|  |
| --- |
| Asia pasific university of technology & innovation |
| Project Report |
| Mobile Application And Systems |
|  |
| **Muhammad Fayzan Siddiqui**  **Student ID: TP033089**  **Lecturer: Selvakumar Samuel** |
| **Date: 5th May 2014** |

|  |
| --- |
|  |

Table of Contents

[**1.** **Introduction** 2](#_Toc387071213)

[**2.** **Application description with story board** 3](#_Toc387071214)

[**3.** **Class Diagram** 13](#_Toc387071215)

[**4.** **Important Source code** 14](#_Toc387071216)

[**5.** **Framework for Data Storage** 25](#_Toc387071217)

[**6.** **Development Platform and Tools** 26](#_Toc387071220)

[**7.** **Application dependencies and Running Instructions** 26](#_Toc387071223)

[**8.** **Future Work** 27](#_Toc387071224)

1. **Introduction**

The cash quiz game is a general knowledge game, in which the player can earn money by using their knowledge. The game will ask questions to the user, it’s a multiple choice question, each question have 4 possible answers, and the player have to select one from them. If the answer is correct the player will win some cash price. The cash price will be double after every correct answer. There are total 6 questions; if the user gives all correct answers they can win $1024.

Each wrong answer will result into game over. If the user give incorrect answer of any question (1, 2, 3, 4) before reaching to the 5th question than the player will win nothing. But after reaching the 5th question if the player loose (i.e. give wrong answer) then they will get $256 (4 questions answered correctly).

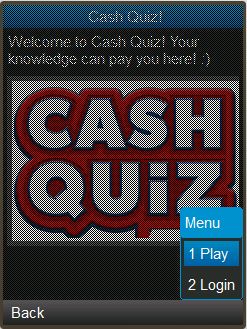
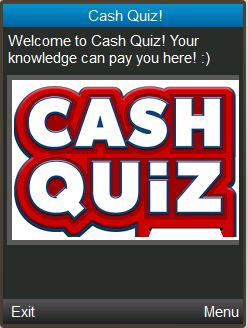
During answering the questions, the player has two life lines which can aid the player decision. One life line is 50:50, if player select this then the application will remove two incorrect answers out of four, now the user have to select the answer from two options. The other life line is audience pool, if this option is used then the application conduct the polling from audience that which answer is correct according to them (in current application polling is done randomly) and then display the polling result to the player, it’s up to the player weather want to go with the answer having higher rating from poll or any other option. These two life line can only be used once in the entire game, so the player need to be think before using it.

1. **Application description with story board**

The application has two parts.

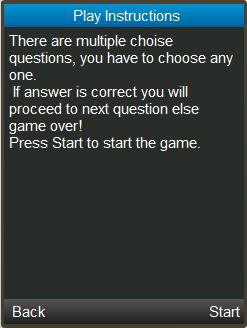
1. Play Game: this is for the player to start the game and play.
2. Admin Panel: this is for the administrator to manage application, it requires admin login.

Welcome screen and menu.

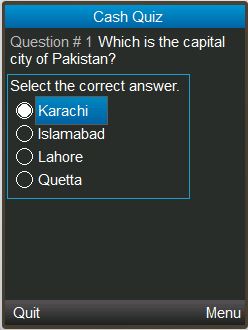


* 1. **Play Game**

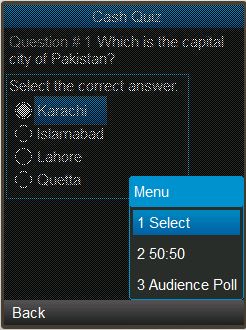
When the play mode is selected form the welcome screen, the play instruction is displayed first.

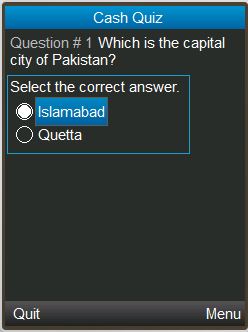
****

After press Start command, the game started and displayed the quiz question.

