UML Class Diagram

UserFactory
- instance: UserFactory

- getInstance(): UserFactory
- createUser(String): User

User
- + <<property>> registrationNumber: RegNum
- + <<property>> password: String
- + <<property>> name: String
- + <<property>> surname: String
- + <<property>> email: Email

Email
- emailAddress: String

+ isValid(): boolean
+ getEmailAddress(): String
+ setEmailAddress(String): void

RegNum
- type: char
- value: Integer

+ getNextValueID(char): int
+ getRegNumString(): String

Professor
- courseList: List<Course>

+ createProject(): void
+ editProject(): void
+ deleteProject(): void
+ createTeam(): void
+ editTeam(): void
+ deleteTeam(): void
+ gradeProject(): void
+ getCourseList(): List<Course>
+ setCourseList(List<Course>): void

Admin
+ createUser(): boolean
+ editUser(): boolean
+ deleteUser(): boolean
+ CreateCourse(): boolean
+ EditCourse(): boolean
+ deleteCourse(): boolean

Student
+ uploadProjectFile(): void
+ viewCourseGrades(): void

Course
- courseID: String
- courseName: String
- professor: Professor
- projectList: List<Project>
- teamList: List<Team>

+ getCourseID(): String
+ getCourseName(): String
+ getProfessor(): Professor
+ getProjectList(): List<Project>
+ getTeamList(): List<Team>
+ setCourseID(String): void
+ setCourseName(String): void
+ setProfessor(Professor): void
+ setProjectList(List<Project>): void
+ setTeamList(List<Team>): void

Team
- teamID: String
- studentList: List<Student>
- projectAssignments: Dictionary<Project, ProjectFile>

+ getTeamID(): String
+ getStudentList(): List<Student>
+ getProjectAssignments(): Dictionary<Project, ProjectFile>
+ setTeamID(String): void
+ setStudentList(List<Student>): void
+ setProjectAssignments(Dictionary<Project, ProjectFile>): void

Database
- connectionString: String

+ CreateTeam(string, List<string>): void
+ GetIds(string): List<int>
+ ValidateCredentials(string, string): string
+ GetUser(string, string): User
+ InsertUser(string, string, string, string, Email): void
+ DeleteUser(string,User): void
+ DeleteStudentFromAllTeams(Student): void
+ DeleteStudentFromTeam(Student, Team): void
+ DeleteProfessorFromProfessorCourses(Professor): void
+ GetProject(string): Project
+ GetProjectForCourse(string): List<Project>
+ InsertProject(string, string, int): void
+ DeleteProject(string): void
+ GradeProject(ProjectFile, int): void
+ GetCoursesForProf(string): List<Courses>
+ InsertStudentToTeam(Course, Student, Team): void
+ UploadProject(string, byte[], string, string): void
+ InsertTeam(string): void
+ GetTeams(string): List<Team>
+ GetTeamsProjectFiles(string): Dictionary<Project, ProjectFile>
+ GetFileDetails(string, string): ProjectFile
+ GetGrade(string, string): int
+ getStudentsOfTeam(string): List<student>
+ EditTeam(string, string, List<string>): void
+ DeleteTeam(string): void
+ getAllCourses(): Dictionary<string, string>
+ DownloadProject(string, string): void
+ EditProject(string, string, string, int, DateTime): void
+ GetTeamOfStudent(Student, string): Team
+ isStudentInTeamAlready(string, string): bool

FormFactoryProducer
+ getFactory(String): AbstractFormFactory

AbstractFormFactory
+ createForm(String): Form

UserFormFactory
+ createForm(String): Form

UserForm
<<Property>> currentUser: User

- logout(): void

Project
- projectID: String
- name: String
- description: String
- maxGrade: double
- due_date: Date

+ getprojectID(): String
+ getname(): String
+ getdescription(): String
+ getmaxGrade(): double
+ getDueDate(): Date
+ setprojectID(String): void
+ setname(String): void
+ setdescription(String): void
+ setmaxGrade(double): void
+ setDueDate(Date): void

ProjectFile
- projectFileID: String
- file: byte[]
- grade: int
- uploadDate: Date

+ getFile(): byte[]
+ getGrade(): int
+ getUploadDate(): Date
+ getProjectFileID(): String
+ setProjectFileID(String): void
+ setFile(byte[]): void
+ setGrade(int): void
+ setUploadDate(Date): void

Form
...
...

C# Form Class
Namespace: System.Windows.Forms

FormLogin
- abstractFormFactory: AbstractFormFactory

- login(): void

FormStudent
~ currentUser

- logout(): void

FormAdmin
~ currentUser

- logout(): void

FormProfessor
~ currentUser

- logout(): void

Relationships and labels: creates, uses, Creates, 1, 0..1, 0..*, 1..*, 0..Project Count