QR SHARE: QR CODE GENERATOR AND SCANNER FOR DATA TRANSFER

A Project Report

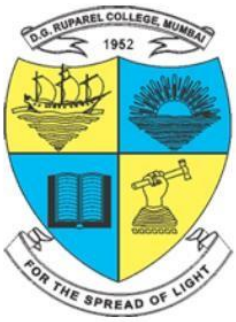
Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Ms. JANHVI NARENDRA BARASKAR

Seat No.:1067185



DEPARTMENT OF INFORMATION TECHNOLOGY

D. G. RUPAREL COLLEGE OF ARTS, SCIENCE & COMMERCE

(Affiliated to University of Mumbai)

SENAPATI BAPAT MARG, MAHIM, MUMBAI, 400 016

MAHARASHTRA

2023-24

MODERN EDUCATION SOCIETY, PUNE
The D. G. Ruparel College
OF ARTS, SCIENCE AND COMMERCE
senapati bapat marg, opp. matunga road station (w. r.), mahim, mumbai 400 016.
tel.: 022-24303733, 022-24303081, 022-24361139 • fax : 022-24303042

Department of Information Technology and Computer Science

Certificate

This is to certify that the project entitled, "_____", is bonafied work of Mr/Ms _____ bearing Seat. No: _____submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide Examiner Course Coordinator
Date:- Date:-

ABSTRACT

The "QR Share" project presents an innovative solution to the challenges inherent in traditional data transfer methods. Leveraging QR code technology, the project endeavours to create a user-friendly android application that streamlines and enhances the information sharing process. This application encompasses QR code generation, scanning, and diverse data transfer options, catering to both personal and professional requirements. By providing a comprehensive approach and seamless integration of QR codes, the project redefines data sharing paradigms, fostering convenience and productivity in today's digital landscape. Through its multifaceted functionalities, "QR Share" contributes to a more efficient and secure method of exchanging information, making strides towards the future of data sharing.

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to all those who have been instrumental in the development of the QR Share: QR Code Generator and Scanner for Data Transfer project. Your support and guidance have been invaluable on this journey.

First and foremost, I would like to express my heartfelt thanks to my mentors for their mentorship, unwavering support, and insightful feedback have been the driving force behind my progress. I would like to express my gratitude to the faculty and staff who have provided resources, assistance, and encouragement throughout the development process.

I would also like to thank my friends and companions who have helped and encouraged me for the successful completion of the project.

Thank you all for your dedication and belief in this project.

DECLARATION

I hereby declare that the project entitled, "**QR Share: QR Code Generator and Scanner for Data Transfer**" done at **D.G. Ruparel College of Arts, Science & Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Janhvi Narendra Baraskar

Signature:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The "QR Share: QR Code Generator and Scanner for Data Transfer" project represents an innovative solution aimed at simplifying and enhancing the process of sharing information and data. In today's digital era, efficient and seamless data transfer is a critical need across various domains. This project leverages the power of Quick Response (QR) codes to create a streamlined and user-friendly mobile application. By developing a comprehensive platform that encompasses QR code generation, scanning, and flexible data transfer options, this initiative addresses the limitations of traditional methods while embracing the convenience of digital exchange.

1.1 Background

In today's rapidly evolving digital landscape, efficient and seamless data transfer has become a fundamental requirement across various domains, ranging from personal communication to business operations. Traditional methods of data transfer, such as manual entry or file sharing, often prove to be time-consuming, error-prone, and limited in their scope. These limitations have spurred the need for innovative solutions that can bridge the gap between cumbersome manual processes and the ease of digital information exchange.

This backdrop has paved the way for the emergence of QR (Quick Response) codes as an innovative solution for simplifying data transfer and information sharing. They are commonly used to transfer data quickly and easily using smartphones and other devices with QR code scanning capabilities.

In response to these challenges, the "QR Share: QR Code Generator and Scanner for Data Transfer" project emerges as a pioneering initiative. This project is rooted in the recognition of Quick Response (QR) codes as a versatile and efficient means of encapsulating information in a machine-readable format. QR codes have found widespread use in various industries and applications, from marketing and inventory management to educational materials and ticketing systems.

The project's core intention is to simplify data transfer and information sharing by harnessing the capabilities of QR codes. QR codes are two-dimensional barcodes, it is a type of matrix barcode that contains information encoded within a pattern of black squares on a white background. QR codes were first invented in 1994 by a Japanese company named Denso Wave. Since then, they have gained popularity due to their ability to store a large amount of data and their ease of scanning. It can encode a wide range of data types, including text, URLs, contact information, and more. With the proliferation of smartphones equipped with QR code scanning capabilities, these codes provide an accessible and user-friendly medium for exchanging information.

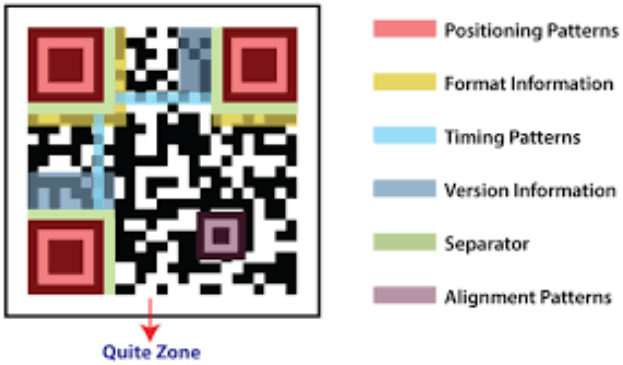


Figure 1: QR Code

The "QR Share" project stands out for its comprehensive approach, combining QR code generation, scanning, and various data transfer options into a single, integrated platform. This holistic design addresses the need for a streamlined and versatile system that enables users to create, scan, and share QR codes effortlessly. By offering features such as QR code customization, flexible data transfer methods, and seamless integration with existing workflows, the project aims to redefine the paradigm of data sharing.

The project's primary functionalities include QR code generation, QR code scanning, and data transfer. Users can effortlessly generate QR codes by inputting various types of data, such as text, URLs, email, websites, phone numbers, SMS, contact information and more, enabling easy sharing with others. Additionally, it provides a QR code scanning feature to effortlessly retrieve shared data. The QR codes serve as a secure and efficient medium to encapsulate the encoded data. Through the integrated QR code scanner, users can easily scan and decode information from QR codes using their device's camera. QR Share offers flexible options for data transfer and information passing. Users can choose to transfer the decoded data to a server for storage, save it

locally on their device, send it via email or messaging apps or display it directly within the application's user interface. This versatility empowers users to seamlessly integrate the project with their preferred workflow and data management systems.

The "QR Share" project builds upon existing work in the realm of QR code technology and information sharing. Previous studies and developments have recognized the potential of QR codes in various domains:

- Education and Engagement
- Marketing and Customer Engagement
- Inventory Management and Automation

These studies collectively emphasize the multifaceted utility of QR codes beyond mere data transfer. QR codes have proven effective in enhancing engagement, bridging physical and digital experiences, and optimizing processes. The "QR Share" project aligns with these insights by addressing limitations in existing QR code generator and scanner apps and focusing on user customization, ease of use, and comprehensive data transfer options.

1.2 Objectives

- User-Friendly Interface: To develop a user-friendly app with an intuitive interface for generating customized QR codes containing different data types.
- Efficient Scanning: To enable quick and accurate scanning of QR codes to retrieve shared digital resources.
- Diverse Data Transfer: To enable flexible data transfer options.
- Data Security: Relevant data protection and privacy regulations to ensure the secure handling of user data and generated QR codes
- History Tracking: Includes a history feature to manage and reference previously generated/scanned QR codes.
- Favourite Feature: Includes a Favourite feature to add the QR code as Favourite for easy access.

1.3 Purpose, Scope, & Applicability

1.3.1 Purpose

- Address Challenges: The purpose is to tackle the limitations of traditional data transfer methods by leveraging QR code technology to create a streamlined and efficient platform for sharing information.
- Enhance Data Sharing: The project aims to simplify data sharing across personal and professional domains, facilitating quick and secure exchange of diverse information.
- Bridging Manual and Digital: By offering a user-friendly app for QR code generation and scanning, the project bridges the gap between manual data entry and digital information exchange.
- User-Centric Approach: The purpose is to prioritize user needs, customization, and data security to redefine how individuals and businesses transfer information.

1.3.2 Scope

1.3.3 Applicability

The “QR Share” project holds both direct and indirect applications which impacts both the computer world and individual.

Direct Applications:

- Efficient Data Sharing: The project directly addresses the need for efficient and streamlined data sharing. Individuals and businesses can use the app to quickly exchange a wide range of information, from contact details to URLs, improving communication.
- User-Friendly Interface: The user-centric design of the app ensures that even individuals with limited technical expertise can benefit from QR code technology, making it accessible to a broader user base.
- Data Protection Compliance: The app's emphasis on data protection and compliance ensures secure information sharing.

Indirect Applications:

- Advancing QR Code Technology: By offering a comprehensive QR code solution, the project contributes to the advancement of QR code technology as a versatile tool for modern data sharing.

- Enhancing Digital Literacy: The project indirectly promotes digital literacy by simplifying the process of using QR codes, making it more accessible to individuals who might be less familiar with advanced technology.
- Boosting Efficiency in Businesses: The project indirectly enhances business operations by providing a tool that streamlines data sharing. This can lead to improved productivity and streamlined communication within organizations.

The "QR Share" project aligns with the computer world by offering an innovative solution that leverages QR code technology. It demonstrates the potential of integrating technology for real-world applications and showcases how computing tools can be harnessed to create efficient, user-friendly solutions.

The project directly benefits individuals by simplifying their data sharing processes. It empowers users to quickly exchange information without the hassle of manual input. Additionally, the project indirectly contributes to digital literacy by introducing users to QR code technology, enhancing their overall digital skills.

1.4 Achievements

I have gained a comprehensive understanding of QR code technology, app development methodologies, and user-centered design principles. The project has deepened my knowledge of data transfer challenges and innovative solutions. In terms of contributions, the "QR Share" app offers a streamlined platform for generating, scanning, and sharing QR codes, simplifying data transfer for diverse applications. During the development phase of the QR Share project, significant knowledge has been gained in various aspects of android application development and data transfer technologies. This project has provided insights into the intricate functionalities of QR codes, their generation, and their role in simplifying data transfer processes. Additionally, it has expanded understanding about data transfer methods, user interface design, and data security considerations within android applications.

Exploring the QR code technology has uncovered its versatility, from encoding different data types to its applicability in various domains. The project's ongoing nature has allowed for continuous learning and adaptation, ensuring that the final product aligns seamlessly with the evolving requirements and user expectations. While the project is still in progress, the achievements so far serve as a strong foundation for the successful completion and future

enhancements of QR Share. The project will successfully meet its goals by developing a user-friendly app that achieves efficient QR code generation and scanning, versatile data transfer options, customization features, and data protection compliance. The project's outcomes have not only met but also exceeded the original objectives, enhancing the potential impact of QR codes in simplifying information exchange in the digital realm.

Chapter 2

System Analysis

2.1 Existing System

Traditional methods of data transfer present several challenges that the "QR Share" project aims to overcome.

2.1.1 Conventional Data Transfer Methods

2.1.1.1 Manual Data Entry

Manual data entry involves the painfully long process of typing information again and again, which is time-consuming and prone to error. This approach lacks efficiency, especially for longer pieces of data, and may result in inaccuracies during the transfer process.

2.1.1.2 File Sharing

File sharing methods often via email attachments or cloud storage services can be cumbersome for quick information exchange. This approach is not well-suited for sharing different types of data quicky, and may not support real-time data updates.

2.1.1.3 Text Messaging

Text messaging platforms are commonly used for sharing short pieces of information. However, they are limited in their ability to handle various data formats and may not be suitable for more complex data transfer scenarios.

2.1.1.4 Contact Sharing

Apps designed for sharing contacts simplify the process for that specific task. However, these apps lack versatility when it comes to sharing a broader range of data types, limiting their usefulness in diverse scenarios.

2.2 Proposed System

The proposed system is centered around the development of the "QR Share" application. This application will harness the capabilities of QR code technology to revolutionize the process of data sharing. Users will experience a seamless and efficient platform that empowers them to generate personalized QR codes containing various data types, such as text, URLs, contact details, and more.

The application's user-friendly interface will facilitate easy QR code scanning, enabling users to quickly retrieve shared data. Moreover, the app's versatility extends to data transfer options, allowing users to choose between various methods, including email, messaging apps, and local storage. Customization features will enable users to personalize QR codes, enhancing branding opportunities for businesses. By providing a user-friendly interface and versatile functionalities, the proposed system aims to bridge the gap between conventional data transfer methods and modern digital exchange.

2.3 Requirement Analysis

The requirement analysis phase critically examines the needs and expectations of users, encompassing both functional and non-functional aspects.

2.3.1 Functional Requirements:

- User Guidance and Intuitive Interface: The application's interface is designed to be intuitive, guiding users seamlessly through QR code generation, scanning, and sharing processes. User-friendly navigation and clear instructions enhance the user experience.
- QR Code Generation and Scanning: The system facilitates effortless creation of QR codes with diverse data types. It ensures efficient scanning using device cameras, guaranteeing accuracy and speed in retrieving shared resources.
- Efficient Data Transfer: The application offers flexible data transfer methods, including sharing decoded data through messaging apps, email, and local storage. Users can select the most suitable channel for sharing.

- Data Protection and Compliance: The system prioritizes data security by securely handling and transmitting user data. It adheres to relevant data protection regulations, ensuring user privacy and legal compliance.

2.3.2 Non-Functional Requirements:

- Performance: The application exhibits optimal performance in QR code generation, scanning speed, and data transfer. Swift response times enhance user experience.
- Reliability: The app is dependable, ensuring accurate and scannable QR codes. It consistently facilitates successful data transfer across various channels.
- Usability: The user interface is designed for ease of use, catering to users of different technical backgrounds. Clear instructions, intuitive design, and straightforward navigation enhance usability.
- Security: The system implements robust security measures to protect user data and QR codes from unauthorized access or misuse. Encryption and data integrity are prioritized.
- Compatibility: The application functions seamlessly across a variety of devices, ensuring broad user reach. It is compatible with different operating systems and screen sizes.
- Scalability: As the user base grows, the application handles increased usage and data traffic without compromising performance or user experience.

The requirement analysis phase aims to comprehensively define the functional and non-functional aspects of the QR Share Project. By addressing user needs, ensuring seamless functionality, and prioritizing security and usability, the project can proceed with a clear understanding of its objectives.

2.4 Hardware Requirements

Hard Disk- A minimum 4 GB or more,

RAM- 4 GB or more.

2.5 Software Requirements

Frontend: Android Studio (Java), XML

Backend: SQLite

2.6 Justification of Platform

- **Justification of Hardware Requirement:**

A minimum of 4 GB of RAM and 4 GB of storage is recommended for the smooth and efficient development of the app using Android Studio. These specifications ensure that the development environment remains responsive and capable of handling the app's complexities, contributing to a productive development process.

