

**HIGHER COLLEGE OF TECHNOLOGY**

**Department of Information Technology**

**Software Engineering Specialization**

**Electronic Examination Marking (E-Marker)**

In Partial Fulfillment of the Requirements for B.Tech Degree with Specialization in Software Engineering

**Submitted by:**

AmalHamood Al-Jabri: 16j1023

Maryam Said Al-Hashmi: 110S091

SalwaHarib Al- Tobi: 16J09564

**Under the guidance of**

Ronette Sosa Rabanal

Semester 2 / 2014



**HIGHER COLLEGE OF TECHNOLOGY**

**Department of Information Technology**

**BONAFIDE CERTIFICATION**

Certified that the project report “Electronic Examination Marking(E-Marker)” is the bonafide work of “Amal Al-Jabri, Maryam Al-Hashmi, Salwa Al-Tobi” who carried out the project work under my supervision for the partial fulfillment of the requirements for “Bachelor ” with specialization in “ Software Engineering ”.

HEAD OF THE DEPARTMENT SUPERVISOR:

DATE:

**2. Table of Contents**

|  |  |  |
| --- | --- | --- |
| **SL. No** | **Chapter Title** | **Page No.** |
| - | Acknowledgement | 4 |
| - | Abstract | 5 |
| 1 | Introduction | 6 |
| 1.1 | Introduction | 6 |
| 1.2 | Problem Statement | 6 |
| 1.3 | Project Objective | 6 |
| 1.4 | Project Description | 6 |
| 1.5 | Action Plan | 6,7 |
| 2 | System Requirements Specification | 8 |
| 2.1 | Introduction | 8 |
| 2.1.1 | Purpose | 8 |
| 2.1.2 | Definition, Acronym's, Abbreviations | 8 |
| 2.2 | Overall description | 8 |
| 2.2.1 | System Perspective | 8 |
| 2.2.2 | System Features | 8 |
| 2.2.3 | Operating Environment | 9 |
| 2.2.3.1 | Hardware Requirement | 9 |
| 2.2.3.2 | Software Requirement | 9 |
| 2.2.4 | Design/Implementation constraints | 9 |
| 2.2.5 | Assumptions and Dependencies | 10 |
| 2.3 | Specific Requirements | 10 |
| 2.3.1 | Functional Requirements | 10 |
| 2.3.2 | Non-Functional Requirements | 10 |
| 2.3.2.1 | Interface Requirements: | 10 |
| 2.3.2.2 | Performance Requirement | 10 |
| 2.3.2.3 | Life Cycle Requirements | 11 |
| 2.3.2.4 | Reliability requirements | 11 |
| 2.3.2.5 | Security requirements | 11 |
| 2.3.2.6 | Safety requirements | 11 |
| 2.4 | External Interface Requirements | 11 |
| 2.4.1 | User Interface | 11 |
| 2.4.1.1 | Welcome Screen | 12 |
| 2.4.1.2 | Exam View Screen | 13 |
| 2.4.1.3 | Add Exam Screen | 14 |
| 2.4.1.4 | Scan Student Sheet | 15 |
| 2.4.1.5 | Add Student Sheet | 16 |
| 2.4.1.6 | View Exam Marks | 17 |
| 3 | System Design | 18 |
| 3.1 | Introduction | 18 |
| 3.2 | System Design Overview | 18 |
| 3.3 | Application Design Detail | 18 |
| 3.3.1 | Table Structure | 18 |
| 3.3.2 | Data Flow Diagram | 19 |
| 3.3.3 | Flow Chart Diagram | 20 |
| 4 | System Implementation | 21 |
| 4.1 | Implementation | 21 |
| 4.1.1 | System Implementation Description | 21 |
| 4.1.2 | Coding | 21 |
| 4.1.2.1 | Login Validation | 21 |
| 4.1.2.2 | Sign up validation | 22 |
| 4.1.2.3 | Add Exam | 22 |
| 4.1.2.4 | Remove Exam | 23 |
| 4.1.2.5 | Scan Student Sheet | 23 |
| 4.1.2.6 | Add student | 23 |
| 4.1.2.7 | (OMR Test 1.exe) from c# project. | 24-32 |
| 4.1.2.8 | (form1.cs ) Interface in c# project. | 32 |
| 4.1.2.9 | View Exam Marks. | 32 |
| 4.1.2.10 | Serializable Interface supported by Java. | 33-34 |
| 4.1.3 | Validation Checks | 35 |
| 4.2 | Testing | 36 |
| 4.2.1 | Black box testing | 36 |
| 4.2.2 | White box testing | 37 |
| 5 | Conclusion and Future | 38 |
| 5.1 | Summary of Findings | 38 |
| 5.2 | Conclusion | 38 |
| 5.3 | Future Scope | 39 |
| - | References | 40 |

**3. List of Figures**

|  |  |  |
| --- | --- | --- |
| **SL. No** | **Name of the diagram** | **Page No.** |
| 1.1 | Action plan Gantt Chart | 6 |
| 1.2 | Action Plan | 7 |
| 1.3 | Welcome Screen | 12 |
| 1.4 | Exam View screen | 13 |
| 1.5 | Add Exam Screen | 14 |
| 1.6 | Scan student sheet | 15 |
| 1.7 | Add student sheet | 16 |
| 1.8 | View Exam Mark | 17 |
| 1.9 | DFD level-0 | 19 |
| 1.10 | DFD level-1 | 19 |
| 1.11 | Flow chart Diagram | 20 |
| 1.12 | Login-validation check | 35 |
| 1.13 | Sign up- validation check | 35 |
| 1.14 | Unmatched Exam ID-validation check | 36 |

**4. List of Tables**

|  |  |  |
| --- | --- | --- |
| **SL. No** | **Name of the table** | **Page No.** |
| 1.1 | Teacher table | 18 |
| 1.2 | Black box-testing table | 34-35 |

**Acknowledgment**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to Ms. Ronette Sosa Rabanalfor her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completing the project. We would like to express our gratitude towards to E-learning administration staff Mr.Nadeem for his kind co-operation and encouragement which helped us in the completion of this project. Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

**Abstract**

This report is about Electronic Examination Marking (E-marker). E-marker is an application used for checking multiple choice questions and generates the scored mark. Teachers will be allowed to enter he correct options and scan the student answer sheet. E-marker application will compare the student answers with the correct answers and generates scored mark.

The reason of us creating this application because the teachers had problems in checking the student papers since they have to go through the questions several times to make sure of their correction. The main Objective of our application is to reduce the lectures time and effort.

**Chapter 1: Introduction.**

**1.1 Introduction:**

**1.2 Problem Statement**:

New technologies have been created to make life easy. In education environment, the teachers face some difficulties to correct some questions like multiple choices and true/false questions. They have to go through the questions several times to make sure of their correction, as long as they will correct each paper manually, a lot of time and effort will be needed.

**1.3 project Objectives:**

New generation creates new things. People nowadays prefer using electronic devices due to the ease of use and time consuming. Electronic Examination Marking(E-Marker) will save lectures effort and time on marking students examination sheets.

**1.4 Project Description:**

E-Marker application is developed for marking exam sheets. E-marker will allow the user to enter the correct options and then scan the student answer sheet. Then the scored mark of the student will be generated based on the correctness of the selected options. In addition E-marker application saves all teacher works under his account.

