Phase 1: Requirements + UML + Refactoring

Group Number: XX, Student Names: A, B, C, D

Wednesday, March 6, 2024

Table of contents

> **Warning**
>
> Important: This document provides insights into how to write each section. Do not just rely on this template, consult the project specification to ensure completeness of your report. Avoid modifying section titles or numbers.

1 Requirement Analysis (4%)

1.1 Current System Analysis

Briefly describe the existing functionalities of the Quackstagram codebase.

1.2 Stakeholder Engagement

Summarize the feedback and additional requirements gathered from potential users and stakeholders (think about the requirements you extracted from the transcript).

1.3 Requirements Specification

1.3.1 Functional Requirements

List the detailed functional requirements.

1.3.2 Non-Functional Requirements

List the detailed non-functional requirements.

1.4 Priortization of Requirements

Indicate the priority of the requirements listed above and provide a rationalization for your prioritization. You don't need to justify each requirement individually but may do so for categories of requirements. For example, if privacy is emphasized as crucial in the interview transcripts, you can explain giving high priority to all privacy-related requirements.

Here is an illustrative table of requirements and prioritization (note: these requirements are fictional and should not be used in your submission):

Requirement ID	Requirement Description	Priority
REQ-01	Implement rainbow color scheme	High
REQ-02	Allow users to upload cat pictures	Medium
REQ-03	Integrate with fictional API XYZ	Low
REQ-04	Support voice commands for navigation	High
REQ-05	Generate unicorn avatars for new users	Medium
REQ-06	Include a feature to find nearest coffee shop	Low

Rationalization for Requirement Prioritization:

- **High Priority:** Implementing a rainbow color scheme and supporting voice commands are essential for ensuring accessibility and attractiveness of the interface.
- **Medium Priority:** Unique aspects like allowing users to upload cat pictures and generating unicorn avatars, which can significantly improve user satisfaction.
- **Low Priority:** While beneficial, integrating with fictional API XYZ and including a feature to find the nearest coffee shop are considered low priority.

2 UML Modeling (4%)

2.1 Current System UML Diagrams

Insert the UML diagrams (structural, behavioral, and interaction) of the current system. Ensure there are at least 2 diagrams of each type.

2.1.1 Structural Diagrams

1. Diagram 1
2. Diagram 2

2.1.2 Behavioral Diagrams

1. Diagram 1
2. Diagram 2

2.1.3 Interaction Diagrams

1. Diagram 1
2. Diagram 2

2.2 Use Cases

Provide detailed use cases illustrating possible interactions with the system. Include at least 6 use cases.

1. Use Case 1
2. Use Case 2
3. …
4. Use Case 6

3 Refactoring (4%)

Include a list of code segments you have refactored and the rationale behind each choice.

Consider the following example and structure your report in a similar manner:

Before Refactoring:

```
public class Calculator {
    public int calculate(int a, int b, String operation) {
        if (operation.equals("add")) {
            return a + b;
        } else if (operation.equals("subtract")) {
            return a - b;
        }
        return 0;
    }
}
```

After Refactoring:

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
```

```
    public int subtract(int a, int b) {
        return a - b;
    }
}
```

Rationale:

1. **Clear Method Responsibilities:** The refactored code provides specific methods for each operation (`add` and `subtract`), improving clarity and making the code easier to read.
2. **Removes Conditional Logic:** Eliminating the `if-else` condition simplifies the `calculate` method's logic, reducing complexity and potential errors in selecting operations.
3. **Enhances Maintainability:** With dedicated methods for each arithmetic operation, adding new operations (like multiply or divide) becomes easier and the codebase more maintainable.

4 Object-Oriented Design (4%)

Include a list of design changes you have made to adhere to good Object-Oriented Design principles along with a justification. You can use a structuring similar to that of Refactoring above (before and after code snippts and an explanation)

5 Proposal for New Functionality (Not graded)

Describe the new functionality that is proposed to be added to the system, including its relevance and expected impact on the overall system.