- **Justification of Software Requirement:**

Android Studio is the primary integrated development environment (IDE) for Android app development. It provides a rich set of tools and features for designing, coding, debugging, and testing Android applications. It will be used for the frontend development of the app, including the user interface (UI) design and interaction logic.

Android Studio provides a comprehensive development environment for Android app development, including tools for designing UI, writing Java code, and testing your app on emulators or physical devices

Java is the primary programming language for Android app development in Android Studio. Java would be used to build the user interface, handle user interactions, and manage the app's frontend logic.

XML is used for designing the app's user interface layout using Android XML files. You'll create XML layout files to define the structure and appearance of app screens.

SQLite is a lightweight and embedded database management system that is widely used in Android app development. It provides a simple and efficient way to store and manage data locally within the app. SQLite will be used as the backend database for storing relevant information, such as scanned QR code data or user preferences

Chapter 3

System Design

3.1 Problem Definition

In today's digital age, the exchange of information between individuals, organizations, and devices is fundamental. Traditional methods of data transfer, such as manual data entry or file sharing, are often cumbersome, time-consuming, and prone to errors. There is a clear need for a solution that simplifies and streamlines the process of sharing information across various domains, from personal communication to business operations. This project aims to address this challenge by leveraging QR (Quick Response) codes as a versatile tool for data transfer and information sharing.

Sub-Problems:

1. Inefficient Data Transfer Methods: Manual data entry and traditional sharing methods are slow, error-prone, and lack versatility in handling different data types.

- Manual Data Entry: Users struggle with the time-consuming process of manually inputting data for sharing.
- Limited Data Types: Existing methods are often restricted in their ability to handle diverse data types such as text, URLs, contact information, and more.

2. Complexity in Sharing and Retrieving Data: Existing solutions for sharing and retrieving data are often disjointed and lack user-friendly interfaces.

- Scanning Challenges: Users encounter difficulties in accurately and swiftly scanning QR codes to retrieve shared data.
- Ineffective Data Transfer: Data transfer options are limited, leading to inconvenience for users in selecting the appropriate method for sharing.

3. Lack of Customization and Branding: Users have limited options for customizing QR codes to reflect their identity or brand.

- Branding Opportunities: Users miss out on branding opportunities when sharing information using generic QR codes.

4. Data Security and Privacy Concerns: With the increasing importance of data security and privacy, users need assurance that their information is handled securely.
- Data Vulnerability: Data shared through QR codes may be vulnerable to unauthorized access or misuse.
- Privacy Compliance: The project must adhere to data protection regulations to ensure user privacy and legal compliance.

5. Lack of User-Friendly History Tracking: Users need a way to manage and reference previously generated and scanned QR codes efficiently.
- Absence of History Feature: Existing applications lack a comprehensive history feature for users to track and manage QR codes.

By addressing these sub-problems, the QR Share project seeks to redefine the paradigm of data sharing by providing a user-friendly, efficient, and secure platform for QR code generation, scanning, and data transfer. This approach aims to simplify the process of sharing information across various contexts while offering users customization options and robust data security measures

3.2 Planning and Scheduling

Planning and scheduling are critical aspects of software development that ensure the project progresses systematically and efficiently. In the case of the QR Share project, planning involves breaking down the project into smaller tasks and considering constraints that dictate when these tasks can be executed. Scheduling, on the other hand, assesses the availability of resources to execute the planned tasks. Below, I'll explain the planning and scheduling process, along with a detailed Gantt chart, emphasizing the use of the iterative model.

- Planning:

In the planning phase, we outline all the tasks required to achieve the project's goals. This involves identifying what needs to be done, who is responsible for each task, and when these tasks should occur. Constraints, such as dependencies between tasks and available resources, are taken into account.

- Scheduling:

Scheduling involves determining whether the necessary resources, including human resources, hardware, and software, are available to execute the planned tasks. This ensures that the project progresses smoothly and on time.

- Gantt Chart:

The Gantt chart below illustrates the project's timeline and tasks using an iterative development model. The iterative model is chosen for its flexibility and adaptability, allowing for incremental development and refinement of features over time. This approach aligns with the project's complexity and evolving requirements.

Month / Phase	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR
Project Planning									
Requirement Gathering									
Analysis and Design									
Implementation									
Testing									
Evaluation									
Documentation									

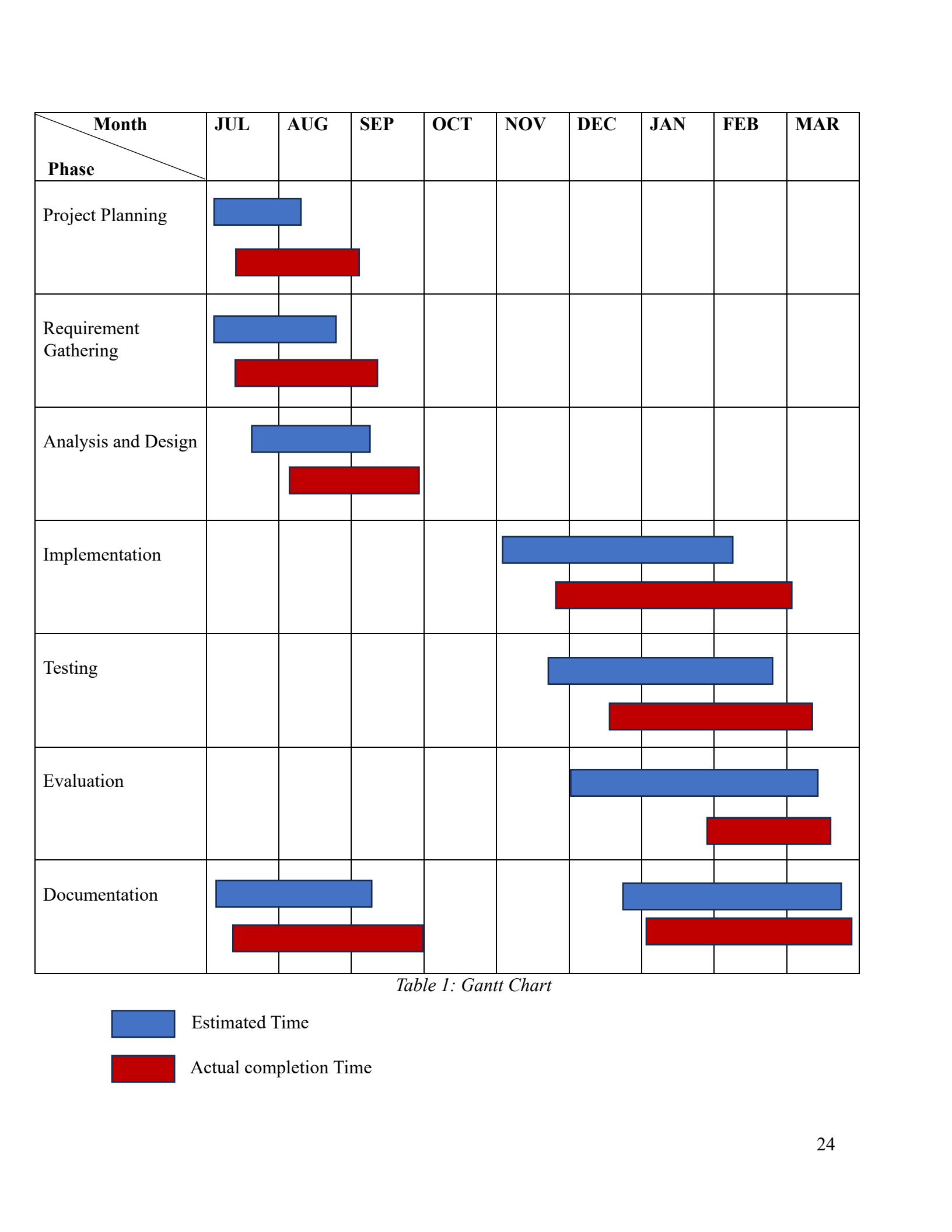


Table 1: Gantt Chart

Estimated Time

Actual completion Time

- Gantt Chart Description:
  The project begins on July 2023.

Each phase is represented as a task block, and their durations are estimated in days.
The project goes through several iterative phases: Project Planning, Requirements Gathering, Analysis and Design, Implementation, Testing, Evaluation, Documentation.

Each phase may overlap with the preceding and following phases, showcasing the iterative nature of development.

Extensive documentation is carried out concurrently with development phases.
This Gantt chart outlines the project's timeline, tasks, and the iterative nature of development. It aids in visualizing the project's progression and ensures efficient resource allocation throughout the development process.

3.3 Preliminary Product Description

The Preliminary Product Description for the QR Share project offers a comprehensive overview of the system's requirements, objectives, functions, and operations.

- Project Requirements and Objectives:

  The QR Share application is designed to address the pressing need for simplified and efficient data transfer and information sharing in today's digital landscape. The core objectives include the development of a user-friendly mobile application with an intuitive interface for generating customized QR codes encompassing various data types. The application must facilitate quick and precise scanning of QR codes for retrieving shared digital resources. Flexible data transfer options are essential, ensuring users can choose the most suitable method for sharing decoded data. Customization, enhance the visual appeal and branding possibilities. Data security is paramount, with stringent measures in place to protect user data and generated QR codes. Additionally, a history tracking feature will assist users in managing and referencing previously generated and scanned QR codes.

- Functions and Operations:

  The QR Share application combines several key functions into a seamless and integrated platform. Users can effortlessly create QR codes by inputting different data

types, enabling easy sharing with others. The application offers QR code scanning capabilities, ensuring the accurate retrieval of shared digital resources through device cameras. To enhance user experience and brand identity, the app allows customization of QR. Robust data security measures guarantee the safe handling and transmission of user data and QR codes. The history tracking feature simplifies user interactions by providing easy access to previously generated and scanned QR codes. By amalgamating these functions, the QR Share app aims to redefine the paradigm of data sharing, bridging the gap between manual data entry and digital information exchange.

In summary, the QR Share project is driven by the essential requirements of user-friendliness, efficiency, customization, data security, and comprehensive data transfer options. It seeks to revolutionize data sharing across a wide range of contexts, enhancing user experiences in personal and professional spheres

3.4 Conceptual Models

For the "QR Share: QR Code Generator and Scanner for Data Transfer" project, conceptual models play a crucial role in understanding the problem domain, defining system operations, and establishing the allowable sequences of these operations. Conceptual models provide a blueprint for system development, ensuring that the project aligns with its objectives and requirements. These models help describe the operations that can be performed on the system and the allowable sequences of those operations. In this project, several conceptual models can be employed to facilitate the development process.

- **Iterative Development Model:**

The project will adopt the Iterative Development Model due to its suitability for projects that require refinement and enhancement over time. In this model, development occurs in small, manageable iterations, each building upon the previous one. Here's how it applies to the QR Share project:

What is the Iterative Model?

- The Iterative Model is a software development approach where the project is divided into small increments or iterations.

- Each iteration represents a complete development cycle, including planning, design, implementation, testing, and evaluation.
- The project evolves through repeated cycles, with each iteration improving upon the previous one.

Why Use the Iterative Model in QR Share?

- Complexity Management: The QR Share project involves various components, from user interfaces to backend databases. The iterative model allows for the gradual development and refinement of these components, making it easier to manage the complexity.
- User-Centric Development: QR Share's success depends on meeting user expectations. The iterative model enables continuous user feedback and refinement, ensuring that the final product aligns closely with user needs.
- Adaptability: The QR Share landscape may evolve during development due to technological advancements or changing user requirements. The iterative approach allows for flexibility and adaptation to these changes.
- Refinement: QR Share requires ongoing refinement to meet user needs and preferences effectively.
- Early Feature Delivery: It allows for early delivery of essential features and continuous improvement based on user input.

Iterative Model Design:

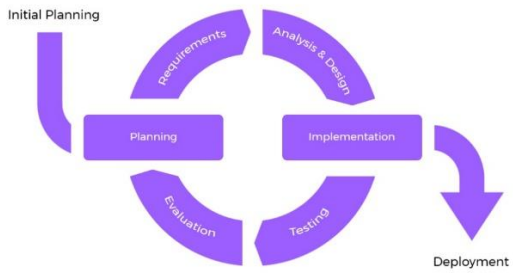


Figure 2: Iterative Model Design

Phases of the Iterative Model:

- Project Planning: Define the project scope, objectives, and initial requirements. Plan iterations and allocate resources.
- Requirement Gathering: Collect and document user requirements, including QR code generation, scanning, and customization needs.
- Analysis and Design: Create detailed system design based on user requirements, including UI design, data structures, and algorithms.
- Implementation: Develop the Android application in Android Studio, focusing on frontend (Java and XML) and backend (SQLite) development.
- Testing: Rigorously test the application to ensure it meets functional and non-functional requirements.
- Evaluation: Collect user feedback, analyse application performance, and identify areas for improvement. After completion of all the previous steps, the project team will evaluate the whole project which will then be handed over.
- Refinement: Based on evaluation results and user feedback, refine the application in subsequent iterations.
- Deployment: after Refinement (or Evaluation) the project would be finally deployed.

3.5 Module Division

Module division involves breaking down a software project into smaller, manageable components or modules. Each module serves a specific function or set of related functions, making it easier to develop, test, and maintain the application. In the context of the QR Share project, the module division can be outlined as follows:

- User Authentication Module: This module handles the user registration, login, and authentication processes.
- User Interface Module: This module is responsible for the user interaction aspects of the application. It includes sub-modules for the main menu, QR code generator screen, scanner screen, and user settings, etc.

- QR Code Generation Module: This module handles the generation of QR codes with various data types
- QR Code Scanning Module: This module focuses on scanning QR codes using the device's camera and decoding the scanned data.
- Data Transfer Module: This module enables the flexible transfer of data to various channels, such as messaging apps and email.
- Data Storage Module: This module is responsible for storing user-generated QR codes and scanned data securely.
- Data Security Module: Ensures the secure handling and encryption of user data and QR codes.
- History Tracking Module: Manages the history feature, allowing users to access previously generated and scanned QR codes.

3.6 Data Dictionary

Entity	Attributes	Description
User	Username	The chosen username of the user.
	Password	The password associated with the user's account.
	Email	User's email address.
	FirstName	User's first name.
	LastName	User's last name.
Admin	AdminID	A unique identifier for Admin.
	AdminUsername	The chosen username of the Admin.
	Password	The password associated with the admin's account.
	Email	Admin's email address.
QRCode	QRCodeID	A unique identifier for each QR code.
	CodeContent	The encoded information stored in the QR code
	CodeType	The type of QR Code (text, url, etc.)
	DateCreated	The date when the QR code was generated.
	QRCodeImage	The image of the Generated/Scanned QR Code
	Favorite	The Favourite QR Code identifier
	UserID	The ID of the user who generated the QR Code.

