Jamie Brindle

ID:06352322
Msc Advanced Computing
Unit Title: Mobile Computing
Unit Code: 63IC7302
Tutor: Dr. N. Whittaker

Assignment1 – Mobile Quiz

Contents

Running the Applications

Using Eclipse:

The entire system has been created from within eclipse thus following eclipses folder structure for both the servlets and the MIDlets and includes the Apache Tomcat6 server, which can all be imported and run from eclipse, however this would require addition eclipse setup including installing extra plugins to support JavaME and to possibly setup the server, depending on which version of eclipse is running. I have however exported both the servlets and the MIDlets to allow them to be used without eclipse

The MIDlet

The MIDlet and associated files are found within the MIDlet directory on the CD provided. To deploy, copy the folder 'mob-assign1-MIDlet' to the 'apps' folder in the wireless toolkit folder and run it through the wireless toolkit front end (ktoolbar). If using different means to run the MIDlet, there's a jar file of the MIDlet suite found in /eclipseProject/mob-assign1-MIDlet/deployed. The server must be up and running before running the MIDlet in order for the MIDlet to connect to the servlets. The MIDlet is currently set to connect to the URL:

http://localhost:8180/mob-assign1-web/servlet/MIDletServlet

However this can easily be changed by altering the text file 'ServerOptions.txt' found where the class files are located

The Server / Servlets

The servlets are found with the 'web' folder on the CD provided. Copy the folder 'mob-assign1-web' to the 'webapps' folder in the Tomcat folder, then start the tomcat server. You should be able to view the servlets index.html page by typing http://localhost:8180/mob-assign1-web in the browser. If tomcat is set up differently, the MIDlet's ServerOptions.txt file may need to be changed.

The MYSQL database

I have used MMU's departmental mysql server and set up the servlets to connect to it. I have also included a MYSQL dump on the CD provided to allow transfer to another mysql server. Mysql location and authentication details where the servlets connect to can be changed by altering the ServerOptions.txt file which is found where the servlets class files are located

Introduction

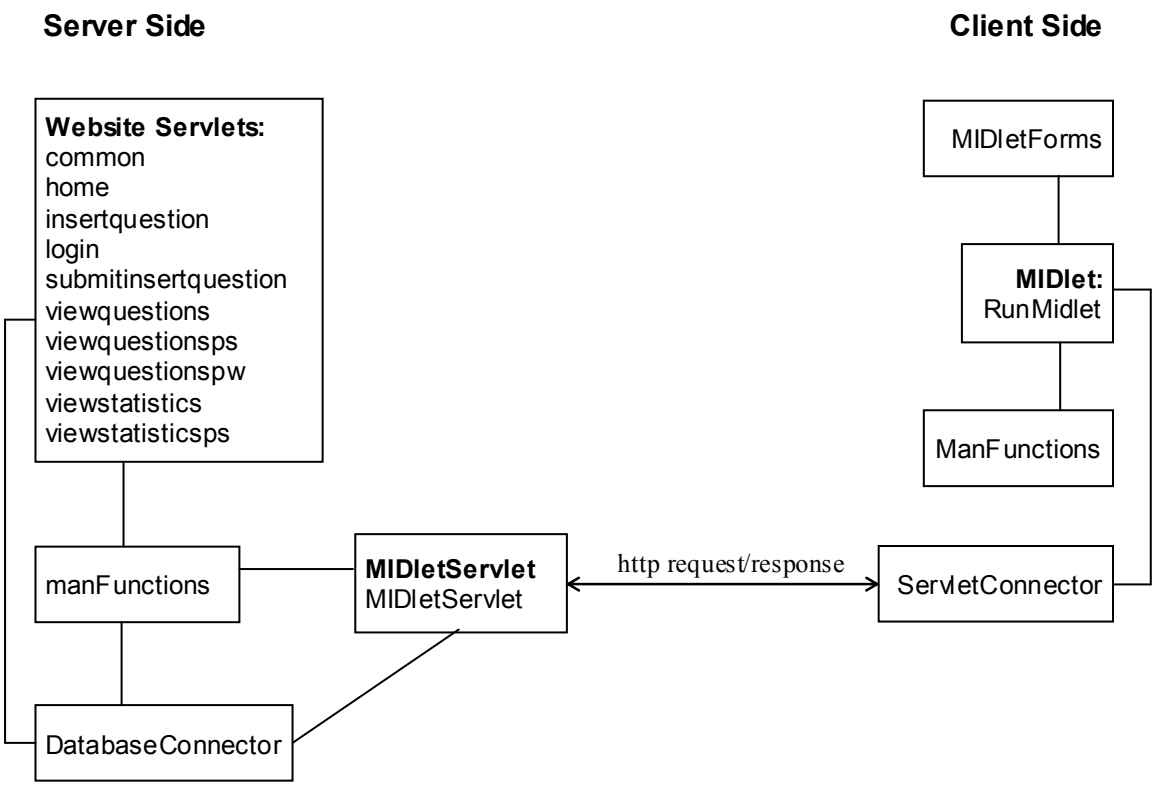
I am to create an electronic voting system whereby say, a lecturer can store a list of questions on a database and his/her students can answer the questions through a mobile device, such as a mobile phone by connecting to a http server using their phone's internet connection, which in turn connects to a database and stores the students answer.

Design and Chosen Route

The system will be created using the Java programming language using the Java 1.6 SDK to program a set of servlets which will run on the an Apache Tomcat server, which MIDlets, programmed using the Java Microedition library will connect to in order to retrieve questions and upload answers to a mysql database. Servlets will also be used to create a dynamic web site which will act as an administrative panel for tutors to view results of a quiz and to upload new questions.

Implementation

Class Structure



The programming and class structure is designed to allow code re-usability, maintainability, abstraction, code simplification and code readability. The 'ManFunction' class stands for manipulation functions, it's merely a class containing a number of methods which manipulate Strings and also perform some checking functions. I've designed the communication between MIDlet and servlet to be much like that of Sockets whereby I've used Strings to be passed to and from MIDlet and Servlet, in which the String will contain a 'keyword' which will act as a command followed by other pieces of information, separated with a colon (:) which act as parameters, hence the need for the String manipulation and checking methods in the ManFunctions class.

The MIDletForms class is tightly connected to the RunMIDlet class, its sole purpose is to create and layout a number of forms for the RunMIDlet class and exists to provide code simplification. The RunMIDlet class will communicate with the MIDlet servlet on the server through the ServletConnector class, in which it's purposes is to simply make the connection to the servlet and relay request / response Strings. This provides abstraction as the RunMIDlet need not have to worry about making the connection, only that it gets data returned to it, which also provides for better maintainability, as we can change the way the MIDlet makes a connection without affecting the rest of the code. The DatabaseConnector class on the server side works in exactly the same way for both website servlets and the MIDlet servlet. All the MIDlet servlet does is work out what command it has been giving, converts the String it has been given into actual parameters and calls the corresponding method in the DatabaseConnector.

JavaDoc has been written and JavaDoc pages have been created for each class and its methods stating what each of them does along with its parameters. These pages can be found in each project in the 'eclipseProject' folder on the CD provided.

Source Code:

Client Side Source Code:

ManFunctions.java

```
import java.io.InputStream;
import java.util.Vector;

/**
 * ManFunctions (Manual/Manipulation Function) class contains re-usable methods which generally
 * manipulate Strings and also check their content.
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class ManFunctions {

	/**
	 * The Manfunctions constructor - takes no parameters
	 */
	protected ManFunctions() {}

	/**
	 * Compares two strings to see if one stream is the same or a substring of another
	 * Created as the J2ME library doesn't contain a method for doing this
	 * @param full is the String we're comparing to
	 * @param searched is the String we're comparing with
	 * @return true if the two strings are the same or one is a subset of another
	 */
	protected boolean contains(String full, String searched) {

		if (full.indexOf(searched) != -1) {
			return true;
		} else {
			return false;
		}
	}

	/**
	 * Reads the contents of an entire text file into a single String and returns an array
	 * @param fileName is the filename and location of the text file we want to read
	 * @return inputArray each index of the array contains a line of the text file
	 */
	protected String[] readFile(String fileName) {

		String inputString = "";
		String[] inputArray;

		InputStream is = getClass().getResourceAsStream(fileName);
		StringBuffer sb = new StringBuffer();

		try {
			int chars;

			while ((chars = is.read()) != -1) {
				sb.append((char) chars);
			}
			inputString = sb.toString();
			inputArray = splitString(inputString, "\n");

			return inputArray;
		} catch (Exception e) {}
```

```
		return null;
	}

	/**
	 * Takes a String and splits it where the particular character is found and is
	 * stored in an array.
	 * the class 'Regex' is not included in the J2ME library
	 * @param aString is the string we want to split
	 * @param splitCondition - identifies where to split the string
	 * @return result which is an array
	 */
	protected String[] splitString(String aString, String splitCondition) {
		Vector nodes = new Vector();
		String separator = splitCondition;

		int index = aString.indexOf(separator);

		while (index >= 0) {
			nodes.addElement(aString.substring(0, index));
			aString = aString.substring(index + separator.length());
			index = aString.indexOf(separator);
		}

		nodes.addElement(aString);

		String[] result = new String[ nodes.size() ];

		if (nodes.size() > 0) {
			for (int loop = 0; loop < nodes.size(); loop++) {
				result[loop] = (String) nodes.elementAt(loop);
			}
		}
		return result;
	}

	/**
	 * Takes a String and splits it where the colon (:), symbol is found
	 * the class 'Regex' is not included in the J2ME library
	 * @param aString is the string we want to split
	 * @return result which is an array
	 */
	protected String[] splitString(String aString) {
		String[] result = splitString(aString, ":");
		return result;
	}
```

```
	/**
	 * Takes a String and returns a true or false boolean value
	 * depending on whether the String contains only numeral digits
	 * or not
	 * @param aString is the string we want to test
	 * @return true if the String only contains numeral digits
	 */
	public boolean containsOnlyNumbers(String aString) {

		if (aString == null || aString.length() == 0) {
			return false;
		}

		for (int i = 0; i < aString.length(); i++) {

			if (!Character.isDigit(aString.charAt(i))
					&& aString.charAt(i) != '.') {
				return false;
			}
		}

		return true;
	}

}
```

MIDletForms.java

```
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;

/**
 * MIDletForms is used in conjunctions with the RunMIDlet class. It's simply a class containing
 * the forms and the form properties of the MIDlet created in the RunMidlet class
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class MIDletForms {

   protected Form welcomeForm, loginForm, mainMenuForm, loginNotSuccessForm,
   subjectSelectionForm, weekSelectionForm,
         answerQuestionsForm, answeredQuestionsForm, statisticsForm;
   protected TextField userIDField, passwordField;
   protected ChoiceGroup mainMenuChoice, subjectGroup, weekGroup, answerOptions, answeredOptions;
   protected StringItem mainMenuMessageStringItem, subjectSelectMessage, weekSelectMessage, AQFormMessage,
         AAQFormMessage, correctAnswer, answerChosen, isCorrect, statisticsHeader, statisticsBody;
   protected Command nextCommand, next, back, save, next2, back2;

   /**
    * Constructor for MIDletForms, which needs to pass in the RunMIDlet class
    *
    * @param r takes the RunMIDlet class
    */
   protected MIDletForms(RunMIDlet r) {
      StringItem space = new StringItem(null, "\n \n \n");
      StringItem space2 = new StringItem(null, "\n");
      StringItem space3 = new StringItem(null, "\n \n");
      StringItem space4 = new StringItem(null, "\n \n");
      StringItem space5 = new StringItem(null, "\n \n");

      // Welcome Form
      welcomeForm = new Form("Welcome");
      welcomeForm.setCommandListener(r);
      welcomeForm.addCommand(new Command("Exit", Command.EXIT, 0));
      welcomeForm.setTitle("Welcome Page");

      // Login Form
      loginForm = new Form("login");
      loginForm.setCommandListener(r);
      loginForm.addCommand(new Command("Exit", Command.EXIT, 0));
      loginForm.addCommand(new Command("OK", Command.OK, 0));
      loginForm.setTitle("Login Page");

      userIDField = new TextField("User ID:  ", null, 12, TextField.ANY);
      userIDField.setLayout(3);
      passwordField = new TextField("Password:", null, 12, TextField.PASSWORD);
      passwordField.setLayout(3);

      StringItem footnote = new StringItem(null,
            "\n \n \nFor Testing Purposes:\nUsername: 123456\nPassword: password");

      loginForm.append(space);
      loginForm.append(userIDField);
      loginForm.append(passwordField);
      loginForm.append(space2);
```

```
loginForm.append(footnote);

// Login Success Form
mainMenuForm = new Form("Login Successful");
mainMenuForm.setCommandListener(r);
mainMenuForm.addCommand(new Command("Exit", Command.EXIT, 0));
mainMenuForm.addCommand(new Command("Next", Command.OK, 0));
mainMenuForm.setTitle("Main Menu");
mainMenuChoice = new ChoiceGroup(null, Choice.EXCLUSIVE);
mainMenuChoice.setLayout(1);

mainMenuChoice.append("Take a Quiz", null);
mainMenuChoice.append("View Results of an Already Answered Quiz", null);
mainMenuChoice.append("View Current Statistics", null);
mainMenuMessageStringItem = new StringItem(null, "");

mainMenuForm.append(mainMenuMessageStringItem);
mainMenuForm.append(space3);
mainMenuForm.append(mainMenuChoice);

// Login Unsuccessful Success Form
loginNotSuccessForm = new Form("Login UnSuccess");
loginNotSuccessForm.setCommandListener(r);
loginNotSuccessForm.addCommand(new Command("Exit", Command.EXIT, 0));
loginNotSuccessForm.addCommand(new Command("Back", Command.OK, 0));
loginNotSuccessForm.setTitle("Login Unsuccessful");
StringItem successMess = new StringItem(null,
    "Your login was unsuccessful, please go back and try again");

loginNotSuccessForm.append(successMess);

// Subject Selection Form
subjectSelectionForm = new Form("Subject Selection");
subjectSelectionForm.setCommandListener(r);
subjectSelectionForm.addCommand(new Command("Back", Command.EXIT, 0));
subjectSelectionForm.addCommand(new Command("Next", Command.OK, 0));
subjectSelectionForm.setTitle("Subject Selection");
subjectSelectMessage = new StringItem(null, "");
subjectGroup = new ChoiceGroup(null, Choice.EXCLUSIVE);
subjectSelectionForm.append(subjectSelectMessage);
subjectSelectionForm.append(space4);
subjectSelectionForm.append(subjectGroup);

// Week Selection Form
weekSelectionForm = new Form("Week Selection");
weekSelectionForm.setCommandListener(r);
weekSelectionForm.addCommand(new Command("Back", Command.EXIT, 0));
weekSelectionForm.setTitle("Week Selection");
weekSelectMessage = new StringItem(null, "");
nextCommand = new Command("Next", Command.OK, 0);

// Answer Questions Form
answerQuestionsForm = new Form("Answer Questions");
answerQuestionsForm.setCommandListener(r);
answerQuestionsForm.addCommand(
    new Command("Save & Quit", Command.EXIT, 0));
next = new Command("Next", Command.OK, 1);
back = new Command("Back", Command.OK, 2);
save = new Command("Save", Command.OK, 3);
answerQuestionsForm.addCommand(next);
answerQuestionsForm.addCommand(back);
answerQuestionsForm.addCommand(save);
answerQuestionsForm.setTitle("Answer Questions");
```

```
        AQFormMessage = new StringItem(null, "");

        answerQuestionsForm.append(AQFormMessage);

        // Answered Questions Form
        answeredQuestionsForm = new Form("Answer Questions - Results");
        answeredQuestionsForm.setCommandListener(r);
        answeredQuestionsForm.addCommand(new Command("Quit", Command.EXIT, 0));
        next2 = new Command("Next", Command.OK, 1);
        back2 = new Command("Back", Command.OK, 2);
        answeredQuestionsForm.addCommand(next2);
        answeredQuestionsForm.addCommand(back2);
        answeredQuestionsForm.setTitle("Answered Questions - Results");
        AAQFormMessage = new StringItem(null, "");
        correctAnswer = new StringItem(null, "");
        answerChosen = new StringItem(null, "");
        isCorrect = new StringItem(null, "");

        answeredQuestionsForm.append(AAQFormMessage);

        // Statistics Form
        statisticsForm = new Form("Statistics");
        statisticsForm.setCommandListener(r);
        statisticsForm.addCommand(new Command("Back", Command.OK, 0));
        statisticsForm.setTitle("Current Statistics");
        statisticsHeader = new StringItem(null,
              "Here are the statistics for the results so far");
        statisticsBody = new StringItem(null, "");

        statisticsForm.append(statisticsHeader);
        statisticsForm.append(space5);
        statisticsForm.append(statisticsBody);
    }
}
```

RunMidlet.java

```
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.StringItem;
import javax.microedition.midlet.MIDlet;

/**
 * RunMIDlet is the main class that is run when a mobile device runs the MIDlet application
 * This application is a multiple choice answer quiz designed for university students, in which this
 * application is intended to reside on a mobile device, which connects to a servlet, typically on the Internet
 * which connects to a database and fetches the questions and stores a students selected answer
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class RunMIDlet extends MIDlet implements Runnable, CommandListener {
            //Declare global variables
   private Display d; //The main display
   private String message, subjectSelectMessage, subjectChoice, userID;
   private MIDletForms forms; //Declare instance of our MIDletForms class
   private ServletConnector connector; //Declare an instance of our ServletConnector class
   private ManFunctions manFun; //Declare an instance of our ManFunctions class
   private String[] subjectList, weeksList;
   private int weekNo, questionNo, maxQuestions, selector, isFirstSubjectListArrival;;
   protected int[] session; //stores a users selected ChoiceGroup

   /**
    * RunMIDlet Constructor
    */
   public RunMIDlet() {
   //initialise global variables
      forms = new MIDletForms(this); //our MIDletForms class instance
      connector = new ServletConnector(); //our ServletConenctor class instance
      manFun = new ManFunctions(); //our ManFunctions class instance
      questionNo = 0;
      maxQuestions = 0;
      session = new int[100];
   }

   /**
    * Similar to the 'main' method - this starts the applications execution
    */
   public void startApp() {
      if (d == null) {
         d = Display.getDisplay(this);
      }
      d.setCurrent(forms.welcomeForm); //set first form visible

      Thread t = new Thread(this);

      t.start();

   }

   /**
    * Used to pause an application - not used in this application
    */
   public void pauseApp() {}

   /**
```

```
 * Used to force quit an application - not used in this application
 */
public void destroyApp(boolean unconditional) {}

/**
 * Used to listen for I/O input and execute the corresponding code
 * @param c is any I/O command picked up
 * @param s is the current display we are listening to
 */
public void commandAction(Command c, Displayable s) {

        /**
    * Whenever the 'Exit' command type is pressed, do the following...
    * This allows for easier navigation due to not having to remember an extensive
    * list of created commands, only a form name.
    */
   if (c.getCommandType() == Command.EXIT) {
      if (d.getCurrent().equals(forms.subjectSelectionForm)) {
         d.setCurrent(forms.mainMenuForm);
      } else if (d.getCurrent().equals(forms.weekSelectionForm)) {
         d.setCurrent(forms.subjectSelectionForm);
      } else if (d.getCurrent().equals(forms.answerQuestionsForm)) {
         questionNo = 0;
         saveQuestions();
         d.setCurrent(forms.mainMenuForm);
      } else if (d.getCurrent().equals(forms.answeredQuestionsForm)) {
         questionNo = 0;
         d.setCurrent(forms.mainMenuForm);
      } else {
         System.out.println("Exiting");
         notifyDestroyed();
      }
   }

   /**
    * Whenever the 'OK' command type is pressed, do the following...
    * This allows for easier navigation due to not having to remember an extensive
    * list of created commands, only a form name.
    */
   if (c.getCommandType() == Command.OK) {
      if (d.getCurrent().equals(forms.welcomeForm)) {
         d.setCurrent(forms.loginForm);

      } else if (d.getCurrent().equals(forms.loginForm)) {

         login();

      } else if (d.getCurrent().equals(forms.loginNotSuccessForm)) {
         d.setCurrent(forms.loginForm);

      } else if (d.getCurrent().equals(forms.mainMenuForm)) {
         if (forms.mainMenuChoice.getSelectedIndex() == 0) {
            subjectSelectMessage = "Please Chose The Subject to Answer Questions For";
            selector = 0; //to allow a different action to execute using the same command
         } else if (forms.mainMenuChoice.getSelectedIndex() == 1) {
            subjectSelectMessage = "Please Chose Which Subject You Previously Answered Questions For";
            selector = 1;
         } else if (forms.mainMenuChoice.getSelectedIndex() == 2) {
            subjectSelectMessage = "Please Chose The Subject You'd Like To View Statistics For";
            selector = 2;
         }
         querySubject();
         d.setCurrent(forms.subjectSelectionForm);
```

```
} else if (d.getCurrent().equals(forms.subjectSelectionForm)) {
   subjectChoice = subjectList[forms.subjectGroup.getSelectedIndex() + 1];
   if (selector == 2) {
      d.setCurrent(forms.statisticsForm);
      retrieveStatistics();
   } else {
      queryWeeks(subjectChoice, selector);
      d.setCurrent(forms.weekSelectionForm);
   }

} else if (d.getCurrent().equals(forms.statisticsForm)) {
   d.setCurrent(forms.subjectSelectionForm);

} else if (d.getCurrent().equals(forms.weekSelectionForm)) {

   questionNo = 0; //reset the global questionNo
   weekNo = forms.weekGroup.getSelectedIndex() + 1;
   if (selector == 0) {
      d.setCurrent(forms.answerQuestionsForm);
      retrieveQuestion();

   } else if (selector == 1) {

      d.setCurrent(forms.answeredQuestionsForm);
      retrieveAnsweredQuestion();
      queryWeekStatistics();

   }
} else if (c.getLabel().equalsIgnoreCase("next")
      && d.getCurrent().equals(forms.answerQuestionsForm)) {
   if (questionNo != 0) { //using for navigation, we don't want to go into negative integers
      session[questionNo] = forms.answerOptions.getSelectedIndex();
   }
   questionNo++;
   maxQuestions = questionNo; //set maxQuestions as we iteratively save answered questions to a database
   retrieveQuestion();
} else if (c.getLabel().equalsIgnoreCase("next")
      && d.getCurrent().equals(forms.answeredQuestionsForm)) {
   questionNo++;
   retrieveAnsweredQuestion();
} else if (c.getLabel().equalsIgnoreCase("back")
      && d.getCurrent().equals(forms.answerQuestionsForm)) {
   if (questionNo != 0) {
      session[questionNo] = forms.answerOptions.getSelectedIndex();
      questionNo--;
      retrieveQuestion();
   } else {
      d.setCurrent(forms.weekSelectionForm);
   }
} else if (c.getLabel().equalsIgnoreCase("back")
      && d.getCurrent().equals(forms.answeredQuestionsForm)) {
   if (questionNo != 0) {
      questionNo--;
      retrieveAnsweredQuestion();
   } else {
      d.setCurrent(forms.weekSelectionForm);
   }

} else if (c.getLabel().equalsIgnoreCase("save")) {
   saveQuestions();

}
```

```
    }

  }

  /**
   * Contains the code that will be executed when the thread is stared, necessary method
   * due to implementing 'Runnable', here we have the starting point to the application, contained
   * within a thread as network failures can cause deadlocks which block anything calling the application
   */
  public void run() {
    forms.welcomeForm.setTitle("Welcome Form");
    StringItem stringItem;

    if (manFun.contains(connector.request("connect"), "connected") == true) {

      message = "\n \n \nYou have connected successfully, please log in to continue";
      stringItem = new StringItem(null, message);
      forms.welcomeForm.addCommand(new Command("Login", Command.OK, 1));
      stringItem.setLayout(3);
      forms.welcomeForm.append(stringItem);
      System.out.println("Connected to MIDlet");
      isFirstSubjectListArrival = 1;
    } else {
      message = "\n \n \nYou have not been able to connect successfully, please exit and try again";
      stringItem = new StringItem(null, message);
      stringItem.setLayout(3);
      forms.welcomeForm.append(stringItem);
      System.out.println("Cannot Connect To MIDlet");
    }
  }

  /**
   * This method contacts the ServletConnector class to athorise a login request, which
   * in turn connects to a servlet on a network which connects to a database
   */
  protected void login() {

    String loginString = "login:" + forms.userIDField.getString() + ":"
        + forms.passwordField.getString();
    String userDetails = connector.request(loginString);

    if (manFun.contains(userDetails, "userdetails") == true) {

      String[] userDetailsArray = manFun.splitString(userDetails);
      String firstName = userDetailsArray[1];
      String secondName = userDetailsArray[2];

      userID = forms.userIDField.getString();

      forms.mainMenuMessageStringItem.setText(
          "Welcome " + firstName + " " + secondName + "\n"
          + "Your login was successful \nPlease choose a task from the menu");

      d.setCurrent(forms.mainMenuForm);
      System.out.println("Login Successful");
    } else {
      d.setCurrent(forms.loginNotSuccessForm);
      System.out.println("Login Unsuccessful");
    }
  }

  /**
```

```
 * This method contacts the ServletConnector class to query for a list of subject names, which
 * in turn connects to a servlet on a network which connects to a database
 */
protected void querySubject() {
   String inputString = connector.request("querySubjects");

   if (manFun.contains(inputString, "nosubjects") == true) {
      forms.subjectSelectMessage.setText("There Are No Subjects Available");
   } else {
      questionNo = 0;
      subjectList = manFun.splitString(inputString);
      forms.subjectSelectMessage.setText(subjectSelectMessage);

      if (isFirstSubjectListArrival == 1) {
         isFirstSubjectListArrival = 0;
         for (int i = 1; i < subjectList.length - 1; i++) {
            forms.subjectGroup.append(subjectList[i], null);
         }
      }
   }
}

/**
 * This method contacts the ServletConnector class to query for a list of week number, which
 * in turn connects to a servlet on a network which connects to a database.
 * @param subjectName - The subject name of the week list to fetch
 * @param selector - defines the difference in which subject weeks to collect - the results or the questions
 */
protected void queryWeeks(String subjectName, int selector) {
   String weeks = "";

   if (selector == 0) {
      weeks = connector.request("queryWeeks:" + subjectName);
   } else {
      weeks = connector.request("queryResultWeeks:" + subjectName);
   }

   StringItem space = new StringItem(null, "\n \n");
   StringItem weekSelectMessage = new StringItem(null, "");

   forms.weekGroup = new ChoiceGroup(null, Choice.EXCLUSIVE);
   forms.weekSelectionForm.deleteAll();

   weeksList = manFun.splitString(weeks);

   if (manFun.contains(weeks, "noweeks") == true) {
      if (selector == 0) {
         weekSelectMessage.setText(
               "Sorry, There Are weeks Available for This Subject");
      } else {
         weekSelectMessage.setText(
               "Sorry, You Have Not Previously Answered Any Questions For This Subject");
      }
      forms.weekSelectionForm.append(weekSelectMessage);
      forms.weekSelectionForm.append(space);
      forms.weekSelectionForm.append(forms.weekGroup);
      forms.weekSelectionForm.removeCommand(forms.nextCommand);
   } else {
      forms.weekSelectionForm.removeCommand(forms.nextCommand);
      if (selector == 0) {
         weekSelectMessage.setText("Please Select An Available Week");
      } else {
         weekSelectMessage.setText(
```

```
                "Please Select A Week to View Results, If You Cannot See Your Week, You Either Haven't Taken "
                    + "That Weeks Quiz or You Forgot to Save Your Results");
        }
        forms.weekSelectionForm.append(weekSelectMessage);
        forms.weekSelectionForm.append(space);
        forms.weekSelectionForm.append(forms.weekGroup);
        forms.weekSelectionForm.addCommand(forms.nextCommand);

        for (int i = 1; i < weeksList.length - 1; i++) {
          forms.weekGroup.append(weeksList[i], null);
        }
      }
    }

    /**
     * This method contacts the ServletConnector class to query for a single question
     * and available answer options, which the user may answer, which
     * in turn connects to a servlet on a network which connects to a database
     */
    protected void retrieveQuestion() {
      String inputString = connector.request(
            "retrieveQuestion:" + weekNo + ":" + questionNo + ":"
            + subjectChoice);
      String[] inputArray = manFun.splitString(inputString);
      StringItem space = new StringItem(null, "\n \n \n");

      forms.answerOptions = new ChoiceGroup(null, Choice.EXCLUSIVE);
      forms.answerQuestionsForm.deleteAll();

      if (manFun.contains(inputString, "retrievedQuestion")) {
        forms.answerQuestionsForm.removeCommand(forms.next);
        forms.AQFormMessage.setText(questionNo + ") " + inputArray[1]);
        forms.answerQuestionsForm.append(forms.AQFormMessage);
        forms.answerQuestionsForm.append(space);
        forms.answerQuestionsForm.append(forms.answerOptions);
        forms.answerQuestionsForm.addCommand(forms.next);

        for (int i = 2; i < inputArray.length - 1; i++) {
          forms.answerOptions.append(i - 1 + ") " + inputArray[i], null);
        }
        if (session[questionNo] > 0) {
          forms.answerOptions.setSelectedIndex(session[questionNo], true);
        }

      } else if (manFun.contains(inputString, "noquestion") && questionNo > 1) {
        forms.AQFormMessage.setText(
              "There are no more questions for this week");
        forms.answerQuestionsForm.removeCommand(forms.next);
        forms.answerQuestionsForm.append(forms.AQFormMessage);
        maxQuestions = questionNo - 1;

      } else {
        forms.answerQuestionsForm.removeCommand(forms.next);
        forms.AQFormMessage.setText(
              "\n \n \n \n \nTo answer a question please chose from one of the four multiple choices. "
                  + "Your selections can be saved at any time by selection save from the right hand menu. "
                  + "You may re-attempt a question at any time during this session.");
        forms.answerQuestionsForm.append(forms.AQFormMessage);
        forms.answerQuestionsForm.addCommand(forms.next);
      }
    }

    /**
```

```
 * This method contacts the ServletConnector class to query for the results
 * of an answered question, which in turn connects to a servlet on a
 *  network which connects to a database
 */
protected void retrieveAnsweredQuestion() {
   String inputString = connector.request(
         "retrieveAnsweredQuestion:" + weekNo + ":" + questionNo + ":"
         + subjectChoice + ":" + userID);
   String[] inputArray = manFun.splitString(inputString);
   StringItem space = new StringItem(null, "\n \n \n");

   int answerChosen = 0;
   int correctAnswer = 0;

   if (inputArray.length > 2) {
      answerChosen = Integer.parseInt(inputArray[1]);
      correctAnswer = Integer.parseInt(inputArray[2]);
   }

   forms.answeredQuestionsForm.deleteAll();
   forms.answeredOptions = new ChoiceGroup(null, Choice.EXCLUSIVE);

   if (manFun.contains(inputString, "retrievedQuestion")) {
      forms.answeredQuestionsForm.removeCommand(forms.next2);
      forms.AAQFormMessage.setText(questionNo + ") " + inputArray[3]);
      forms.answeredQuestionsForm.append(forms.AAQFormMessage);
      forms.answeredQuestionsForm.append(space);
      forms.answeredQuestionsForm.append(forms.answeredOptions);
      forms.answeredQuestionsForm.addCommand(forms.next2);

      for (int i = 4; i < inputArray.length - 1; i++) {
         forms.answeredOptions.append(i - 3 + ") " + inputArray[i], null);
      }

      forms.answerChosen.setText(
            "\n \nThe Answer You Chose is: " + answerChosen);
      forms.correctAnswer.setText(
            "The Correct Answer is: " + correctAnswer);
      if (answerChosen == correctAnswer) {

         forms.isCorrect.setText("You Got This Question Correct");
      } else {
         forms.isCorrect.setText("You Got This Question Incorrect");
      }
      forms.answeredQuestionsForm.append(forms.answerChosen);
      forms.answeredQuestionsForm.append(forms.correctAnswer);
      forms.answeredQuestionsForm.append(forms.isCorrect);

      if (inputArray.length > 1 && correctAnswer > 0) {
         forms.answeredOptions.setSelectedIndex(correctAnswer - 1, true);
      }

   } else if (manFun.contains(inputString, "noquestion") && questionNo > 1) {
      forms.AAQFormMessage.setText("This is the end of the results");
      forms.answeredQuestionsForm.removeCommand(forms.next2);
      forms.answeredQuestionsForm.append(forms.AAQFormMessage);

   } else {
      forms.answeredQuestionsForm.removeCommand(forms.next2);
      forms.AAQFormMessage.setText(queryWeekStatistics());

      forms.answeredQuestionsForm.append(forms.AAQFormMessage);
      forms.answeredQuestionsForm.addCommand(forms.next2);
```

```
        }
    }

    /**
     * This method contacts the ServletConnector class to query for the statistical results
     * for a particular week that questions have been answered for, which
     * in turn connects to a servlet on a network which connects to a database
     */
    protected String queryWeekStatistics() {
        String responseString = "";
        String inputString = connector.request(
                "retrieveWeekStatistics:" + weekNo + ":" + subjectChoice + ":"
                + userID);
        String[] inputArray = manFun.splitString(inputString);

        responseString = "\n \n \n \n \nSelect 'Next' from the menu to view your results for each question\n \n"
                + "You scored " + inputArray[0] + " out of " + inputArray[1]
                + " this week\nGiving you a total of " + inputArray[2];

        return responseString;
    }

    /**
     * This method contacts the ServletConnector class to query for the statistical results
     * for all weeks that questions have been answered for, which
     * in turn connects to a servlet on a network which connects to a database
     */
    protected void retrieveStatistics() {
        String inputString = connector.request(
                "retrieveStatistics:" + subjectChoice + ":" + userID);
        String[] inputArray = manFun.splitString(inputString);

        if (manFun.containsOnlyNumbers(inputArray[1])) {
            forms.statisticsBody.setText(
                    "\nYour Current Average = " + inputArray[0] + "%"
                    + "\nEveryone's Total Average = " + inputArray[1] + "%"
                    + "\nThe Highest Average = " + inputArray[2] + "%"
                    + "\nThe Lowest Averaage of = " + inputArray[3]);
        } else {
            forms.statisticsBody.setText(
                    "There are no results available for this subject");
        }
    }

    /**
     * Iteratively saves the answer choices of all answered questions, which has been being
     * stored in an array. This is done via passing in a string to a servlet on some network
     * by the ServletConnector
     */
    protected void saveQuestions() {
        for (int i = 1; i <= maxQuestions; i++) {
            connector.request(
                    "saveQuestion:" + subjectChoice + ":" + userID + ":"
                    + weekNo + ":" + i + ":" + (session[i] + 1));
        }

    }
}
```

