School of Computing Sciences

Module: CMP-5012B Software Engineering
Assignment: Project

Set by: Jaejoon Lee – jaejoon.lee@uea.ac.uk
Checked by: Deborah Taylor – Debbie.Taylor@uea.ac.uk
Date set: 31 January 2022
Value: 90% (10% is allocated to the Synoptic Project.)

Date due: Project Requirements and Design Report (45%): 11/03/2022
- Report – 100% (*team submission*)

* You can find the details on the marking scheme.

Team Demo and Individual Reflective Report (45%): 20/05/2022
- Demo and Code: 90% (*team submission*)
- Reflective Report: 10% (*individual submission*)

* Team demos will be organised in Week 12.

Returned: Within 20 working days of submission/demo
Submission: All Reports and Source Code: Blackboard

Aims

- Requirements analysis and design of a software system:
  - Undertake requirements analysis and OO design for an extensive team-based software engineering project.
  - Understand the architectural and design principles underpinning robust software systems.
- Implement of design into software using an iterative and incremental development approach.
- Use team-based version control tools (e.g. Git) for distributed and collaborative software development.

Learning outcomes

- Software Design: Describe the objectives of software design in terms of software requirements and quality attributes
- Software Modelling: Develop UML models including class diagrams, use cases, sequence diagrams, state diagrams
- Software Model and Code: Understand the relationship between UML models and code, and transfer the diagrams into code
- Teamwork: Work together in teams in order to tackle technical problems in a (group) project context
- Project Management: Develop skills for coping with and managing parallel implementation tasks

School of Computing Sciences

Project Specification

Overview

The goal of the project is to give you experience of a team-based development project in a realistic setting such as you might encounter it in an industrial environment. In this project you can apply the knowledge you learned during the lectures. The project involves analysing a given project spec, planning and managing the project execution, designing, implementing and testing system modules, integrating these systems modules with those programmed by others, and presenting the final product.

Through this project you will gain a better understanding of teamwork and the issues related to large scale projects (e.g. fixed deadlines, integration with others' work, team and project management, project planning, etc.). Therefore, you should work as a team and behave responsibly towards your fellow team members and the project supervisors (i.e., the teaching team members).

Relationship to formative assessment

Formative assessment is based on weekly laboratory meetings with the Lecturers and ATs (Associate Tutors).

Deliverables

1. Project requirements and design report: no more than 4,000 words (excluding figures, tables, and references) but no limit on pages.
2. Team demo, individual reflective report, and source code.

Project Setup

- **Team:**
  - A team is consisting of **5** members – teams of 5 are the common team sizing for Agile development projects.
  - You can decide your team members, but the 5 members must be in the same lab slot in the timetable, and not cross different labs.
  - You need to decide a team leader[1] among your team members.
  - The team leader should send the team member list (names and email addresses) to both jaejoon.lee@uea.ac.uk and Debbie.Taylor@uea.ac.uk by the end of Wednesday week 1. (Please also include the selected project title for your team.)
  - if you do not have a full 5 then still let us know who is assigned and we will then randomly allocate an extra person/people after Wednesday.
  - If you cannot find your team by the end of Wednesday, you will be randomly assigned, with no changes accepted after Wednesday week 1.

[1] The team leader will be the main contact point for the communication between the team and the teaching team (i.e., Lecturers and ATs)

- **Project Topic:**
  - Two candidate project scenarios are provided in Appendix A of this document.
  - You can decide a topic and the team leader should email your choice to both jaejoon.lee@uea.ac.uk and Debbie.Taylor@uea.ac.uk, when you send the team member list.

- **Programming language:**
  - JavaScript and Node.js (Please revisit CMP-4011A: Web-based Programming)

- **Project Schedule Overview:**
  - The main tasks for the first six weeks are about documenting:
    1. Requirements: Shall statements and Use cases
    2. Test cases: Test cases based on the requirements
    3. Software design: Architecture design and object-oriented design
    4. Project planning
    5. You may start implementing a prototype from week 4 once you have the requirements and design in place.

  - The remaining weeks are about system implementation based on the documents:
    1. We will follow an adapted Scrum agile approach
    2. You should implement the first sprint features identified in your document and do a prototype demo in week 8 for formative feedback
    3. You should implement the remaining features identified in your document in the second sprint by week 11.
    4. You should demo the whole system in week 12. All team members must attend the demo and explain their contributions to the project.
    5. Individual reflective report and source code should be submitted in week 12.

Marking scheme

1. Requirements and Design Document (45%)

(Please note a word template file will be provided for the report.)

