COMP5590 – Software Development Group Project – Assessment A2

You need to submit project deliverables after each Sprint. We will assess your project work after Sprint 3 is completed.

A2 Overview

At this stage, you will implement the functionality for your group's assigned interface. The requirements are now clarified by the customer. You need to implement the following functionality:

All Groups: Authentication (login/logout) and authorisation checks (logging access) functionality:

- (Authentication) The system should allow a user to log in with their username and password, and log out. When the user logs in, the system should show the user's messages on the welcome screen.
- (Authorisation checks) The system should log all access from a user, i.e. who accessed what functionality and when.

Groups A and D: Functionality for the "Patient" interface:

- The system should allow a new user to register as a patient, choose a doctor from the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to change their doctor using the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to arrange a booking with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the patient. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to view their bookings by entering a month and year.
- The system should allow a patient to reschedule a booking with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the patient. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to view the visit details regarding a past booking, for which the doctor provided visit details and prescriptions.
- (For five people groups) The system should allow a patient to view all doctors with their summary information, e.g. name, phone number.

- (For five people groups) The system should allow a patient to view a doctor's details, e.g. name, phone number, background, along with the doctor's availability for bookings in a given month and year.

Groups B and E: Functionality for the "Doctor" interface:

- The system should allow a doctor to view their bookings by entering a month and year.
- The system should allow a doctor to view their own patients with their summary information, e.g. name, phone number.
- The system should allow a doctor to enter visit details and prescriptions regarding a past booking. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a doctor to view the visit details and prescriptions regarding a past booking, for which the doctor provided visit details and prescriptions.
- The system should allow a doctor to edit the visit details and prescriptions regarding a past booking, for which the doctor provided visit details and prescriptions. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a doctor to assign a new doctor to a patient using the list of all doctors. The system should then send confirmation messages to the patient and both doctors.
- (For five people groups) The system should allow a doctor to view all patients (not only own patients) with their summary information, e.g. name, phone number.
- (For five people groups) The system should allow a doctor to send a message to an admin/receptionist using the list of all admin/receptionist personnel.

Groups C and F: Functionality for the "Admin/Receptionist" interface:

- The system should allow an admin/receptionist to enter a new doctor by providing their details, e.g. name, phone number, background. The system should then send a confirmation message to the doctor.
- The system should allow an admin/receptionist to enter a new patient by providing their details, e.g. name, phone number, and assign a doctor to the patient using the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to change a patient's doctor using the lists of all patients and doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to arrange a booking for a patient (selected from the list of all patients) with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the admin/receptionist. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to view bookings by selecting a doctor from the list of all doctors, by selecting a patient from the list of all patients, or by entering a month and year.

- The system should allow an admin/receptionist to remove or reschedule a booking. If the doctor is not available for the chosen date and time (in case of a rescheduling), the system should warn the admin/receptionist. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- (For five people groups) The system should allow an admin/receptionist to view all doctors with their summary information, e.g. name, phone number.
- (For five people groups) The system should allow an admin/receptionist to view all patients with their summary information, e.g. name, phone number.

A2 Details

GUI: You will implement a standalone Java application (e.g. with a JFrame UI as shown in the Eclipse demo or something similar). The *appearance* of the interface is not important as long as the required functionality is provided. That is, you are not expected to implement a *fancy looking* interface. Also see the Oracle tutorial about JFrame: https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html

Database: You need to have a working database that interacts with your program to store and retrieve relevant entities (e.g. doctors, patients, bookings, messages). You can use any database – a recording has been uploaded to Moodle about connecting to a MySQL database from an Eclipse project. You do not need to host the database remotely, each group member can install their local copy and work with test data. Your class supervisor does not need access to your database. Also see a simple MySQL tutorial: https://www.vogella.com/tutorials/MySQLJava/article.html

Git: One member of your group needs to set up the project's version control on the School's GitLab (https://git.cs.kent.ac.uk/) as shown in the Git demo uploaded to Moodle, and invite other group members as well as the class supervisor to the project. You can assign your class supervisor "Reporter" or "Developer" role, so they will have access to activity and code for the project. Note that every group member needs to commit and push their changes to the GitLab repository for the parts they implemented, so that we can monitor individual contributions. This is not limited to Java code, you can commit design or documentation related files.

We will monitor GitLab usage to identify inactive members. Each group member is expected to make reasonably frequent changes to Git. Having only a single commit/push to the repository at the end of a Sprint is a good indication of inactivity.

