Task Management System:

Design and Testing Summary

Mikaella Ziade

Marianne Moukashar

Adam Fawzi

Design Summary

Object Oriented Structure

The program is designed using object-oriented programming. Key components:

1. **Classes:**
- A general task takes the role of a base class.
- Used subclasses for specific classes (e.g. One-on-one Meeting and General Meeting are subclasses of the Event Class)
2. **Encapsulation:**
- Every class contains private variables with public getter and setter methods.
3. **Inheritance:**
- Enables general use of any task while allowing customization for specific cases.
4. **Polymorphism:**
- The displayinformation() method is overridden in derived class for displaying specific task information.

Data Management

- Tasks are stored in a dynamic vector that enables one to add, remove, and loop through the list as desired.
- The existence of two different vectors allows for the implementation of undo and task reversal operations.

User Interactions

- The main menu interface uses numbered options for easy access.
- Input validation confirms user input correct data (eg. Dates, times, titles that are not blank).

Flexibility and Extensibility

- New Tasks are easily added by extending the Task class with minimal changes.
- Features such as reminders and conflict checking are in the pipeline for future implementation.

System Architecture

- Core Classes: Task, Time, Location, Event, Meeting.
- Workflow:

1. Data entered by users to formulate, update, or query tasks.
2. Tasks operate on location and time parameters to arrange or select outcomes.

Lessons Learned

- Encapsulation is necessary to trace different responsibilities and tasks in programming.
- Subclass and inheritance complicate maintenance; composition may be considered in future iterations.

Testing Summary

Testing Framework

The project uses Google Test for unit testing to check core requirement.

Test Coverage

1. **Class Methods:**
- Ensured proper implementation of get and set methods for all classes (eg. Time, Location).
- Tested displayinformation() method for base and derived classes.
2. **Task Operations:**
- Creating a new task, Editing a task and deleting a task.
- Tasks are searched using attributes such as title, time, location.
3. **Input Validation:**
- Made certain that invalid inputs are not accepted.

Challenges

1. Changes made on test scripts due to difficulty simulating user input during tests.
2. Many modifications were made to the code during development, so the test cases required constant updates.

Lessons Learned

1. Running tests automatically using Google Test saved a lot of time.

2. Checking for invalid input during the test helped detect any errors making the program more reliable.