CS321 Team Project

The goal of this project is to go through all aspects of the software development life cycle in a small team. This will give you real experience in each stage from problem formation, requirements gathering and analysis, design, implementation (somewhat) and testing. You will use UML diagrams, text descriptions, and scheduling tools to create your deliverables.

You are starting a new small software company. This is your first project. You want to create something that people will use. You are the software team. You should assume other teams exist in your company (marketing sales, tech support, etc...). These other teams will be stakeholders (people who have an interest in what you're doing). NOTE: These are virtual people, you have to think for them in this project, or you can ask the professor or TA to answer questions to these people.

You will be required to extend an existing piece of non-trivial software for this project, the CS321 Quiz Game. Below is a description of the system. Beware that this description was provided by the customer; it might have ambiguities, inconsistencies, or even missing information!

The Quiz Game system will allow students and teachers to practice taking quizzes for their class. Quizzes are composed of one or more questions, which can be of three different types; short answer, multiple choice, and coding. Any user can take any of the existing quizzes, and the most recent score will be stored on an external grade server. To access the system, each user will have been provided with a username and password from their administrator (these are generally handed out on pieces in paper during lab sections).

Users will spend most of their time taking quizzes. To do so, they must first login with their username and password. Once logged in, they can choose to take an existing quiz, create a question to add to a quiz, or view their grades.

If they choose to take a quiz, they will go through all of the questions in order, answering any that they have not correctly answered on this attempt, or quitting any time. The quiz will loop until they either answer all questions correctly, or quit, at which point the user is presented with their score. This score is recorded on the external grade server within 60 seconds of completing the quiz. While short answer and multiple choice questions will be easily compared simply by diff-ing the expected answer with what the student answers, for a coding question there may be multiple correct implementations. For this reason, for such questions, the users' code will be compiled, and then run using the sample input, and compared to the expected output for correctness.

If there are no existing quizzes to choose from, the user can create a question. Creating a question happens by specifying the question itself, and then choosing a type of question. If it is a simple short answer question, the user provides the answer; if it is a multiple choice question, they provide a list of options and then the correct answer; and if it is a coding question, they provided a sample input, sample output, and the correct answer. Then, the user will specify to which quiz the question should belong, or specify a new quiz by giving the quiz a name.

If the user is not creating a question or taking a quiz, they can view their grades on all quizzes so far.

The grade server stores all quizzes ever taken by the user, but this system will only display the most recent grade for any particular quiz. Besides students, the system will also recognize the role of a teacher; when a teacher chooses to view grades, they will be shown all newest quiz grades for all students. Occasionally, teachers will want to manually change a grade; they and only they will be allowed to do so when viewing all student grades by choosing the student and quiz they want to update, and then entering the new grade. This new grade will then be stored on the external grade server.

The system shall store the users and quizzes locally in a database, so that the system can be periodically taken offline if necessary.

The system shall connect with the external grade server each time the application is run. This external grade server will periodically log user activity. Once the user finishes the interaction with the Quiz Game, the external grade server will be notified.

Finally, the system will email all users their most recent quiz grades on a weekly basis.

You will also be required to extend this system with at least one non-trivial feature. For example, the current system functions as a command-line tool; it might be nice to convert it to a GUI if you would be interested in learning Java Swing. Or, you could add an extension such as the ability to keep track of various user statistics, develop new types of quizzes, or make an interface to store more information on the grade server, etc.

After each deliverable, a solution will be provided on Piazza, as all students are implementing the same system (with the exception of the additional feature). A use case diagram will yield a class diagram, and eventually the professor will post a Java implementation of the system consistent with all solutions to all deliverables. You should use that implementation to add an additional feature to the product, and assume its API for writing unit tests. Be careful however; when inheriting other people's code, or open source software, it may contain bugs! You'll want to test the software well before adding additional features! You do not have to fix any bugs you find, unless it interferes with your feature.

The project deliverables are detailed below. The first one will involve selecting a Team Leader. The Team Leader will remain consistent throughout the semester, and will be responsible for overseeing a Team Manager, which will be a different person for every deliverable. Each deliverable should be submitted as a PDF unless noted otherwise.

Comment [GG1]: Each writing assignment includes a description, as well as rationales, strategies, or suggestions for writing in this field/genre.

