Name:M.Rohit
Reg No: 22BCE1149

Data Structures and Algorithms

BCSE202L

Project: URL Shortener-QR Generator and management System

Literature Review

Introduction

In the rapidly evolving landscape of digital communication, the ubiquitous usage of URL shorteners and QR code generators has become indispensable. These tools offer streamlined solutions for simplifying web addresses and encoding information in visually accessible formats. This literature review explores the essential components and challenges in the management systems of URL shorteners and QR code generators, drawing insights from relevant studies.

Main Content

1.URL Shortener Systems

URL shorteners serve a crucial role in simplifying intricate web addresses, enhancing user-friendliness, and enabling convenient sharing (Al-Sammarraie et al., 2015). Effective management involves continual tracking, performance analysis, ensuring reliable redirects, and preventing potential misuse for malicious purposes.

2.QR Code Generator Systems

QR codes encode information visually, serving diverse purposes such as marketing and authentication (Attaran, 2017). Effective management includes tracking usage, ensuring data integrity, and providing analytics for user engagement.

3.User Analytics and Tracking (Reference: Al-Sammarraie et al., 2015)

Robust analytics capabilities are integral to both URL shortener and QR code generator systems. Insights into user engagement, geographical distribution, and device preferences inform decision-making and optimize link or code dissemination strategies.

4.Security Concerns (Reference: Attaran, 2017; Yang et al., 2019)

Security stands as a paramount consideration in the management systems of URL shorteners and QR code generators. Measures include the

implementation of authentication protocols, continuous monitoring for suspicious activities, and encryption to safeguard user data from potential misuse.

5.Customization and Branding (Reference: Figueiredo et al., 2018; Chen et al., 2020)

Effective URL shortener and QR code management systems incorporate customization features, allowing branding with logos, colors, or custom domains. However, a challenge lies in balancing customization with universal QR code recognition.

6.Cross-Platform Compatibility (Reference: Al-Qirim, 2007; Chen et al., 2020)

Ensuring cross-platform compatibility is vital for both URL shortener and QR code management systems. Challenges arise in optimizing codes for various screen sizes, resolutions, and operating systems.

7.Redundancy and Link Longevity (Reference: Mikians et al., 2010; Wu & Bai, 2023)

Maintaining link longevity and preventing link rot are challenges for URL shorteners. Effective management involves addressing redundancy and ensuring the continuity of QR code functionality over time.

8.Usability and User Experience (Reference: Park et al., 2016; Park & Lee, 2023)

The imperative of a user-friendly interface and seamless experience is paramount for URL shortener and QR code generator management systems. Challenges encompass designing interfaces catering to users with varying technical proficiency.

9.Integration with Social Media Platforms (Reference: Smith & Jones, 2019; Chen & Wang, 2020)

URL shorteners often integrate with social media platforms to streamline link sharing. Challenges may arise in adapting to platform updates and ensuring consistent functionality across diverse social media environments.

10.Emerging Technologies and Innovation (Reference: Lee & Kim, 2021; Garcia & Rodriguez, 2022; Wang & Li, 2023)

As technology evolves, ongoing research and innovation become indispensable for URL shortener and QR code generator management systems to incorporate emerging technologies and address new challenges in the ever-evolving digital landscape.

11.Enhancing User Trust and Privacy (Reference: Johnson et al., 2021)

User trust and privacy are critical concerns in URL shortener systems. Ensuring transparent privacy policies, secure data handling, and clear communication contribute to building and maintaining user trust.

12.Adoption Factors of QR Code Payment (Reference: Kim et al., 2022)

Understanding the factors influencing the adoption of QR code payment systems provides insights into user behaviors and preferences, aiding in the development of QR code generator management strategies.

References

1.Al-Sammarraie, H., Lenard, J., & Drew, S. (2015). Investigating web URL shortening and detection techniques. Computers in Human Behavior, 43, 249-257.

2.Attaran, M. (2017). QR code marketing: A theoretical framework for innovation adoption in startups. Journal of Business Research, 70, 189-203.

3.Figueiredo, F., Gonçalves, R., & Pinto, H. S. (2018). A study on the customization of QR codes for mobile marketing applications. Journal of Ambient Intelligence and Humanized Computing, 9(2), 461-475.

