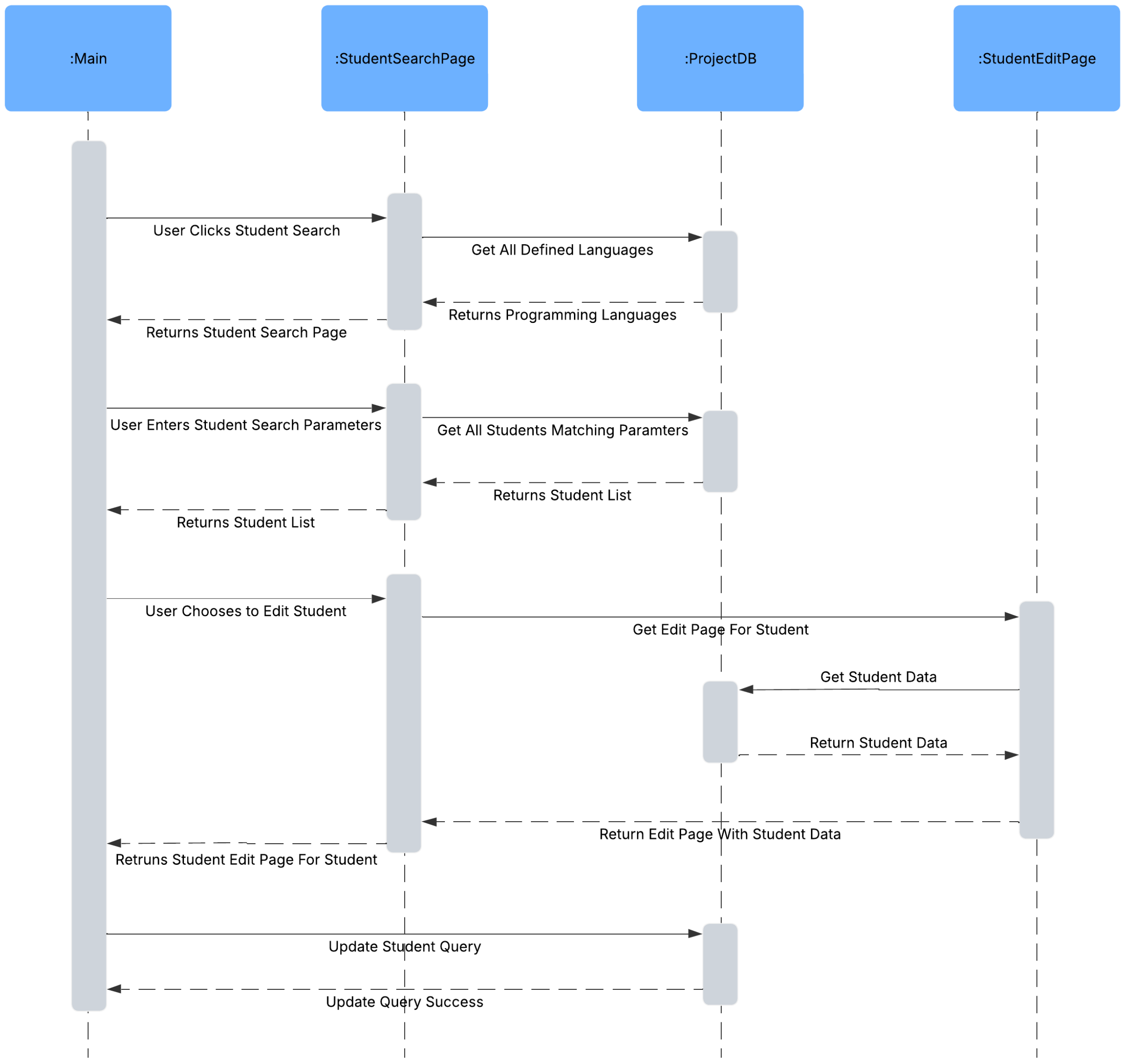
3.2.2 MVC Design Pattern UML Class Diagram

StudentSphere uses the Model–View–Controller (MVC) class design pattern: domain (Model), service (Controller), and pages (View), as shown in the diagram.


StudentSphere - MVC Design Pattern
pages «View»
Page
AddLanguagePage
EditStudentPage
StudentCommentsPage
ViewStudentPage
CreateStudentPage
ViewStudentsPage
HomePage
service «Controller»
Catalog
- fileName : String
+ getFileReader() : FileReader
+ getFileWriter() : FileWriter
+ getFileWriterAppend() : FileWriter
CommentCatalog
- items : ObservableList<Comment>
+ readSavedComments() : ObservableList<Comment>
+ add(comment : Comment) : boolean
+ removeFromStudent(studentID : int) : void
LanguageCatalog
- items : ObservableList<ProgrammingLanguage>
+ items() : ObservableList<ProgrammingLanguage>
+ add(rawName : String, errorOut : StringBuilder) : boolean
+ update(original : ProgrammingLanguage, newRawName : String, errorOut : StringBuilder) : boolean
+ remove(toRemove : ProgrammingLanguage) : boolean
StudentCatalog
- items : ObservableList<Student>
+ readSavedStudents() : ObservableList<Student>
+ add(student : Student, errorOut : StringBuilder) : int
+ update(student : Student, errorOut : StringBuilder) : boolean
+ remove(toRemove : Student) : boolean
+ addComment(comment : Comment) : boolean
domain «Model»
Comment
- id : int
- studentID : int
- message : String
- date : LocalDate
ProgrammingLanguage
- name : String
Student
- id : int
- fullName : String
- academicStatus : AcademicStatuses
- jobStatus : JobStatuses
- currentJob : String
- knownLanguages : ObservableList<ProgrammingLanguage>
- knownDatabases : ObservableList<Databases>
- preferredProfessionalRole : ProfessionalRoles
- comments : ObservableList<Comment>
- futureServiceFlag : FutureServiceFlags

3.3 UML Sequence Diagrams

Use Case: Updating a student profile


:Main
:StudentSearchPage
:ProjectDB
:StudentEditPage
User Clicks Student Search
Get All Defined Languages
Returns Programming Languages
Returns Student Search Page
User Enters Student Search Parameters
Get All Students Matching Paramters
Returns Student List
Returns Student List
User Chooses to Edit Student
Get Edit Page For Student
Get Student Data
Return Student Data
Return Edit Page With Student Data
Retruns Student Edit Page For Student
Update Student Query
Update Query Success