QR Code Generator

A Project Report

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

BY

Atmaram Kishor Pawar

SEAT NO: 4021733

Under the esteemed guidance of

Mrs. Calvina Suhas Maharao

DEPARTMENT OF INFORMATION TECHNOLOGY

ST. GONSALO GARCIA COLLEGE

VASAI, 401201

MAHARASHTRA

2022-23

ST. GONSALO GARCIA COLLEGE OF ARTS AND COMMERCE,VASAI

(Affiliated to University of Mumbai)

VASAI-MAHARASHTRA-401201

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled "**QR Code Generator** ", is bonafied work of **Mr. Atmaram Kishor Pawar** bearing Seat No. **4021733** Submitted in partial fulfilment of the Requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Project Guide **Coordinator**

Internal Examiner **External Examiner**

Date: **College Seal**

ABSTRACT

Quick Response (QR) codes seem to appear everywhere these days. We can see them on posters, magazine ads, websites, product packaging and so on. Biggest reason for selecting this project topic is the popularity of QR codes is growing rapidly all around the world. QR code has more data storage as compared to the Barcode. QR code has high data storage capacity. A single QR Code symbol can contain up to 7,089 numerals (200 times the amount of data storage capacity of the traditional 1-D barcode). QR code takes less space to store large data and information. Anyone can make their own QR code according to their need, but users required a QR Generator software or app to make their own QR code.

There are many QR code generating software and app available, but many QR Code generator services allow users to create limited number of QR code for free then they asking for a subscription fee for generating QR code. Online free QR code generators are not safe as you think and not so fast comparatively paid QR code generators. So from this project I provide a safe and fastQR code generating services to users to make their own QR code according to their need. This QR code generator is read much faster and scanned from any angle within 360 degrees i.e. no need to align the scanner with the code symbol.

This is the main reason for selecting QR code generator app as a project topic to provide a safe and fast QR generation.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciate to our Principal **Dr. SOMNATH VIBHUTE,** Head of Information and Technology Department **Dr. SANGITA DUBEY** and special thanks to our Project guide **Prof. CALVINA SUHAS MAHARAO** who gave the golden opportunity to this wonderful project on the topic '**QR Code Generator** '.

I am extremely grateful for their kind support and guidance and for their valuable suggestion which made me complete the project. I would also like to express my sincere thanks to all those who have encouraged me and willingly helped me in completion of my project.

DECLARATION

I hereby declare that the project entitled, "**QR Code Generator"** done at place **St. Gonsalo Garcia College**, 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 **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENTS

List of Tables

List of Figures

SYNOPSIS

Introduction :

Quick Response (QR) codes seem to appear everywhere these days. We can see them on posters, magazine ads, websites, product packaging and so on. A QR code is a type of matrix bar code or two-dimensional code that can store data information and designed to be read by smartphones.

QR stands for “Quick Response” indicating that the code contents should be decoded very quickly at high speed. The information encoded may be text, a URL or other data. The popularity of QR codes is growing rapidly all around the world. Nowadays, mobile phones with built-in camera are widely used to recognize the QR Codes. QR code are able to store much more data, they can include over 3000 characters on a very small space.

The QR code system was invented in 1994 by Masahiro Hara from the Japanese company Denso Wave, was initially used for tracking inventory in vehicle parts manufacturing. The idea behind the development of the QR code is the limitation of the barcode information capacity (can only hold 20 alphanumeric characters).

Now QR codes are used in many other fields, from commercial tracking to entertainment, in-store product labeling, and in those applications that are aimed at smartphone users. By using QR code generator app, users can generate and print their own QR codes for others to scan and use. QR code also useful in some novel application fields including mobile marketing, online advertising, electronic ticket, electronic coupon, electronic payment, identification, academics, information security, etc. Depending on users purpose, we can use our QR generator to create QR Codes to open a website, listen to music, watch YouTube videos, store image files, connect to a WiFi network, and much more.

Frontend :

- Python : Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Kivy : Kivy is a graphical user interface opensource Python library that allows you to develop multi-platform applications on Windows, macOS, Android, iOS, Linux, and Raspberry-Pi.

Backend :

- Google Colaboratory : Colab is basically a free Jupyter notebook environment running wholly in the cloud.
- Buildozer : Buildozer is a tool that aim to package mobiles application easily.
- SQLiteStudio : SQLiteStudio is desktop application for browsing and editing SQLite database files. It is aimed for people, who know what SQLite is, or what relational databases are in general.

Software requirements :

- PyCharm : PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers.
- SQLiteStudio : SQLiteStudio is desktop application for browsing and editing SQLite database files. It is aimed for people, who know what SQLite is, or what relational databases are in general.

Hardware requirements :

- Desktop or Laptop

Memory requirements :

- Operating System : Windows 8 or higher, macOS, Linux
- Processor : Intel i7-8550U
- Memory : 2GB minimum
- Screen resolution :1280x1024 or higher
- Internet connection : Required

Reason for selecting this project topic :

Biggest reason for selecting this project topic is the popularity of QR codes is growing rapidly all around the world. QR code has more data storage as compared to the Barcode.QR code has high data storage capacity. A single QR Code symbol can contain up to 7,089 numerals (200 times the amount of data storage capacity of the traditional 1-D barcode). QR code takes less space to store large data and information. Anyone can make their own QR code according to their need, but users required a QR Generator software or app to make their own QR code.

There are many QR code generating software and app available, but many QR Code generator services allow users to create limited number of QR code for free then they asking for a subscription fee for generating QR code. Online free QR code generators are not safe as you think and not so fast comparatively paid QR code generators. So from this project I provide a safe and fast QR code generating services to users to make their own QR code according to their need. This QR code generator is read much faster and scanned from any angle within 360 degrees i.e. no need to align the scanner with the code symbol.

This is the main reason for selecting QR code generator app as a project topic to provide a safe and fast QR generation.

Scope :

QR Codes are everywhere. You might have seen them on the product packaging, billboards, business cards, museums, and restaurants. They are a form of contactless technology that bridges the gap between physical and digital experiences. Hence, many businesses and city admins are actively leveraging QR Codes for promotional and operational use cases. Today many businesses are looking forward to implementing advanced technologies to engage their audiences. The QR Codes can be used in every industry for different use cases, from sharing website links and text to multimedia, they can do it all.

In this QR code generator app I will add some features like customizing QR code in different colors and different shapes, to look QR code more attractive. Also add some security password to each QR code so if user generates a QR code for a particular person or a group of persons, to read or see information or data they have to enter the security password.

Instead of make a universal QR code generator app, I will make a QR code generator app for particular industries like smart menu card for restaurant industry, smart brochure for fashion ,advertising and marketing, smart QR code ID for companies, schools, colleges, also make a QR shopping app for shopping grocery, vegetables, fruits and so on.

Chapter 1

Introduction

1.1 Background

Quick Response(QR) Code is used in all over the world, it looks like a small box which includes a random series of black and white pixels. Even though QR Code is a tiny symbol, a website address, specifications of particular products or personal information can be included in this symbol.

A Quick Response (QR) Code is a two dimensional bar code consisting of grid of tiny black squares on a white background. Due to its ability to store more information and fast readability, it gained popularity over the traditional bar codes composed of black bars and white spaces. Quick Response (QR) codes seem to appear everywhere these days. We can see them on posters, magazine ads, websites, product packaging and so on. A QR code is a type of matrix bar code or two-dimensional code that can store data information and designed to be read by smartphones.

QR stands for “Quick Response” indicating that the code contents should be decoded very quickly at high speed. The information encoded may be text, a URL or other data. The popularity of QR codes is growing rapidly all around the world. Nowadays, mobile phones with built-in camera are widely used to recognize the QR Codes. QR code are able to store much more data, they can include over 3000 characters on a very small space.

