PROJECT MANUAL

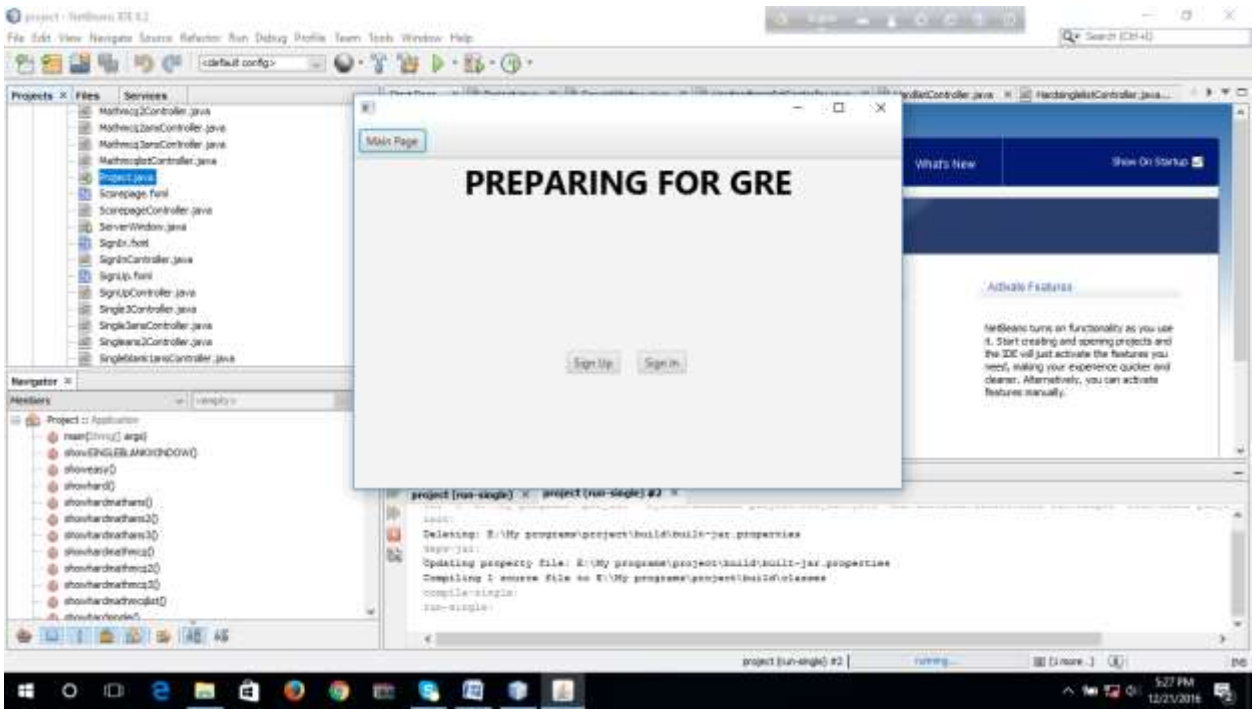
“Preparing for GRE”

This term project was aimed to design an application named “Preparing for GRE” where users will be able to give exams and get scores. They would get to know the answers of the questions after they submit their script.

Firstly, this had a server-client system, where the server held all the files having the question bank as well as the users’ names and their corresponding passwords.