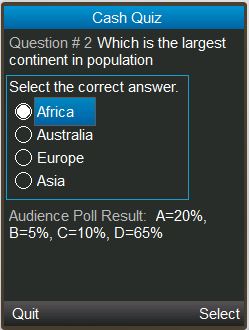


The question will have 3 command in menu Select, 50:50 and Audience Poll. The 50:50 and Audience Poll will be removed once used.

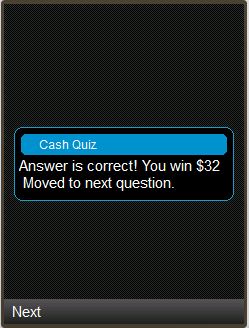


When50:50 option is selected it will remove the two incorrect answers.****

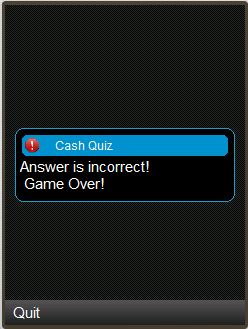
When audience poll is selected it will display the poll result.



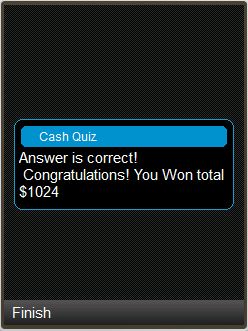
If selected answer is correct it will show the win amount.



If answer is incorrect the game is over.

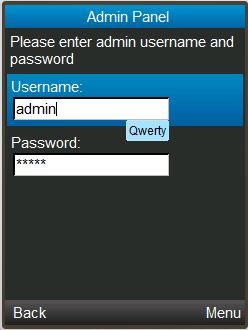


Giving all correct answers, the player will win total amount $1024.

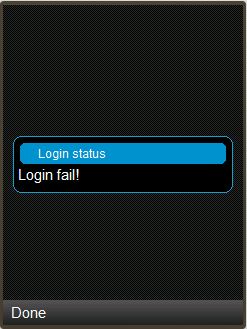


* 1. **Admin Panel**

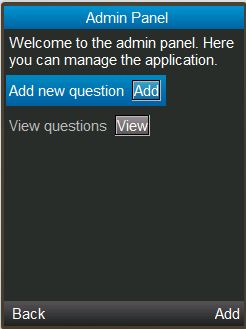
When admin is selected from the welcome screen, it will ask for admin username and password.



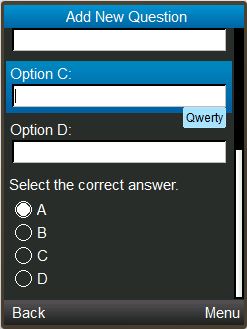
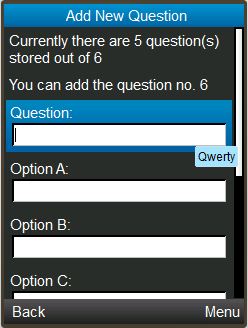
If username and password is incorrect, it will show login fail error.



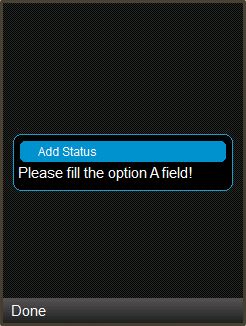
Upon successful login, the admin panel is displayed. Admin panel have two menus Add and View questions.



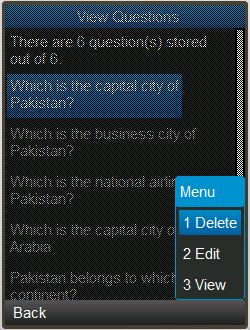
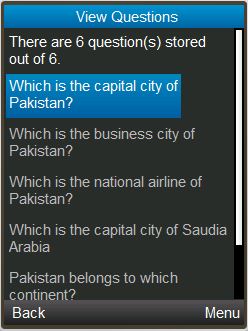
If add question is selected it will display a form to add new question.