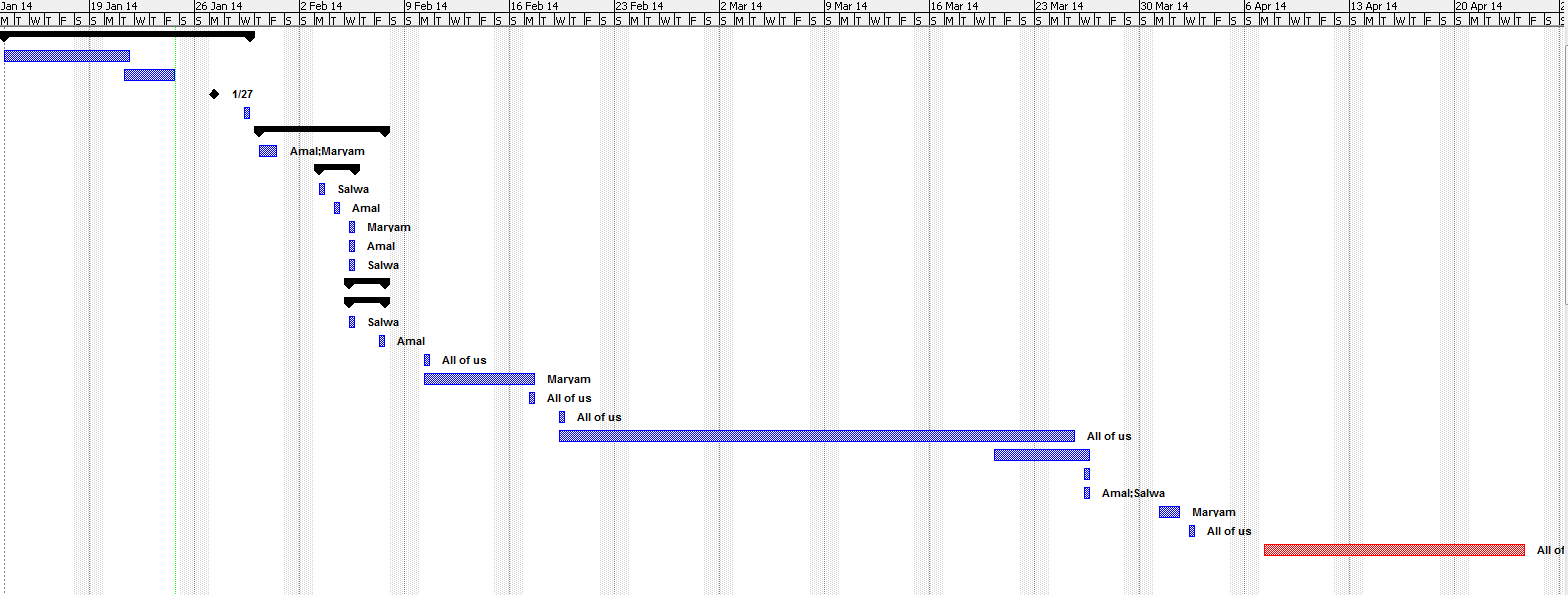
**1.5 Action Plan:**

Figure1.1 Action Plan Gantt Chart

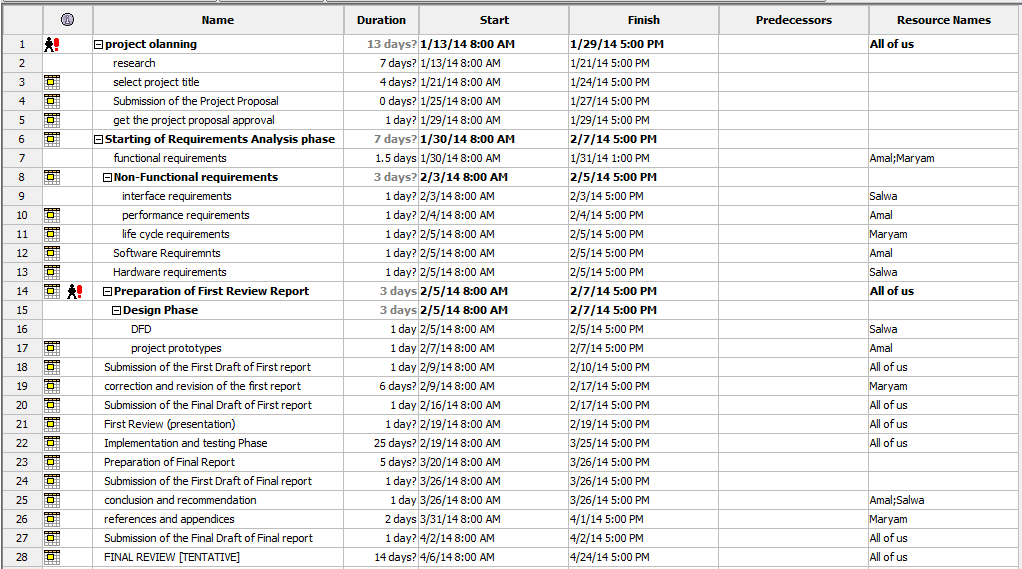


Figure 1.2: Action Plan

**Chapter2: System Requirement Specification**

**2.1 Introduction:**

**2.1.1 Purpose:**

The purpose of this Software Requirement Specification document is to provide a detailed overview of the provided software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the product and its functionality.

**2.1.2 Definition, Acronym's, Abbreviations:**

* **User**: the person who interact with the system(teacher).
* **E-marker**: the name of the application.
* **Student answer sheet**: the student's answer paper.
* **Administrator**: who will be responsible of administering the system data and the authority to access the application.

**2.2 Overall Description:**

**2.2.1 System Perspective:**

E-marker is a Java application which is developed for marking exam sheet. Teacher will provide the correct options of the exam and will scan the student answer sheet. Comparison functionality will be implemented to compare the student shaded options with the correct options of the exam. Later on, the application will calculate student’s score mark and generate the list of student names and their mark it to inform the teacher.

**2.2.2 System features:**

* The user must log in to the system using his user name and password.
* The system will automatically generate random passwords for new users.
* The user can add many exams wanted.
* The student can take more than one exam for the specific subject.
* The system allows the teacher to enter the exam correct options.
* The system saves all teacher's exams under the working account.
* The system calls the scanner window to scan the required sheets.
* A progress bar is shown when the scanning is processed.
* The system shows a list of student names with their marks.
* There is no specific quality for scanned papers as long as the shaded options are clear.
* There is no images quality needed.
* The system will be able to accept images that have been taken with any suitable devices (phones, cameras… etc.).
* The system is safe.
* The system is easy to use.
* The system is fast.

**2.2.3 Operating environment:**

This program will operate in different types of operating systems such as: Windows, Apple Mac OSX, Linux and Solaris. As long as they have the Java Virtual Machine (JVM) on the system. They will be able to run the application.

**2.2.3.1 Hardware Requirement**:

1. Scanner:Canon CanoScanLiDe 110, USB connection.
2. Personal Computer:Laptop: Intel Core i5-3317U Processor, 4 GB DDR3L SDRAM, 500 GB Hard Drive.

**2.2.3.2 Software Requirement:**

1. Visual Studio and Eclipse: Software tools which are required for the interface design phase and the implementation phase for the application.
2. SqlLite: Will be used as a database for the application.
3. Java Development Kit: Used to compile and debug.
4. Java Virtual Machine: Used to run the application.
5. Canon canoScanLiDe 110 software driver: To get the executable file to run the scanning window.