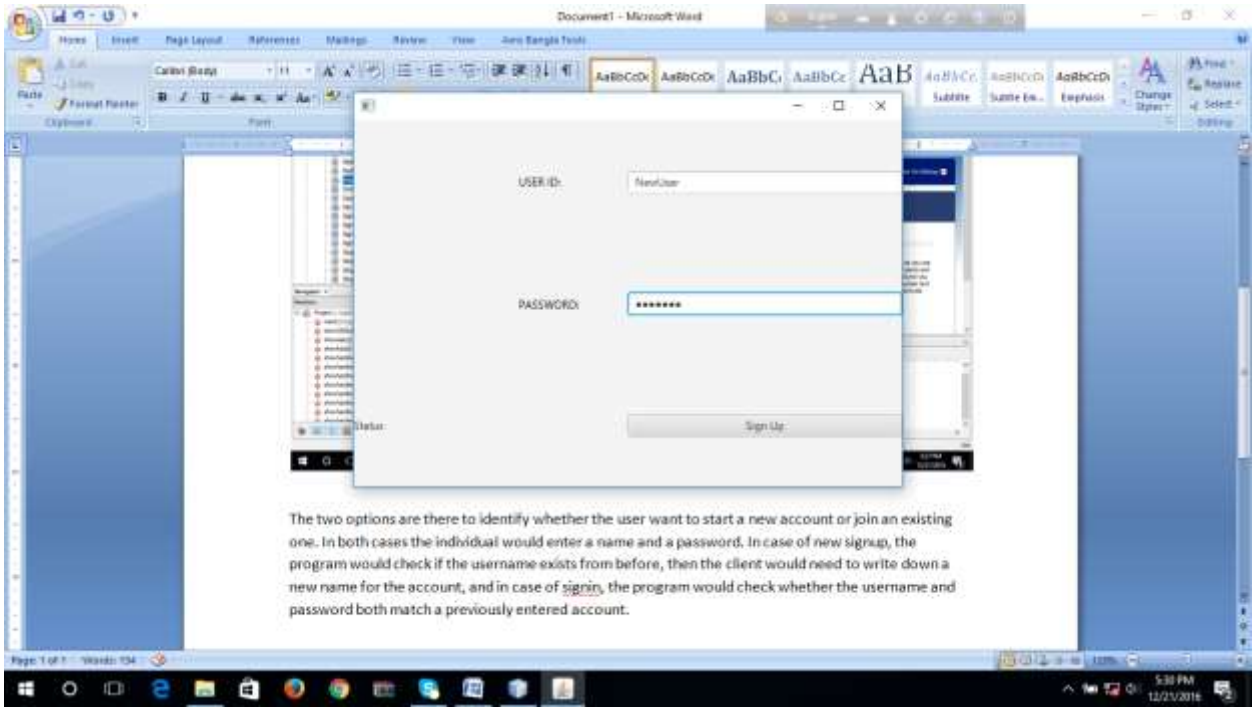
Scenebuilder has been used to prepare the application and therefore fxml files were involved.

At first, after building the project, the ServerWindow class is made to run, which establishes a serversocket. This is the server of the program.
Next, the Project class is run which is the client of the entire program. Running ‘Project’ class causes the program to begin with the following scene:

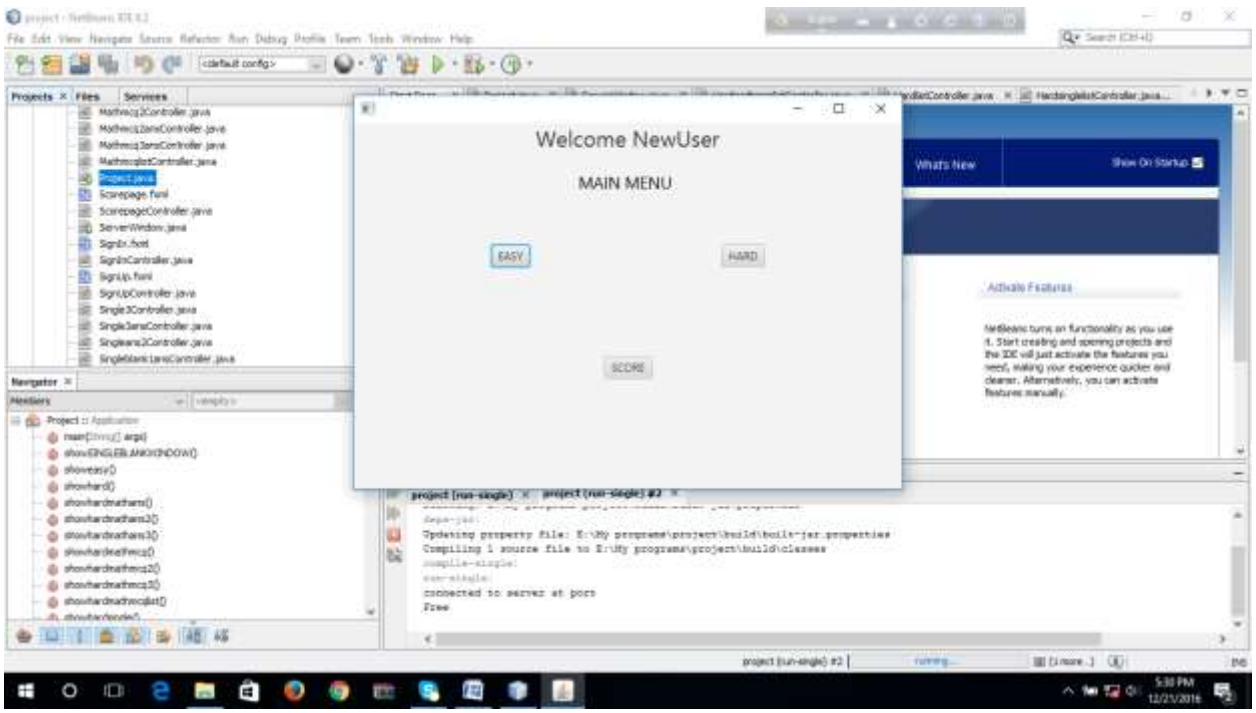


The two options are there to identify whether the users want to start a new account or join an existing one. In both cases the individual would enter a name and a password. In case of new signup, the program would check if the username exists from before, then the client would need to write down a new name for the account, and in case of signin, the program would check whether the username and

password both match a previously entered account. Signup is controlled by SignUpController and Sign in through SignInController classes respectively.

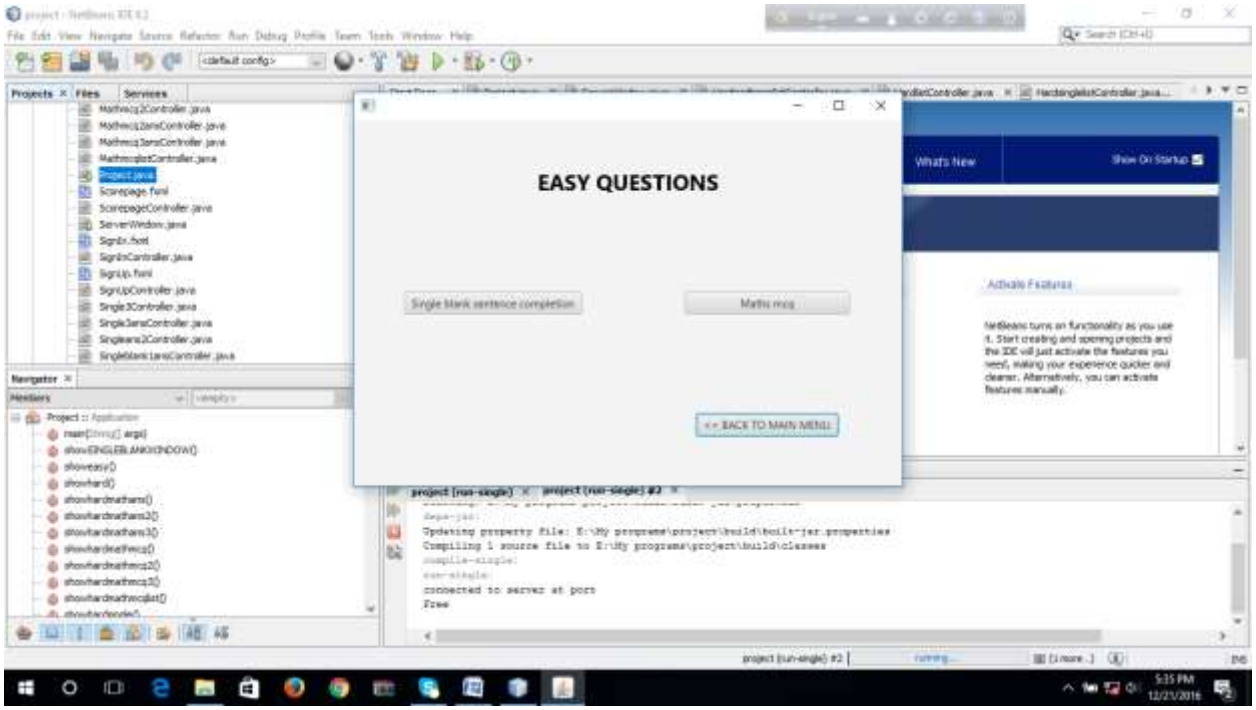


After signing up or signing in, the following scene is found which is controlled by the HomeController class:

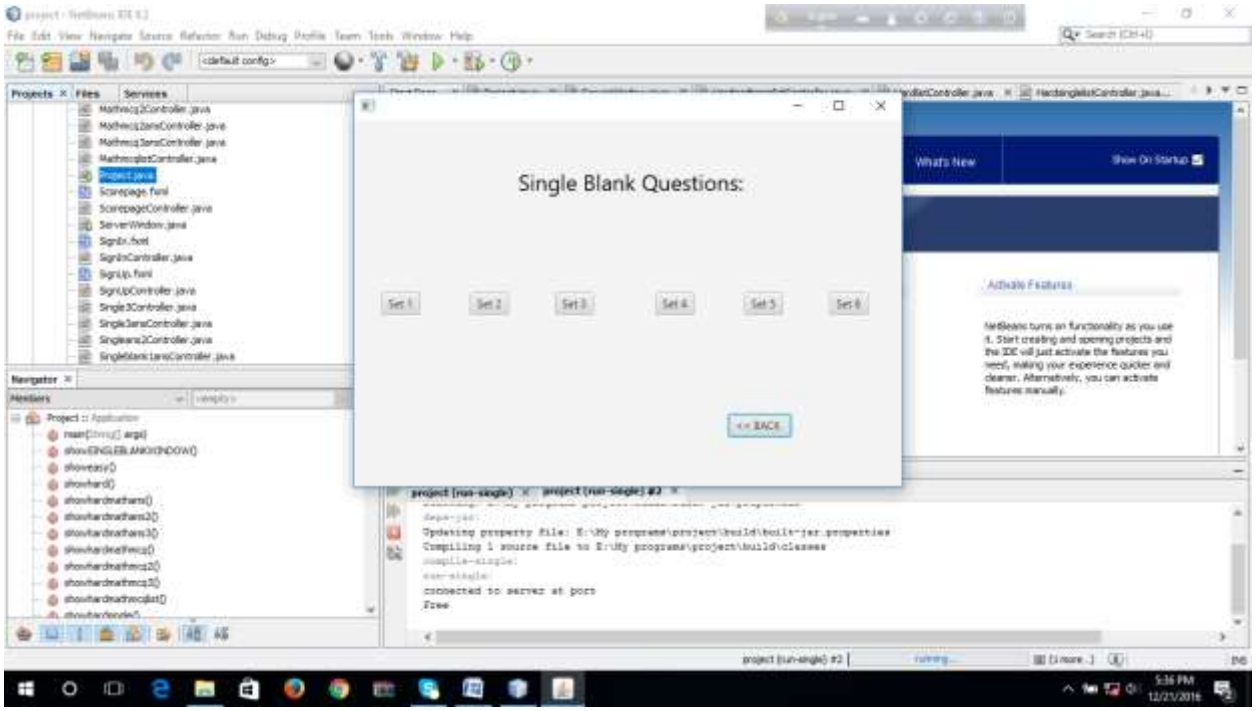


This is the home window or the Main Menu, it contains three buttons that lead to three different scenes. Easy contains easy sets of questions, whereas hard contains hard sets of questions. Score contains the total score of the user combining all the exams taken in total.

