NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

SOFTWARE QUALITY ENGINEERING

ASSIGNMENT 01: QUALITY PLAN

Team Members	1. Faaiz Kadiwal (21K-3830) 2. Ammar Asdaque (21K-3923) 3. Anser Tayyab (21K-3909)
Course Instructor	Ms. Syeda Rubab Jaffar
Submission Date	03-October-2023

PROJECT QUALITY PLAN

FAST UNIVERSITY FOOD ORDERING APPLICATION

DOCUMENT ID: **[FastFood-001]**

ISSUE DATE: **03-October-2023**

Version 1.0.0

ABOUT TEAM

Roles	***Names***
Quality Assurance Manager	*Faaiz Kadiwal*
Project Manager	*Syed Ammar Asdaque*
Developer(s)	*Faaiz, Ammar, Anser*
QA Team Member(s)	*Faaiz, Ammar, Anser*

VERSION HISTORY

Version	***Revision Date***	***Description of Change***	***Author***
1.0.0	*October 2, 2023*	*Initial Draft*	*Faaiz Kadiwal*

FOR FURTHER QUERIES CONTACT OUR TEAM,
K213830@nu.edu.pk
K213923@nu.edu.pk
K213909@nu.edu.pk

INTRODUCTION

*The Quality Plan for the **Fast University Food Ordering Application** outlines the strategies and processes to ensure the delivery of a high-quality, real-time food ordering application within the university campus. This document establishes quality objectives, verification and validation activities, procedures for problem reporting, and the tools and methodologies employed throughout the project.*

QUALITY OBJECTIVES

Our quality objectives for the Fast University Food Ordering Application have been carefully crafted to ensure the successful development and operation of a real-time food ordering system. Each objective is tailored to address specific aspects critical to the application's functionality and user satisfaction:

Reliability:
Ensure the application operates without errors, providing a seamless ordering experience and ensuring correctness.

- ***Objective:** Achieve a system uptime of at least 99.9% to guarantee uninterrupted service for users.*
- ***Measurement:** Monitor server logs and track incidents to ensure that the application remains error-free during peak usage times.*

Performance:
Achieve optimal speed and responsiveness to handle simultaneous orders efficiently.

- ***Objective:** Maintain an average response time of under 2 seconds for critical user interactions, such as placing an order.*
- ***Measurement:** Employ performance monitoring tools to track response times, server load, and resource utilization under various load conditions.*

Usability:
Design an intuitive interface for easy navigation and order placement.

- ***Objective:** Attain a user satisfaction rating of at least 90% based on usability feedback.*

- ***Measurement:*** *Conduct usability testing sessions with representative users to evaluate the ease of navigation and order placement.*

Security:

Implement robust security measures to protect user data and transaction integrity.

- ***Objective:*** *Achieve compliance with industry standards for data security and implement encryption for sensitive user information.*
- ***Measurement:*** *Regularly conduct security audits and vulnerability assessments to ensure the application's resistance to potential threats.*

Compliance:

Adhere to relevant data protection and security standards applicable to food ordering systems.

- ***Objective:*** *Ensure compliance with local and international data protection regulations, such as GDPR or equivalent.*
- ***Measurement:*** *Regularly review and update data protection policies and practices to align with evolving standards.*

Maintainability:

Develop code and documentation for future updates and enhancements.

- ***Objective:*** *Maintain a code maintainability score of at least 80% based on code analysis tools.*
- ***Measurement:*** *Utilize code review and documentation practices to ensure that future updates can be implemented seamlessly.*

Customer Satisfaction:

Continuously gather user feedback to improve the application's features and functionality.

- ***Objective:*** *Maintain a customer satisfaction rating of at least 85% based on feedback surveys and reviews.*
- ***Measurement:*** *Regularly collect and analyze user feedback through surveys, reviews, and support interactions to identify areas for improvement.*

By adhering to these quality objectives, we aim to deliver a Fast University Food Ordering Application that not only meets but exceeds the expectation of its users. Regular monitoring, measurement, and improvement efforts will ensure the application's ongoing success and its ability to adapt to changing user needs and industry standards.

QUALITY ROLES AND RESPONSIBILITIES

Roles	***Responsibilities***
Quality Assurance Manager	*Oversight of overall quality control process.*
Project Manager	*Coordination of quality efforts within project.*
Developer(s)	*Adherence to coding and quality standards.*
QA Team Member(s)	*Execution of testing and quality assurance tasks.*

VERIFICATION AND VALIDATION ACTIVITIES

To ensure the quality and reliability of the Fast University Food Ordering Application, a comprehensive set of verification and validation activities will be executed throughout the development lifecycle. Each activity is designed to address specific aspects of the application, ranging from code quality to security and performance. Here's an in-depth explanation of each activity:

Code Reviews:
Regular reviews by the development team to ensure coding standards and identify potential issues.