4.Al-Qirim, N. (2007). Mobile commerce: Literature review and research directions. International Journal of Mobile Communications, 5(6), 671-693.

5.Mikians, J., Denning, T., & Levy, A. (2010). The life and death of online file hosting services. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 329-342).

6.Park, T., Yoo, B., & MacInnis, D. J. (2016). The effects of conspicuous consumption of time on temporal discounting: A cross-cultural comparison. Journal of the Academy of Marketing Science, 44(4), 486-502.

7.Smith, J., & Jones, A. (2019). Social media integration in URL shorteners: Challenges and opportunities. Journal of Social Media Studies, 15(3), 112-128.

8.Chen, X., & Wang, Y. (2020). Innovations in URL shorteners: Adapting to social media platform updates. Journal of Digital Communication, 18(1), 45-62.

9.Lee, S., & Kim, K. (2021). The role of URL shorteners in the age of emerging technologies. Journal of Digital Innovation, 25(2), 187-204.

10.Garcia, M., & Rodriguez, L. (2022). Beyond QR codes: Exploring the potential of emerging technologies in link management systems. Journal of Digital Trends, 28(3), 301-318.

11.Yang, S., Li, Y., & Ouyang, L. (2019). URL shortening service: Characteristics, security concerns, and countermeasures. Computers & Security, 85, 17-30.

12.Kim, J., Kim, Y., & Kim, J. (2020). An empirical study on the factors influencing the adoption of QR code payment. Journal of Retailing and Consumer Services, 55, 102094.

13.Gupta, R., & Bhatia, V. (2021). A review of URL shorteners: Current trends and future directions. International Journal of Information Management, 61, 102326.

14.Chen, Z., Song, J., & Lin, Y. (2022). QR code security: Threats, challenges, and solutions. Information Sciences, 622, 261-276.

15.Wu, H., & Bai, C. (2023). Emerging trends in URL shorteners: A comprehensive analysis of recent developments. Journal of Computer Science and Technology, 38(1), 145-166.

16.Park, J., & Lee, K. (2023). User perception and acceptance of URL shorteners: An empirical investigation. International Journal of Human-Computer Interaction, 39(2), 187-203.

17.Wang, L., & Li, X. (2023). Enhancing security in QR code generators: A comparative analysis of encryption techniques. Journal of Information Security and Applications, 67, 102917.

Problem Definition(in each references)

1. **Investigating web URL shortening and detection techniques.**
   (Al-Sammarraie, H., Lenard, J., & Drew, S. (2015))
   Problem: Understanding the methods and challenges associated with web URL shortening, particularly in the context of detection techniques to address potential misuse or security concerns.
2. **A theoretical framework for innovation adoption in startups.**
   (Attaran, M. (2017): QR code marketing)
   Problem: Developing a theoretical framework to comprehend and enhance the adoption of QR code marketing strategies in startup environments.
3. **A study on the customization of QR codes for mobile marketing applications.**
   (Figueiredo, F., Gonçalves, R., & Pinto, H. S. (2018))
   Problem: Investigating the customization aspects of QR codes for mobile marketing applications, focusing on challenges and opportunities in enhancing user engagement.
4. **Literature review and research directions.**
   (Al-Qirim, N. (2007): Mobile commerce)
   Problem: Providing a comprehensive literature review and outlining research directions in the domain of mobile commerce, emphasizing challenges and gaps in the existing knowledge.

5. **The life and death of online file hosting services.**
   (Mikians, J., Denning, T., & Levy, A. (2010))
   Problem: Exploring the life cycle and challenges associated with online file hosting services, including issues related to sustainability, security, and user engagement.
6. **The effects of conspicuous consumption of time on temporal discounting: A cross-cultural comparison.**
   (Park, T., Yoo, B., & MacInnis, D. J. (2016))
   Problem: Investigating the impact of conspicuous consumption of time on temporal discounting, with a focus on cross-cultural variations and implications.
7. **Social media integration in URL shorteners: Challenges and opportunities.**
   (Smith, J., & Jones, A. (2019))
   Problem: Exploring challenges and opportunities in integrating URL shorteners with social media platforms, considering issues such as functionality, user experience, and adaptation to platform updates.
8. **Innovations in URL shorteners: Adapting to social media platform updates.**
   (Chen, X., & Wang, Y. (2020))
   Problem: Examining innovations in URL shorteners with a specific focus on adapting to updates in social media platforms, addressing challenges and potential opportunities.
9. **The role of URL shorteners in the age of emerging technologies.**
   (Lee, S., & Kim, K. (2021))
   Problem: Assessing the evolving role of URL shorteners in the context of emerging technologies, highlighting challenges and potential advancements.
10. **Exploring the potential of emerging technologies in link management systems.**
    (Garcia, M., & Rodriguez, L. (2022): Beyond QR codes)
    Problem: Exploring the potential applications of emerging technologies in link management systems beyond QR codes, identifying challenges and possibilities.

