HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM384 SOFTWARE ENGINEERING LABORATORY
EXPERIMENT SHEET
2019 Spring

Instructors: Asst. Prof. Dr. Ayça TARHAN, Dr. Tuğba Gürgen ERDOĞAN

Teaching Assistants: R. A. Burcu YALÇINER, R. A. Pelin CANBAY, R. A. Nebi YILMAZ

1. Purpose and Scope

The purpose of this experiment sheet is to guide students who enroll BBM384 course in developing their laboratory projects. The projects are pre-defined and the process defined in this document is applied through the development of the projects. While applying the process, the students follow the schedule given by the Instructor at the beginning of the course.

2. Abbreviations

SRS	:	Software Development Life Cycle
SRS	:	Software Requirements Specification
TCD	:	Test Case Definition
SDD	:	Software Design Description
STR	:	Software Test Report
GUI	:	Graphical User Interface

3. Roles and Responsibilities

Roles	**Responsibilities**
Student	Responsible for conforming the requirements of this process, and for timely generation of the outputs of this process.
Teaching Assistant(s)	Responsible for timely evaluation and feedback of the outputs generated by the students.
Instructor(s)	Responsible for resolution of issues likely to occur between teaching assistants and the students.
Stakeholder	Any person that has a relation with the system to be developed (including students, teaching assistants, and instructors).
Team Member	A student that takes role in the development of the system. Undertaken roles include Software Project Manager, Software Analyst, Software Architect, Software Developer, Software Change/Configuration Manager, and Software Tester. Each student must have Software Developer role during development. A student may have more than one role during development.

4. **Inputs**

(i) Vision Document Template
(ii) Project Plan Template
(iii) Software Requirements Specification Template
(iv) Use Case Definition Template
(v) Test Case Definition Template
(vi) Graphical User Interface Design Template
(vii) Architecture Notebook Template
(viii) Risk Management Report Template
(ix) Configuration/Change Management Report Template
(x) Software Design Description Template
(xi) Software Coding Standard
(xii) Test Script Template
(xiii) Software Test Report Template
(xiv) Github Flow: https://guides.github.com/introduction/flow/

NOTE:

- Risk Management Report will be created by using the lists on the page: https://docs.google.com/spreadsheets/d/1J0feW4nEGAq4IjDPz4ALfEJutSmhEgq8y65orrQgJAc/edit#gid=1278480004
- Configuration/Change Management Report will be created by using the lists on the page: https://docs.google.com/spreadsheets/d/1J0feW4nEGAq4IjDPz4ALfEJutSmhEgq8y65orrQgJAc/edit#gid=670414547

5. **Outputs**

(i) Software Vision Document
(ii) Software Project Plan
(iii) Software Requirements Specification (SRS)
(iv) Use Case Definitions (as an attachment to SRS)
(v) Graphical User Interface Definitions (as an attachment to SRS)
(vi) Test Case Definitions (as an attachment to SRS)
(vii) Architecture Notebook
(viii) Risk Management Report
(ix) Configuration/Change Management Report
(x) Software Design Document
(xi) Software Code (and Software Coding Standard if modified)
(xii) Software Test Report (STR)
(xiii) Test Case and Test Script Definitions (as an attachment to STR)
(xiv) Project's Github Repository (master and member branches, issues, pull requests, commits etc.)

6. **System to be developed:** You are expected to develop an online shopping system such as Morhipo, Trendyol etc. The expected online shopping system is such platform that enables the customers, who are getting involved in the system, do online shopping of clothes, shoes, electronic devices, accessory, personal care products etc. within their budgets, without visiting shops, e.g., in bad weather conditions and not wasting their limited time. While developing this system, you are expected to develop a flexible, maintainable, attractive and extensible system. Your application should provide Open-Close Principle (i.e. it should be close for modification and open for extension).

All software development activities will be done as a Github project. Github is the best way to build software together. Each member must create a branch where s/he can safely experiment and make changes. S/he will use a pull request to get feedback on his/her changes. After the changes are reviewed, s/he will merge them into master branch and deploy his/her code to the project.

System modules: The online shopping system includes four main modules. These are: administration module, customer module, seller module, and payment module.

