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

2025 Semester 2

Contents

Introduction

This semester, you will design and develop a software product that will consist of **TWO** projects (Assignment 1 and Assignment 2). Both projects must be on the **same topic.** You can work **individually or in a group of 2 students**. The marking rubric is the same for individual projects and group projects. We strongly recommend you work on the project in a group. For a team project, every team member will have a significant contribution in terms of coding. You may want to select a project from the list below **or** come up with your own.

- Student Information Management System
- Course Selection System – a program for AUT students to select papers
- Virtual Academic Advisor – a chatbot for providing advice to AUT students for study planning
- Knowledge Repository – a system for knowledge management
- A Simple Q/A system – a system for lecturers to input Q/A, allowing AUT students to search
- A Simple ERP system – Enterprise Resource Planning system
- Service Desk System – a system for customer support
- Software Project Registration system – a system for students to register their PDC/SC project
- Workforce Planning System
- Ticket Booking System (for train, games, shows, etc.)
- Hotel Booking System
- Online Shopping System
- Inventory Management System
- Conference Management System
- Foreign Language (Vocabulary) Learning Software
- Card Game
- Virtual Pet Game
- Puzzles (Sudoku, Crosswords, Mazes, …)
- RPG game
- Chess Games
- Board Games
- Deal or No Deal
- Who Wants to Be a Millionaire?

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.

Late submission of labs is not counted. For semesters with different structures:

1. 6 Weeks + Break + 6 Weeks:
    - First assessment for LABS: Covers Weeks 1-7
    - Second assessment for LABS: Covers Weeks 8-11
2. 7 Weeks + Break + 5 Weeks:
    - First assessment for LABS: Covers Weeks 1-6
    - Second assessment for LABS: Covers Weeks 8-11

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

Requirements of Assignment 1

- Object-Oriented (OO) Programming concepts must be applied to the project. Important OO concepts, i.e., encapsulation, abstraction, inheritance and polymorphism, must be reflected.
- In **Software Development Project 1**, you will need to develop a **Command-line User Interface (CUI)** version of the software product using **Java Programming Language**.
- In **Software Development Project 1,** you will need to apply ALL the fundamental OOP concepts (abstraction, encapsulation, inheritance, polymorphism), try to follow the SOLID design principles, create multiple classes with relationships, and use text files to store input and output data from the program.
- The source code of Project 1 will be submitted by the end of **Week 7.**
- 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 23 and JDK 21.**
- The program should be easy to build and run without any manual configuration.
- 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 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 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.
- One of the team members needs to submit the project if you work in a group.
- You need to submit one compressed **ZIP** file that contains:
    1. The project folder with all the source codes and related files, e.g., image files, text files, unit tests, etc.
    2. A very short (less than one page) report, including:
        - the project setup (e.g., user name and password)
        - 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.

3. 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.

- Submission Guideline
    1. Find your group ID from Canvas.
    2. 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.
    3. 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".
    4. 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 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.

Marking Guideline – Assessment 1

Weight	Requirements	Excellent (80%-100%)	Good (60% - 80%)	Satisfactory (40%-60%)	Unsatisfactory (0%-40%)
15%	Complete/show efforts in attempting to complete at least 3 lab exercises out of 6 ***Late submission is not counted***	Complete three labs	Complete two labs	Complete one lab	No lab completion
15%	**User Interface (CUI)** • The program can handle users' inputs from the CUI properly. • The interface is easy for users to interact with	Users can figure out how to use the system and interact with it easily. CUI can give user-friendly messages when a false input is given. The output layout is clean and nice. Users can quit at any step without being given an error message.	Users still can interact with the CUI but find it not very convenient. Not enough info is given when a user provides any false input. Users are not able to exit the system 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. Exceptions are thrown when starting the program.	The program cannot run. No CUI is populated for users to interact.
20%	**File I/O and Collections** • The program inputs and outputs data from/to text files successfully • File I/O must contribute to the functionalities of the project. • Use classes/methods taught in this paper are used to manage File I/O • Appropriate Collections are utilised in the program.	There is a File I/O component (a java class), which manages file reading and writing. There are more than three functions involving File reading and more than three functions involving File writing. The File I/O classes delivered in the lectures are utilised. Two kinds of collections (List, Map, Set) are properly applied to the programs.	File I/O is used in the program. Both file reading and writing are applied, but there are very limited interactions with files. The classes delivered in lectures are used. Some of the collections (List, Map, Set) are applied to the programs, but some appear not appropriate.	There is File I/O, but the classes delivered in the lectures are missing. There are a few interactions with files, but either File reading or File writing is missing. File I/O functions are scattered everywhere in the program. Only one kind of Java Collections is used, and it appears inappropriate.	No file I/O component. No collections are used.

20%	**Software functionality and usability** • The program is easy to compile and run **without any manual configurations** (e.g., setup input/output files, import .jar files, etc.) • The program works as expected without any errors • The functionality of the program is easy to learn and follow • Complexity and robustness 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. 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. 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 invalid values.	The program cannot be opened by using NetBeans IDE. The program cannot be compiled or run. 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 • OOP concepts (abstract, encapsulation, inheritance and polymorphism) are well applied. • Follow the SOLID design principles. • The comments in the code are useful and appropriate • The program demonstrates codes modularity and	The project can be compiled without any errors. There are more than 9 reasonable classes (10+) with reasonable methods. The relationships among the classes are well presented. All the OOP concepts are applied, including abstraction, encapsulation, inheritance, and polymorphism. SOLID design principles 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 7-9 reasonable classes with reasonable methods. The relationships among the classes are well presented. Some OOP concepts are applied, including abstraction, encapsulation, inheritance, and polymorphism. Comments of methods are given, and the codes are easy to read. The program is relatively robust but throws 1-2 exceptions.	The project can be compiled without any errors. There are 4-6 reasonable classes with reasonable methods. The relationships among the classes are missing. Some OOP concepts are applied, including abstraction, encapsulation, inheritance, and polymorphism. Comments of methods are missing, and the codes are not easy to read. The program is not very robust and throws more than 2 exceptions.	The project cannot be compiled. There are more than 1-3 reasonable classes with reasonable methods. The relationships among the classes are missing. OOP concepts are not well applied. 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.

	reusability • The code executes without runtime errors • The error handling is thorough and robust • The class structure reflects good design	The program is very robust and can handle all kinds of inputs without throwing an exception.			
-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.			