**2.2.4 Design/ Implementation Constraint:**

1. The system must be connected to the scanner to scan the sheets.
2. The system must be connected to the database to store the username and password.
3. The user should first scan the student sheet and then browse the scanned picture from its location.

**2.2.5 Assumption and dependencies:**

1. The system will generate score mark without decimal points.
2. The system will detect the shaded options in case the student shaded two options at a time, a reduced mark will be given for the particular question.
3. The Exam ID that’s given from the teacher must match the Exam ID written in student answer sheet.
4. The teacher must enter the exact image path of student answer sheet along with the correct teacher name(username account) and student name, if any mismatched values then an error will be generated.

**2.3 Specific Requirements**

**2.3.1 Functional Requirements:**

* User should be able to login in by username and password.
* The system should be able to generate random passwords for new users.
* User should be able to add/view/delete exams.
* The system should be able to call the auto-scanner window.
* The system should be able to scan student answer sheet.
* The system should be able to compare between student options and the exam correct options.
* User should be able to view student scored mark.
* The system must be able to save all User’s works under the user account.

**2.3.2 Non-Functional requirements:**

**2.3.2.1 Interface Requirements:**

* The system should be user friendly.

**2.3.2.2 Performance Requirements:**

* The system should be able to react for each user click within maximum 1 seconds.
* The system should be available for service when requested by end-users.

**2.3.2.3 Life Cycle requirements:**

* The system should be maintainable.
* The system should be portable across different computer system.

**2.3.2.4 Reliability requirements:**

* The system shall have strategy for error detection and correction.
* The system should be able to handle any failure occurrence.

**2.3.2.5 Security requirements:**

* The system should ensure the integrity and components from accidental or malicious damage.
* The access permissions for system data may only be changed by the systems data administrator.

**2.3.2.6 Safety requirements:**

* The system should no longer operate if security attacks have become obvious.
* The system should no longer operate in case of fire.

**2.4 External Interface Requirements**

**2.4.1 User Interfaces**

This section will provide the description of logical characteristics of each interface between the system and the users. The following are the screens of E-marker application.

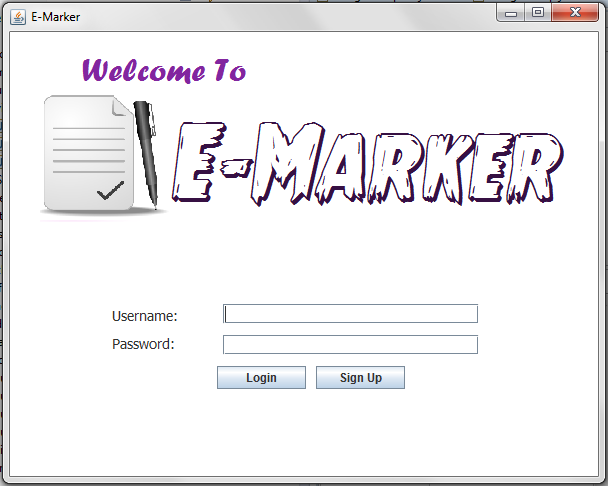


Figure 1.3 Welcome Screen.

**2.4.1.1 Welcome screen :**

The startup page of the application where the user name and password must be provided by the teacher to use E-marker application. New users can register to the application by clicking the Sign Up button.

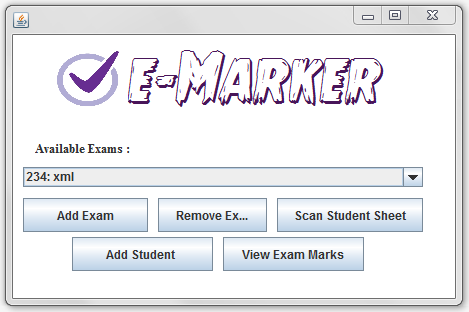


Figure 1.4 Exam View

**2.4.1.2 Exam View Screen:**

This screen will pop up once the login process is successful. The teacher must add new exam with the exam name and exam ID. A unique Exam ID must be entered for each exam added and it should match the student answer sheet Exam ID for the correction. Fill-in form will be provided for the teacher to fill the exam correct options which later will be compared with the student answers. They can also delete unwanted exams from the dropdown list.

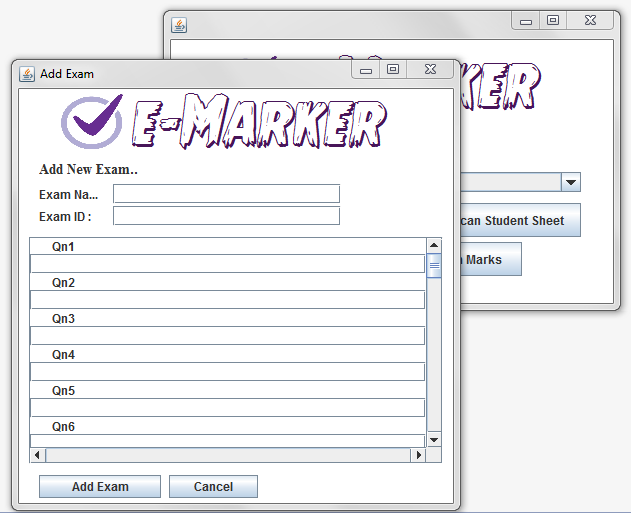
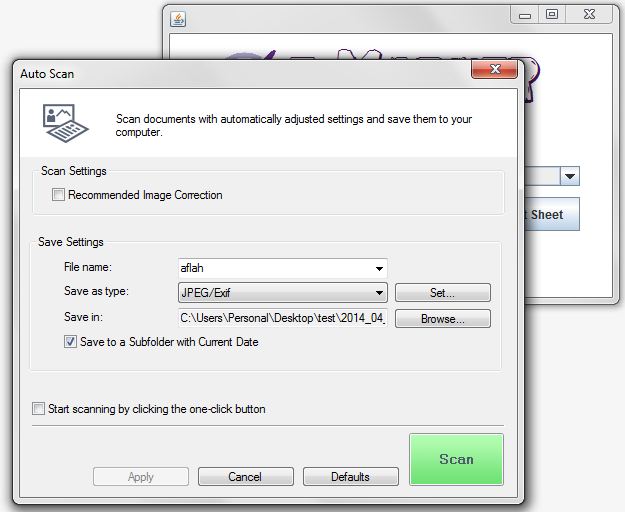


Figure 1.5 Add Exam Screen

**2.4.1.3 Add Exam Screen :**

This screen will pop up to the User once (add Exam button) is clicked. The User must enter the Exam name, Exam ID and the correct options of the particular exam. Exam correct options will be saved in array for the comparison. If the user skipped one question the array will break and take the total number of questions as the maximum score mark of the particular exam. Teacher can add as many as exams as he wants but with unique Exam ID.

Figure 1.6 Scan Student Sheet

**2.4.1.4 Scan Student Sheet :**

E-marker application calls the executable file of the scanner to allow the users to scan student sheets. Users can specify the path where to save the student scanned sheet.

