CS 2212
Course Management
Design Document
Group 3

Document Sign-Off

Name (Position)	Date
Claudiu Cretu	April 9th 2018
Fawaz Mahbouba	April 9th 2018
Damien Dorobek	April 9th 2018
Brayden Horth	April 9th 2018

Contents

1

Introduction

2 Overview

Text providing a roadmap to the sections and diagrams in your document

The logical architecture for this project has been provided to us. This means that the architecture section covers the architecture from the sample code. The UML diagrams consist of classes that we have either edited or created ourselves. In the UML section we have provided tables that contain any new methods that we have created or edited and a short description of each.

3

Major Design Decisions

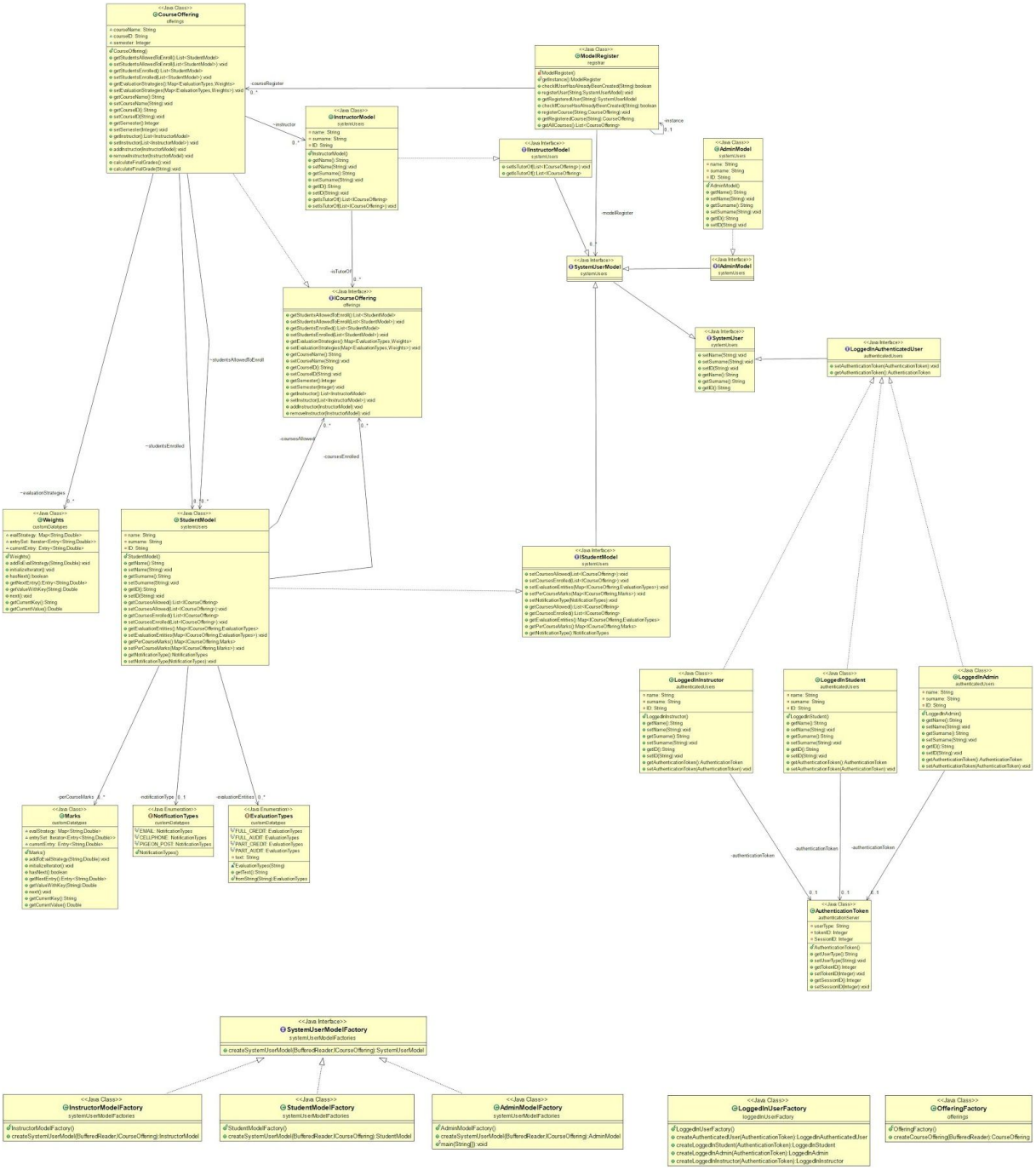
Text describing significant design choices, and modularization criteria.