	During this stage you will form into groups of 4-5 students. You may choose your own teams, however they may be adjusted if necessary to have equal numbers of students in all teams. If you cannot find a team, please see me and you will be assigned to a team. One challenge of software engineering (and life in general) is making decisions based on limited information. You need to choose a team, but you do not have enough information about the other students in the class to do this well, but you need to do it. Deliverable: A short written document including:

Team Formation
14 points
(due 9/7)

- Your team (company) name, your team members' names and emails - 3 pts
- At least one time during the week you can meet in person if needed (all team members available) - 1 pt
- What additional feature or non-trivial functionality your team will provide for the Quiz Game. Later in the semester you will be given the source code of the current version of the game; it is run through the command line and written in Java - 5 pts
- Describe how the team leader was chosen, and why your team believes that using this method will result in the best leader for this team project. This description should be no less than five sentences in length - 5 pts

Turn this deliverable in through Piazza, under the team_formation folder. You may make a public or private post. In the future, all deliverables will be submitted through svn.

Use Cases
42 points

You and your team will create at least seven use cases for the Quiz Game project. You will complete the following steps:

1. The Team Leader will select a Team Manager for the project. The Team Manager will create a folder on svn called `Use_Cases`. Within that, the Team Manager will create the subfolders `Individual` and `Team`.
2. Each group member will individually create a Use Case Diagram from the description of the Quiz Game problem above. See the grading rubric below. Each team member will submit their diagram though svn to the `Use_Cases/Individual` folder (their name should be in the filename).
3. The team will meet and/or discuss the individual use case diagrams and arrive at a consolidated diagram, which they will submit to the `Use_Cases/Team` folder.
4. The Team Manager will assign each group member 1-2 use cases to create use case descriptions for (some use cases are longer than others). The Team Manager will place a template/blank file in the `Use_Cases/Team` folder that initially just has the finalized use case diagram inside of it. Each member will create the use case descriptions, and add them to the shared file mentioned, checking for consistency/compatibility as they update the file with their changes. See the grading rubric below.

Your use cases must include your additional feature. If your additional feature is a GUI, that is not a use case - just include that as a note.

Grading Rubric for Use Case Diagram: (18 points)

Comment [GG2]: Assignment prompts describe grading criteria.

(due 9/18)

At least seven use cases	7 points
At least one includes relation, correctly implemented	2 points
At least one extends relation, correctly implemented	2 points
At least one generalization relation, correctly implemented	2 points
At least two actors, correctly implemented	2 points
At least one external system, correctly implemented	2 points
Includes your additional feature	1 point

Grading Rubric for Use Case Descriptions: (24 points)

Each team member must put their name next to the use case descriptions they wrote, and must write at least 250 words	loss of all points if missing
Descriptions avoid using language that includes implementation details	7 points (one each use case)
Descriptions have all described functionality for basic and alternative flows	7 points (one each use case)
Descriptions correctly use includes and/or preconditions	7 points (one each use case)
Descriptions complete all fields in the template	7 points (one each use case)

You and your team will create at least seven functional, and three non-functional requirements for the Quiz Game project. You will complete the following steps:

1. The Team Leader will choose a new Team Manager for this deliverable. The Team Manager will create a folder on svn called `Requirements`. Within that, s/he will create the subfolders `Individual` and `Team`.
2. Each group member will individually create 7 functional requirements and two non-functional requirements. See the grading rubric below. Each team member will submit their chart though svn to the `Requirements/Individual` folder (their name should be in the filename).
3. The Team Manager will place a template/blank file in the `Requirements/Team` folder that initially just has a table with columns {unique ID, priority, type (functional or non-functional), source, contained in use case(s), and description. The team will meet and/or

Requirements
19 points
(due 9/23)

discuss the individual requirements, and arrive at a consolidated set, potentially expanding it to include all system functionality, which they will submit to the `Requirements/Team` folder. See the grading rubric below.