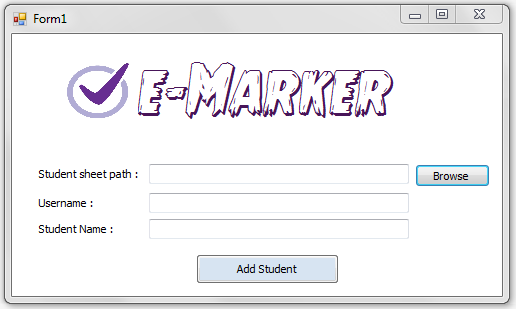


Figure 1.7 Add Student sheet

**2.4.1.5 Add Student Sheet :**

Once the (add student button) is clicked from the (Exam View screen) the application will allow the user to browse the scanned sheet. User must enter username and student Name and click add student. The application will read the shaded options from the particular student sheet selected by the user. If Exam ID of the student sheet does not match with the Exam ID of the added exam then the application will stop the process of calculating scored mark.

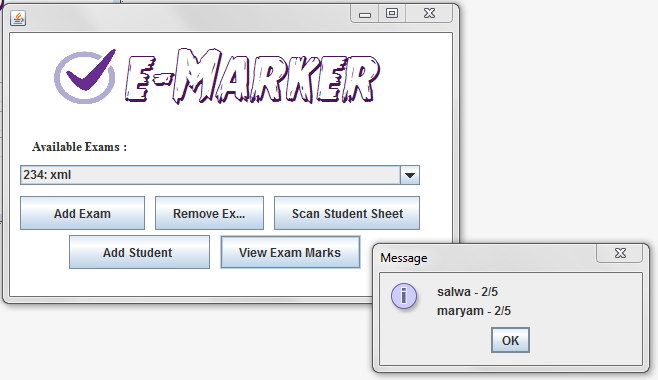


Figure 1.8 View Marks.

**2.4.1.6 View Marks:**

Once the process of reading the student options and calculating the scored mark is done. User can view the scored mark of student by clicking (view Exam Marks) button. As much as the teacher correct exam of multiple students ,all the student names and marks will be listed down and saved under the user account .

**Chapter3: SYSTEM DESIGN**

* 1. **Introduction**

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces. This document will be produced with a brief description of the Electronic Examination Marking system.

* 1. **System Design Overview**

Electronic Examination Marking (E-marker) is an application which allows user to scan student answer sheet. The application will process the comparison between student options and the exam correct options then generates student scored mark. The following displays show the structure of E-marker application.

**3.3 Application Design Detail**

**3.3.1 Table Structure**

i. Table Name: Teacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Constraints** | **Description** |
| Teacher\_ID | Number | 4 | Not Null | Primary Key |
| Username | Character | 30 | Not Null |  |
| Password | Number | 10 | Not Null |  |

1.1 : Teacher table

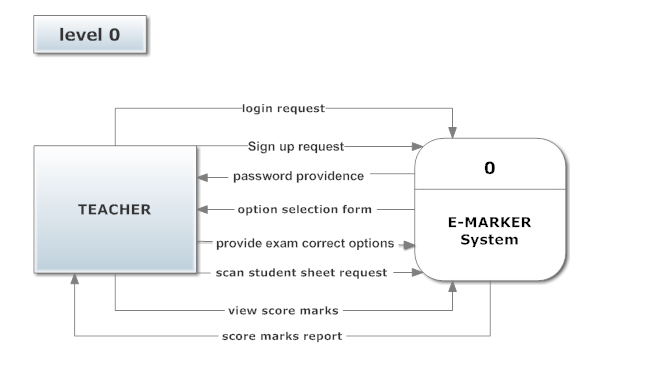
**3.3.2 Data Flow Diagrams**

Figure 1.9 : DFD level-0

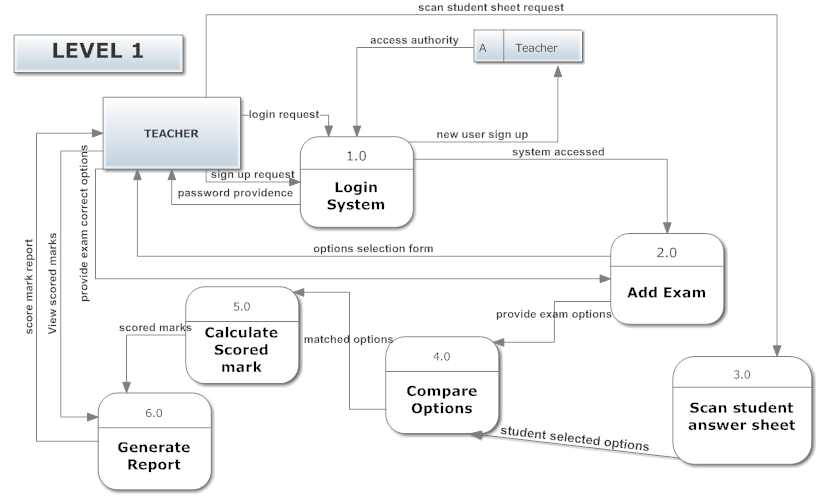


Figure 1.10 : DFD level-1

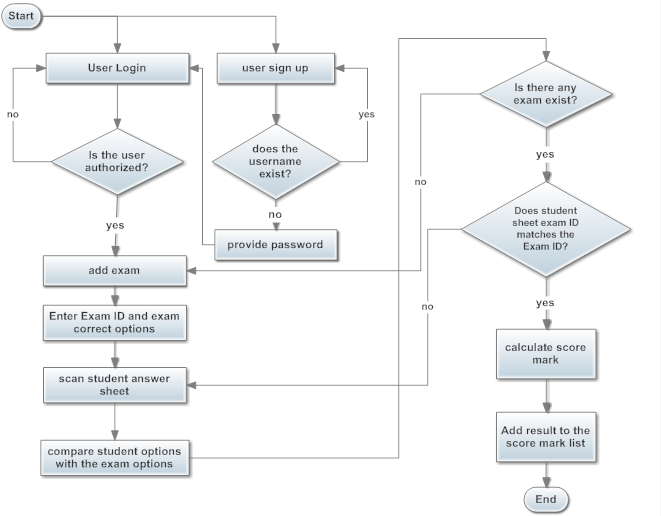
**3.3.3 Flow chart Diagram**

Figure 1.11 : Flow Chart Diagram

**Chapter 4: SYSTEM IMPLEMENTATION**

**4.1 Implementation**

**4.1.1 System Implementation Description**

At first, the project requirements and design was converted to sketches that explains how the final application would look like and at the same time some flow lines were shown from one screen to another to give a mental image of how the required system would look like and its functionality.

The first major step was the object-oriented design(OOD). It was to convert every element in the project to a model class by defining its instance variables and methods. The second major step was creating the Database Adapter for the SQLite database. This class handled all the communications to the database. The Adapter had methods for performing Create, Read, Update, and Delete operations (CRUD). The third step was designing the views and screens for the application.Last step, the actual coding and functionality was made, this is where the threads, connections, functionalities, and everything else was written and put together to complete the system requirements in a right manner.