If the group does not have a GitLab project by the end of Sprint 3 or does not share the GitLab project with their class supervisor, all group members will lose 25% of the A2 mark.

A2 Deliverables

Deliverables for Sprint 1

- Implement authentication and one other functionality (all groups)
- Code: properly indented and commented, complying with naming conventions and quality standards, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and results (showing tests passed), including a brief two-page test specifications document explaining what tests are performed for each class/method

- Group organisation plan: brief one-page document describing who does what (e.g. leads for each feature, DB design, and tests) and plan for collaboration each week

Deliverables for Sprint 2

- Implement two functionality for three people groups / implement three functionality for four people groups / implement four functionality for five people groups
- Code: properly indented and commented, complying with naming conventions and quality standards, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and results (showing tests passed), including a brief two-page test specifications document explaining what tests are performed for each class/method

Deliverables for Sprint 3

- Implement authorisation checks (all groups); and one other functionality for three people groups / two other functionality for four people groups / three other functionality for five people groups
- Code: properly indented and commented, complying with naming conventions and quality standards, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and results (showing tests passed), including a brief two-page test specifications document explaining what tests are performed for each class/method
- Database design document: brief two-page document describing tables, columns, keys, and justification of design choices (you can include diagrams)
- Quality assurance: brief two-page document showing examples of code reviews, refactoring, issue tracking within the group
- Video demonstration: clearly showing how each feature works and how the database is manipulated as a result (e.g. when a new patient is registered, show how the database is updated), about one minute for each feature

How to submit: Create a .zip archive from all the deliverables and make your submission as a group on Moodle. Only one member from each group needs to make the submission. You need to make a Moodle submission for every Sprint.

How to Achieve Good Marks

Deliver a system that provides what is required for each feature. Implementing beyond what is required is not going to earn you additional marks. Only delivering a working system is not enough. You should spend effort on all the required deliverables: e.g. documentation, tests, code quality.

You can distribute tasks within your group as you like, e.g. one group member leads DB related code, another member leads UI related code. However, for some groups (if group members do not necessarily work comfortably with others), it is recommended that each group member takes the lead on certain project features and implements them fully (UI, DB, backend, tests) to minimise dependency among group members.

Although this is group work, we will assess each member of the group individually, via GitLab contributions. Make sure you commit/push your own work using your Kent email ID. As long as you are doing the work you are supposed to (with evidence on GitLab), you do not need to worry about others not doing their part.

This is the distribution of marks (out of 30):

- 16 marks for implementing required functionality (assessed mainly via the video demonstration and supported by code on GitLab)
- 5 marks for code quality, including the quality assurance document
- 5 marks for testing, including the test specifications document
- 1 mark for group organisation plan
- 3 marks for database design document

Late Submission and Plagiarism

Late or non submission of coursework

The penalty for late or non submission of coursework is normally that a mark of zero is awarded for the missing piece of work and the final mark for the module is calculated accordingly. Your group as a whole is responsible for monitoring its use of materials. This means that there is a risk that if plagiarism is found, the whole group will be held responsible.

Plagiarism and Duplication of Material

Senate has agreed the following definition of plagiarism: "Plagiarism is the act of repeating the ideas or discoveries of another as one's own. To copy sentences, phrases or even striking expressions without acknowledgment in a manner that may deceive the reader as to the source is plagiarism; to paraphrase in a manner that may deceive the reader is likewise plagiarism. Where such copying or close paraphrase has occurred the mere mention of the source in a bibliography will not be deemed sufficient acknowledgment; in each such instance it must be referred specifically to its source. Verbatim quotations must be directly acknowledged either in inverted commas or by indenting." The work you submit must be your own, except where its original author is clearly referenced. We reserve the right to run checks on all submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism. When you use other peoples' material, you must clearly indicate the source of the material using the Harvard style (see https://www.kent.ac.uk/ai/styleguides.html).

In addition, substantial amounts of verbatim or near verbatim cut-and-paste from web-based sources, course material and other resources will not be considered as evidence of your own understanding of the topics being examined.

The School publishes an on-line Plagiarism and Collaboration Frequently Asked Questions (FAQ) which is available at: https://www.cs.kent.ac.uk/students/plagiarism-faq.html.

Work may be submitted to Turnitin for the identification of possible plagiarism. You can find out more about Turnitin at the following page: https://www.kent.ac.uk/ai/using-turnitin.html.