Administration Module:

- Manages sellers who are providing products to their customers,
- Manages services that are provided to the customers,
- Manages menus with new products added by the sellers,
- Sends messages and/or e-mail when there is a promotion,
- Shows advertisement to the customers depending on his/her shopping search history,
- Gets customer details, which are personal information entered to the system by the customer, including payment status, shopping history of the customer, total amount of the shopping etc.,
- Manages account of working employees, and
- Confirms the sales.

Customer Module:

- There are two types of customers; registered customers and non-registered customers.
- Non-registered customers can only search and view products.
- Registered customers have an account for privacy and security purposes.
- The registered customers can access/manage their accounts, search/view the products, add the products selected to their shopping carts and purchase them.
- The registered customers should check their delivery status and compare products offered by different sellers.

Seller Module:

- Responsible for adding new products to the online shopping system.
- Manages its account.
- Views its product status to update and control their stocks, deletes the product which is out of stock, and adds new products to the system.

Payment Module:

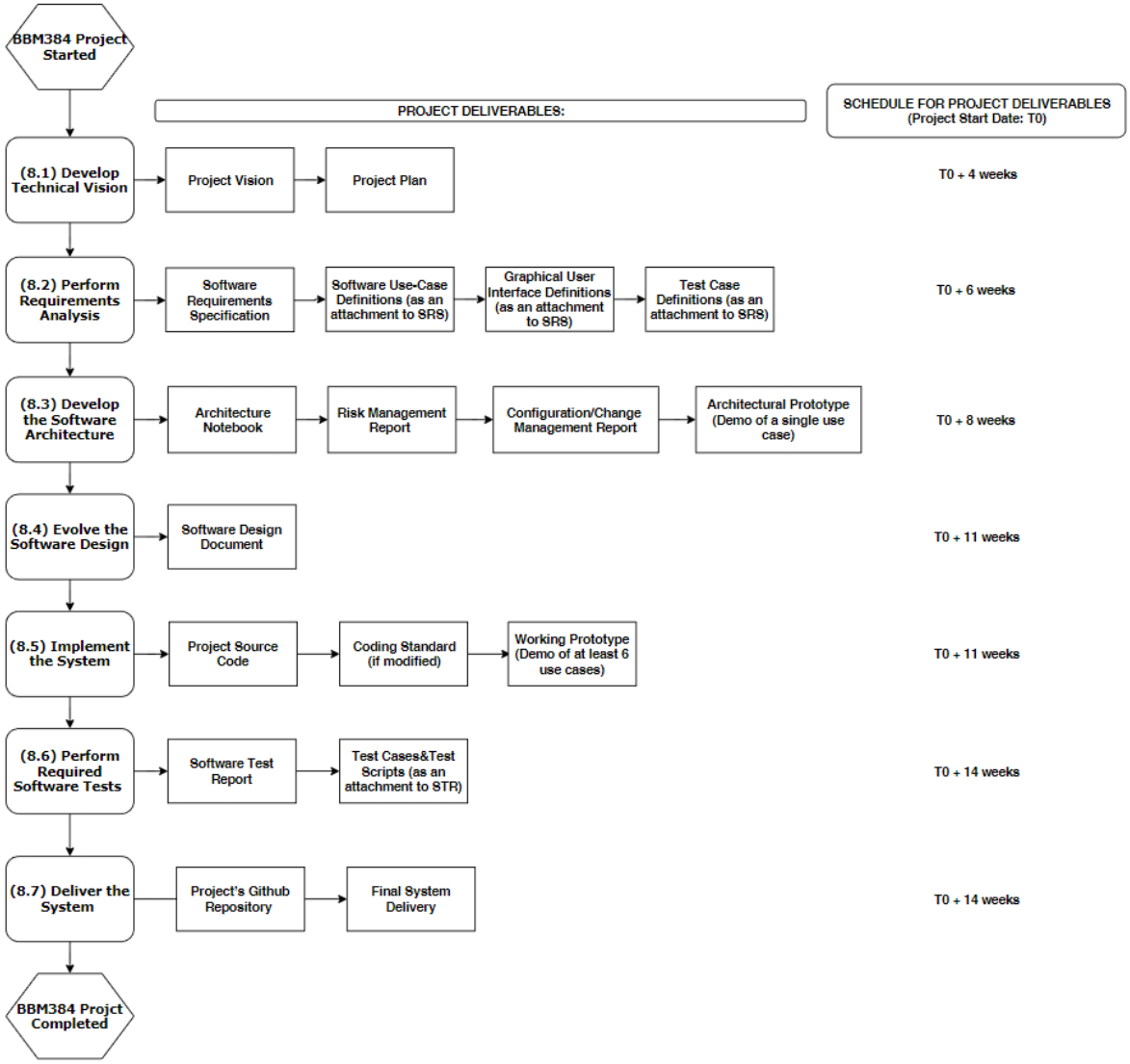
- It is a third party application which is used for making secure payment.
- There are different payment services such as Credit Card Payment Service, PayPal, and Transferring Money via PTT etc.
- This third party application is a user of your software system to be developed, yet an already existent application and will not be developed in the context of your software project. It is an external application and will be used for enabling the customers to make online payment for their purchased products.