The QR code system was invented in 1994 by Masahiro Hara from the Japanese company Denso Wave, was initially used for tracking inventory in vehicle parts manufacturing. The idea behind the development of the QR code is the limitation of the barcode information capacity (can only hold 20 alphanumeric characters).

The main difference between barcodes and QR codes is one of physical dimensions. Barcodes can be scanned in a line. This means that data is limited to what can be placed in that one stretch of stripes. QR codes, on the other hand, add another dimension from which information can be written and scanned. Instead of a single line, these labels can be read both vertically and horizontally. The development team behind the QR code wanted to make the code easy to scan so that operatives did not waste time getting it at the right angle. They also

wanted it to have a distinctive design to make it easy to identify. This led them to choose the iconic square shape that is still used today.

Now QR codes are used in many other fields, from commercial tracking to entertainment, in-store product labeling, and in those applications that are aimed at smartphone users. By using QR code generator app, users can generate and print their own QR codes for others to scan and use. QR code also useful in some novel application fields including mobile marketing, online advertising, electronic ticket, electronic coupon, electronic payment, identification, academics, information security, etc. Depending on users purpose, we can use our QR generator to create QR Codes to open a website, listen to music, watch YouTube videos, store image files, connect to a Wi-Fi network, and much more.

QR codes are a form of contactless technology that bridges the gap between physical and digital experiences. Hence, many businesses and city admins are actively leveraging QR Codes for promotional and operational use cases. Today many businesses are looking forward to implementing advanced technologies to engage their audiences. The QR Codes can be used in every industry for different use cases, from sharing website links and text to multimedia, they can do it all.

1.2 Objective

The popularity of QR codes is growing rapidly all around the world. QR Code works just like a barcode but, is much faster as it processes the information within few seconds. QR Codes do not require specials devices and can be scanned by a smartphone camera but barcode require barcode reader device to scan barcode effectively. QR Code can be used in many ways like to show product details, track product delivery, to show menu to customers in restaurants, for accessing social media platforms, to get access to any webinar or any other site or link, and to transfer money .

A barcode is one-dimensional, which means that scanners use only horizontal direction to scan the barcode. A barcode is one-dimensional, the storage of information capacity is limited and the barcode can store less than 20 characters. A QR Code enables to store several of information that is hundred times than the capacity of traditional barcode stores information.

QR Code is capable of storing various types of data, e.g. numeric and alphabetic characters, kanji, kana, hiragana, symbols, binary, and control codes. QR Code can store maximum 7,089 characters in one symbol. QR Code can immediately connect people to virtual environment of information and entertainment. In addition, convenient and fast features of QR Code also attract people to use it.

QR Code has a characteristic that it can be read in 360 degree direction. The traditional one dimensional barcode recognizes the information only plus-minus ten degrees which is relatively smaller than QR Code. QR Code not only can be read in 360 degree direction, but QR Code also can be read in high speed. The secret of QR Code reading direction in 360 degrees is that position the detection patterns located at the three corners of the symbol can locate the QR Code. QR Codes are usually very small in size. It is possible that if we print a QR Code on paper or any wrapper, some portion might get damaged a bit. However, QR Code will work even if only 30% of the code is unreadable.

The continued growth of mobile usage means more opportunities to use a QR code. A Smart Insights report states that of all world-wide internet users, 80% will access it from a smartphone. According to Print robot, there was an 11% increase in the number of QR codes generated from 2015 to 2016. And a 2015 Scanbury report showed 4.3 scans per person, up sssfrom 4.0 scans per person in 2014 or 7.5% growth.

The rise of popularity of video as a content marketing format means people want more ways to engage with video. According to Brafton, in 2017 online video will account for 74% of all web traffic, and 28% of smartphone users watch a video on their devices at least once a day. QR codes provide a great way to connect to a video from a non-digital medium, like displaying movie previews from a poster, food prep steps from a recipe, or a video introduction from a resume.

Marketers use branded QR Codes for their campaigns. This helps them build brand recognition while grabbing audience attention to get more scans. QR codes on direct mail pieces can also help marketers gain useful information from their campaigns. Many businesses and city admins are actively leveraging QR Codes for promotional and operational use cases. As per the data by blue bite, QR Code interaction saw a surge of 94% from 2018 to 2020. QR Code usage has seen a further rise from 2020 to 2021. In fact, reports suggest that one billion smartphones will have access to QR Codes by the end of 2022.

1.3 Purpose and Scope

1.3.1 Purpose

The application of QR Codes is diversifying with their growing application in numerous industries. And here are the two most soaring opportunities for QR Codes ahead: In the era of social distancing, contactless operations have become the norm. From contactless payments to promotions, QR Coded will be seen everywhere and if you're planning on running a campaign, you must ponder overriding the wave of contactless technology with QR Codes too. Today, nothing engages users better than an interactive experience. And with the QR Codes' ability to connect the offline world to online content, you can create interactive experiences for your audience. This could be to hand out coupons, get them to visit your website, or even show them a movie trailer.

Considering the benefits of QR Codes, their applications are going to grow manifold in the upcoming years. Businesses will be seen incorporating this technology to ensure smart operations. Many industries are using QR Codes already. These include textiles, real estate, manufacturing, IT, etc. And many of them are looking forward to leveraging the QR Code technology to level up the scale of their operations and promotions.

Many businesses in the food and beverage industry are using QR Codes for different use cases. These include marketing, inventory management, and even anti-counterfeiting measures. Many manufacturers add QR Codes to their product packaging. When scanned, this QR Code takes the consumers to a landing page where they can verify the product's authenticity. In fact, relaying product information is another soaring use case of QR Codes in this industry. Manufacturers often are restricted from adding complete product information to the packaging due to limited space. But with a QR Code, they can share text, video, images, and even web links with the customers. Today customers are highly conscious about what they're buying, this helps build trust among them.

The automotive sector has always been the target of counterfeiters. To overcome the issue, many brands are looking forward to using QR Code labels. This would help customers differentiate genuine products from the pool of fake ones.

The healthcare industry is expanding the utilization of QR Codes for many use cases such as: Preventing drug counterfeiting ,Verifying the authenticity of vaccine certificates, Getting easy access to a patient's medical history

1.3.2 Scope

QR Codes are everywhere. You might have seen them on the product packaging, billboards, business cards, museums, and restaurants. They are a form of contactless technology that bridges the gap between physical and digital experiences. Hence, many businesses and city admins are actively leveraging QR Codes for promotional and operational use cases. Today many businesses are looking forward to implementing advanced technologies to engage their audiences. The QR Codes can be used in every industry for different use cases, from sharing website links and text to multimedia, they can do it all.

In this QR code generator app I will add some features like customizing QR code in different colours and different shapes, to look QR code more attractive. Also add some security password to each QR code so if user generates a QR code for a particular person or a group of persons, to read or see information or data they have to enter the security password.

Instead of make a universal QR code generator app, I will make a QR code generator app for particular industries like smart menu card for restaurant industry, smart brochure for fashion ,advertising and marketing, smart QR code ID for companies, schools, colleges, also make a QR shopping app for shopping grocery, vegetables, fruits and so on.

Chapter 2

System Analysis

2.1 Existing System

The QR code system was invented in 1994 by Masahiro Hara from the Japanese company Denso Wave, was initially used for tracking inventory in vehicle parts manufacturing. A barcode is one-dimensional, which means that scanners use only horizontal direction to scan the barcode. A barcode is one-dimensional so the storage of information capacity is limited and the barcode can store less than 20 characters. A QR Code enables to store several of information that is hundred times than the capacity of traditional barcode stores information.