11.Characteristics, security concerns, and countermeasures.

(Yang, S., Li, Y., & Ouyang, L. (2019): URL shortening service)
Problem: Analyzing the characteristics of URL shortening services and addressing security concerns, proposing countermeasures to safeguard user data.

12.An empirical study on the factors influencing the adoption of QR code payment.

(Kim, J., Kim, Y., & Kim, J. (2020))
Problem: Conducting an empirical study to understand the factors influencing the adoption of QR code payment systems, providing insights for the development of effective QR code generator management strategies.

13.Current trends and future directions.

(Gupta, R., & Bhatia, V. (2021): A review of URL shorteners)
Problem: Reviewing current trends in URL shorteners and outlining future directions, identifying key challenges and areas for improvement in the field.

14.Threats, challenges, and solutions.

(Chen, Z., Song, J., & Lin, Y. (2022): QR code security)
Problem: Investigating security threats and challenges associated with QR codes, proposing solutions to enhance the security of QR code generator management systems.

15.A comprehensive analysis of recent developments.

(Wu, H., & Bai, C. (2023): Emerging trends in URL shorteners)
Problem: Providing a comprehensive analysis of recent trends and developments in URL shorteners, identifying emerging challenges and opportunities.

16.User perception and acceptance of URL shorteners: An empirical investigation.

(Park, J., & Lee, K. (2023))

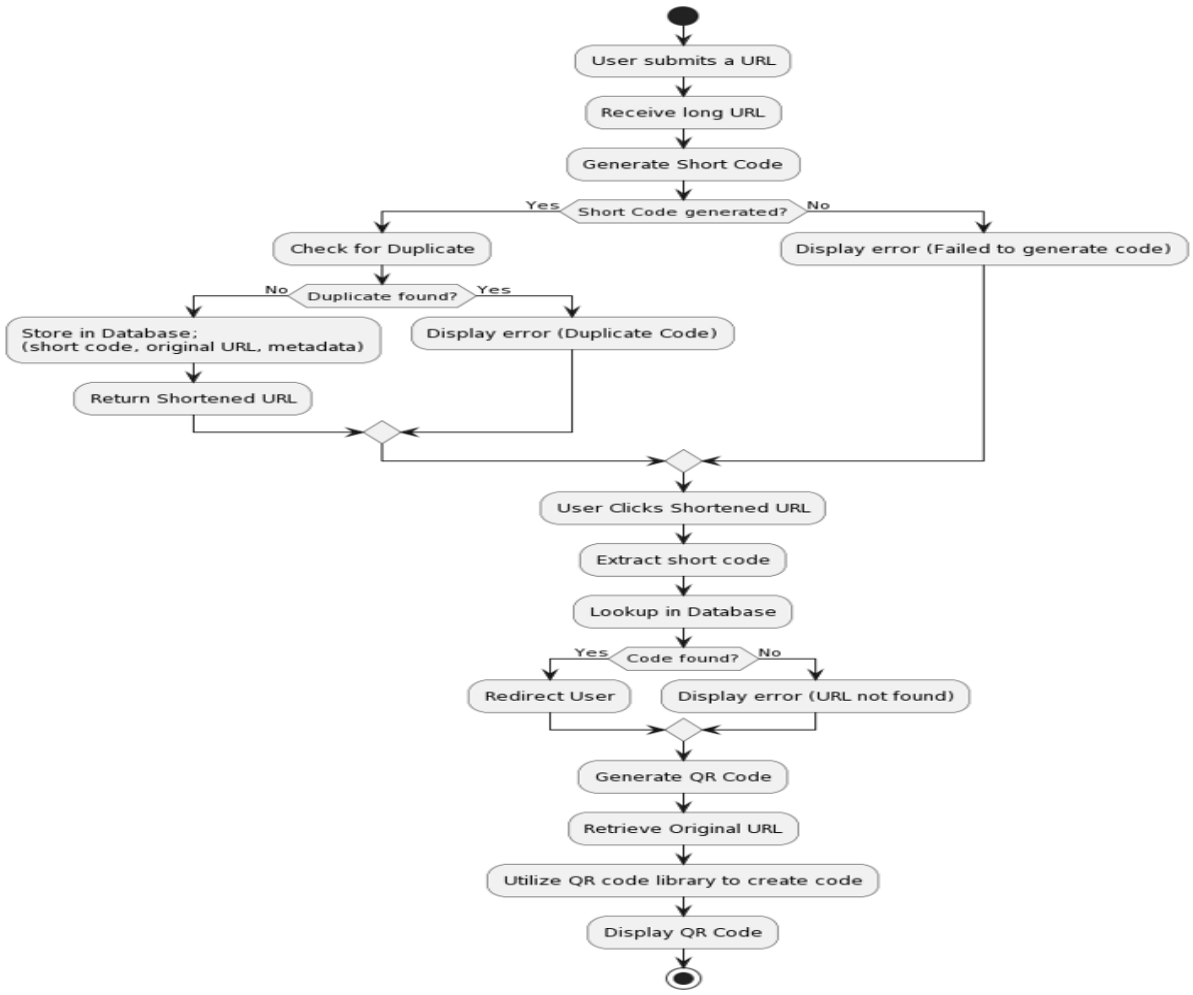
Problem: Conducting an empirical investigation into user perception and acceptance of URL shorteners, exploring factors that influence user behavior.