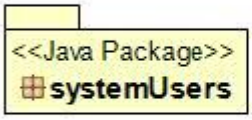
Our class that runs the system is the operating class. From this class the user can login and be authenticated. Once a user is authenticated, they will be directed towards methods that only users of their specific type can access. Users are stored in a csv file with their IDs and passwords. Everytime a user is created, they are added to this csv file.

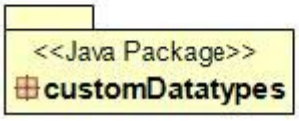
4 Architecture

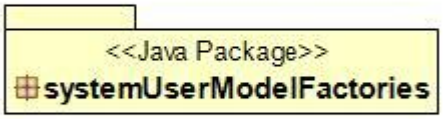
Component diagrams, and deployment diagrams in separate sections. If you need nested diagrams please use nesting levels. Include explanations on the functionality of each component. Comment of you are using any specific architectural style. Make sure you include component and deployment diagrams from the testing infrastructure as well.

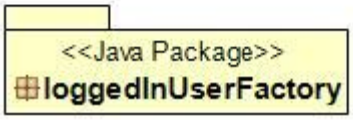
It was specified that the logical architecture has been provided to us for this project. Below is diagrams describing the logical architecture of the project.


CourseOffering
ModelRegister
InstructorModel
AdminModel
SystemUserModel
IAdminModel
ICourseOffering
SystemUser
LoggedInAuthenticatedUser
Weights
StudentModel
IStudentModel
LoggedInInstructor
LoggedInStudent
LoggedInAdmin
Marks
NotificationTypes
EvaluationTypes
AuthenticationToken
SystemUserModelFactory
InstructorModelFactory
StudentModelFactory
AdminModelFactory
LoggedInUserFactory
OfferingFactory









5 Detailed Class Diagrams

6 UML Class Diagrams

Detailed class diagrams for each class you modify or write for the extensions. You can separate the class diagrams per module they appear.Tables should also be included listing the methods of each class with a short description of what each method does. Please indicate if you a specific design pattern is used in your class diagrams.

Below are tables describing any methods that we created or edited in existing classes or any methods that we created in new classes.

AdminModel	Methods	Description
	services()	The services method will serve as the Admin's operations interface. Here the Admin can choose what operations he wants to

		perform.
	startSystem()	startSystem() will access the systemOperations.SystemState class in order to turn on the system. If the system is already on then an error is produced.
	stopSystem()	stopSystem() will access the systemOperations.SystemState class in order to turn off the system. If the system is already off then an error is produced.
	readCourseRecord()	readCourseRecord() will read any course text files that are available to read. It will create course and student objects and store them in the ModelRegister. This method prevents duplicated courses of being added.
	setPassword()	setter method to set the password of the Admin
	getPassword()	getter method to return the password of the Admin

StudentModel	Methods	Description
	enroll()	Allows a Student to enroll into a specified course if they are allowed to enroll.
	setNotificationStatus()	Allows a Student to set their notification status
	setNotificationPreference()	Lets a Student set their preference of notifications
	printRecord()	Print the record of a specified course
	services()	The services method will serve as the Student's

		operations interface. Here the Student can choose what operations he wants to perform.

InstructorModel	Methods	Description
	services()	This method presents the user with the operations they can perform as an Instructor.
	calculateFinalGrade()	This method calculates the final grade for a Student or all the Students in a course, depending on what the Instructor wants.
	addMark()	This method adds a mark for a Student enrolled in a course.
	modifyMark()	This method adds a mark for a Student enrolled in a course.
	printClassRecord()	This method prints the record for the class the user enters.
	sendStudentNotification()	This method sends a Student notification to the Student passed as a parameter.

CourseOffering	Methods	Description
	calculateFinalGrade	This method calculates final grades using the weights and marks utility classes. It was edited to return a final grade variable.