History	HistoryID	A unique identifier for History of QR Codes.
	QRCodeID	The ID of the Generated QR Code.
	Action	Description of the action performed (e.g., QR code generation, scan).
	Timestamp	The date and time when the action occurred
	UserID	The ID of the User who access the history.
Favourite	FavoriteID	A unique identifier for Favourite QR Codes.
	QRCodeID	The ID of the QRCode that was marked as a Favourite
	CodeContent	The Content of the Favourite QR Code
	CodeType	The Type of the Favourite QR Code
	Timestamp	The date and time when the action occurred
	QRCodeImage	The Image of the Favourite QR Code
	UserID	The ID of the user who marked as a Favourite

Table 2: Data Dictionary

3.7 E-R Diagrams

An entity relationship model, also called an entity- relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

It is blueprint of database.

ER Diagrams contain different symbols that use

1. rectangles to represent entities,
2. ovals to define attributes and
3. diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Purpose of ERD

- The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
- The ERD serves as a documentation tool.
- Finally, the ERD is used to connect the logical structure of the database to users.

In particular, the ERD effectively communicates the logic of the database to users

Components:

- Entity: A definable thing-such as a person, object, concept, or event-that can have data stored about it. Think of entities as nouns. An entity is denoted as a rectangle in an ER diagram
- Relationship: How entities act upon each other or are associated with each other. Think of relationships as verbs.
- Attribute: A property or characteristic of an entity. Often. shown as an oval or circle.
- Cardinality: Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many.

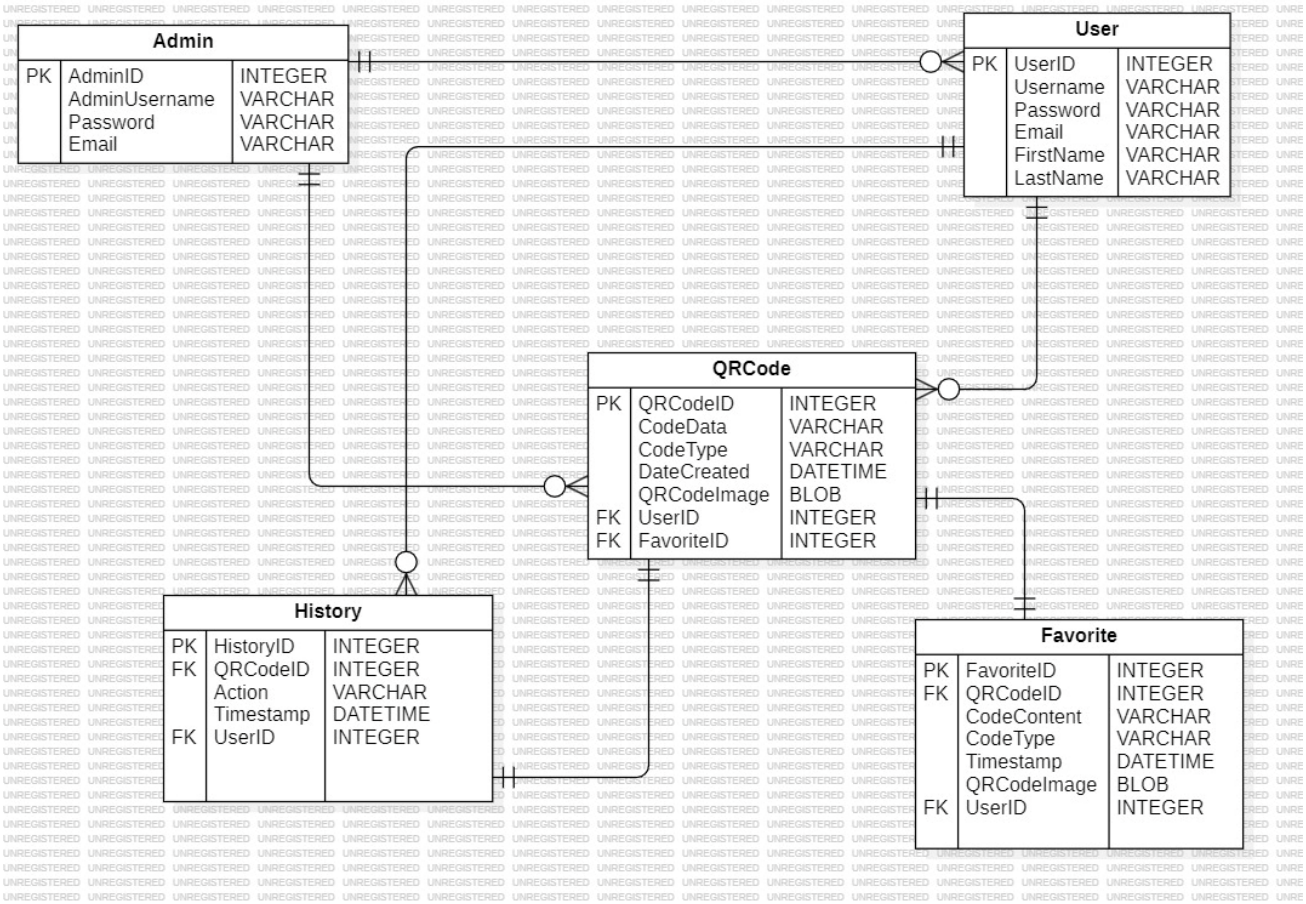


Figure 3: ER Diagram

The QR Share Project's Entity-Relationship Diagram (ERD) defines the essential components of the system for sharing QR codes and data. The core entities include User, QRCode, Favorite, and History. QRCode store data, its type, creation dates, their owners, etc. Scan capture who, when, and which QR code was scanned, while Share details the sender, method of transfer. History records actions like QRCode generation and Scanning history (Time, date) and Favourite keeps the tract of QRCode and ScanQRCode which are added in favourite. Relationships indicate how users, admins, and QR codes interact.

3.8 UML Diagrams

3.8.1 Use-Case Diagram:

A use case diagram is a graphical representation of the interaction between the elements of a system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

The second thing which is relevant to use cases are the actors. Actors can be defined as something that interacts with the system. Actors can be a human user, some internal applications, or may be some external applications. When we are planning to draw a use case diagram, we should have the following items identified.

A use case diagram is a visual representation in the Unified Modelling Language (UML) that illustrates how a system interacts with external entities, also known as actors, to achieve specific functionalities or goals. It's a powerful tool in software engineering and system design for capturing and communicating the functional requirements of a system from a user's perspective.

Key components of a use case diagram include:

- Actors: Actors represent the external entities that interact with the system. Actors can be users, other systems, or even hardware devices. Each actor plays a specific role and initiates one or more use cases.

- Use Cases: Use cases are represented as ellipses and describe specific functionalities or actions that the system performs. They represent a set of interactions between the system and one or more actors to achieve a particular goal.
- Associations: Lines connecting actors and use cases illustrate the relationships or interactions between them. These associations show which actors are involved in which use cases.
- System Boundary: The system boundary, often depicted as a box or rectangle, represents the scope of the system being modelled. It encloses all the use cases and actors relevant to that system.

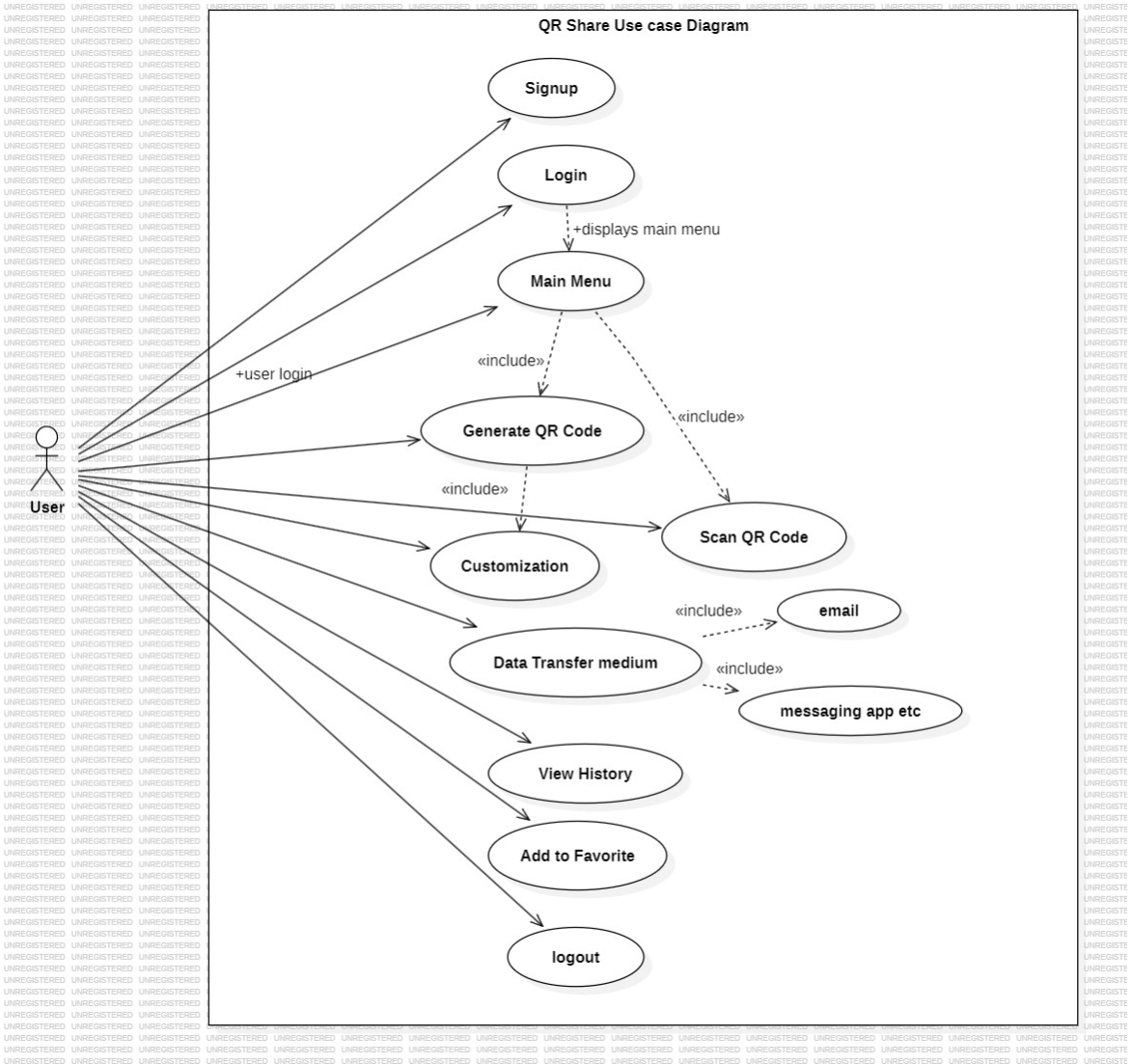


Figure 4: Use case Diagram

This use case diagram illustrates the interactions between actors (User and Admin) and the various use cases within the QR Share Android application project. Users can perform actions related to QR codes, manage their history, and mark QR codes as favorites. Admins have access to an admin dashboard for managing the app's content and user accounts, etc. The diagram provides an overview of the app's functionality and how users and admins interact with it.

3.8.2 Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The basic purpose of activity diagrams is to capture the dynamic behaviour of the system. It is also called object-oriented flowchart.

This UML diagram focuses on the execution and flow of the behaviour of a system instead of implementation. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagram deal with all type of flow control by using different elements such as fork, join, etc. Activity diagrams consist of activities that are made up of actions that apply to behavioural modelling technology

Activity Diagram Notations:

- Initial State: The starting state before an activity takes place is depicted using the initial state. A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State
- Action or Activity State: An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically, any action or event that takes place is represented using an activity.
- Action Flow or Control flows: Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another. An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow

- Decision node and Branching: When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.
- Guards – A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.
- Fork – Fork nodes are used to support concurrent activities. n When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities.
- . Join – Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.
- Merge or Merge Event – Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chose

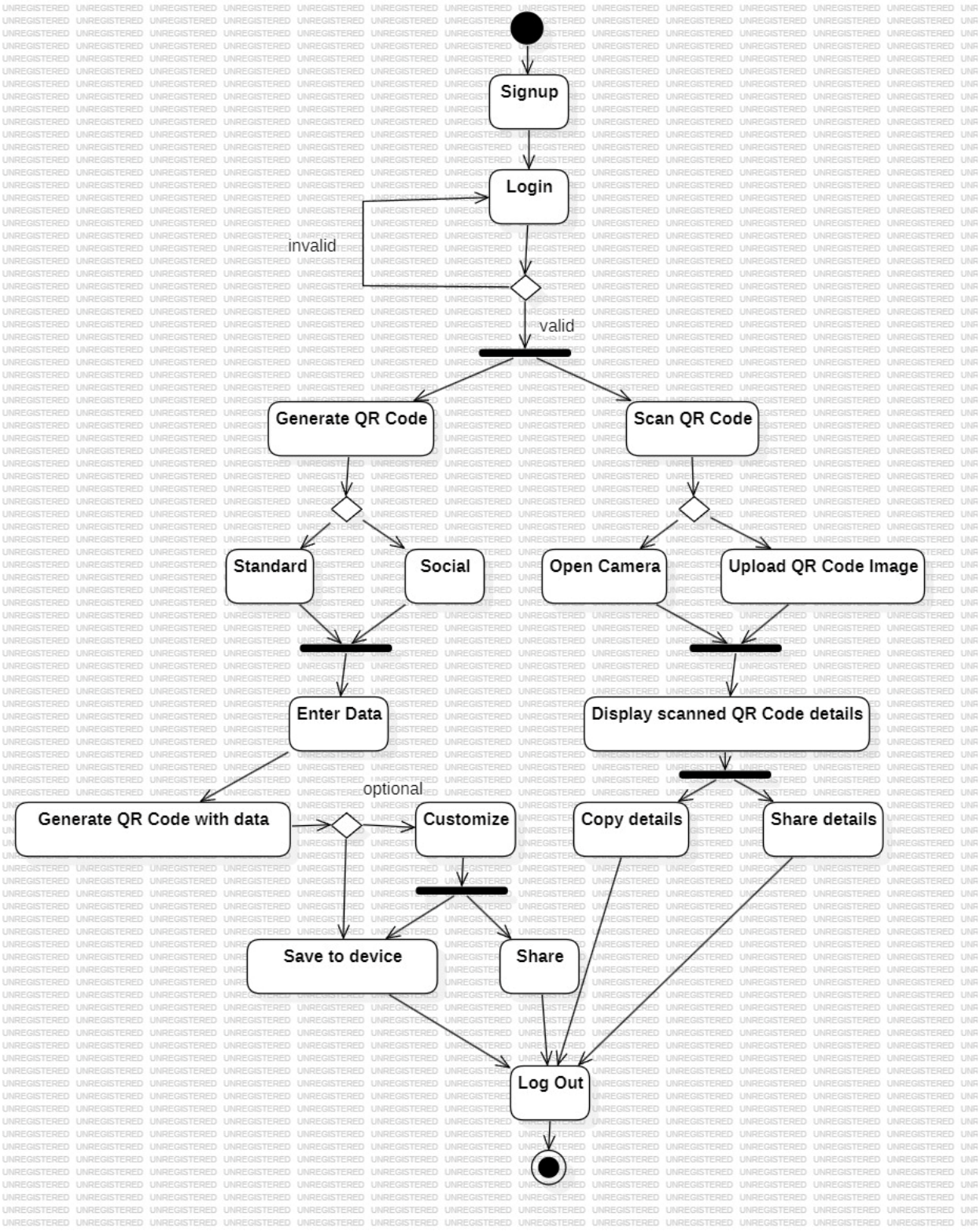


Figure 5: Activity Diagram

This activity diagram represents the data transfer process within the QR Share app. It outlines the major activities and decision points involved in generating, scanning, and sharing QR codes.

