Group 7

Hilda Hermunen, Veera Ruotsalainen, Patrick Scott

Sprint 6

# Acceptance Testing Plan

1. Introduction

   Acceptance testing is a quality assurance process that evaluates whether the product meets the required specifications. It can take many different forms, depending on the product. The goal is to evaluate the system in a real-world scenario from the perspective of the end user.

   The group's project called StudyFlow is a program where a user can add different events to a timetable with possibilities to update and delete the events. The acceptance testing should focus on core user workflows mentioned before.

2. Defining Acceptance Criteria

   User interface should be intuitive to use. Users should be able to add different events, such as class schedule, assignments, exams and personal study sessions. Users should be able to update and delete any added event. The timetable has a week view, and users should be able to change the week, and the timetable should display correct events related to that specific week. The user should be able to choose a language setting: English, Korean, or Arabic.

3. Designing Acceptance Test
   3.1 Functional Tests

   Functional tests verify the core functionalities described in the previous section. They should ensure that both the code and the database work as intended. Functional tests can be written with JUnit and Mockito.

   The program should be able to add new events to the database, retrieve that data from the database and then display it in the user interface. Program should be able to update any data of any event, and that updated data should be correctly changed in database and user interface. The program should be able to delete any event from the database and user interface.

## 3.2 Usability Tests

Usability tests involve real users performing common tasks on the system while being observed. Usability tests can reveal pain points in the user interface. The objective is to ensure that the user interface is intuitive and meets the user expectations. Since the target users are students with no required technical background, the system should be simple and intuitive to use, even for those with limited technological experience. The tests should include students with varying levels of technological skills.

The user should be able to use the core functionalities in their chosen language without confusion. Adding, updating and deleting events should be straightforward. Possible user input errors should be handled gracefully, with clear feedback to help user correct them. The user should be able to easily change the language at any point from the options that the user interface given.

## 3.3 Performance Tests

Performance testing measures response time, system throughput, and resource consumption under different load levels. Since this group project is designed for local use by a single user, performance testing should focus on responsiveness, efficiency, and memory usage.

Responsiveness testing should include measuring startup time, as well as the time it takes to add, update, and delete events. These tests can be implemented using JUnit in combination with a stopwatch.

Efficiency testing evaluates how well the application handles large amounts of data. The program should continue to function smoothly and without significant delays, even when working with a large dataset. These tests can be implemented using JUnit in combination with a stopwatch.

Memory usage testing focuses on identifying memory leaks and verifying that memory is released properly after deleting or modifying data. This can be done using JUnit and the Runtime class to measure memory usage before and after specific operations.