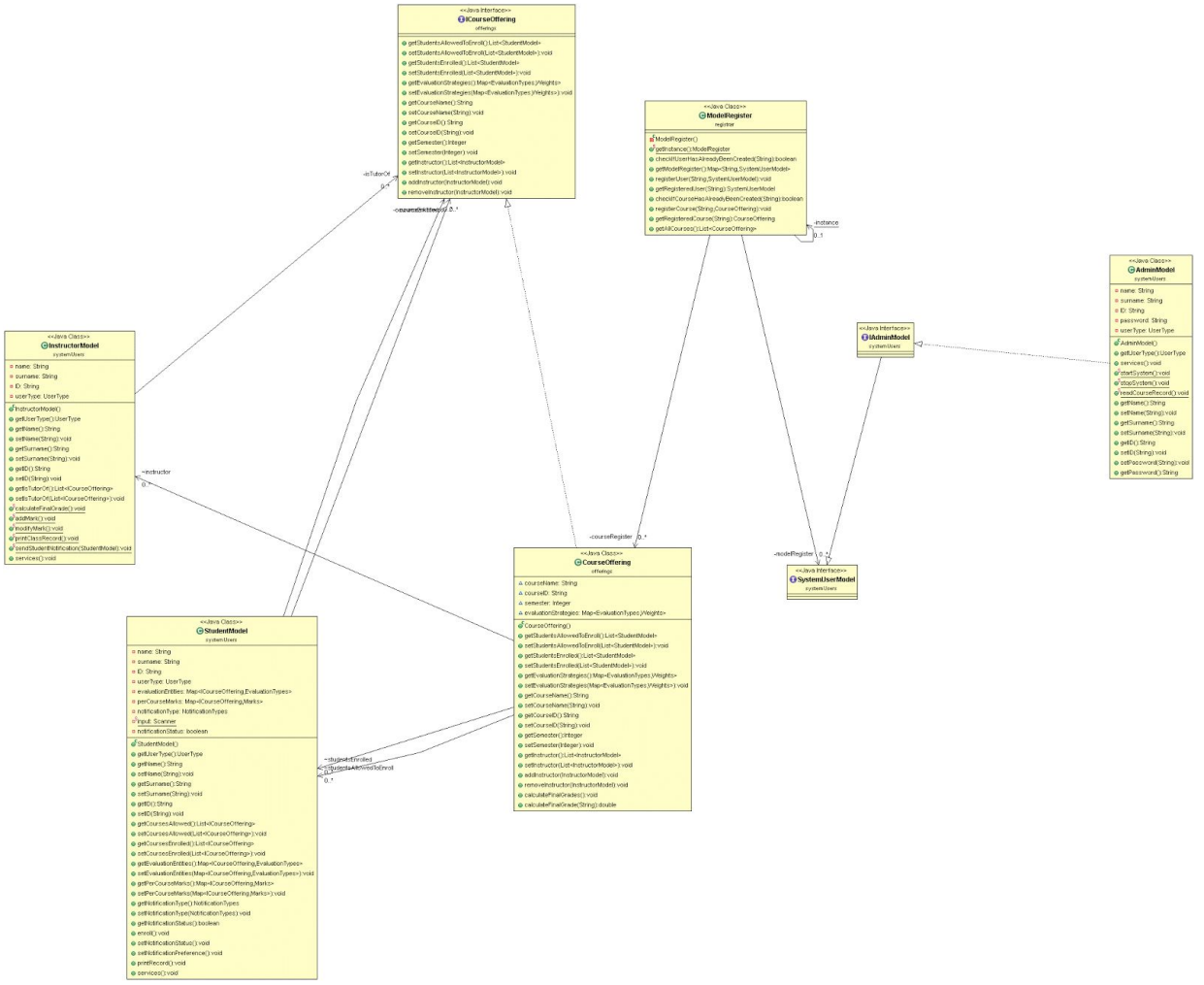
UserType	Methods	Description
	getText()	Return the type of the user
	fromString()	Searches for a user type

SendEmail	Methods	Description
	sendFromGmail()	Send email with information to the user.

SystemState	Methods	Description
	setSystemOn()	This method will set the instance variable *systemOn* in the method to the state that is passed to it.
	isOn()	This method checks whether or not the system is on.
	systemStateAsString()	This method returns a message of whether or not the system is on.