- ***Frequency:*** *Code reviews will be conducted regularly, ideally after the completion of each significant code change or feature implementation.*
- ***Participants:*** *Development team members will actively participate in code reviews, offering diverse perspectives and insights.*
- ***Focus Areas:*** *Code reviews will emphasize adherence to coding standards, best practices, and the identification of potential issues such as logical errors, code smells, and maintainability concerns.*

Unit Testing:

Developers will perform unit testing to verify the functionality of individual components.

- ***Testing Frameworks:*** *Unit tests will be written using appropriate testing frameworks (e.g., JUnit for Java, NUnit for C#) to validate the functionality of isolated code units.*
- ***Automation:*** *Whenever possible, unit tests will be automated to ensure swift execution and integration with continuous integration (CI) processes.*
- ***Scope:*** *Unit testing will cover individual functions, methods, or classes to confirm their correctness under various scenarios.*

Integration Testing:

Ensure seamless interaction between different components of the application.

- ***Test Environments:*** *Integration tests will be conducted in environments that closely resemble the production setup.*
- ***Data Flow:*** *The focus of integration testing will be on validating the interaction and data flow between interconnected components.*
- ***Tools:*** *Automated testing tools may be employed to streamline the execution of integration tests and enhance coverage.*

System Testing:

Comprehensive testing of the entire system to validate its compliance with requirements.

- ***End-to-End Testing:*** *System testing will simulate real-world scenarios to validate the application's overall functionality and user interactions.*
- ***Test Data:*** *Diverse sets of test data will be used to ensure that the system performs consistently across different input conditions.*
- ***Regression Testing:*** *System tests will include regression testing to confirm that new features or changes do not adversely affect existing functionalities.*

User Acceptance Testing (UAT):

Involve end-users to ensure the application meets their expectations.

- ***Real User Scenarios:*** *End-users will be engaged to test the application in a real-world context, providing valuable feedback on usability, user interface, and overall satisfaction.*
- ***User Feedback Sessions:*** *UAT will involve interactive sessions with end-users to gather insights, address concerns, and incorporate user preferences.*

Performance Testing:
Assess the application's performance under various user load conditions.

- ***Load Testing:*** *Simulate different levels of user load to evaluate the application's responsiveness, scalability, and stability under normal and peak conditions.*
- ***Tools:*** *Performance testing tools like Apache JMeter or Gatling may be employed to measure response times, throughput, and resource utilization.*

Security Testing:
Conduct thorough security audits and penetration testing to identify and mitigate vulnerabilities.

- ***Security Audits:*** *Regular security audits will be conducted to assess the application's codebase, configurations, and data handling processes.*
- ***Penetration Testing:*** *Ethical hacking techniques will be employed to identify potential vulnerabilities, and necessary measures will be taken to address any security concerns.*

By systematically implementing these verification and validation activities, the development team aims to deliver a Fast University Food Ordering Application that not only meets the specified requirements but also exceeds user expectations in terms of functionality, security, and performance. Regular feedback loops, thorough testing, and continuous improvement efforts will contribute to the overall success and reliability of the application.

PROBLEM REPORTING AND CORRECTIVE ACTIONS

In the development and maintenance of the Fast University Food Ordering Application, a systematic approach will be taken to address defects and issues through a structured problem reporting and corrective actions process. The goal is

to identify, track, analyze, and resolve issues efficiently. Below is a detailed explanation of each step in the process:

Problem Identification:

Defects and issues will be reported through a standardized process, including severity assessment.

- ***Standardized Reporting Process:*** *Team members and stakeholders will use a standardized format or tool to report defects and issues. This ensures that all relevant information, including a detailed description, steps to reproduce, and severity assessment, is captured.*
- ***Severity Assessment:*** *A predefined severity scale will be used to assess the impact of each reported issue. This scale may include categories such as Critical, Major, Minor, or Low. Severity levels help prioritize the order in which issues are addressed.*

Issue Tracking:

Utilize an advanced issue tracking system, such as Jira, to log and track issues, assigning those to responsible individuals.

- ***Jira Issue Tracking:*** *Jira, a robust issue and project tracking tool, will be utilized for logging and tracking issues. Each reported issue will be documented as a Jira ticket, providing a centralized location for issue management.*
- ***Assignment to Responsible Individuals:*** *Each Jira ticket will be assigned to the team member or stakeholder responsible for addressing the issue. Clear ownership ensures accountability and a streamlined resolution process.*

Root Cause Analysis:

Conduct a thorough root cause analysis to identify underlying reasons for issues.

- ***Root Cause Analysis Techniques:*** *Various techniques, such as the "5 Whys" or fishbone diagrams, will be employed to systematically identify the root causes of reported issues. This analysis aims to understand the underlying reasons rather than just addressing surface-level symptoms.*
- ***Cross-Functional Collaboration:*** *Root Cause Analysis will involve collaboration among relevant team members, including developers, testers, and domain experts, to gain diverse perspectives and insights.*

Corrective Actions:
Based on the root cause analysis, implement corrective actions promptly.