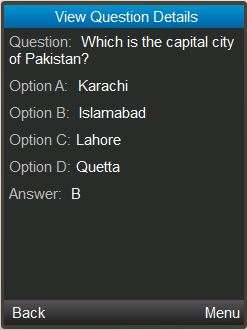
All fields of the add question form are required fields, application validate all fields before add question.

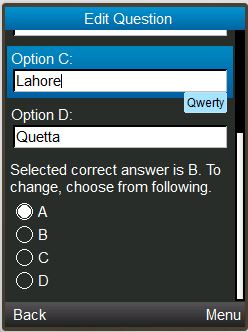
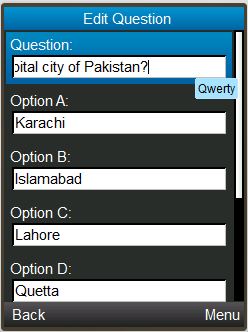
If view question is selected, it will display a list of already added question along with option to Delete, Edit and detail view.



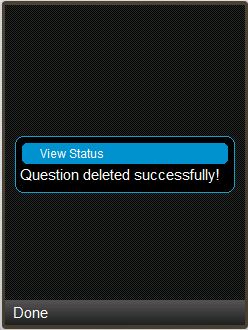
When view is selected it will show the detailed view of selected question.



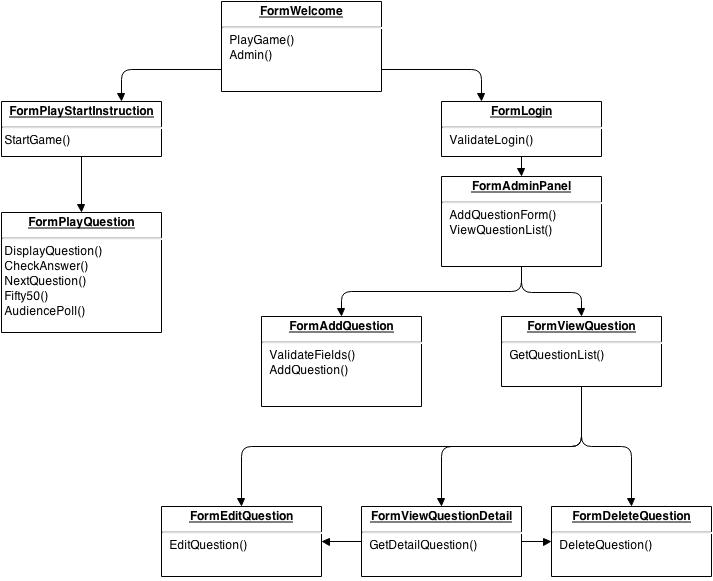
When edit command is selected it will show the form to edit the question.



When delete command is selected it will delete selected question.



1. **Class Diagram**



1. **Important Source code**
   1. **Admin Login**

if (c == cmdOk) {

if (tfUsername.getString().equals("admin") && tfPassword.getString().equals("admin")) {

//success

FormAdminPanel frmAdminPanel = new FormAdminPanel(midlet);

display.setCurrent(frmAdminPanel);

}else {

//fail

atStatus.setString("Login fail!");

display.setCurrent(atStatus);

}

}

* 1. **Add Question**

if (c == cmdAdd) {

if(txtQuestion.getString().equals("")) {

atStatus.setString("Please fill the question field!");

display.setCurrent(atStatus);

return;

}

if(txtOptA.getString().equals("")) {

atStatus.setString("Please fill the option A field!");

display.setCurrent(atStatus);

return;

}

if(txtOptB.getString().equals("")) {

atStatus.setString("Please fill the option B field!");

display.setCurrent(atStatus);

return;

}

if(txtOptC.getString().equals("")) {

atStatus.setString("Please fill the option C field!");

display.setCurrent(atStatus);

return;

}

if(txtOptD.getString().equals("")) {

atStatus.setString("Please fill the option D field!");

display.setCurrent(atStatus);

return;

