Quality Assurance plan

Priyesh Patel – prp6, Matthew Mukalere – mm871, Euel Domingo – ejd29, Sahib Jabbal – ssj9

Coding and Documentation Guidelines

Coding

- File names will clearly show what the file contains and will be written in camel case with the addition of the first letter being capitalised, eg an authentication controller java file should be named as: AuthenticationController.java.
- Method heads will also be clearly named to best reflect the purpose of the method and will follow the standard camel case convention of thisIsAnExample. Parameters must have meaningful names to describe their contents, e.g. a username parameter should not be named x it should be named something meaningful such as username.
- Methods will be fully commented include @param and @return descriptions
- Each Java file should contain, in the following order, any import statements (avoiding wildcard imports when possible), class description and group name, and exactly one top-level class.
- Sections should be separated by a single blank line.
- Braces are to be to used even when optional e.g. single line if statements
- Code must be properly indented, check with auto indent function of IntelliJ and self-judgement before saving the code.

Documents

- File name conventions follow a similar style to code files however spaces are permitted between words
- Document font will be Corbel and of size 12 for the body, 14 for the Main heading, subheadings will be in bold
- The footers shall contain the name of each member as well as page numbering on the right

Review Planning

Reviewing Documents

- We will use GitLab to share the documents within the group for each member to review the document and provide their feedback.
- Each member will have documents to make. When a member is finished, he will commit the document on Git repository and inform everyone else in the team for feedback. We will use the document against the planned progress and the requirements. The feedback and changes will be documented on Git. When a document needs amending, the member will edit the document and

commit to the Git repository stating the changes made. In this way, we can keep track of the reviews and changes made on the documents. This will also show that a review has taken place.

- We will use the comments feature available on MS Word to add comments on the documents to provide our feedback
- At every meeting, we will review each other's work that was assigned to them at the previous meeting at the beginning of the meeting.

Reviewing Software

- We will have regular meetings and review the software progress within the group
- We will use GitLab to store the different versions of the software while developing which allows us to review the software development while programming
- We will communicate with each other through WhatsApp while programming for clarifications
- We will use comments while programming the software for other members to understand the code while programming

Reviewing actual progress against planned progress

- We will keep a record of the planned progress of the project and review the planned progress regularly to ensure that the actual progress is on track

How will we adjust plans and processes in the light of review outcomes?

- We will modify the initial Gannt Chart to accommodate any delays that may occur
- We will make necessary changes in the shortest time possible to meet the required deadlines

Software Testing & Bug Reporting

We will be using a white box testing technique in order to test the system. After coding the system, the system will be tested with this technique and then update the version where necessary. The test will involve the programmer and the program opened using IntelliJ. During testing the programmer will test its functionality, such as Login in. The process will involve the programmer entering the details such as the username, password and access level of the user logging in.

The table below will be used as guidance for testing the main functionality of the application and will be used for acceptance testing. The table serves as a template for now.

White Box testing, what will involve in the process

White box testing involves the programmer, the program opened using Intellij and the list of main functionalities. The programmer will test the implemented program based on the given functionality. He will record the test no, the functionality, the steps to produce it which will be the input of the tester, the output and whether the test has failed or not. In this case, the programmer will be able to know what functionalities that are not working and then change or update the code in order for it to work. In summary, white box testing involves the tester fixing any issues with the program as soon as the functionality being tested fails.

Test no.	Test	Input	Expected Output	Actual Output	Test Successful (Yes/No)
	Open application		System starts up		
	Close/quit application		System terminates		
	Log In		System provides access rights		
	Log Out		Server revokes access rights		
	Authorisation Check		Grant requested access		
	Read Personal details		Delivers record to user		
	Create personal details record		New record is saved		
	Amend personal details record		Amended record saved		
	Create new review record		New review record is saved		
	Read review record		Delivers review record to user		
	Amend review record		Amended review record saved		
	Read past completed review records		Delivers past review records		
	Perform review		Review completed		
	Allocate Reviewer		Informs Employee and Manager/Director reviewers of the review details		

Bug reporting

We will be using the table below as the template for reporting any bugs. This will keep track of any defects the system will generate. Any reports of bug will be documented and be given accurate and updated status.

Bug reports will be stored in the git repository. Bugs will be dealt with according to their priority, severity and status. For example a bug with status of new, high priority and minor severity will involve any available programmer to fix the issue. Another example is that if a bug is on Critical Severity, it will involve all the programmers in order to fix the issue. Any bugs that are on a 'Fixed' status are stored and will be used for future reference if a similar bug occurs. The team is responsible for checking and fixing bugs that are reported.

Integration of Bug Reporting

A bug will be filled up by team member who will received a call from a user reporting an issue within the system. The team member will fill up the form as accurate as possible and allocate the bug fix to the available team member who can fix the bug. The developer that will be fixing the bug will be informed. To make sure that the team member takes responsibility on the bug, he will have a deadline for fixing the bug. The deadline will be set in accordance with the severity and priority of the bug. After the bug has been fixed, he will update the bug report and commit changes to Git with the message about the fixed bug.

Bug ID:	*<This gives a bug a unique identifier number so it can be used for future references>*
Bug Name:	*<Gives the name of the bug >*
Date:	*<This tells when the bug happened/reported>*
Name of Reporter:	*<This tells who was using the system and reported the bug>*
Software Version:	*<This tells the version of the software that was being used when the bug was reported>*
Summary:	*<This explains what was the reporter doing and what happened.>*
Operating System:	*<This tells what operating system the reporter was using when the bug happened>*
Steps to reproduce:	*<This lists the steps of the issue so the developer can recreate it>*

Priority:	*<(High, Medium, Low) It is based on when the bug will be fixed>*
Severity:	*<(Minor, Major, Critical) This tells the impact has the bug on the software and the users>*
Status:	*<(New, Fixed, Could not Reproduce, Closed, Invalid, Need more information and Assigned) This is used to track the status of he bug>*
Notes:	*<Extra information about the bug>*

Follow up on test failures and retesting

The team is responsible for testing the system and database regularly in order to find and fix any issues as soon as they come up. Any failures will be documented as a bug in order for the team to keep track of the failure. Whenever a bug is fixed, bug report is updated and retesting is initiated.