<>

<Skip>

<Phase3.1>

|  |  |  |  |
| --- | --- | --- | --- |
| # | Student Name | ID | Responsibilities |
| 1 | Yara Aldosari | 438201077 | All |
| 2 | Amal Aldoss | 438202770 | All |
| 3 | Shahad Alsweed | 438202122 | All |
| 4 | Nouf Alfakhri | 438200192 | All |
| 5 | Alanoud Alrashed | 438201169 | All |

**Instructor name:**

**Nora alzamel**

# PROJECT QUALITY PLAN

|  |
| --- |
| 1. Introduction to the project |
| Pre-ordering Service Application that provides the customer a way to pre-order from restaurants and cafes’ without having to wait in line. Also, it benefits the merchants to expose themselves to thousands of users, making their customers happier by preparing the orders ahead. |

|  |
| --- |
| 2. Purpose of Quality Plan for project |
| The purpose of this plan is to specify the procedures, techniques and methodologies that will be performed in regards to achieving the SQA (Software Quality Assurance) Plan throughout the “Skip” Project. The objective of the plan is to assure timely delivery and quality of the software within the scope of agreed requirements and conformance to specified resources. Also, the plan aims to assure the software development, operation, evaluation and acceptance standards are documented and followed. Furthermore, ensuring the continuity, review and maintenance of the SQA plan throughout the project and the carry out of necessary document updates that are based on project execution results. |
|  |

|  |
| --- |
| 3. Project Scope |
| Skip application that helps the customers and the merchants manage their time. It helps the customers by giving them the ability to order before arriving and monitoring when the order will be ready. It also helps the merchants by allowing them to prepare orders ahead of time which helps the workflow to go smoothly. Both the merchants and the customers must register to use the application. The merchant can add several branches to their store. The customer can view all the stores and order from the one they are interested in.  Skip application will:   * benefits the merchants by helping them expose themselves to thousands of users. * Save customers time by helping them skip straight to order pick up line and avoid the ordering line. |

|  |
| --- |
| 4. General Constraints |

1. The project will be using the GitHub environment for managing different versions of the work.
2. Applications data storage will be only limited to Amazon’s AWS Cloud Storage
3. Users must have their correct username and password in order to use the application.
4. The application shall comply with the ISO15504 Standard.

|  |
| --- |
| 5. Functional Requirements |

1. The system shall allow customers to search for restaurants and cafes.
2. The system shall allow store owners to enlist their shop.
3. The system shall allow customers to view order details (order summery, order total).
4. The system shall allow store owners to add items to the shop menu.
5. The system shall allow store owners to add special offers.
6. The system shall allow customers to apply only one discount code to their order.
7. The system shall allow customers to rate the store from a scale of one to five.

|  |
| --- |
| 6. System and Integration Requirements |

1. The application GUI shall adapt to the operating system of the user's device.
2. The system shall encrypt the user's data using Advanced Encryption Standard (AES).
3. The system shall not violate its security requirements as a result of system maintenance.
4. The system must be able to import, and export excel sheets of all daily orders of customers per shop.

|  |
| --- |
| 7. Quality goals |
|  |

|  |  |
| --- | --- |
| **Requirements** | **McCall's Quality Factors** |
| 1. The user shall be able to complete pre-ordering the food in 3 steps. 2. The user shall be able to learn the functionality of the system in 10 mins. 3. The system shall support accessibility to color blind users by providing alternative method to view texts and other information displayed by the system. 4. The error rate of users submitting their payment details during checkout must not exceed 10 percent. | **Usability** |
| 1. The failure rate of the system during working hours (7:00AM - 9:00PM) must be less than 3 times per month. 2. The system’s downtime shall not exceed 3 minutes per month. | **Reliability** |
| 1. The system shall be able to calculate the total time required to prepare the order with 99% level of accuracy. 2. The probability of a non-accurate total amount including vat (value added tax) won’t exceed 1%. 3. The system shall not take longer than 2 seconds to load the store’s menu. 4. The system’s date format should be specified as date/month/year. | **Correctness** |
| 1. The system MTTR (mean time to repair) shall not take longer than 2 hours. 2. The size of the software module shall not exceed 100 statements. 3. The system software documentation should be clear, self-descriptive, and have a high degree of consistency. | **Maintenance** |
| 1. The system will report attempts by unauthorized persons to obtain customer’s credit card information from the system’s database. 2. Customers should be allowed to view their orders and no other customers orders. They should edit only their orders. 3. The system should allow only the merchants to accept customers' orders. | **Integrity** |
| 1. The system shall not use more than 20 MB storage space from the user’s phone. 2. The system shall not use more than 10% of battery power in two hours of time. 3. At least 20% of the processor capacity available to the system shall be unused during peak hours (from 7a.m to 12 pm). | **Efficiency** |
| 1. The System shall be able to operate on IOS and Android operating systems. | **Portability** |

|  |
| --- |
| 8. Planned Reviews |

The planned project reviews will be conducted for each stage in software development. Planned review activities followed in this project consist of project reviews and inspection reviews.

* **Project Reviews**

**Phase 1: Requirements Gathering and Analysis**

1. Software Requirement Specification Review.
2. Development Plan Review.

**Phase 2: Design**

1. Preliminary Design Review.
2. Detailed Design Review.
3. Software Design Review.
4. Database Design Review.

**Phase 3: Implementation**

1. Post-Implementation Review.
2. Coding Documentation Review.

**Phase 4: Testing**

1. Software Test Procedure Review.
2. Test Plan Review.
3. Test Readiness Review.

**Phase 5: Maintenance**

1. System Audit Review.
2. Maintenance Review.

* **Inspection**

**Phase 1: Requirements Gathering and Analysis**

1. **Software Requirement Specification Document**: are requirements specified consistent with project objectives?
2. **Correctness:** do any requirement conflict with other requirements?
3. **Traceability:** is each requirement correctly identified?
4. **Quality attributes:** are all the nonfunctional requirements complete and quantified?