}

Question objQuestion = new Question();

objQuestion.setQuestion(txtQuestion.getString());

objQuestion.setOptionA(txtOptA.getString());

objQuestion.setOptionB(txtOptB.getString());

objQuestion.setOptionC(txtOptC.getString());

objQuestion.setOptionD(txtOptD.getString());

objQuestion.setAnswer(cgAnswer.getString(cgAnswer.getSelectedIndex()));

AdminMethods actionMethod = new AdminMethods();

if(actionMethod.addQuestion(objQuestion) > 0 ){

atStatus.setString("Question added successfully!");

txtQuestion.setString("");

txtOptA.setString("");

txtOptB.setString("");

txtOptC.setString("");

txtOptD.setString("");

cgAnswer.setSelectedIndex(1, true);

}else{

atStatus.setString("Uable to add question.");

}

display.setCurrent(atStatus);

loadContent();

}

* 1. **Delete Question**

private boolean deleteQuestion(Question question) {

AdminMethods actionMethods = new AdminMethods();

boolean retVal;

if(actionMethods.deleteQuestion(question)) {

atStatus.setString("Question deleted successfully!");

title.setText("There are " + (numQuestions-1) + " question(s) stored out of 6. ");

retVal = true;

} else {

atStatus.setString("Unable to delete question");

retVal = false;

}

display.setCurrent(atStatus);

return retVal;

}

* 1. **Edit Question**

if (c == cmdEdit) {

Question newQuestion = new Question();

newQuestion.setQuestion(strQuestion.getString());

newQuestion.setOptionA(optionA.getString());

newQuestion.setOptionB(optionB.getString());

newQuestion.setOptionC(optionC.getString());

newQuestion.setOptionD(optionD.getString());

newQuestion.setAnswer(answer.getString(answer.getSelectedIndex()));

this.editQuestion(this.objQuestion, newQuestion);

}

* 1. **View Questions**

private void renderQuestions() {

AdminMethods actionMethods = new AdminMethods();

this.questions = new Question[actionMethods.getNumberQuestions()]; // clear previous state

this.questions = actionMethods.getAllQuestions();

numQuestions = questions.length;

title.setText("There are " + numQuestions + " question(s) stored out of 6. ");

//question 1

if (numQuestions >= 1) {

strQuestion1 = new StringItem(questions[0].getQuestion(), "");

cmdDeleteQuestion1 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion1 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail1 = new Command("View", Command.ITEM, 1);

strQuestion1.setDefaultCommand(cmdDeleteQuestion1);

strQuestion1.setDefaultCommand(cmdEditQuestion1);

strQuestion1.setDefaultCommand(cmdViewQuestionDetail1);

strQuestion1.setItemCommandListener(this);

this.append(strQuestion1);

}

//question 2

if (numQuestions >= 2) {

strQuestion2 = new StringItem(questions[1].getQuestion(), "");

cmdDeleteQuestion2 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion2 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail2 = new Command("View", Command.ITEM, 1);

strQuestion2.setDefaultCommand(cmdDeleteQuestion2);

strQuestion2.setDefaultCommand(cmdEditQuestion2);

strQuestion2.setDefaultCommand(cmdViewQuestionDetail2);

strQuestion2.setItemCommandListener(this);

this.append(strQuestion2);

}

//question 3

if (numQuestions >= 3) {

strQuestion3 = new StringItem(questions[2].getQuestion(), "");

cmdDeleteQuestion3 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion3 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail3 = new Command("View", Command.ITEM, 1);

strQuestion3.setDefaultCommand(cmdDeleteQuestion3);

strQuestion3.setDefaultCommand(cmdEditQuestion3);

strQuestion3.setDefaultCommand(cmdViewQuestionDetail3);

strQuestion3.setItemCommandListener(this);

this.append(strQuestion3);

}

//question 4

if (numQuestions >= 4) {

strQuestion4 = new StringItem(questions[3].getQuestion(), "");

cmdDeleteQuestion4 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion4 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail4 = new Command("View", Command.ITEM, 1);

strQuestion4.setDefaultCommand(cmdDeleteQuestion4);

strQuestion4.setDefaultCommand(cmdEditQuestion4);

strQuestion4.setDefaultCommand(cmdViewQuestionDetail4);

strQuestion4.setItemCommandListener(this);

this.append(strQuestion4);