1. Introduction and Project Ideas [10% marks]
   1.1 Main objective and overview
   1.2. Analysis of similar systems
   1.3. Feature matrix of similar systems

2. Requirements Analysis [40% marks]
   2.1 Use Case Diagram
   2.2 Use Case Table
   2.3 Shall-Statements
      2.3.1 First Sprint Features (Week 6 – Week 8)
      2.3.2 Test Cases for First Sprint
      2.3.3 Second Sprint Features (Week 9 – Week 11)

2.3.4 Test Cases for Second Sprint

3. Object-Oriented Design [40% marks]
   3.1 Class Diagram (Abstract)
   3.2 Message Sequence Diagram
   3.3 Detailed Class Description

4. Implementation Plans [10% marks]
   4.1 Task list and team member allocation
   4.2 Project plan for implementation, including milestones and deliverables
   4.3 Activity network with critical path, and Gantt chart

2. Team Demo and Reflective Report (45%)

2.1 Team Demo (90%)
(The guidelines for the demo will be provided after the Easter break.)
(The final demo mark for individuals can be moderated, if there is a discrepancy in the team member contributions with clear evidence.)

	Items	Assessment (%)	Comments
1	Implementation of identified features (20%)		
2	Quality of features (50%)		
3	Evidence of testing (20%)		
4	Quality of demo (10%)		

2.2 Individual Reflective Report (max 2 pages) (10%)
(Please note a word template file will be provided for the reflective report.)

	Items	Assessment (%)	Comments
1	Reflection on project experience (50%)		
2	Team member contribution identification (50%)		

3. Guidelines for marks:

70%+: Exemplary range and depth of attainment of the learning objectives related to the item. The writing is very clear and concise. (Excellent/Outstanding/Beyond Expectation)

60-70%: Conclusive attainment of nearly all learning objectives related to the item. The writing is well structured and expressed. (Good – Very Good)

50-60%: Clear attainment of most of the learning objectives related to the item, some more securely grasped than others. (Average – Good)

40-50%: Acceptable attainment of the learning objectives related to the item but shows clear weaknesses in describing the item. (Poor)

< 40%: Little, no or very poor attempt to describe the item. (Very Poor)

A note on use of external sources and plagiarism

Please note that while use of texts, or online sources, is encouraged in order to 'learn' design and programming principles, use of functions, lists, etc; they are not a substitute for completing the work yourself. It is not appropriate to find solutions or part solutions to assignments and submit them as your own work. Neither is it allowed to post questions on online forums requesting help or solutions to specific assignment tasks. To do either (copying/requesting) would be in breach of the university's regulations on plagiarism and collusion (General Regulation 18).

In the instances where you do use code (or any other work) copied from any source, you must acknowledge the source (e.g. including a comment with the URL and author alongside the copied sections). If in doubt, approach the coursework setter to discuss what is appropriate.

APPENDIX A

Candidate System 1: Study Planner

1. OVERVIEW

UEA is always looking for ways to support students in their studies and with many of the modules and assignments requiring detailed planning, prioritisation and tracking of tasks, the university wants to you to develop the "Study Planner" application. The goal of the Study Planner is to help students schedule tasks based on their module schedule and to define tasks and their dependencies for which they can then track their progress. Initially this application will be offered by the School of Computing Sciences but support for other schools in the future should be considered.

2. SOLUTION DESCRIPTION

The Study Planner application is a support tool that students can use to schedule and keep track of their study activities and to determine their progress towards completing their coursework assignments. For this, the student has to be able to record details of activities, which includes start date, end date, module and coursework assignment it relates to, task progress, and dependencies that can exist between tasks (such as one task can only be started after another has been completed). The application should be able to visualise the planned tasks and their dependencies by means of a Gantt chart representation. To monitor progress in a glance the application must support a Study Dashboard that highlights upcoming deadlines, progress towards milestones, etc.

The module, coursework and deadline information for modules taken by the students should be acquired by loading a file containing these details which should be provided by the hub on an individual student basis. It is up to design team to propose a file format for this and the details that the hub would be expected to provide. If this initial experiment for the Study Planner is successful it could be further integrated with the hub in the future by allowing the submission of coursework files and integrating news feeds with updates from module organisers.

Four central capabilities are required by the solution. The capability to:

1. Load module, coursework and deadline information from a defined file format
2. Ability to define study tasks, details, milestones and deadlines
3. Ability to record study activities that contribute towards completing study tasks and milestones
4. Visualise activities, dependencies, intermediate milestones and deadlines in a Gantt chart representation as well as a study progress dashboard that highlights upcoming deadlines, progress towards completing milestones and time spent for each module

The basic workflow of the Study Planner Application is outlined below:

Creating a Semester Study Profile

1. Initiating a new semester study profile
    a. The user creates a new study profile for the upcoming semester by clicking the appropriate button and provides an identifier for this semester (e.g. Spring Semester 2017-2018)
    b. Upon creation the system prompts the user for the semester file provided by the hub. If an invalid data file is provided an error is raised and the user is prompted to provide a valid data file before allowing the creation of the semester study profile to be completed
    c. For each module listed in the data file the application creates a content area that will be used to display activities, milestones, deadlines, etc.
2. Inspecting the semester study profile
    a. The user is presented with an overview area where he/she can select the modules in the current semester profile.
    b. Upon selecting one of the modules the user is taken to the detailed module page that lists activities, milestones, deadlines, Gantt charts, etc.