3.8.3 Sequence Diagram:

A sequence diagram is the most commonly used interaction diagram.

Interaction diagram: An interaction diagram is used to show the interactive behaviour of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

Sequence Diagrams: A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations:

- Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- Lifelines: A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. We display a lifeline in a rectangle called head with its name and type. The head is located on top of a vertical dashed line.
- Messages: Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram

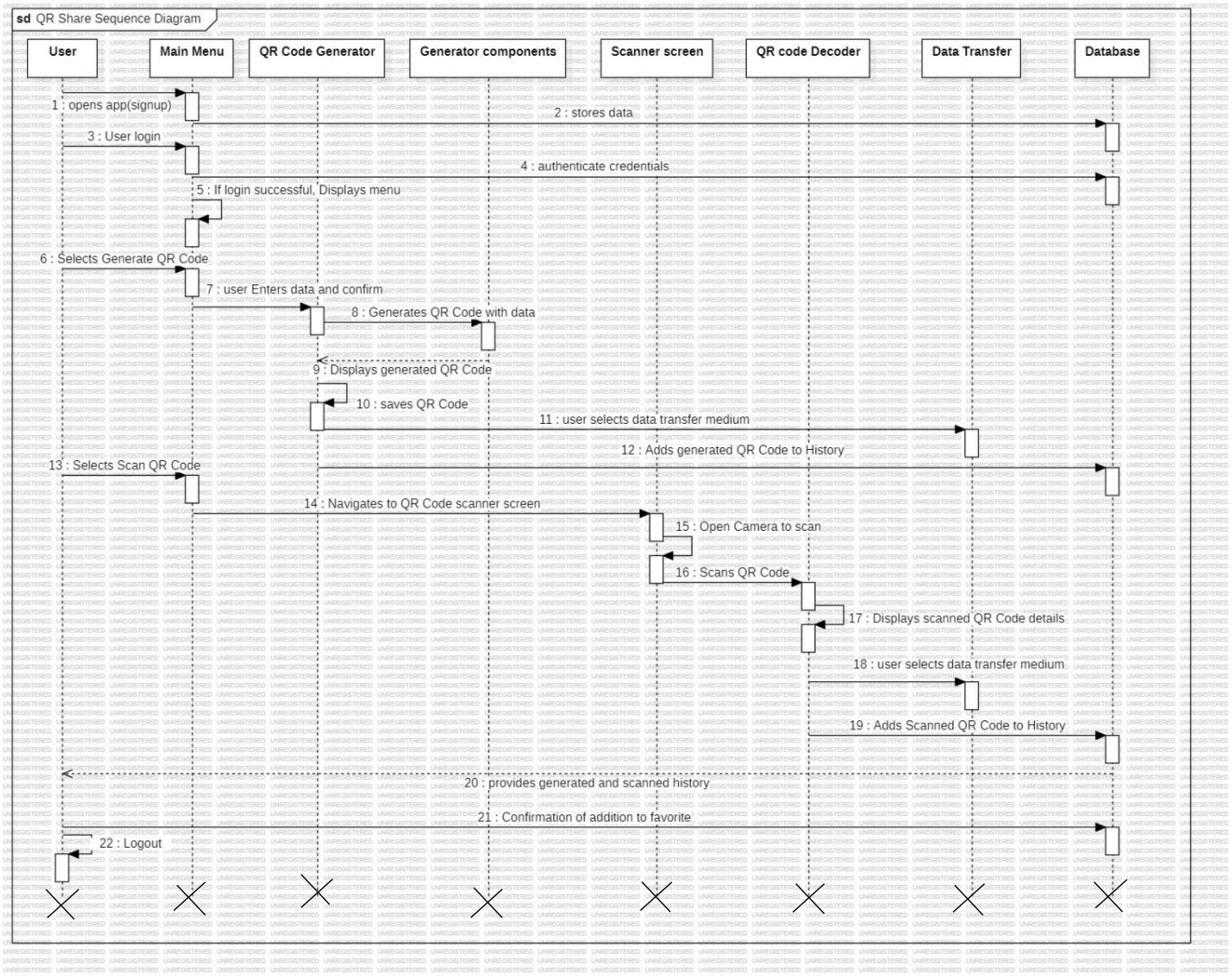


Figure 6: Sequence Diagram

This sequence diagram illustrates the user's interaction with the QR Share application. The diagram showcases various components including the user interface, QR code generation, QR code scanning, QR code decoder, data transfer activities and the database.

The "Data Transfer" lifeline indicates the initiation of data transfer activities, which can activate specific messaging or email apps based on user choices.

This sequence diagram provides a high-level overview of the QR Share app's data transfer functionality, showing how users can generate, scan, and share QR codes seamlessly.

3.8.4 Class Diagram:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Classes: Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes. Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition (left-aligned, not bolded, and lowercase), and write operations into the third

Relationships In UML, relationships are of three types:

- Dependency: A dependency is a relationship between two or more classes where a change in one class causes changes in another class. It forms a weaker relationship.
- Generalization: A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class. For example, The Current Account, Saving Account, and Credit Account are the generalized form of Bank Account
- Association: It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship. For example, a department is associated with the college.

Aggregation: An aggregation is a subset of association, which represents has a relationship. It is more specific than association.

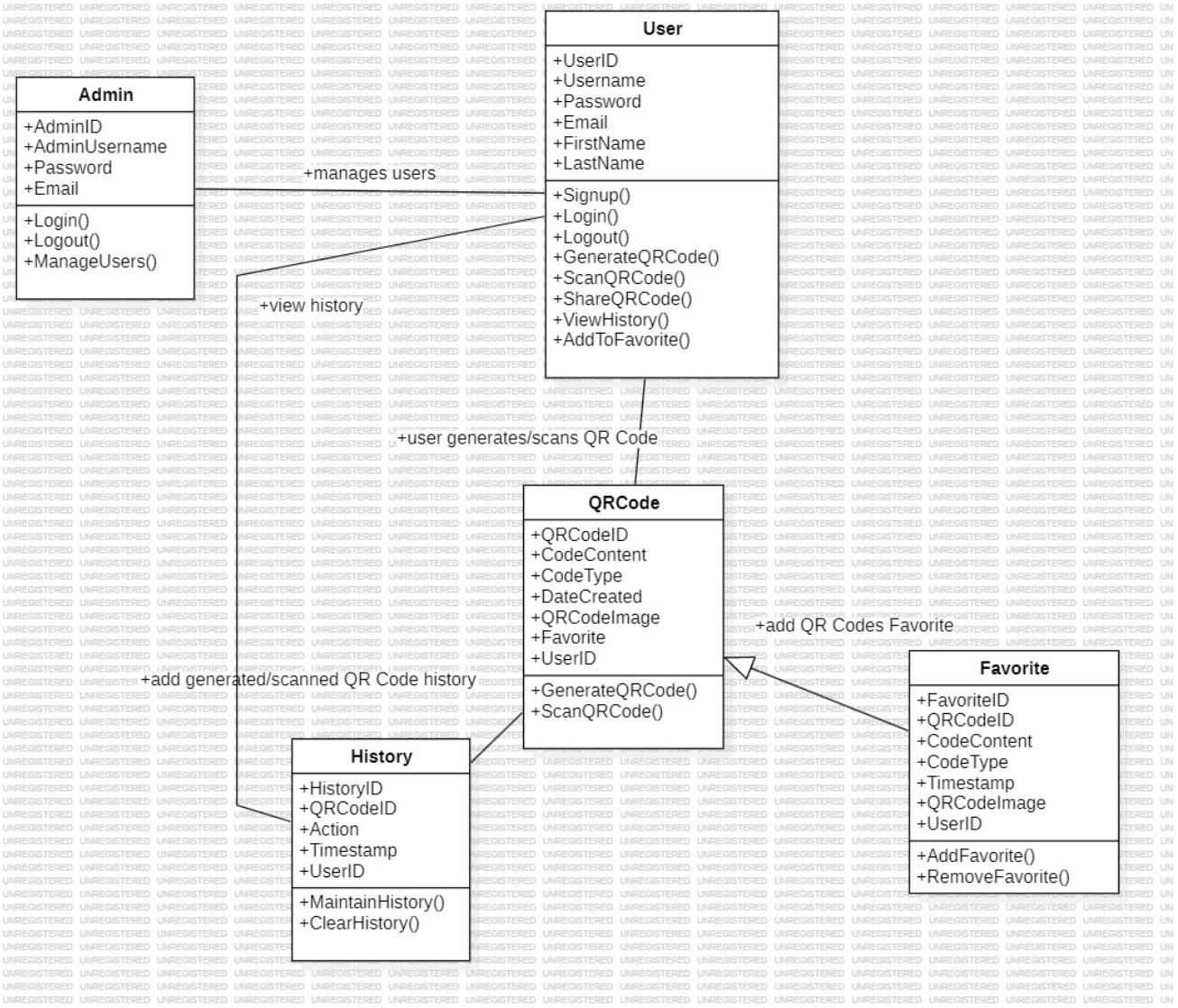


Figure 7: Class Diagram

The QR Share Class Diagram provides a detailed representation of the core components and interactions within the QR Share project. It comprises several essential classes, including User and Admin, representing users and administrative personnel, respectively. Users have the capability to generate, customize, scan, share QR codes, and view their action history. Admins possess the added privilege of managing users. QRCode class represents the QR codes created within the system, while ScanQRCode, Share, and Customization classes signify scanning events, data transfer events, and customization events, respectively. Lastly, the History and Favorite class maintains a record of Generated & Scanned QR Code history and Favourite QR Code in the system respectively.

This class diagram serves as a blueprint for the QR Share project, outlining the classes, their attributes, operations, and associations that govern the system's functionality and data flow. It offers a clear understanding of how users interact with QR codes and system's architecture, including data objects, user interactions, and administrative control, facilitating a clear understanding of the QR Share project's functionality.

3.8.5 Object Diagram:

In UML, object diagrams provide a snapshot of the instances in a system and the relationships between the instances. By instantiating the model elements in a class diagram, you can explore the behaviour of a system at a point in time.

An object diagram is a UML structural diagram that shows the instances of the classifiers in models. Object diagrams use notation that is similar to that used in class diagrams. However, while class diagrams show the actual classifiers and their relationships in a system, object diagrams show specific instances of those classifiers and the links between those instances at a point in time. You can create object diagrams by instantiating the classifiers in class, deployment, component, and use-case diagrams.

Before drawing an object diagram, the following things should be remembered and understood clearly −

- Object diagrams consist of objects.
- The link in object diagram is used to connect objects.
- Objects and links are the two elements used to construct an object diagram.

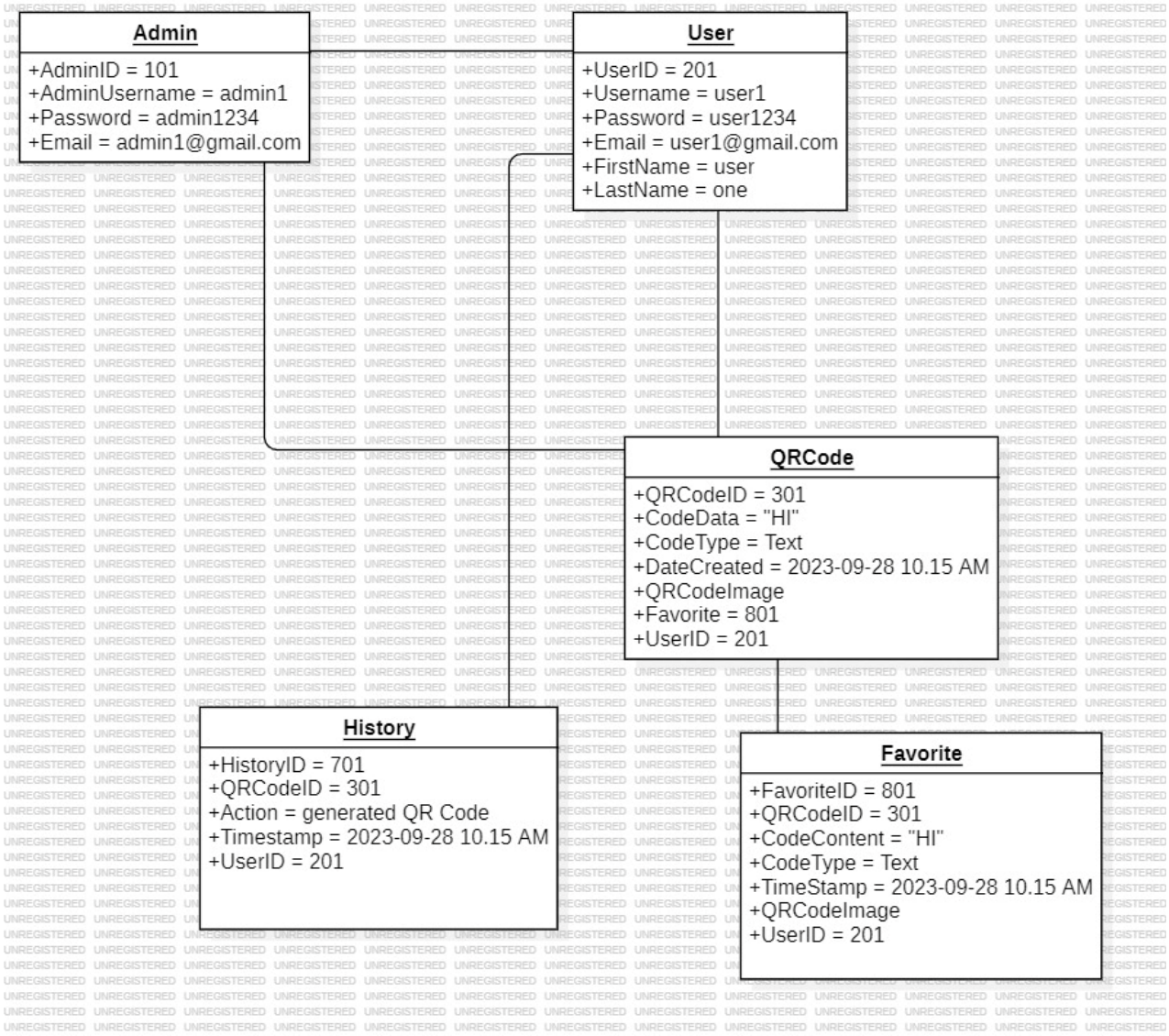


Figure 8: Object Diagram

The object diagram for the QR Share project includes classes representing various entities in the system, such as User, Admin, QRCode, ScanQRCode, Customization, History and Favorite. These classes have slots that match the ones described in the data dictionary. The diagram also shows associations between these classes, indicating their relationships, such as Admin managing Users, Users generating QR codes, Users scanning QR codes, and Users having a history of actions. This object diagram provides a visual representation of instances of these classes and their relationships within the QR Share system.