}

//question 5

if (numQuestions >= 5) {

strQuestion5 = new StringItem(questions[4].getQuestion(), "");

cmdDeleteQuestion5 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion5 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail5 = new Command("View", Command.ITEM, 1);

strQuestion5.setDefaultCommand(cmdDeleteQuestion5);

strQuestion5.setDefaultCommand(cmdEditQuestion5);

strQuestion5.setDefaultCommand(cmdViewQuestionDetail5);

strQuestion5.setItemCommandListener(this);

this.append(strQuestion5);

}

//question 6

if (numQuestions >= 6) {

strQuestion6 = new StringItem(questions[5].getQuestion(), "");

cmdDeleteQuestion6 = new Command("Delete", Command.ITEM, 1);

cmdEditQuestion6 = new Command("Edit", Command.ITEM, 1);

cmdViewQuestionDetail6 = new Command("View", Command.ITEM, 1);

strQuestion6.setDefaultCommand(cmdDeleteQuestion6);

strQuestion6.setDefaultCommand(cmdEditQuestion6);

strQuestion6.setDefaultCommand(cmdViewQuestionDetail6);

strQuestion6.setItemCommandListener(this);

this.append(strQuestion6);

}

}

* 1. **View Detail Questions**

private void loadContent() {

strQuestion.setText(this.objQuestion.getQuestion());

optionA.setText(this.objQuestion.getOptionA());

optionB.setText(this.objQuestion.getOptionB());

optionC.setText(this.objQuestion.getOptionC());

optionD.setText(this.objQuestion.getOptionD());

answer.setText(this.objQuestion.getAnswer());

}

* 1. **Display Play Question**

private void loadContent() {

if (this.questions.length != 6) {

strQuestion.setText("Questions are missing!");

return;

}

strQuestion.setText(this.questions[this.currQuestionNo-1].getQuestion());

strQuestion.setLabel("Question # " + this.currQuestionNo);

this.idxOptA = cgAnswer.append(this.questions[this.currQuestionNo-1].getOptionA(), null);

this.idxOptB = cgAnswer.append(this.questions[this.currQuestionNo-1].getOptionB(), null);

this.idxOptC = cgAnswer.append(this.questions[this.currQuestionNo-1].getOptionC(), null);

this.idxOptD = cgAnswer.append(this.questions[this.currQuestionNo-1].getOptionD(), null);

}

* 1. **Check Correct Question**

if (c == cmdSelect) {

String correctAnswer = this.getCorrectAnswer();

if (cgAnswer.getString(cgAnswer.getSelectedIndex()).equals(correctAnswer)){

this.winAmount = this.winAmount \* 2;

if (this.currQuestionNo < 6) {

atStatus.setString("Answer is correct! You win $" + this.winAmount + " \n Moved to next question.");

atStatus.addCommand(cmdNext);

} else {

atStatus.setString("Answer is correct! \n Congratulations! You Won total $" + this.winAmount);

atStatus.addCommand(cmdFinish);

}

atStatus.setCommandListener(this); //set the listner to alert

display.setCurrent(atStatus, d);

} else {

if (this.currQuestionNo > 4) { // atleat win something...

atStatus.setString("Answer is incorrect! \n But you still win $" + this.winAmount);

atStatus.addCommand(cmdFinish);

} else {

atStatus.setString("Answer is incorrect! \n Game Over!");

atStatus.setType(AlertType.ERROR);

atStatus.addCommand(cmdQuit);

}

atStatus.setCommandListener(this); //set the listner to alert

display.setCurrent(atStatus, d);

}

}

* 1. **50/50 Life Line**

private void get5050(){

if (this.questions[this.currQuestionNo-1].getAnswer().equals("A")) {

cgAnswer.delete(idxOptD);

cgAnswer.delete(idxOptB);

}

if (this.questions[this.currQuestionNo-1].getAnswer().equals("B")) {

cgAnswer.delete(idxOptC);

cgAnswer.delete(idxOptA);