Plan Study Tasks and Milestones

1. After loading the data file the module profiles are populated with coursework assignments and exams, their weightings and their deadlines
2. The user can now start planning their semester by defining study tasks that contribute towards completing coursework assignments or preparing for an exam.
3. Study tasks must belong to a specific assessment event of the module (coursework, exam) and the application must at least capture the time that will be spent, the type of task (studying, programming, writing, etc.). The type of task is a defined set of types.
4. Each task must have a requirement criterion in it based on which progress and completion can be assessed, such as time studied, book chapters covered, assignment requirements completed, etc.
5. Tasks can have dependencies on each other, such as a task cannot be started before another has been completed. Other tasks can be done in parallel. The application must support this information in the tasks.
6. Each task must have the possibility to add notes
7. In addition, the user can define milestones (intermediate deliverables with their own deadline) that are part of completing coursework or preparing for an exam. Milestones must be related to the tasks that are required to be completed before the milestone is achieved
8. Note that deadlines can change as a result of the module organiser changing it or by applying for an extension. This must be supported by the application.

Capture Study Progress

1. Study progress is captured by defining study activities such as programming or writing
2. For each activity a quantity must be captured that can be used to evaluate its contribution, such as having studied for three hours, created three coursework parts, etc.
3. Each activity must be attached to at least one task, and the details of the activity contribute towards completing the task. For example if a task requires five hours of studying and a related activity lists that the user has only studied two hours, this means that in the task there are three hours of studying left
4. In addition for each activity the time spent must be captured
5. Each activity must have the possibility to add notes
6. An activity can be attached to multiple tasks and thereby contribute to the completion of both

Study Dashboard

1. To examine how the study for the semester is progressing the user can open the Study Dashboard
2. When the dashboard opens the application checks whether any deadlines for modules are approaching or have passed
3. Next the application determines whether the tasks associated with the deadline have been completed (i.e. the activities associated with the task are sufficient to satisfy the task requirement)
    a. If all the tasks for a specific deadline have been completed it will be added to the deadlines section of the dashboard
    b. If not all tasks for a deadline have been completed and the deadline has not passed, it is added to the upcoming deadlines section of the dashboard
    c. Otherwise the deadline is added to the missed deadline section of the dashboard
4. For each deadline a progress bar must be displayed that highlights how far the user is along to completing the work
5. The Study Dashboard must allow for users to examine the individual module planning and tracking they have done by opening a Gantt chart representation of the data they have entered. This should both visualise the tasks, milestones and deadlines, as well as the activities and how this is contributing towards completing the tasks.

Candidate System 2: Parking Management System (PMS)

1. OVERVIEW

UEA is evaluating the possibility of using a parking management system (PMS) to ease the parking problems on campus. A call for tender is being issued for the development of a proof-of-concept that will help UEA make their assessment.

2. SOLUTION DESCRIPTION

The PMS solution must support two types of users and the high-level features for them are as follows:

N.B., only one pre-registered administrator (admin) user will be in the system when the PMS is installed.
N.B., It is assumed that the parking spaces managed by PMS are all unoccupied when PMS is installed.

1) The admin type user can:

- 1.1) graphically monitor the overall status of parking space usage
- 1.2) monitor the status in numbers (e.g., total spaces, available spaces, occupied spaces, reserved spaces, etc.)
- 1.3) add/remove parking lots
- 1.4) block parking spaces (e.g., for road repair) and release them afterwards
- 1.5) reserve certain spaces for special events on campus
- 1.6) receive parking requests and approve/reject them
- 1.7) assign a specific location to a parking request or allow the PMS to assign the closest location available to the requested destination automatically, when approving requests
- 1.8) change the assigned locations, if needed (e.g., unexpected pre-occupation reports from drivers, etc.)
- 1.9) manage driver user accounts (i.e., remove/ban users)
- 1.10) send/receive text messages to/from drivers for communication

2) Driver type users can:

- 2.1) register to the PMS with a personal profile (e.g., username, password, contact information, etc.)
- 2.2) send a parking request with an indication of their destination on campus, for example, the department, college or building name, from a pull-down menu, along with the arrival/departure dates and times
- 2.3) make payments
- 2.4) receive their assigned parking space with a map from the main gate
- 2.5) notify their arrival at the allocated parking space with the current GPS coordinates
- 2.6) notify their departure, when leaving the campus
- 2.7) send/receive text messages to/from the admin user for communication

In addition to supporting the users, PMS can provide the following features:

3) PMS can:

- 3.1) manage driver user accounts (i.e., new registrations)
- 3.2) maintain the status of parking spaces
- 3.3) provide separate GUIs to the admin and driver users
- 3.4) find the nearest parking space to a requested destination
- 3.5) calculate the appropriate parking charges (e.g., hourly/daily charges with a limit of maximum stay of 10 days) and collect payments via credit/debit cards from drivers
- 3.6) handle the communication between the admin and drivers
- 3.7) alert the admin user when the GPS coordinates received from a driver and the allocated parking space do not match
- 3.8) alert the admin user when an arrival notification has not received within one hour from the booked arrival time
- 3.9) alert the admin user when a driver stays longer than one hour from the booked departure time