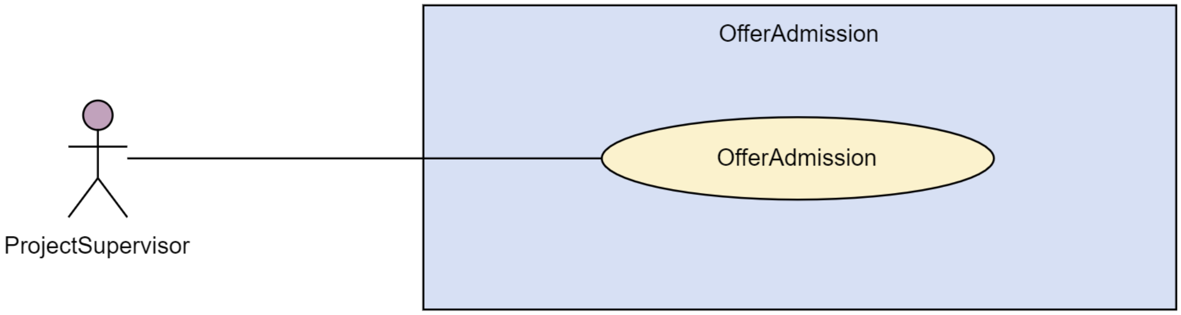
Assignment 2
Group 5

Contents

1. Use case textual description:

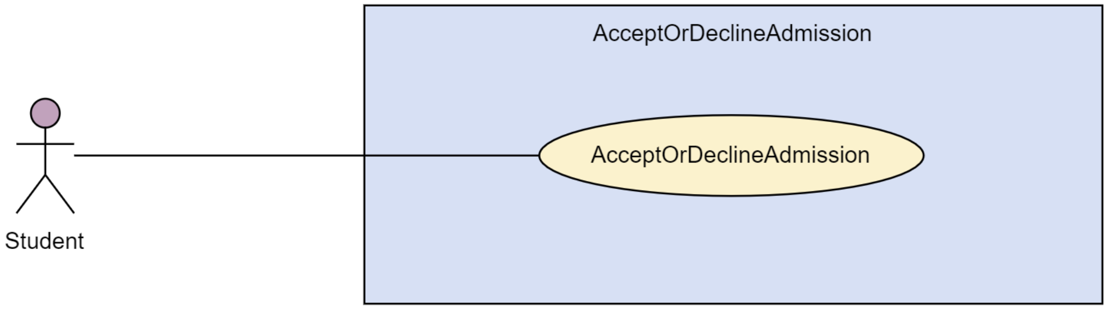
1.1 OfferAdmission



Textual description:

Use case name	***OfferAdmission***
Participating actors	Participants: ***ProjectSupervisor***
Flow of events	1) The ***ProjectSupervisor*** evaluate student profile. 2) Filter the list of selected students. 3) The ***ProjectSupervisor*** offer admission letter
Entry condition	• The ***User*** identified by ***GradRec***
Exit condition	-
Quality requirements	• The ***Users*** must receive an acknowledgment either in case of failed or successful action.

2.1 AcceptOrDeclineAdmission



Textual description:

Use case name	***AcceptOrDeclineAdmission***
Participating actors	Participants: ***Student***
Flow of events	1. The ***Student*** evaluate admission letter. 2. Accept admission a. In case of multiple offers ***Student*** accept one and decline others. b. Student enroll course and pay fees. 3. Decline admission c. Decline all admission letters. 4. In both case confirmation sends to ***ProjectSupervisor***
Entry condition	• The ***User*** identified by ***GradRec***
Exit condition	-
Quality requirements	• The ***Users*** must receive an acknowledgment either in case of failed or successful action.

2. Design Goals:

Security: Users of GredRec must be authenticated by the system before accessing functionality such as creating research project or creating student profile. Information like passwords will be stored in encrypted format. The system will provide a secure channel for the communication and ensure the privacy of the user's information.