**4.1.2 Coding**

**4.1.2.1 Login Validation**

JButton btnLogin = **new** JButton("Login");

btnLogin.setFont(**new** Font("Times New Roman", Font.*PLAIN*, 14));

btnLogin.addActionListener(**new** ActionListener() {

@SuppressWarnings("deprecation")

**public** **void** actionPerformed(ActionEvent arg0) {

**int** count=0;

**if**(Db.*userExists*(txtUser.getText(), passwordField.~~getText~~())) {

// user exists

Teacher t = **new** Teacher(txtUser.getText());

**new** ExamView(t).setVisible(**true**);

dispose();

}

**else** {

//user doesn't exist

JOptionPane.*showMessageDialog*(**null**, "User not found");

count++;

**if**(count==3){

System.*exit*(0);

}

}

}

});

**4.1.2.2 Sign up validation**

btnLogin.setBounds(205, 86, 101, 23);

panel.add(btnLogin);

JButton btnSignUp = **new** JButton("Sign Up");

btnSignUp.setFont(**new** Font("Times New Roman", Font.*PLAIN*, 14));

btnSignUp.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

String username = JOptionPane.*showInputDialog*("Enter your username:");

**if**(!Db.*isTeacherNameExists*(username)) {

**int** password = (**int**)(Math.*random*()\*99999);

JOptionPane.*showMessageDialog*(**null**, "Your password is: "+String.*valueOf*(password));

Db.*addTeacher*(username, String.*valueOf*(password));

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "Username already used.");

}

}

});

**4.1.2.3 Add Exam**

JButton btnAddExam = **new** JButton("Add Exam");

btnAddExam.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

// **TODO**: Validation and number format validation

**int** examId = Integer.*parseInt*(textField\_1.getText());

String examName = textField.getText();

Exam exam = **new** Exam(examId, examName);

ArrayList<Integer> choices = **new** ArrayList<Integer>();

**for**(**int** i=0; i<40; i++) {

String x = questionFields.get(i).getText();

**if**(x.length() == 0) {

**break**;

}

choices.add(Integer.*parseInt*(x));

}

exam.getKeySheet().setChoices(choices);

teacher.getExams().add(exam);

teacher.saveToFile();

**new** ExamView(teacher).setVisible(**true**);

dispose();

}

});

**4.1.2.4 Remove Exam**

JButton buttonRemove = **new** JButton("Remove Exam");

buttonRemove.setBounds(145, 163, 109, 34);

buttonRemove.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

teacher.getExams().remove(comboBox.getSelectedIndex());

teacher.saveToFile(); // save to teahcer file

updateComboBox();

}

});

**4.1.2.5 Scan Student Sheet**

The following codes is used to call the executable file of the scanner.

JButton buttonView = **new** JButton("Scan Student Sheet");

buttonView.setBounds(264, 163, 146, 34);

buttonView.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try** {

Runtime.*getRuntime*().exec("C:\\Program Files (x86)\\Canon\\MP Navigator EX 4.0\\mpnex40.exe");

} **catch** (IOException ee) {

ee.printStackTrace();

}

}

});

**4.1.2.6 Add student**

JButton btnAddResult = **new** JButton("Add Student ");

btnAddResult.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

**try** {

//call the executable file of c# project

Runtime.*getRuntime*().exec("OMR Test 1.exe");

} **catch**(Exception e) {}

System.*exit*(0);

}

});

**4.1.2.7 (OMR Test 1.exe) from c# project.**

this codes will be responsible of reading the options in student answer sheet along with reading Exam ID and pass it to E-marker.jar

namespace OMRReader\_test1

{

public class OMR

{

System.Drawing.Image scannedImage;

public void action(string teacher, string student, string imagePath)

{

/// Get scan image and resize it

scannedImage = System.Drawing.Image.FromFile(imagePath);

scannedImage = (System.Drawing.Image)ImageUtilities.ResizeImage((Bitmap)scannedImage, 2100, 2100 \* scannedImage.Height / scannedImage.Width);

Bitmap unf = new Bitmap(scannedImage);

OpticalReader reader = new OpticalReader();

scannedImage = (System.Drawing.Image)reader.ExtractOMRSheet(unf, "sheets.xml", OMREnums.OMRSheet.A550);

/// Get user selected options

Rectangle[] Blocks = new Rectangle[]

{

OMRSheetReader.GetSheetPropertyLocation("sheets", OMREnums.OMRSheet.A550, OMREnums.OMRProperty.tensBlock1),

OMRSheetReader.GetSheetPropertyLocation("sheets", OMREnums.OMRSheet.A550, OMREnums.OMRProperty.tensBlock2),

OMRSheetReader.GetSheetPropertyLocation("sheets", OMREnums.OMRSheet.A550, OMREnums.OMRProperty.tensBlock3),

OMRSheetReader.GetSheetPropertyLocation("sheets", OMREnums.OMRSheet.A550, OMREnums.OMRProperty.tensBlock4)

};

List<Bitmap[]> bmps = new List<Bitmap[]>();

for (int i = 0; i < 4; i++)

{

bmps.Add(SliceOMarkBlock(scannedImage, Blocks[i], 10));

}

string ans = "";

foreach (Bitmap[] blk in bmps)

{

foreach (Bitmap line in blk)

{

ans += rateSlice(line, 5) + ",";

}

}

Console.WriteLine(ans);

/// Get exam ID

OpticalReader rr = new OpticalReader();

int examId = Convert.ToInt32(rr.getRegNumOfSheet(scannedImage, OMREnums.OMRSheet.A550, "sheets.xml", false).ToString());

Console.WriteLine(examId);

Console.Read();

/// Return to Java application

string jarPath = "E-Marker.jar";

string argumentsFortheJarFile = "\"" + teacher + "\" \"" + student + "\" " + examId + " " + ans;

System.Diagnostics.Process clientProcess = new Process();

clientProcess.StartInfo.FileName = "java";

clientProcess.StartInfo.Arguments = @"-jar "+ jarPath +" " + argumentsFortheJarFile;

clientProcess.Start();

clientProcess.WaitForExit();

}

private Bitmap ExtractOMRSheet(Bitmap basicImage, int fillint, int contint, string OMRSpecsSheetAddress)

{

scannedImage = (System.Drawing.Image)flatten(basicImage, fillint, contint);

Application.DoEvents();

//return ExtractPaperFromFlattened(new Bitmap(scannedImage), basicImage, minblb\_tb.Value, fillint, contint, OMRSpecsSheetAddress);

return ExtractPaperFromFlattened(new Bitmap(scannedImage), basicImage, 0, fillint, contint, OMRSpecsSheetAddress);

}

public Bitmap flatten(Bitmap bmp, int fillint, int contint)

{

// step 1 - turn background to black

ColorFiltering colorFilter = new ColorFiltering();

colorFilter.Red = new IntRange(0, fillint);

colorFilter.Green = new IntRange(0, fillint);

colorFilter.Blue = new IntRange(0, fillint);

colorFilter.FillOutsideRange = false;

colorFilter.ApplyInPlace(bmp);

AForge.Imaging.Filters.ContrastCorrection Contrast = new ContrastCorrection(contint);

AForge.Imaging.Filters.Invert invert = new Invert();

AForge.Imaging.Filters.ExtractChannel extract\_channel = new ExtractChannel(0);

//AForge.Imaging.Filters.Threshold thresh\_hold = new Threshold(thresh\_tb.Value);

AForge.Imaging.Filters.Threshold thresh\_hold = new Threshold(0);

bmp = invert.Apply(thresh\_hold.Apply(extract\_channel.Apply(Contrast.Apply(bmp))));

return bmp;

}

private bool isSame(UnmanagedImage img1, UnmanagedImage img2)

