COMP603/ENSE600
Program Design & Construction / Software Construction – Assessment

Software Development Project 2

2023 Semester 2

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

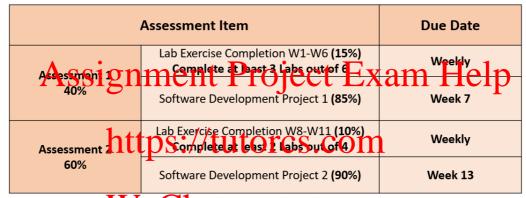# Contents

Introduction

In Assignment 2, you/your group will work on the **same topic**. You can either extend the project (re-use the code from Assignment 1) **or** develop a project from scratch.

Important dates

- You need to register your group by the end of Week 3 by joining a group from Canvas.
- Project 1 source code submission: Week 7, by Friday 11:59 pm
- Project 2 source code submission: Week 13, by Friday 11:59 pm

Overview of the Assessments

- There are two assessments. The first and second take **40%** and **60%**, respectively. Each assessment is comprised of lab exercises and a software development project. Refer to the table below:

Assessment Item		Due Date
Assessment 1 40%	Lab Exercise Completion W1-W6 **(15%)** Complete at least 3 Labs out of 6	Weekly
	Software Development Project 1 **(85%)**	Week 7
Assessment 2 60%	Lab Exercise Completion W8-W11 **(10%)** Complete at least 2 labs out of 4	Weekly
	Software Development Project 2 **(90%)**	Week 13

*Where "**complete**" means you will need to attempt to complete the task and upload it to Canvas.*

- **To pass the course**, To pass the course, students must satisfy the stated learning outcomes and achieve a minimum **overall grade of C- (50%)**.

Requirements of Assignment 2

- In **Software Development Project 2**, you will need to develop a **Graphical User Interface (GUI)** version of the software product using **Java Programming Language**.

- **In Software Development Project 2**, you can either upgrade the software (Project 1) to / newly develop a GUI version. Meanwhile, you will include **a database component** to the product, implement **Design Patterns**, use **Git** for version control, eliminate **code smells**, and include necessary **Unit Tests**.

- In **Software Development Project 2,** you will also need to apply OOP concepts (abstraction, encapsulation, inheritance, polymorphism), try to follow the SOLID design principles, and create multiple classes with relationships.

- **Progress of the project needs to be tracked** via multiple Git commits and all the histories must be retrieved through the ".git" hidden folder (placed under your project folder) or from GitHub.

- The source code of Project 2 will be submitted by the end of **Week 13**.

- Refer to the **important dates** for the due date of both project assignments.

- The program needs to be **bug-free** and has **robust error handling**.

- You need to develop the projects by using **NetBeans 14 and JDK 11.**

- You need to use **Apache Derby DB** (also called JavaDB: http://db.apache.org/derby/) as the database management system. No other types of databases are allowed.

- You need to develop unit tests (using Junit Framework) for testing some important functionalities of the program.

- The program should be easy to build and run without any configuration. The DB should be set up automatically **without** any manual configurations, including manually launching a DB service.

- You need to have an open mind about the functionality of your software project and try your best to make your program robust, interesting, and easy to use.

- You will be expected to develop your own Java code for the projects. You may use the Java standard library and other external libraries.

- You are encouraged to learn more beyond the lectures and apply what you have learnt to the projects. However, the project requirements stated in this document should be satisfied.

- You also need to include a very short (less than one page) report, stating the project setup, GitHub URL (if there is one), and the contribution of each teammate (if you work in a group). If any team member gives less than 40% contribution/no contribution in **terms of coding**, the student will be given a penalty or even fail the course.

- You need to record a short video (less than 5 minutes) using any tools (e.g., MS Teams) to demonstrate your project AND explain the code. The recorded video must be included in the submission.

- Read **Marking Guideline** carefully for the details.

ChatGPT

ChatGPT (Chat Generative Pre-trained Transformer) is a type of language model that is designed to process and generate natural language text. It is based on a deep learning algorithm called a transformer, which uses a neural network to process sequences of text. In this assignment:

- You can use ChatGPT to assist you with programming

- You can use ChatGPT to learn to program

- You can use ChatGPT to debug your program

- You can use ChatGPT to generate PART of the program

- When utilising ChatGPT to generate a method or class, please include a reference or comment in the code that indicates that the code was generated using ChatGPT. It is important to note that any classes or methods that are generated by ChatGPT should not be considered as part of the classes that you have developed.

- However, using ChatGPT to generate an entire program even with minor modifications is considered plagiarism. If we suspect that a project is fully generated, we will request the team to present the project and ask questions about the code.