The Object Diagram for the QR Share project illustrates instances of key classes and their attributes, offering a snapshot of how users interact with QR codes and the system's functionalities. Users have their unique UserIDs and personal information. QRCode showcase individual QR code instances with details like QRCodeID, CodeData, CodeType, DateCreated, UserID and CustomizationID. ScanQRCode represent scanning events, while Share denotes a data transfer event, each capturing relevant details. Customization represents a QR code customization event. History record user actions related to QR codes. Favourite records the QR Codes which are added as Favourite. This Object Diagram provides a visual representation of these instances and their relationships, offering a concise overview of the system's functioning.

3.8.6 Component Diagram:

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behaviour is explained by the provided and required interfaces.

Key Concepts of Component Diagrams:

- Components: In a component diagram, components represent modular and self-contained units of software or hardware within a system. Components can range from large software modules to smaller, reusable classes or libraries. Each component encapsulates a set of related functionalities and may expose interfaces for interaction with other components.
- Interfaces: Interfaces define how components interact with each other. They specify a contract or set of methods that a component should provide to other components. Interfaces are depicted as small squares on the edge of a component, connected by lines to indicate dependencies or associations.

- Dependencies: Dependencies represent relationships between components, indicating that one component relies on another for its functionality. Dependencies can be directed (with an arrow) or undirected, depending on the nature of the relationship.
- Provided and Required Interfaces: Components often provide interfaces that define the services they offer, and they require interfaces from other components to access external functionality. These provided and required interfaces are typically labelled on the component shapes.

In summary, component diagrams are valuable tools for visualizing and designing the architecture of software systems. They help software architects and developers understand, organize, and manage the components of a system, facilitating better communication and decision-making during the software development process.

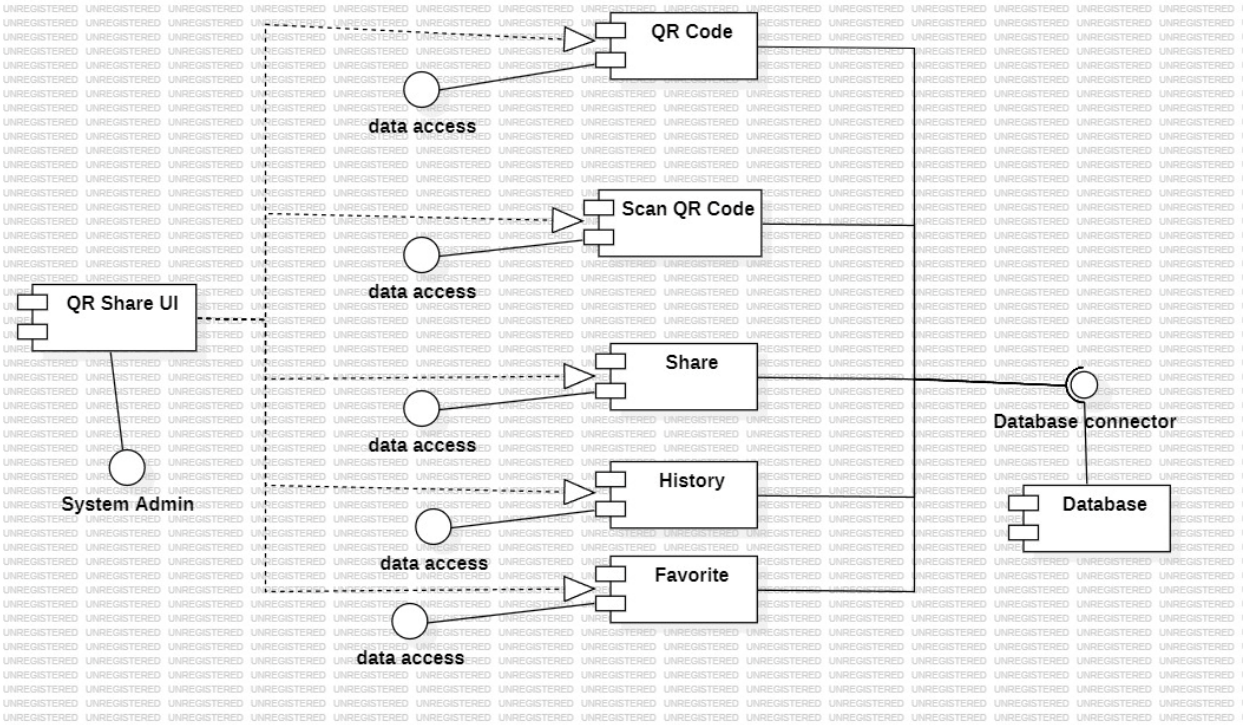


Figure 9: Component Diagram

A component diagram for the QR Share project provides a visual representation of the system's architecture and how its various components interact. In this diagram, key components include the User Interface, QR Code, Customize QR Code, QR Code Scanner, Database, Share,

History and Favourite. The User Interface component serves as the user-facing part of the application, connecting users to the rest of the system. The QR Code and Scanner components handle the creation and scanning of QR codes, while the Database stores user data and QR codes. The Share component manages different data transfer methods, and the History and Favourite allows the user to access history and Favourite QR Codes respectively. These components communicate through interfaces, specifying how they interact and share information. This component diagram offers a clear overview of how different elements work together to provide the functionalities of the QR Share project.

3.8.7 Collaboration Diagram:

Collaboration Diagram shows the object organization, it is also known as "Communication Diagram".

- In this diagram the method call sequence is indicated by some numbering technique
- The numbers indicate how the methods are called one after another
- The method calls are similar to that of a sequence diagram.
- But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization

Element and its description:

- Object: The objects interacting with each other in the system. Depicted by a rectangle with the name of the object in it, preceded by a colon and underlined.
- Relation/Association: A link connecting the associated objects. Qualifiers can be placed on either end of the association to depict cardinality.
- Messages: An arrow pointing from the commencing object to the destination object shows the interaction between the objects. The number represents the order/sequence of this interaction

Collaboration diagrams are particularly useful for visualizing and understanding the sequence of interactions between objects in a dynamic context. They are valuable during the

design phase of software development when developers need to illustrate how different components or objects collaborate to achieve a specific task or behaviour.

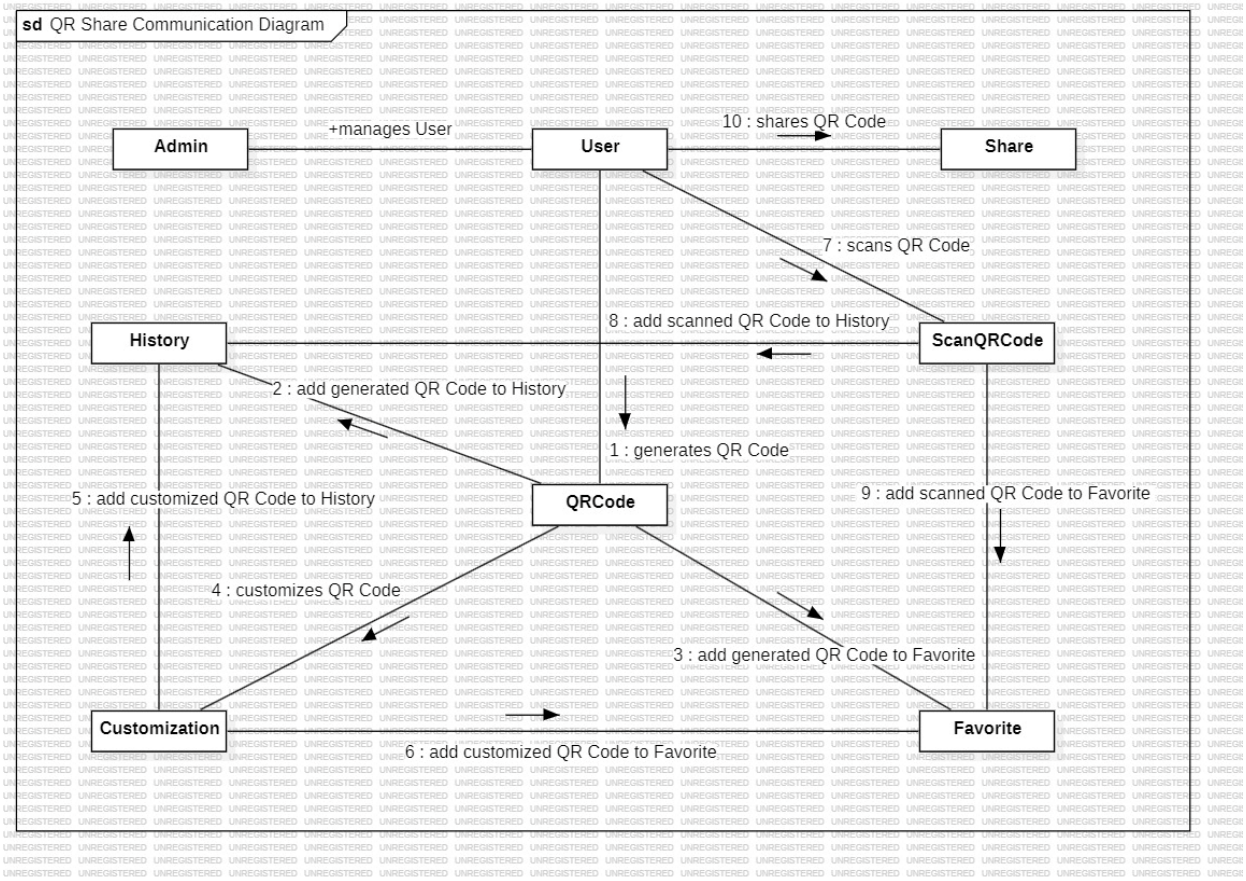


Figure 10: Collaboration Diagram

A collaboration diagram for the "QR Share: QR Code Generator and Scanner for Data Transfer" project would visually represent how various components or objects within the system interact to achieve specific functionalities. In this project, there are several key entities, including the QR Code Generator, QR Code Customization, QR Code Scanner, Users, Admins, Data Transfer Services, History and Favorite.

The diagram would illustrate how these entities collaborate to ensure smooth user experiences. For instance, the User interacts with the QR Code Generator, Scanner, sharing components for QR Code generation, scanning and sharing purpose. The Share Services come into play for data transfer, allowing users to share coded and decoded data conveniently.

Chapter 4

Implementation & Testing

4.1 Code

- **Generate_Fragment.java**

```
public class Generate_Fragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_generate_, container, false);
        // Initialize and set onClick listeners for each image view representing different types of QR codes
        // Each image view corresponds to a specific type of QR code (e.g., text, website, contact, etc.)
        // When clicked, it starts the respective activity to generate the corresponding QR code
        // Text Activity
        ImageView img_text = view.findViewById(R.id.img_text);
        img_text.setOnClickListener(v -> startActivity(new Intent(getActivity(), TextActivity.class)));

        // Website Activity
        ImageView img_website = view.findViewById(R.id.img_website);
        img_website.setOnClickListener(v -> startActivity(new Intent(getActivity(), WebsiteActivity.class)));
```

```
        // Contact Activity
        // Similar onClick listeners are set for other image views corresponding to different types of QR codes

        return view;
    }
    // onResume method is overridden to set the title of the action bar when the fragment is resumed
    // The action bar title is set to "GENERATE"
    @Override
    public void onResume() {
        super.onResume();
        if (getActivity() != null) {
            Objects.requireNonNull(((AppCompatActivity) getActivity()).getSupportActionBar()).setTitle("GENERATE");
        }
    }
}
```

- **TextActivity.java**

```
public class TextActivity extends AppCompatActivity {
  // handling variables

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_text);
        // Set the initial title in onCreate or wherever needed
        getSupportActionBar().setTitle("Generate");
        // Initialize DatabaseHelper
```

```
        databaseHelper = new DatabaseHelper(this);

        //btn_create click
        btn_create.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                String text = etxt_text.getText().toString();
                if(text.trim().isEmpty()){
                    Toast.makeText(TextActivity.this, "Please enter text",
Toast.LENGTH_SHORT).show();
                }
                else{
                    String textInfo = "Text: " + text+"\n";

                    try{
                        bitmap = textToImageEncode(textInfo);
                        img_gen_qr_code.setImageBitmap(bitmap);

                        //for database
                        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                        bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream);
                        byte[] imageByteArray = outputStream.toByteArray();

                        // Insert QR code details into database
                        String timestamp = new SimpleDateFormat("yyyy/MM/dd_HH:mm:ss",
Locale.getDefault()).format(new Date());
                        databaseHelper.insertQRCode(textInfo, "GENERATED : Text", timestamp,
imageByteArray);
```

```
                    // Method to get the URI of the bitmap
                    private Uri getImageUri(TextActivity textActivity, Bitmap bitmap) {
                        ByteArrayOutputStream bytes = new ByteArrayOutputStream();
                        bitmap.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
                        String path =
MediaStore.Images.Media.insertImage(textActivity.getContentResolver(), bitmap,
                                "QR_Code", null);
                        return Uri.parse(path);
                    }
                });

            }
            catch(WriterException e){
                e.printStackTrace();
                Toast.makeText(TextActivity.this, "Failed to generate QR code",
Toast.LENGTH_SHORT).show();

            }
        }
    }
});
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_WRITE_EXTERNAL_STORAGE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            // Permission granted, proceed with saving the image
            saveImageToGallery(bitmap);
```

```
            } else {
                // Permission denied, show a message or take appropriate action
                Toast.makeText(TextActivity.this, "Permission denied, unable to save image", Toast.LENGTH_SHORT).show();
            }
        }
    }
//to save image to gallery
    private void saveImageToGallery(Bitmap bitmap) {
        // Create a directory to save the image
        File imagesFolder = new File(getExternalFilesDir(null), "Images");
        if (!imagesFolder.exists()) {
            imagesFolder.mkdirs();
        }
        String timestamp= new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
        String fileName = "QR_Code_" + timestamp+ ".jpg";

        // Save the bitmap to a file
        File file = new File(imagesFolder, fileName);
        try {
            OutputStream fos = new FileOutputStream(file);
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos);
            fos.flush();
            fos.close();
            // Update the media store so the image appears in the gallery app
            MediaStore.Images.Media.insertImage(getContentResolver(), file.getAbsolutePath(), fileName, null);
            Toast.makeText(TextActivity.this, "Saved to Device", Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
```

```
            e.printStackTrace();
            Toast.makeText(TextActivity.this, "Failed to save image",
Toast.LENGTH_SHORT).show();
        }
    }
//text to image conversion method
    private Bitmap textToImageEncode(String value) throws WriterException{
        BitMatrix bitMatrix;
        try{
            bitMatrix = new MultiFormatWriter().encode(value,
                    BarcodeFormat.DATA_MATRIX.QR_CODE, QRCodeWidth, QRCodeWidth,
null);
        }
        catch(IllegalArgumentException e){
            return null;
        }

        int bitMatrixWidth = bitMatrix.getWidth();
        int bitMatrixHeight = bitMatrix.getHeight();
        int[] pixels = new int[bitMatrixWidth * bitMatrixHeight];

        for (int y = 0; y < bitMatrixHeight; y++){
            int offSet = y * bitMatrixWidth;
            for(int x = 0; x < bitMatrixWidth; x++){
                pixels[offSet + x] = bitMatrix.get(x, y)?

getResources().getColor(R.color.black):getResources().getColor(R.color.white);
                }
        }
        Bitmap bitmap = Bitmap.createBitmap(bitMatrixWidth, bitMatrixHeight,
Bitmap.Config.ARGB_4444);
```

```
        bitmap.setPixels(pixels, 0, 500, 0, 0, bitMatrixWidth, bitMatrixHeight);
        return bitmap;
    }
}
```