{

int count = 0, tcount = img2.Width \* img2.Height;

for (int y = 0; y < img1.Height; y++)

for (int x = 0; x < img1.Width; x++)

{

Color c1 = img1.GetPixel(x, y), c2 = img2.GetPixel(x, y);

if ((c1.R + c1.G + c1.B) / 3 > (c2.R + c2.G + c2.B) / 3 - 10 &&

(c1.R + c1.G + c1.B) / 3 < (c2.R + c2.G + c2.B) / 3 + 10)

count++;

}

//return (count \* 100) / tcount >= sim\_tb.Value;

return (count \* 100) / tcount >= 1;

}

private Bitmap ExtractPaperFromFlattened(Bitmap bitmap, Bitmap basicImage, int minBlobWidHei, int fillint, int contint, string OMRSheets)

{

BitmapData bitmapData = bitmap.LockBits(

new Rectangle(0, 0, bitmap.Width, bitmap.Height),

ImageLockMode.ReadWrite, bitmap.PixelFormat);

// lock image

// step 2 - locating objects

BlobCounter blobCounter = new BlobCounter();

blobCounter.FilterBlobs = true;

blobCounter.MinHeight = minBlobWidHei;

blobCounter.MinWidth = minBlobWidHei;

blobCounter.ProcessImage(bitmapData);

Blob[] blobs = blobCounter.GetObjectsInformation();

bitmap.UnlockBits(bitmapData);

Graphics g = Graphics.FromImage(bitmap);

Pen yellowPen = new Pen(Color.Yellow, 2); // circles

Pen redPen = new Pen(Color.Red, 2); // quadrilateral

Pen brownPen = new Pen(Color.Brown, 2); // quadrilateral with known sub-type

Pen greenPen = new Pen(Color.Green, 2); // known triangle

Pen bluePen = new Pen(Color.Blue, 2); // triangle

Rectangle[] rects = blobCounter.GetObjectsRectangles();

Blob[] blobs2 = blobCounter.GetObjects(bitmap, false);

System.Drawing.Image compImg = System.Drawing.Image.FromFile("lc.jpg");

UnmanagedImage compUMImg = UnmanagedImage.FromManagedImage((Bitmap)compImg);

List<IntPoint> quad = new List<IntPoint>();

try

{

//g.DrawRectangles(yellowPen, rects);

foreach (Blob blob in blobs2)

{

if (

((double)blob.Area) / ((double)bitmap.Width \* bitmap.Height) > 0.0001 &&

((double)blob.Area) / ((double)bitmap.Width \* bitmap.Height) < 0.005 &&

blob.Rectangle.X < (bitmap.Width) / 4)

{

if ((double)blob.Rectangle.Width / blob.Rectangle.Height < 1.4 &&

(double)blob.Rectangle.Width / blob.Rectangle.Height > .6)

{

compUMImg = UnmanagedImage.FromManagedImage(ImageUtilities.ResizeImage(compImg, blob.Rectangle.Width, blob.Rectangle.Height));

if (isSame(blob.Image, compUMImg))

{

g.DrawRectangle(yellowPen, blob.Rectangle);

quad.Add(new IntPoint((int)blob.CenterOfGravity.X, (int)blob.CenterOfGravity.Y));

}

}

}

}

}

catch (ArgumentException) { MessageBox.Show("No Blobs"); }

try

{

if (quad[0].Y > quad[1].Y)

{

IntPoint tp = quad[0];

quad[0] = quad[1];

quad[1] = tp;

}

}

catch

{

}

compImg = System.Drawing.Image.FromFile("rc.jpg");

compUMImg = UnmanagedImage.FromManagedImage((Bitmap)compImg);

try

{

//g.DrawRectangles(yellowPen, rects);

foreach (Blob blob in blobs2)

{

if (

((double)blob.Area) / ((double)bitmap.Width \* bitmap.Height) > 0.0001 &&

((double)blob.Area) / ((double)bitmap.Width \* bitmap.Height) < 0.004 &&

blob.Rectangle.X > (bitmap.Width \* 3) / 4)

{

if ((double)blob.Rectangle.Width / blob.Rectangle.Height < 1.4 &&

(double)blob.Rectangle.Width / blob.Rectangle.Height > .6)

{

compUMImg = UnmanagedImage.FromManagedImage(ImageUtilities.ResizeImage(compImg, blob.Rectangle.Width, blob.Rectangle.Height));

if (isSame(blob.Image, compUMImg))

{

g.DrawRectangle(yellowPen, blob.Rectangle);

quad.Add(new IntPoint((int)blob.CenterOfGravity.X, (int)blob.CenterOfGravity.Y));

}

}

}

}

}

catch (ArgumentException) { MessageBox.Show("No Blobs"); }

try

{

if (quad[2].Y < quad[3].Y)

{

IntPoint tp = quad[2];

quad[2] = quad[3];

quad[3] = tp;

}

}

catch

{

}

yellowPen.Dispose();

redPen.Dispose();

greenPen.Dispose();

bluePen.Dispose();

brownPen.Dispose();

g.Dispose();

//// put new image to clipboard

//Clipboard.SetDataObject(bitmap);

// and to picture box

if (quad.Count == 4)

{

if (((double)quad[1].Y - (double)quad[0].Y) / ((double)quad[2].Y - (double)quad[3].Y) < .75 ||

((double)quad[1].Y - (double)quad[0].Y) / ((double)quad[2].Y - (double)quad[3].Y) > 1.25)

quad.Clear();

else if (quad[0].X > bitmap.Width / 2 || quad[1].X > bitmap.Width / 2 || quad[2].X < bitmap.Width / 2 || quad[3].X < bitmap.Width / 2)

quad.Clear();

}

if (quad.Count != 4)

{

if (contint <= 60)

{

if (contint >= 0)

{

contint += 5;

contint \*= -1;

return ExtractOMRSheet(basicImage, fillint, contint, OMRSheets);

}

else

{

contint \*= -1;

contint += 10;

return ExtractOMRSheet(basicImage, fillint, contint, OMRSheets);

}

}

else

{

MessageBox.Show("Extraction Failed.");

return basicImage;

}

}

else

{

IntPoint tp2 = quad[3];

quad[3] = quad[1];

quad[1] = tp2;

QuadrilateralTransformation wrap = new QuadrilateralTransformation(quad);

wrap.UseInterpolation = false;

Rectangle sr = OMRSheetReader.GetSheetPropertyLocation(OMRSheets, OMREnums.OMRSheet.A550, OMREnums.OMRProperty.SheetSize);

wrap.AutomaticSizeCalculaton = false;

wrap.NewWidth = sr.Width;

wrap.NewHeight = sr.Height;

wrap.Apply(basicImage).Save("LastImg.jpg", ImageFormat.Jpeg);

System.Drawing.Image imgl = (System.Drawing.Image)wrap.Apply(basicImage);

Graphics gg = Graphics.FromImage(imgl);

Pen pr = new Pen(Brushes.Red, 2);

pr.Dispose();

gg.Dispose();

return (Bitmap)imgl;

}

}

private System.Drawing.Point[] ToPointsArray(List<IntPoint> points)

{

System.Drawing.Point[] array = new System.Drawing.Point[points.Count];

for (int i = 0, n = points.Count; i < n; i++)

{

array[i] = new System.Drawing.Point(points[i].X, points[i].Y);

}

return array;