QR Code works just like a barcode but, is much faster as it processes the information within few seconds. QR Codes do not require specials devices and can be scanned by a smartphone camera but barcode require barcode reader device to scan barcode effectively. QR Code can be used in many ways like to show product details, track product delivery, to show menu to customers in restaurants, for accessing social media platforms, to get access to any webinar or any other site or link, and to transfer money .

The popularity of QR codes is growing rapidly all around the world. QR code has more data storage as compared to the Barcode. QR code has high data storage capacity. QR code takes less space to store large data and information. Anyone can make their own QR code according to their need, but users required a QR Generator software or app to make their own QR code.

There are many QR code generating software and app available, but many QR Code generator services allow users to create limited number of QR code for free then they asking for a subscription fee for generating QR code. Online free QR code generators are not safe as you think and not so fast comparatively paid QR code generators. So from this project I provide a safe and fast QR code generating services to users to make their own QR code according to their need. This QR code generator is read much faster and scanned from any angle within 360 degrees i.e. no need to align the scanner with the code symbol . This is the main reason for selecting QR code generator app as a project topic to provide a safe and fast QR generation.

2.2 Propose System

A QR code (quick response code) is a type of two dimensional (2D) bar code that is used to provide easy access to online information through the digital camera on a smartphone or tablet. The information encoded may be text, a URL or other data. The popularity of QR codes is growing rapidly all around the world. Nowadays, mobile phones with built-in camera are widely used to recognize the QR Codes. QR code are able to store much more data, they can include over 3000 characters on a very small space.

QR Code works just like a barcode but, is much faster as it processes the information within few seconds. QR Codes do not require specials devices and can be scanned by a smartphone camera but barcode require barcode reader device to scan barcode effectively. QR Code can be used in many ways like to show product details, track product delivery, to show menu to customers in restaurants, for accessing social media platforms, to get access to any webinar or any other site or link, and to transfer money .

QR Code is capable of storing various types of data, e.g. numeric and alphabetic characters, kanji, kana, hiragana, symbols, binary, and control codes. QR Code can store maximum 7,089 characters in one symbol. QR Code can immediately connect people to virtual environment of information and entertainment. In addition, convenient and fast features of QR Code also attract people to use it.

QR Code has a characteristic that it can be read in 360 degree direction. The traditional one dimensional barcode recognizes the information only plus-minus ten degrees which is relatively smaller than QR Code. QR Code not only can be read in 360 degree direction, but QR Code also can be read in high speed. The secret of QR Code reading direction in 360 degrees is that position the detection patterns located at the three corners of the symbol can locate the QR Code. QR Codes are usually very small in size. It is possible that if we print a QR Code on paper or any wrapper, some portion might get damaged a bit. However, QR Code will work even if only 30% of the code is unreadable.

QR codes have become common in consumer advertising. Typically, a smartphone is used as a QR code scanner, displaying the code and converting it to some useful form such as a standard URL for a website, thereby obviating the need for a user to type it into a web browser. QR code has become a focus of advertising strategy, since it provides a way to access a brand's website more quickly than by manually entering a URL.

Although initially QR codes are used to track parts in vehicle manufacturing, QR codes are used over a much wider range of applications. These include commercial tracking, entertainment and transport ticketing, product and loyalty marketing and in-store product labelling .Many of these applications target mobile-phone users (via mobile tagging). Users may receive text, add a vCard contact to their device, open a URL, or compose an e-mail or text message after scanning QR codes. They can generate and print their own QR codes for others to scan and use by visiting one of several pay or free QR code-generating sites or apps. QR codes storing addresses and URLs may appear in magazines, on signs, on buses, on business cards, or on almost any object about which users might want information. Users with a camera phone equipped with the correct reader application can scan the image of the QR code to display text, contact information, connect to a wireless network, or open a web page in the phone's browser.

QR codes have been used to establish "virtual stores", where a gallery of product information and QR codes is presented to the customer, e.g. on a train station wall. The customers scan the QR codes, and the products are delivered to their homes. This use started in South Korea, and Argentina, but is currently expanding globally. Walmart, Procter & Gamble and Woolworths have already adopted the Virtual Store concept.

QR codes can be used to store bank account information or credit card information, or they can be specifically designed to work with particular payment provider applications. There are several trial applications of QR code payments across the world. In developing countries like China, India and Bangladesh QR code payment is a very popular and convenient method of making payments.

Restaurants can present a QR code near the front door or at the table allowing guests to view an online menu, or even redirect them to an online ordering website or app, allowing them to order and possibly pay for their meal without having to use a cashier or waiter. At table serve restaurants, QR codes enable guests to order and pay for their meals without a waiter involved. The QR code contains the table number so servers know where to bring the food. This application has grown especially since the need for social distancing during the 2020 COVID- 19 pandemic prompted reduced contact between service staff and customers.

Serialised QR codes have been used by brands and governments to let consumers, retailers and distributors verify the authenticity of the products and help with detecting counterfeit products, as part of a brand protection program. However, the security level of a

regular QR Code is limited since QR Codes printed on original products are easily reproduced on fake products, even though the analysis of data generated as a result of QR Code scanning can be used to detect counterfeiting and illicit activity.

The QR code used in food traceability system, as they provide detailed information about food, as well as information that helps them in their purchasing decision. If QR Codes are serialised, consumers can access a web page showing the supply chain for each ingredient, as well as information specific to each related batch, including meat processors and manufacturers, which helps address the concerns they have about the origin of their food.

2.3 Requirement Analysis

In this project, we will build a QR Code generator using Python .To build the QR code generator project we need to follow the below steps :

- Importing the modules.
- Creating the main window.
- Taking the input of the text/URL, location to store the QR code, name of the QR code and the size of the QR code.
- Writing the function to generate and save the QR Code.

Also use kivy, Google Colaboratory and Buildozer to create App.

- Kivy is a free and open-source Python library used for developing mobile applications and other multitouch application software with a Natural User Interface
- Colab is basically a free Jupyter notebook environment running wholly in the cloud
- Buildozer is a tool that aim to package mobiles application easily.
- Buildozer is a tool that aim to package mobiles application easily.

2.4 Hardware Requirements

Here is the list of tools and equipment used to develop the system :

- Operating System : Windows 8 or higher, macOS, Linux
- Processor : Intel i7-8550U
- Memory : 2GB minimum
- Screen resolution :1280x1024 or higher
- Keyboard : Standard windows Keyboard
- Internet connection : Required

2.5 Software Requirements

- PyCharm :

  PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers.

- Python :

  Python is a high-level, interpreted, interactive and object-oriented scripting language.

- Kivy :

  Kivy is a graphical user interface opensource Python library that allows you to develop multi-platform applications on Windows, macOS, Android, iOS, Linux.

- Google Colaboratory :

  Colab is basically a free Jupyter notebook environment running wholly in the cloud.

- Buildozer :

  Buildozer is a tool that aim to package mobiles application easily.

- SQLiteStudio : SQLiteStudio is desktop application for browsing and editing SQLite database files. It is aimed for people, who know what SQLite is, or what relational databases are in general.

2.6 Justification of platform

- **Python**

The main reason for selecting python as a project language because Python is efficient and reliable, allowing developers to create powerful applications with a minimum of effort. Python is a high-level, open source programming language initially released in 1991. The language was designed to improve upon Java and C++, both of which had significant problems with code readability.

Python is a dynamic programming language that enables developers to complete code in fewer steps than its predecessors. The language is frequently used to create complex games, web applications, and desktop software. Python's extensive support libraries prescript many programming jobs and help reduce the length of the code.