}

if (this.questions[this.currQuestionNo-1].getAnswer().equals("C")) {

cgAnswer.delete(idxOptD);

cgAnswer.delete(idxOptB);

}

if (this.questions[this.currQuestionNo-1].getAnswer().equals("D")) {

cgAnswer.delete(idxOptC);

cgAnswer.delete(idxOptA);

}

}

* 1. **Audience Poll Life Line**

private void getAudiencePoll() {

if (this.questions[this.currQuestionNo-1].getAnswer().equals("A")) {

strPollResult.setText("A=75%, B=10%, C=5%, D=10%");

this.append(strPollResult);

}

if (this.questions[this.currQuestionNo-1].getAnswer().equals("B")) {

strPollResult.setText("A=15%, B=70%, C=10%, D=5%");

this.append(strPollResult);

}

if (this.questions[this.currQuestionNo-1].getAnswer().equals("C")) {

strPollResult.setText("A=10%, B=10%, C=60%, D=20%");

this.append(strPollResult);

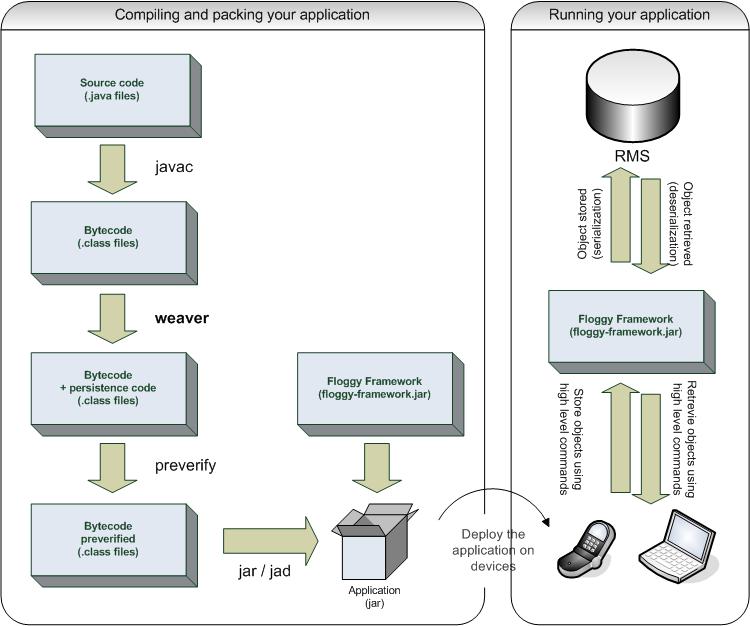
}

1. **Framework for Data Storage**

To store data in RMS a framework called Floggy is used. Floggy is an object persistent framework for J2ME/MIDP applications. It abstracts the data persistence details and reduce the development and maintenance efforts.

Framework website: http://floggy.sourceforge.net

* 1. **Framework Architecture**



Source of diagram: http://floggy.sourceforge.net/index.html

1. **Development Platform and Tools**

Following platforms and tools are used in the application development.

Operating System: Windows 7  
Platform: J2ME  
Programming Language: Java  
DataStore: Record Management System  
Framework: Floggy – for object persistence  
IDE: NetBeans

1. **Application dependencies and Running Instructions**
   1. **Framework**

The application uses Floggy framework for object persistence. To build the application the build.xml file point to the location where Floggy library (.jar) is placed.

In build.xml file the value of following property need to the location where Floggy is installed.

<property name="floggy.path" value="C:\Program Files\Java\floggy-persistence-1.4.0"/>

* 1. **Admin Panel**

To access the admin panel valid admin username and password is required.

Following is the username and password.

Username: admin  
Password: admin

1. **Future Work**

To improve the game application, certain things can be done in future.

* Add more multimedia, like play appropriate sound when correct or wrong answer accordingly.
* Add timer to each question.
* Create play levels like easy, normal, and difficult.
* Save player name with high score.
* Integrate with social networking (like Facebook) to share the player score/win prize.