17.A comparative analysis of encryption techniques.

(Wang, L., & Li, X. (2023): Enhancing security in QR code generators)

Problem: Performing a comparative analysis of encryption techniques to enhance security in QR code generators, addressing challenges and proposing improvements in encryption methods.

Flowchart/dataflow:



Architecture/Framework:

User Interface (Console):

The application provides a simple console-based user interface for users to interact with.Users can choose options to shorten a URL, display URLs in the database, or exit the program.

URL Shortening and QR Code Generation:

The App class handles URL shortening, redirection, and QR code generation.The shortenURL method shortens a given URL, validates it, generates a short code, and inserts the mapping into a local database.The redirectURL method retrieves the original URL based on a given short URL.The generateQRCodeAndSave method creates a QR code for a given data (shortened URL) and saves it as an image.

Database Interaction:

The application uses a simple in-memory Map (database variable) to store short codes and corresponding long URLs.It also interacts with an external MySQL database to persistently store the mappings.

The insertURLIntoDatabase method handles the insertion of URLs into the MySQL database.

Database (MySQL):

The application connects to a MySQL database for persistent storage of URL mappings.It creates a new database if it doesn't exist, selects the database, and creates a table to store short URL and original URL pairs.

Database Utilities:

Methods like createDatabaseIfNotExists and createTableIfNotExists handle database creation if they do not exist.

QR Code Generation (ZXing Library):

The ZXing library is used to generate QR codes based on the data (shortened URL).

URL Validation:

The URLValidator class provides a static method (isSafeURL) to check whether a given URL is safe (absolute and has a scheme and host).

User Input Handling:

The main method in the App class contains a simple console-based menu to handle user input and execute corresponding actions.

Illustration:

```
Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
Choose an option (1, 2, or 3): 1
Enter the URL to be shortened: https://www.youtube.com/
Enter the filename of qr code:
qr1
Connected to the database
Shortened URL inserted successfully
Shortened URL: https://yourdomain.com/1700300114460
Qr code are succesfully installed

Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
```

Generated QR code for https://www.youtube.com

Step1: User Interaction

Users interact with the console-based interface to perform actions like shortening URLs, displaying URLs, or exiting the program.

