Department of Electrical and Computer Engineering
ECSE 321 Introduction to Software Engineering
Winter 2020

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** for your viewpoint **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 backend 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 a 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 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 Travis-CI 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 Travis CI, 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 Travis CI to initiate continuous integration and delivery. 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. Each group is expected to **continuously use** the **issue tracking** features on GitHub and an **agile project board** in order to create and manage the project backlog. Each issue needs to have an assignee to trace core responsibilities within the team.

The team should provide a welcome page that introduces the team and describes the main scope of the project in the README.md file in the root of each 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 also accompanied with a succinct *project report* as *part of the project wiki* which records the meeting minutes and the key design decisions taken by the team. This project report should be navigable from the README.md file of the project. Altogether, the team should comply with all Technological Constraints mentioned in the project description.

Submission

The project will be carried out with the same teams as you worked with for Deliverable 1. 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 ***Sunday, March 1st, 2020 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. A team member may optionally email a confidential statement of work to the instructor before the due date of the deliverable. A statement of work first lists in point form the parts of the deliverable to which the team member contributed. In addition, the statement of work also describes whether the workload was distributed fairly evenly among the team members. A statement of work may be used to adjust the mark of a team member who is not contributing sufficiently to the deliverable. It is not necessary to send a statement of work, if a team distributed the work for the deliverable fairly evenly and each team member contributed sufficiently.

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, Travis CI, 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.