Your requirements must include your additional feature.
Grading Rubric for Requirements: (19 points)

Functional requirements describe all of the functionality of the system.	10 points
At least two plausible non-functional requirements are included	4 points
All requirements have a unique ID	1 point
All requirements have a priority	1 point
All requirements have a source (the customer)	1 point
All requirements are labeled as functional or non-functional	1 point
All requirements map into appropriate use cases	1 point

Here are some lessons learned about requirements

Class Diagram
27 points
(due 9/30)

You and your team will use your existing deliverables to decide upon a set of classes for the system. Your team will complete the following:

1. The Team Leader will select a new Team Manager for this deliverable. The Team Manager will create a folder on svn called `Class_diagram`. Within that, s/he will create the subfolders `Individual` and `Team`.
2. Each group member will individually create a class diagram. See the grading rubric below. Each team member will submit their diagram though svn to the `Class_diagram/Individual` folder (their name should be in the filename).
3. The Team Manager will place a template/blank file in the `Class_diagram/Team`. The team will meet and/or discuss the class diagrams, to arrive at a consolidated class diagram which they will place in this folder. See the grading rubric below.

Your class diagram must include your additional feature.
Grading Rubric for Class Diagram: (27 points)

The class diagram accounts for all of the canonical requirements.	13 points
At least one dependency is correctly implemented	2 points
At least one aggregation is correctly implemented	2 points
At least one generalization is correctly implemented	2 points
All associations have multiplicities	2 points

All associations have navigations	2 points
All attributes have types	2 points
All methods have argument and return types	2 points

Dynamic Analysis Diagrams
26 points
(due 10/9)

You and your team will use your existing deliverables to create dynamic modeling diagrams for your system. Your team will complete the following:

1. The Team Leader will choose a new Team Manager. The Team Manager will create a folder on svn called `Dynamic_models`.
2. The Team Manager will assign each group member to create one of {swimlane diagram for a use case, sequence diagram for a use case, state diagram for a use case, state diagram for the system, (and another sequence diagram if the team has five members)}. All diagrams must be for different use cases, but be careful which use cases you select - some use cases will not meet all of the requirements below. See the grading rubric below. Each team member will submit their diagram though svn to the `Dynamic_models` folder (their name and the type of diagram should be in the filename).
3. Once all diagrams have been completed, each team member is responsible for reviewing the other team members' diagrams for correctness.

One of the diagrams must document/use your additional feature.
Grading Rubric for Dynamic Analysis Diagram: (26 points)

The swimlane diagram contains at least two actors.	2 points
The swimlane diagram contains a start and end state.	2 points
The swimlane diagram contains at least one includes of another (abstract) swimlane diagram.	2 points
The swimlane diagram contains at least one fork and join.	2 points
The swimlane diagram contains at least one decision, correctly labeled.	2 points
The sequence diagram(s) contains at least two classes.	2 points
The sequence diagram(s) contains at least one option or loop.	2 points
The sequence diagram(s) contains at least one synchronous call.	2 points
The sequence diagram(s) contains at least one asynchronous call.	2 points

The sequence diagram(s) has correct labels on all messages.	2 points
The state diagrams contain a label on each trigger.	2 points
The state diagrams contain a start and end state.	2 points
The state diagrams contain a side effect on ar least one trigger.	2 points

Implementation
15 points
(due 10/30)

Your team will implement your additional feature in class using pair programming. This will be graded during the demo presentations of your software during your final exam. The Team Leader will choose a new Team Manager for this deliverable. The Team Manager will be responsible for making programming assignments.

Unit tests
10 points
(due 10/30 - System Tests also due this date, see below)

You and your team will use the class diagram as an API to write unit tests against (test driven development). Your team will complete the following:

1. The Team Leader will select a new Team Manager for this deliverable. The Team Manager will create a folder on svn called `Unit_tests`.
2. The Team Manager will assign each group member one or two classes to write unit tests for (the GradeServer class is an external system in this scenario so it cannot be one of the classes you test). See the grading rubric below. Each team member will submit their unit tests though svn to the `Unit_tests` folder (their name should be in comments inside the file).
3. Once all unit tests have been completed, each team member is responsible for reviewing the other team members' diagrams for correctness.
4. This deliverable should be submitted as .java files.

Grading Rubric for Unit tests: (10 points)

All constructors are tested.	3 points
All non-getters/setters are tested.	7 points

Think about how you design your unit tests. What kind of coverage are you trying to achieve? You will need to discuss these items in a future, individual homework assignment.

