CO559 – Software Development
Group Project – Assessment A2

- Set: Monday W17
- Sprint 1 Deadline: Friday W18
- Sprint 2 Deadline: Friday W20
- Sprint 3 Deadline: Friday W22

A2 Overview

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

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

Groups A, D, and G: Functionality for "Patient":

- 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 "Doctor":

- 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 "Admin/Receptionist":

- 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 allow an admin/receptionist to enter a new patient 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 user interface as shown in the LecSem 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. See the Oracle tutorial on JFrame for more information: 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 – see a simple MySQL tutorial for more information: https://www.vogella.com/tutorials/MySQLJava/article.html

Git: One group member needs to set up the project's revision control on the School's GitLab (https://git.cs.kent.ac.uk/) as shown in the LecSem demo, and invite other group members and the class supervisor to 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.

Scrum stand-up meetings: At this point, inactive members for A1 in each group have been identified and your group has been notified if you have inactive members. Additional measures will be in place for A2 to further monitor inactive members. Each team will set up two scrum stand-up meetings every week (plus one additional scrum stand-up meeting during the classes designated for "Group Work").

- If you miss four or more scrum stand-up meetings in total, you will lose 25% of your A2 mark, and part of this lost mark will be distributed among the remaining team members.
- If you miss six or more scrum stand-up meetings in total, you will lose 50% of your A2 mark, and part of this lost mark will be distributed among the remaining team members.
- If you miss eight or more scrum stand-up meetings in total, you will lose 100% of your A2 mark, and part of this lost mark will be distributed among the remaining team members.

You need to document who was present in each meeting and who reported what during each meeting. We will also monitor GitLab usage to identify inactive members.

A2 Deliverables

You need to deliver the following user stories for each Sprint:

- Sprint 1: implement authentication and one user story (all groups)
- Sprint 2: implement two user stories for three people groups / implement three user stories for four people groups / implement four user stories for five people groups

- Sprint 3: implement authorisation (all groups) and one user story for three people groups / two user stories for four people groups / three user stories for five people groups

Deliverables for each Sprint

- Code: properly indented and commented, complying with naming conventions, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and result logs
- Reports of Scrum stand-up meetings: attendance and who reported what for each meeting
- Database design document (evolving): tables, columns, keys, and justification of design choices

Additional deliverables for Sprint 1

- 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
- Effort estimation and prioritisation for all user stories to be implemented
- Plan for version control: brief one-page document describing how Git is utilised, e.g. how often changes are pushed to GitLab, use of branching

Additional deliverables for Sprint 3

- Quality assurance: evidence of code reviews, refactoring, issue tracking
- Video demonstration: showing how each feature works, about one minute for each feature
- User manual: short document (about three pages) explaining how each feature can be used by the assigned role for your group

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

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. evidence of continuous team collaboration, tests, code reviews and quality.

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.