Project Code Submission

- The project must be submitted via Canvas.

- You need to submit a compressed **ZIP** file that contains:

  o The project folder with all the source codes and related files, e.g., image files, text files, unit tests, etc.

  o A very short (less than one page) report, including:

    - the project setup (e.g., user name and password)

    - GitHub URL (if there is one), and

    - the contribution of each team member if this is group work (this item is not required if you work individually). If any team member gives less than 40% contribution/no contribution in **terms of coding**, the student will be given a penalty or even fail the course.

  o A short video (less than 5 minutes), including

    - Project demonstration, showing all the features by running your project

    - Code explanation, explaining class structures, methods and processes.

  o *For Project 2, you need to include the database folder, which contains all the database files in the compressed ZIP file. You can set the DB directory inside your project folder.*

  o *For Project 2, you need to include .git (hidden folder), having all the historical commits and branches.*

- Submission Guideline

  o Find your group ID from Canvas.

  o When you submit your project, please compress the whole project as a .zip (**not .rar, .7z or any other type**), and rename the zipped file with your group ID and the student ID of members.

  o For example, if the group ID is 9, and the student ID is 1234567, then the name of the submitted file should be "P09_1234567.zip". If you have a partner in your group whose student ID is 7654321, then the name should be "P09_1234567_7654321.zip".

  o Please submit a complete **NetBeans project (Ant or Maven)**. Any improper submission (e.g., non-NetBeans project, several Java files or compiled classes, lack of essential files, improper name of the submitted file) will affect the assessment of your project.

Important Notes

- Plagiarism and self-plagiarism will result in a mark of **zero** in software development and be reported to the faculty. We detect Plagiarism by using commercial software https://codequiry.com/.

  Plagiarism means:
  - Download open-source projects from online sources, e.g., GitHub, and modify the codes
  - Re-use past students' assignments with modifications

- Re-use the assignment of other AUT courses, e.g., ADA, DSA, etc.
- Re-use the assignment the assignment of PDC/SC when repeating this course
- Purchase assignments from any sources
- Generate **the entire project** (with slight modifications) using generative AI tools, e.g., ChatGPT

However, you can:
- Re-use the sample codes and lab solutions given to you.
- Download a utility class/algorithm for your project.
- Download any existing models (e.g., machine learning model, language model)

- Late submission penalty will be applied (5% penalty per day up to a maximum of 5 days. Late assessments after 5 days will not be accepted).

- You have the responsibility to keep and back up different versions of your programs. You may also consider using cloud storage, e.g., OneDrive, Dropbox, and Google Drive, for backup. Losing data (code) will not be considered a valid reason for special consideration.

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Marking Guideline – Assessment 2

Weight	Requirements	Excellent (80%-100%)	Good (60% - 80%)	Satisfactory (40%-60%)	Unsatisfactory (0%-40%)
10%	Complete/show efforts in attempting to complete at least 2 lab exercises out of 4	Complete 2 labs	Complete one lab		No lab completion
20%	**User Interface (GUI)** • Clear and well-designed graphical user interface following common standards • The interface is easy for users to interact with	Users can figure out how to use the system and interact with it easily. GUI can give user-friendly messages when a false input is given. The GUI components are organized nicely. Users can quit at any step without being given an error message. Users also can go back to the menu or restart the system/game at any time.	Users still can interact with the GUI but find it not very convenient. Not enough info is given when a user provides any false input. Users are not able to go back to the menu or restart the system at any time. The output layout is fine but not nicely presented.	The system is hard to interact with, and the users need to figure out how to use it. The GUI components are not well organized, and users may have to re-size the frame before using it. Exceptions are thrown when starting the program.	The program cannot run. No GUI is populated for users to interact.
20%	**Database** • The program contains a database element • Database interactions and operations (input and output) using JDBC/Hibernate are part of the code. • JDBC/Hibernate must contribute to the functionalities of the project.	The project contains a database element, managing data reading and writing. There are more than three database interactions involving reading and more than three interactions involving writing. Derby Embedded mode is utilized, and the program can connect to the database without starting the database service.	The project contains a database element, managing data reading and writing. There are fewer than three database interactions involving reading or fewer than three interactions involving writing. Derby Embedded mode is utilized, and the program can connect to the database without starting the database service.	There is a database component, but the same database connection data is scattered in a few classes. There are a few interactions with the database, but either reading or writing is missing. Derby Network mode is utilized, and the program cannot connect to the database without starting the database service.	No database component. The program has a database component, but it doesn't contribute to the interaction with the database. Other databases are utilized in the program, e.g., MySQL, Oracle, MSSQL, MongoDB etc.