Although Python is an effective choice for many kinds of development projects, its usefulness in web development is worth specific recognition. Using available open-source libraries, Python developers can get their web applications up and running quickly and easily And while other languages, such as Java or .NET, might offer increased performance, the speed and developer experience provided by Python makes it an obvious choice for those who need a quick solution that they can depend on. At the same time, Python's variety of available resources offers a unique opportunity to integrate other application types into websites .

Dropbox is one of the most popular web and desktop applications built with Python. Developers used the programming language on a large percentage of their server-side code to allow users to seamlessly share files over the cloud. In fact, the language is so versatile that tech companies like Spotify and Reddit used the language to build their desktop and web applications.

- **PyCharm**

PyCharm is one of the most popular Python IDEs .The main reason PyCharm for the creation of this IDE was for Python programming, and to operate across multiple platforms like Windows, Linux, and macOS. The IDE comprises code analysis tools, debugger, testing tools, and also version control options. It also assists developers in building Python plugins with the help of various APIs available. The IDE allows us to work with several databases

directly without getting it integrated with other tools. Although it is specially designed for Python, HTML, CSS, and Java script files can also be created with this IDE. It also comes with a beautiful user interface that can be customized according to the needs using plugins.

PyCharm comes with a smart code editor that facilitates writing high-quality Python code. It offers an enhanced level of code comprehension and readability by means of distinct colour schemes for keywords, classes, and functions, i.e., syntax and error highlighting. In addition to offering the smart code completion feature, the code editor generates instructions for completing the current code. Identifying errors and issues is much more comfortable, along with linter integration and quick fixes.

PyCharm provides support for integrating a range of tools. These tools vary from helping in enhancing the code productivity to facilitate dealing with data science projects. An IDE comes with support for debugging and testing programs. To accomplish the same, PyCharm features an integrated Python debugger and integrated unit testing with line-by-line code coverage.

Python developers can also use PyCharm for creating web applications. As such, the Python IDE provides support for popular web technologies, including CSS, HTML, JavaScript, TypeScript. Additionally, it also includes support for template languages, and SQL. Live editing is also available in PyCharm, i.e., developers can create/modify a web page while pushing it live simultaneously. Hence, changes can be followed directly on a web browser. Building web applications using AngularJS or NodeJS is also available .Everything you do in PyCharm, you do within the context of a project. It serves as a basis for coding assistance, bulk refactoring, coding style consistency, and so on.

- **Kivy**

Kivy is a great tool for developing Android Apps. This cross-platform Python framework can be deployed to Windows, Mac, Linux, and Raspberry Pi. It supports multitouch events in addition to regular keyboard and mouse inputs. Kivy even supports GPU acceleration of its graphics, since they’re built using OpenGL ES2. The project uses the MIT license, so you can use this library for free and commercial software.

When we create an application with Kivy, you're creating a Natural User Interface or NUI. The idea behind a Natural User Interface is that the user can easily learn how to use your software with little to no instruction .Kivy does not attempt to use native controls or widgets. All of its widgets are custom-drawn. This means that Kivy applications will look the same across all platforms. However, it also means that your app's look and feel will differ from your user's native applications. This could be a benefit or a drawback, depending on your audience.

- **Google Colaboratory**

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members, just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook . we can share your Google Colab notebooks very easily. No need to install any modules to run any code, modules come preinstalled within Google Colab. Google Colab has a great collection of snippets you can just plug in on your code.

- **Buildozer**

Buildozer is a tool that aim to package mobiles application easily. It automates the entire build process, download the prerequisites like python-for-android, Android SDK, NDK, etc. Buildozer manages a file named buildozer.spec in your application directory, describing your application requirements and settings such as title, icon, included modules etc. It will use the specification file to create a package for Android, iOS, and more.

Using Buildozer as the easiest way to make a full APK or AAB. You can also run your Kivy app without a compilation step with the Kivy Launcher app. Kivy applications can be released on an Android market such as the Play store, with a few extra steps to create a fully signed AAB (Android App Bundle).

- SQLiteStudio :

SQLiteStudio is desktop application for browsing and editing SQLite database files. It is aimed for people, who know what SQLite is, or what relational databases are in general. One of SQLite's greatest advantages is that it can run nearly anywhere. SQLite has been ported to a wide variety of platforms: Windows, MacOS, Linux, iOS, Android, and more. Windows users in particular can use precompiled binaries for regular Win32, UWP, WinRT, and . Net. SQLite is used to develop embedded software for devices like televisions, cell phones, cameras, etc.