This code snippet defines the TextActivity class, which generates a QR code from text input. It includes functionality to convert text to a QR code bitmap, save the generated QR code image to the device gallery, and handle permission requests for saving images. Additionally, it sets the initial title of the activity and initializes a database helper. The textToImageEncode method converts text into a QR code bitmap, saveImageToGallery saves the bitmap image to the device gallery, and onRequestPermissionsResult handles permission requests. Similar to this other Activities like WebsiteActivity, EventActivity, SMSActivity will have similar methods and functions.

- **Scan_Fragment.java**

```
public class Scan_Fragment extends Fragment {
    // UI Views
    // Barcode Scanner
    // Constants for permissions
    private static final int CAMERA_REQUEST_CODE = 100;
    private static final int STORAGE_REQUEST_CODE = 101;

    // Arrays of permissions required
    private String[] cameraPermissions;
    private String[] storagePermissions;

    // Uri of the image
    private Uri imageUri = null;

    // Activity result launchers
    private final ActivityResultLauncher<String> requestPermissionLauncher;
```

```
    private final ActivityResultLauncher<Intent> galleryActivityResultLauncher;
    private final ActivityResultLauncher<Intent> cameraActivityResultLauncher;

    // Constructor
    public Scan_Fragment() {
        // Initialize activity result launchers
        requestPermissionLauncher=registerForActivityResult(new
ActivityResultContracts.RequestPermission(), this::handlePermissionResult);
        galleryActivityResultLauncher=registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), this::handleGalleryResult);
        cameraActivityResultLauncher=registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), this::handleCameraResult);
    }

    // Other methods and lifecycle callbacks are omitted for brevity
    // Method to handle camera button click
    private void openCamera() {
        isCameraRequest = true;
        if (checkCameraPermissions()) {
            pickImageCamera();
        } else {
            requestCameraPermission();
        }
    }
    // Method to handle gallery button click
    private void openGallery() {
        isCameraRequest = false;
        if (checkStoragePermissions()) {
            pickImageGallery();
        } else {
            requestStoragePermission();
```

```
        }
    }
    // Method to detect QR code from the selected image
    private void detectResultFromImage() {
        try {
            InputImage inputImage = InputImage.fromFilePath(getContext(), imageUri);
            Task<List<Barcode>> barcodeResult = barcodeScanner.process(inputImage)
                    .addOnSuccessListener(this::extractBarCodeQRCodeInfo)
                    .addOnFailureListener(e -> Toast.makeText(getContext(), "Failed scanning
due to " + e.getMessage(), Toast.LENGTH_SHORT).show());
        } catch (Exception e) {
            Toast.makeText(getContext(), "Failed due to " + e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }
    // Method to extract information from scanned QR codes
    private void extractBarCodeQRCodeInfo(List<Barcode> barcodes) {
        StringBuilder resultBuilder = new StringBuilder();

        // Extract information from each barcode
        for (Barcode barcode : barcodes) {
            // Extract information based on barcode type
            switch (barcode.getValueType()) {
                // Handle different barcode types (WiFi, URL, Email, etc.)
            }
        }
        // Display the result in the UI
        txt_result.setText(resultBuilder.toString());
        // Start the Scan_Result activity to display the scanned result
        Intent intent = new Intent(getContext(), Scan_Result.class);
        intent.putExtra("scanned_result", resultBuilder.toString());
```

```
        intent.putExtra("qr_code_image_uri", imageUri);
        startActivity(intent);

        // Insert scanned QR code details into the database
        String timestamp = new SimpleDateFormat("yyyy/MM/dd_HH:mm:ss",
    Locale.getDefault()).format(new Date());
        DatabaseHelper databaseHelper = new DatabaseHelper(getContext());
        databaseHelper.insertQRCode(resultBuilder.toString(), "SCANNED", timestamp,
    imageByteArray);
    }
    // Other methods and permission handling functions are omitted for brevity
}
```

This code snippet contains the main functionality for scanning QR codes using the device's camera or selecting an image from the gallery. It initializes UI elements, handles button clicks for opening the camera or gallery, and processes the detected QR codes. Additionally, it includes methods for requesting permissions, handling permission results, and extracting information from scanned QR codes.

4.2 Testing Approach

- **Test Case Design**

Index	Test Case	Test Data	State	Test Input Values	Expected Result
1	User Signup	User registration details (name, email, password)	Valid	Valid user registration details Name: abc Email: abc@gmail.com Password: abc123	User account created successfully, redirected to user dashboard

2	User Signup	User registration details (name, email, password)	Invalid	Invalid user registration details (incorrect number, email) Name: abc Email: abcdef Password: a1!!2	Please enter valid details
3	User Login	User login details (email, password)	Valid	Valid user login details Email: abc@gmail.com Password: abc123	User successfully logged in, redirected to user dashboard
4	User Login	User login details (email, password)	Invalid	Invalid user login details (incorrect email, password) Email: abc@gmail.com Password: abc122323	Please enter valid email or password
5	QR Code Generation	User inputs data for QR Code (text, url, email etc.)	Valid	User input data Email: abc@gmail.com	QR code generated with user data, displayed on the screen
6	QR Code Generation	User inputs data for QR Code (text, url, email etc.)	Invalid	User input data phone: 123	Please Enter valid Details
7	QR Code Scanning	QR code image scanned by the user's device camera	Valid	QR code for scanning or selecting QR code from device to scan	QR code decoded, displayed data matches the input data

8	QR Code Scanning	QR code image scanned by the user's device camera	Invalid	Invalid QR Code (Some other image)	Not a valid QR Code
9	Data Transfer	User selects a data sharing method (email, etc.)	Valid	Select data transfer option like: Gmail or messaging app, etc	Data transferred successfully to the selected method
10	History Tracking	User accesses the History feature	Valid	Accesses the history of QR Codes	System displays the user's history of generated and scanned QR codes
11	Favourite Feature	User marks a QR code as a favourite	Valid	Selects a QR code to mark as a favourite	QR code marked as a favourite for the user, stored for future reference
12	Favourite Feature	User removes a QR code from favourite	Valid	Selects a QR code to remove from favourite	QR code get removes from favourite section.
13	User logout	N/A	Valid	User clicks on logout button	User is logged out of the system and redirected to the signup page.

Table 3: Test Cases

4.2.1 Unit Testing

- What is Unit Testing?

Unit testing is a software testing technique that focuses on evaluating individual components or units of code in isolation. A "unit" typically refers to the smallest testable part of an application, such as a function, method, or class. In unit testing, these components are examined independently to verify that they perform as intended.

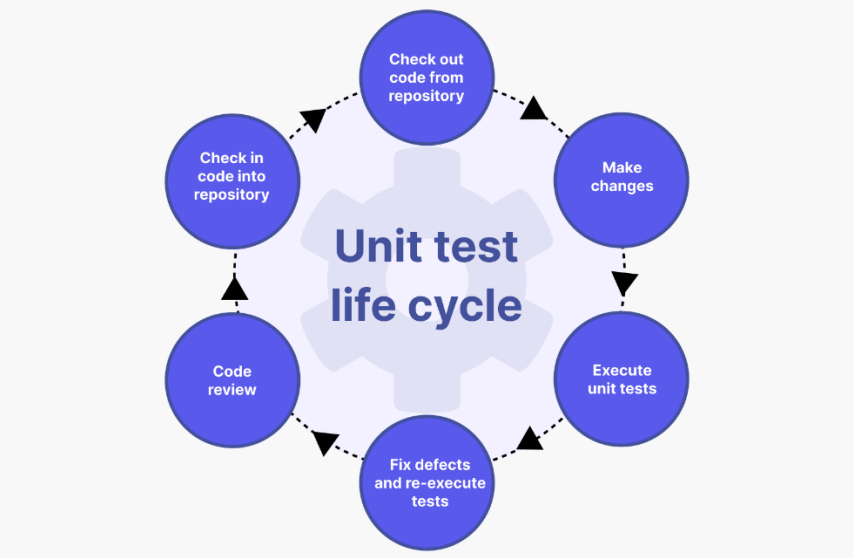


Figure11: Unit test life cycle

- Importance of Unit Testing:

  Unit testing serves several critical purposes within the software development process:

  - Bug Identification: It helps detect and rectify bugs or defects at an early stage, reducing the cost and effort required for later-stage debugging.
  - Isolation of Issues: By testing units in isolation, developers can pinpoint the source of errors more precisely, facilitating quicker resolution.
  - Code Verification: Unit tests validate that individual components conform to their design specifications, ensuring code correctness.
  - Regression Testing: Unit tests can be automatically rerun whenever code changes are made, preventing the introduction of new defects during the development process.

- Why is Unit Testing used?

  Unit testing is employed for various reasons, including:

  - Quality Assurance: It enhances the overall quality and reliability of the software by identifying and fixing issues early in development.
  - Code Refactoring: It provides confidence that code changes or optimizations do not introduce new defects.

- Documentation: Unit tests serve as executable documentation, explaining how a component is intended to function.

4.2.2 Integration Testing

- What is Integration Testing?

Integration testing is a software testing approach that focuses on assessing the interactions between various components or modules of a system. Unlike unit testing, which tests individual units in isolation, integration testing examines how these units work together when integrated into a larger system.

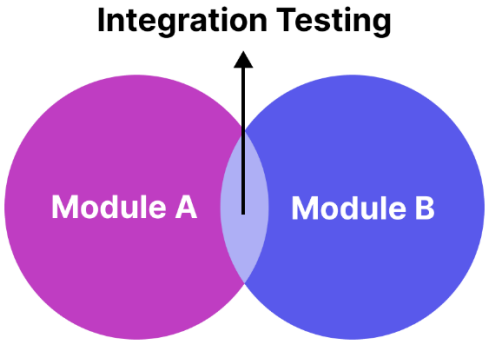


Figure 12: Integration Testing

- Importance of Integration Testing:

Integration testing plays a crucial role in ensuring the reliability and functionality of a software system:

- Interaction Verification: It verifies that different components can interact correctly and share data as expected.
- Data Flow: Integration testing evaluates the flow of data between modules, ensuring that data is processed correctly as it moves through the system.
- Interface Compatibility: It ensures that the interfaces between components are compatible and that data is transmitted accurately between them.

- Why is Integration Testing Used?

Integration testing is employed for several reasons:

- Interaction Validation: It validates that various components interact seamlessly, minimizing the risk of integration-related defects.
- System Validation: It helps assess whether the integrated system meets its design specifications and functional requirements.
- Risk Mitigation: It identifies integration issues early, reducing the likelihood of costly defects emerging in later phases.

In summary, unit testing and integration testing are essential components of the testing approach for the "QR Share: QR Code Generator and Scanner for Data Transfer" project.

Unit testing focuses on individual code units, ensuring their correctness and reliability, while integration testing examines the interactions and data flow between these units within the larger system, ensuring seamless operation.

Both testing types are crucial for identifying and addressing defects early in the development process and ensuring the software meets its design and functional requirements.