- ***Action Plan Development:*** *A structured action plan will be developed based on the findings of the root cause analysis. This plan outlines specific steps and measures to address the identified root causes.*
- ***Implementation of Corrective Actions:*** *The development and testing teams will collaboratively implement the corrective actions outlined in the action plan. This may involve code modifications, process improvements, or other necessary adjustments.*

Validation:
After corrective actions are taken, conduct retesting to verify issue resolution.

- ***Retesting Process:*** *A dedicated retesting process will be conducted to validate that the corrective actions effectively address the reported issues. Test cases related to the specific issues will be executed to ensure successful resolution.*
- ***Validation Criteria:*** *Clear criteria for validating issue resolution will be defined, including the expected behavior and performance. Validation will be considered successful only if the identified issues no longer persist.*

By following this structured problem reporting and corrective actions process, the development team aims to address issues promptly, improve overall product quality, and ensure the reliability and stability of the Fast University Food Ordering Application. This systematic approach supports continuous improvement and enhances the application's performance and user satisfaction over time.

TOOLS, TECHNIQUES AND METHODOLOGIES

Our development and testing processes for the Fast University Food Ordering Application will incorporate a set of tools, techniques, and methodologies aimed at ensuring efficiency, reliability, and adherence to industry standards. Below is a detailed explanation of each component:

Development Tools:
Utilize industry-standard tools such as Visual Studio Code for coding, debugging, and version control.

- ***Visual Studio Code (VS Code):*** *This open-source code editor by Microsoft is highly versatile and supports various programming languages. It provides an integrated development environment (IDE) with features like syntax highlighting, debugging support, and Git version control integration.*
- ***Git:*** *Integrated with VS Code, Git will be used for version control. This distributed version control system enables collaborative development, tracking changes, and managing different code versions.*
- ***GitHub or GitLab:*** *Hosting repositories on platforms like GitHub or GitLab facilitates collaboration, code review, and project management. These platforms enhance version control with features like issues tracking and pull requests.*

Testing Tools:

Implement automated testing using tools like Selenium for efficient and thorough testing.

- ***Selenium:*** *Selenium is a widely-used open-source testing framework for web applications. It supports automation across multiple browsers and platforms, making it suitable for our web-based food ordering application.*
- ***JUnit/TestNG (Java), NUnit (C#):*** *These testing frameworks will complement Selenium for writing and executing automated test scripts. They provide a structure for organizing tests and reporting results.*
- ***Cucumber/Gherkin:*** *For behavior-driven development (BDD), Cucumber with Gherkin syntax will be employed. This facilitates collaboration between non-technical and technical team members, ensuring that the code meets specified behavior.*

Coding Standards:

Adhere to the established coding standards of the university and industry best practices.

- ***University Coding Standards:*** *Follow the coding standards set by the university to maintain consistency across the development team. This includes conventions for naming, formatting, and documentation.*
- ***Industry Best Practices:*** *Incorporate widely accepted industry best practices such as code modularity, readability, and scalability. Adhering to these*

practices ensures that codebase is maintainable and follows recognized standards.

Agile Methodology:

Follow an Agile development approach, with regular sprints and continuous integration.

- ***Scrum Framework:*** *Adopt the Scrum framework within the agile methodology for project management. This involves organizing work into time-boxed iterations called sprints, typically lasting two weeks. It includes ceremonies like sprint planning, daily stand-ups, sprint reviews, and retrospectives.*
- ***Jira/Asana/Trello:*** *Utilize project management tools like Jira, Asana, or Trello to plan, track, and manage tasks during sprints. These tools support collaboration, backlog management, and real-time visibility into project progress.*
- ***Continuous Integration (CI):*** *Implement CI practices to automate the process of code integration and testing. Tools like Jenkins or GitLab CI can be employed to automatically build, test, and deploy code changes, ensuring that new features are continuously integrated into the application.*

Adopting these tools, techniques, and methodologies will contribute to the development of a high-quality and efficiently managed Fast University Food Ordering Application. Regular collaboration, automated testing, and adherence to coding standards will enhance the reliability and maintainability of the application throughout its development lifecycle.

CONCLUSION

This Quality Plan reaffirms our commitment to delivering a high-quality Fast University Food Ordering Application. Adherence to quality goals, verification and validation activities, problem reporting procedures, and the use of appropriate tools and methodologies will ensure the successful delivery of this essential application within the university campus.

APPROVAL

This Quality Plan has been reviewed and approved by:

Syed Ammar Asdaque, Project Manager

Faaiz Kadiwal, Syed Anser Tayyab, Syed Ammar Asdaque

03-October-2023