20%	**Software Functionality and Usability** • The program is easy to compile and run **without any manual configurations** (e.g., setup DB, import .jar, etc.) • The users interact with the program without any errors (e.g., the program can handle invalid data input by giving clear error messages and instructions) • The complexity of the functionality	The program can be opened by NetBeans IDE, compiled, and running without any issues. The program can run without any run time error by giving the expected inputs. No manual configuration is required, including starting the database service, adding jars, etc. The functionality of the program is complex enough, having most of the common features of the project implemented (dependent on the size of the team). The implemented features are robust. Any input can be handled perfectly without throwing exceptions. Meanwhile, messages can be prompted to the end-users for the correct input.	The program can be opened by NetBeans IDE, compiled, and running without any issues. No manual configuration is required, including starting the database service, adding jars, etc. The features of the system are not complex enough, missing some common features of the project (dependent on the size of the team) The system works well with valid inputs but gives no hints when a user inputs an invalid value.	The program cannot be opened by using NetBeans IDE. The program cannot be compiled or run. Manual configuration is required, including starting the database service. The features of the system are very simple, having less than three functional points. The system throws exceptions even given valid inputs.	
30%	**Software Design & Implementation** • The program can be compiled successfully • The purpose of the code is easy to understand by reading it • The comments in the code are useful and appropriate • The code executes without runtime errors • The version control is applied • The error handling is thorough and robust • The class structure reflects good design	The project can be compiled without any errors. There are more than 11 reasonable classes (12+) with reasonable methods. The relationships among the classes are well presented. All the OOP concepts are applied, including abstraction, encapsulation, inheritance, and polymorphism. The program is well-structured, having a clear separation of UI and model. Design patterns are applied where appropriate. Comments of methods are given, and the codes are easy to read.	The project can be compiled without any errors. There are 8-11 reasonable classes with reasonable methods. The relationships among the classes are well presented. Some of the OOP concepts are applied. The program is relatively well-structured but without a clear separation of UI and model. Design patterns are applied where appropriate. Comments of methods are given, and the codes are easy to read. There are obvious code smells. Version control is applied but	The project can be compiled without any errors. There are 5-7 reasonable classes with reasonable methods. The relationships among the classes are well presented. Some of the OOP concepts are applied. The program is not well-structured, and UI and model are mixed together. Design patterns are not applied. Comments of methods are missing, and the codes are hard to read. There are obvious code smells. Version	The project cannot be compiled. There are more than 1-4 reasonable classes with reasonable methods. The relationships among the classes are missing. OOP concepts are not well applied. The program is not well-structured, and design patterns are missing. The program may contain only a few JFrameForms

	• Design patterns are implemented correctly where appropriate • A good coding style is used, following appropriate coding standards • The code design follows OO design good practice • There are no obvious code smells • Sufficient test cases are included, which are well written. The important functionalities are properly tested.	There is no obvious code smell. Version control is applied, showing multiple commits and branches merging. The program is very robust and can handle all kinds of inputs without throwing an exception. At least five test cases are included. The tests are well written, cover the important functionality well, and the tests are well-named.	doesn't show a lot of commits or branches merging. The program is relatively robust but still throws 1-2 exceptions. At least five test cases are included. Some test cases fail to test the important features.	control is applied but doesn't show a lot of commits or branches merging. The program is not very robust and throws more than 2 exceptions. Less than five test cases are included. Some test cases fail to test the important features.	without any other Java beans and models. Comments of methods are missing, and the codes are not easy to read. The program is not robust at all, and it throws many exceptions. No test cases or test cases are NOT implemented using the Junit framework.
-0% to -100%	Others • Late Submission • Unsatisfactory submission • Plagiarism Detection • A video for the project demonstration and code explanation • Contribution (if you work in a group, please explain the contributions of EACH of the team members)	Late submission penalty: 5% penalty per day up to a maximum of 5 days. Late assessments after 5 days will not be accepted Unsatisfactory submission: Cannot open using NetBeans IDE -5%, Not a ZIP file -5%, Not following the submission file naming convention -5% Late submission with SCA: no penalty if the assignment is submitted within the SCA period. Plagiarism: If the assignment is classified as Plagiarism, all the items (except for labs) will be given 0. Contribution/Team issue: Less than 40% of the contribution will gain 60% of the total project marks. No contribution will get 0. SCA Rules: If you intend to lodge a Special Consideration Application, please hold your assignment submission. If a student submits an SCA AND the assignment before the due date, the late submission will NOT be graded. Missing video: The team will be requested to record a video or join a Teams session to present the project and explain the code. Failure to submit a video and not attending the Teams review session will result in a DNC.			