Chapter 5

Results and Discussions

5.1 Test Report

Test Case Id	Test Case Description	Test Steps	Expected Results	Pass/ Fail
TC001	User Registration	1. Open the QR Share app 2. Click on 'Sign Up' button 3. Fill in valid details	User should be successfully registered	Pass
TC002	Generate QR Code	1. Open the QR Share app 2. Navigate to 'Generate' section 3. Enter the required information 4. Click on 'Generate' button	QR code generation screen should be displayed QR code should be generated and displayed	Pass
TC003	Scan QR Code	1. Open the QR Share app 2. Navigate to 'Scan' section 3. Position the camera to scan a QR code or select QR Code from Device storage 4. Tap to capture the QR code	QR code scanning screen should be displayed Scanned data should be displayed	Pass

TC004	View History	1. Open the QR Share app 2. Navigate to 'History' section 3. Browse through the list of scanned QR codes 4. Select a QR code entry	History screen should be displayed QR code details should be displayed	Pass
TC005	Add to Favorites	1. Open the QR Share app 2. Click on Favourite button after generating or scanning QR Codes	QR code details screen should be displayed QR code should be added to favorites	Pass
TC006	Remove from Favorites	1. Open the QR Share app 2. Navigate to 'Favorites' section 3. Select a QR code entry 4. Click on 'Remove from Favorites' button	Favorites screen should be displayed QR code should be removed from favorites	Pass
TC007	Clear History	1. Open the QR Share app. 2. Navigate to 'History' section 3. Click on 'Clear History' button 4. Confirm the action	History screen should be displayed Confirmation dialog should appear History should be cleared	Pass
TC008	User Logout	1. Open the QR Share app	User should be logged in	Pass

		2. Navigate to More section 3. Click on 'Logout' button	User should be logged out and redirected to login screen	

Table 4: Test Report

5.2 User Documentation

- **Successful Signup**

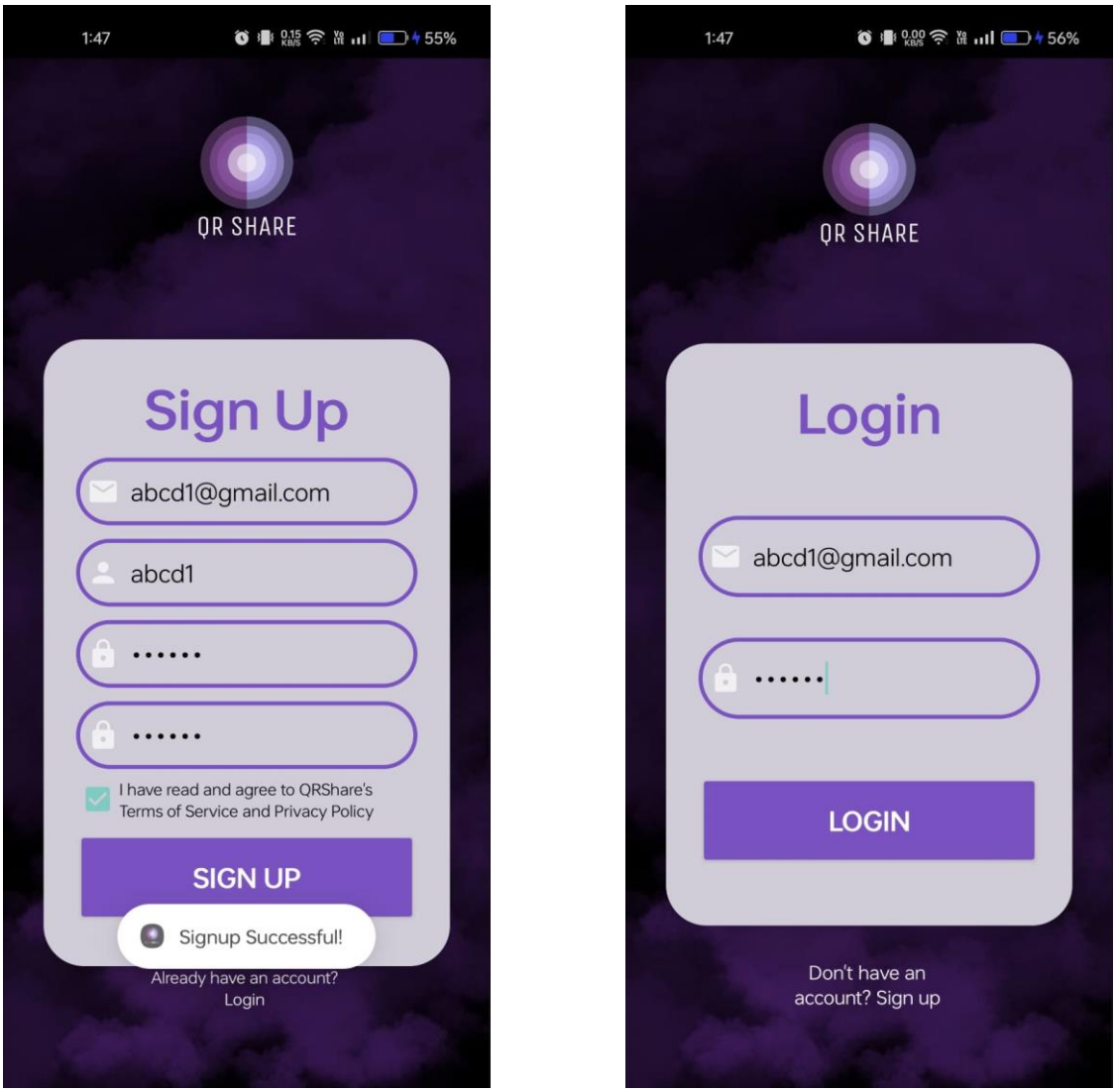


Figure 11: Signup Page

Upon successful signup, the user is directed to the login screen where they can enter their credentials to access the application. The signup process ensures that the user's information is securely stored and can be used for authentication during login.

- **Successful Login**

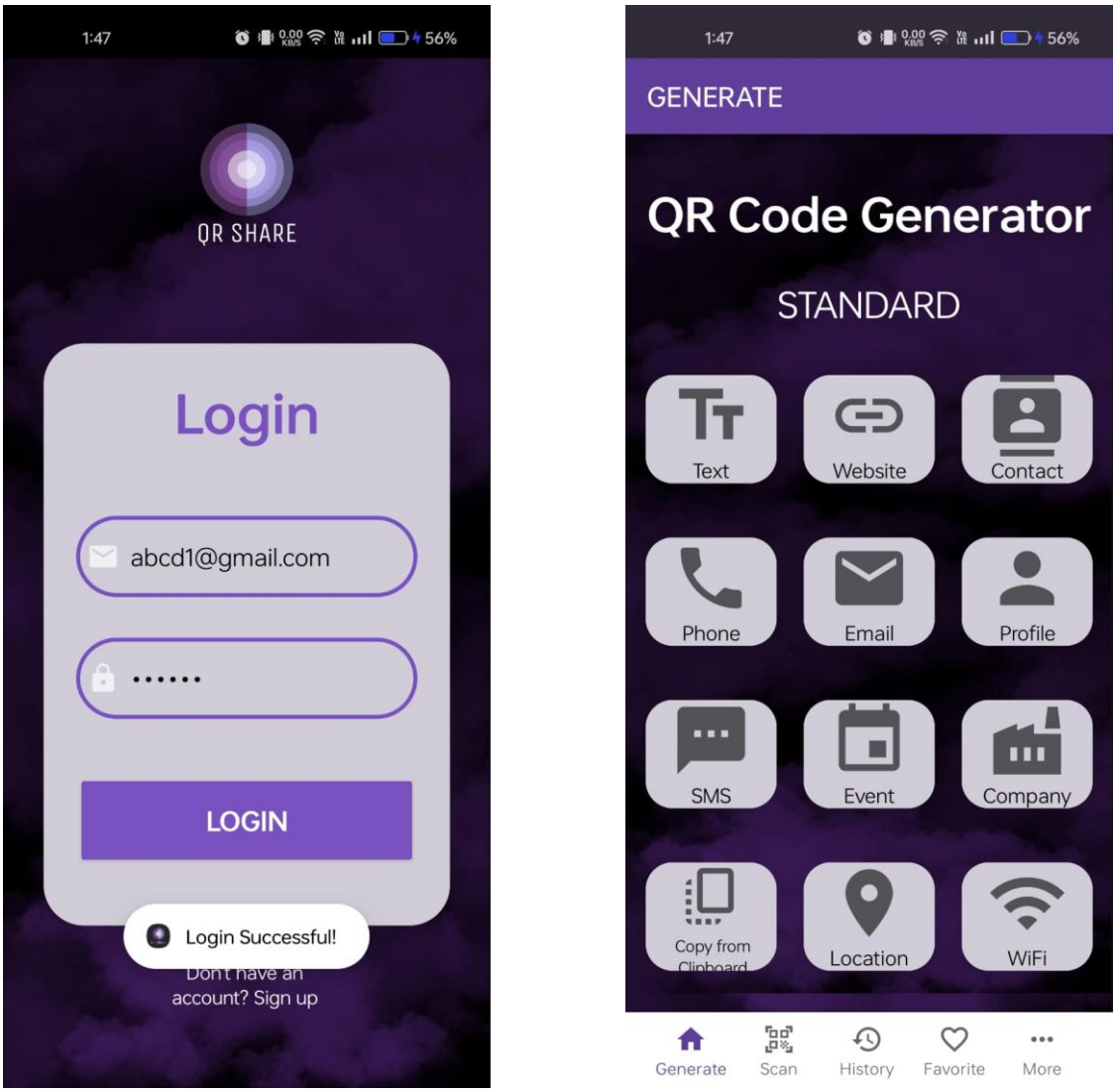


Figure 12: Login Page

After entering valid credentials, the user gains access to the main functionality of the application. They can then proceed to generate, scan, and manage QR codes as per their requirements.

- **Generating QR Codes**

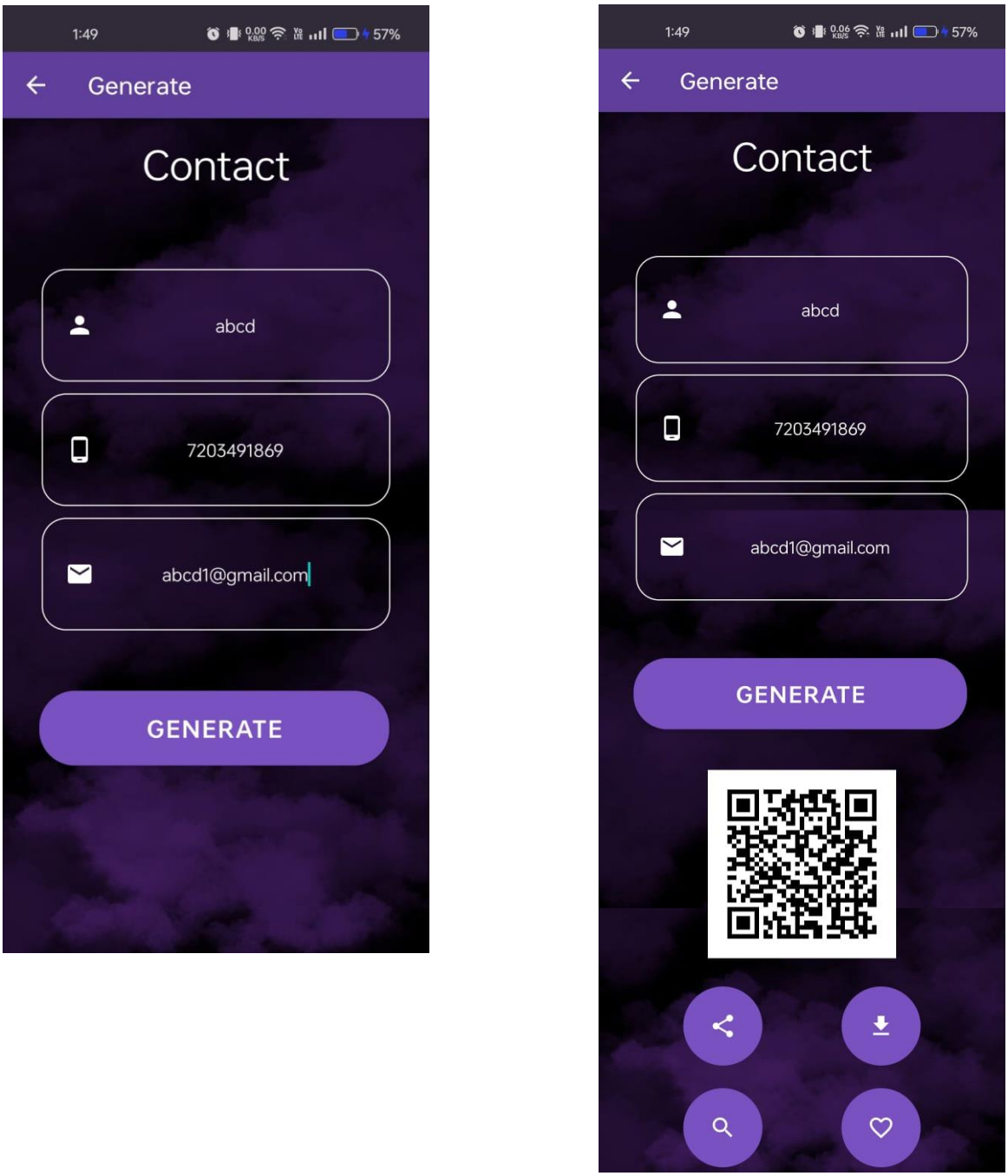


Figure 13: Generate Page

The "Generate QR Code" screen allows users to input text, URLs, contact information, or other data types to generate corresponding QR codes. After entering the desired data, users can click on the "Generate" button to create the QR code, which is then displayed on the screen.

- **Scanning QR Codes**

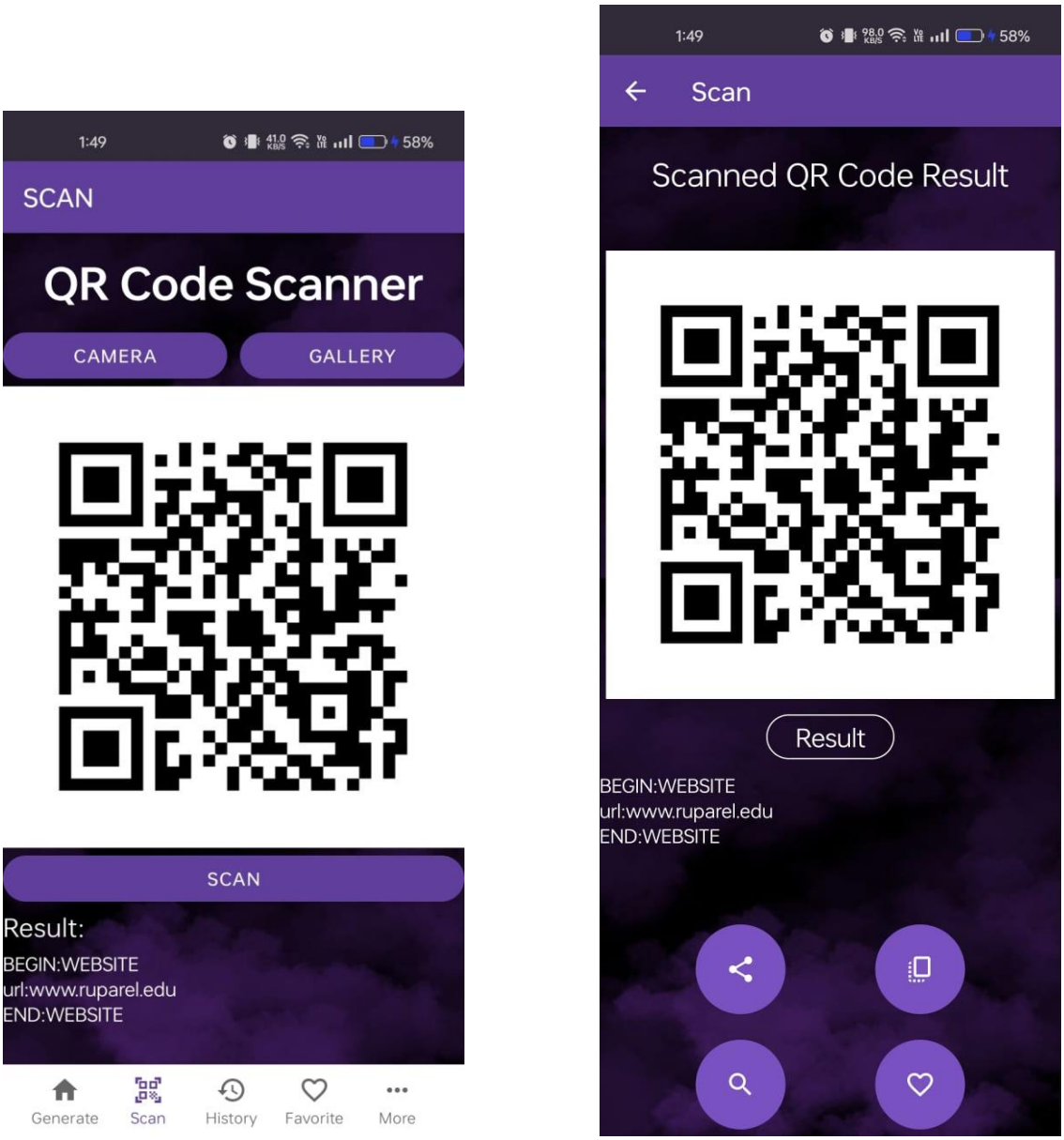


Figure 14: Scan Page

The "Scan QR Code" Fragment enables users to use their device's camera to scan QR codes from physical or digital sources or from Gallery . Upon successfully scanning a QR code, the application extracts the encoded information and presents it to the user for further action.