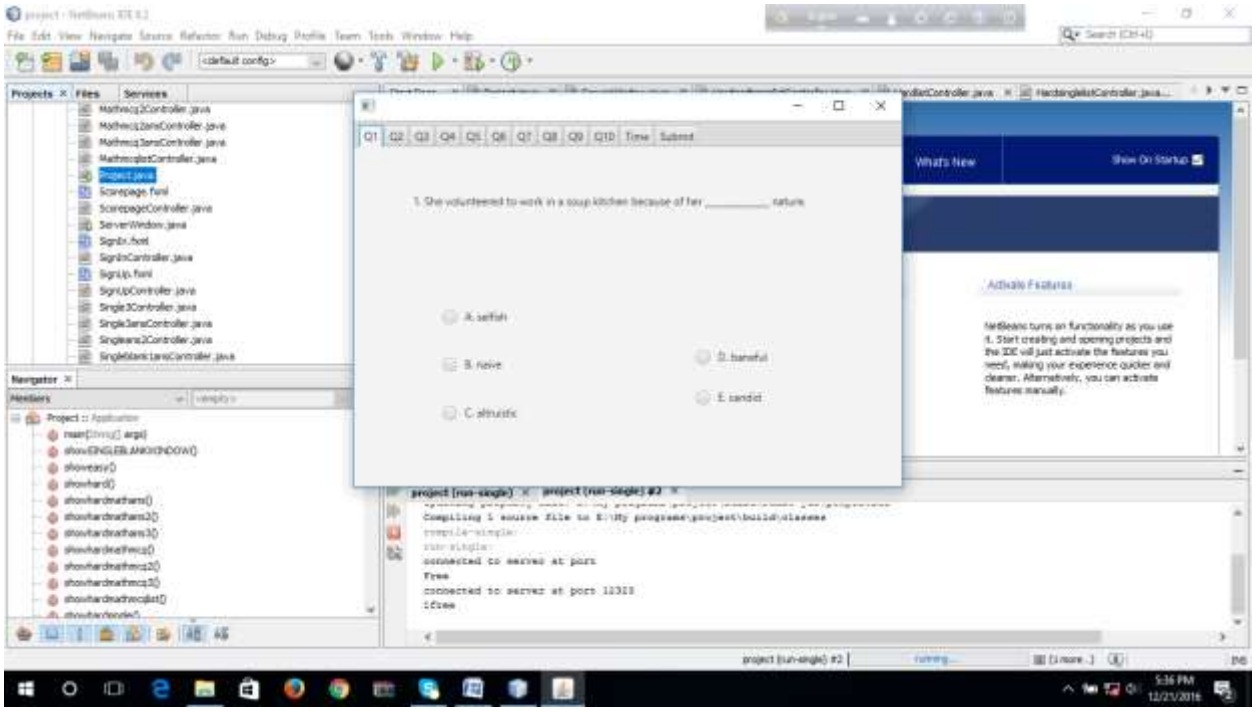
Both easy and hard question sets contain two subjects: English and Maths controlled by EasySetController class:



There are question sets in all the subjects, both easy and hard. The hard sets of questions are presented in a similar window which is again controlled by the HardlistController class. The following window appears if one chooses to answer the English questions or Single blank Sentence Completion questions which is again controlled by SingleblanklistController class and in case of hard ones, it is controlled by HardsinglelistController class.



