A MINI PROJECT REPORT

**On**

**QR BOD**

**Submitted by**

**Arpita Singh (161500126)**

**Tejbir Singh (161500584)**

**Supervised by**

**Mr. Pankaj Kapoor**

(Technical Trainer)

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA University**

**Mathura- 281406, INDIA**

**January, 2019**

**Department of Computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



**Declaration**

We hereby declare that the work which is being presented in the Mini Project “**QR BOD”,** in partial fulfillment of the requirements for Mini-Project LAB, is an authentic record of our own work carried under the supervision of **Mr. Pankaj Kapoor, Technical Trainer, GLA University, Mathura**.

**Tejbir Singh**

**Arpita Singh**



**Department of Computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**

**CERTIFICATE**

This is to certify that the project entitled **“QR BOD”** carried out in Mini Project – II Lab is a bonafide work done by **Arpita Singh (16500126)**, **Tejbir Singh (161500584),** and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

**Signature of Supervisor:**

**Name of Supervisor:**

**Date:**

**ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received

Our heartiest thanks to **Dr. (Prof). Anand Singh Jalal,** Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe special debt of gratitude to **Mr. Pankaj Kapoor,** Technical Trainer Department of CEA, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Arpita Singh

Tejbir Singh

**Table of Contents**

|  |  |
| --- | --- |
| Declaration ii |  |
| Certificate iii |  |
| Acknowledgments iv |  |
| Abstract v |  |
| Table of Contents vi |  |
| 1. **Introduction 1** |  |
| * 1. Overview………………………………………………… 1 |  |
| * 1. Motivation ………………………………………………. 2 2 ………………………..………………………………………………............. |  |
|  |  |
| 1. **Software Requirement Analysis 3** |  |
| 2.1 Tools Used………….……………………………………. 3 …………………………………………………….. |  |
| 2.2 Problem ………………………………………………….. 3  2.3 Define the module and their functionalities……….……... 4  2.4 Requirements……………………………………………… 10  ………..………………………………… |  |
| 1. **Software Design 11** |  |
| * 1. Sequence Diagram……………………………………….. 11 |  |
| 3.2 Data Flow Diagram………………………………………. 12 |  |
|  |  |
| 1. **Libraries** …………………………………………………….. **13** |  |
| 1. **Sample Code** ……………………………………………….. **38** 2. **References/Bibliography**……………………………………. **43** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Chapter - 1** **Introduction**

* 1. **Overview**

Our project is very useful in all the digitally driven devices, just by one-click we can get the information about anything or person we want to, it not only just gives the thing name but can also give the geo coordinates, other information. We should be able to watch the movies, match, anything just by one scan from our mobile phones. Through the development phase of this project, we will be using technologies like-Java Language

* 1. **Motivation**

“We are going to enter into the world, where everything happens before the blink of our eyes” this is a quote which shows the future. We are here just to doing it possible by just one scan we are able to get the information of everything in the world. We can create the QR code of anything in this world by which the reach of anything to anyone is possible anywhere and simply. QR codes is a step towards the cashless economy which is the easiest way to do any transaction, so by QR code we want to create the identity of anything in this world.

**Chapter – 2**

**Software Requirement Analysis**

**2.1 Tools Used**

**Java Language-** Java is a general-purpose programming language that is class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to "bytecode" that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture**.**

**IntelliJ Idea-** IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

**2.2 Problem**

As information is increasing day bay day it is very difficult to store and show all information in single page. Using QR-CODE we can store all the information in single image and by scanning the image we can get all the information store in it.

**2.3 Modules and their Functionality**

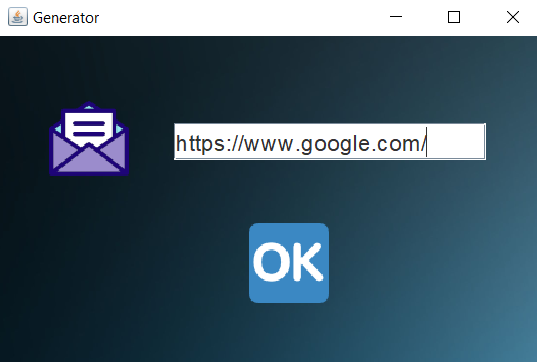
**GUI modules- (Swing, AWT, ImageIO)**

**Swing-** Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window class(AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.

