COMP 2452 – Software Development 2

Course Description

Calendar entry

This course provides more advanced coverage of current approaches to software design, validation and verification, human interface design, computer security, and information management. May not be held with COMP 2150. Prerequisites: COMP 2450 and COMP 2400.

General Course Description

Software development is an ongoing process: the needs we (as humans) have from software change over time, so we need to change our software to meet those new needs.

In this course we will continue building large and complex software by discussing modular software design (design patterns and software architectures), automating testing with software testing frameworks, model and store information and relationships in a database, ensure that our software and data are secure, introduce ideas from AI and ML, and design and build simple graphical user interfaces that adhere to interface design principles.

Detailed Prerequisites

Before entering this course, a student should be able to:

- Design and implement larger, interactive, text-based applications (COMP 2450).
- Verify and validate that code meets a set of well-defined expectations through manual and basic automated testing (COMP 2450).
- Model data and its relationships, persist data in a structured format (COMP 2450).
- Describe and build secure applications through appropriate programming practices (COMP 2450).
- Analyze the impact that a program's design choices may have on users and other people and factors within the system's ecosystem (COMP 2450).
- Design, analyze, and implement a finite state machine (COMP 2400).

Course Goals

By the end of this course students will:

- Plan and make changes to existing large, interactive applications (adding GUIs, applying object-oriented concepts, state machines, and event-based paradigms).
- Iteratively apply structured testing to a large, interactive application using a testing framework.
- Persist and retrieve data models and their relationships in a database.

- Implement basic artificial intelligence algorithms.
- Apply standard design principles and heuristics to interface and interactive system design.

Learning Outcomes

Software design

Students should be able to:

- Identify and use some basic design patterns (adapter, iterator, etc).
- Identify dependencies between components, and strategies for minimizing those dependencies (coupling and cohesion).
- Refactor an existing application to use a simple software architecture (e.g., MVC).

Software verification and validation

Students should be able to:

- Write a thorough and well-organized set of unit tests.
- Use a testing framework to automate running test suites.

Information management

Students should be able to:

- Model relationships between data model objects in a database (e.g., uniqueness, relationship constraints, transactions).
- Create, populate, and query data in a database.
- Use a programming language to interact with a database; adding, removing, updating, and finding data based on user interactions.

Programming practices for secure applications

Students should be able to:

- Begin to identify the kinds of risks associated with storing user data/information.
- Securely represent and store credentials for authentication purposes.
- Implement a state machine that ensures their application only takes acceptable inputs and rejects invalid inputs.
- Use tools such as prepared statements to prevent SQL injection attacks.

Designing software for humans

Students should be able to:

- Implement simple graphical interfaces following 1–2 standard design principles (e.g., feedback) using a modern API.

- Explain how the graphical interface adheres to the design principles.

Artificial intelligence

Students should be able to:

- Describe and implement basic artificial intelligence algorithms (e.g., a decision tree).