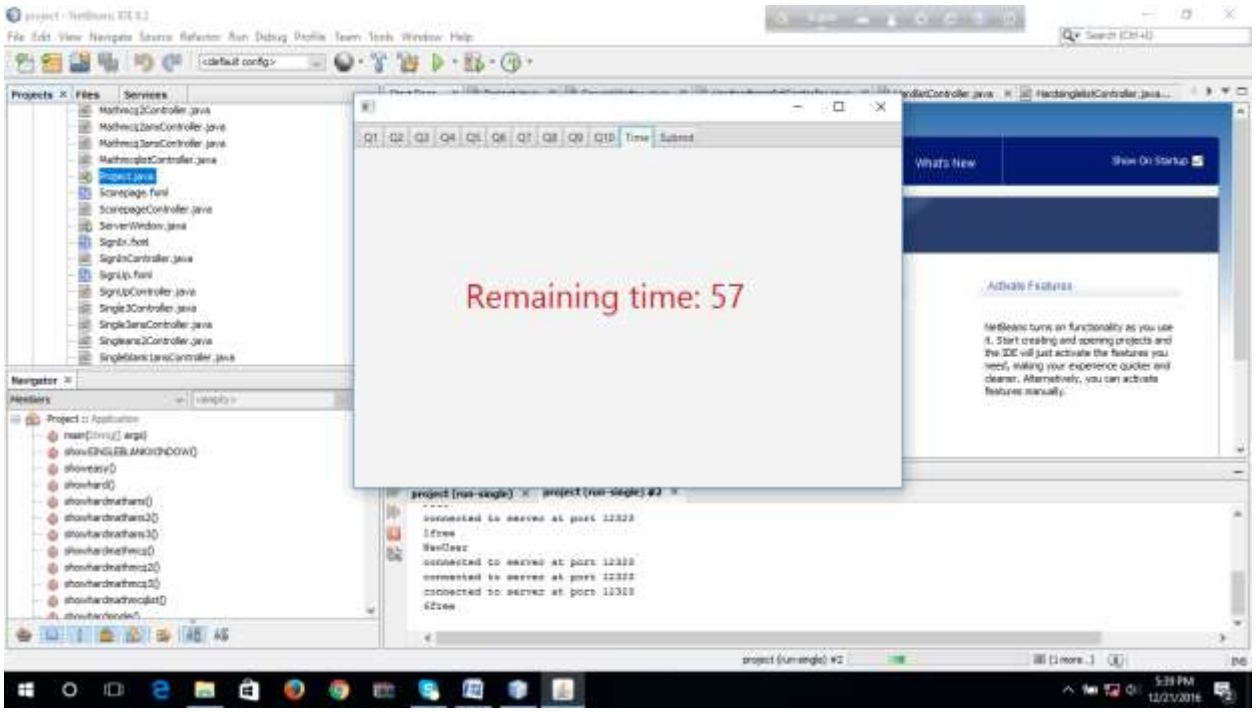
The questions appear in the following way, in TabPane, each question in one Tab, by pressing on any particular set:



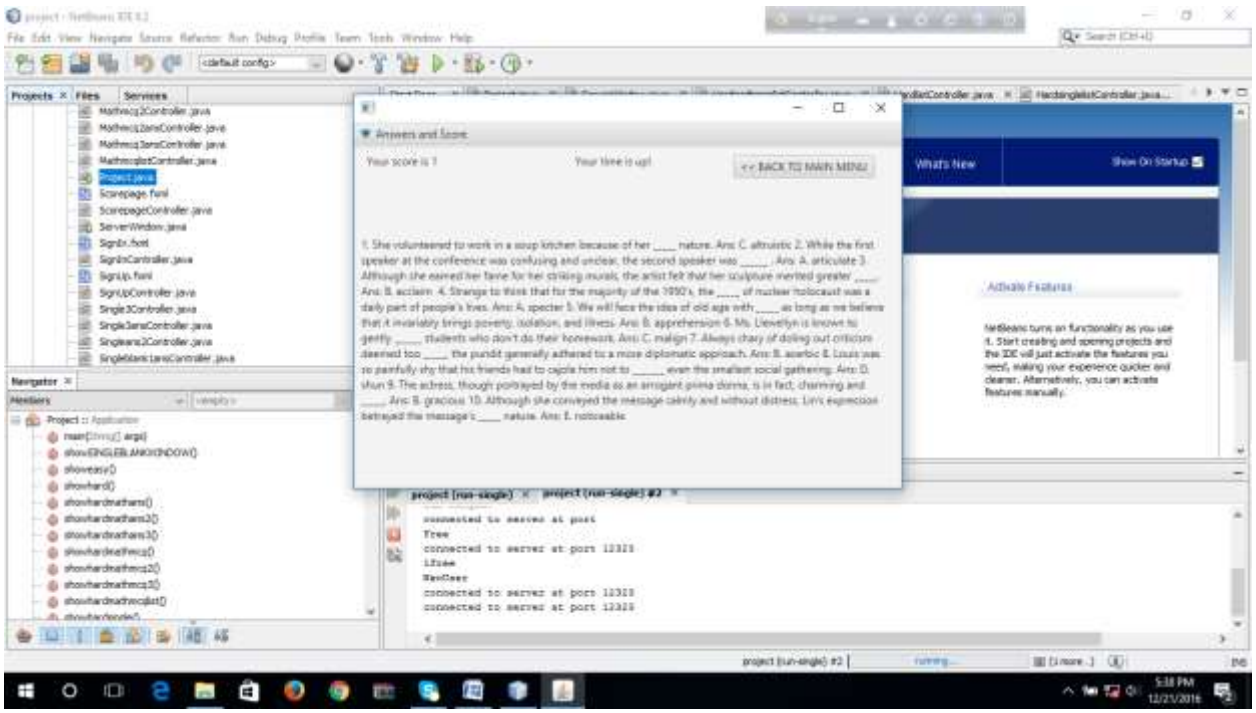
The same goes for every other set or every other subject as well as hard sets of questions. They are all individually controlled by separate classes, for example set 1 in single blank sentence completion is controlled by SingleblankwindowController class, set 1 in easy maths is controlled by