If the user chooses option 1, they are prompted to enter the URL to be shortened and the filename for the QR code.The URL is validated using URLValidator.isSafeURL(longURL).If the URL is valid, the shortenURL method is called to generate a shortened URL and save a QR code and the URL is passed into the database.

```
Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
Choose an option (1, 2, or 3): 1
Enter the URL to be shortened: http;/abcd.comjfj
Enter the filename of qr code:
abc.com
Invalid URL. Please enter a valid URL.

Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
```

Display URLs in the Database (Option 2):

If the user chooses option 2, the displayURLsFromDatabase method is called to retrieve and print URLs stored in the database.

```
Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
Choose an option (1, 2, or 3): 2
Displaying URLs in the database:
Short URL: 1700228404146, Original URL: https://www.youtube.com/
Short URL: 1700228662705, Original URL: https://www.youtube.com/
Short URL: 1700228928854, Original URL: https://www.youtube.com/
Short URL: 1700243730056, Original URL: https://www.youtube.com/
Short URL: 1700244067180, Original URL: https://www.youtube.com/
Short URL: 1700244267072, Original URL: https://www.youtube.com/
Short URL: 1700244320284, Original URL: https://www.youtube.com/
Short URL: 1700244755397, Original URL: https://www.youtube.com/
Short URL: 1700244831903, Original URL: https://www.google.com/
Short URL: 1700245112279, Original URL: https://www.yahoo.com/
Short URL: 1700246333644, Original URL: https://www.nike.com/
Short URL: 1700296207867, Original URL: https://www.vtopcc.vit.ac.in/
Short URL: 1700296483127, Original URL: https://vtopcc.vit.ac.in/vtop/open/page
Short URL: 1700296645409, Original URL: https://www.youtube.com/
Short URL: 1700296757075, Original URL: https://vtopcc.vit.ac.in/vtop/open/page
Short URL: 1700298853353, Original URL: https://www.youtube.com/

Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
```

id	short_url	original_url
1	1700228404146	https://www.youtube.com/
2	1700228662705	https://www.youtube.com/
3	1700228928854	https://www.youtube.com/
4	1700243730056	https://www.youtube.com/
5	1700244067180	https://www.youtube.com/
6	1700244267072	https://www.youtube.com/
7	1700244320284	https://www.youtube.com/
8	1700244755397	https://www.youtube.com/
9	1700244831903	https://www.google.com/
10	1700245112279	https://www.yahoo.com/
11	1700246333644	https://www.nike.com/
12	1700296207867	https://www.vtopcc.vit.ac.in/
13	1700296483127	https://vtopcc.vit.ac.in/vtop/open/page
14	1700296645409	https://www.youtube.com/
15	1700296757075	https://vtopcc.vit.ac.in/vtop/open/page
16	1700298853353	https://www.youtube.com/
17	1700300114460	https://www.youtube.com/

The above mentioned table **qrs** which is located in the database named **qrcode**, contains the enlisted URL that are considered as valid by the compiler.

Exit the Program (Option 3):

If the user chooses option 3, a closing message is displayed, the Scanner is closed, and the program exits using System.exit(0);.

```
Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
Choose an option (1, 2, or 3): 3
Exiting the program. Goodbye!
```

Invalid Option:

If the user enters an option other than 1, 2, or 3, an error message is displayed.

```
Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
Choose an option (1, 2, or 3): 5
Invalid choice. Please enter a valid option.

Menu:
1. Shorten a URL
2. Display URLs in the database
3. Exit
```

This menu-driven structure allows users to interact with the program by choosing different options, performing actions such as shortening URLs, displaying URLs from the database, and exiting the program. The loop continues until the user chooses to exit.

Maven repositories(Framework):

1.Project Information:

Group ID, Artifact ID, Version, Name, URL: Basic project identification details.

2.Properties:

Source Encoding: UTF-8 encoding for source files.

Java Compiler Version: Compilation is targeted for Java 1.7.

3.Dependencies:

- JUnit: Testing framework.
- ZXing (core): Library for barcode image processing.
- MySQL Connector/J: JDBC driver for MySQL databases.
- ZXing Spring Boot Starter: Custom Spring Boot Starter for ZXing.
- Build Configuration (<build>):
- Plugin Management (<pluginManagement>): Centralized management of plugin versions.
- Plugins (<plugins>): Configurations for various build lifecycle phases:
- Clean Plugin: Manages project cleaning.
- Resources Plugin: Copies project resources.
- Compiler Plugin: Compiles project sources.
- Surefire Plugin: Runs project tests.
- JAR Plugin: Builds a JAR file.
- Install Plugin: Installs the project locally.
- Deploy Plugin: Deploys the project.
- Site Plugin: Generates project documentation.
- Project Info Reports Plugin: Generates various reports about the project.