You will produce a set of test cases for your project. These test cases will be for a subset of your project, not the whole thing. Your team will complete the following:

1. The Team Leader will select a new Team Manager for this deliverable. The Team Manager will create a folder on svn called `System_tests`.
2. The Team Manager will assign each group member one or two use cases to write system tests for. See the grading rubric below. Each team member will submit their system tests though svn to the `System_tests` folder (their name should be in comments inside the file).
3. Once all system tests have been completed, each team member is responsible for reviewing the other team members' tests for correctness.

System/Integration Test Cases
10 points
(due 10/30)

Grading rubric for system tests: (10 points)

Each system test covers all the basic and alternate flows of the selected use case.	5 points
Each system test step uses proper language and instructions for the test inputs (in a deterministic way a person using the system could execute them)	3 points
Each system test step uses proper language and instructions for the expected results (in a way they could be compared for correctness)	2 points

The test case should follow the Sample System Test Case. By this point you should have working code, so the the "Actual Results", and the two following it, should also be filled out.

Helpful Hints: If a test step says "the user presses button x or button y" -- make two test cases, one for button x and one for button y. If a test result says "if the password was valid, the user goes forward, otherwise they get sent to the login screen" -- make two tests, one for a valid login and one for an invalid login. Test cases should be repeatable and clearly test one scenario.

Write a short document (at least 1000 words) for students next semester with respect to how to successfully work in a team environment. Find one over-arching theme or thesis about successful groupwork, and break down your arguments into three components. Use examples from your experience of activities that were productive, and activities that were less useful or hurtful.

This assignment is part of the WAC requirements for the course, and will consist of three deliverables:

- An outline
- A draft
- A final deliverable

Project Team Review Essay 50 points (due 11/17, 12/3, and date of final exam)

The three components are described below:

Essay outline: 10 points (due 11/17)

Outline is a bulleted list	1 point
Outline contains a theme/thesis as the first bullet	2 points
Outline contains three bullets that relate to the thesis (each should be a full sentence tying in to the thesis)	3 points
Each of the three bullets above contains at least one sub-bullet of evidence/experience to support the argument	3 points
Outline has a meaningful title	1 point

Essay Draft: 30 points (due 12/3)

The essay has an opening paragraph that provides a brief introduction/illustration of what teamwork is on a software project	3 points
The essay contains a theme/thesis as the basis for the subsequent arguments	4 points
The essay uses proper spelling and grammar	6 points
The essay flows cleanly between different paragraphs	5 points
Each paragraph ties in to the thesis	6 points
Each paragraph uses examples to support its point	6 points
Writing quality estimate	(Up to 10 points)

Final essay: 10 points (due by final exam; bring in old essay and new one)

The essay will be graded on overall writing quality; these points were estimated for the draft; students have an opportunity to earn all 10 points by addressing any suggestions made by the professor on the draft.	10 points

The essay outline and draft will be graded in-class.

You will be using svn to maintain and turn in your deliverables for this project. Once I have your team emails I will setup a project there for you. For each assignment, make sure to name the file as stated in this document, and place it in a folder with the name as described. Most deliverables are required to be in a single file. If you do not follow this convention, then I will not be able to grade your assignments.

Comment [GG3]: Assignment describes grading criteria for writing assignments.

Comment [GG4]: Assignment indicates required draft deadline and revision deadline, with sufficient time for thoughtful faculty feedback and for student revision.

Comment [GG5]: Assingment describes that the instructor will provide feedback on writing.

Comment [GG6]: Assignment describes specific strategies and conventions necessary for student success on this assignment.

For each deliverable, please include your team name, and the names of all of your team members at the top of the first page of the document(s) submitted. Failure to do so will result in a loss of 3 points.

Every team member must contribute equally to each assignment. You should be using svn to assign parts of the deliverables to team members. In addition, each team member must use the svn repository of svn to upload their work/changes. If there are any disputes about contributions to assignments, I will first examine the tasks assigned to the team members through svn, and then look at who committed what to the repository. I reserve the right to retroactively adjust points for deliverables if it turns out that students were not participating equally. Note: showing up for a team meeting, uploading the template file with minimal changes, or submitting a poorly-implemented version of your task do not count as work that had academic merit and will therefore will not receive points. Please note that your deliverables will be graded as a group - therefore, all team members are responsible for ensuring the correctness of the deliverables as a whole. If problems arise with specific individuals, please let me know immediately - I have removed people from groups before for not doing any meaningful work.