ServletConnector.java

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;

/**
 * ServletConnector is the class that connects to a servlet on some network via HTTP requests and responses.
 * This class simply outputs a String, which is the request which in this case has been created similarly to
 * Sockets, the string will contain a command followed by parameters, seperated by the colon symbol (:).
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class ServletConnector {
		//declare global variables
   private String MIDletOutput;
   protected String serverInput;
   protected ManFunctions manFun;
   private String URL, fileName;
   private String[] fileInput;

   /**
    * ServletConnector constructor. Upon constructing, it will open a text file to seek
    * an url, which is the location on the network of where to connect to
    */
   public ServletConnector() {
		//initialise global variables
      manFun = new ManFunctions(); //our ManFunctions class instance

      fileName = "serverOptions.txt";
      fileInput = manFun.readFile(fileName);

      MIDletOutput = "";
      serverInput = "";
      URL = "";

			for (int i = 0; i < fileInput.length; i++) {
         if (manFun.contains(fileInput[i], "http")) {
            URL = fileInput[i];
         }
      }
   }

   /**
    * The method which send the quest String to the servlet on some network
    * @param s is the String sent to the servlet
    * @return response - the response string from the servlet
    */
   protected String request(String s) {
      String response = "";

      MIDletOutput = "MIDletOutput=" + s;

      HttpConnection hc = null;
      InputStream in = null;
      OutputStream out = null;

      try {
         hc = (HttpConnection) Connector.open(URL);
         hc.setRequestMethod(HttpConnection.POST);
         hc.setRequestProperty("Content-Type",
```

```
                "application/x-www-form-urlencoded");

            out = hc.openOutputStream();
            out.write(MIDletOutput.getBytes()); //convert to byes and send

            in = hc.openInputStream();
            int length = (int) hc.getLength();
            byte[] serverInput = new byte[length];

            in.read(serverInput); //the response from the serlvet

            response = new String(serverInput);
            return response;

        } catch (IOException ioe) {
            System.out.println("Servlet Connection Failure");
            return "Connection Failure";
        } finally {
            try {
                if (out != null) {
                    out.close();
                }
                if (in != null) {
                    in.close();
                }
                if (hc != null) {
                    hc.close();
                }
            } catch (IOException ioe) {}
        }
    }
}
```

Server Side Source Code:

MIDletServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * MIDletServlet is a serlvet acting for a MIDlet which connects to this class remotely, communicating via http
request/responses
 * The prime purpose for this servlet is to allow the MIDlet to connect to a database through an intermediatory. As the
MIDlet
 * is limited to bandwidth and network speeds, this servlet can do the more demanding network tasks such as querying
database
 * and rely the information back to the MIDlet in the form of a string in a similar way to using Sockets
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class MIDletServlet extends HttpServlet {

   private static final long serialVersionUID = 1L;
   private String serverInput = "";
   private DatabaseConnector connector;
   private ManFunctions manFun;

   /**
    * Constructor for MIDletServlet, calling instances of other associated class
    */
   public MIDletServlet() {
      manFun = new ManFunctions();
      connector = new DatabaseConnector();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {

      if (request.getAttributeNames() != null) {
```

```
serverInput = request.getParameter("MIDletOutput");

response.setContentType("text/plain");
PrintWriter out = response.getWriter();

if (serverInput.equalsIgnoreCase("connect")) {
   out.println("connected");
   System.out.println("A MIDlet has Connected");

} else if (serverInput.contains("login")) {

   String[] loginQuery = manFun.splitString(serverInput);

   if (loginQuery.length < 2) {
      System.out.println("No Login Details Entered for Query");
      System.out.println("Login Failed");
      out.println("loginFailed");
   } else {
      String responseString = connector.queryLogin(loginQuery[1],
            loginQuery[2], false);

      if (responseString.length() < 3) {
         out.println("loginFailed");
         System.out.println("No UserID or Password, Login Failed");
      } else {
         out.println(responseString);
      }
   }

} else if (serverInput.contains("querySubjects")) {
   String responseString = connector.querySubjects();

   if (responseString.length() < 3) {
      out.println("nosubjects");
      System.out.println("No Subjects Found");
   } else {
      out.println("subjectList:" + responseString);
   }
} else if (serverInput.contains("queryWeeks")) {

   String[] input = manFun.splitString(serverInput);

   String responseString = connector.queryWeeks(input[1], 0);

   if (responseString.length() < 3) {
      out.println("noweeks");
      System.out.println("No Weeks Found");
   } else {
      out.println("weekList:" + responseString);
   }
} else if (serverInput.contains("queryResultWeeks")) {

   String[] input = manFun.splitString(serverInput);

   String responseString = connector.queryWeeks(input[1], 1);

   if (responseString.length() < 3) {
      out.println("noweeks");
      System.out.println("No Weeks Found");
   } else {
      out.println("weekList:" + responseString);
   }
```

```
        } else if (serverInput.contains("retrieveQuestion")) {
          String[] input = manFun.splitString(serverInput);

          String responseString = connector.retrieveQuestion(input[1],
              input[2], input[3]);

          if (responseString.length() < 3) {
            out.println("noquestion");
            System.out.println("No Question Found");
          } else {
            out.println("retrievedQuestion:" + responseString);
          }
        } else if (serverInput.contains("retrieveAnsweredQuestion")) {
          String[] input = manFun.splitString(serverInput);

          String responseString = connector.retrieveAnsweredQuestion(
              input[1], input[2], input[3], input[4]);

          if (responseString.length() < 6) {
            out.println("noquestion");
            System.out.println("No Question Found");
          } else {
            out.println("retrievedQuestion:" + responseString);
          }
        } else if (serverInput.contains("saveQuestion")) {
          String[] input = manFun.splitString(serverInput);

          connector.saveAnswers(input[1], input[2], input[3], input[4],
              input[5]);
        } else if (serverInput.contains("retrieveWeekStatistics")) {
          String[] input = manFun.splitString(serverInput);
          String responseString = connector.retrieveWeekStatistics(
              input[1], input[2], input[3]);

          out.println(responseString);

        } else if (serverInput.contains("retrieveStatistics")) {
          String[] input = manFun.splitString(serverInput);
          String responseString = connector.retrieveStatistics(input[1],
              input[2]);

          out.println(responseString);

        } else {
          out.println("unknown command");
        }
      } else {
        System.out.println("No MIDlet Connection");
      }
    }
}
```

ManFunctions.java

```
import java.io.InputStream;
import java.util.Vector;
import java.util.regex.Pattern;

/**
 * ManFunctions (Manual/Manipulation Function) class contains re-usable methods which generally
 * manipulate Strings and also check their content.
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class ManFunctions {

            /**
    * The Manfunctions constructor - takes no parameters
    */
   protected ManFunctions() {}

   /**
    * The database doesn't contain a list of weeks for a particular subject so
    * a query has to be done on the database to see which week is associated with each question
    * which comes back as a large list with lots of duplicate week numbers. This method
    * removes the duplicate and as a header 'week' to each week item and returns it as a single
    * string with each week seperated via the colon (:) symbol
    * @param weeksVector is the vector containing the full list of raw weeks
    * @return weeksList - The string containing the sorted weeks
    */
   protected String sortWeeks(Vector<Integer> weeksVector) {

      String weeksList = "";

      if (weeksVector.size() > 0) {

         for (int i = 0; i < weeksVector.size(); i++) {
            if (!(weeksList.contains(weeksVector.get(i).toString()))) {
               weeksList = weeksList + "Week "
                     + weeksVector.get(i).toString() + ":";
            }
         }
      }

      return weeksList;
   }

   /**
    * Reads the contents of an entire text file into a single String and returns an array
    * @param fileName is the filename and location of the text file we want to read
    * @return inputArray each index of the array contains a line of the text file
    */
   protected String[] readFile(String fileName) {

      String inputString = "";
      String[] inputArray;

      InputStream is = getClass().getResourceAsStream(fileName);
      StringBuffer sb = new StringBuffer();

      try {
         int chars;

         while ((chars = is.read()) != -1) {
            sb.append((char) chars);
```

```
            }
            inputString = sb.toString();
            inputArray = splitString(inputString, "\n");

            return inputArray;
        } catch (Exception e) {}
        return null;
    }

    /**
     * Takes a String and splits it where the particular character is found and is
     * stored in an array.
     * @param aString is the string we want to split
     * @param splitCondition - identifies where to split the string
     * @return items which is an array
     */
    protected String[] splitString(String aString, String splitCondition) {
        String REGEX = splitCondition;

        Pattern p = Pattern.compile(REGEX);
        String[] items = p.split(aString);

        return items;
    }

    /**
     * Takes a String and splits it where the colon (:), symbol is found
     * @param aString is the string we want to split
     * @return items which is an array
     */
    protected String[] splitString(String aString) {
            String [] items = splitString(aString, ":");
            return items;
    }

    /**
     * Takes a String and and tests whether it contains only numeral digits or not
     * returning a boolean value
     * @param aString is the string we want to split
     * @return true if the string only contains numbers
     */
    public boolean containsOnlyNumbers(String aString) {

                        if (aString == null || aString.length() == 0) {
                                return false;
                        }

                        for (int i = 0; i < aString.length(); i++) {

                                if (!Character.isDigit(aString.charAt(i))
                                        && aString.charAt(i) != '.') {
                                        return false;
                                }
                        }
                        return true;
            }
}
```

DatabaseConnector.java

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DecimalFormat;
import java.util.Vector;

/**
 * DatabaseConnector is the class which connects to a mysql database on some network and
 * contains a list of methods to be used by the MIDlet application and the quiz administration panel
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class DatabaseConnector {
        //declare global variables
   private Connection conn;
   private ManFunctions manFun;
   private String responseString;
   private String database, user, password, url, mysqlClass;
   private String[] serverDetails;

   /**
    * DatabaseConnector constructor - also uses a ManFunctions methods to read
    * a text file which contains database location and authorisation details
    */
   public DatabaseConnector() {
        //initialise global variables
      manFun = new ManFunctions();
      responseString = "";
      serverDetails=manFun.readFile("serverOptions.txt");

      database = serverDetails[5];
      user = serverDetails[8];
      password =  serverDetails[11];
      url =  serverDetails[14] + database;
      mysqlClass =  serverDetails[17];
   }

   /**
    * Makes the connection to the database
    * @throws IOException
    */
   protected void connect()
      throws IOException {

      try { // where the mysql driver is found
         Class.forName(mysqlClass).newInstance();
      } catch (Exception e) {
         System.err.println(e);
      }

      // connecting to database
      try {
         conn = DriverManager.getConnection(url, user, password);
         System.out.println("Connected to MySQL Database");

      } catch (SQLException se) {
         System.err.println(se);
```

```
    }
  }

  /**
   * Queries a login details in database
   * @param userID
   * @param password
   * @param isTutor
   * @throws IOException
   * @return responseString - response from database
   */
  protected String queryLogin(String userID, String password, boolean isTutor) throws IOException {
    connect();
    System.out.println("Querying Users");
    responseString = "";
    String selectSQL = "";

    if (isTutor) {
      selectSQL = "select * from tutors where userid ='" + userID
          + "' && Password ='" + password + "'";
    } else {
      selectSQL = "select * from users where userid ='" + userID
          + "' && Password ='" + password + "'";
    }

    Statement stmt;

    try {
      stmt = conn.createStatement();
      ResultSet rs1 = stmt.executeQuery(selectSQL);

      while (rs1.next()) {
        responseString = "userdetails:" + rs1.getString("firstName")
            + ":" + rs1.getString("secondName");
      }

      stmt.close();

    } catch (SQLException e) {
      responseString = "";
    }
    closeConnection();
    return responseString;

  }

  /**
   * Queries a subject list in database
   * @throws IOException
   * @return responseString - response from database
   *
   */
  protected String querySubjects() throws IOException {
    connect();

    System.out.println("Querying Subjects");
    responseString = "";
    String selectSQL = "select * from subjects";
    Statement stmt;

    try {
      stmt = conn.createStatement();
      ResultSet rs1 = stmt.executeQuery(selectSQL);
```

```
        while (rs1.next()) {
            responseString = responseString + rs1.getString("subjectname")
                    + ":";
        }
        stmt.close();
    } catch (SQLException e) {
        responseString = "";
    }
    closeConnection();
    return responseString;
}

/**
 * Queries a week list in database
 * @throws IOException
 * @return responseString - response from database
 * @param subjectName
 * @param selector
 */
protected String queryWeeks(String subjectName, int selector) throws IOException {
    connect();

    String selectSQL2 = "";
    String subjectID = "";

    System.out.println("Querying Week");
    Vector<Integer> weeksVector = new Vector<Integer>();

    responseString = "";

    String selectSQL = "select * from subjects where subjectname='"
            + subjectName + "'";
    Statement stmt;

    try {
        stmt = conn.createStatement();
        ResultSet rs1 = stmt.executeQuery(selectSQL);

        while (rs1.next()) {
            subjectID = rs1.getString("subjectid");
        }
        stmt.close();
    } catch (SQLException e) {
        responseString = "";
    }

    if (selector == 0) {

        selectSQL2 = "select * from questions where subjectid = '"
                + subjectID + "'";
    } else {
        selectSQL2 = "select * from results where subjectid = '" + subjectID
                + "'";
    }
    Statement stmt2;

    try {
        stmt2 = conn.createStatement();
        ResultSet rs2 = stmt2.executeQuery(selectSQL2);

        while (rs2.next()) {
            weeksVector.add(rs2.getInt("weekno"));
```

```
        }

        stmt2.close();
        responseString = manFun.sortWeeks(weeksVector);

    } catch (SQLException e) {
        responseString = "";
    }

    closeConnection();

    return responseString;
}

/**
 * Queries questions for a particular question in database
 * @throws IOException
 * @return responseString - response from database
 * @param weekNo
 * @param questionNo
 * @param subjectName
 */
protected String retrieveQuestion(String weekNo, String questionNo, String subjectName)
throws IOException {
    connect();

    String theQuestion = "";
    Vector<String> answers = new Vector<String>();

    responseString = "";

    String subjectID = "";

    System.out.println("Querying Questions");

    String selectSQL = "select * from subjects where subjectname='"
            + subjectName + "'";
    Statement stmt;

    try {
        stmt = conn.createStatement();
        ResultSet rs1 = stmt.executeQuery(selectSQL);

        while (rs1.next()) {
            subjectID = rs1.getString("subjectid");
        }

        stmt.close();
    } catch (SQLException e) {
        responseString = "";
    }

    String selectSQL2 = "select * from questions where subjectid = '"
            + subjectID + "' && weekno = '" + weekNo + "' && questionno='"
            + questionNo + "'";
    Statement stmt2;

    try {
        stmt2 = conn.createStatement();
        ResultSet rs2 = stmt2.executeQuery(selectSQL2);

        while (rs2.next()) {
            theQuestion = rs2.getString("question");
```

```
            answers.add(rs2.getString("answers"));

        }
        stmt2.close();

    } catch (SQLException e) {
        responseString = "";
    }

    closeConnection();

    for (int i = 0; i < answers.size(); i++) {
        responseString = responseString + answers.get(i) + ":";
    }
    responseString = theQuestion + ":" + responseString;

    if (responseString.equalsIgnoreCase(":")) {
        responseString = "";
    }

    return responseString;
}

/**
 * Queries a results for a particular question in database
 * @throws IOException
 * @return responseString - response from database
 * @param weekNo
 * @param questionNo
 * @param subjectName
 * @param userID
 */
protected String retrieveAnsweredQuestion(String weekNo, String questionNo, String subjectName, String userID)
throws IOException {
    connect();

    String theQuestion = "";
    int answerChosen = 0;
    int correctAnswer = 0;

    Vector<String> answers = new Vector<String>();

    responseString = "";

    String subjectID = "";

    System.out.println("Querying Results");

    String selectSQL = "select * from subjects where subjectname='"
            + subjectName + "'";
    Statement stmt;

    try {
        stmt = conn.createStatement();
        ResultSet rs1 = stmt.executeQuery(selectSQL);

        while (rs1.next()) {
            subjectID = rs1.getString("subjectid");
        }
        stmt.close();
    } catch (SQLException e) {
        responseString = "";
    }
```

```
    String selectSQL2 = "select * from results, questions where results.subjectid = '"
            + subjectID + "' && results.weekno = '" + weekNo
            + "' && results.questionno='" + questionNo
            + "' && questions.questionno='" + questionNo
            + "' && questions.weekno = '" + weekNo
            + "' && questions.subjectid= '" + subjectID
            + "' && results.userid = '" + userID + "'";
    Statement stmt2;

    try {
        stmt2 = conn.createStatement();
        ResultSet rs2 = stmt2.executeQuery(selectSQL2);

        int i = 0;

        while (rs2.next()) {
            i++;
            theQuestion = rs2.getString("question");
            answers.add(rs2.getString("answers"));
            answerChosen = rs2.getInt("results.answerno");
            if (rs2.getInt("correctanswer") == 1) {
                correctAnswer = i;
            }

        }
        stmt2.close();
    } catch (SQLException e) {
        responseString = "";

    }

    closeConnection();

    for (int i = 0; i < answers.size(); i++) {
        responseString = responseString + answers.get(i) + ":";
    }
    responseString = correctAnswer + ":" + answerChosen + ":" + theQuestion
            + ":" + responseString;

    if (!(responseString.length() > 3)) {
        responseString = "";
    }

    return responseString;
}

/**
 * inserts details of an answered question in database
 * @throws IOException
 * @param weekNo
 * @param questionNo
 * @param subjectName
 * @param userID
 * @param answerNo
 */
protected void saveAnswers(String subjectName, String userID, String weekNo, String questionNo, String answerNo)
throws IOException {
    connect();
    int selectorInt = 0;
    String isCorrect = "0";

    responseString = "";
```

```
String subjectID = "";

System.out.println("Inserting Results");

String selectSQL = "select * from subjects where subjectname='"
      + subjectName + "'";
Statement stmt;

try {
   stmt = conn.createStatement();
   ResultSet rs1 = stmt.executeQuery(selectSQL);

   while (rs1.next()) {
      subjectID = rs1.getString("subjectid");
   }
   stmt.close();
} catch (SQLException e) {}

String selectSQL2 = "select * from results where subjectid = '"
      + subjectID + "' && weekno = '" + weekNo + "' && questionno='"
      + questionNo + "'" + "&& userid='" + userID + "'";
Statement stmt2;

try {
   stmt2 = conn.createStatement();
   ResultSet rs2 = stmt2.executeQuery(selectSQL2);

   while (rs2.next()) {
      selectorInt = 1;

   }
   stmt2.close();
} catch (SQLException e) {}

String selectSQL3 = "select answerno from questions WHERE questionno='"
      + questionNo + "' && weekno='" + weekNo + "' && subjectID='"
      + subjectID + "' && correctanswer='1'";

;

Statement stmt3;

try {
   stmt3 = conn.createStatement();
   ResultSet rs3 = stmt3.executeQuery(selectSQL3);

   while (rs3.next()) {
      if (answerNo.contains(rs3.getString("answerno"))) {
         isCorrect = "1";
      }

   }
   stmt3.close();
} catch (SQLException e) {}

if (selectorInt == 0) {
   try {
      PreparedStatement pstmt;

      pstmt = conn.prepareStatement(
            "insert into results (subjectid, userid, weekno, questionno, answerno, iscorrect) VALUES ('"
                  + subjectID + "', '" + userID + "', '" + weekNo
```

```
                    + "', '" + questionNo + "', '" + answerNo
                    + "', '" + isCorrect + "')");

            int insert = 0;

            selectorInt = 0;

            insert = pstmt.executeUpdate();
            if (insert == 0) {
               System.out.println(
                     insert + ": Insert answered question failure");
            } else {
               System.out.println(insert + ": Inserting answered question");
            }
            pstmt.close();
         } catch (SQLException e) {
            System.out.println("Insert answered question failure");
         }

      } else {
         try {
            PreparedStatement pstmt;

            pstmt = conn.prepareStatement(
                  "update results SET answerno = '" + answerNo
                  + "', iscorrect='" + isCorrect + "' where subjectid='"
                  + "" + subjectID + "' && userid='" + userID
                  + "' && weekno='" + weekNo + "' && questionno='"
                  + questionNo + "'");

            int insert = 0;

            selectorInt = 0;

            insert = pstmt.executeUpdate();

            if (insert == 0) {
               System.out.println(
                     insert + ": Update answered question failure");
            } else {
               System.out.println(insert + ": Updating answered question");
            }

            pstmt.close();
         } catch (SQLException e) {
            System.out.println("Update answered question failure");
         }
      }

      closeConnection();
   }

   /**
    * Inserts or updates a question into dabase
    * @throws IOException
    * @param weekNo
    * @param questionNo
    * @param subjectName
    * @param theQuestion
    * @param answerNo
    * @param answerOption
    * @param correctAnswer
    */
```

```
protected void insertQuestion(String questionNo, String weekNo, String subjectName,
      String theQuestion, String answerNo, String answerOption, String correctAnswer)
throws IOException {

   connect();

   String subjectID = "nonSubject";

   String selectSQL = "select * from subjects where subjectname='"
         + subjectName + "'";
   Statement stmt;

   try {
      stmt = conn.createStatement();
      ResultSet rs1 = stmt.executeQuery(selectSQL);

      while (rs1.next()) {
         subjectID = rs1.getString("subjectid");

      }

      stmt.close();
   } catch (SQLException e) {}

   if (subjectID.equalsIgnoreCase("nonSubject")) {
      try {
         PreparedStatement pstmt;

         pstmt = conn.prepareStatement(
               "insert into subjects(subjectname) values ('"
                     + subjectName + "')");
         int insert = 0;

         insert = pstmt.executeUpdate();

         if (insert == 0) {
            System.out.println(insert + ": Insert subject failure");
         } else {
            System.out.println(insert + ": Inserting Subject");
         }

         pstmt.close();

         stmt = conn.createStatement();
         ResultSet rs1 = stmt.executeQuery(selectSQL);

         while (rs1.next()) {
            subjectID = rs1.getString("subjectid");

         }

         stmt.close();
      } catch (SQLException e) {
         System.out.println("Insert subject failure");
      }

   }

   String selectSQL2 = "select * from questions where questionno='"
         + questionNo + "'" + "&& weekno = '" + weekNo
         + "' && subjectid = '" + subjectID + "'" + "&& answerno = '"
         + answerNo + "'";
   Statement stmt2;
```

```
boolean doesExist = false;

try {
   stmt2 = conn.createStatement();
   ResultSet rs2 = stmt2.executeQuery(selectSQL2);

   while (rs2.next()) {
      doesExist = true;