**AWT**- The Abstract Window Toolkit (AWT) is Java's original platform- dependent windowing, graphics, and user-interface widget toolkit, preceding Swing. The AWT is part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface (GUI) for a Java program

**ImageIO-** javax.imageio.ImageIO is a utility class which provides lots of utility method related to images processing in Java. Most common of them is reading form image file and writing images to file in java.

**ZXING**- Zxing, pronounced as Zebra Crossing, is an open source, multi-format 1D/2D barcode image processing library implemented in java. We have used this library in the program to get the image of the QR Code and displaying on the screen.

**Input Format**

**Output Format**

**2.4 Requirements**

1. **Hardware Requirements**

* Minimum 4 GB RAM, and
* Minimum i3 processor.

1. **Software Requirements**

* Any windows based operating system,
* JAVA,
* INTELLIJ, and
* JAVA Libraries.

**Chapter – 3**

**Software Design**

**3.1 Sequence Diagram**

Sequence Diagram is an interaction diagram that details how operations are carried out -- what messages are sent and when. Sequence diagrams are organized according to time. The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

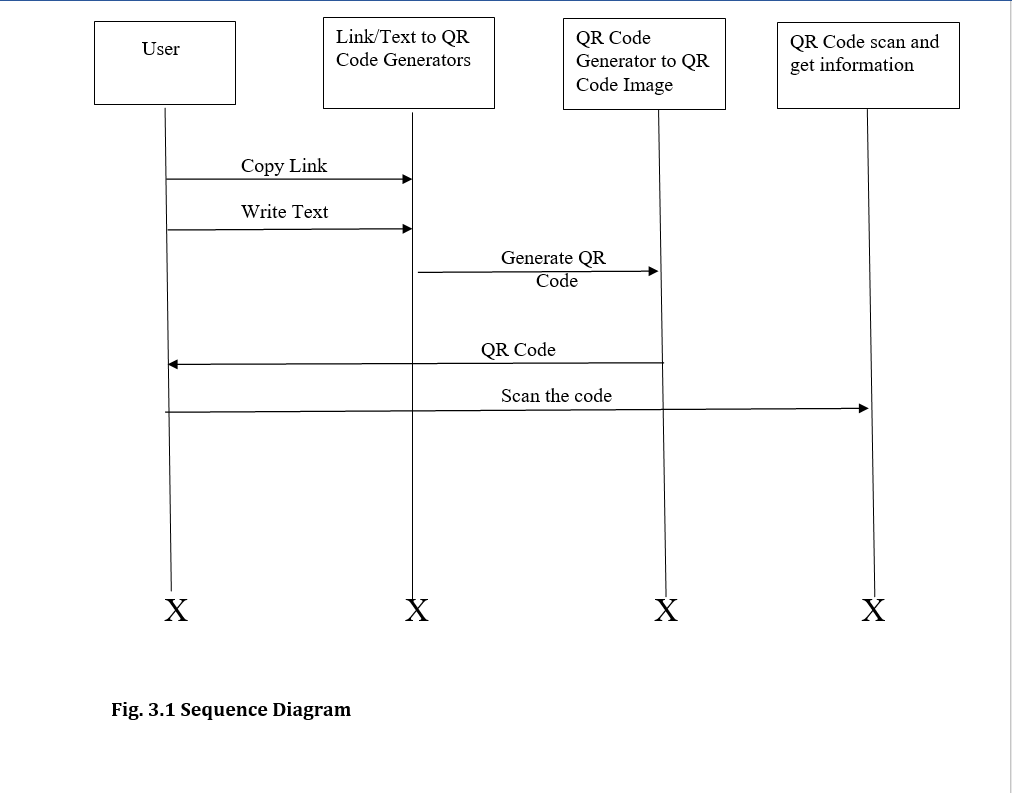


Fig. 3.1 Sequence Diagram

**3.2 Data Flow Diagram**

A Data Flow Diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and non technical audiences, from developer to CEO. That’s why DFDs is remains so popular after all these years. While they work well for dataflow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

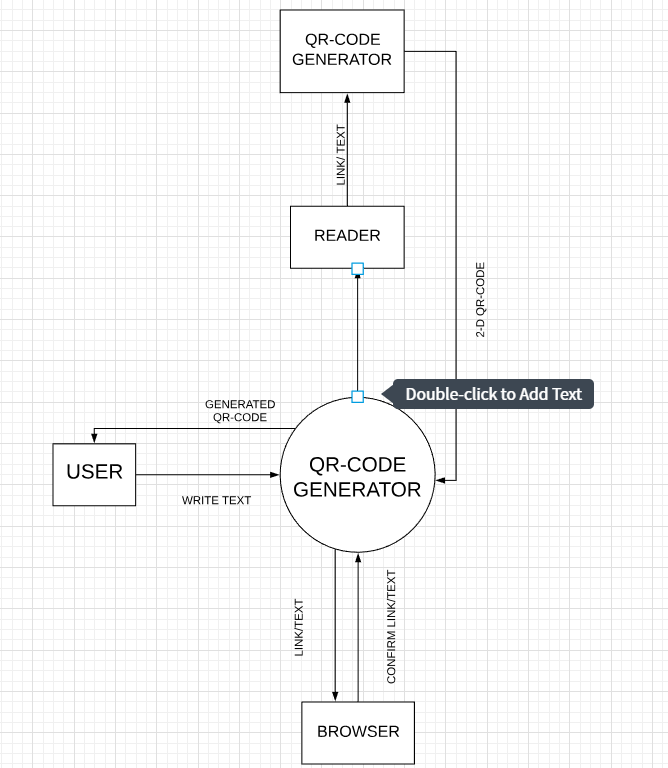


Fig. 3.2.1 Level 0 DFD

**Chapter – 4**

**Libraries**

**4.1 Libraries used**

**1. Zxing Library**

We’ll use Google’s Zxing library to generate QR codes for our application.

Zxing, pronounced as Zebra Crossing, is an open source, multi-format 1D/2D barcode image processing library implemented in java. We have used this library in the program to get the image of the QR Code and displaying on the screen. And also, we have used it into the reading of the QR Code when we want to read and get the output.

2. **Another Library**

**Swing**

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window class(AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

**AWT**

The Abstract Window Toolkit (AWT) is Java's original platform-dependent windowing, graphics, and user-interface widget toolkit, preceding Swing. The AWT is part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface (GUI) for a Java program. AWT is also the GUI toolkit for a number of Java ME profiles. For example, Connected Device Configuration profiles require Java runtimes on mobile telephones to support the Abstract Window Toolkit.

**ImageIO**

javax.imageio.ImageIO is a utility class which provides lots of utility method related to images processing in Java. Most common of them is reading form image file and writing images to file in java. You can write any of .jpg, .png, .bmp or .gif images to file in Java.

**Chapter – 5**

**Sample Code**

* 1. **Generation of QR code**

package quickresponseCodes.implementation;

import com.google.zxing.BarcodeFormat;

import com.google.zxing.WriterException;

import com.google.zxing.client.j2se.MatrixToImageWriter;

import com.google.zxing.common.BitMatrix;

import com.google.zxing.qrcode.QRCodeWriter;

import quickresponseCodes.Author;

import java.io.File;

import java.io.IOException;

@Author(name="Tejbir Singh and Arpita Singh",date="26 Feb 2019")

public class Generation {

public static void generateQRCode(String text,int width,int height, String path)throws IOException,WriterException{

QRCodeWriter writer=new QRCodeWriter();

BitMatrix bitMatrix=writer.encode(text,BarcodeFormat.QR\_CODE,width,height);

