Final Exam Project for SI, DLS and SQ Courses on Software PBA

This project brief includes the overall project description. This is common to all three courses. It also includes specific information for each course's own requirements and a description of the security requirements for the project.

Objectives

The objective of this project is to enable you to demonstrate knowledge and skills acquired in the System Integration, Development of Large Systems, and Software Quality courses.

The project involves design, implementation, and documenting of an integrated software system with a business context.

The following must be submitted on Wiseflow before the exam and will be discussed at the exam.

- DLS – a link to the codebase and a synopsis (a short report of maximum 5 sides)
- SI – a link to the codebase, a video of 10 minutes and a synopsis (a short report of maximum 5 sides)
- SQ – a link to the codebase and a video of 10 minutes

The development of the project will be done in groups, but the examination is individual. Every group member is expected to be able to provide argumentation regarding the whole project and its parts, as well as to highlight their own individual contribution to the solution.

Problem Definition

Your team is responsible for a project on behalf of a food delivery business.

Your group is free to decide on a customer, use cases and implementation scenarios, development environments and integration platforms, within the scope of the requirements listed below.

Task

Your task is to design, develop, implement and deploy a modern integrated software system, which provides business services and automates business processes, related to your customer's domain of activities. To fulfill the task, your team needs to complete the detailed tasks in each section of the brief – SI, DLS and SQ. The following scenario provides an overview of the high-level requirements for the product. Ensure that any requirements in the course-specific parts of the document are kept to, including the type of modelling to be done, architectural, environment and tooling, testing and quality assurance strategies and practices etc. It is important to use the scenario to guide the overall delivery but take account of specific needs of DLS, SI and SQ sections.

Scenario – You should use the scenario provided below

Your customer is a national provider of food delivery services MTOGO operating in multiple languages (currently English and Danish). Customers have an account and can use it to order either

online (web or mobile) or through a telephone ordering service. The food is prepared and delivered by restaurants and food businesses according to their own menu, which can be viewed on the MTOGO Website or mobile app but which MTOGO do not participate in selecting (only displaying).

Restaurants pay a fee to MTOGO for using the service and a variable share of the total order value before VAT (moms) based on order size, from a base of 6% for orders below 101DKK to 3% for orders over 1.000DKK. The business has not agreed a way to model the sliding scale of fees. It could for example be:

First 100 Kr 6%
Amount to 500 Kr 5%
Amount to 1.000 Kr 4%
Anything more 3%

You can choose if this model fits or you want to have a more advanced calculation that reduces the fee gradually rather than in such big steps. Customers do not pay a delivery charge, so all menu prices include free delivery.

MTOGO employs local agents to collect and deliver food. Following a recent court case under the European Human Rights law these agents are now considered employees and bonuses are paid based on order value delivered, late/early working and customer reviews. You are free to decide how big these bonuses should be but remember the amount the company receives, so it must allow the company to still make a profit. You can create your own rules for when a bonus is paid.

Customers receive updates either by SMS or via the MTOGO app as their order progresses through the preparation and delivery process.

All payments are made with the order and the delivery agents do not handle payment at all. Payment processing is provided by an outside supplier, and you are not required to provide integration for that (but you could mock the calling of the service and responses). After delivery customers receive a feedback request and this includes a rating for the food, the overall experience and the individual delivery person. Management wants to see a dashboard of order status (number of open orders, average orders over 24-hour periods, average length of time to process etc.).

At the moment there is no system for handling customer (the person ordering the food) or supplier (restaurant, brewery, catering company etc.) complaints as part of this application but there is a desire to include this at a later date. You need to allow your application to be able to include this new set of services.

Due to the performance requirements for the application, it must be capable of scaling to help maintain growth in the customer base from the current 300.000 people to 1.500.000 people and orders from 3.600.000 to 18.000.000 over the next 5 years.

Requirements

General Requirements

As it is a project with different learning objectives for each course (DLS, SI, SQ), this exam project relates to the content you have learned and the assignments you have worked on during the

semester for each of the courses. Therefore, you must consider the mandatory requirements of your exam project solution as explained in each of the individual course elements below.

Security Requirements

- You will have studied elements of secure application development in the three modules. It is important that you take security into account when developing your solution.

Submission

Each course has its own requirements for documenting the project. Read the requirements for each course carefully.

You must ensure access to all code and relevant documentation for the project. All your code must be collected in one codebase.

Notes

1) This is a group project. The recommended group size is 4 students. Groups larger than this will need prior agreement with all of the teachers.

2) The scope of the project should enable fulfilment of the requirements

3) The project will be delivered to Wiseflow, in terms defined by the administration. This means you will upload links to repositories and any other material for each exam separately.

4) Improvements to the project quality can be made between the delivery date and the exam date, but there shouldn't be fundamental changes.

Exam

The exams are all individual, oral and graded according to the 7-step scale.

System Integration – External Censor

Software Quality – Internal Censor

DLS – External Censor

The exam format will be to discuss your project and there will be questions about the project and the curriculum for 25 minutes including grading.