- When the customer clicks on the purchase button, s/he is transferred to https in place of http and in terms of trusted and verified certificates of payment module, and the customers should be assured of their security and privacy.

System Development Constraints

- The developed system can be either a web application or a mobile application.
- Each group must follow Github Flow to keep change and configuration management activities: https://guides.github.com/introduction/flow/. Each group member will create own branch, get feedback for its own code by creating pull requests, and deploy own code to master branch.
- Model-View-Controller (MVC) architectural pattern will be used for this system.
- JAVA programming language must be used to developed the system.
- The student enrolled to the course can use any database management system.
- JAVA programming language can be used to provide connection between the database and the software code.

7. Process Flow



8. Process Activities

8.1. Develop Technical Vision & Project Plan

This activity includes the following tasks.

(i) Develop Technical Vision

The students define the vision for the future system by describing the problem and features based on stakeholder requests. The students document the vision as conformant to *Vision Document Template*.

(ii) Develop Project Plan

The students plan the system development by describing the content of the system, project objectives, project organization, development parameters (effort and cost), development process steps and project milestones. The project plan is documented as conformant to *Project Plan Template*.

8.2. Perform Requirements Analysis

The students analyze the requirements of the system to be developed. The students apply use-case analysis to identify the requirements of the system and document the requirements as conformant to *Software Requirements Specification Template*.

The students document uses cases as conformant to *Use Case Definition Template* or draw activity diagram for each uses case. The students also define graphical user interfaces that the users will interact while executing use cases, as conformant to *Graphical User Interface Design Template*. The students will submit both documents as an attachment to SRS.

At the end of the analysis, the students also define test cases and data for critical use-case scenarios as basis for functional and acceptance tests. The students document test cases as conformant to *Test Case Definition Template* and submit the document as an attachment to SRS.

This activity includes the following tasks.

(i) Identify and Outline Requirements

This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.

The purpose of this task is to identify and capture functional and non-functional requirements for the system. These requirements form the basis of communication and agreement between the stakeholders and the development team on what the system must do to satisfy stakeholder needs. The goal is to understand the requirements at a high-level so that the initial scope of work can be determined.

(ii) Detail Use Case Scenarios

This task describes how to detail use-case scenarios for the system.

The purpose of this task is to describe use-case scenarios in sufficient detail to validate understanding of the requirements, to ensure concurrence with stakeholder expectations, and to permit software development to begin.

(iii) Detail System-Wide Requirements

This task details one or more requirement that does not apply to a specific use case.

The purpose of this task is to describe one or more system-wide requirements (that could not be captured by use-cases and scenarios) in sufficient detail to validate understanding of the requirements, to ensure concurrence with stakeholder expectations, and to permit software development to begin.

(iv) Define Test Cases

This task requires development of the test cases and test data for the requirements to be tested.

The purpose of this task is to achieve a shared understanding of the specific conditions that the solution must meet.

8.3. Develop the Architecture

The students design, implement, test, and integrate the solution for a number of critical use-case scenarios, and provide a demo of this solution.

The purpose of this activity is to propose a solution to meet the requirements of the system to be developed, and to prove a working prototype of the solution.