MatrixToImageWriter.writeToFile(bitMatrix,"PNG",new File(path));

}

}

* 1. **Reading the QR Code**

package quickresponseCodes.implementation;

import com.google.zxing.\*;

import com.google.zxing.client.j2se.BufferedImageLuminanceSource;

import com.google.zxing.common.HybridBinarizer;

import quickresponseCodes.Author;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

@Author(name="Tejbir Singh and Arpita Singh",date="22 feb 2019")

public class Reader {

public static String decodedText;

public static String decodeQRCode(File qrCodeimage) throws IOException,NotFoundException{

BufferedImage bufferedImage = ImageIO.read(qrCodeimage);

LuminanceSource source = new BufferedImageLuminanceSource(bufferedImage);

BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));

Result result = new MultiFormatReader().decode(bitmap);

return result.getText();

}

public static void decode(String filename) {

File file=new File(filename);

try {

decodedText = decodeQRCode(file);

}

catch (Exception h){h.printStackTrace();}

System.out.println("Decoded text = " + decodedText);

}

}

* 1. **Creating the frame for the QR Code**

package quickresponseCodes.frames;

import quickresponseCodes.Author;

import quickresponseCodes.implementation.Generation;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

@Author(name="Arpita Singh and Tejbir Singh",date="15 Feb 2019")

public class CreationFrame{

static JFrame frame;

static JPanel panel;

static JLabel w,e,back;

static BufferedImage s,d;

static JTextField z;

private ImageIcon background;

private Image img,temp;

public CreationFrame(){

frame=new JFrame("Generator");

frame.setSize(450,300);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(2);

panel=new JPanel();

panel.setLayout(null);

try{

s=ImageIO.read(new File("images/message.png"));

d=ImageIO.read(new File("images/ok.png"));

}

catch (IOException h){

h.printStackTrace();

}

w=new JLabel(new ImageIcon(s));

e=new JLabel(new ImageIcon(d));

z=new JTextField(10);

z.setFont(new Font("Dubai",0,18));

background = new ImageIcon("images/securitybackground.jpg");

img = background.getImage();

temp = img.getScaledInstance(500, 600, 4);

background = new ImageIcon(temp);

back = new JLabel(background);

back.setLayout(null);

back.setBounds(0, 0, 500, 600);

w.setBounds(40,50,64,64);

z.setBounds(140,70,250,30);

e.setBounds(200,150,64,64);

e.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

try{

Generation.generateQRCode(z.getText(),300,300,"./newfile.png");}

catch (Exception h){h.printStackTrace();}

JOptionPane.showMessageDialog(frame,"Generated!");

}

});

panel.add(w);panel.add(z);panel.add(e);

panel.add(back);

frame.add(panel);

frame.setVisible(true);

}

}

* 1. **Reading the Frame for QR Code**

package quickresponseCodes.frames;

import quickresponseCodes.Author;

import quickresponseCodes.Browse;

import quickresponseCodes.implementation.Reader;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

@Author(name="Arpita Singh and Tejbir Singh",date="20 Feb 2019")

public class ReadingFrame {

private static JFrame frame;

private static JPanel panel;

private static JLabel back,q,w;

private ImageIcon background;

private Image img,temp;

private static JTextField f;

private static BufferedImage a,s;

public ReadingFrame(){

frame=new JFrame();

frame.setSize(500,550);

frame.setDefaultCloseOperation(2);

frame.setLocationRelativeTo(null);

panel=new JPanel();

panel.setLayout(null);

try {

a = ImageIO.read(new File("./newfile.png"));

s = ImageIO.read(new File("images/ereader.png"));

}

catch (IOException e){

e.printStackTrace();

}

q=new JLabel(new ImageIcon(a));

q.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

JOptionPane.showMessageDialog(frame,"I am just a QR-Code haha!\nThough you can try to decode me");