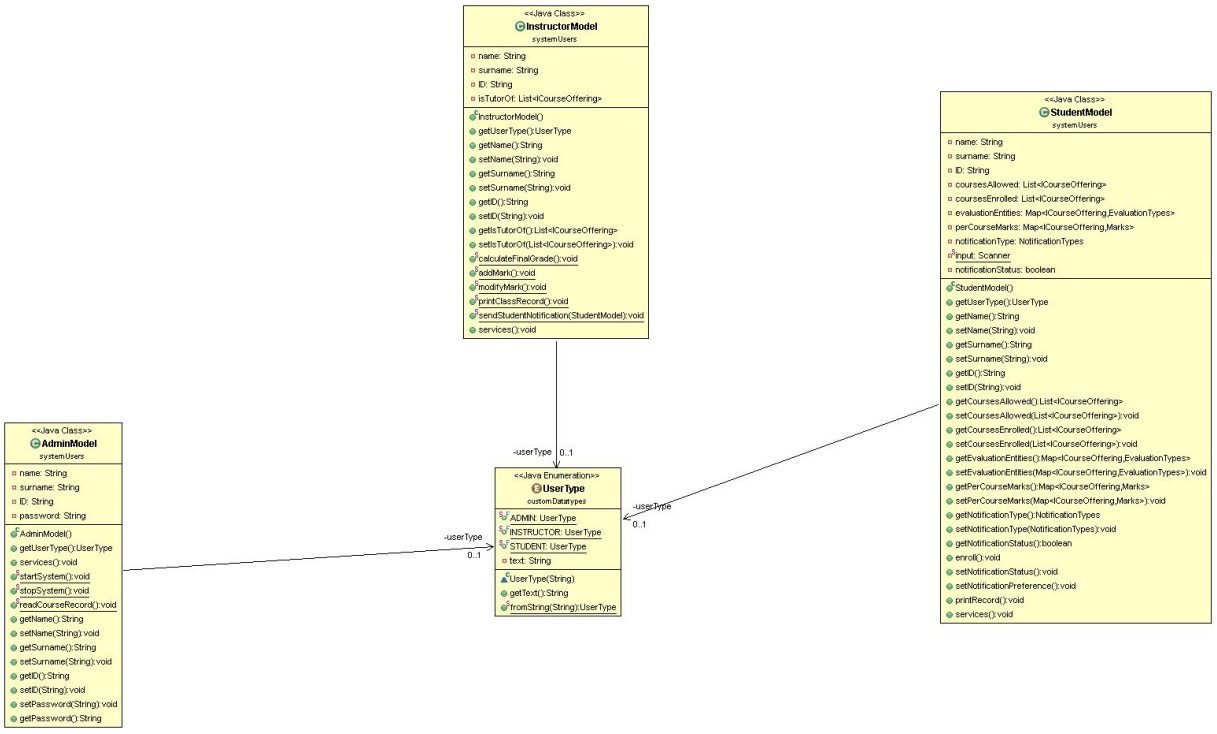
System Users Diagram

The following diagram shows the UML class diagrams of the updated system users and how they interact with each other..