Important Dates

Hand-in: 09 December 2024 for SI, 6 January 2025 for SQ and 8 January 2025 for DLS

Exam:

SI: 16-17 December 2024 (week 51)

SQ: 09-10 January 2025 (week 2)

DLS: 15-16 January 2025 (week 3)

Individual Course Elements

Development of Large Systems

Tasks

Your task is to apply a development process and use best practices and design techniques relevant to the development of a large, distributed software system. To fulfill the task, your team needs to:

1. Specify requirements for the system and apply techniques for dividing the system into subsystems
2. Use architectural patterns relevant to the development of the system
3. Use a version control system and a system for continuous integration and delivery.

Requirements

The following are mandatory requirements:

1) In the report, explain how Agile methods such as XP and frameworks such as SAFe are used to deliver applications and how business imperatives such as cost, quality and resources drive the development process.

2) From the scenario prepare necessary documentation of the business domain by showing a domain model and how this translates into subsystems. (for example using user stories, use case diagram, sequence diagrams etc.)

3) Provide evidence that you have made use of a CI/CD pipeline in the development and delivery of your solution (you will be asked to demonstrate this at the exam)

4) Describe the advantages and disadvantages of using a CI/CD based approach to delivering software

5) Demonstrate that you understand how scalability is managed through the use of tools such as Terraform. This should be in your report as a discussion of scalability and in the code by using tools such as Kubernetes and Docker Swarm.

6) Show how you have taken notice of quality measurement techniques in the design and development process. This can be done in the report by describing planning techniques (e.g. Story points, planning poker), the role of adopting best practices for team collaboration (e.g. branching and pull requests), domain-driven design and the use of CI/CD tools to automate quality assurance. In the exam you will be asked to demonstrate how you applied ideas such as code reviews and anti-fragile engineering techniques.
7) Explain the role of error handling and use of logging in improving the quality of the application. You should demonstrate this using tools such as Prometheus and Grafana. In the exam you will be expected to describe and demonstrate how logging works and its role in quality assurance and continuous delivery/deployment

System Integration

Tasks

Your task is to implement a modern large integrated software system, which provides business services and automates business processes, related to MTOGO's domain of activities. To fulfill the tasks your team needs to:

Assume the customer has used a legacy software system so far. Analyzing the domain, identify and address some potential problems in legacy systems (e.g. lack of functionality, performance issues, operation difficulties, data storage and transformation issues, customer dissatisfaction, etc.);
Select several business cases and scenarios, where you can help solve the problems by applying integration and modernization techniques; develop your solution ready for demonstration of the integration work you have done Show the development process and the final product in a short video clip.

Build an integrated solution that meets the requirements below

This project's solution needs to show that you have developed competences and skills in system integration. It should provide sufficient evidence of your ability to integrate existing systems in connection with the development of new systems, and to develop new systems, supporting future integration.
The following mandatory requirements to the solution refer to the course content (all points apply):

1. A Composite System

The software system integrates a variety of separate applications, components, and data sources of three types: legacy application (this can be a monolith based on the idea of ports and adapters so have one or two components like order and delivery management and mock other services such as billing), and microservices

2. It Has Business Context

The integration architecture design reflects the business context and implements domain-driven design, and uses some enterprise integration patterns such as ports and filters, and enables storage, integration and transformation of various data sources (use the saga pattern here)

3. Applies Various Integration Technologies

The development implements a large variety of integration, communication, and automation techniques, such as: Message Brokers, APIs and an API Gateway, event-driven integration, microservices composition, discovery, and management tools.

4. Implements Quality Standards and Best Practices

The product demonstrates implementation of decoupling and configuration of components;

Client Application

There are no specific requirements for developing client applications. Publicly available instruments, such as `Swagger`, `Postman`, `curl`, console `CLI`, `htmx`, which provide a basic interface for illustrating the functionality of the integrated system can be used, too.

Documentation

There is *no* requirement for writing a long report. Instead, the group is expected to add

a. in the codebase, a `README.md` file, in which the product is briefly explained
b. A synopsis which includes diagrams, *Including an architectural diagram of the whole system and its components.* The synopsis is a short report of maximum 5 sides.
c. integration development process and considerations are included as part of the synopsis.
b. A 10-minute video, where the business cases, problems and solutions, choices and considerations are discussed, demonstrated, and evaluated by the group members. A link to the video or a copy of the video itself must be available to the examiners from Wiseflow

Software Quality

Final Exam Project Requirements for Software Quality

Course: Software Quality

1. Test Plan and Coverage:

Task: Develop a detailed **test plan** that outlines the testing strategy for the **MTOGO system**. The plan should include:

- **Unit tests** for individual modules like order processing, user accounts, and delivery agent management.
- **Integration tests** for verifying interactions between the restaurant, delivery, and payment services.
- **System tests** for the overall functionality of the food ordering and delivery process.
- **Acceptance tests** that ensure the system meets the requirements for handling various order workflows (e.g., customer feedback, bonuses for delivery agents).