   }
   stmt2.close();
} catch (SQLException e) {}

if (doesExist == false) {

   try {
      PreparedStatement pstmt;

      pstmt = conn.prepareStatement(
            "insert into questions (questionno, weekno, answerno, answers, " +
            "correctanswer, subjectid, question)"
                  + " values ('" + questionNo + "', " + "'"
                  + weekNo + "', " + "'" + answerNo + "', " + "'"
                  + answerOption + "', " + "'" + correctAnswer
                  + "', " + "'" + subjectID + "', " + "'"
                  + theQuestion + "')");
      int insert = 0;

      insert = pstmt.executeUpdate();

      if (insert == 0) {
         System.out.println(insert + ": Insert question failure");
      } else {
         System.out.println(insert + ": Inserting question");
      }
      pstmt.close();

   } catch (SQLException e) {
      System.out.println("Insert question failure");
   }

} else {
   try {
      PreparedStatement pstmt;

      pstmt = conn.prepareStatement(
            "update questions set answers='" + answerOption
            + "', correctanswer='" + correctAnswer + "', question='"
            + theQuestion + "'" + " where questionno='" + questionNo
            + "' && weekno='" + weekNo + "' && subjectid='"
            + subjectID + "' && answerno ='" + answerNo + "'");
      int insert = 0;

      insert = pstmt.executeUpdate();

      if (insert == 0) {
         System.out.println(insert + ": Update question failure");
      } else {
         System.out.println(insert + ": Updating question");
      }

      pstmt.close();
```

```
        } catch (SQLException e) {
            System.out.println("Update question failure");
        }

    }

}

/**
 * Queries statistics of the results of a particular week for a particular user from the database
 * @throws IOException
 * @return responseString - response from database
 * @param weekNo
 * @param subjectName
 * @param userID
 */
protected String retrieveWeekStatistics(String weekNo, String subjectName, String userID)
throws IOException {
    String responseString = "";

    connect();

    String subjectID = "";

    responseString = "";

    String selectSQL = "select * from subjects where subjectname='"
            + subjectName + "'";
    Statement stmt;

    try {
        stmt = conn.createStatement();
        ResultSet rs1 = stmt.executeQuery(selectSQL);

        while (rs1.next()) {
            subjectID = rs1.getString("subjectid");

        }
        stmt.close();
    } catch (SQLException e) {
        responseString = "";
    }

    String selectSQL2 = "select * from results, questions where results.subjectid = '"
            + subjectID + "' && questions.subjectid= '" + subjectID
            + "' && results.weekno = '" + weekNo
            + "' && questions.weekno = '" + weekNo + "' && results.userid='"
            + userID + "' && questions.questionno=results.questionno";
    Statement stmt2;

    System.out.println(selectSQL2);

    try {
        stmt2 = conn.createStatement();
        ResultSet rs2 = stmt2.executeQuery(selectSQL2);

        int i = 0;
        int correctAnswerCount = 0;
        int resultsCount = 0;
        int tempInt = 0;

        while (rs2.next()) {
```

```
            i++;

            if (!(rs2.getInt("resultid") == tempInt)) {
               resultsCount++;
               tempInt = rs2.getInt("resultid");
               i = 1;
            }

            if (rs2.getInt("correctAnswer") == 1) {
               if (rs2.getInt("results.answerno") == i) {
                  correctAnswerCount++;
               }
            }

         }
         stmt2.close();

         double percentage = 0.00;
         DecimalFormat deci = new DecimalFormat("##.##");

         percentage = ((double) correctAnswerCount / (double) resultsCount)
               * 100;

         responseString = correctAnswerCount + ":" + resultsCount + ":"
               + deci.format(percentage) + "%";

      } catch (SQLException e) {
         responseString = "";
      }

      closeConnection();

      return responseString;
   }

   /**
    * Queries statistics of the results of a particular subject for for all users
    * @throws IOException
    * @return responseString - response from database
    * @param subjectName
    * @param userID
    */
   public String retrieveStatistics(String subjectName, String userID) throws IOException {
      connect();
      String responseString = "";

      String subjectID = "";

      String selectSQL = "select * from subjects where subjectname='"
            + subjectName + "'";
      Statement stmt;

      try {
         stmt = conn.createStatement();
         ResultSet rs1 = stmt.executeQuery(selectSQL);

         while (rs1.next()) {
            subjectID = rs1.getString("subjectid");

         }
         stmt.close();
      } catch (SQLException e) {
         responseString = "";
```

```
}

String selectSQL2 = "select * from results where subjectid='"
      + subjectID + "'";
Statement stmt2;

DecimalFormat deci = new DecimalFormat("##.##");
double userAverage = 0.00;
double totalAverage = 0.00;
double highestAverage = 0.00;
double lowestAverage = 100.00;

int userResultsCount = 0;
int userCorrectCount = 0;
int tempUserID = 0;
int tempCorrectCount = 0;
int tempTotalCount = 0;

Vector<Integer> userIDs = new Vector<Integer>();
Vector<Integer> results = new Vector<Integer>();
Vector<Integer> sortedUserIDs = new Vector<Integer>();
Vector<Integer> sortedCorrectCount = new Vector<Integer>();
Vector<Integer> sortedTotalCount = new Vector<Integer>();
Vector<Double> allAverages = new Vector<Double>();

try {
   stmt2 = conn.createStatement();
   ResultSet rs2 = stmt2.executeQuery(selectSQL2);

   while (rs2.next()) {
      userIDs.add(rs2.getInt("userid"));
      results.add(rs2.getInt("iscorrect"));

      if (rs2.getString("userid").contains(userID)) {
         userResultsCount++;
         if (rs2.getString("iscorrect").contains("1")) {
            userCorrectCount++;
         }
      }
   }

   stmt2.close();

   for (int i = 0; i < userIDs.size(); i++) {
      tempUserID = userIDs.get(i);
      if (!(sortedUserIDs.contains(tempUserID))) {
         sortedUserIDs.add(tempUserID);
         for (int x = 0; x < userIDs.size(); x++) {
            if (userIDs.get(x) == tempUserID) {
                  tempTotalCount++;
               if (results.get(x) == 1) {
                  tempCorrectCount++;
               }

            }
         }
         sortedCorrectCount.add(tempCorrectCount);
         sortedTotalCount.add(tempTotalCount);
         tempCorrectCount = 0;
         tempTotalCount = 0;
      }
   }
```

```
        for (int i = 0; i < sortedUserIDs.size(); i++) {
        allAverages.add(((double)sortedCorrectCount.get(i) /
                        (double)sortedTotalCount.get(i)) * 100);
        }

        for (int i = 0; i < allAverages.size(); i++) {
            totalAverage = totalAverage + allAverages.get(i);
            if(allAverages.get(i)<lowestAverage) {
                        lowestAverage = allAverages.get(i);
            }
            if(allAverages.get(i)>highestAverage) {
                        highestAverage = allAverages.get(i);
            }
        }

        totalAverage = totalAverage / allAverages.size();

        userAverage = ((double) userCorrectCount / (double) userResultsCount)
              * 100;

        responseString = (deci.format(userAverage) + ":"
              + deci.format(totalAverage) + ":"
              + deci.format(highestAverage) + ":"
              + deci.format(lowestAverage) + "%");

      } catch (SQLException e) {
        responseString = "";
      }

      return responseString;
    }

    /**
     * Closes the connection to the dabase
     * @throws IOException
     */
    protected void closeConnection()
      throws IOException {
      try {
        conn.close();
        System.out.println("Connection Closed to MySQL Database");

      } catch (SQLException se) {
        System.err.println(se);
      }
    }
}
```

common.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * common is a simply class containing re-usable methods allowing code minimisation for other servlet classes
 * which are part of the student quiz administration website and typically involves extensive http allowing
 * the web pages to render dynamically
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class common extends HttpServlet {

   private static final long serialVersionUID = 1L;
   protected Connection conn = null;
   protected PrintWriter out;

   /**
    * Simply contains typical html code used by virtually all servlets for this applications
    *
    * @param request
    * @param response
    */
   protected void startHTMLCode(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");

      out.println(
            "<meta http-equiv='Content-Type' content='text/html; charset=iso-8859-1'>");
      out.println("</head>");
      out.println("<body>");
      out.println("<table width='70%' border='0'><tr>");
      out.println(
            "<td width='90%'><font size='5' face='Arial, Helvetica, sans-serif'>"
                  + "Welcome to The Online Mobile Quiz Management Site</font></td>");
      out.println("</tr>");
      out.println("</table>");
      out.println("<table width='85%' height='281' border='0'>");
      out.println("<tr><td>  </td></tr>");
      out.println("<tr align='center'>");
   }

   /**
    * Simply contains typical html code used by virtually all servlets for this applications
    *
    * @param request
    * @param response
    */
   protected void endHTMLCode(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");

      out.println("</td>");
```

```
    out.println("</tr>");
    out.println("</table>");
    out.println("</body>");
    out.println("</html>");
  }

  /**
   * Simply contains typical html code used by virtually all servlets for this applications
   *
   * @param request
   * @param response
   */
  protected void docTypeState(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");

    String docType = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 "
        + "Transitional//EN\">\n";

    out.println(docType);
  }

  /**
   * Used to clear this applications browser cookies
   *
   * @param request
   * @param response
   */
  protected void clearCookies(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");

    String[] cookieList = {
      "validation", "firstname" + "secondname"};

    Cookie[] cookies = request.getCookies();

    if (cookies != null) {
      for (Cookie cookie: cookies) {
        for (int i = 0; i < cookieList.length; i++) {
          if ((cookie.getName().equals(cookieList[i]))) {
            cookie.setMaxAge(0);
            response.addCookie(cookie);
          }
        }
      }
    }
  }

  /**
   * Authorises a user by seeing if a particular cookie and its corresponding value is present
   *
   * @param request
   * @param response
   */
  protected boolean validateUser(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");

    boolean validated = false;
```

```
    Cookie[] cookies = request.getCookies();

    if (cookies != null) {
      for (Cookie c: cookies) {
        if ((c.getName().equals("validation"))
            && (c.getValue().equals("yes"))) {
          validated = true;
          break;
        }
      }
    }
    return validated;
  }

  /**
   * Redirects a user to the login page and prevents access to unauthorised location if a user
   * is not logged in
   *
   * @param request
   * @param response
   */
  protected void loginFailureHTML(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    out.println("<html>");
    out.println("<META HTTP-EQUIV='refresh' CONTENT='3;URL=login'>");
    out.println("<head>");
    out.println(
        "You are not logged in or Login failed, you will be redirected to the login page<br>");
    out.println(
        "If you are not redirected in 3 seconds "
            + "<a href='/mob-assign1-web/servlet/login'>click here</a>");
    out.println("</head>");
  }

  /**
   * Simply contains typical html code used by virtually all servlets for this applications
   *
   * @param request
   * @param response
   */
  protected void navigationHTML(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");

    out.println("</tr>");
    out.println("<tr align='left'>");
    out.println("<td height='153' width='30%' valign='top'><blockquote>");
    out.println(
        "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='home'>"
            + "Home</a></p>");
    out.println(
        "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='login'>"
            + "Log In</font></a></p>");
    out.println(
        "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='viewquestionsps'>"
            + "View Questions</a></font></p>");
    out.println(
        "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='insertquestion'>"
            + "Insert Questions</a></font></p>");
    out.println(
```

```
            "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='viewstatisticsps'>"
                + "View Statistics</a></font></p>");
        out.println(
            "<p><font size='4' face='Arial, Helvetica, sans-serif'><a href='logout'>"
                + "Log Out</a></font></p>");
        out.println("</blockquote></td>");
    }
}
```

Home.java

```
import java.io.IOException;
import java.util.Random;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class home extends common {

   private String inputString = "";
   private String inputArray[];
   private String firstName = "";
   private String secondName = "";

   private static final long serialVersionUID = 1L;

   protected DatabaseConnector connector;
   protected ManFunctions manFun;

   /**
    * Constructor which creates instances of other associated classes
    */
   public home() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();
```

```
boolean newbie = true;
Cookie[] cookies = request.getCookies();

if (cookies != null) {
   for (Cookie c: cookies) {
      if ((c.getName().equals("repeatVisitor"))
            && (c.getValue().equals("yes"))) {
         newbie = false;
         break;
      }
   }
}

String ifVisitor = "I'm Glad You Returned";

if (newbie) { //if no validation cookies found
   Cookie returnVisitorCookie = new Cookie("repeatVisitor", "yes");

   returnVisitorCookie.setMaxAge(60 * 60 * 24 * 31); //one month
   response.addCookie(returnVisitorCookie);
   ifVisitor = "Hello Stranger";

} else {
   Random randgen = new Random();
   int randnum = randgen.nextInt(7);

   switch (randnum) { //For displaying different messages each time the page is displayed
   case 0:
      ifVisitor = "Welcome Back!";
      break;

   case 1:
      ifVisitor = "Hello Again, Nice To See You Again";
      break;

   case 2:
      ifVisitor = "Back Again Already?";
      break;

   case 4:
      ifVisitor = "What Are You After This Time?";
      break;

   case 5:
      ifVisitor = "Greatings";
      break;

   case 6:
      ifVisitor = "Well Hello Again!";
      break;
   }
}

boolean validated = false;

//validate the user by searching cookies
if (request.getParameter("userid") != null) {
   clearCookies(request, response);
} else {
   if (cookies != null) {
      for (Cookie cookie: cookies) {
         if ((cookie.getName().equals("validation")
```

```
                && (cookie.getValue().equals("yes")))) {
            validated = true;
          }
          if ((cookie.getName().equals("firstname"))) {
            firstName = cookie.getValue();
          }
          if ((cookie.getName().equals("secondname"))) {
            secondName = cookie.getValue();
          }
        }
      }
    }

    if (validated == false) { //get userID and password from login text fields
      String userID = request.getParameter("userid");
      String password = request.getParameter("password");

      inputString = connector.queryLogin(userID, password, true);

      if (!(inputString.contains("userdetails"))
          || inputString.length() < 12) {
        loginFailureHTML(request, response); //redirect to login page
      } else {
        validated = true;
        inputArray = manFun.splitString(inputString);
        firstName = inputArray[1];
        secondName = inputArray[2];

        //create new validation cookies
        Cookie validationCookie = new Cookie("validation", "yes");

        validationCookie.setMaxAge(-1);
        response.addCookie(validationCookie);
        Cookie detailsCookie1 = new Cookie("firstname", firstName);

        detailsCookie1.setMaxAge(-1);
        response.addCookie(detailsCookie1);
        Cookie detailsCookie2 = new Cookie("secondname", secondName);

        detailsCookie2.setMaxAge(-1);
        response.addCookie(detailsCookie2);
      }
    }

    //if user exists...
    if (inputString.contains("userdetails") || validated == true) {

      docTypeState(request, response);
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Home</title>");

      startHTMLCode(request, response);

      out.println(
          "<td height='60' align='center' valign='middle' colspan='5'>"
              + "<font size='5' face='Arial, Helvetica, sans-serif'>"
              + ifVisitor + "</font></td>");

      navigationHTML(request, response);

      out.println(
          "<td valign='top'><br><font size='3' face='Arial, Helvetica, sans-serif'>");
```

```
            out.println(
                  "Hello " + firstName + " " + secondName
                  + " and welcome to the online mobile quiz management site. "
                  + "<br><br>Please select a task from the menu to the left");
            out.println("</font></td>");

            endHTMLCode(request, response);
        }
    }
}
```

insertquestions.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class insertquestion extends common {

   private static final long serialVersionUID = 1L;

   /**
    * Constructor which creates instances of other associated classes
    */
   public insertquestion() {}

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response);
      } else {

         docTypeState(request, response);
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Insert Questions</title>");
         startHTMLCode(request, response);
         out.println(
               "<td height='60' align='center' colspan='5'>"
                     + "<font size='5' face='Arial, Helvetica, sans-serif'>Insert Questions</font></td>");
```

```
        navigationHTML(request, response);

        out.println(
              "<td width='20%' align='left' valign='top'><p> </p>");

        if (validateUser(request, response) == false) {
           loginFailureHTML(request, response); //redirect to login page
        } else {
           out.println(
                 "<font size='3' face='Arial, Helvetica, sans-serif'>");
           out.println("<p>Please fill in the fields below.</p>");
           out.println(
                 "<p>If you enter a question number with a corresponding week and "
                       + "subject name that already exists, it will be overwritten.</p>");
           out.println(
                 "<p>If you enter a subject name that doesn't exist, "
                       + "it will automatically be created.</p>");
           out.println(
                 "<p>You you must fill in a minimum of two answer fields.</p></font>");

           out.println(
                 "<form name='form1' method='post' action='submitinsertquestion'>");
           out.println(
                 "<font size='3' face='Arial, Helvetica, sans-serif'>");
           out.println("<p>Question No:");
           out.println("<input name='questionno' type='text' size='6'>");
           out.println(
                 " Week No: <input name='weekno' type='text' size='6'></p>");
           out.println(
                 "<p>Subject Name: <input name='subjectname' type='text' size='20'>");
           out.println(
                 "Correct Answer No: <input name='correctanswerno' type='text' size='6'></p>");
           out.println("<p>Question:</p>");
           out.println(
                 "<p><textarea name='thequestion'cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 1:</p>");
           out.println(
                 "<p><textarea name='answeroption1' cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 2:</p>");
           out.println(
                 "<p><textarea name='answeroption2' cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 3:</p>");
           out.println(
                 "<p><textarea name='answeroption3' cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 4:</p>");
           out.println(
                 "<p><textarea name='answeroption4' cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 5:</p>");
           out.println(
                 "<p><textarea name='answeroption5' cols='100' rows='2'></textarea></p>");
           out.println("<p>Answer Option 6:</p>");
           out.println(
                 "<p><textarea name='answeroption6' cols='100' rows='2'></textarea></p>");
           out.println(
                 "<p><input type='submit' name='Submit' value='Submit'></p>");
           out.println("</font></form>");

           endHTMLCode(request, response);
        }
     }
  }
}
```

Login.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class login extends common {

   private static final long serialVersionUID = 1L;

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");
      out = response.getWriter();

      docTypeState(request, response);
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Login</title>");
      startHTMLCode(request, response);
      out.println(
            "<td height='60' align='center' colspan='5'>"
                  + "<font size='5' face='Arial, Helvetica, sans-serif'>Log In</font></td>");

      navigationHTML(request, response);

      out.println("<td width='40%' align='center' valign='top'><p> </p>");
      out.println("<form method='post' action='home'>");
      out.println(
            "<p><strong><font face='Arial, Helvetica, sans-serif'>Login</font></strong></p>");
      out.println("<p><font face='Arial, Helvetica, sans-serif'> User ID:");
      out.println("<input type='text' name='userid' />");
      out.println("</font></p>");
      out.println(
            "<p><font face='Arial, Helvetica, sans-serif'>Password:   ");
```

```
        out.println("<input type='password' name='password'/>");
        out.println("</font></p>");
        out.println("<font face='Arial, Helvetica, sans-serif'>");
        out.println("<input type='submit' name='submit' value='login'/>");
        out.println("</font></form>");
        out.println(
                "<br><br><font face='Arial, Helvetica, sans-serif'>For Testing Purposes<br>"
                        + "User ID: 54321<br>Password: password</font></td>");
        out.println("<td width='30%' align='center' valign='top'> </td>");

        endHTMLCode(request, response);
    }
}
```

Logout.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class logout extends common {

   private static final long serialVersionUID = 1L;

  /**
   * Method to execute upon a Get request/response, this simply calls the doPost method
   * This is common work-around no not have to uniquely specify with the request/request is
   * a Get or a Post
   * @param request
   * @param response
   * @throws IOException
   */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");
      out = response.getWriter();
      clearCookies(request, response);

      //redirect to login page
      out.println("<html>");
      out.println("<META HTTP-EQUIV='refresh' CONTENT='3;URL=login'>");
      out.println("<head>");
      out.println("<title>Log Out</title>");
      out.println("You have been logged out<br>");
      out.println(
            "If you are not redirected in 3 seconds "
                  + "<a href='/mob-assign1-web/servlet/login'>click here</a>");
      out.println("</head>");
      out.println("</html>");
   }
}
```

Submitquestion.java

```
import java.io.IOException;
import java.util.Vector;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class submitinsertquestion extends common {

   private static final long serialVersionUID = 1L;
   protected DatabaseConnector connector;
   protected ManFunctions manFun;
   private Vector<String> answerOptions;

   /**
    * Constructor which creates instances of other associated classes
    */
   public submitinsertquestion() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
      answerOptions = new Vector<String>();

   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response);
      } else {
```

```
docTypeState(request, response);
out.println("<html>");
out.println("<head>");
out.println("<title>Insert Questions</title>");
startHTMLCode(request, response);
out.println(
      "<td height='60' align='center' colspan='5'>"
            + "<font size='5' face='Arial, Helvetica, sans-serif'>Insert Questions</td>");

navigationHTML(request, response);

out.println(
      "<td width='40%' align='center' valign='top'><p> </p>");

if (validateUser(request, response) == false) {
   loginFailureHTML(request, response); //redirect to login page
} else {

   String questionNo = request.getParameter("questionno");
   String weekNo = request.getParameter("weekno");
   String subjectName = request.getParameter("subjectname");
   String correctAnswerNo = request.getParameter("correctanswerno");
   String theQuestion = request.getParameter("thequestion");

   answerOptions.add(request.getParameter("answeroption1"));
   answerOptions.add(request.getParameter("answeroption2"));
   answerOptions.add(request.getParameter("answeroption3"));
   answerOptions.add(request.getParameter("answeroption4"));
   answerOptions.add(request.getParameter("answeroption5"));
   answerOptions.add(request.getParameter("answeroption6"));

   int answerOptionCounter = 0;

   //make sure required fields aren't empty or have incorrect datatypes in them
   if (questionNo.equalsIgnoreCase("")
         || weekNo.equalsIgnoreCase("")
         || subjectName.equalsIgnoreCase("")
         || subjectName.equalsIgnoreCase("")
         || theQuestion.equalsIgnoreCase("")
         || answerOptions.get(0).equalsIgnoreCase("")
         || answerOptions.get(1).equalsIgnoreCase("")
         || manFun.containsOnlyNumbers(questionNo) == false
         || manFun.containsOnlyNumbers(weekNo) == false
         || manFun.containsOnlyNumbers(correctAnswerNo) == false) {

      out.println(
            "<font size='3' face='Arial, Helvetica, sans-serif'>");
      out.println(
            "You have not entered all the required fields"
                  + "or have entered incorrect data into them such as letters"
                  + "or symbols in the fields that require a number, please go back and try again");

      out.println("</font>");
      out.println(
            "<form name='form1' method='post' action='insertquestion'>");
      out.println(
            "<p><input type='submit' name='Back' value='Back'></p></form>");

   } else {
      for (int i = 0; i < answerOptions.size(); i++) {
         if (!(answerOptions.get(i).equalsIgnoreCase(""))) {
            answerOptionCounter++;
```

```
            } //check which answer option text fields are empty and count the answer options
          }

          for (int i = 1; i <= answerOptionCounter; i++) {
            int isCorrectAnswer = 0;

            //if the answer is correct or not, it's stored as a boolean value on the database
            if (i == Integer.parseInt(correctAnswerNo)) {
              isCorrectAnswer = 1;
            } else {
              isCorrectAnswer = 0;

            }

            //attempt to insert or update the new / existing question
            connector.insertQuestion(questionNo, weekNo, subjectName,
                theQuestion, Integer.toString(i),
                answerOptions.get(i - 1),
                Integer.toString(isCorrectAnswer));
          }

          out.println(
              "<font size='3' face='Arial, Helvetica, sans-serif'>");
          out.println(
              "Your question has been inserted into the database succesfully");
          out.println(
              "<form name='form1' method='post' action='insertquestion'>");
          out.println(
              "<p><input type='submit' name='Back' value='Add Another Question'></p></form>");

        }
        out.println(
            "<td width='20%' align='left' valign='top'><p> </p>");

        endHTMLCode(request, response);
      }
    }
  }
}
```

Viewquestion.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class viewquestions extends common {

   private static final long serialVersionUID = 1L;

   private DatabaseConnector connector;
   private ManFunctions manFun;

   /**
    * Constructor which creates instances of other associated classes
    */
   public viewquestions() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response); //redirect to login page
      } else {
         String questionNo = request.getParameter("questionno");
         String weekNo = request.getParameter("weekno");
         String subjectName = request.getParameter("subjectname");
```

```
int questionNoInt = Integer.parseInt(questionNo);

String inputString = "";

inputString = connector.retrieveQuestion(weekNo.substring(5), //get rid of the letters from the week number
      Integer.toString(questionNoInt), subjectName);

docTypeState(request, response);
out.println("<html>");
out.println("<head>");
out.println("<title>View Questions</title>");
startHTMLCode(request, response);
out.println(
      "<td height='60' align='center' colspan='5'>"
            + "<font size='5' face='Arial, Helvetica, sans-serif'>View Questions</font></td>");

navigationHTML(request, response);

out.println(
      "<td width='40%' align='center' valign='top'><p> </p>");

if (inputString.length() > 3) { //if question exists
   String inputArray[] = manFun.splitString(inputString);

   out.println(
         "<font size = '5' face='Arial, Helvetica, sans-serif'>"
               + "" + questionNoInt + ": " + inputArray[0]
               + "<br><br></font>");

   out.println(
         "<font size = '4' face='Arial, Helvetica, sans-serif'>");
   for (int i = 1; i < inputArray.length; i++) {

      out.println(i + ": " + inputArray[i] + "<br>");
   }

   out.println("</font>");
   if (questionNoInt != 1) { //if not at the beginning of the quesions, we need a 'back' button
      out.println(
            "<form name='form1' method='post' action='viewquestions'>");
      out.println(
            "<input type='hidden' name='subjectname' value='"
                  + subjectName + "'>");
      out.println(
            "<input type='hidden' name='weekno' value='"
                  + weekNo + "'>");
      out.println(
            "<input type='hidden' name='questionno' value='"
                  + (questionNoInt - 1) + "'>");
      out.println(
            "<br><input type='submit' name='Back' value='Back'></form>");
   }

   out.println(
         "<form name='form2' method='post' action='viewquestions'>");
   out.println(
         "<input type='hidden' name='subjectname' value='" //to carry forward variables
               + subjectName + "'>");
   out.println(
         "<input type='hidden' name='weekno' value='" + weekNo //to carry forward variables
         + "'>");
   out.println(
```

```
                "<input type='hidden' name='questionno' value='" //to carry forward variables
                    + (questionNoInt + 1) + "'>");
            out.println(
                "<p><input type='submit' name='Back' value='Next'></p></form>");
        } else {
            if (questionNoInt > 1) {
                out.println(
                    "<font size = '4' face='Arial, Helvetica, sans-serif'>"
                        + "There are no more questions for this week<br><br>");
                out.println(
                    "<form name='form2' method='post' action='viewquestions'>");
                out.println(
                    "<input type='hidden' name='subjectname' value='" //to carry forward variables
                        + subjectName + "'>");
                out.println(
                    "<input type='hidden' name='weekno' value='" //to carry forward variables
                        + weekNo + "'>");
                out.println(
                    "<input type='hidden' name='questionno' value='" //to carry forward variables
                        + (questionNoInt - 1) + "'>");
                out.println(
                    "<p><input type='submit' name='Back' value='Back'></p></form>");
            }
        }

        out.println(
            "<form name='form3' method='post' action='viewquestionspw'>");
        out.println(
            "<input type='hidden' name='subjectname' value='"
                + subjectName + "'>");
        out.println(
            "<p><input type='submit' name='Back' value='Back To Week Selection'></p></form>");

        out.println(
            "<td width='30%' align='center' valign='top'> </td>");

        endHTMLCode(request, response);
    }
  }
}
```

Viewquestions.ps

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class viewquestionsps extends common {

   private static final long serialVersionUID = 1L;

   private DatabaseConnector connector;
   private ManFunctions manFun;

   /**
    * Constructor which creates instances of other associated classes
    */
   public viewquestionsps() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response); //redirect to login page
      } else {

         String inputString = connector.querySubjects();
```

```
        docTypeState(request, response);
        out.println("<html>");
        out.println("<head>");
        out.println("<title>View Questions</title>");
        startHTMLCode(request, response);
        out.println(
              "<td height='60' align='center' colspan='5'>"
                    + "<font size='5' face='Arial, Helvetica, sans-serif'>View Questions</font></td>");

        navigationHTML(request, response);

        out.println(
              "<td width='40%' align='center' valign='top'><p> </p>");

        if (inputString.length() > 2) {
           String[] inputArray = manFun.splitString(inputString);

           out.println(
                 "<p><font face='Arial, Helvetica, sans-serif'>"
                       + "Please Select a Subject</font></p>");
           out.println(
                 "<form name='form1' method='post' action='viewquestionspw'>");
           out.println("<select name='subjectname'>");

           for (int i = 0; i < inputArray.length; i++) {
              out.println("<option>" + inputArray[i] + "</option>");
           }

           out.println("</select><br>");
           out.println(
                 "<p><input type='submit' name='Next' value='Next'></p></form>");