- **History and Clearing History**

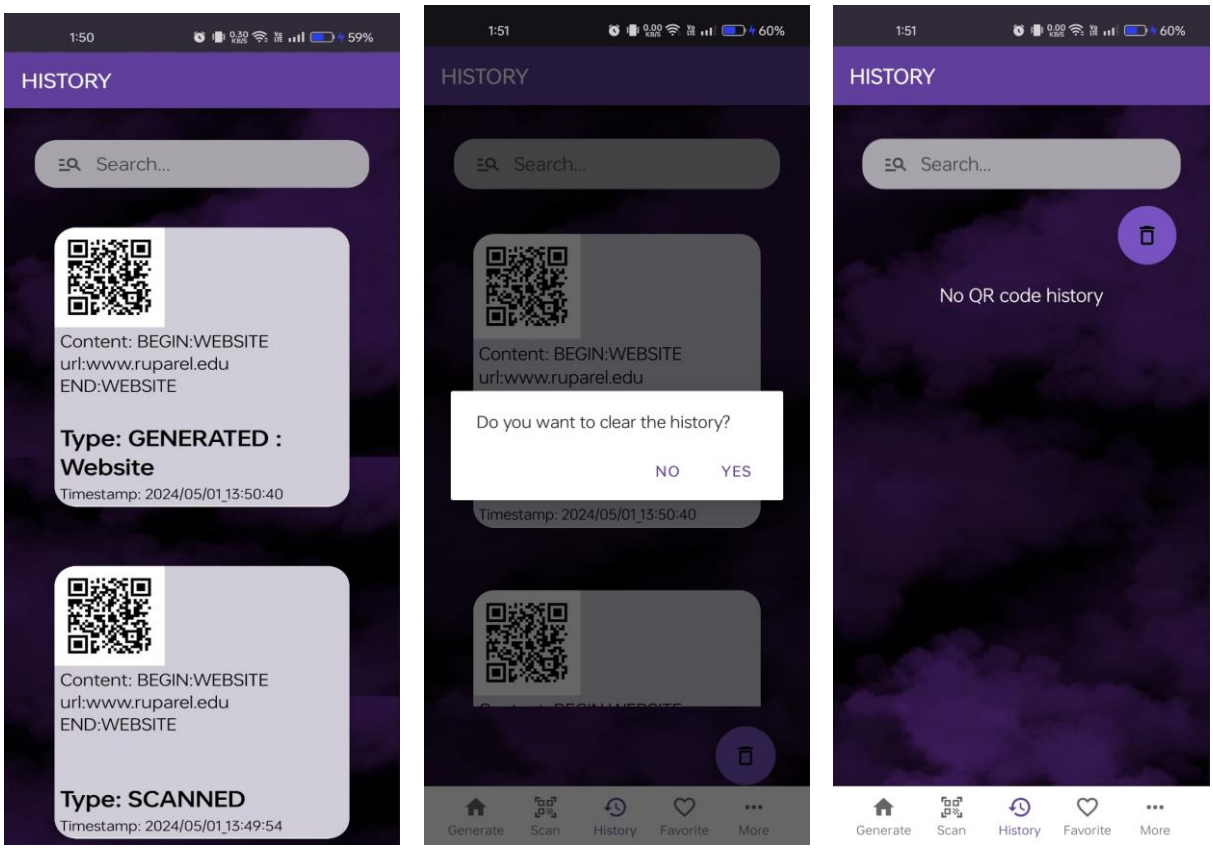


Figure 15: History Page

Whenever a QR code is generated or scanned, the application automatically adds a record to the user's history. This feature allows users to review their previous QR code activities and easily access recently generated or scanned codes.

Users have the option to clear their QR code history if they wish to remove previously saved records. This action helps declutter the history list and provides a fresh start for new QR code activities.

- **Adding to Favourite**

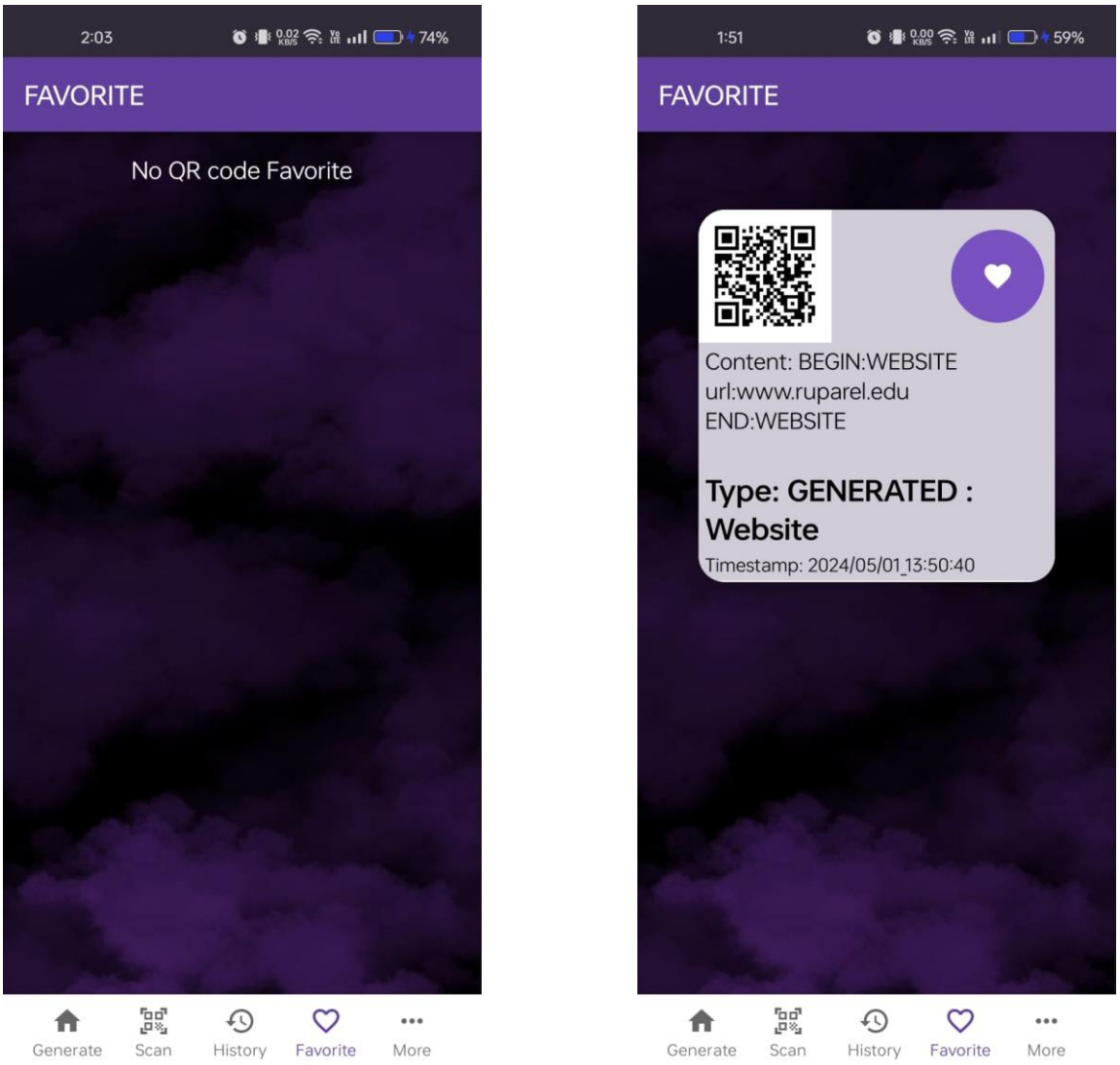


Figure 16:Favorite Page

Users can mark specific QR codes as favorites by clicking on corresponding Favourite button after generating or scanning the QR Code. Favorite QR codes are easily accessible for future reference. User can also remove the Favourite QR Code by clicking on remove Favorite button present in Favorite fragment.

- **Successful Logout**

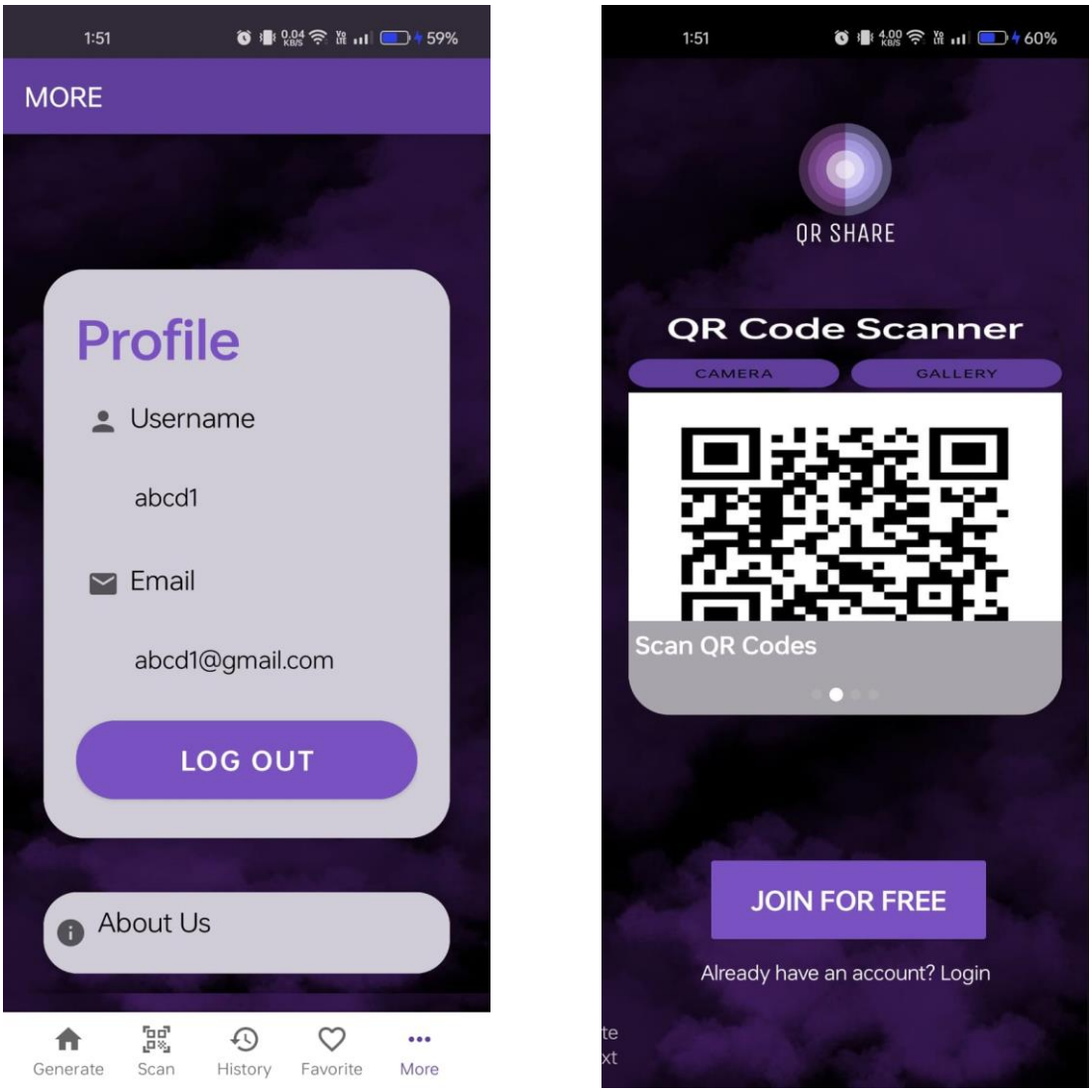


Figure 17: Logout Page

Upon Logout button click, the user is directed to the main screen where they can either Sign up or Login to access the application.

Chapter 6

Conclusion and Future Work

The development and implementation of QRShare: QR Code Generator and Scanner application have culminated in a versatile tool for efficient data transfer and management. Through the integration of QR code generation and scanning capabilities, users can easily encode and decode various types of information, ranging from text to URLs, contact details, and more. The application's intuitive interface and robust functionality offer users a seamless experience for creating, sharing, and storing QR codes.

Despite its strengths, QRShare has certain limitations that warrant consideration. Firstly, while the application provides a convenient solution for individual users, its scalability to enterprise-level usage may require further optimization and customization. Additionally, the current version of QRShare primarily focuses on basic QR code functionalities, and there is room for expansion into more advanced features such as custom logo embedding, batch processing, and enhanced data encryption. Moreover, compatibility across different devices and operating systems remains a challenge that requires ongoing refinement and testing.

Looking ahead, there are several avenues for enhancing QRShare and extending its utility. One promising direction involves incorporating advanced customization options, such as the ability to add logos or branding elements to generated QR codes. This feature would cater to businesses and organizations seeking to personalize their QR code campaigns for marketing or branding purposes. Furthermore, expanding the application's capabilities to support bulk QR code generation and scanning would streamline workflow processes for users dealing with large datasets or frequent code interactions.

In conclusion, QRShare represents a solid foundation for facilitating QR code-based data transfer and management. By addressing its current limitations and embracing future opportunities for expansion and improvement, QRShare has the potential to evolve into a comprehensive solution for individuals and businesses seeking efficient and secure QR code functionality in their daily operations.

Chapter 7

Reference

[1] ZXing GitHub Repository. Retrieved from https://github.com/zxing/zxing Katalon Resources Center Blog. Retrieved from https://katalon.com/resources-center/blog/

[3] QR Code. Retrieved from https://www.qrcode.com/en/about/

[4] Tutorialspoint. Retrieved from https://www.tutorialspoint.com/

[5] ResearchGate. Retrieved from www.researchgate.net

[6] Saturn Cloud [2] Blog. Retrieved from https://saturncloud.io/blog/

[7] International Journal of Advanced Research in Science, Communication, and Technology. Retrieved from https://ijarsct.co.in/

[8] Atlan. Retrieved from https://atlan.com/

[9] IBM Developer. Retrieved from https://developer.ibm.com/