McGill

Department of Electrical and Computer Engineering
ECSE 321 Introduction to Software Engineering
Fall 2021

Project Deliverable 2: Backend Services, Behavior Modeling, and Testing

This deliverable consists of the following parts:

1 Backend Implementation and Evolution of Persistence Layer

As the main technical deliverable, you need to **implement all use cases as business methods** and make them **available as RESTful services** using Java Spring technology. The services need to persist data by using the persistence layer developed in Sprint 1. Should the implementation of the services require any adaptations of the persistence layer or the domain model, you need to evolve that part of your system as well in Sprint 2. Finally, you need to provide brief **documentation for each of your RESTful service endpoints** that can be called externally on the project wiki (e.g., precise URL of service endpoint, accepted operation, parameters, description of functionality).

2 Software Quality Assurance Plan and Report

You need to provide a plan for software quality assurance in the project wiki. This plan should **define appropriate test coverage criteria** you wish to achieve for the business methods implemented in the backend. Your test plan should document **which unit testing approaches you used** to achieve the target coverage criteria. Moreover, you should **measure and report the test coverage** achieved by your unit test suite (see below). The software quality assurance plan should also provide details if your team is using **any other means of quality assurance** (e.g., code reviews, coding conventions, static analysis tools, etc.).

3 Unit Testing of Backend

You need to develop a **test suite for unit testing** of the business methods **using the JUnit** framework. These unit tests should be **executed in isolation**, i.e., they must be separated from the underlying database. The coverage of the test suite must be measured and reported (see above).

4 Integration Testing of Backend Services

As part of integration testing, you should also demonstrate the **successful execution of all your RESTful services** by developing one (interface) test for each of your own services.

5 Build System, Continuous Integration, and Delivery

A build system is needed to automate the process of compiling, packaging, and deploying all code developed for the backend (including the persistence layer developed in Sprint 1). The continuous integration pipeline on GitHub Actions is expected to successfully compile, build, run the unit test suite of the backend services of your application, and deploy a new version of your backend application using Heroku. Note that integration tests are not required to run on GitHub Actions, but you need to define a specific Gradle task runnable with *./gradlew integrationTest* that runs the integration tests. Furthermore, persistence layer tests from Sprint 1 are also expected to run on the CI server using a test database.

Every push to the GitHub repository should trigger a build job on GitHub Actions to initiate continuous integration. Only the main/master branch should be delivered to Heroku. The build system and the CI specification must conform to the Technological Constraints above.

6 Project Management and Project Report

A key aspect of agile software development is the use of a project backlog to coordinate development and project documentation activity. Your group is expected to make use of the issue tracking features on GitHub to create and manage the project backlog. Each issue needs to have an assignee to trace core responsibilities within your team. Your team should provide a welcome page that introduces all group members and describes the main scope of the project in the *README.md* file in the root of your team's repository. In addition, the *README.md* file should contain an overview table with names, team roles, and individual efforts (in hours) with separated entries for each deliverable. Project Deliverable 2 shall be accompanied with a succinct **project report as part of the project wiki** which records the meeting minutes, the key design decisions taken by your team, and the success spectrum exercise. This project report should be navigable from the *README.md* file. Altogether, your team should comply with all the Technological Constraints.

Submission

Your team will be assigned a private repository within GitHub, which should be used for this project. The course staff also has access to your GitHub repository. For Deliverable 2, your team is required to submit a commit link (i.e., a URL representing your last GitHub commit that counts as a deliverable) by ***Monday, Nov. 8 at 11:59pm***. Each team member must make contributions to the deliverable. A team member who does not contribute to the deliverable receives a mark of 0 for the deliverable.

Marking Scheme (10% of total score)

Component	***Points***
Implementation of backend services (business methods + RESTful services) including functionality and coding style	35
Software quality plan and report	10
Unit test suite of business methods	15
Integration tests for RESTful services	10
Build system + Continuous integration (Gradle, GitHub Actions, Heroku)	15
Project management (e.g., backlog, issues, sprint planning, teamwork report, documentation quality)	15
Total Marks:	100

Note: The total mark may be adjusted based on the actual contributions of a team member to the deliverable.

Detailed Marking Scheme

Implementation of backend services	**35**
Use-cases are covered by business methods	3
Coverage and richness of functionality	5
Business methods are exposed via RESTful API	3
Create methods are exposed via the RESTful API	3
Delete methods are exposed via the RESTful API	2
Simple read methods are exposed via the RESTful API	2
Custom query methods are exposed via the RESTful API	3
Persistence layer is successfully used	3
Transactions are properly handled	2
There are comments for nontrivial code blocks	1
Code is free from unused variables and unused imports	1
Methods are not too long (approx. 20 lines)	1
Naming of variables, methods follow conventions	1
It is well-documented how to call RESTful service endpoints (URL, params)	3
It is well-documented what RESTful service endpoints provide as result	2
Software quality plan and report	**10**
Test plan is structured properly and clearly mentions the scope	3
Test plan includes test coverage criteria of the team	3
The team successfully applied unit testing approaches	4
Unit test suite of business methods	**15**
Complexity of tests for the business methods	3
Unit tests investigate edge cases / boundary values	2
Functionality / correctness of unit test case implementations	2
Side effects of test code are reverted upon completion (clear-up)	2
Database mocks are used for unit testing of business methods	3
Technology is properly used (JUnit, Mockito)	2
Test code follows coding conventions (e.g., names of tests, etc.)	1
Integration tests for RESTful services	**10**
Tests are present for create methods of REST API	4
Tests are present for delete methods of REST API	3
Tests are present for update methods of REST API	3
Build system and CI specification	**15**
It is documented how to run the application and where it is deployed to	2
A .GitHub Actions.yml build file for the main branch is present (attempted CI)	2
The main branch build succeeds on GitHub Actions - project is properly built	1
Unit testing succeeds on GitHub Actions	2
The project builds locally with Gradle (gradle build -xtest or ./gradlew build -xtest)	1
Tests run and pass locally	2
Project is automatically deployed to Heroku	3
Gradle task for running integration tests is added	2
Project Management and Project Report	**15**
GitHub is properly used for storing sources	1
GitHub branches are properly used	1
Commits are assigned to issues	1
Backlog is maintained in GitHub Projects	1
Issues are properly categorized / tagged	1
Issues are assigned to milestones	1
Issues are assigned to responsible person	2
Project deliverable report provided on wiki	1
Lifecycle of issues is continuously managed in GitHub	2
Project Report has welcome page (introduction to group, scope of project)	1
Documentation of team operation (minutes of meeting recorded, design decisions recorded)	1
Individual roles are detailed	1
Individual efforts are summarized in the table	1
TOTAL	**100**