}

private List<System.Drawing.Point> afPointListToSystem(List<IntPoint> points)

{

List<System.Drawing.Point> list\_ = new List<System.Drawing.Point>();

for (int i = 0, n = points.Count; i < n; i++)

{

list\_.Add(new System.Drawing.Point(points[i].X, points[i].Y));

}

return list\_;

}

private int rateSlice(Bitmap slice, int OMCount)

{

Rectangle[] cropRects = new Rectangle[OMCount];

Bitmap[] marks = new Bitmap[OMCount];

for (int i = 0; i < OMCount; i++)

{

cropRects[i] = new Rectangle(i \* slice.Width / OMCount, 0, slice.Width / OMCount, slice.Height);

}

int crsr = 0;

foreach (Rectangle cropRect in cropRects)

{

Bitmap target = new Bitmap(cropRect.Width, cropRect.Height);

using (Graphics g = Graphics.FromImage(target))

{

g.DrawImage(slice, new Rectangle(0, 0, target.Width, target.Height),

cropRect,

GraphicsUnit.Pixel);

}

marks[crsr] = target;

crsr++;

}

long maxPD = (slice.Width / OMCount) \* slice.Height \* 255;

List<long> inks = new List<long>();

List<long> fullInks = new List<long>();

foreach (Bitmap mark in marks)

{

inks.Add(InkDarkness(mark));

fullInks.Add(inks[inks.Count - 1]);

}

int indofMx = -1, indofMn = -1;

long maxD = 0, minD = 0; ;

for (int i = 0; i < OMCount; i++)

{

if (inks[i] > maxD)

{

maxD = inks[i];

indofMx = i;

}

}

minD = maxD;

for (int i = 0; i < OMCount; i++)

{

if (inks[i] < minD)

{

minD = inks[i];

indofMn = i;

}

}

for (int i = 0; i < OMCount; i++)

{

inks[i] -= minD - 1;

}

bool parallelExist = false, spe = false, tpe = false, fpe = false;

for (int i = 0; i < OMCount; i++)

{

if (i != indofMx)

{

if ((double)fullInks[indofMx] / fullInks[i] <= 2)

{

if (tpe) fpe = true;

if (spe) tpe = true;

if (parallelExist) spe = true;

parallelExist = true;

}

}

}

int negScore = parallelExist ? -1 : 0;

negScore = spe ? -2 : negScore;

negScore = tpe ? -3 : negScore;

negScore = fpe ? -4 : negScore;

if (!parallelExist)

return indofMx + 1;

bool atleastOneUnfilled = false;

for (int i = 0; i < OMCount; i++)

{

if (i != indofMx)

{

if ((double)fullInks[indofMx] / fullInks[i] >= 3)

atleastOneUnfilled = true;

}

}

if (atleastOneUnfilled)

return negScore;

return 0;

}

private long InkDarkness(Bitmap OMark)

{

int darkestC = 255, lightestC = 0;

UnmanagedImage mark = UnmanagedImage.FromManagedImage(OMark);

for (int y = 0; y < OMark.Height; y++)

for (int x = 0; x < OMark.Width; x++)

{

Color c = mark.GetPixel(x, y);

if (((c.R + c.G + c.B) / 3) > lightestC)

{

lightestC = ((c.R + c.G + c.B) / 3);

}

if (((c.R + c.G + c.B) / 3) < darkestC)

{

darkestC = ((c.R + c.G + c.B) / 3);

}

}

int dc = 0;

for (int y = 0; y < OMark.Height; y++)

for (int x = 0; x < OMark.Width; x++)

{

Color c = mark.GetPixel(x, y);

if (((c.R + c.G + c.B) / 3) < (lightestC + darkestC) / 2)

{ dc += 255; }

}

return dc;

}

private Bitmap[] SliceOMarkBlock(System.Drawing.Image fullSheet, Rectangle slicer, int slices)

{

List<Rectangle> cropRects = new List<Rectangle>();

Bitmap[] bmps = new Bitmap[slices];

for (int i = 0; i < slices; i++)

{

cropRects.Add(new Rectangle(slicer.X, slicer.Y + (slicer.Height / slices) \* i, slicer.Width, slicer.Height / slices));

}

Bitmap src = (Bitmap)fullSheet;

int crsr = 0;

foreach (Rectangle cropRect in cropRects)

{

Bitmap target = new Bitmap(cropRect.Width, cropRect.Height);

using (Graphics g = Graphics.FromImage(target))

{

g.DrawImage(src, new Rectangle(0, 0, target.Width, target.Height),

cropRect,

GraphicsUnit.Pixel);

}

bmps[crsr] = target;

crsr++;

}

return bmps;

throw new Exception("Couldn't slice");

}

}

}

**4.1.2.8 (form1.cs ) Interface in C# project.**

namespace OMRReader\_test1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string path;

OpenFileDialog file = new OpenFileDialog();

if (file.ShowDialog() == DialogResult.OK)

{

path = file.FileName;

textBox1.Text = path;

}

}

private void button2\_Click(object sender, EventArgs e)

{

OMR omr = new OMR();

omr.action(textBox2.Text, textBox3.Text, textBox1.Text);

Application.Exit();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

**4.1.2.9 View Exam Marks.**

JButton btnViewExam = **new** JButton("View Exam Marks");

btnViewExam.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

StringBuilder sb = **new** StringBuilder();

**int** index = comboBox.getSelectedIndex();

Exam exam = teacher.getExams().get(index);

KeySheet ks = exam.getKeySheet();

ArrayList<Integer> ksChoices = ks.getChoices();

**int** maxScore = ksChoices.size();

ArrayList<AnswerSheet> answerSheets = exam.getAnswerSheets();

**for**(AnswerSheet as: answerSheets) {

**int** score = 0;

ArrayList<Integer> asChoices = as.getChoices();

**int** i = 0;

**for**(**int** ch: ksChoices) {

**if**(ch == asChoices.get(i)) {

score++;

}

i++;

}

sb.append(as.getStudentName());

sb.append(" - ");

sb.append(score);

sb.append("/");

sb.append(maxScore);

sb.append("\n");

}

JOptionPane.*showMessageDialog*(**null**, sb.toString());

}

});

**4.1.2.10 Serializable Interface supported by Java.**

This interface support to functions save to file and load from file. With the help of this codes the teacher object will be saved in a file with an extension of (.ser) where the file will be contain all the objects related to the object teacher. This Interface allows us to load from our file and view all the saved exams related to a particular username.