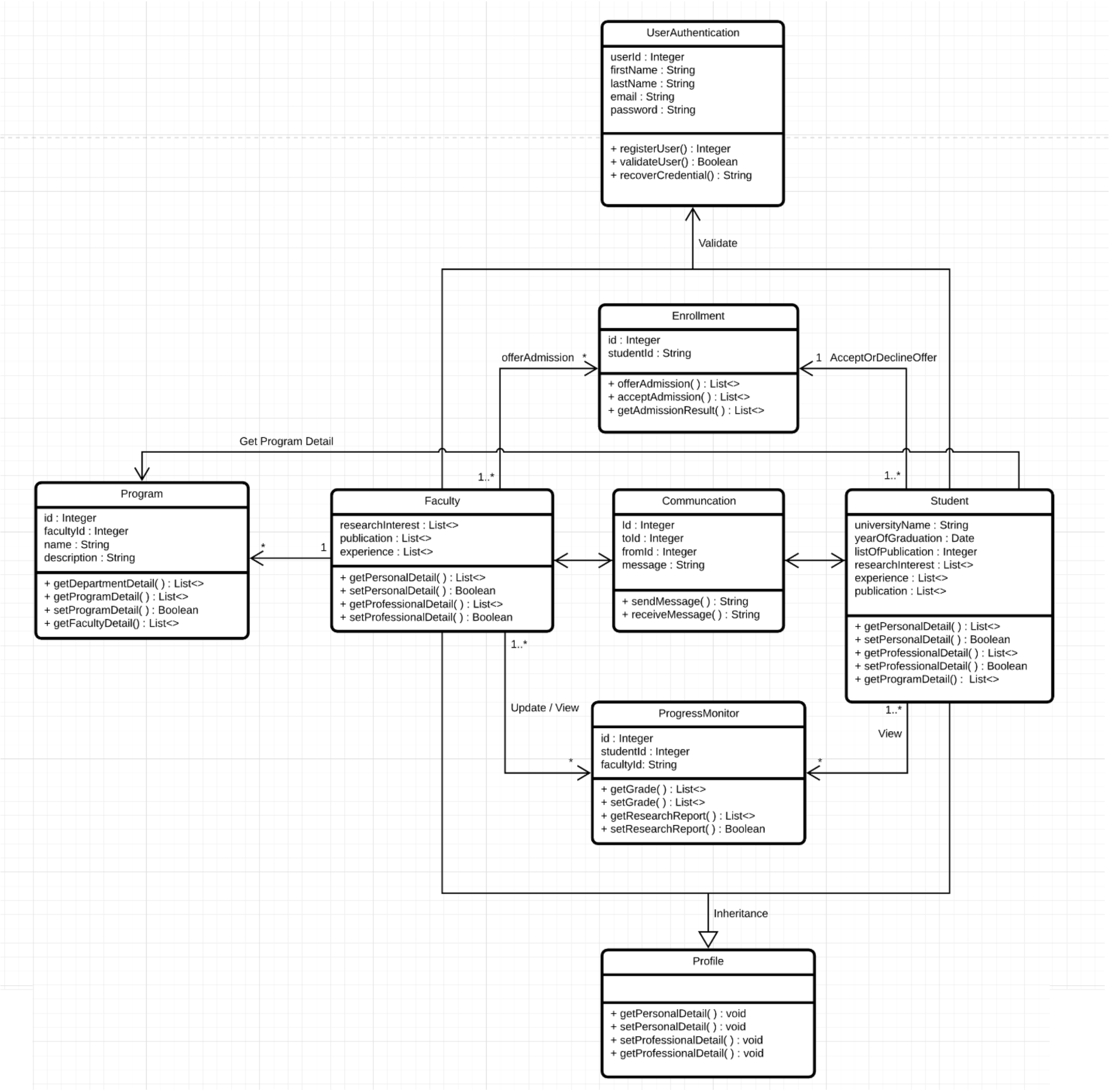
Reliability: GredRec system must be reliable in terms of persisting data on the permanent storage system (like Flat-File, MYSQL or MongoDB etc.) and be able to distinct different kinds of data. The student profile mapping algorithm should be consistent with the logic of creating the mapping between program and student based on the available information.

Usability: GredRec system should have a consistent look and feel across all user's interface. It should also be consistent with user privilege level, for example only the project supervisor can access the create program functionality, and all registered students should only have access to published programs. The system should allow easy start of multiple communication channels either with project supervisors or other students.