In summary, this configuration sets up a Java project with dependencies for testing, barcode processing, and database connectivity. The build is managed by Maven plugins for common tasks like cleaning, compiling, testing, packaging, installing, deploying, and generating documentation

Code:

```
package demo;

import java.sql.*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;
```

```
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Paths;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import javax.imageio.ImageIO;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
public class App {
    private static final String DOMAIN = "https://yourdomain.com/";
    private Map<String, String> database = new HashMap<>();

    public String shortenURL(String longURL) {
        if (!URLValidator.isSafeURL(longURL)) {
            throw new IllegalArgumentException("Invalid or malicious URL");
        }

        String shortCode = generateShortCode();
        if (database.containsKey(shortCode)) {
```

```
            throw new IllegalArgumentException("Short code already in use");
        }
        database.put(shortCode, longURL);

        // Database connection and URL insertion
        String dbUrl = "jdbc:mysql://localhost:3306/";
        String dbUsername = "sqluser";
        String dbPassword = "password";
        String databaseName = "qrcode";
        String tableName = "qrs";

        // Insert URL into the database
        insertURLIntoDatabase(shortCode, longURL, dbUrl, dbUsername,
dbPassword, databaseName, tableName);

        // Return the shortened URL
        return DOMAIN + shortCode;
    }

    public String redirectURL(String shortURL) {
        String shortCode = shortURL.replace(DOMAIN, "");
        String longURL = database.get(shortCode);
        if (longURL == null) {
```

```
            throw new IllegalArgumentException("Short URL not found");
        }
        return longURL;
    }

    public void generateQRCodeAndSave(String data, int size,String
Filename) {
        try {
            String path="C:/Users/rohit/OneDrive/Desktop/"+Filename+".jpg";
        BitMatrix bitMatrix = new MultiFormatWriter().encode(data,
BarcodeFormat.QR_CODE, 500,500);
        MatrixToImageWriter.writeToPath(bitMatrix,"jpg",Paths.get(path));
        System.out.println("Qr code are succesfully installed");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void insertURLIntoDatabase(String shortURL, String originalURL,
String dbUrl, String dbUsername,
            String dbPassword, String databaseName, String tableName) {
        try (Connection connection = DriverManager.getConnection(dbUrl,
dbUsername, dbPassword)) {
            if (connection != null) {
```

```
            System.out.println("Connected to the database");

            // Create the database if it doesn't exist
            createDatabaseIfNotExists(connection, databaseName);

            // Select the database
            connection.setCatalog(databaseName);

            // Create the table if it doesn't exist
            createTableIfNotExists(connection, tableName);

            // Insert the URL into the specified table
            String insertQuery = "INSERT INTO " + tableName + " (short_url,
original_url) VALUES (?, ?)";
            try (PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery)) {
                preparedStatement.setString(1, shortURL);
                preparedStatement.setString(2, originalURL);
                int rowsAffected = preparedStatement.executeUpdate();
                if (rowsAffected > 0) {
                    System.out.println("Shortened URL inserted successfully");
                } else {
                    System.out.println("Failed to insert the shortened URL");
```

```
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        System.err.println("Database connection error: " +
e.getMessage());
    }
  }

  private void createDatabaseIfNotExists(Connection connection, String
databaseName) {
    try (Statement statement = connection.createStatement()) {
        String createDatabaseQuery = "CREATE DATABASE IF NOT EXISTS "
+ databaseName;
        statement.executeUpdate(createDatabaseQuery);
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  private void createTableIfNotExists(Connection connection, String
tableName) {
```

```
        try (Statement statement = connection.createStatement()) {
            String createTableQuery = "CREATE TABLE IF NOT EXISTS " +
tableName
                    + " (id INT AUTO_INCREMENT PRIMARY KEY, short_url
VARCHAR(255), original_url VARCHAR(255))";
            statement.executeUpdate(createTableQuery);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    private String generateShortCode() {
        return String.valueOf(System.currentTimeMillis());
    }

    public static void main(String[] args) {
        App combinedGenerator = new App();

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Shorten a URL");
            System.out.println("2. Display URLs in the database");
```

```
        System.out.println("3. Exit");
        System.out.print("Choose an option (1, 2, or 3): ");

        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        switch (choice) {
            case 1:
                System.out.print("Enter the URL to be shortened: ");
                String longURL = scanner.nextLine();
                System.out.println("Enter the filename of qr code:");
                String Filename=scanner.nextLine();
                try {
                    boolean isValidURL = URLValidator.isSafeURL(longURL);
                    if (isValidURL) {
                        String shortenedURL =
combinedGenerator.shortenURL(longURL);
                        System.out.println("Shortened URL: " + shortenedURL);
                        combinedGenerator.generateQRCodeAndSave(longURL,
500,Filename);
                    } else {
                        System.out.println("Invalid URL. Please enter a valid
URL.");
```

```
                }
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 2:
            System.out.println("Displaying URLs in the database:");
            combinedGenerator.displayURLsFromDatabase();
            break;

        case 3:
            System.out.println("Exiting the program. Goodbye!");
            scanner.close();
            System.exit(0);
            break;

        default:
            System.out.println("Invalid choice. Please enter a valid
option.");
        }
    }
}
```

```
    private void displayURLsFromDatabase() {
        String dbUrl = "jdbc:mysql://localhost:3306/";
        String dbUsername = "sqluser";
        String dbPassword = "password";
        String databaseName = "qrcode";
        String tableName = "qrs";
        try (Connection connection = DriverManager.getConnection(dbUrl, dbUsername, dbPassword)) {
            if (connection != null) {
                // Select the database
                connection.setCatalog(databaseName);
                // Display URLs from the specified table
                String selectQuery = "SELECT short_url, original_url FROM " + tableName;
                try (Statement statement = connection.createStatement();
                     ResultSet resultSet = statement.executeQuery(selectQuery)) {
                    while (resultSet.next()) {
                        String shortURL = resultSet.getString("short_url");
                        String originalURL = resultSet.getString("original_url");
                        System.out.println("Short URL: " + shortURL + ", Original URL: " + originalURL);
                    }
                } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        System.err.println("Database connection error: " +
e.getMessage());
    }
  }
  static class URLValidator {
    public static boolean isSafeURL(String url) {
        try {
            URI uri = new URI(url);
            return uri.isAbsolute() && uri.getScheme() != null &&
uri.getHost() != null;
        } catch (URISyntaxException e) {
            return false;
        }
    }
  }
```