        } else {
           out.println(
                 "<font face='Arial, Helvetica, sans-serif'>There are no subjects available");

        }

        out.println(
              "<td width='30%' align='center' valign='top'> </td>");

        endHTMLCode(request, response);
     }
  }
}
```

Viewquestionspw

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class viewquestionspw extends common {

   private static final long serialVersionUID = 1L;

   private DatabaseConnector connector;
   private ManFunctions manFun;

   /**
    * Constructor which creates instances of other associated classes
    */
   public viewquestionspw() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response); //redirect to login page
      } else {
         String inputString = "";
         String subjectName = request.getParameter("subjectname");
```

```
        inputString = connector.queryWeeks(subjectName, 0);

        docTypeState(request, response);
        out.println("<html>");
        out.println("<head>");
        out.println("<title>View Questions</title>");
        startHTMLCode(request, response);
        out.println(
            "<td height='60' align='center' colspan='5'>"
                + "<font size='5' face='Arial, Helvetica, sans-serif'>View Questions</font></td>");

        navigationHTML(request, response);

        out.println(
            "<td width='40%' align='center' valign='top'><p> </p>");

        if (inputString.length() > 2) {
          String[] inputArray = manFun.splitString(inputString);

          out.println(
              "<p><font face='Arial, Helvetica, sans-serif'>"
                  + "Please Select an Available Week for subject: "
                  + subjectName + "</font></p>");
          out.println(
              "<form name='form1' method='post' action='viewquestions'>");
          out.println("<select name='weekno'>");

          for (int i = 0; i < inputArray.length; i++) {
            out.println("<option>" + inputArray[i] + "</option>");
          }

          out.println("</select><br>");
          out.println(
              "<input type='hidden' name='subjectname' value='"
                  + subjectName + "'>");
          out.println("<input type='hidden' name='questionno' value='1'>"); //to carry forward variables

          out.println(
              "<p><input type='submit' name='Next' value='Next'></p></form>");

          out.println(
              "<form name='form1' method='post' action='viewquestionsps'>");
          out.println(
              "<p><input type='submit' name='Back' value='Back'></p></form>");

        } else {
          out.println(
              "<font face='Arial, Helvetica, sans-serif'>"
                  + "There are no weeks for this subject</font>");
          out.println(
              "<form name='form1' method='post' action='viewquestionsps'>");
          out.println(
              "<p><input type='submit' name='Back' value='Back'></p></form>");

        }

        out.println(
            "<td width='30%' align='center' valign='top'> </td>");

        endHTMLCode(request, response);
      }
    }
}
```

Viewstatistics.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class viewstatistics extends common {

	private static final long serialVersionUID = 1L;
	protected DatabaseConnector connector;
	protected ManFunctions manFun;

	/**
	 * Constructor which creates instances of other associated classes
	 */

	public viewstatistics() {
		connector = new DatabaseConnector();
		manFun = new ManFunctions();
	}

	/**
	 * Method to execute upon a Get request/response, this simply calls the doPost method
	 * This is common work-around no not have to uniquely specify with the request/request is
	 * a Get or a Post
	 * @param request
	 * @param response
	 * @throws IOException
	 */
	public void doGet(HttpServletRequest request,
			HttpServletResponse response)
	throws ServletException, IOException {
		doPost(request, response);
	}

	/**
	 * Method to execute upon a Post request/response
	 * @param request
	 * @param response
	 * @throws IOException
	 */
	public void doPost(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException {

		response.setContentType("text/html");
		out = response.getWriter();

		//authenticate user
		if (validateUser(request, response) == false) {
			loginFailureHTML(request, response);
		} else {

			docTypeState(request, response);
			out.println("<html>");
```

```
                        out.println("<head>");
                        out.println("<title>View Statistics</title>");
                        startHTMLCode(request, response);
                        out.println(
                                        "<td height='60' align='center' colspan='5'>"
                                        + "<font size='5' face='Arial, Helvetica, sans-serif'>View
Statistics</font></td>");

                        navigationHTML(request, response);

                        out.println(
                                        "<td width='40%' align='center' valign='top'><p> </p>");

                        if (validateUser(request, response) == false) {
                                loginFailureHTML(request, response); //redirect to login page
                        } else {
                                out.println(
                                                "<font size='4' face='Arial, Helvetica, sans-serif'>");
                                out.println("<br>Statistics are as follows:<br><br>");

                                String subjectName = request.getParameter("subjectname");
                                String inputString = connector.retrieveStatistics(subjectName,
                                "0");

                                String inputArray[] = manFun.splitString(inputString);

                                if(manFun.containsOnlyNumbers(inputArray[1])==true) {

                                        out.println("Total Average: " + inputArray[1] + "%<br>");
                                        out.println("Highest Average: " + inputArray[2] + "%<br>");
                                        out.println("Lowest Average: " + inputArray[3] + "<br>");

                                } else {
                                        out.println("There are no statistic available for this subject");
                                }

                                out.println(
                                "<form name='form1' method='post' action='viewstatisticsps'>");
                                out.println(
                                "<p><input type='submit' name='Back' value='Back'></p></form>");

                                out.println(
                                "<td width='30%' align='center' valign='top'> </td>");

                                endHTMLCode(request, response);
                        }
                }
        }
}
```

Viewstatisticsps.java

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * A servlet which acts as a dynamically created html web page for the quiz administration panel
 * and includes user authentication via created browser cookie values. It supposed both Get and Post
 * requests
 *
 * @author Jamie Brindle (06352322) - Msc - Manchester Metropolitan University
 */
public class viewstatisticsps extends common {

   private static final long serialVersionUID = 1L;

   private DatabaseConnector connector;
   private ManFunctions manFun;

   /**
    * Constructor which creates instances of other associated classes
    */
   public viewstatisticsps() {
      connector = new DatabaseConnector();
      manFun = new ManFunctions();
   }

   /**
    * Method to execute upon a Get request/response, this simply calls the doPost method
    * This is common work-around no not have to uniquely specify with the request/request is
    * a Get or a Post
    * @param request
    * @param response
    * @throws IOException
    */
   public void doGet(HttpServletRequest request,
         HttpServletResponse response)
      throws ServletException, IOException {
      doPost(request, response);
   }

   /**
    * Method to execute upon a Post request/response
    * @param request
    * @param response
    * @throws IOException
    */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      out = response.getWriter();

      //authenticate user
      if (validateUser(request, response) == false) {
         loginFailureHTML(request, response); //redirect to login page
      } else {

         String inputString = connector.querySubjects();
```

```
        docTypeState(request, response);
        out.println("<html>");
        out.println("<head>");
        out.println("<title>View Statistics</title>");
        startHTMLCode(request, response);
        out.println(
              "<td height='60' align='center' colspan='5'>"
                    + "<font size='5' face='Arial, Helvetica, sans-serif'>View Statistics</font></td>");

        navigationHTML(request, response);

        out.println(
              "<td width='40%' align='center' valign='top'><p> </p>");

        if (inputString.length() > 2) {
           String[] inputArray = manFun.splitString(inputString);

           out.println(
                 "<p><font face='Arial, Helvetica, sans-serif'>"
                       + "Please Select a Subject</font></p>");
           out.println(
                 "<form name='form1' method='post' action='viewstatistics'>");
           out.println("<select name='subjectname'>");

           for (int i = 0; i < inputArray.length; i++) {
              out.println("<option>" + inputArray[i] + "</option>");
           }

           out.println("</select><br>");
           out.println(
                 "<p><input type='submit' name='Next' value='Next'></p></form>");
        } else {
           out.println(
                 "<font face='Arial, Helvetica, sans-serif'>There are no subjects available");
        }

        out.println(
              "<td width='30%' align='center' valign='top'> </td>");

        endHTMLCode(request, response);
     }
  }
}
```

Testing

Thorough testing was carried out at every part of the development of this application in which a lot of error checking was implemented into the code, such as code that tests the validity of datatypes in fields, connection problems, empty datasets and null value testing. There was also a final application test whereby screenshots were taken. This final test revealed no errors. The screenshot are attached to the end of this document.

Evaluation

In the completion of this assignment it has proven interesting, rewarding and educational, the only problem I seemed to have is a time constraint, whereby I had to make sacrifices in development. Had I more time I would have included more functionality. For example in the mobile quiz administration website, I have a single page that allows the addition of questions but do not allow much usability, data integrity or functionality, the tutor can only add new questions and overwrite others and provides little in terms of database management. For example, if a tutor enters the details of a new question, they can not view a question for amendment, it's done automatically without their knowledge if the same question number and week number already exists on the database. Tutors cannot delete questions, tutors cannot see what week or question number they're up to, there is very little in terms of error checking or why a particular error occurs, only that it has occurred and that the database wasn't updated, and if a tutor enters a subject name that isn't found on the database, even though a new subject name is automatically added to the subject table in the database, given tutors the ability to add new subjects, it still aids poor data integrity. For example if a tutor makes a spelling mistake while inputting the subject name it would result in a poor database, it would be much better to have a separate page whereby tutors can enter a new subject, then when they wish to add a question they could select the subject from a drop down list.

I would also like to have improved the look of both the MIDlet and the website as it appears too simple and bland. The sacrifices however that I made are mearly aesthetic and I chose to focus a lot more on the quality and functionality of the code, the application does serve its purposes.

Future Development

The idea behind this application is sound and i believe it could actually work if put into use, however for the moment it requires that the user must connect to the internet on their mobile device in order to participate in the quiz, which would result in unnececssery expenses as it costs to go online with a mobile device, perhaps another, inexpensive approach may be beneficial. An example might be to establish a conenction via Bluetooth or some other means.

Database Structure and Testing Screenshots

Follows

Database Tables:

questions				
Field	**Type**	**Null?**	**Key**	**Extra**
questionid	int(8)	no	primary	auto-increment
questionno	int(3)	no		
weekno	int(3)	no		
answerno	int(2)	no		
answers	varchar(100)	no		
correctanswer	int(2)	no		
subjectid	varchar(8)	no		
question	varchar(100)	no		

subjects				
Field	**Type**	**Null?**	**Key**	**Extra**
subjectid	int(4)	no	primary	auto-increment
subjectname	varchar(20)	no		

tutors				
Field	**Type**	**Null?**	**Key**	**Extra**
userid	int(6)	no	primary	auto-increment
password	varrchar(20)	no		
firstname	varchar(20)	no		
secondname	varchar(20)	no		

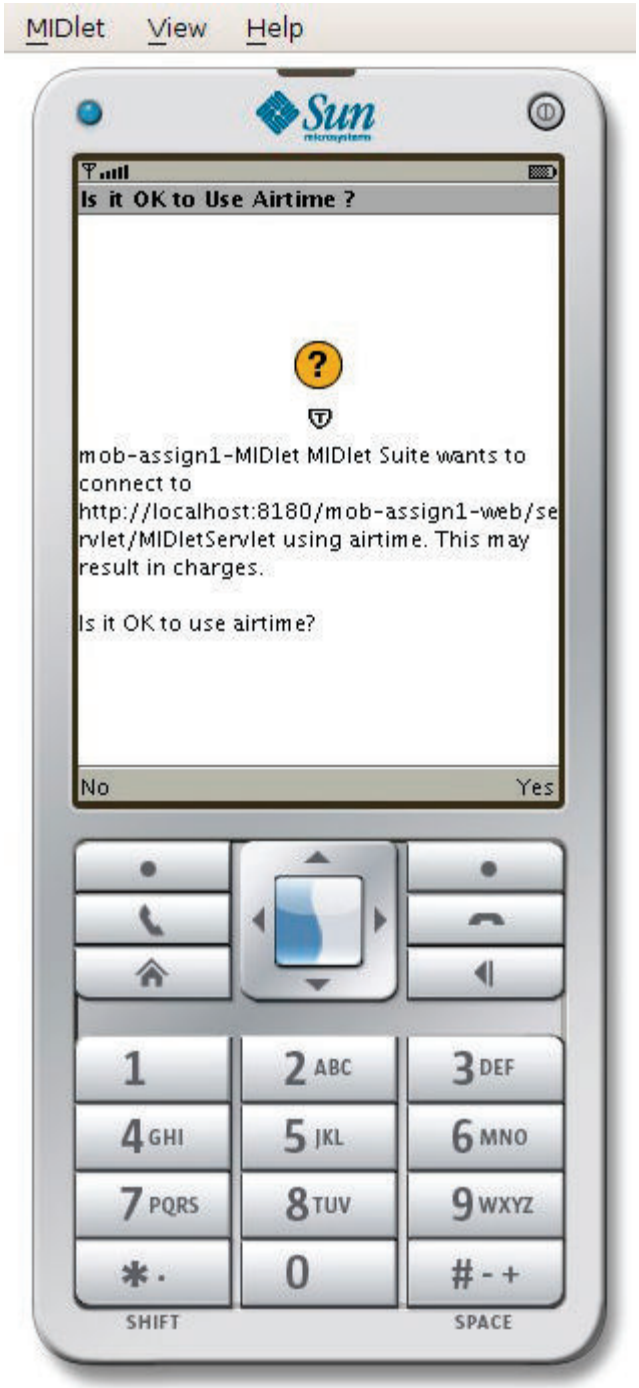
users				
Field	**Type**	**Null?**	**Key**	**Extra**
userid	int(6)	no	primary	auto-increment
password	varrchar(20)	no		
firstname	varchar(20)	no		
secondname	varchar(20)	no		

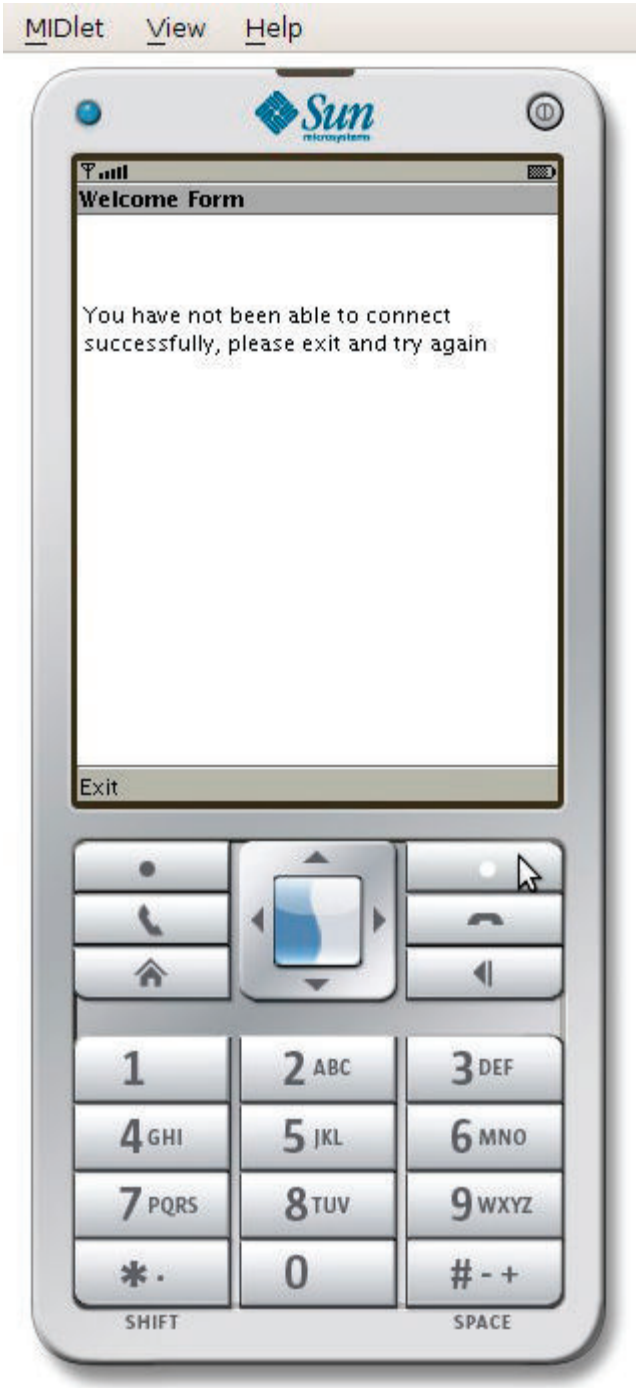
results				
Field	**Type**	**Null?**	**Key**	**Extra**
resultidid	int(8)	no	primary	auto-increment
subjecrid	int(4)	no		
userid	int(8)	no		
weekno	int(3)	no		
questionno	int(3)	no		
answerno	int(2)	no		
iscorrect	int(2)	no		

Testing:

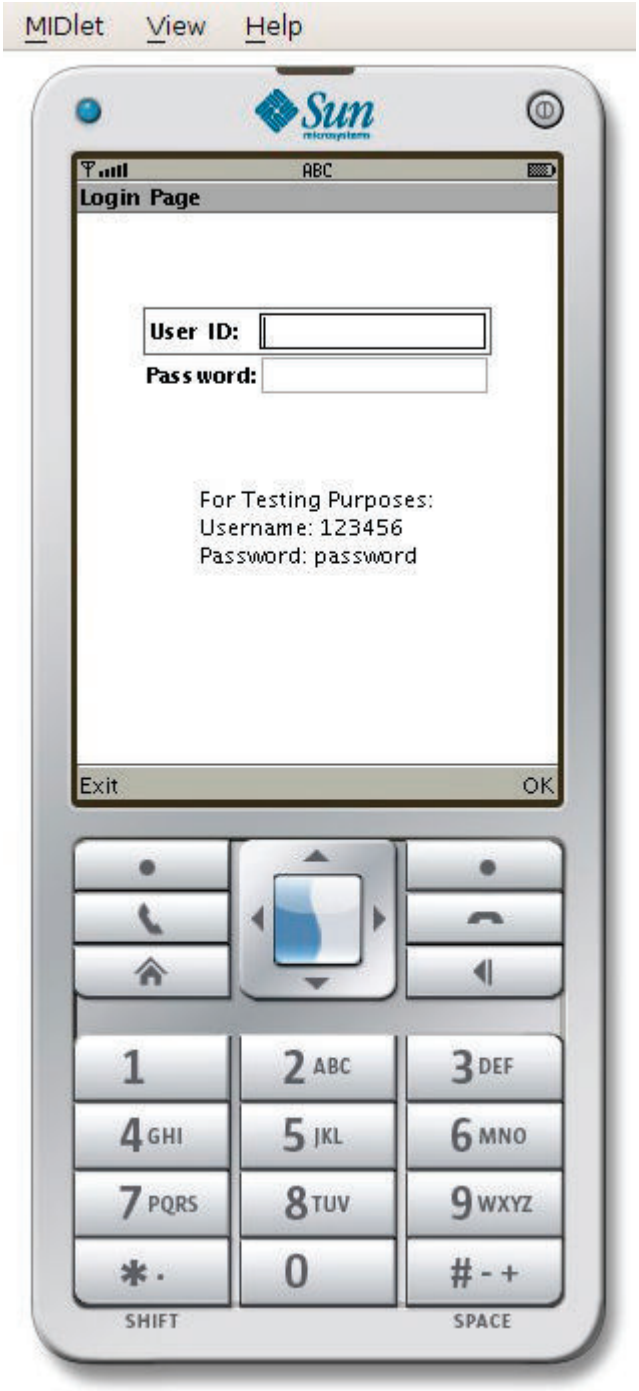
Midlets:

1. Starting the application	**2. Making the connection with no server**
Pre: WTK emulator should open	Pre: Connection failure warning should appear
Post: success:	Post: success

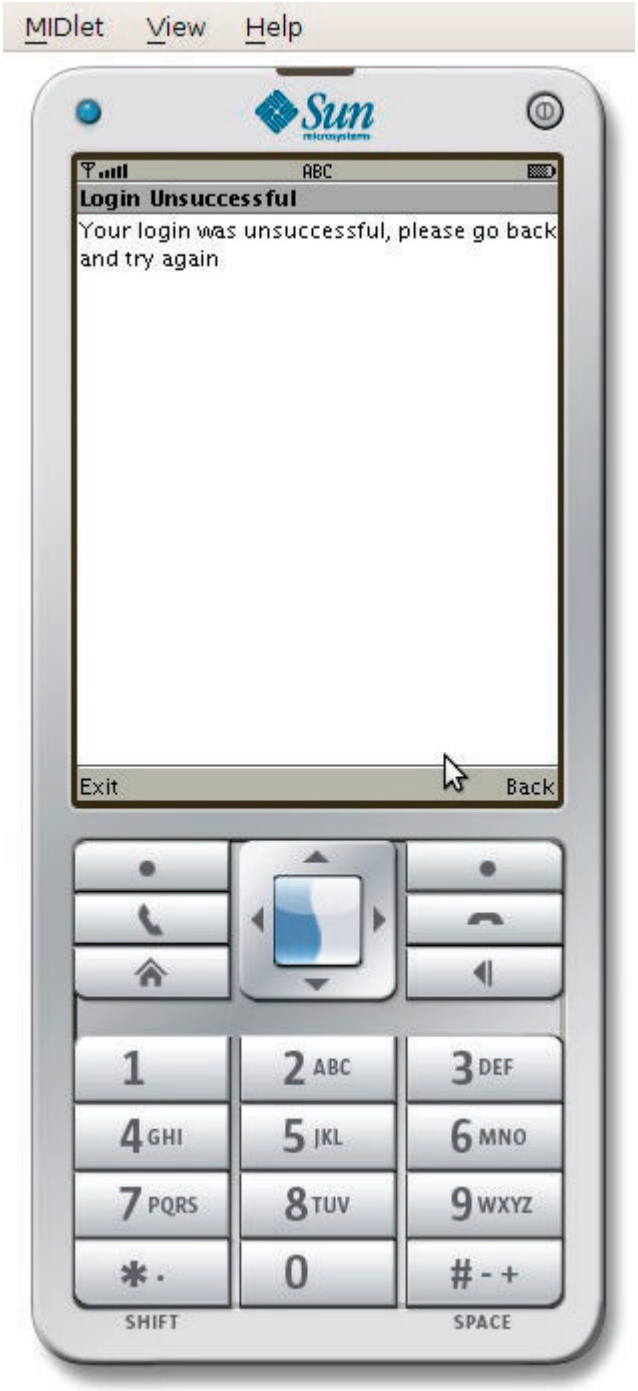


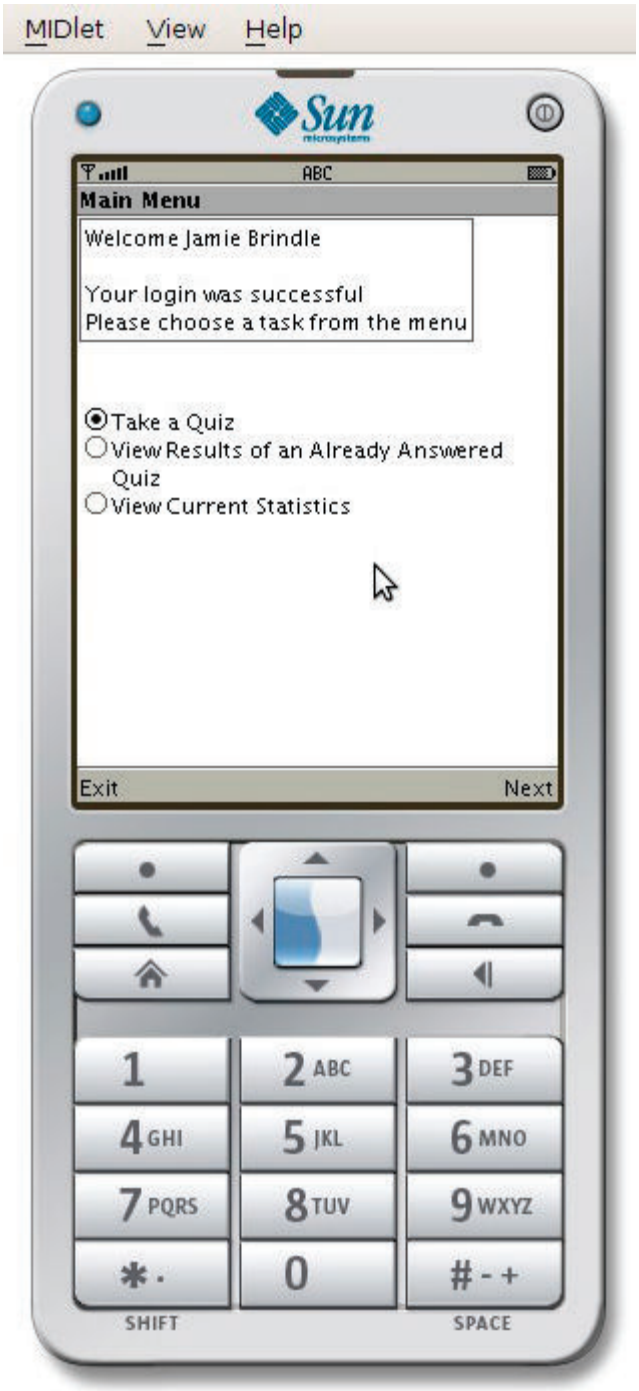


3. Making the connection with a server	4. Pressing 'next' to login
Pre: Connection success message should appear	Pre: Login form should appear
Post: success	Post: success

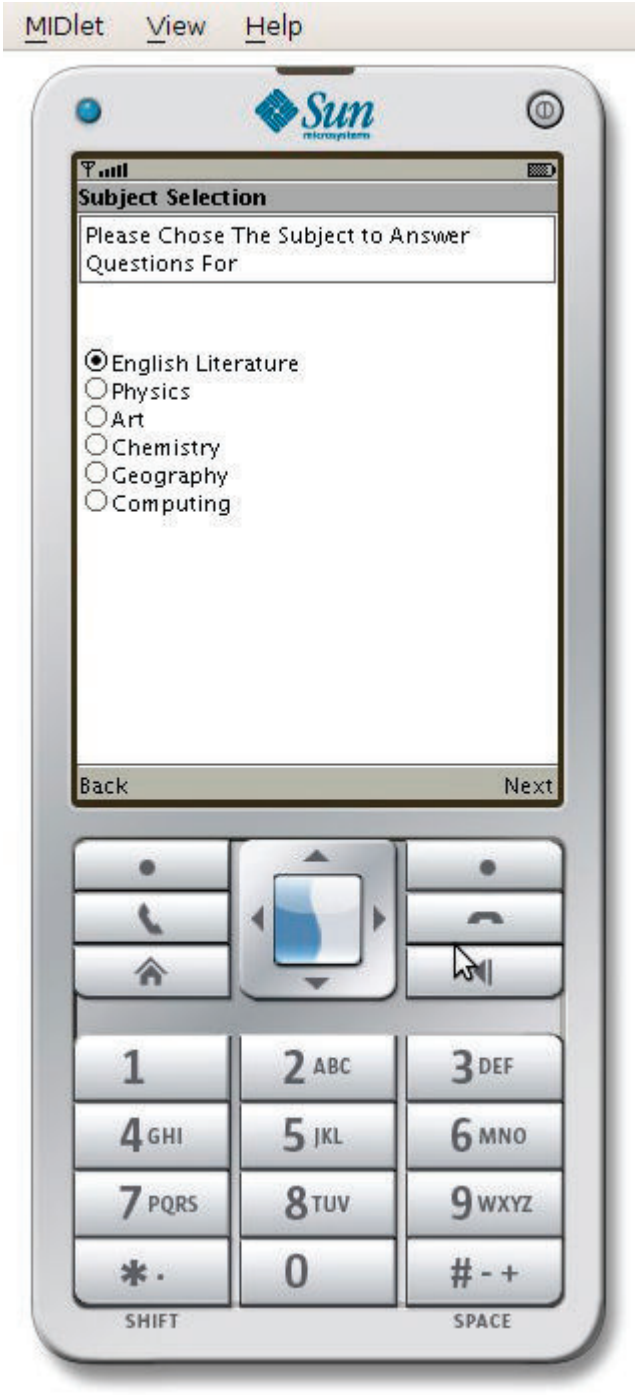


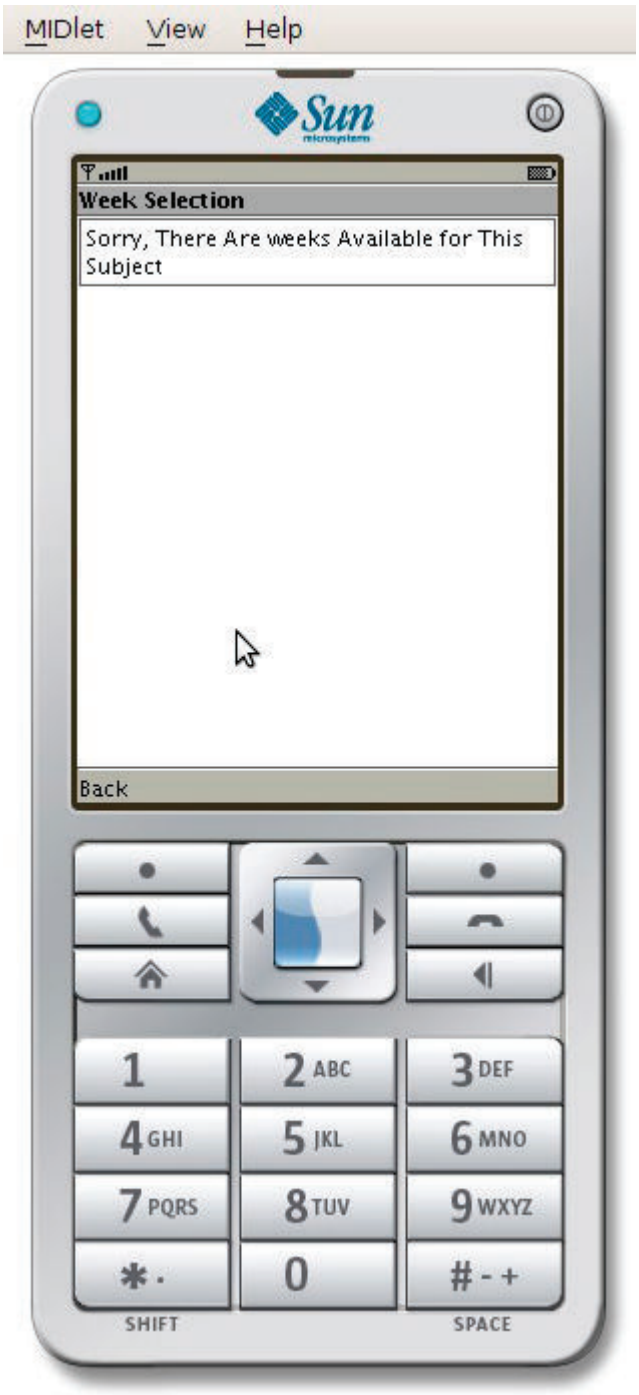
5. Incorrect login details entered	6. Correct login details entered
Pre: login incorrect message should appear	Pre: welcome / main menu form should appear
Post: success	Post: success