Chapter 3

System Design

3.1 Module Design

3.1.1 Login and Signup

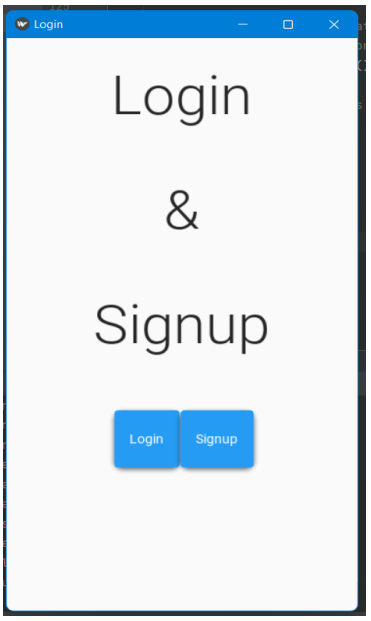

Fig 3.1 Login & Signup

3.1.2 User Login

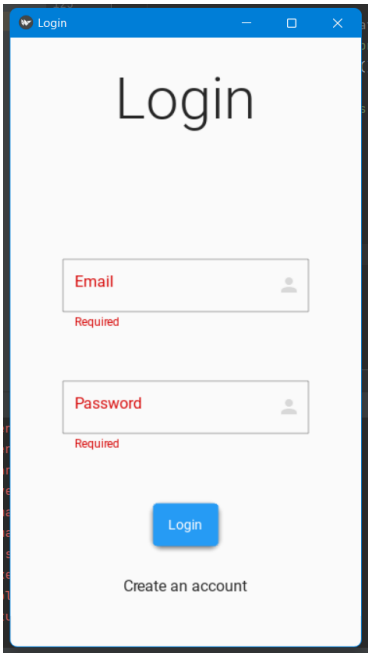

Fig 3.2 User Login

3.1.3 User Signup

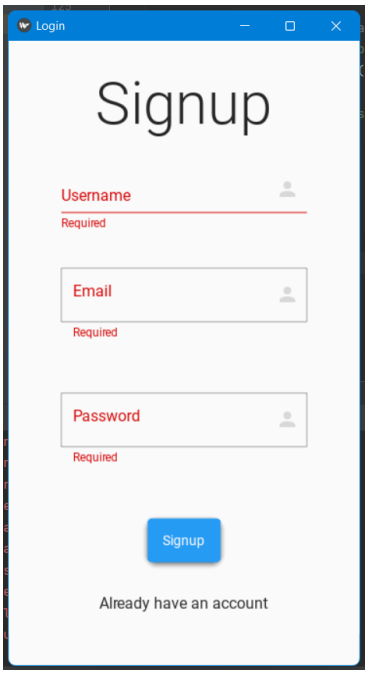

Fig 3.3 User Signup

3.1.4 Home Page

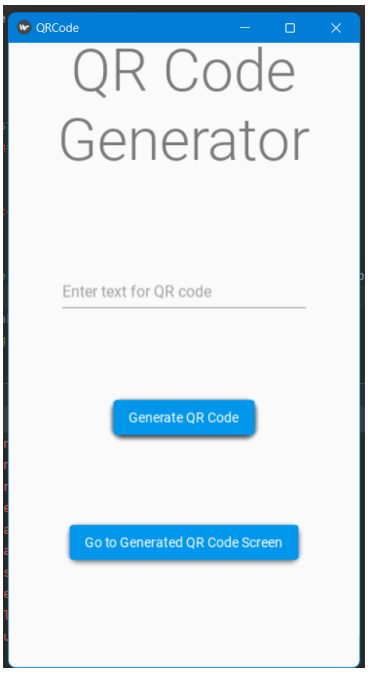

Fig 3.4 Home page

3.1.5 Enter text for QR Code

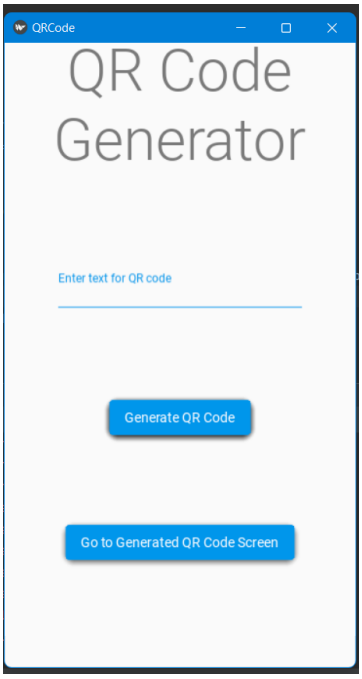

Fig 3.5 Enter text for QR Code

3.1.6 Generate QR Code

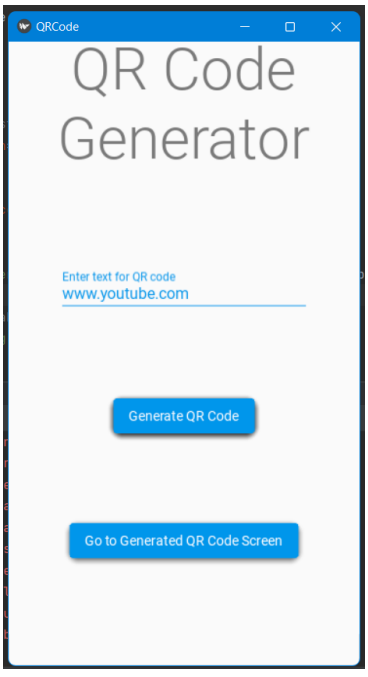

Fig 3.6 Generate QR Code

3.1.7 Go to Generated QR Code Screen

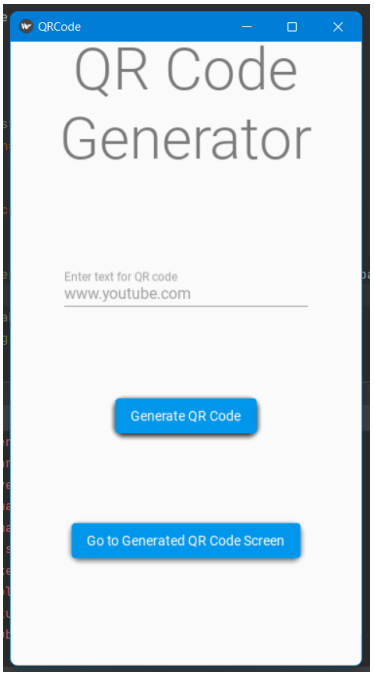

Fig 3.7 Go to Generated QR Code Screen

3.1.8 View QR Image

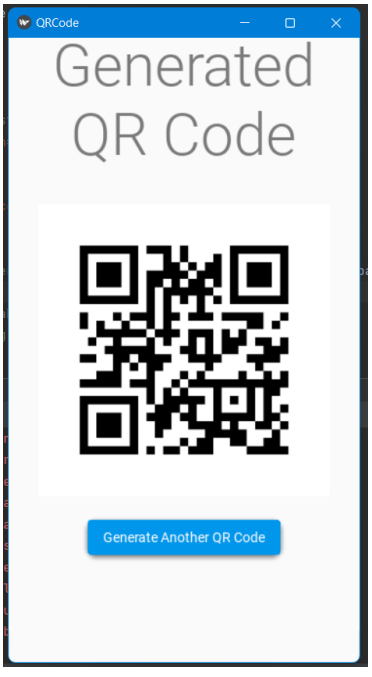

Fig 3.8 View QR Code

3.1.9 Generate another QR Code

Fig 3.9 Generate another QR Code

3.1.10 Similarly create another QR Code

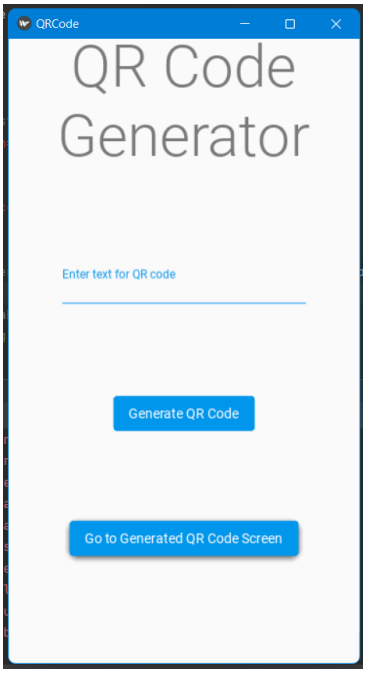

Fig 3.10 Home Page

3.2 Data Dictionary

Sr no.	Notations	Explanation
1.	Login	To authenticate user account
2.	Signup	To register for a new account
3.	User	A person who utilize the services
4.	User name	Username is an identification used by person
5.	User account	An identity created for a person
6.	Email	An email is a unique identifier for an email account.
7.	Password	Used to verify the identity of a user during the authentication process.
8.	URL	Uniform Resource Locator is a unique identifier used to locate a resource on the Internet.
9.	Generate	Designed or produce QR Code
10.	View image	A visual representation of QR image
11.	Display	Capable of representing information visually
12.	Make another	Produce new QR Code