**Phase 2: Design**

1. **Software Design Document**: are system features clearly documented and designed?
2. **Database Design:** does the data design ensure information preservation?
3. **UML Models:** do the classes in the class diagrams have a corresponding description provided in the architectural design document?

**Phase 3: Implementation**

* + - 1. **Coding Documentation Review**: are all functions in the source code implemented correctly?
      2. **Post-Implementation Review:** Can better data structures or more efficient algorithms be used?

**Phase 4: Testing**

**Test Readiness Review:** do the planned tests verify the system requirements in the SRS document?

* + - 1. **Test Plan Review**: is there sufficient coverage to test functions correctly?

**Phase 5: Maintenance**

**System Auditing:** are all customer orders clearly documented?

1. **Adaptive maintenance:** can the system run correctly with an increasing number of customer orders during peak hours?

|  |
| --- |
| 9. Planned Verification Tests |

**Integration Testing**

* Testing System integration of Mada, SADAD, Apple Pay and VISA Payment System.
* Testing System integration of “Google-maps” to show store location to users.

**Recovery Testing**

* Testing the ability of the system to recover from system failure within a 2 hours time frame.

**Correctness Testing**

* Testing the ability of the system to carry out correct calculations of total price including vat 99% of the time.

**Security Testing**

* Applying inputs that force all error messages.
* Testing access permissions.
* Testing the ability of the system to report attempts by unauthorized persons.

|  |
| --- |
| 10. Planned Validation Tests |

**Requirements based functional testing:**

* The system shall allow store owners to enlist their shop.
  + Testing the ability of store owners to add their shop in the application.
* The system shall allow store owners to add items to the shop menu.
  + Testing the ability of the system to allow store owners to expand their menu.
* The system shall allow customers to apply only one discount code to their order.
  + Testing the ability of the system to apply only one customer discount code per order.
* The system shall allow customers to search for restaurants and cafes.
  + Testing the ability of application customers to search available restaurants and cafes enlisted in the application.

**Usability Testing**

* Testing the ability of users to be able to complete pre-ordering the food in 3 steps.

|  |
| --- |
| 11. Planned Acceptance Tests |

First acceptance testing will be after the **Software Design** phase: There will be a prototype that will be given to the user to try the system.

second acceptance testing will be after the **Implementation** phase: a version of the

application will be given to the user to determine whether the software system has met the requirement specifications.

|  |
| --- |
| 12. Planned configuration management. |

**Software Storage:**

The application will use AWS as a reliable, scalable, and secure space to store the software. It offers Persistent local storage for relational and NoSQL databases, data warehousing, backup and recovery services.

**Security and Backups:**

The application will be using AWS tools to encrypt the data both in transit and at rest to help ensure that only authorized users can access it, using keys managed by AWS Key Management System (KMS).

For the backups, the system will use AWS backup and recovery services using the monthly backup plan.

**Version Control:**

The application will be using the Git version control tool. it offers several functionalities like:

* 1. Code changes can be very easily and clearly tracked.
  2. Easily maintainable and robust.
  3. Offers an amazing command line utility known as git bash.

Git is a Distributed Version Control system, team members don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

|  |
| --- |
| 13. Quality Assurance Process Metrics: |

**Phase 1: Requirements Gathering and Analysis**

* Measuring Requirements Gathering and Analysis phase quality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Metric** | **What it does?** | **Data need to be collected?** |
| Correctness (Qcr) |  | Measures the requirements that have been correctly validated | -Rc: number of requirements that have the same interpretation.  -Rt: the total number of requirements. |

**Phase 2: Design**

* Measuring Design phase quality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Metric** | **What it does?** | **Data need to be collected?** |
| Structural Complexity (Si) |  | Measures the complexity of a module. | the number of modules that are directly invoked. |

**Phase 3: Implementation**

* Measuring Implementation phase quality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Metric** | **What it does?** | **Data need to be collected?** |
| Complexity of a procedure (PC) | PC = LOC \* (FAN-IN \* FANOUT)2 | Measures the complexity of a system module. | -LOC: Lines of Code of the procedure  -Fan-In: number of local flows into that procedure plus the number of data structures from which this procedure retrieves information.  -Fan-Out: number of local flows from that procedure plus the number of data structures which that procedure updates. |

**Phase 4: Testing**

* Measuring Testing phase quality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Metric** | **What it does?** | **Data need to be collected?** |
| Critical Defects Percentage |  | Measures the percentage of critical defects in the software. | -Total number of critical defects.  -Total number of reported defects. |

**Phase 5: Maintenance**

* Measuring Maintenance phase quality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric Name** | **Metric** | **What it does?** | **Data need to be collected?** |
| Average Severity of Software System Failures  (ASSSF) |  | Calculates the software failures detected during a period of one year. | -NYF: Total number of software failures detected during a year of maintenance service.  -WYF: weighted number of yearly software failures detected during a year of maintenance service. |

|  |
| --- |
| 14. Project Team Quality Responsibilities |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Signature | Date |
| Yara Aldosari | All |  | 1/4/2020 |
| Amal Aldoss | All |  | 1/4/2020 |
| Shahad Alsweed | All |  | 1/4/2020 |
| Nouf Alfakhri | All |  | 1/4/2020 |
| Alanoud Alrashed | All |  | 1/4/2020 |