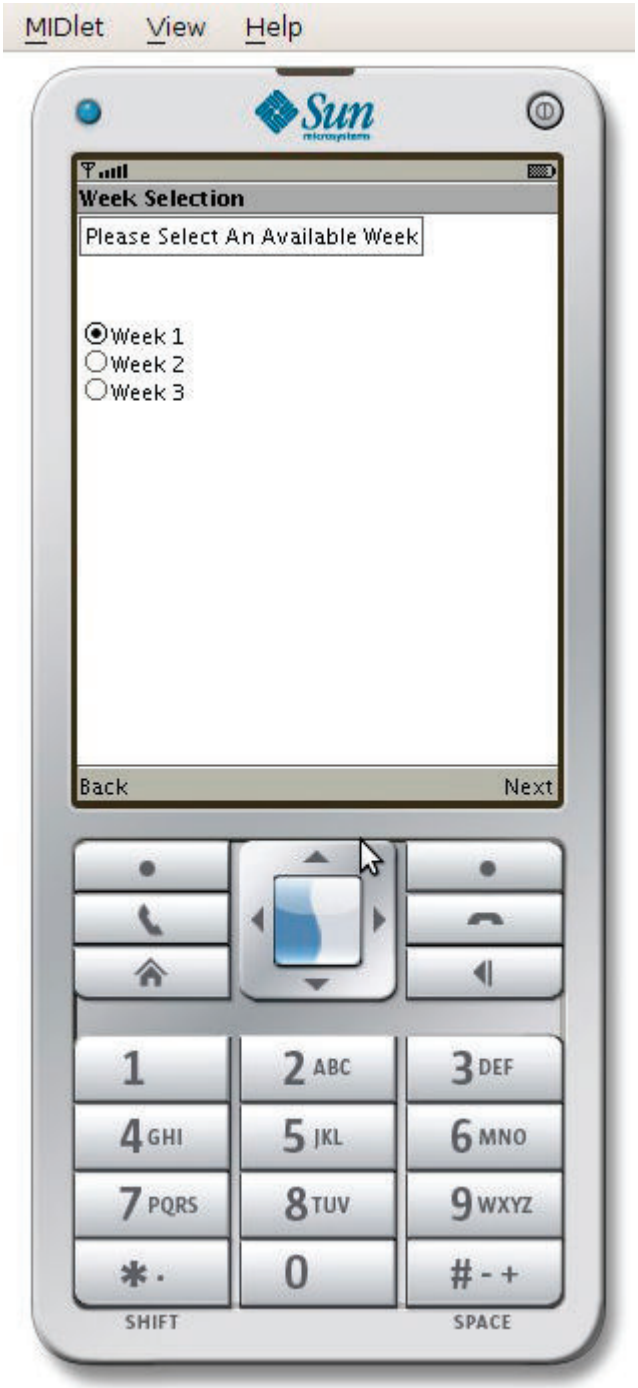


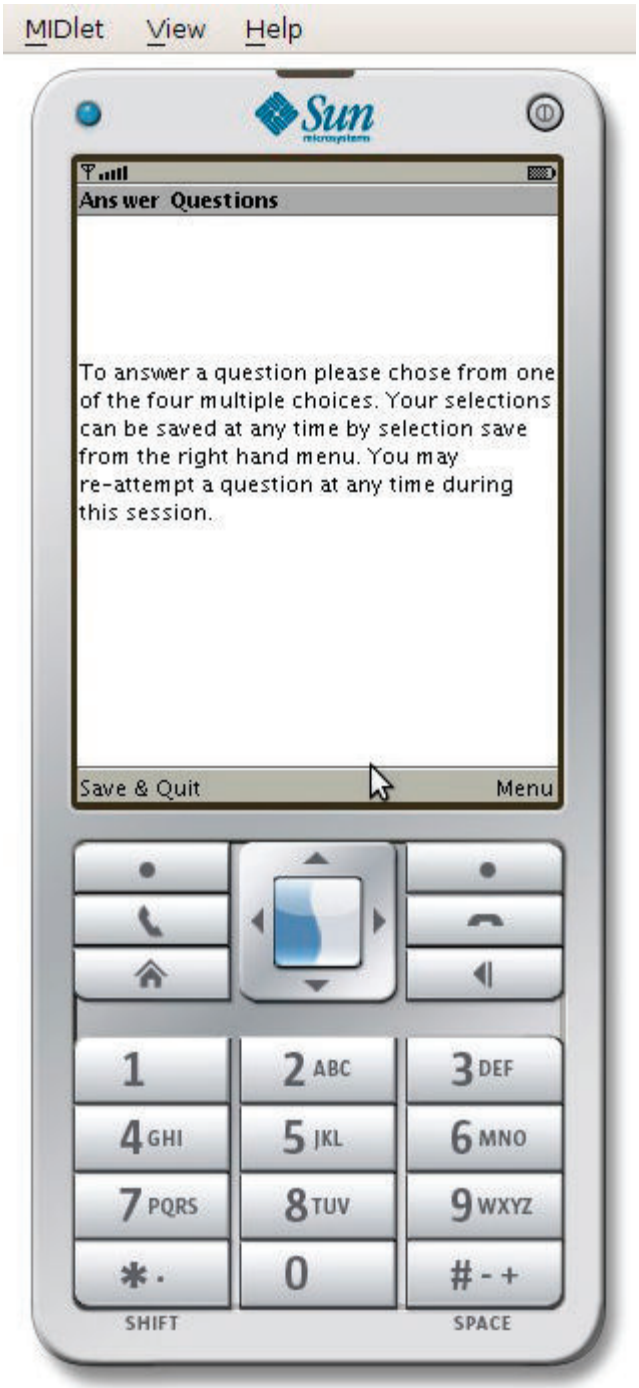
7. 'Taking the quiz' selected	8. subject selected containing no weeks
Pre: should prompt for a subject	Pre: no weeks available message should appear
Post: success	Post: success



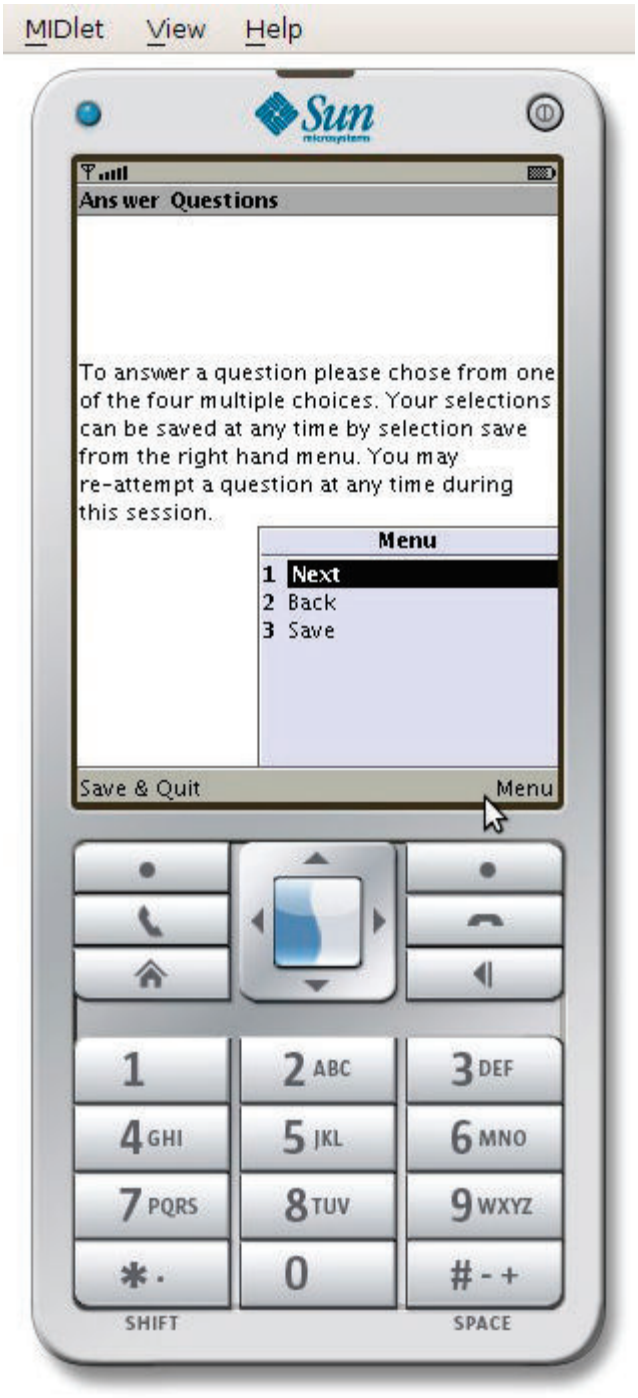


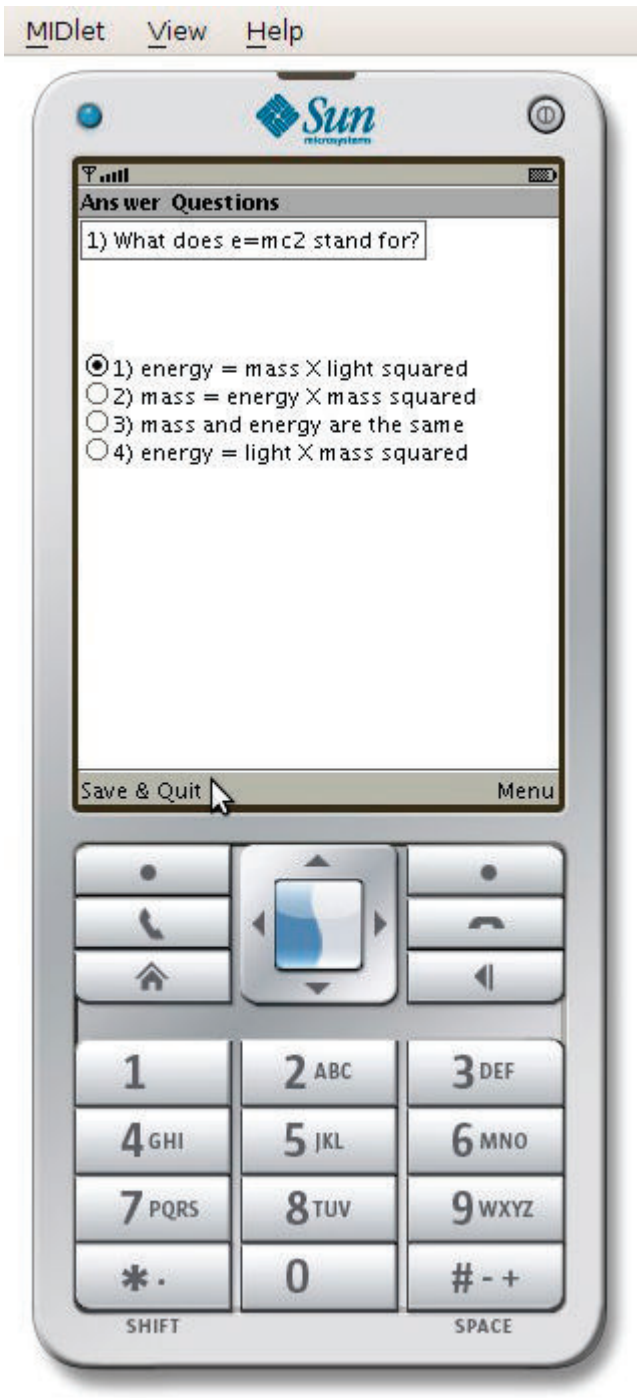
9. Selecting a subject with available weeks	10. week 1 selected
Pre: available week list should appear	Pre: quiz brief should appear
Post: success	Post: success



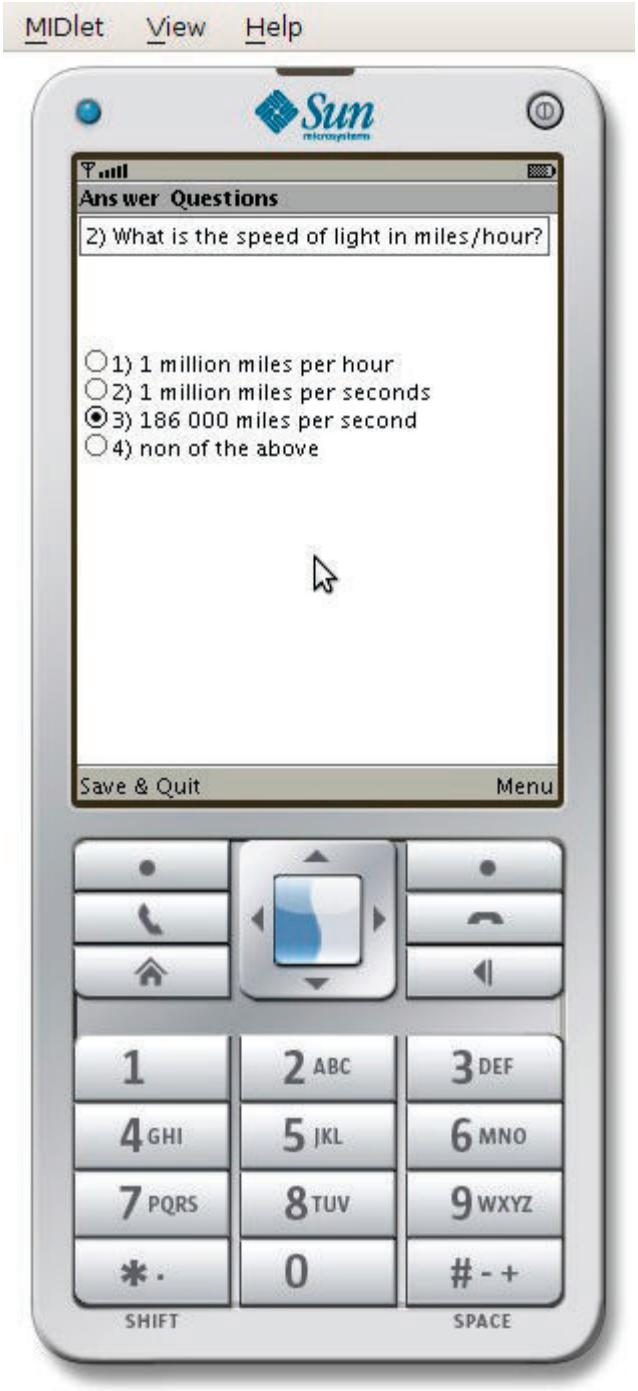


11. Menu selected	12. next selected
Pre: navigate menu should appear	Pre: a question and available answers should appear
Post: success:	Post: success

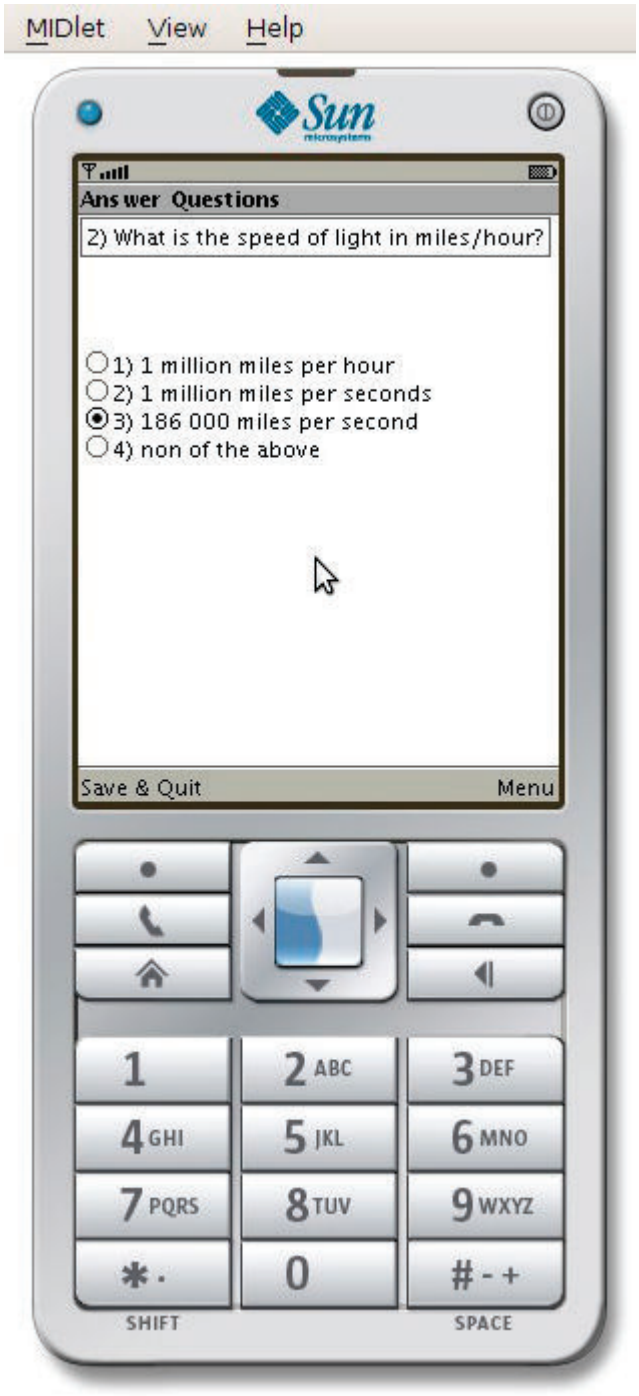


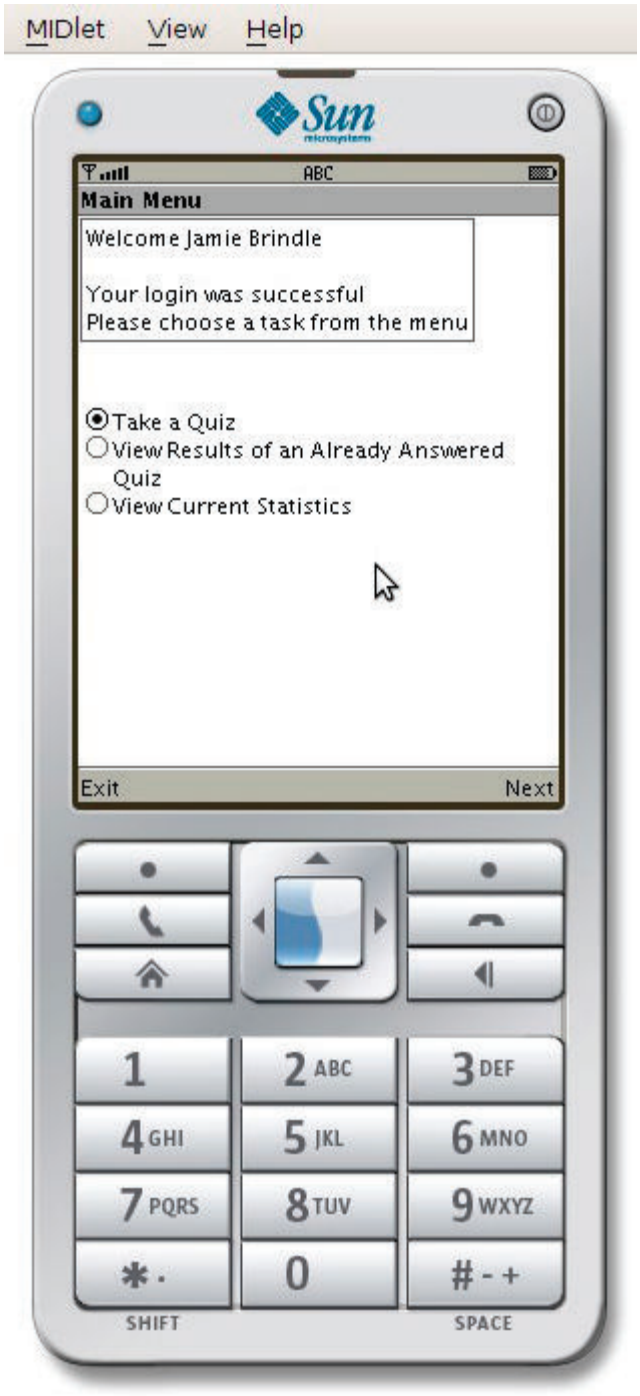


13. 1 selected and menu next selected	14. 3 selected and menu next selected
Pre: next question should appear	Pre: this is the end of the questions for this week, end of questions message should appear
Post: success:	Post: success

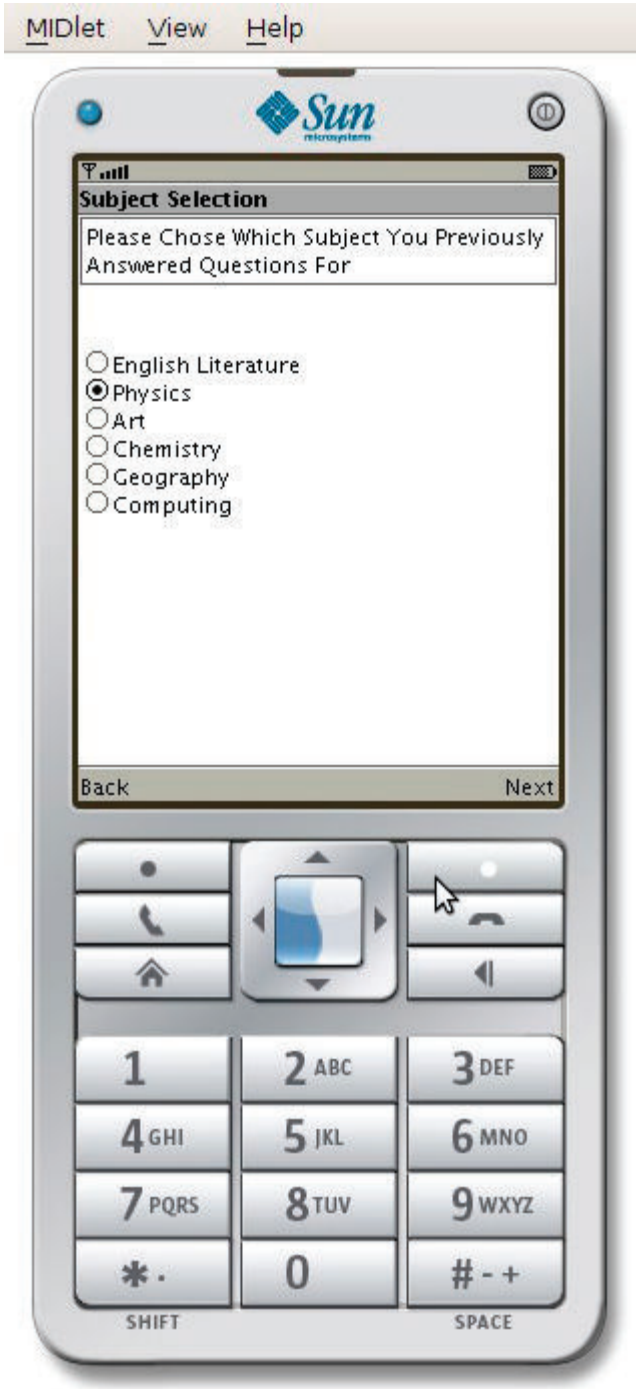


15. Menu-back selected	16. Save and quit selected
Pre: previous question should appear and still have answer 3 selected (the chosen answer)	Pre: should return to the main menu / welcome form and new values should be stored in the database
Post: success	Post: success