Table 3.1 Data Dictionary

3.3 UML Diagram

3.3.1 Activity Diagram

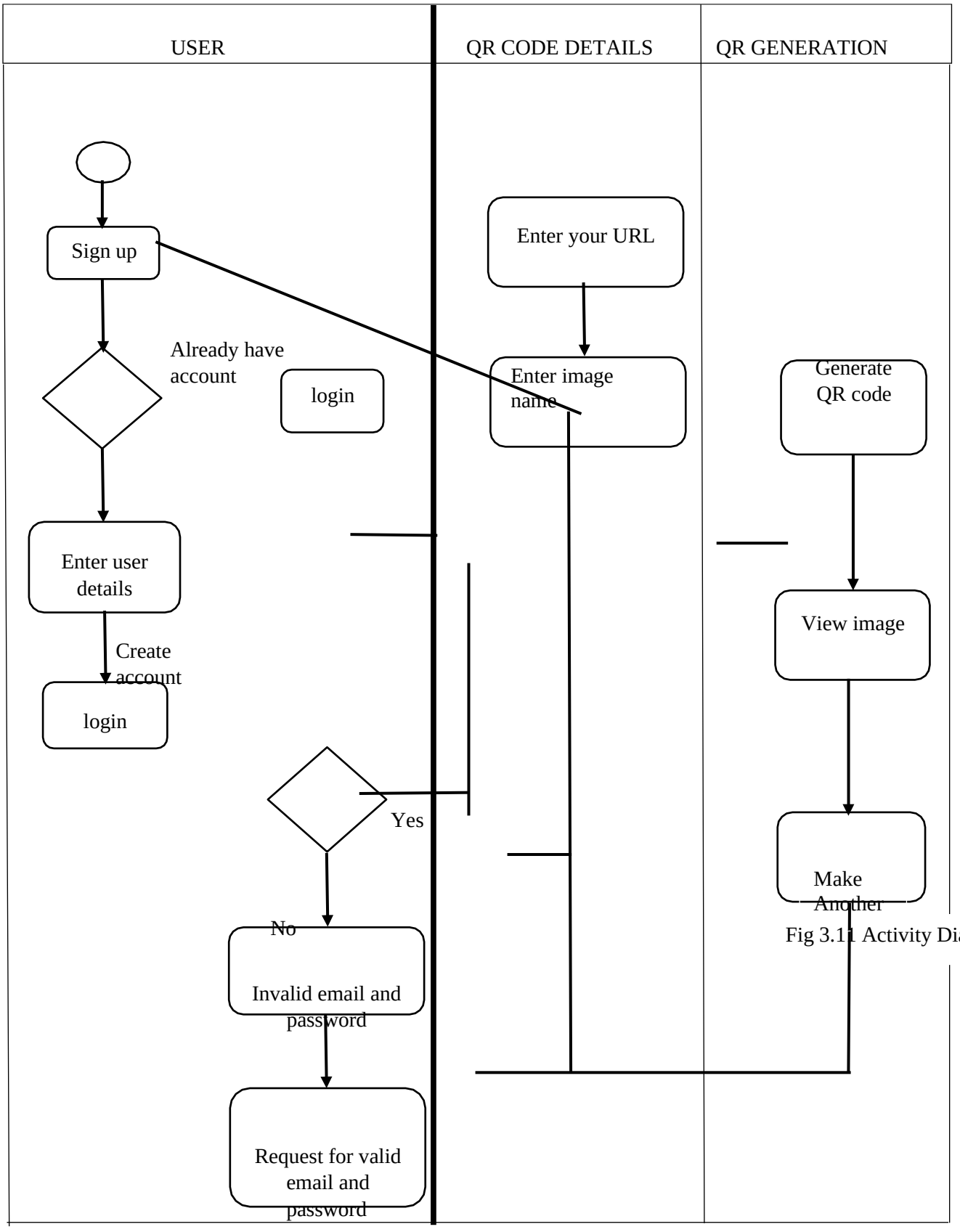

Fig 3.11 Activity Diagram

3.3.2 Class Diagram

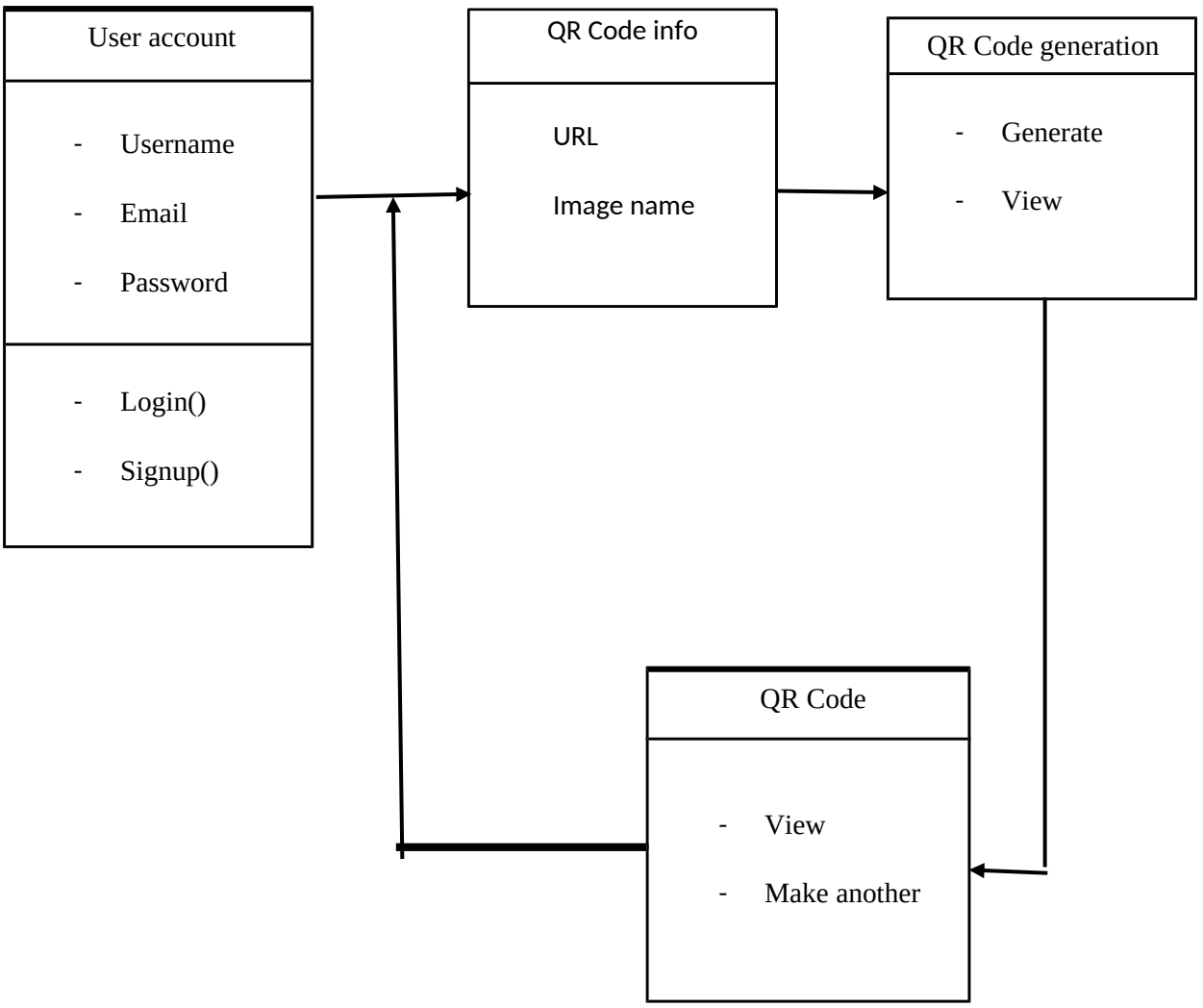

Fig 3.12 Class Diagram

3.3.3 Collaboration Diagram

