YUCONZ

Meeting Minutes

Meeting purpose:	Workshop
Date:	20190305
Time:	1200-1300
Location:	CSPC1
Attendees:	Marin Donchev [MD485], James Dyer [JD556], Nicholas Salter [NDS8]
Minute taker:	JD556

Next Steps:	Checkpoint date:	Assigned to:	Due Date:

Points to discuss next meeting:

Decisions made: (what, why, impacts)
1. Ranked coding tasks to be completed by importance (See Discussion point 3b). 2. Decided to ask Miles Roman what roles should have the ability to add, amend and remove users from the HR system.

Miscellaneous details:
JDG23 absent due to sickness. David Barnes was present at workshop to answer questions from 1220-1300.

Discussion:

1. Reviewed requirements for Stage 5.
    a. Noted that multiple additional documents are required for the Stage 5 project Corpus.
        i. Index.html
            1. An HTML file linking to all PDF deliverables (using relative paths)
        ii. UM.pdf
            1. A User Manual for the system for users of the system.
        iii. Status.pdf
            1. A status report that details what has been completed of the user's requirements and what has been left incomplete.
            2. This needs to be completed regardless of the level of completion of each requirement.
        iv. Log.pdf
            1. A log file of authentication and authorisation checks for one or more typical sessions with the system.
        v. A video demonstration of the system

            1. Maximum 5 minutes, to be uploaded on Youtube as a private/unlisted video.
        b. Noted that a sequence diagram does not need to be provided, as of revision of brief by client on 20190304.
2. NDS8 began work on index.html during the meeting.
    a. JD556 reminded NDS8 that no external links (including to images) are allowed in the index, as it must be self-contained.
3. Discussed current state of the code for the Yuconz project.
    a. MD485 and JD556 noted that ensuring persistence of state between executions of the program via implementing a database has higher priority than implementing a mockup for requirements in this stage.
    b. Ranked tasks to complete for code by importance:
        i. Implementing database using SQLite.
        ii. Implement annual review functionality.
        iii. (In no particular order):
            1. Rewrite AppController to use a UserInterface Class.
            2. Hash passwords.
            3. Remove typecast from AppController.
4. Decided to ask Miles Roman what roles should have the ability to add, amend and remove users from the HR system.
    a. Such actions may include “add user”, “change password”, “change roles” and “revoke login rights”.
    b. MD485 noted that this should be of low priority, as it was not directly required by the client.
5. Asked David Barnes several questions and received response:
    a. JD556 asked whether the log.pdf should be annotated, as otherwise it would be difficult to read and would include >20 log records for the unit tests alone.
        i. Response: Records in log.pdf should be selected such that it can be reasonably understood that all actions are being logged.
    b. JD556 noted that log.pdf would include both Authentication and Authorisation logs, but each is currently stored in CSV format with differing columns.
        i. Response: That is correct, and each should be labelled with its source, and potentially with annotations on what action the records correspond to.
    c. MD485 asked how the User Manual should be organised.
        i. Response: There are two ways of doing it. You can organise by the user’s role – “If you’re an HR Employee, read this section. If you’re a Director, read this section”, or you can include a general section “You will be prompted to enter your username and password, then select a role. You will then see the following actions based on the role you have selected”, and then include a manual for specific actions such as “Read a personal details record”.
    d. Asked for advice on writing the AppController class, as it currently handles multiple actions, including the login screen and action selection when logged in.
        i. Response: Menus can be separated into their own class, with AppController calling each when needed.
6. Discussed how to implement Menus as their own class.
    a. MD485 proposed creating a Menu class that holds an ArrayList<String> of Options, which then calls the operation selected by the user from AppController.

b. JD556 proposed that the Menu could pass an Enum representing which method to call back to the AppController, or a lambda/other way of passing a method directly could be used, such as class::methodName.
c. Concluded that more research into Java lambdas is needed.
d. MD485 suggested checking World of Zuul's menu system.
    i. Checked the parser and relevant classes, noted that user inputs are parsed into the first "command" word which specifies which action to take, and additional "modifier" words, which specify which object or direction that command is related to.
    ii. Noted that command words are differentiated by the controller using String.equals, and each command was hard-coded in.