Mathmcq1Controller class etc. In all of these scenes, the functions are similar but the questions received are different:

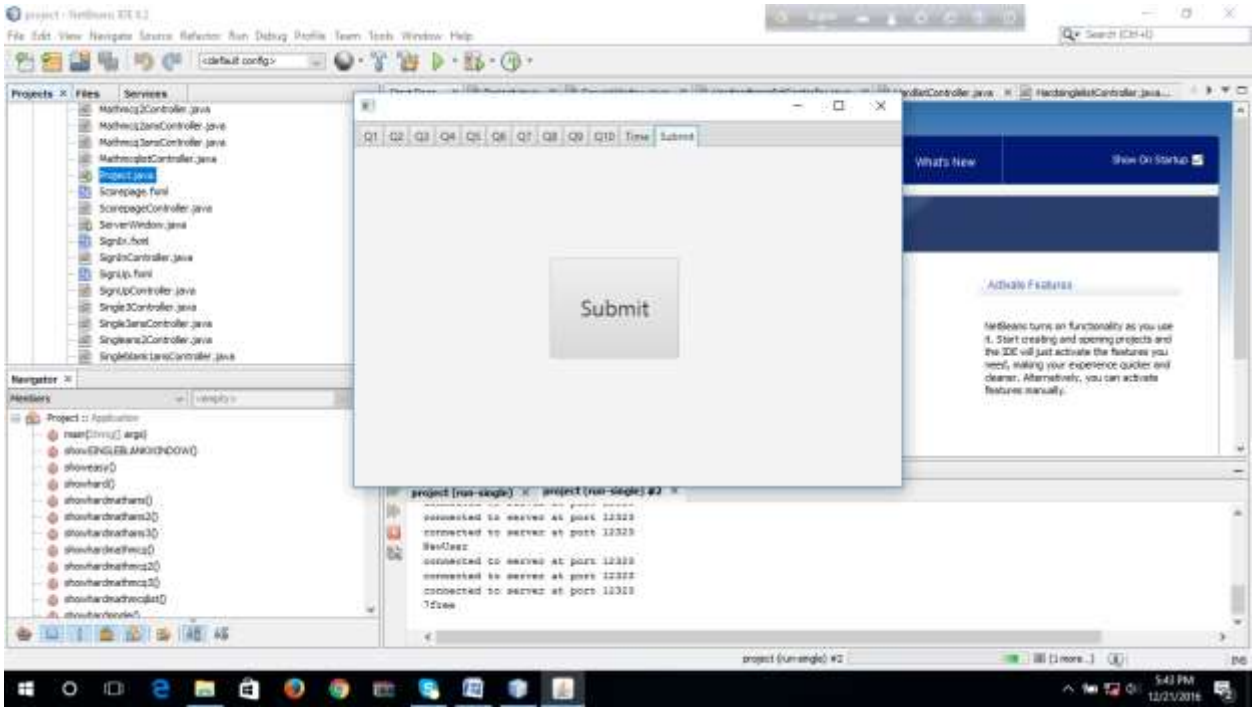
Here the time tab shows the remaining time in seconds:



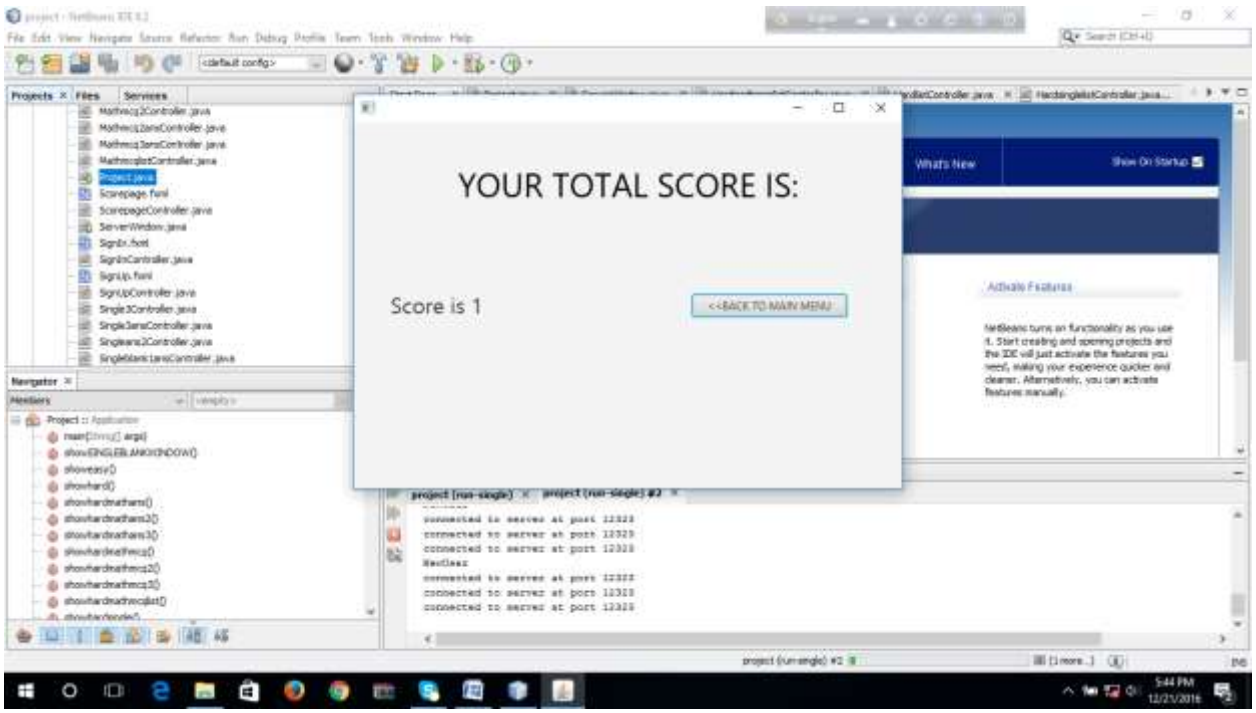
If time runs out, automatically the following window comes up, showing all the answers as well:



Otherwise the user has the option to click on submit. In both cases, the window that appears stays the same but the label shows different texts.



Finally, the user can check the total score by going to the score option from the main menu, controlled by ScorepageController class:



The user can go back to the main menu and resume the tests that are given. But if the user has already given a test, the new score is not counted, only the initial score is saved.

Besides these, the appropriate .txt files (in TextContent sub folder in ProjectRoot folder) contain the question bank as well as username+password along with scores and also the questions that have been taken (in codeFORuser). The .fxml files are included as well in TextContent according to the submission instructions supplied.

DIFFERENT JAVA CLASSES USED:

Various classes were used to perform different functions. For example: For networking in tcp system, ServerSocket, Socket, DataInputStream, DataOutputStream etc. were used. To form the timer, Timeline class was used. Stage, Scene, Pane etc. were used to form the GUI code with the help of .fxml file. FileInputStream, FileWriter etc. were used to deal with the text documents.

JAVA COMMANDS TO RUN SERVER/CLIENT:

In order to run the program from cmd, the following commands are required:

To run the server and client:

After the particular directory has been selected where the classes are stored, since ServerWindow.java is the server class so it is compiled with the following command:

For example, in my computer it lies in Local Disc C having package project:

C:\Users\username>cd C:\project

C:\project\>javac ServerWindow.java Project.java

C:\>java project.ServerWindow

C:\>java project.Project