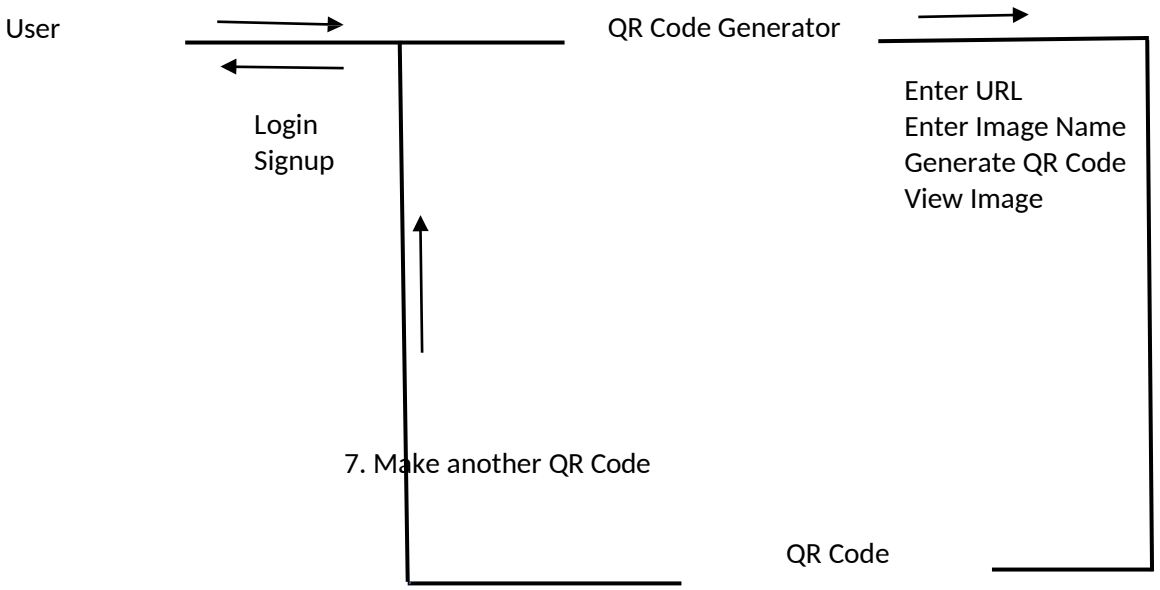

Fig 3.13 Collaboration Diagram

3.3.4 Component Diagram

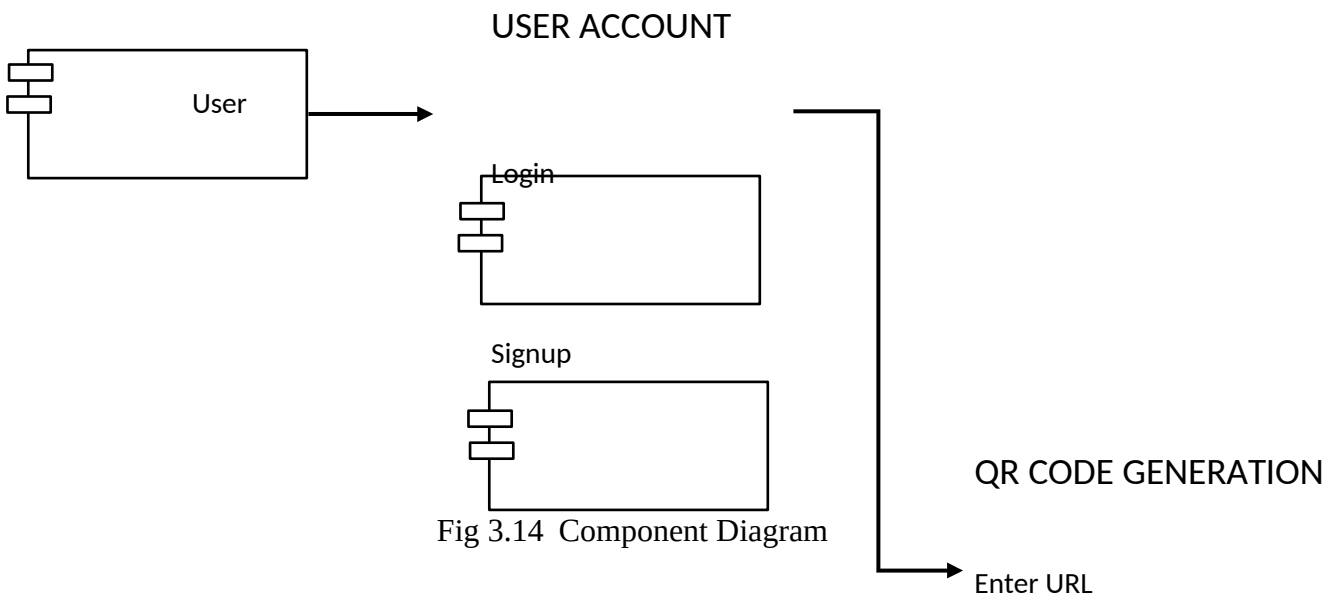

Fig 3.14 Component Diagram

3.3.5 Data Flow Diagram

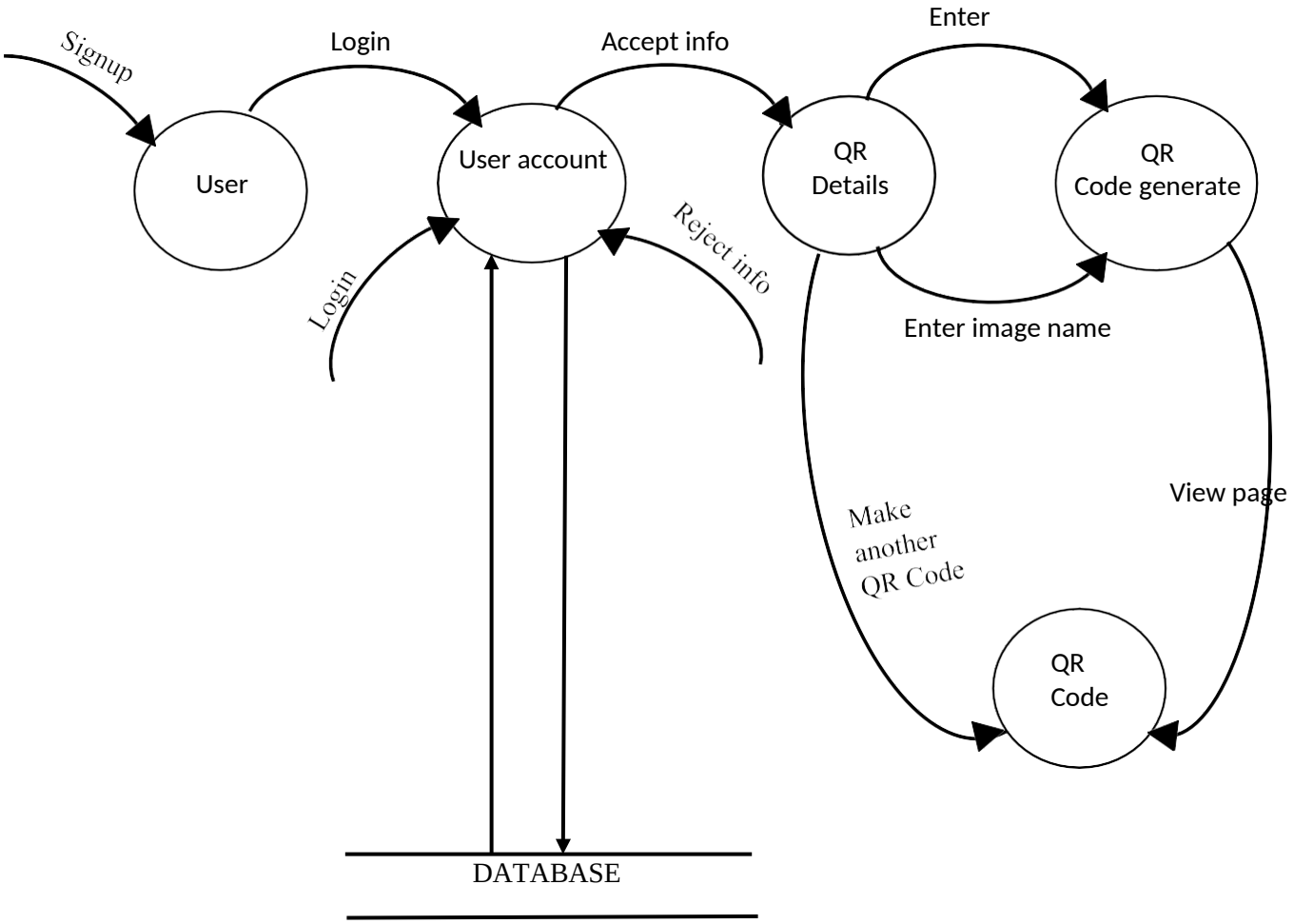