Modifiability: GredRec system will allow departments to edit their draft programs until they are published and/or before the last date of submission. Similarly, students will be allowed to edit their draft profile before the final submission and before the program deadline.

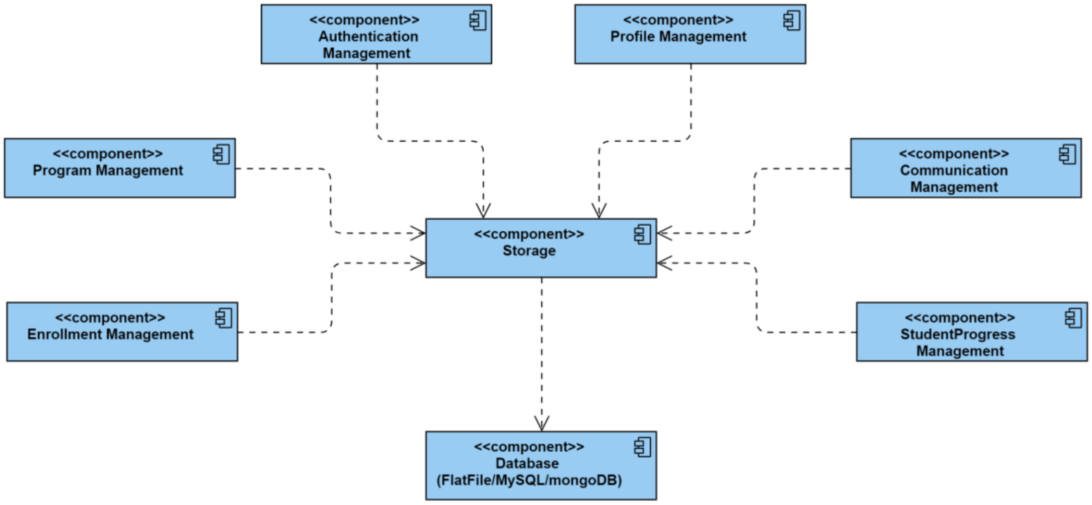
Fault tolerance: In case of error, the system should inform the user about the failure with a proper message without affecting other parts of the working system.

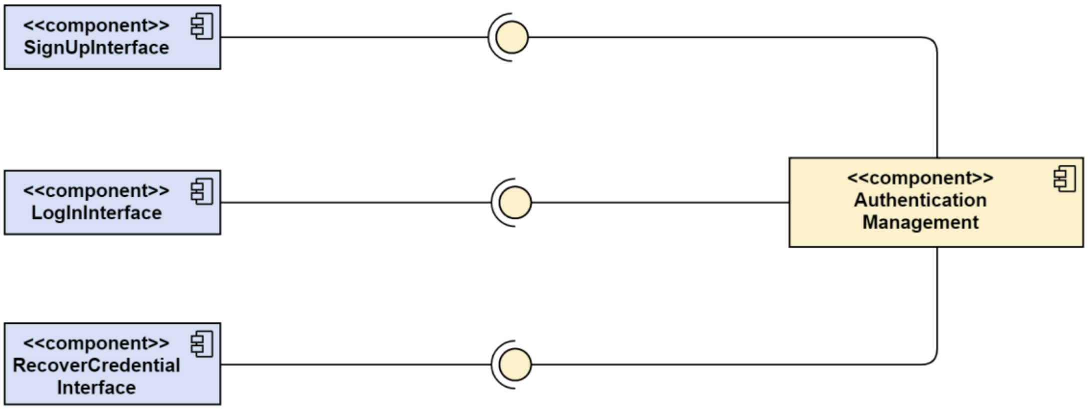
3. Object modelling:


UserAuthentication
userId : Integer
firstName : String
lastName : String
email : String
password : String
+ registerUser() : Integer
+ validateUser() : Boolean
+ recoverCredential() : String
Validate
Enrollment
id : Integer
studentId : String
+ offerAdmission( ) : List<>
+ acceptAdmission( ) : List<>
+ getAdmissionResult( ) : List<>
offerAdmission
*
1
AcceptOrDeclineOffer
Get Program Detail
1..*
1..*
Program
id : Integer
facultyId : Integer
name : String
description : String
+ getDepartmentDetail( ) : List<>
+ getProgramDetail( ) : List<>
+ setProgramDetail( ) : Boolean
+ getFacultyDetail() : List<>
*
1
Faculty
researchInterest : List<>
publication : List<>
experience : List<>
+ getPersonalDetail( ) : List<>
+ setPersonalDetail( ) : Boolean
+ getProfessionalDetail( ) : List<>
+ setProfessionalDetail( ) : Boolean
Communcation
Id : Integer
toId : Integer
fromId : Integer
message : String
+ sendMessage( ) : String
+ receiveMessage( ) : String
Student
universityName : String
yearOfGraduation : Date
listOfPublication : Integer
researchInterest : List<>
experience : List<>
publication : List<>
+ getPersonalDetail( ) : List<>
+ setPersonalDetail( ) : Boolean
+ getProfessionalDetail( ) : List<>
+ setProfessionalDetail( ) : Boolean
+ getProgramDetail() : List<>
1..*
Update / View
1..*
View
ProgressMonitor
id : Integer
studentId : Integer
facultyId: String
+ getGrade( ) : List<>
+ setGrade( ) : List<>
+ getResearchReport( ) : List<>
+ setResearchReport( ) : Boolean
*
*
Inheritance
Profile
+ getPersonalDetail( ) : void
+ setPersonalDetail( ) : void
+ setProfessionalDetail( ) : void
+ getProfessionalDetail( ) : void

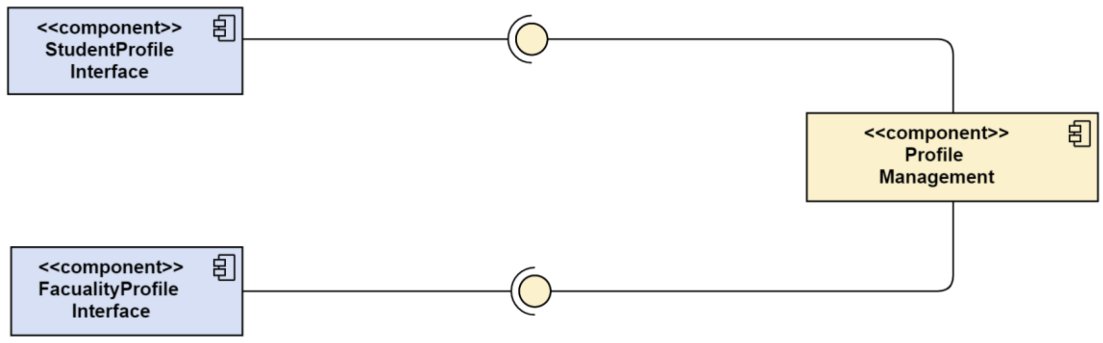
4. Decomposing of the system:

System Decomposition:

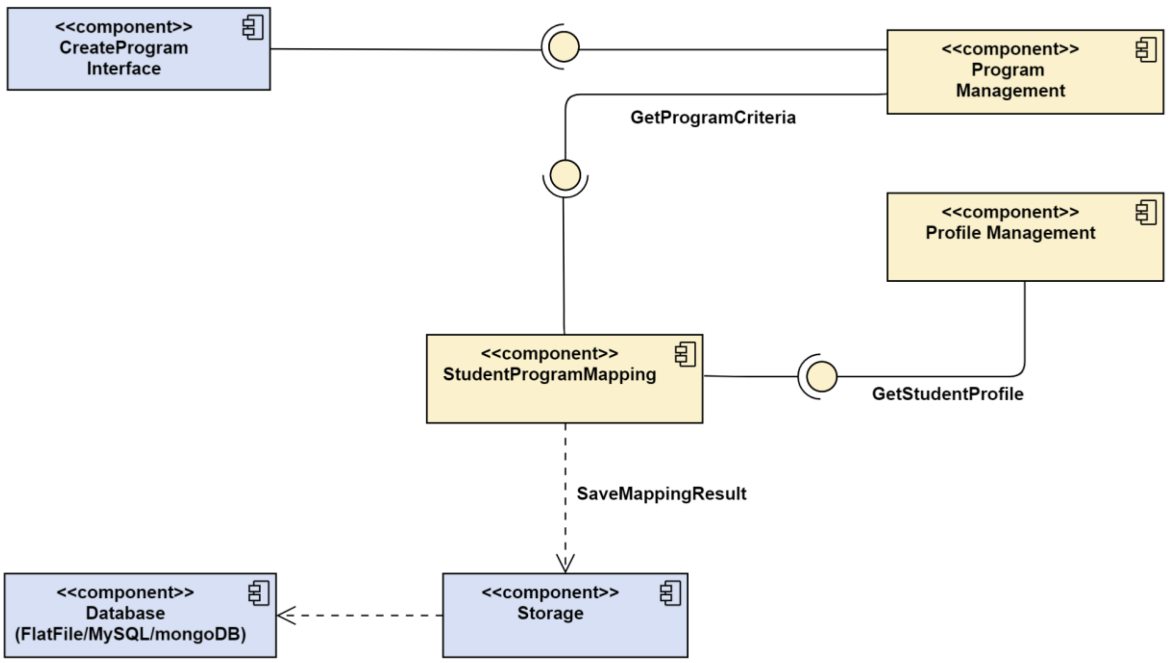

<<component>> Authentication Management
<<component>> Profile Management
<<component>> Program Management
<<component>> Communication Management
<<component>> Storage
<<component>> Enrollment Management
<<component>> StudentProgress Management
<<component>> Database (FlatFile/MySQL/mongoDB)


<<component>> SignUpInterface
<<component>> LogInInterface
<<component>> RecoverCredential Interface
<<component>> Authentication Management

SignUpInterface, LoginInInterface and RecoverCredentialInterface require services from Authentication Management to manager user authentication and recover credentials.



StudentProfileInterface and FacualityProfileInterface require services from Profile Management component create and manage student and factuality details.



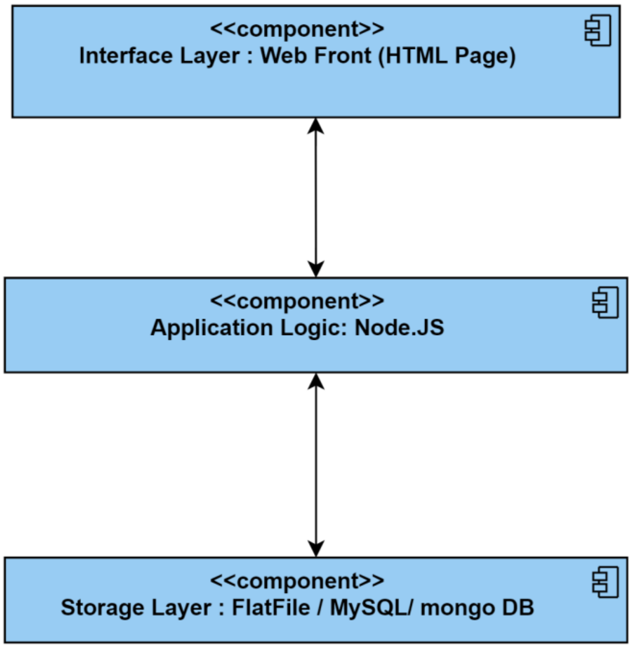
CreateProgramInterface require services from Program Management component to update or create new programs.

StudentProgramMapping component uses Program and Profile Management component to get program criteria and new student profile details to create mapping between student and program then save result in database.

5. Logical architecture:

Gradrec system implemented with three tire architecture.

- The **interface layer** includes all boundary objects that deal with the web pages (HTML page).
- The **application logic** layer includes all control and entity objects, realizing the processing, rule checking, and notification required by the application with the help Node.JS.
- The **storage layer** realizes the storage, retrieval, and query of persistent objects. In of these storage Flat File / MySQL/ mongo DB.


<<component>>
Interface Layer : Web Front (HTML Page)
<<component>>
Application Logic: Node.JS
<<component>>
Storage Layer : FlatFile / MySQL/ mongo DB

6. Git Hub And Website Link:

Web Site Link:

http://sc-5.cs.mun.ca/

Git Hub Link:

https://github.com/kaushlenderk/GraduateRecruitment