The students document architectural details as conformant to *Architecture Notebook Template*, and provide a demo of the working prototype of their solution.

This activity includes the following tasks.

(i) Design the Solution

The students identify the elements and devise the interactions, behavior, relations, and data necessary to realize some functionality. The students render the design visually to aid in solving the problem and communicating the solution.

The purpose of this task is to describe the elements of the system so that they support the required behavior, are of high quality, and fit within the architecture.

(ii) Implement Developer Tests

The students implement one or more tests that enable the validation of the individual implementation elements through execution.

The purpose of this task is to prepare to validate an implementation element (e.g. an operation, a class, a stored procedure) through unit testing. The result is one or more new developer tests.

Developer testing is different from other forms of testing in that it is based on the expected behavior of code units rather than being directly based on the system requirements.

(iii) Implement Solution

The students implement source code to provide new functionality or fix defects.

The purpose of this task is to produce an implementation for part of the solution (such as a use-case scenario, a class, or component), or to fix one or more defects. The result is typically new or modified source code, which is referred to the implementation.

(iv) Run Developer Tests

The students run tests against the individual implementation elements to verify that their internal structures work as specified.

The purpose of this task is to verify that the implementation works as specified.

(v) Integrate and Create Build

This task describes how to integrate all changes made by developers into the code base and perform the minimal testing to validate the build.

The purpose of this task is to integrate all changes made by all developers into the code base and perform the minimal testing on the system increment in order to validate the build. The goal is to identify integration issues as soon as possible, so they can be corrected easily by the right person, at the right time.

(vi) Refine the Architecture

After performing the tasks above and implementing a working prototype of the architecture, the students may refine the architecture to an appropriate level of detail to support development.

8.4. Evolve the Design

After demonstrating a working architecture of the system and refining the architecture as defined in activity 8.3, the students evolve the architecture into a complete design of the system. The students document system design as conformant to *Software Design Description Template*.

8.5. Implement the System

After proving a prototype of system architecture and defining the design, the students implement the system to meet entire set of system requirements including use case scenarios, system-wide requirements, and non-functional requirements. The students use

Software Requirements Specification and follow Software Design Description to carry out this activity.

The students may implement the system as a whole or in iterations. In case of iterative development, the project schedule announced for the delivery of working software iterations will be followed.

The students are expected to develop and follow a Coding Standard while implementing the system. Here is an example *Java Coding Standard*. The students may readily utilize this standard or may define a new one according to their preferences.

8.6. Test the System

The students, from a system perspective, test and evaluate the developed requirements. The students develop test scripts for the selected test cases and document test scripts as conformant to *Test Script Template*. The students run the tests as defined in the test scripts and document test results as conformant to *Software Test Report Template*.

This activity includes the following tasks. If the students develop the system in iterations, they typically apply these tasks for each iteration.

(i) Implement Tests

The students implement Test Scripts to validate a Build of the solution. The students organize Test Scripts into suites, and collaborate to ensure appropriate depth and breadth of test feedback.

The purpose of this task is to implement step-by-step Test Scripts that demonstrate the solution satisfies the requirements. The students use Test Case Definition created by activity 8.2 (and delivered as an attachment to SRS) as input to this task.

(ii) Run Tests

The students run the appropriate tests scripts, analyze results, articulate issues, and document test results.

The purpose of this task is to provide feedback about how well a build satisfies the requirements.

8.7. Deliver the System

The students deliver a complete, defect-free system that meets all the requirements specified in Software Requirement Specification. The students provide source code and relevant data for the system in execution.

The acceptance of the final system is carried out by Teaching Assistants under the supervision of the Instructor. The assistants may run different tests, in addition to the ones specified in the Software Test Report, on the system during acceptance. It is recommended that students ensure defect-free execution of their systems prior to delivery.

9. References

(i) OpenUP Process Framework
https://www.eclipse.org/epf/openup_component/openup_vision.php

(ii) IEEE std 830-1998 Recommended Practice for Software Requirements Specifications

(iii) IEEE std 1016-1998 Recommended Practice for Software Design Descriptions

(iv) IEEE std 829-1998 Standard for Software Test Documentation