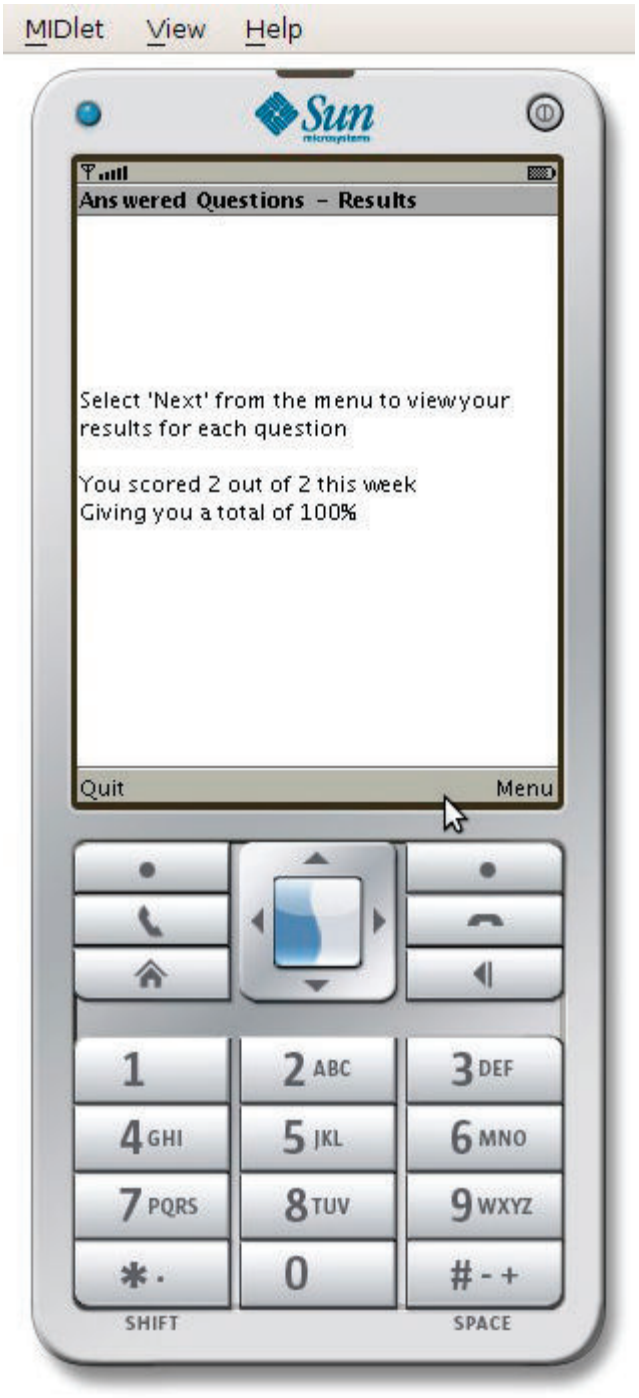


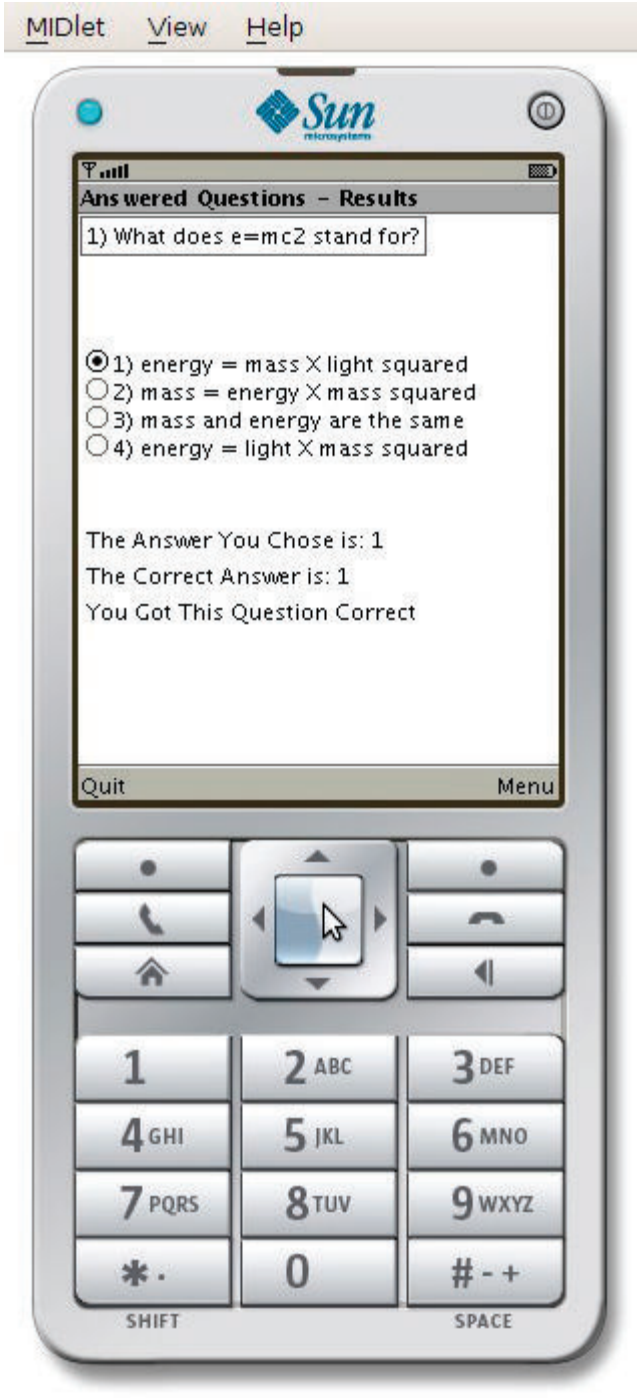


17. view results of previous quiz (2nd option) selected	**18. physics selected (same as the quiz we have just done)**
Pre: should prompt for a subject to be selected	Pre: a list of weeks should appear that we have completed (I have already completed the other questions for other weeks previously before testing)
Post: success:	Post: success

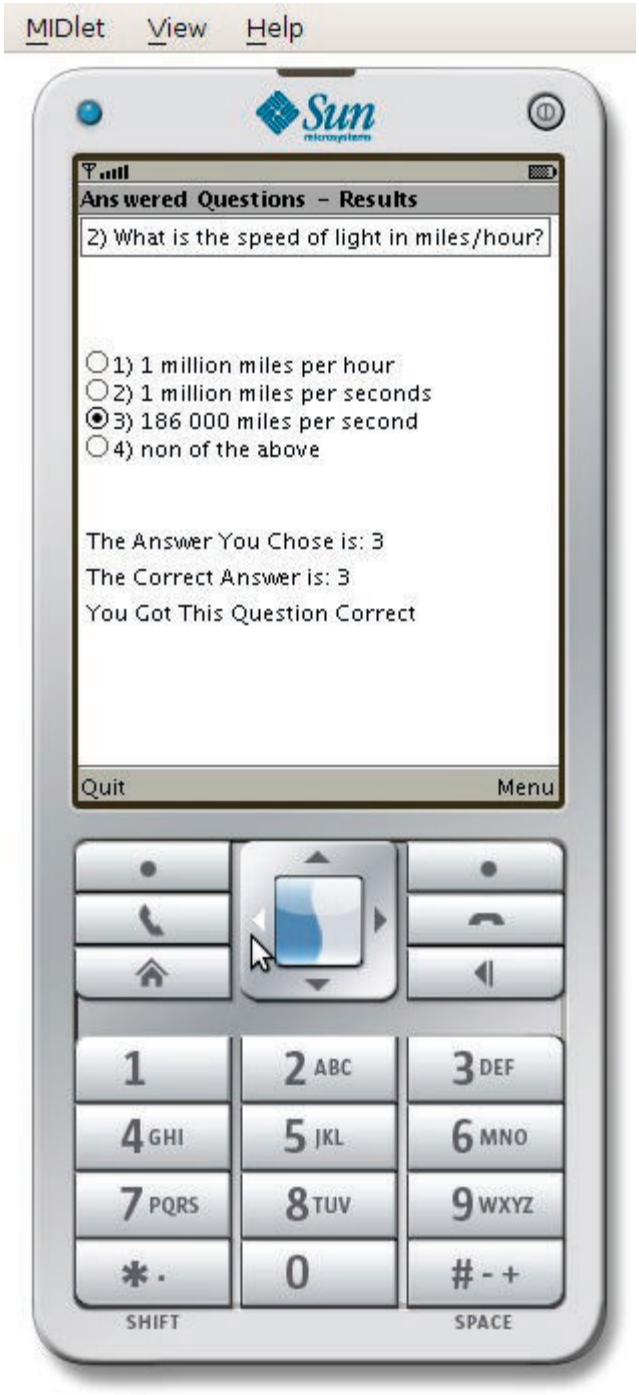


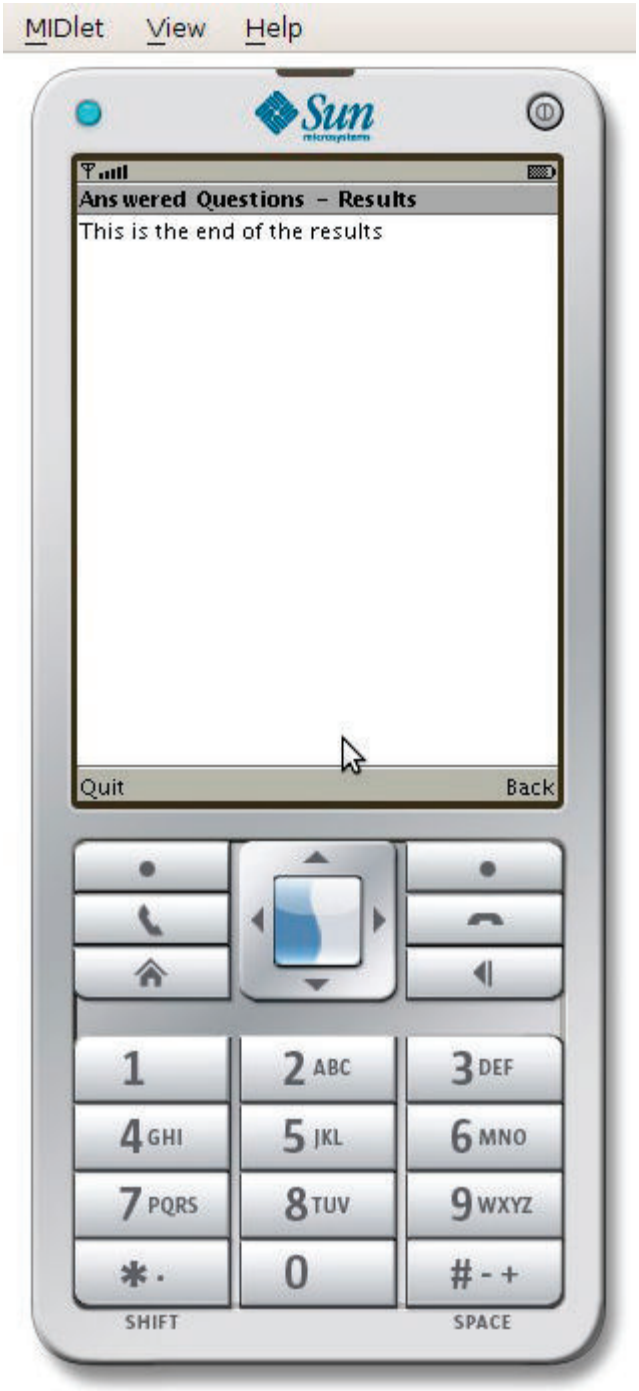
19. week 1 selected	20. Menu and next selected
Pre: a summer of that weeks quiz should appear	Pre: should show the question that was answered along with the users choice pre selected
Post: success:	Post: success



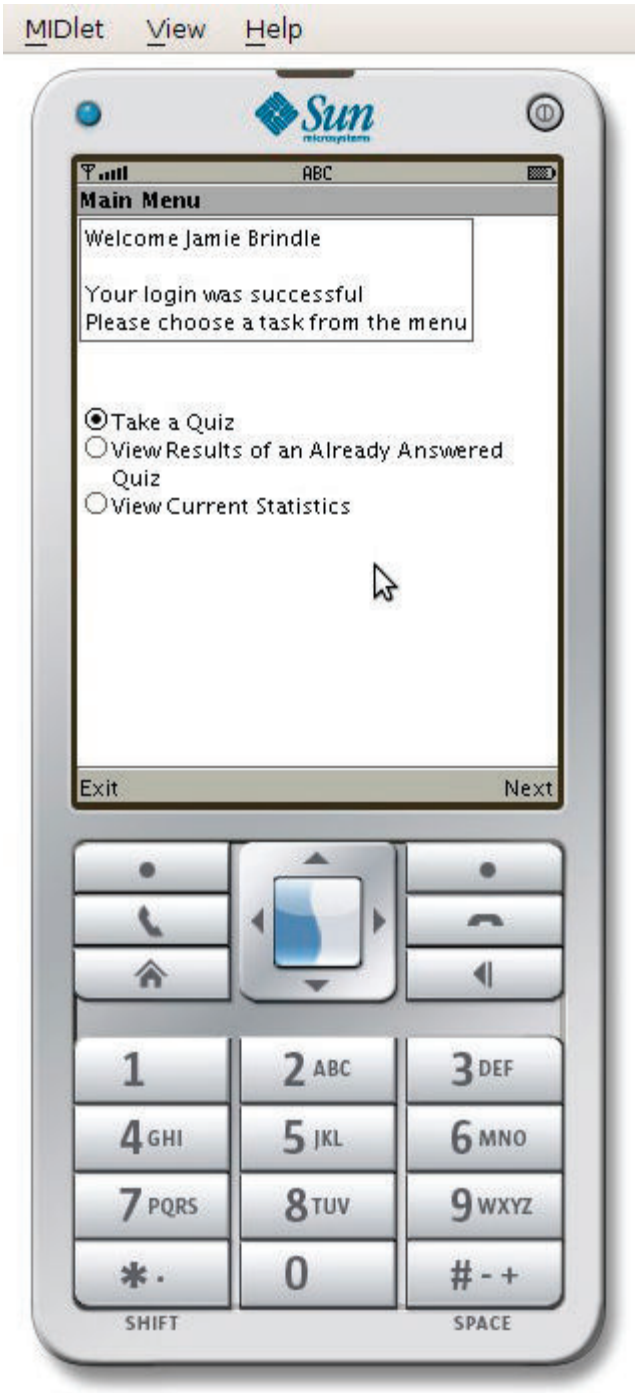


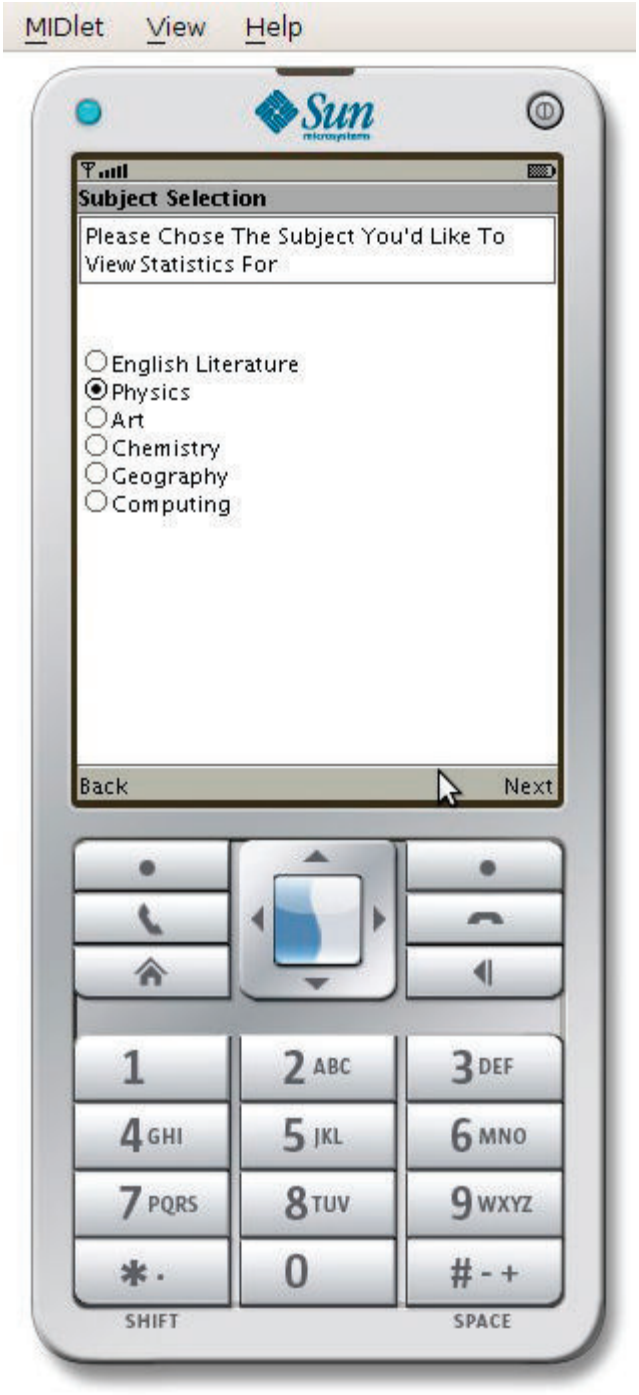
21. Menu and next selected	22. menu and next selected
Pre: should show the next answered question, again selecting the users choice	Pre: message should appear stating that no more questions were answered. Note, not all questions have to be answered in the quiz, the results will only show what questions the user has answered
Post: success:	Post: success



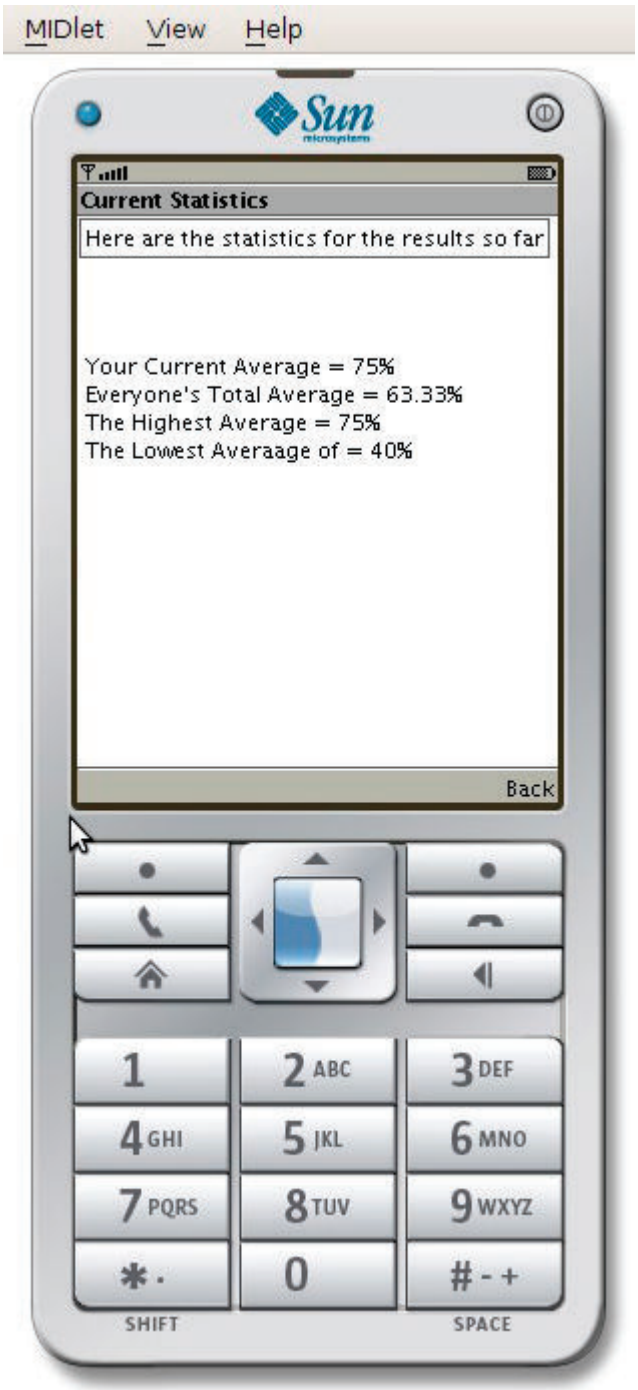


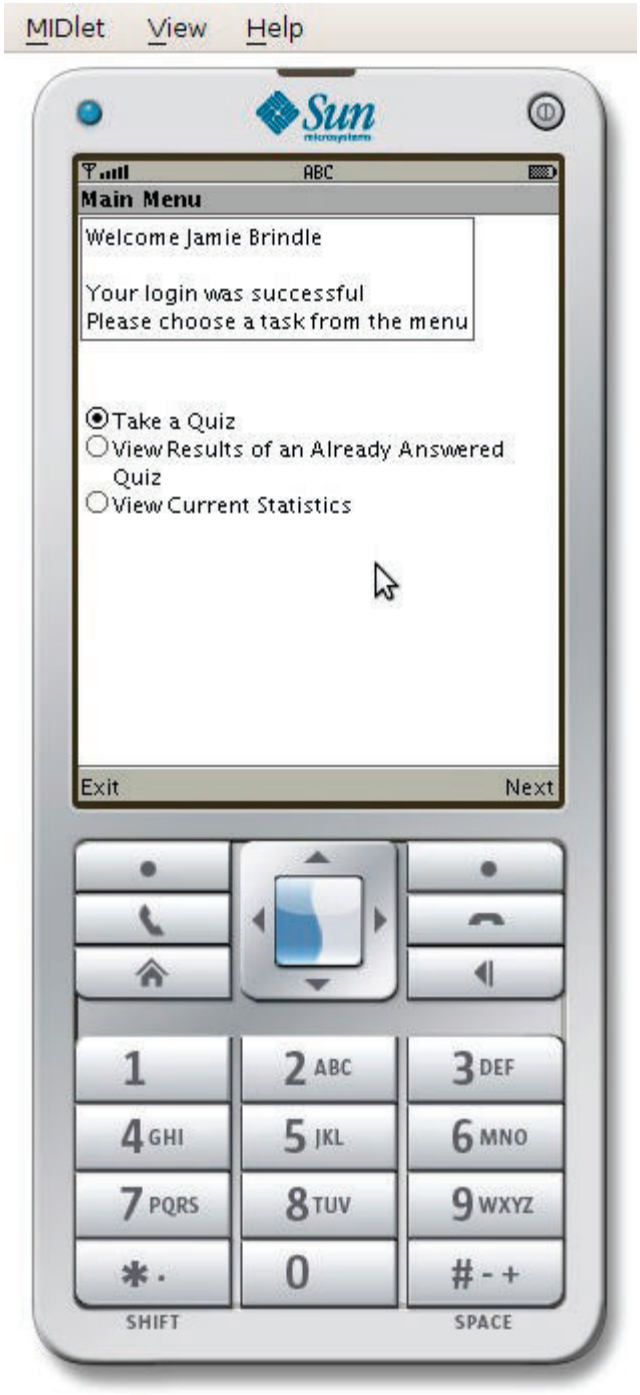
23. Quit selected	24. view statistics selected
Pre: The mean menu / welcome form should appear	Pre: subject list prompt should appear
Post: success	Post: success





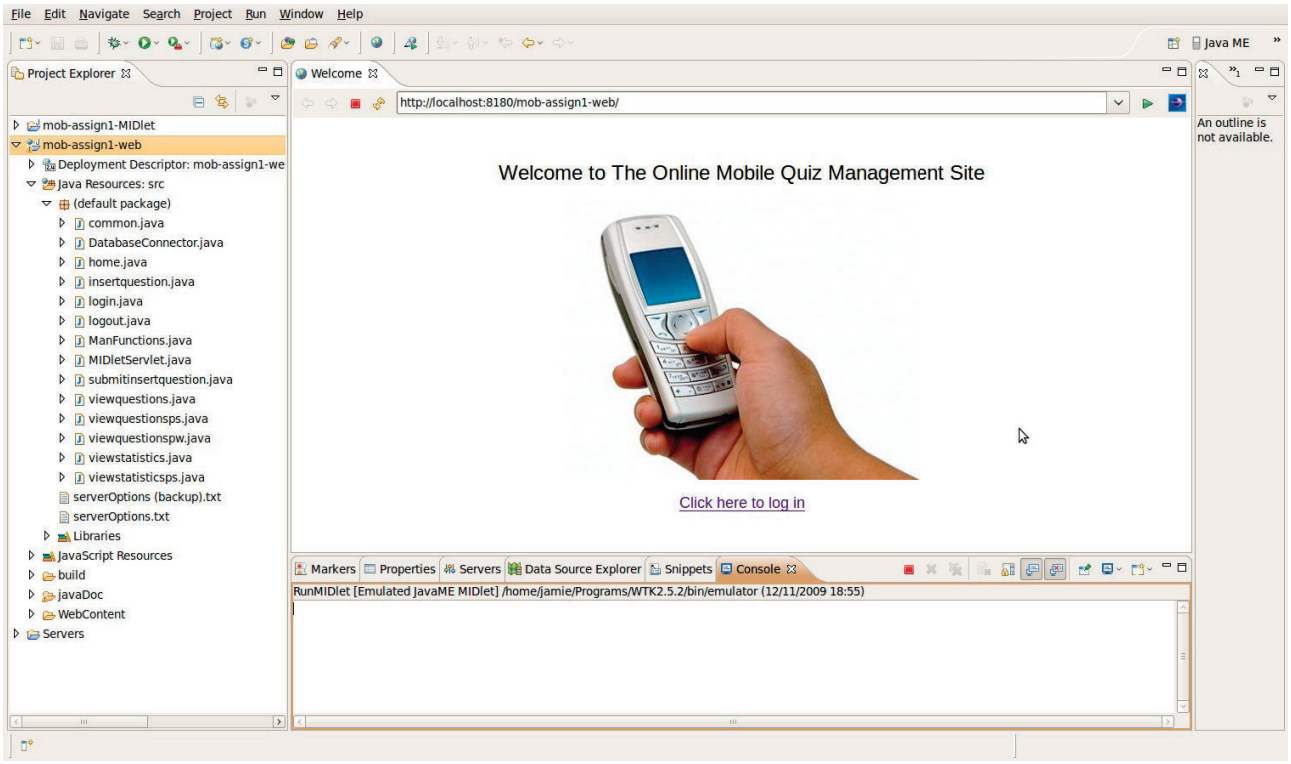
25. Physics selected	26. quit selected
Pre: statistics page should appear	Pre: mean menu / welcome form should appear
Post: success	Post: success



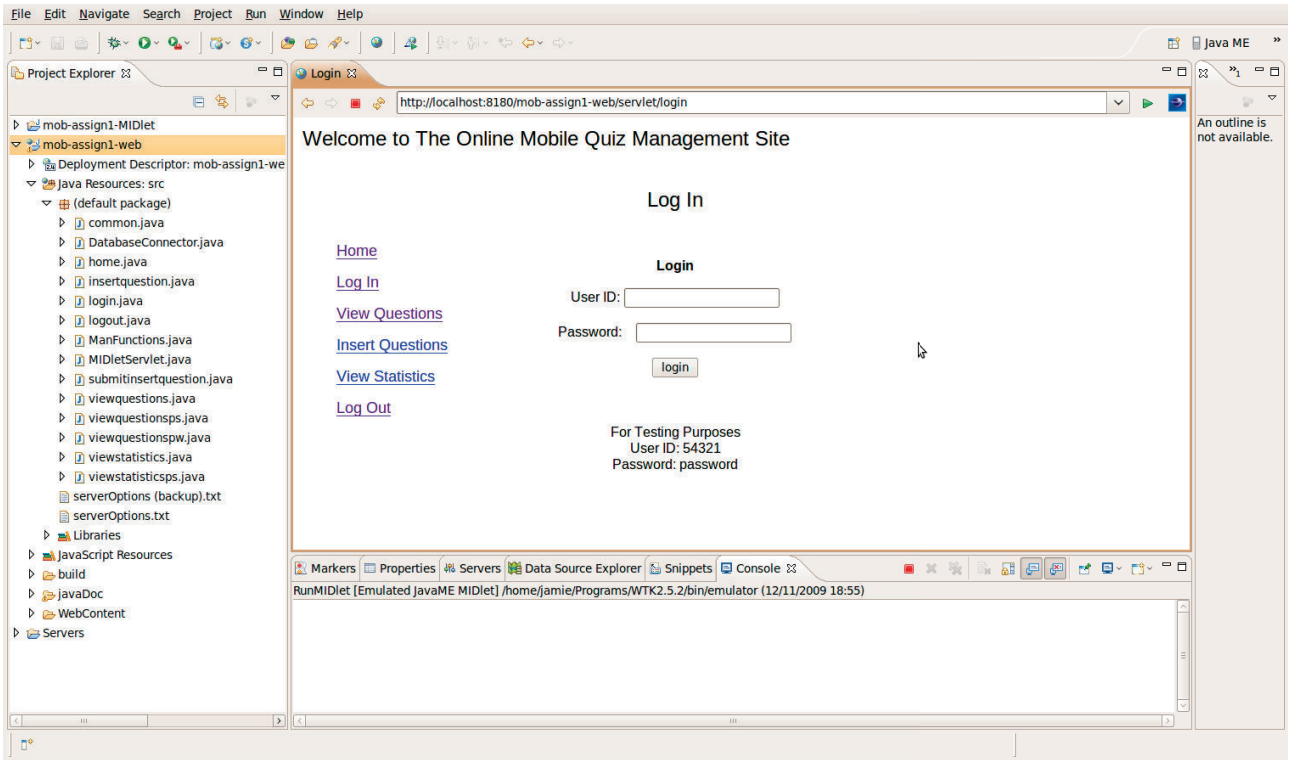


The Administration Web Page and Servlets:

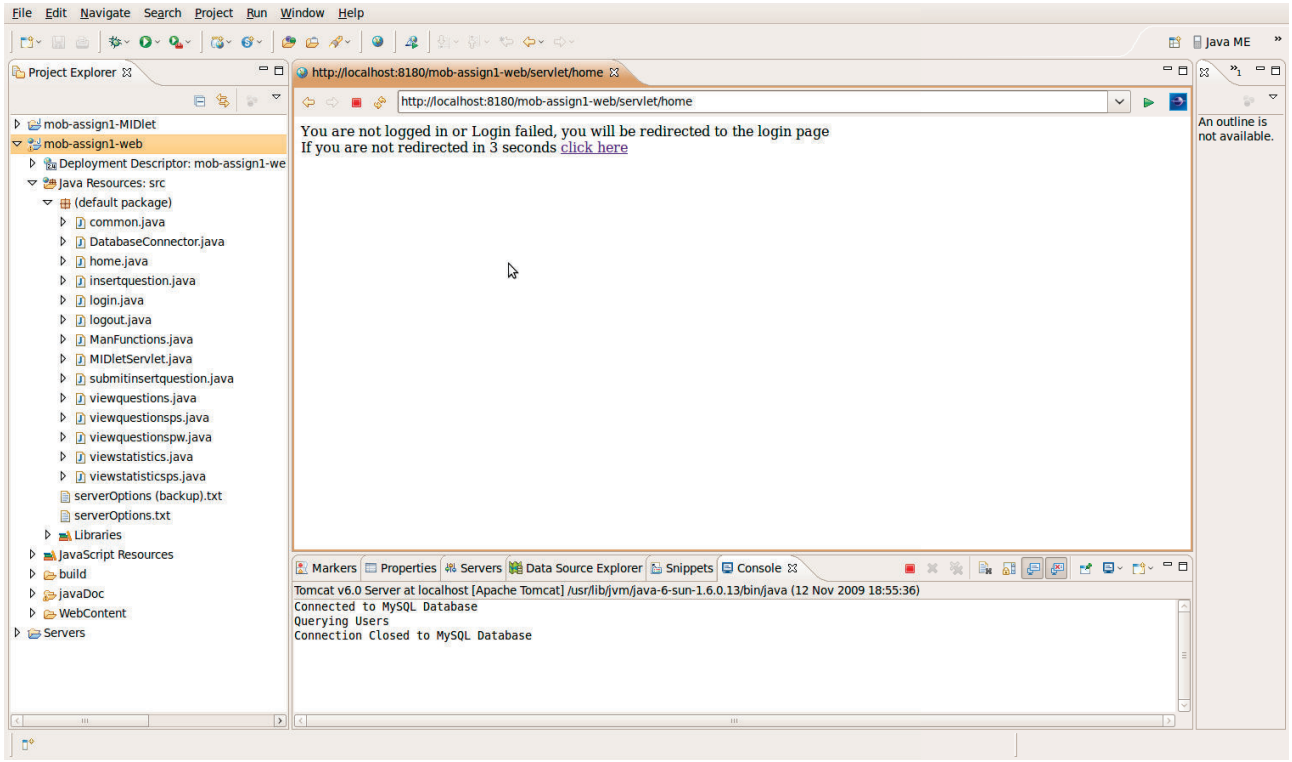
index:



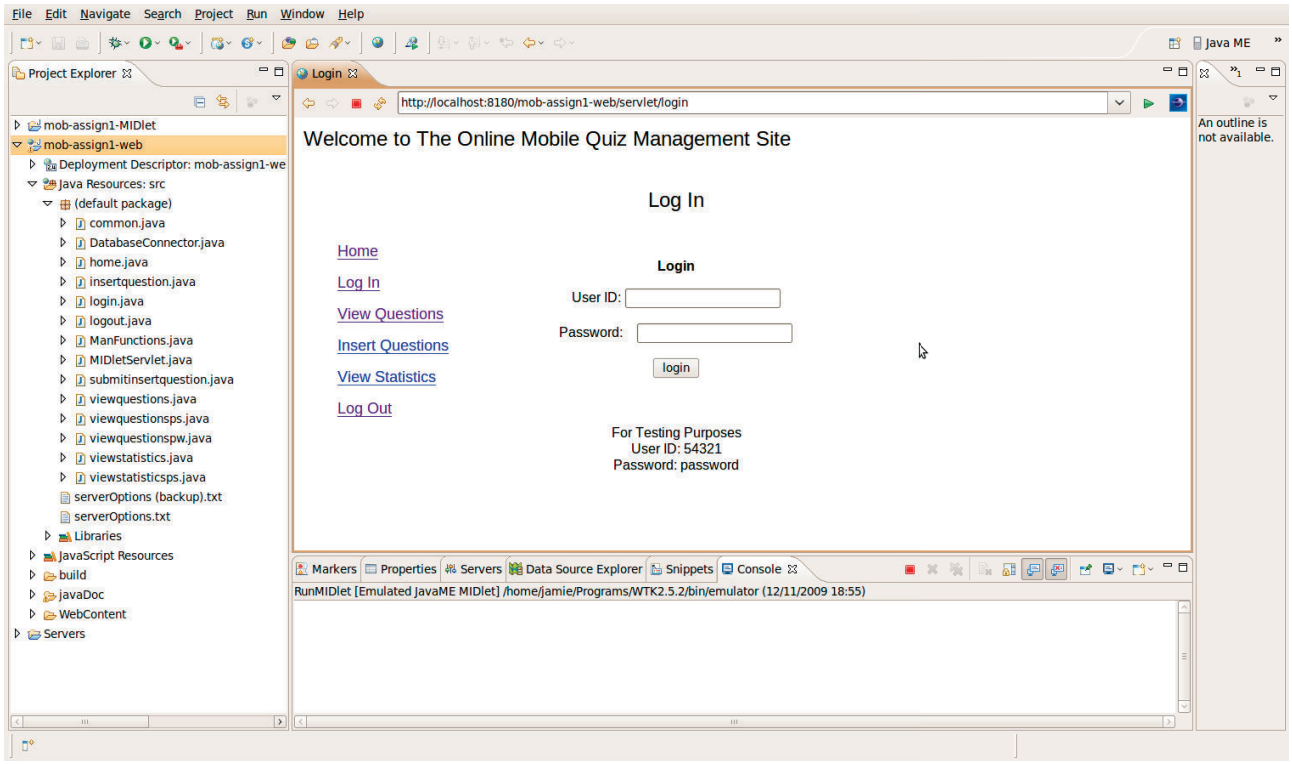
Clicked to login: Should take us to the login website



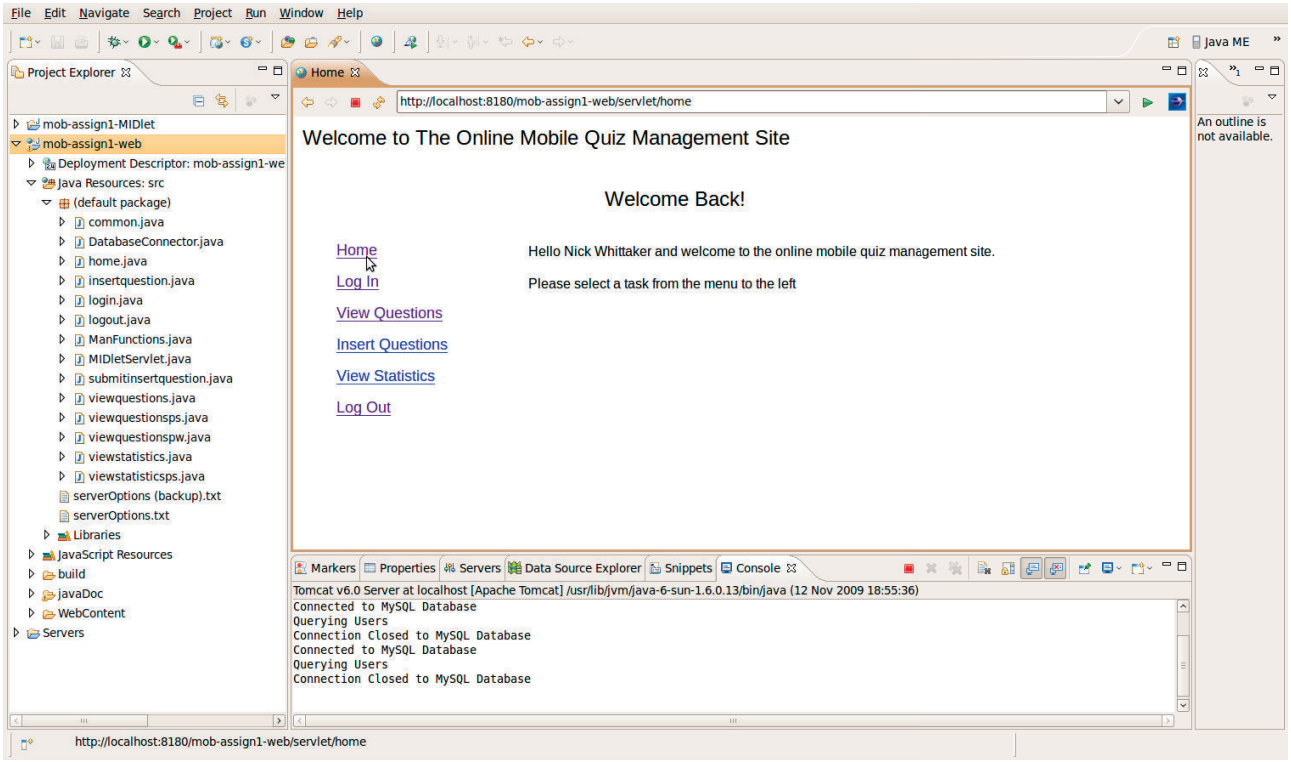
No other parts of the website should be accessible: A not logged in message should appear before being redirected to the login page



after 3 seconds...

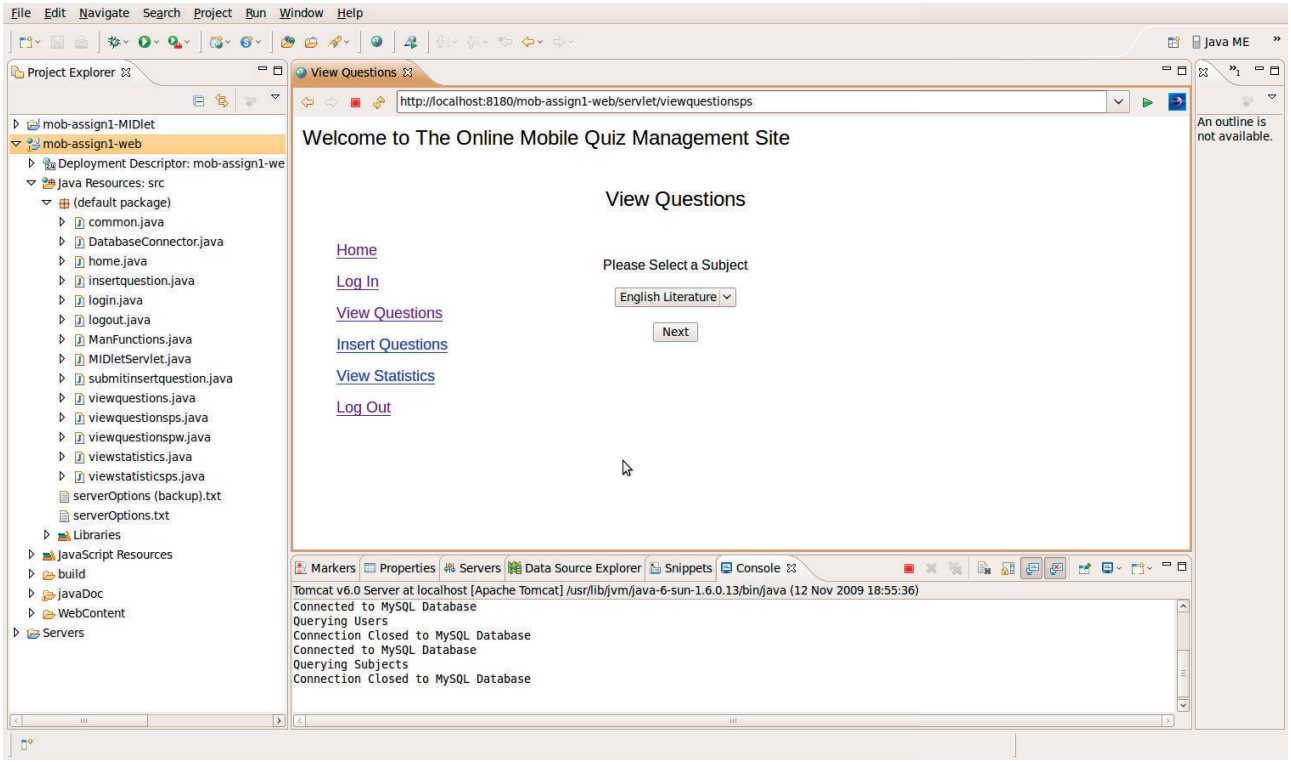


Login details entered and submitted: Should being us to the welcome page and also parts of the website should be accessible

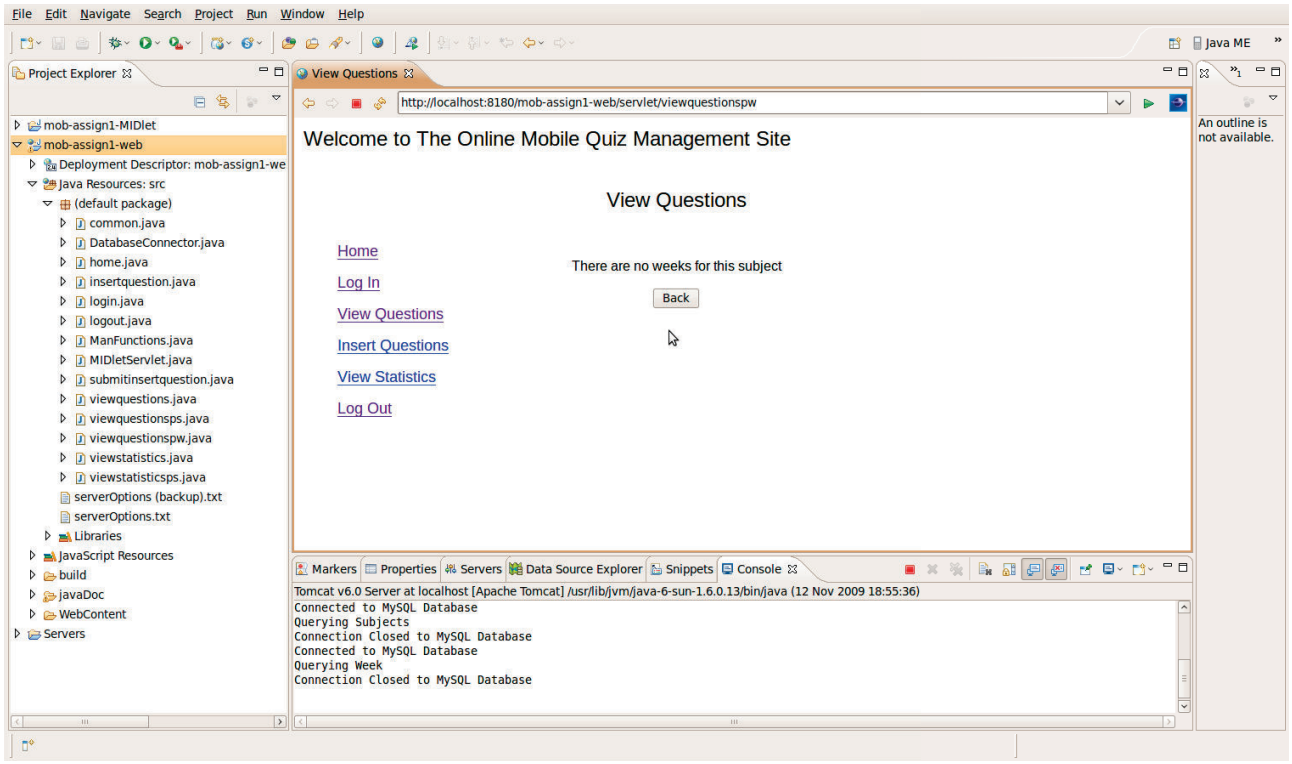


Cookies are included as part of the website, a long term cookie is created, hence the 'welcome back' message, other parts of the website are not accessible due to a single session validation cookie being set to true.

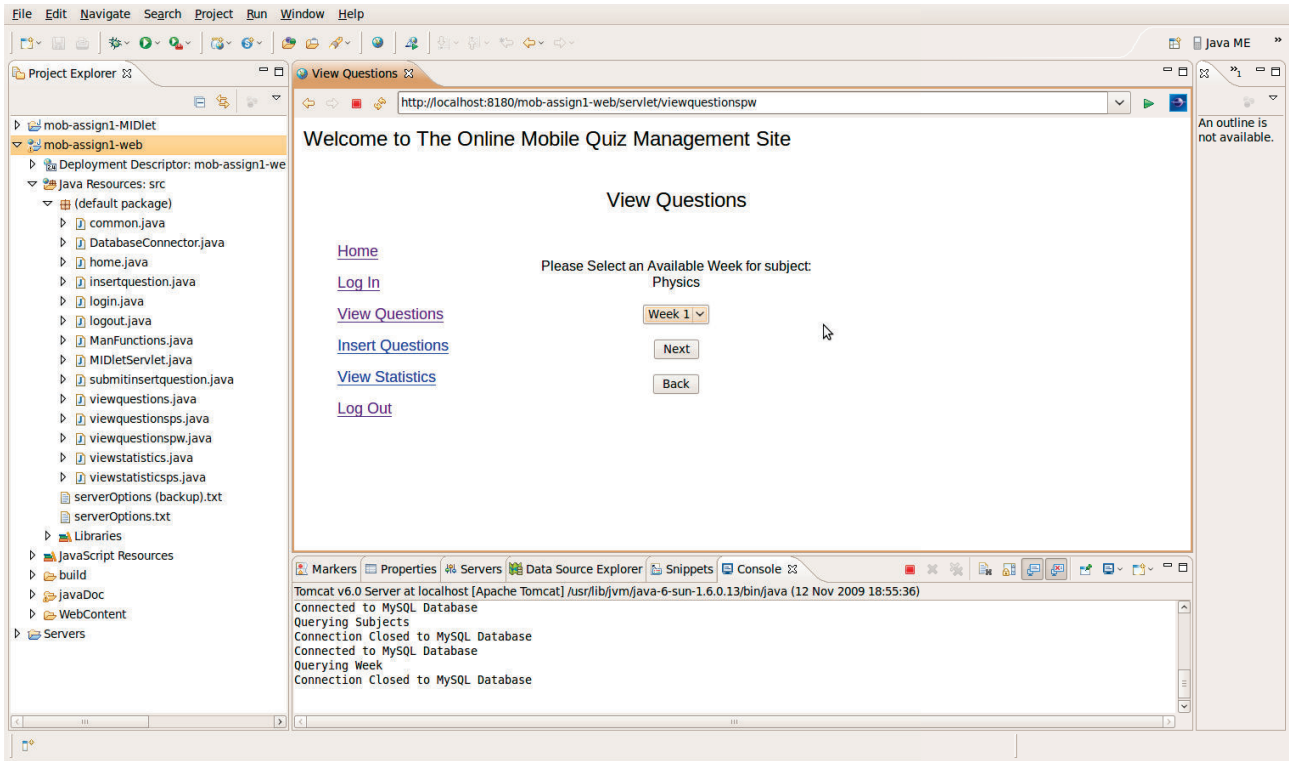
View questions selected: Should bring up a 'select subject' page:



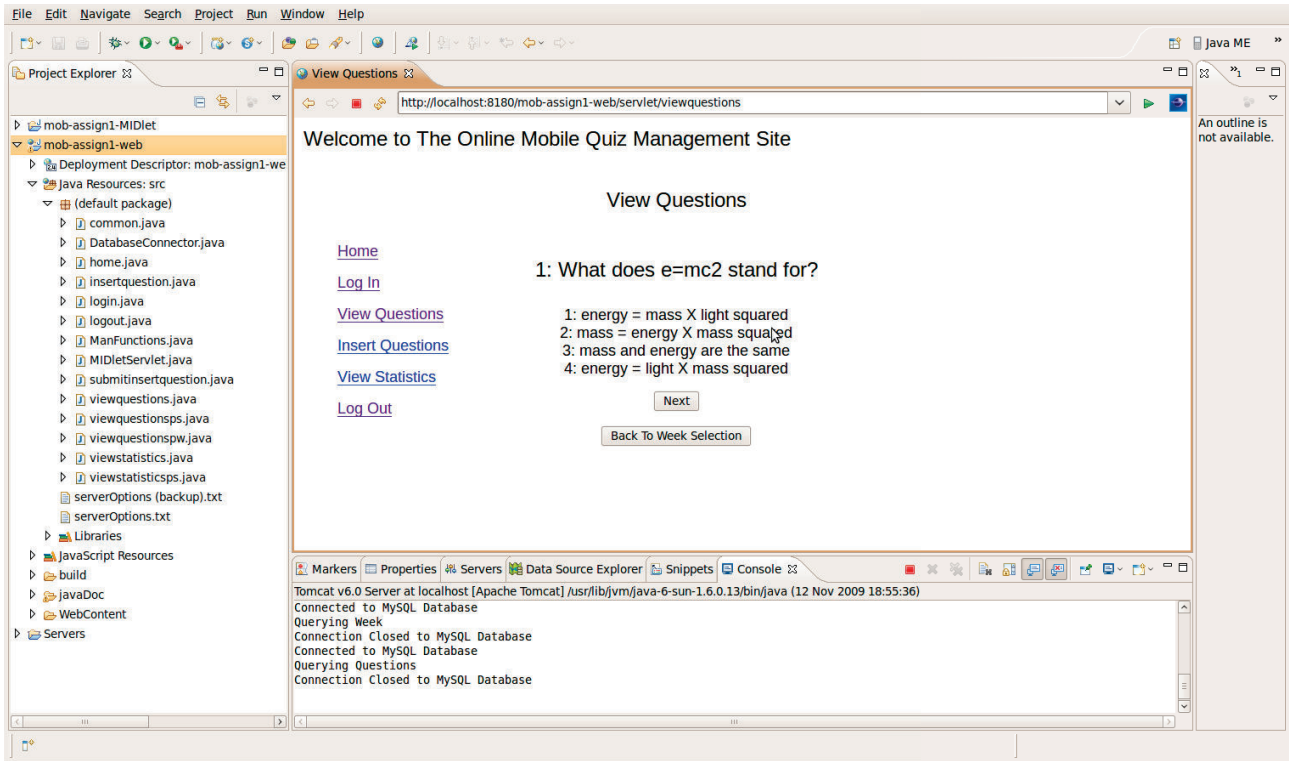
Subject selected which contains no weeks: Should give 'no questions available' message:



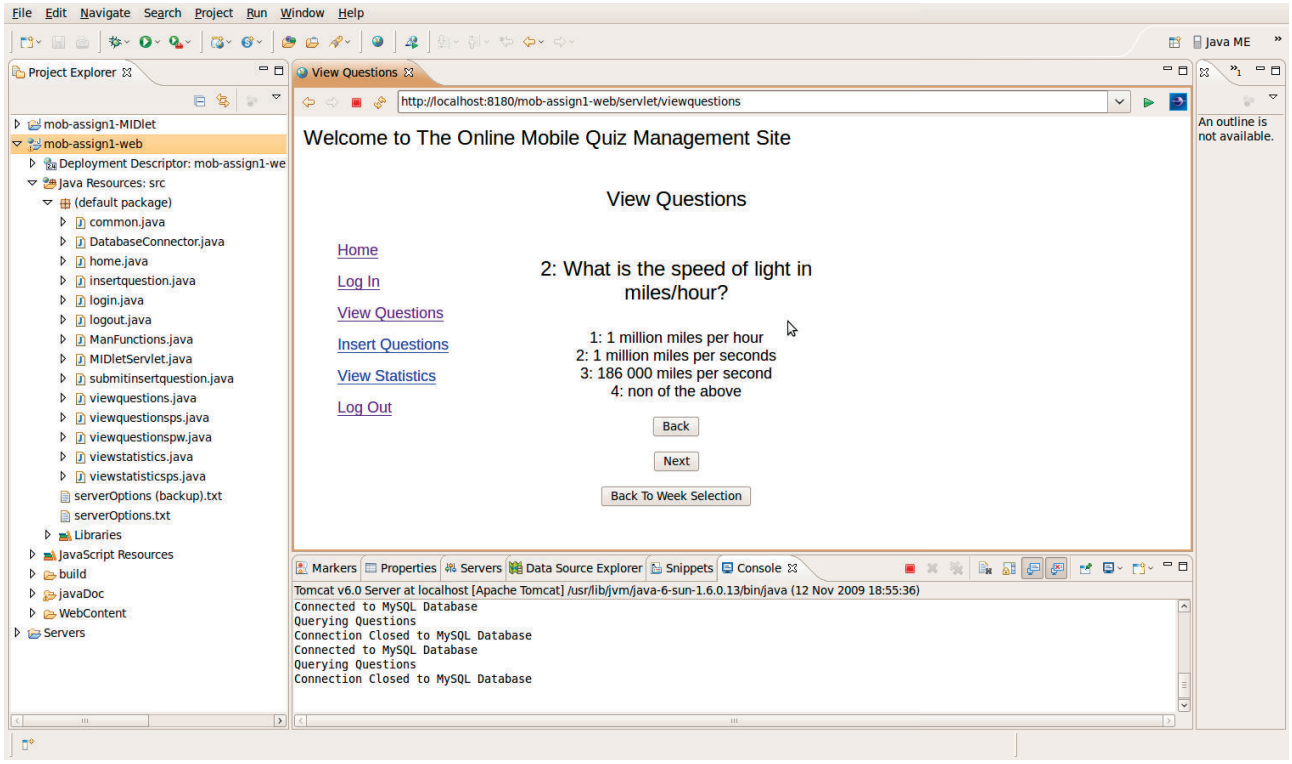
Back is selected, then the subject 'Physics' is selected which does contain questions: A week selection page should appear



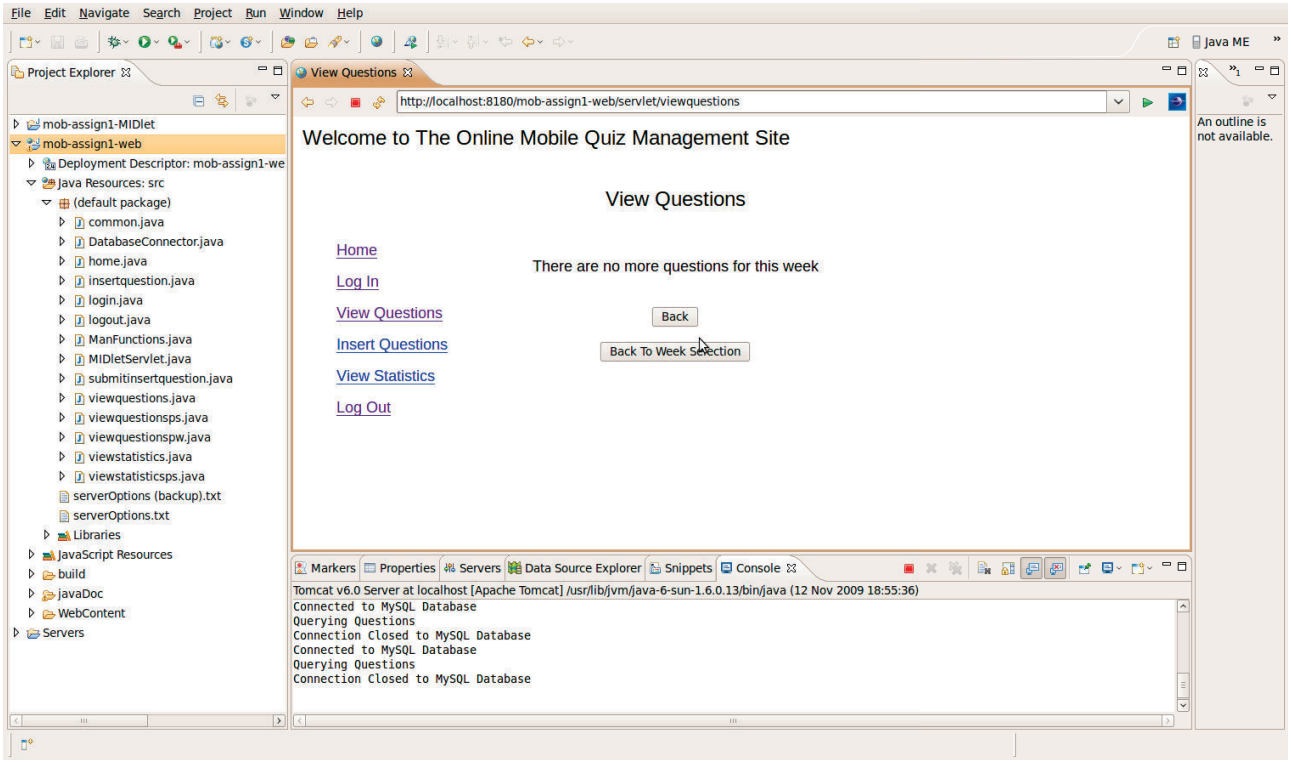
Week 1 selected: Should bring up the first question