Coverage Requirement: Set and justify a **minimum test coverage goal** (For example, 65% coverage) for the project.

- Use tools like **JaCoCo** (Java) or **Coverlet** (C#) to measure and document the coverage.
- The coverage should focus on business-critical areas such as order placement, customer notifications, and delivery status updates.

2. Automated Testing Setup:Task: Set up automated testing as part of the **CI/CD pipeline** for the MTOGO system.

- Use tools like **Maven** (for Java) or equivalent C# tools to integrate **unit testing**, **integration testing**, and **regression testing**.
- Ensure that the tests are triggered automatically with every code change (e.g., through GitHub Actions or Jenkins).
- Implement **mocking** frameworks (e.g., **Mockito**, **EasyMock**, or **Moq**) to simulate external services like payment processors and delivery status notifications.

3. Regression Testing:

Task: Implement **regression testing** to ensure that adding new features to the MTOGO system (e.g., scaling to handle larger customer bases, new bonus calculations for agents) does not introduce bugs or break existing functionality.

Demonstrate how the regression test suite is maintained and executed during the development process, ensuring that new updates to the ordering system (e.g., changes to customer feedback handling) do not affect core functionalities.

4. Validation and Verification of Tests:

Task: For each level of testing (unit, integration, system, and acceptance), demonstrate how the tests validate that the MTOGO system works as intended.

Ensure that your tests verify:

- Correct processing of customer orders.
- Accurate communication between the payment service and the delivery service.
- Correct handling of delivery agent bonuses based on order reviews and time metrics.

Apply **Equivalence Partitioning** and **Boundary Value Analysis** to critical areas such as:

- The calculation of order fees (e.g., varying fees based on order size).
- Delivery agent bonuses based on customer reviews (e.g., review scores ranging from 1 to 5).

5. Static Code Analysis and Code Reviews:

Task: Use **PMD** (Java) or **FxCop** (C#) to run static code analysis on the MTOGO codebase.

- Identify code quality issues such as performance inefficiencies, security vulnerabilities, and best practice violations.
- Optionally, integrate **SonarQube** for more comprehensive analysis, including detecting **code smells** and **duplicated code**.

Perform a **peer code review** based on the guidelines from the **Code Review** and **Software Review** articles.

- Focus on improving code readability, maintainability, and testability.

- Document any changes made to the code following the review process

6. Taint analysis

Task: Use Taint Analysis in the coding process to ensure input and output are handled correctly.

- Identify all potential sources of untrusted input, such as user input fields, APIs, or external files.
- Track the flow of this untrusted data through the code to ensure it is properly sanitized and validated before use.
- Ensure that all outputs derived from untrusted inputs are properly encoded and safe, preventing injection vulnerabilities or data leakage.

Task: Implement secure handling of data throughout the entire application lifecycle.

- Ensure all input is validated at entry points.
- Sanitize and escape potentially harmful characters or data.
- Conduct regular audits to ensure taint analysis processes remain effective and up to date.

7. Design patterns

Task: Select the appropriate design patterns throughout the coding process.

- Identify recurring problems in the codebase and map them to relevant design patterns (e.g., Singleton, Factory, Adapter...).
- Implement the selected design patterns, ensuring the code remains modular, reusable, and maintainable.
- Regularly review and refactor the code to ensure the chosen design patterns are still optimal as the project evolves.

6. Code refactoring

Task: Use the technique of code refactoring throughout the coding process.

- Continuously identify and improve inefficient, redundant, or complex code structures without altering functionality.
- Break down large functions or classes into smaller, more manageable components for better readability and maintainability.
- Replace hardcoded values and magic numbers with constants or configuration settings to improve code clarity and flexibility.

Task: Ensure the code remains clean and efficient through regular refactoring.

- Eliminate code duplication by abstracting common logic into reusable methods or classes.
- Simplify complex conditionals and loops to improve readability and reduce the chance of errors.
- Perform thorough testing after each refactoring step to ensure that no new bugs or regressions are introduced.

General Deliverables:

Ensure the time is evenly distributed in the video so that each topic mentioned in the list below is thoroughly covered.

- A **Git repository** containing:
    - Full source code for the MTOGO system.
    - Test code (unit, integration, system, and acceptance tests).
    - CI/CD pipeline configuration for automated testing.
- **A video outlining the following from your project:**
    - **A demonstration of your project**
    - **A demonstration of your CI/CD pipeline**
    - The overall test plan.
    - Evidence of unit, integration, system, and acceptance testing.
    - The regression testing strategy.
    - Demonstrate the application of taint analysis during the code evaluation process.
    - Showcase the incorporation of design patterns in the final codebase, detailing the relevant implementations and justifying their significance.
    - Provide evidence of a structured code refactoring process, highlighting its overall benefits.
    - Reflection on test effectiveness, coverage, and improvements made during the development process.
    - showing the results of static code analysis and the changes made to address key issues.
    - **SonarQube report** (optional) for more comprehensive code quality analysis.