}

});

w=new JLabel(new ImageIcon(s));

f=new JTextField(10);

f.setFont(new Font("Dubai",0,15));

f.setEditable(false);

background = new ImageIcon("images/managementbackground.jpg");

img = background.getImage();

temp = img.getScaledInstance(500, 550, 4);

background = new ImageIcon(temp);

back = new JLabel(background);

back.setLayout(null);

back.setBounds(0, 0, 500, 550);

q.setBounds(90,50,300,300);

w.setBounds(50,420,64,64);

f.setBounds(160,430,250,30);

w.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

try{

Reader.decode("./newfile.png");}

catch (Exception h){h.printStackTrace();}

f.setText(Reader.decodedText);

Browse.surf(f.getText());

}

});

panel.add(q);panel.add(w);panel.add(f);

panel.add(back);

frame.add(panel);

frame.setVisible(true);

}

}

**5.Author interface for writing our name**

package quickresponseCodes;

public @interface Author{

String name();

String date();

}

**6.Browsing the text**

package quickresponseCodes;

import java.awt.Desktop;

import java.net.URI;

import java.net.URL;

@Author(name="Arpita Singh and Tejbir Singh",date="28 feb 2019")

public class Browse {

public static boolean isValidURL(String urlString) {

try {

URL url = new URL(urlString);

url.toURI();

return true;

} catch (Exception exception) {

return false;

}

}

public static void surf(String URL) {

Desktop d = Desktop.getDesktop();

try {

if(isValidURL(URL)) {

d.browse(new URI(URL));

}

}

catch (Exception e){e.printStackTrace();}

}

}

1. **Menu file for running the program**

package quickresponseCodes;

import quickresponseCodes.frames.CreationFrame;

import quickresponseCodes.frames.ReadingFrame;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

@Author(name="Tejbir Singh and Arpita Singh",date="2 March 2019")

public class Menu {

static JFrame frame;

static JPanel panel;

static BufferedImage a,s,d;

static JLabel back,w,e,r;

private ImageIcon background;

private Image img,temp;

private Menu(){

frame=new JFrame("QR CODES");

frame.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

int a=JOptionPane.showConfirmDialog(frame,"Are you sure?");

if(a==0)

frame.dispose();

}

});

frame.setSize(500,500);

frame.setLocationRelativeTo(null);

panel=new JPanel();

panel.setLayout(null);

try{

a=ImageIO.read(new File("images/pen.png"));

s=ImageIO.read(new File("images/eye.png"));

d=ImageIO.read(new File("images/exit.png"));

}

catch (IOException e){

e.printStackTrace();

}

w=new JLabel(new ImageIcon(a));

e=new JLabel(new ImageIcon(s));

r=new JLabel(new ImageIcon(d));

background = new ImageIcon("images/homeworkbackground.jpg");

img = background.getImage();

temp = img.getScaledInstance(500, 500, 4);

background = new ImageIcon(temp);

back = new JLabel(background);

back.setLayout(null);

back.setBounds(0, 0, 500, 500);

w.setBounds(50,120,128,128);

w.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

new CreationFrame();

}

});

e.setBounds(300,120,128,128);

e.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

new ReadingFrame();

}

});

r.setBounds(200,300,128,128);

r.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

JOptionPane.showMessageDialog(frame,"Have a good day!");

frame.dispose();

}

});

panel.add(w);panel.add(e);panel.add(r);

panel.add(back);

frame.add(panel);

frame.setVisible(true);

JOptionPane.showMessageDialog(frame,"This is a QR code applet created by Arpita Singh and Tejbir Singh.\nYou can create your own your QR-Code, and can also read them\nAll Images are clickable!");

}

public static void main(String ...s){

new Menu();

}

}

**Chapter - 5**

**Reference/Bibliography**

* <https://blog.hubspot.com/blog/tabid/6307/bid/29449/how-to-create-a-qr-code-in-4-quick-steps.aspx>
* <https://www.github.com>
* <https://www.wikipedia.com>
* <https://ieeexplore.ieee.org/document/8120658/>
* <https://crunchify.com/java-simple-qr-code-generator-example/>
* <https://www.pdfdrive.com/java-the-complete-reference-7th-edition-e3625514.html>