**public** **class** Teacher **implements** Serializable {

**public** **static** String *FILE\_NAME*;

**private** String username;

**private** ArrayList<Exam> exams;

**public** Teacher(String username) {

exams = **new** ArrayList<Exam>();

*FILE\_NAME* = username+".ser";

loadFromFile();

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getUsername() {

**return** **this**.username;

}

**public** **void** setExams(ArrayList<Exam> exams) {

**this**.exams = exams;

}

**public** ArrayList<Exam> getExams() {

**return** **this**.exams;

}

**public** **void** saveToFile() {

**try** {

FileOutputStream fileOut = **new** FileOutputStream(*FILE\_NAME*);

ObjectOutputStream out = **new** ObjectOutputStream(fileOut);

out.writeObject(**this**);

out.close();

fileOut.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** loadFromFile() {

File file = **new** File(*FILE\_NAME*);

**if** (!file.exists())

**return**; // Exit if file doesn't exist

Teacher teacher = **null**;

**try** {

FileInputStream fileIn = **new** FileInputStream(*FILE\_NAME*);

ObjectInputStream in = **new** ObjectInputStream(fileIn);

teacher = (Teacher) in.readObject(); // convert object to exam

in.close();

fileIn.close();

} **catch** (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

// Set values from file to this object

**this**.setUsername(teacher.getUsername());

**this**.setExams(teacher.getExams());

}

**public** **void** deleteFile() {

File file = **new** File(*FILE\_NAME*);

file.delete();

}

**public** **boolean** fileExists() {

File file = **new** File(*FILE\_NAME*);

**if**(file.exists()) **return** **true**;

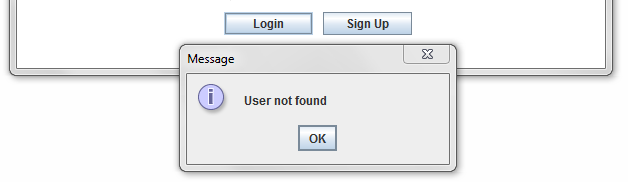
**else** **return** **false**;

}

}

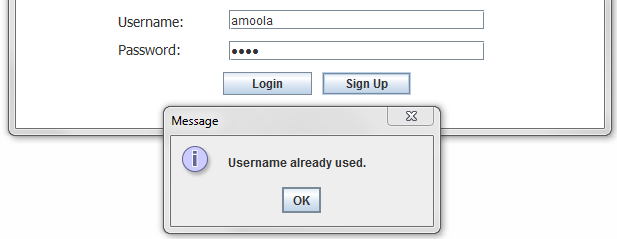
**4.1.3 Validation Checks**

E-marker application has several validation checks. The first validation is to check the correctness of username and password from the database, a message box will appear if the login is unsuccessfully as shown below:



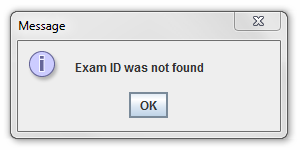
1.12 : Login-validation check

The second validation is to check if the username exist in the database, if it exist then the sign up cannot be processed unless the user provide un existing username. a message box will be popped up to confirm the validation:



1.13 : Sign up- validation check

The third validation is to check whether student sheet image has an Exam ID that matches with the Exam ID of the exam. If there is mismatching then the application generates an error message as shown :



1.14 : Unmatched Exam ID- validation check

**4.2 Testing**

**4.2.1 Black Box Testing:**

We first started by doing Black Box testing. We prepared multiple test cases and expected output for each test case. Here's some of our Black Box test cases:

|  |  |
| --- | --- |
| **Black Box Test case** | **Expected output** |
| * If the teacher skipped one text box while entering the exam correct options. * If the teacher browse invalid picture. * If the teacher browse student sheet image which has unmatched Exam ID with the particular Exam ID needed. * If student shaded 2 or more options in the same question. * If student left an empty option. * If teacher took a photo of student answer sheet using mobile camera. * If the teacher sign out from the application and logged in again to check student's marks. * If student shaded all 40 questions and the exam have only 10 questions. * If the teacher removed a particular exam from the drop down list of exams. * If the teacher adds new exam which has similar Exam ID of an existing exam in drop down list. | * Exam correct options will be saved in the array and it breaks where ever there is a null value in a particular index, then it saved the maximum score mark as the size of the array. * The application will stop the process and generate an error message. * An error message will pop up (Exam ID was not found). * 1 mark will be reduced. * 1 mark will be reduced. * The process of reading the student options will be progressed since there is no quality of the image is needed. * All the details will be stored in a file under the teacher account, once the login is successfully processed the file will be loaded so the teacher can view his previous work. * first 10 questions will be taken and the remaining questions will be ignored. * The list of student names and marks of that particular exam will be deleted as well. * the correction will be based on the previous exam which has that Exam ID, Exam ID must always be unique. |

1.2 : Black box-testing table

**4.2.2White box testing:**

After Black Box testing was completed. We started making sure that every aspect was working correctly internally in the system, so we performed **White Box testing** as followed:

* Test if the entered user name in sign up button does exist or not.
* Test if the user name and password is matched.
* Test if there is a null value in text field while entering exam correct options.
* Test to make sure that the application opens the executable file of c# project.
* Test to make sure C# project always supplying the correct data by passing it through command line.
* Test to make sure that the application calls the executable file of the scanner.
* Test to make sure that the application calculates the correct score mark of the student.
* Test to make sure that all exams that have been added by the teacher is saved under the teacher's account.
* Test to make sure that the teacher can only browse images.
* Test to make sure that the teacher browse student sheet image which has the same Exam ID that have been added to the exam.

**Chapter 5: Conclusion and Future**

**5.1 Summary of Findings:**

Our project is about marking multiple choice questions by scanning the student answer sheet. Based on the shaded circles the application will identify the options selected and calculate the scored mark of a particular student.

* The problems that we found during the implementation:
  + There used to be some problems on finding the executable file of the scanner when the scanner driver wasn't installed.
  + It was hard to pass parameters from java project to c# using command line arguments.
  + There were issues in runtime environment because c# project couldn't figure the location of java project unless we keep them under the same folder.
  + It was hard to create our own scanning window since the scanner does not have SDK.

**5.2 Conclusion**

As a group, we start working on our project which is Electronic Examination Marking aiming that we will achieve our goal. Our main scope is to find the score mark of the student by marking it through our application. The marking can be done by scanning the student answer sheet. During the execution phase of the project the mentioned problems came up which leads us to changed our plan trying to find best solutions to overcome these issues. The problem with the scanner window was solved by installing the scanner driver and call the auto scan window into our application rather than creating from scratch. The other problem was passing parameters from java project to c# which contains the image path, teacher name and student name. This issue was solved by creating an interface in c# that allows the user to browse the required student answer sheet enter teacher name and student name rather than passing it from java.

**5.3 Future Scope**

We are aiming to improve E-marker application by increasing some features which will be very helpful to the users:

* Recommendations :
  + Provide a template for the key answer sheet based on the number of questions and options that have been provided by the teacher.
  + Generate a PDF which contains list of student names and marks to support printing document.
  + Create scanning window from java application to reduce user's effort of scanning the image, saving image somewhere and browse it for the correction.
  + Provide the user with student sheet image and the scored mark along with the correct options of the exam for the evidence.
  + Track the history of adding and removing exams.

**References**

[1] Kathy Sierra and Bert Bates, Head First Java, O'Reilly Media, USA, February 2005.

[2] Eric Freeman, Head First Design Patterns, O'Reilly Media, USA, Nov 2004.

[3] Donald G. Drake, Introduction to Java threads, JavaWorld, USA, Jan 1996, <http://www.javaworld.com/jw-04-1996/jw-04-threads.html>

# [4] Umar.techBOY,Optical Marks Recognition (OMR) Engine 1.0, USA,20 Aug 2013,

# <http://www.codeproject.com/Articles/451169/Csharp-Optical-Marks-Recognition-OMR-Engine>

# [5] [Jeff Friesen](https://today.java.net/pub/au/174),Acquire Images with TWAIN and SANE

# , USA,November 18, 2004,

# <https://today.java.net/article/2004/11/16/java-tech-acquire-images-twain-and-sane-part-1>