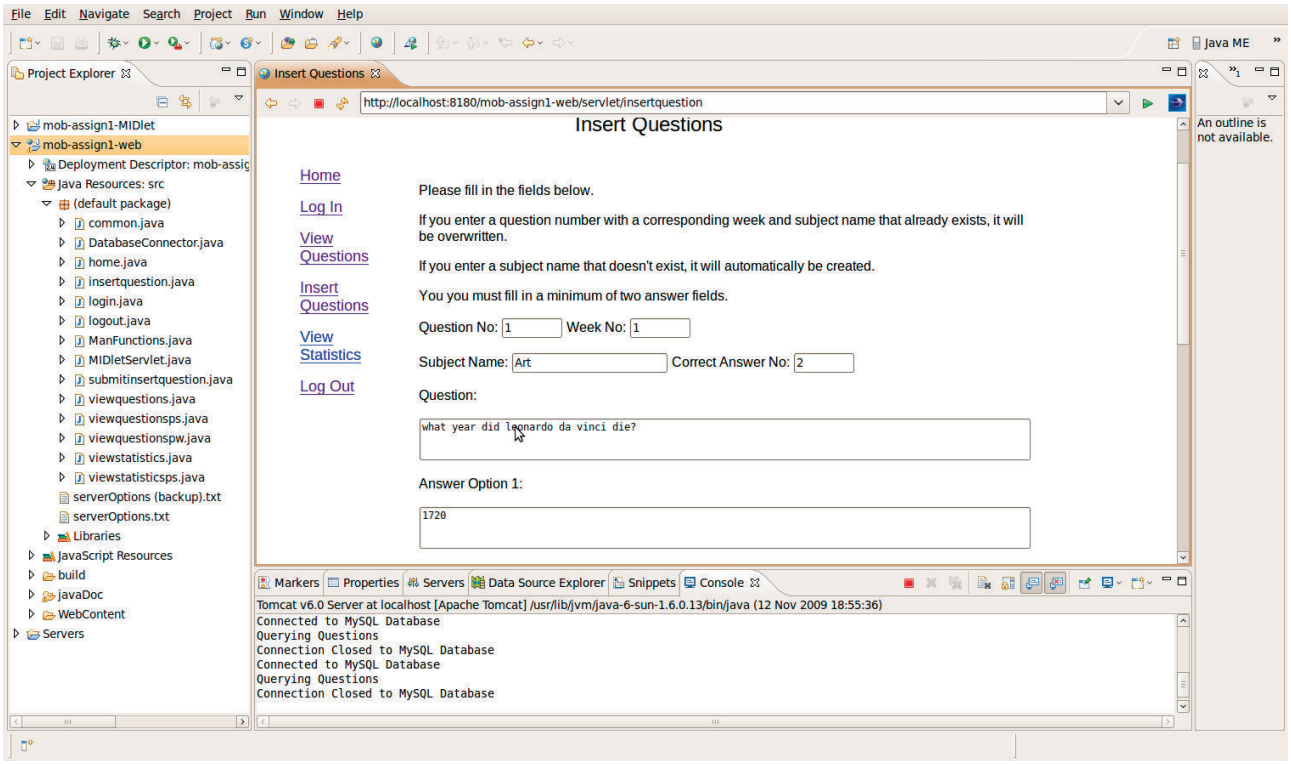
Next selected: Should bring up the next question



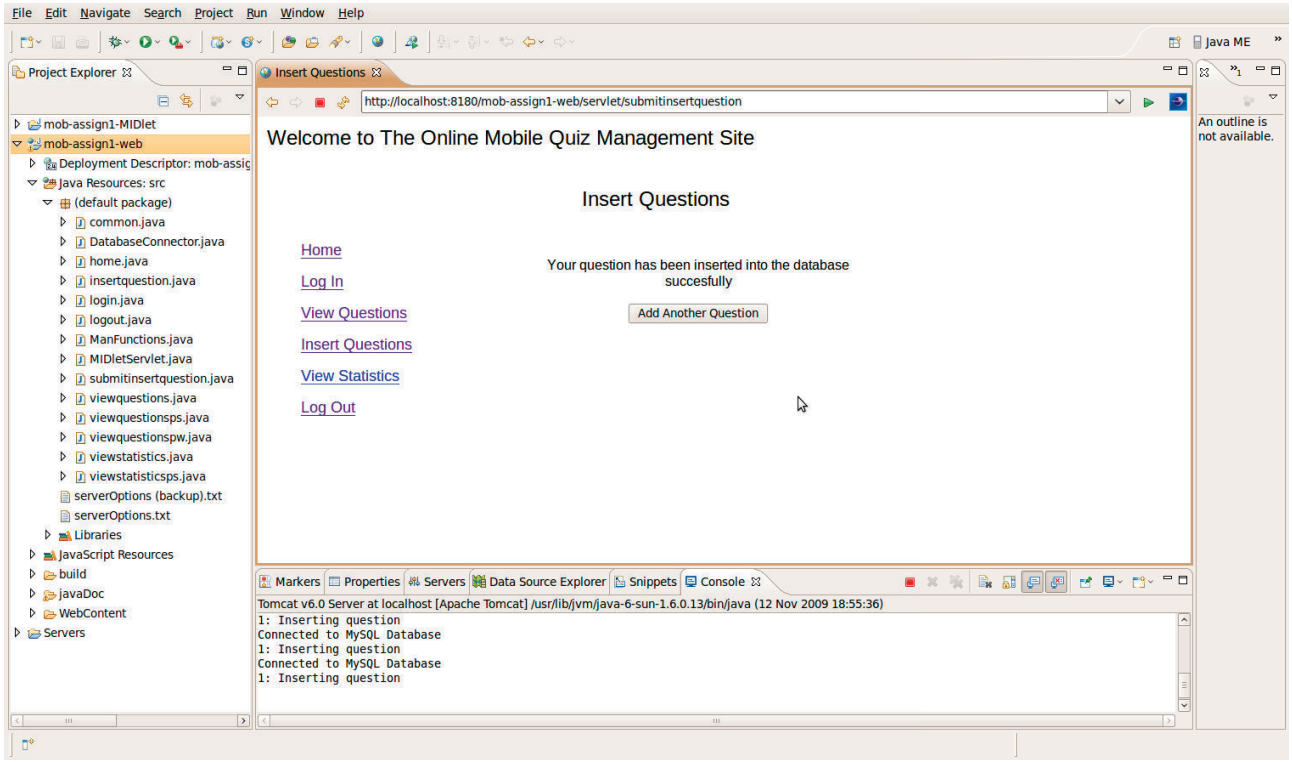
Next selected: As there are no more questions for this week, a 'no more questions' message should appear:



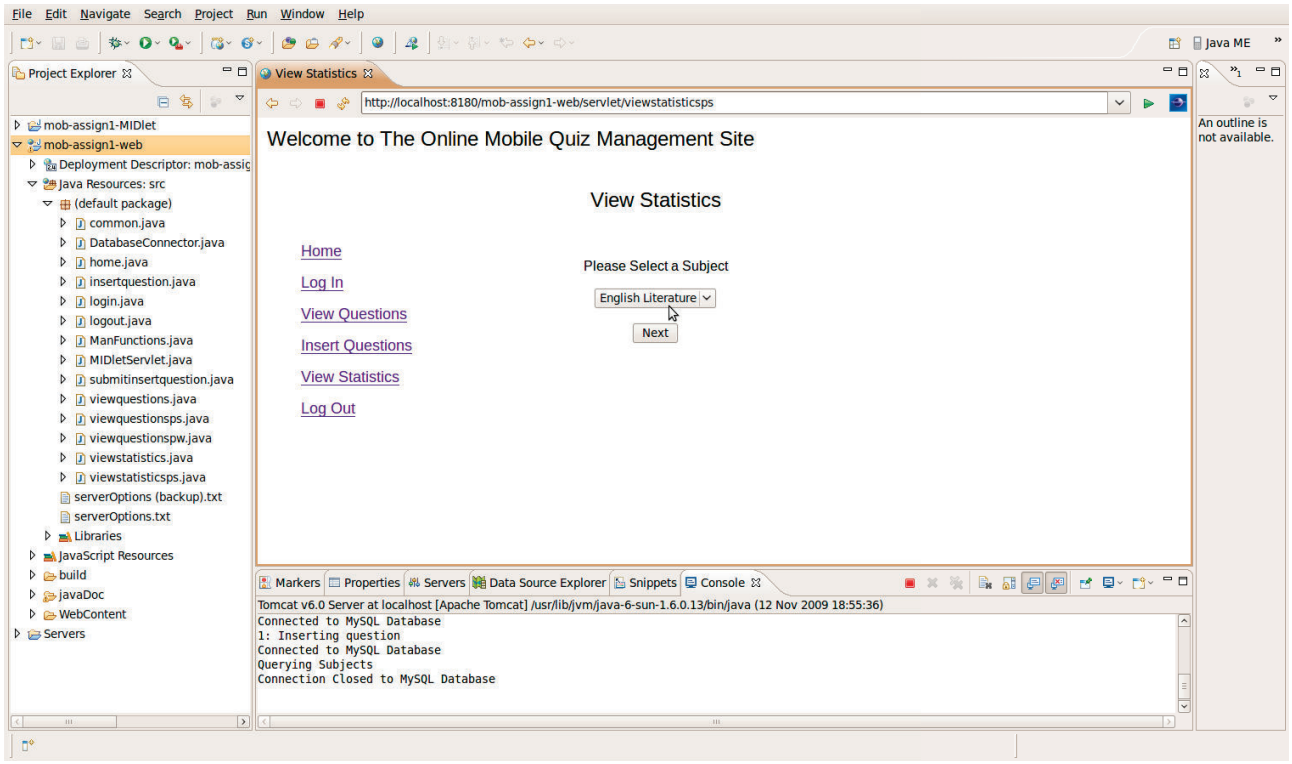
Insert questions link selected: Should bring up a page allowing a tutor to add / amend a question:



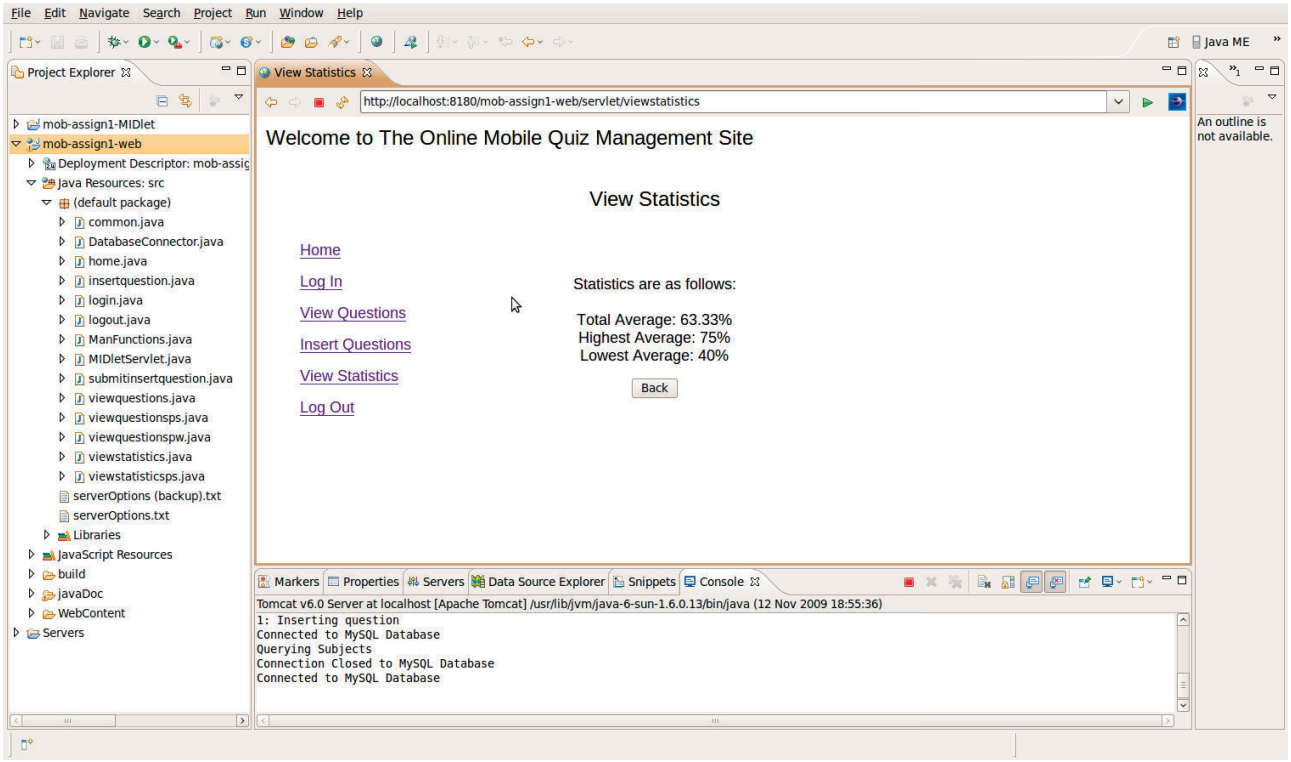
A new question is filled out, for a new week for a subject containing no current questions and submitted: A page should appear confirming the insertion of that question to the database:



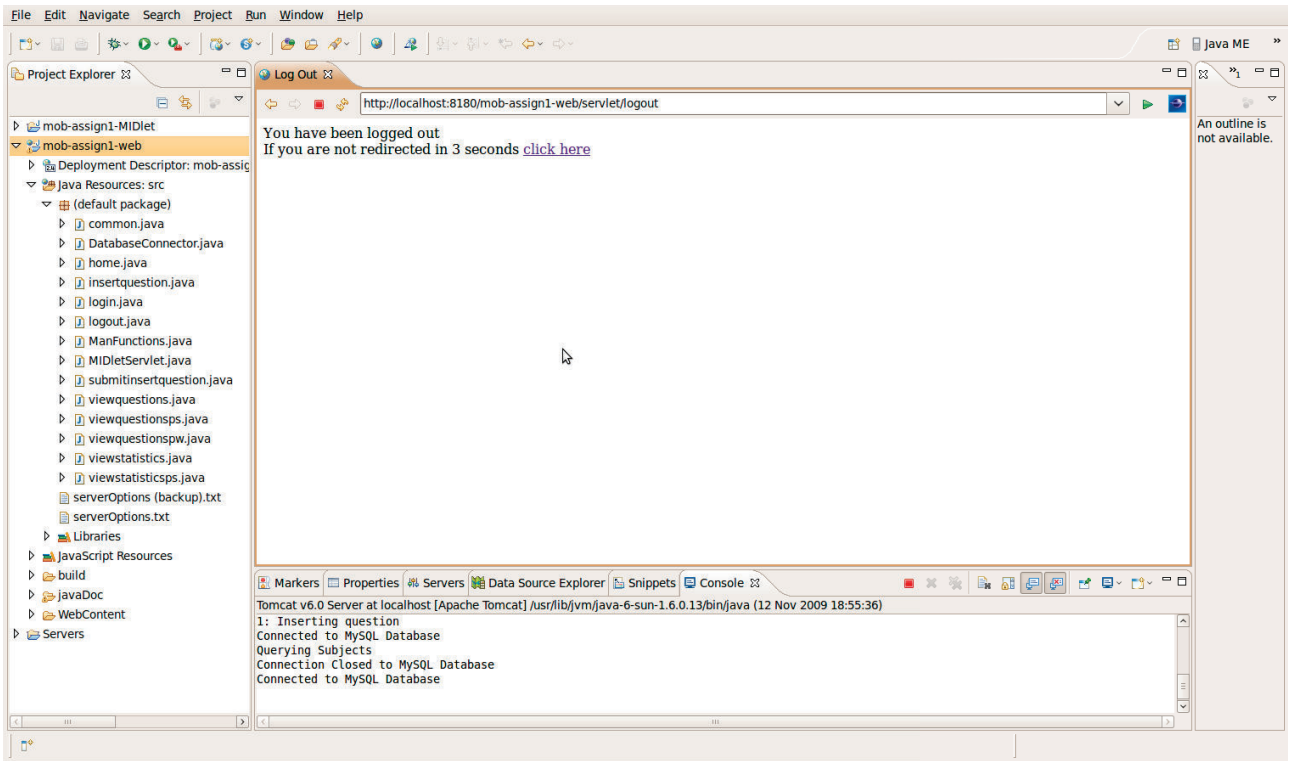
The view statistics link is pressed: Should bring up a subject selection page:



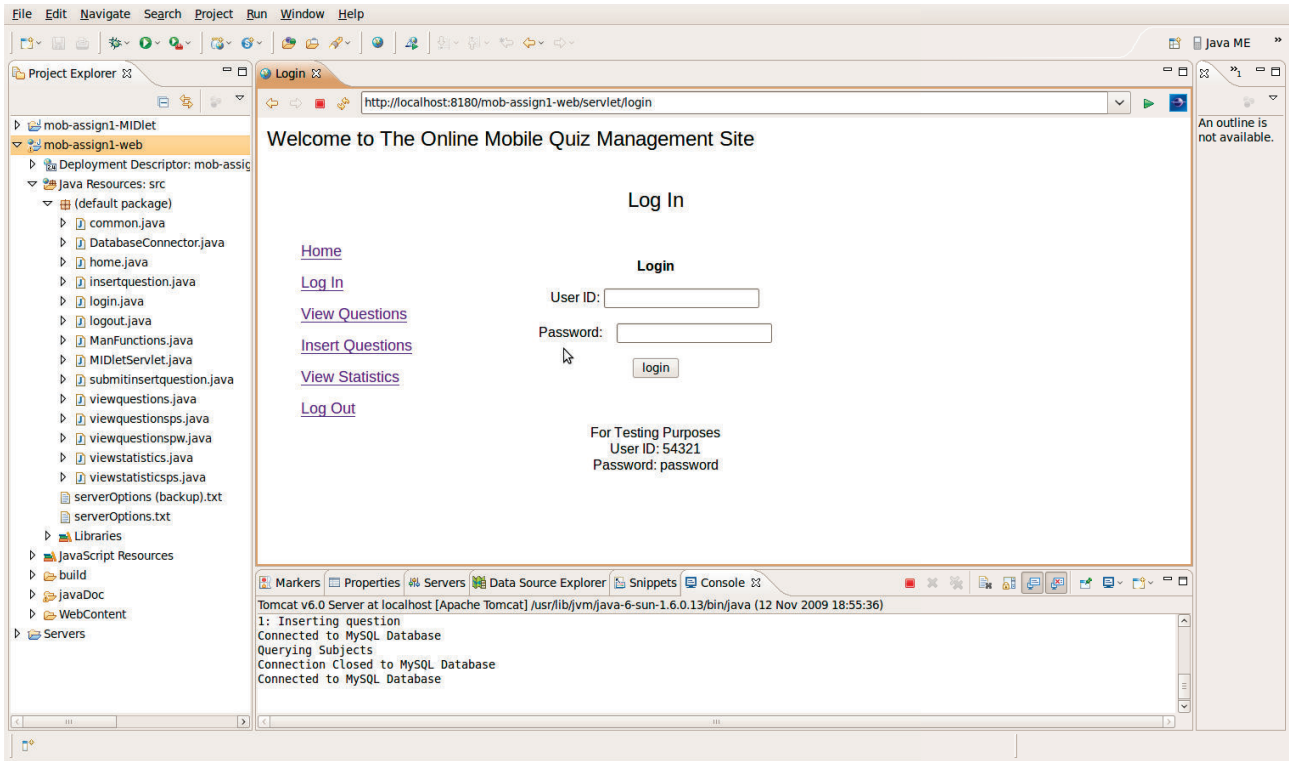
Physics is selected: Should bring up the statical results for physics, which states a list of average percentage results of users that have answered the questions



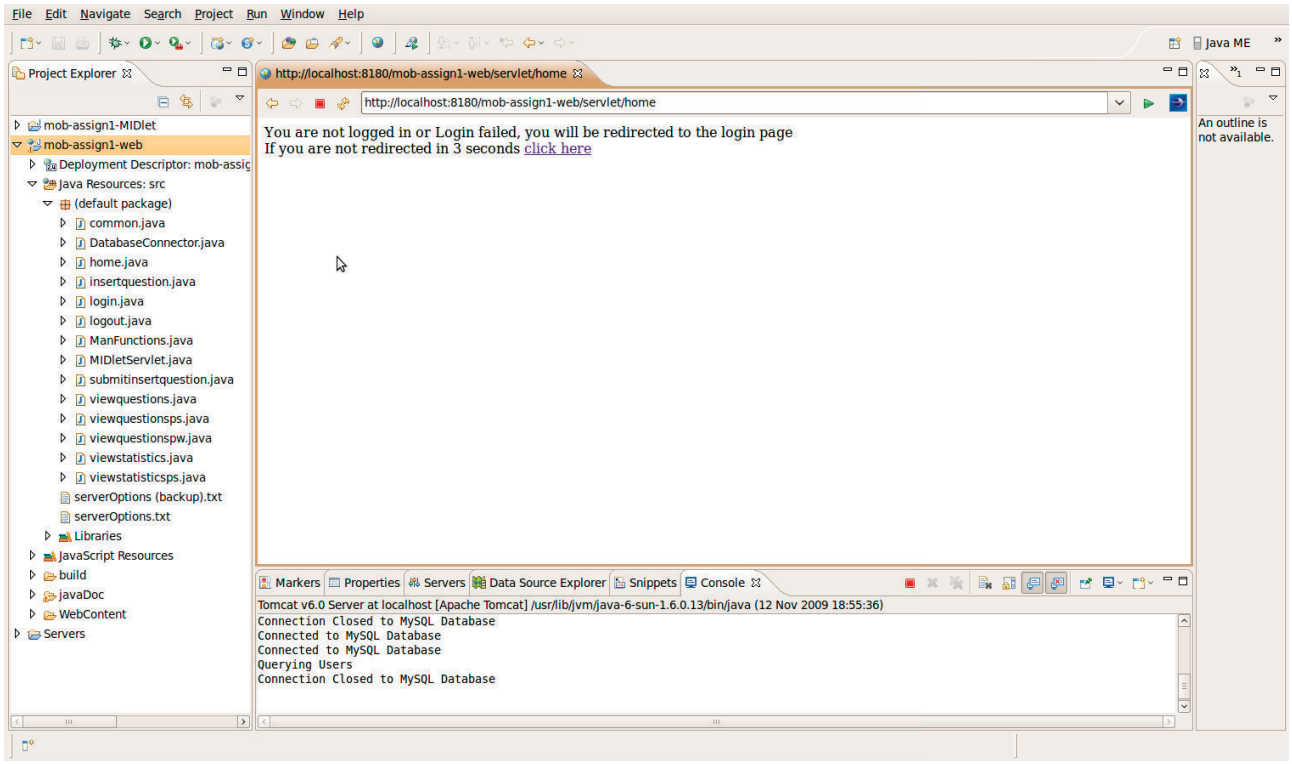
Logout link is pressed: Should bring up a log out message before being redirected to the login page:



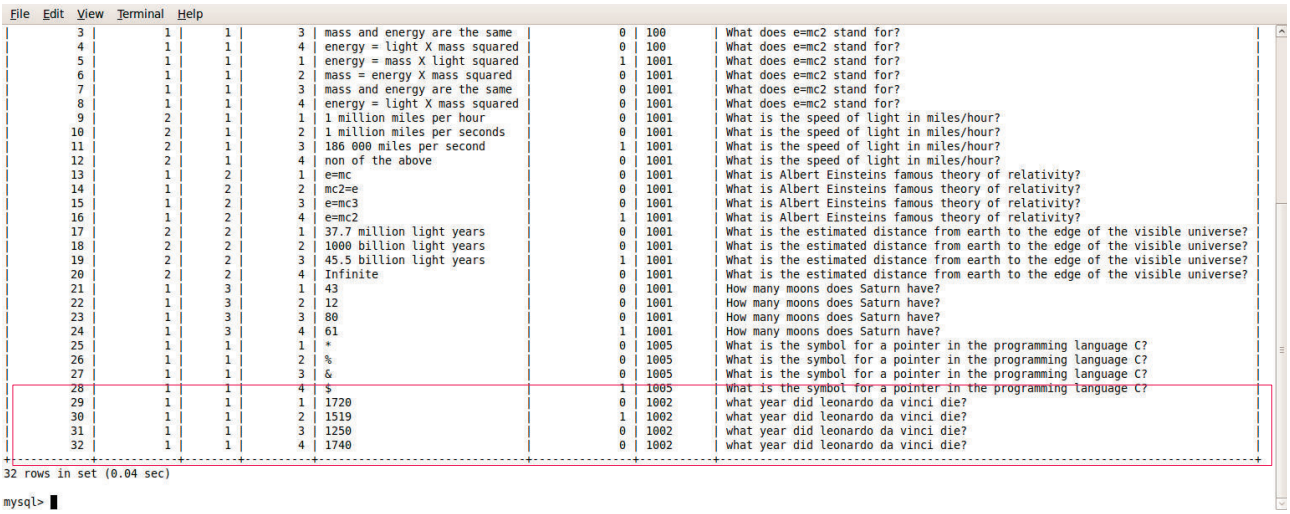
then the login page...



Now all other parts of the pages should be blocked as the validation cookie has now been set to false:



Confirmation of mysql database question insertion:



```
File  Edit  View  Terminal  Help
|         3 |          1 |       1 |         3 | mass and energy are the same  |             0 | 100       | What does e=mc2 stand for?                                                                  |
|         4 |          1 |       1 |         4 | energy = light X mass squared |             0 | 100       | What does e=mc2 stand for?                                                                  |
|         5 |          1 |       1 |         1 | energy = mass X light squared |             1 | 1001      | What does e=mc2 stand for?                                                                  |
|         6 |          1 |       1 |         2 | mass = energy X mass squared  |             0 | 1001      | What does e=mc2 stand for?                                                                  |
|         7 |          1 |       1 |         3 | mass and energy are the same  |             0 | 1001      | What does e=mc2 stand for?                                                                  |
|         8 |          1 |       1 |         4 | energy = light X mass squared |             0 | 1001      | What does e=mc2 stand for?                                                                  |
|         9 |          2 |       1 |         1 | 1 million miles per hour      |             0 | 1001      | What is the speed of light in miles/hour?                                                   |
|        10 |          2 |       1 |         2 | 1 million miles per seconds   |             0 | 1001      | What is the speed of light in miles/hour?                                                   |
|        11 |          2 |       1 |         3 | 186 000 miles per second      |             1 | 1001      | What is the speed of light in miles/hour?                                                   |
|        12 |          2 |       1 |         4 | non of the above              |             0 | 1001      | What is the speed of light in miles/hour?                                                   |
|        13 |          1 |       2 |         1 | e=mc                          |             0 | 1001      | What is Albert Einsteins famous theory of relativity?                                       |
|        14 |          1 |       2 |         2 | mc2=e                         |             0 | 1001      | What is Albert Einsteins famous theory of relativity?                                       |
|        15 |          1 |       2 |         3 | e=mc3                         |             0 | 1001      | What is Albert Einsteins famous theory of relativity?                                       |
|        16 |          1 |       2 |         4 | e=mc2                         |             1 | 1001      | What is Albert Einsteins famous theory of relativity?                                       |
|        17 |          2 |       2 |         1 | 37.7 million light years      |             0 | 1001      | What is the estimated distance from earth to the edge of the visible universe?              |
|        18 |          2 |       2 |         2 | 1000 billion light years      |             0 | 1001      | What is the estimated distance from earth to the edge of the visible universe?              |
|        19 |          2 |       2 |         3 | 45.5 billion light years      |             1 | 1001      | What is the estimated distance from earth to the edge of the visible universe?              |
|        20 |          2 |       2 |         4 | Infinite                      |             0 | 1001      | What is the estimated distance from earth to the edge of the visible universe?              |
|        21 |          1 |       3 |         1 | 43                            |             0 | 1001      | How many moons does Saturn have?                                                            |
|        22 |          1 |       3 |         2 | 12                            |             0 | 1001      | How many moons does Saturn have?                                                            |
|        23 |          1 |       3 |         3 | 80                            |             0 | 1001      | How many moons does Saturn have?                                                            |
|        24 |          1 |       3 |         4 | 61                            |             1 | 1001      | How many moons does Saturn have?                                                            |
|        25 |          1 |       1 |         1 | *                             |             0 | 1005      | What is the symbol for a pointer in the programming language C?                             |
|        26 |          1 |       1 |         2 | %                             |             0 | 1005      | What is the symbol for a pointer in the programming language C?                             |
|        27 |          1 |       1 |         3 | &                             |             0 | 1005      | What is the symbol for a pointer in the programming language C?                             |
|        28 |          1 |       1 |         4 | $                             |             1 | 1005      | What is the symbol for a pointer in the programming language C?                             |
|        29 |          1 |       1 |         1 | 1720                          |             0 | 1002      | what year did leonardo da vinci die?                                                        |
|        30 |          1 |       1 |         2 | 1519                          |             1 | 1002      | what year did leonardo da vinci die?                                                        |
|        31 |          1 |       1 |         3 | 1250                          |             0 | 1002      | what year did leonardo da vinci die?                                                        |
|        32 |          1 |       1 |         4 | 1740                          |             0 | 1002      | what year did leonardo da vinci die?                                                        |
+-----------+------------+---------+-----------+-------------------------------+---------------+-----------+---------------------------------------------------------------------------------------------+
32 rows in set (0.04 sec)

mysql>
```

Entering wrong database into the insert questions page text fields. Such as a letter where a number should be: Should bring up an error warning: