HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM384 SOFTWARE ENGINEERING LABORATORY
EXPERIMENT SHEET
2021 Spring

Instructors: Assoc. Prof. Dr. Ayça KOLUKISA TARHAN, Dr. Tuğba Gürgen ERDOĞAN

Teaching Assistants: R. A. Burcu YALÇINER, Dr. Tuğba Gürgen ERDOĞAN

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

SDLC	:	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/1Ar2n9xVvJZnJp8qWdupdZZY14bhXhUwF/edit#gid=768757863
- Configuration/Change Management Report will be created by using the lists on the page: https://docs.google.com/spreadsheets/d/1Ar2n9xVvJZnJp8qWdupdZZY14bhXhUwF/edit#gid=9565548

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 Social Interest eClub (SIeC) system such as clubs in Facebook etc. The expected online Social Interest eClub system is such platform that enables users, who are getting involved in the system via Internet, can find social friends and can join social clubs according to their interests and hobbies.

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 Social Interest eClub system includes three main modules. These are: administration module, member module and social club admin module.

Administration Module:

- Logins to the system by entering his/her login id and password.
- Adds different types of club categories (book discussion club, theatre club, trekking club, cinema club, chess club, video games club etc.), sub-categories (e.g.: poetry discussion, essay discussion, literature discussion; pantomime, performance art; action films, fantasy films, comedy films; action-adventure games, strategy games, simulation games) to the Social Interest eClub system.
- Establishes questionnaire about social clubs in order to determine the member's interests. After the user registration, the system should present this questionnaire to the user and propose social clubs to the user regarding the user's answer.
    - The system should calculate how much interest the user is in each social club according to his/her answer results and this calculation should be made as a percentage calculation. If the percentage result is greater then 50, then the system should offer this club to the user. Otherwise, the system should not allow the user to enroll the club.
- Takes new club requests which are not defined in the system from the members.
    - If the request for a certain club is greater than or equal to 3, s/he adds the club and its sub-clubs to the system.
- Deletes the sub-clubs which are not used 90 days or more and should give. information to the sub-club members before deleting it by sending e-mail to the members.
- Manages his/her account and member accounts.
- Gets member details, which are personal information entered to the system by the member, including interest status.
- Manages the social clubs and sub-clubs provided to the members.
- Shows new opened clubs to the members and offers new questions about this clubs to the members. Updates his/her interest status according to answers s/he will give.
- If the sub-club admin uses slang words in public/private messages, or during the online/offline events or be complaint about him/her, the admin should ban this sub-club admin. The status of the banned sub-club admin is changed from admin status to regular member status. Therefore, s/he cannot be admin of any sub-clubs any further.

Member Module:

- There are two types of members; registered members and non-registered members.
- Non-registered members can only search and view social clubs and their sub-clubs.
- Registered members have an account for privacy and security purposes.

- Registered members should answer the questionnaire after sign-up.
- The registered members can access/manage their accounts, offer new clubs and/or sub-clubs, can enroll the clubs if his/her interest rate is greater than or equal to 50%, can send public and private text messages to the members in the same club.
- The registered member can check their interest status and view the new opened/ recommended clubs/sub-clubs by the system.
- The registered member can rate clubs/sub-clubs and upload comments about them.
- The registered member can ask to be the admin of the sub-club s/he enrolled.

Social Sub-club Admin Module:

- Each social sub-club should have an admin.
- Social sub-club admin is selected randomly from the members who have sent requests to be the admin of the sub-club.
- S/he creates online and/or offline events.
  - Online events and offline events should be scheduled by determining the member's availability via some online tools like when2meet.
  - Online events should be done by using online tools such as Zoom, Google Meet etc.
- S/he manages the members of the sub-club.
- If the member uses slang words in public/private messages, or during the online/offline events or be complaint about him/her, the sub-club admin should ban this member.
  - The banned member cannot join this sub-club for the following 5 days and cannot join the following online and offline events.
  - If the member is banned more than 3 times, the sub-club admin can dismiss him/her from the sub-club.

Further Functional Requirements List for the Social Interest eClub System (in addition to the above)

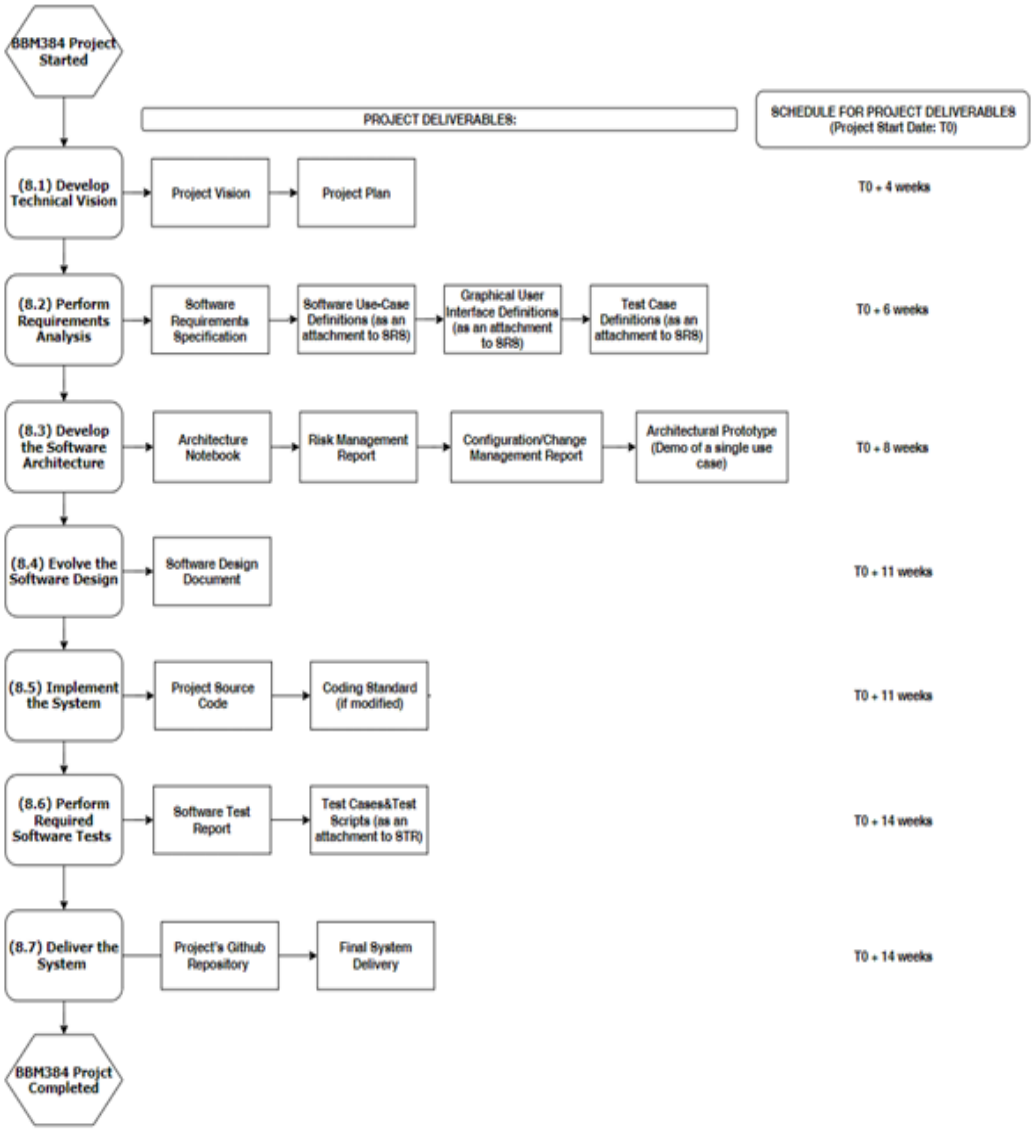
- The system shall give a warning when the admin/member/social club admin enters invalid/ wrong login id and/or password.
- The system should send e-mail to the user when the user clicks the "forgot password" button.
- The system shall provide password protection for admin account, sub-club admin and member accounts.
- The system shall provide the ability to search the clubs and sub-clubs.
- The system shall provide the ability to search the members in the same sub-clubs.
- The system shall provide the ability to send private/public messages to the members in the same sub-clubs.
- The system shall maintain reviews of clubs/sub-clubs and ratings on clubs/sub-clubs.

NOTE: You may add additional functional requirements different from the described above and must add non-functional requirements.

System Development Constraints

- The system to develop can be either a web application or a mobile application - not both!
- There must be minimum 9 and maximum 12 use-cases in your system to develop.
- Each group must follow GitHub Flow to keep change and configuration management activities: https://guides.github.com/introduction/flow/
  - Each group member must 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 must be used for this system.
- JAVA or C# programming language must be used to developed the system.
- For frontend development, groups can use some frameworks such as React, Angular etc.
- Groups can use any database management system such as Oracle, MySQL, SQL Server.
- JAVA or C# programming language must 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) IEEE std 830-1998 Recommended Practice for Software Requirements Specifications
(ii) IEEE std 1016-1998 Recommended Practice for Software Design Descriptions
(iii) IEEE std 829-1998 Standard for Software Test Documentation