Software Systems Design and Testing

COS6028-B

Coursework 1

Weighting: 40% of final module mark
Due date: 13 March 2026, 3pm

Department of Computer Science
University of Bradford

Learning Outcomes Assessed

- **LO1**: Demonstrate a critical awareness and ability to use practical techniques for software development with regard to design, implementation and testing.
- **LO2**: Demonstrate an understanding of practical challenges associated with the development of a significant software system.
- **LO3**: Demonstrate an ability to use testing methods appropriate in finding errors in different stages of the software development life cycle.

Submission Requirements

Submit via Canvas:

1. A PDF report.
2. A ZIP containing a Java project.
3. A video demonstration of up to 5 minutes, in MP4 or similar format. You may use any screen-recording tool to create the video, for example, Microsoft Teams.

Academic Integrity

Referencing and plagiarism:
`https://www.bradford.ac.uk/library/help/referencing-and-plagiarism/`

Coursework Description

Intermediate software deliverable and report (2500 words).

Scenario

As part of the University of Bradford's commitment to digital security, all students and staff are required to use Multi-Factor Authentication (MFA) when signing in to University systems. MFA provides an additional layer of protection beyond your username and password, helping to keep your personal information and University data secure. When you log in, MFA triggers a two-step verification process. After entering your password, you must confirm your identity using a second method—usually on a separate device. For most users, this second step takes place on their mobile phone, using an app notification, an authenticator app, a code sent by text message or email, or a phone call.

This extra step ensures that the person accessing your account is genuinely you. Even if someone else manages to get hold of your password, it's highly unlikely that they also have access to your phone. MFA therefore significantly reduces the risk of unauthorised access to University systems.

Your task is to design, implement and test an MFA prototype for the University of Bradford. Your solution must fully adhere to the requirements described in the following sections.

1 Design [15 marks]

Design a software system that provides sign-in functionality using MFA for the University of Bradford. Your design must demonstrate the effective use of one or more appropriate design patterns and adherence to relevant SOLID principles. Your solution must ensure that:

- The implementation details of individual authentication methods are fully decoupled from the code that uses them.
- It is possible to add new MFA methods with minimal changes to existing client code.

Provide a UML class diagram that shows the main classes, interfaces and their relationships. Accompany the diagram with a description of the classes and interfaces, their key methods and an explanation of the design patterns used. Discuss the following:

- The benefits of the chosen design patterns and why they are suitable for an MFA system.
- Which SOLID principles your design upholds and how these principles are reflected in your solution.

Criterion	**Marks**
Accuracy and completeness of UML diagram and accompanying description	5
Use of appropriate design patterns and clear explanation of their benefits	5
Explanation of how SOLID principles are upheld	5

2 Implementation [15 marks]

Develop and demonstrate a Java application that implements your design. Your solution must provide a prototype implementation of at least two additional authentication methods, in addition to password-based authentication. You may hard-code a few user accounts or store them in a file; you are not required to use a database. Include screenshots showing the execution of the application, each accompanied by a clear description explaining what the screenshot demonstrates.

Hint: There are Java APIs for sending text messages or emails—some service providers offer free SMS trials—and also TOTP (Time-based One Time Password) libraries that you may use.

Criterion	**Marks**
Correctness and completeness of the implementation, reflecting the proposed design	5
Quality, clarity and organisation of the code	5
Screenshots are clear, relevant and accurately described	5

3 General questions (not scenario-specific) [10 marks]

Question 1 [5 marks]

Which design patterns do you find most useful? Provide an example of a design pattern that you think you will use in the future and describe a typical scenario in which you would benefit from it.

Criterion	Marks
Explanation of patterns found useful	2
Example of a future pattern and application scenario	3

Question 2 [5 marks]

The Singleton pattern restricts the instantiation of a class to a single instance. Although widely used, it is also controversial and is often described as an "anti-pattern". Research this pattern, explain how it works and discuss why it may be considered undesirable in some design contexts.

Criterion	Marks
Explanation of how the Singleton pattern works	2
Critique and research-informed reasoning	3

4 Unit testing [20 marks]

For all the classes and methods implemented, provide comprehensive JUnit tests. These should be included in the Java project, placed in a separate test package. The tests must cover each individual method from each class. Use a NetBeans plugin to measure code coverage and include screenshots showing:

- the JUnit test report statistics;
- the percentage of code coverage achieved by your tests, as measured by the plugin.

Criterion	Marks
Comprehensive JUnit tests covering each individual method from each class	10
Good JUnit test report statistics	5
Good code coverage results	5

5 Functional testing [10 marks]

Choose a non-trivial method of your implementation and illustrate how to apply equivalence class partitioning and boundary value analysis to generate test data. Provide the following:

- the identified equivalence classes;
- the combinations derived from these classes;

- the relevant boundaries;
- the specific input values selected for each corresponding test case.

Criterion	Marks
Clear identification of relevant equivalence classes	3
Correct derivation of combinations based on these classes	2
Correct identification of boundary values	2
Selection of appropriate test input values with clear justification	3

6 Structural testing [10 marks]

Choose another method from your implementation, different from the one selected for functional testing. This should be a more complex method (e.g., one with higher cyclomatic complexity). For this method, present the following:

- the Control Flow Graph (CFG);
- the feasible execution paths;
- the test cases you would construct for each path.

Illustrate how you design these test cases to achieve:

- statement coverage;
- decision coverage;
- condition coverage;
- MC/DC coverage.

Criterion	Marks
Correct construction and presentation of the CFG	2
Identification and explanation of feasible execution paths	2
Design of appropriate test cases for each path	2
Explanation of how tests achieve statement, decision, condition and MC/DC coverage	4

7 Video demonstration [20 marks]

Provide a video demonstrating and explaining the execution of your system and tests.

Criterion	Marks
Clear demonstration and explanation of the system execution	10
Clear demonstration and explanation of the test execution	10