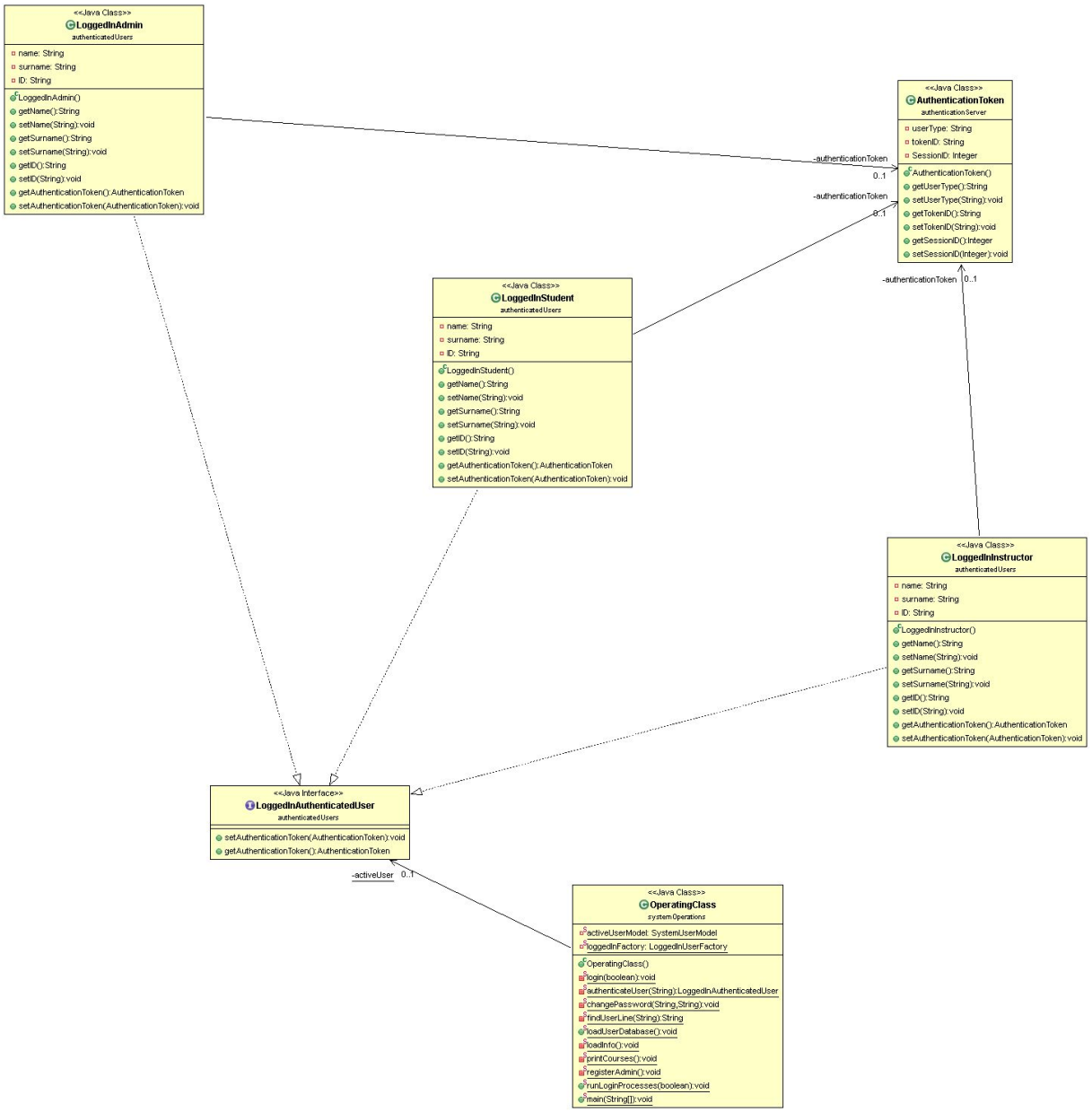
User Type Diagram

The following diagram shows the UML class diagrams of the system users and how they interact with the UserType class.



Operating Class Diagram

The following diagram shows the UML class diagrams for how the OperatingClass interacts with the rest of the system.


<<Java Class>>
LoggedInAdmin
authenticatedUsers
name: String
surname: String
ID: String
LoggedInAdmin()
getName():String
setName(String):void
getSurname():String
setSurname(String):void
getID():String
setID(String):void
getAuthenticationToken():AuthenticationToken
setAuthenticationToken(AuthenticationToken):void
<<Java Class>>
AuthenticationToken
authenticationServer
userType: String
tokenID: String
SessionID: Integer
AuthenticationToken()
getUserType():String
setUserType(String):void
getTokenID():String
setTokenID(String):void
getSessionID():Integer
setSessionID(Integer):void
-authenticationToken
0..1
-authenticationToken
0..1
-authenticationToken
0..1
<<Java Class>>
LoggedInStudent
authenticatedUsers
name: String
surname: String
ID: String
LoggedInStudent()
getName():String
setName(String):void
getSurname():String
setSurname(String):void
getID():String
setID(String):void
getAuthenticationToken():AuthenticationToken
setAuthenticationToken(AuthenticationToken):void
<<Java Class>>
LoggedInInstructor
authenticatedUsers
name: String
surname: String
ID: String
LoggedInInstructor()
getName():String
setName(String):void
getSurname():String
setSurname(String):void
getID():String
setID(String):void
getAuthenticationToken():AuthenticationToken
setAuthenticationToken(AuthenticationToken):void
<<Java Interface>>
LoggedInAuthenticatedUser
authenticatedUsers
setAuthenticationToken(AuthenticationToken):void
getAuthenticationToken():AuthenticationToken
-activeUser
0..1
<<Java Class>>
OperatingClass
systemOperations
activeUserModel: SystemUserModel
loggedInFactory: LoggedInUserFactory
OperatingClass()
login(boolean):void
authenticateUser(String):LoggedInAuthenticatedUser
changePassword(String,String):void
findUserLine(String):String
loadUserDatabase():void
loadInfo():void
printCourses():void
registerAdmin():void
runLoginProcesses(boolean):void
main(String[]):void