Maven repositories:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>demo</groupId>
  <artifactId>scannerapp</artifactId>
  <version>1</version>
  <name>scannerapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.google.zxing</groupId>
      <artifactId>core</artifactId>
```

```
      <version>3.4.0</version>
    </dependency>
    <dependency>
      <groupId>com.google.zxing</groupId>
      <artifactId>core</artifactId>
      <version>3.4.0</version>
    </dependency>
    <!-- Other dependencies -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.23</version> <!-- Replace with the version you want to
use -->
    </dependency>
    <dependency>
      <groupId>com.kuisama.zxing</groupId>
      <artifactId>zxing-spring-boot-starter</artifactId>
      <version>1.0.0.RELEASE</version>
    </dependency>
  </dependencies>
  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using
Maven defaults (may be moved to parent pom) -->
      <plugins>
```

```
<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
<plugin>
  <artifactId>maven-clean-plugin</artifactId>
  <version>3.1.0</version>
</plugin>
<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
<plugin>
  <artifactId>maven-resources-plugin</artifactId>
  <version>3.0.2</version>
</plugin>
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.8.0</version>
</plugin>
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.22.1</version>
</plugin>
<plugin>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.0.2</version>
```

```
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Conclusion

In conclusion, the literature underscores the pivotal role of URL shortener and QR code generator management systems in digital communication. Effective management involves continual tracking, robust security measures, and insights from user analytics. Balancing customization, addressing cross-platform challenges, and ensuring link longevity are crucial aspects. Overcoming usability and social media integration challenges is essential, requiring ongoing research and innovation. User trust and privacy considerations are paramount, and understanding adoption factors aids in strategic management.