Fig 3.15 Data Flow Diagram

3.3.6 Deployment Diagram

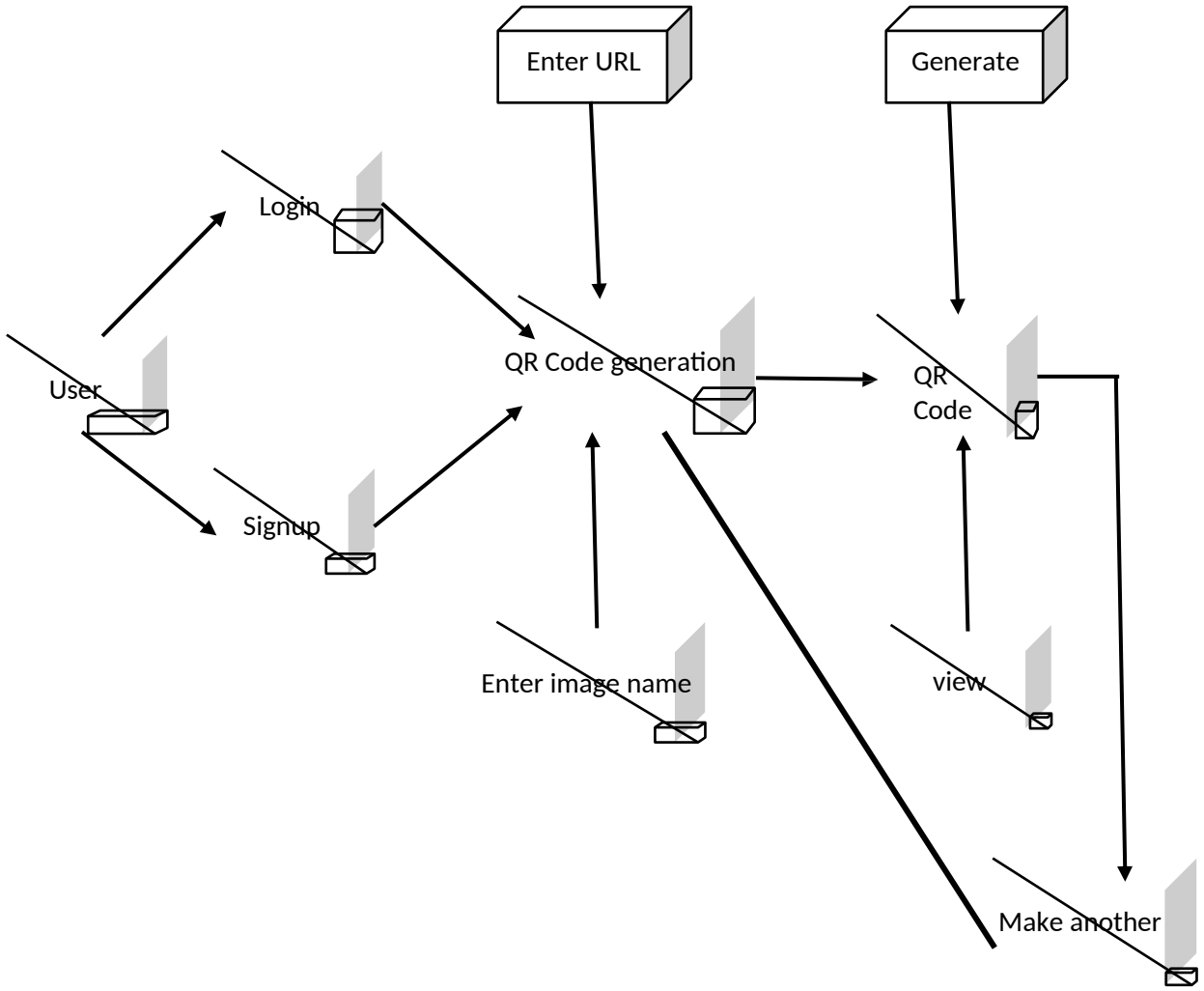

Fig 3.16 Deployment Diagram

3.3.7 ER Diagram

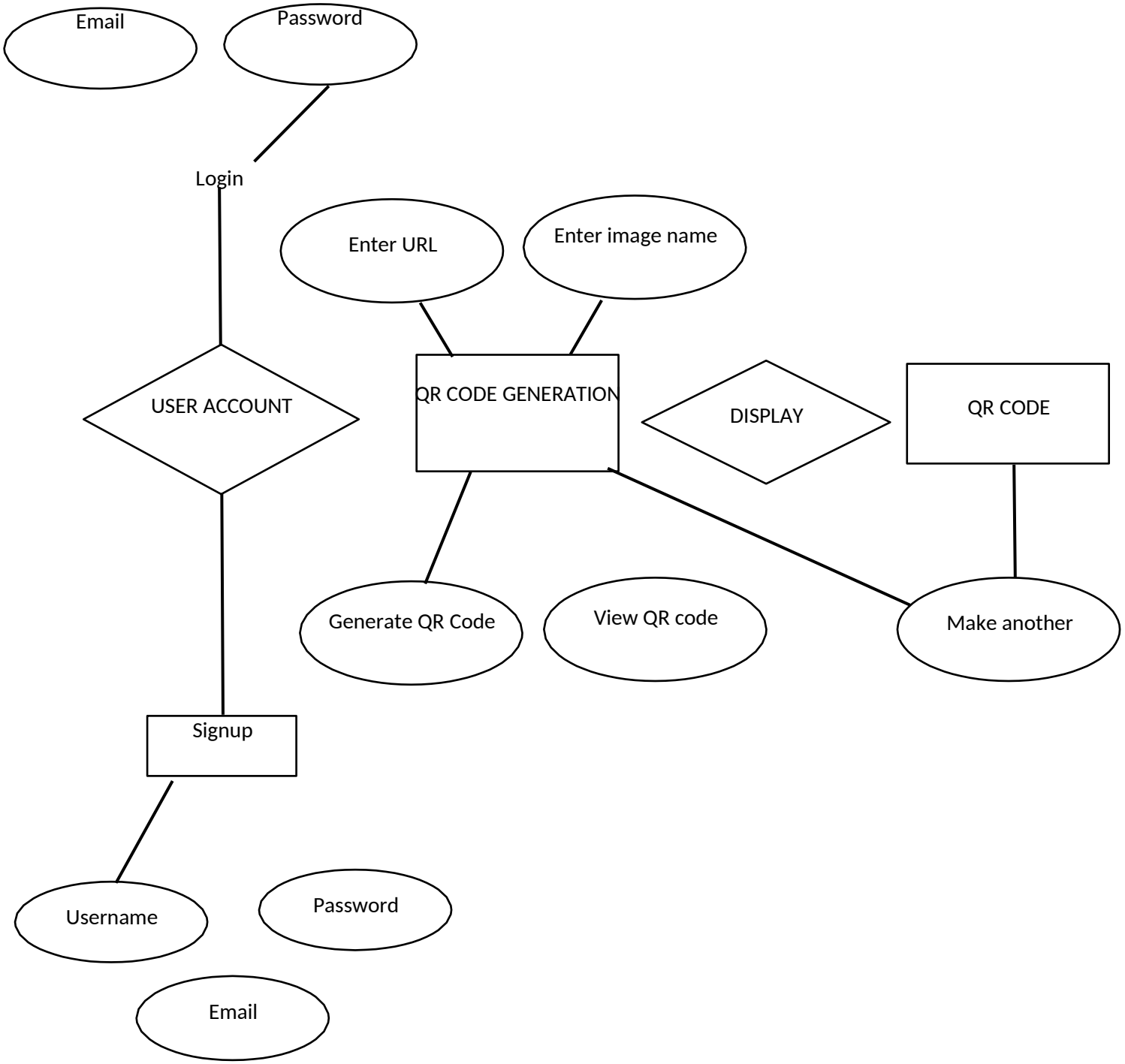

Fig 3.17 ER Diagram

3.3.8 Sequence Diagram

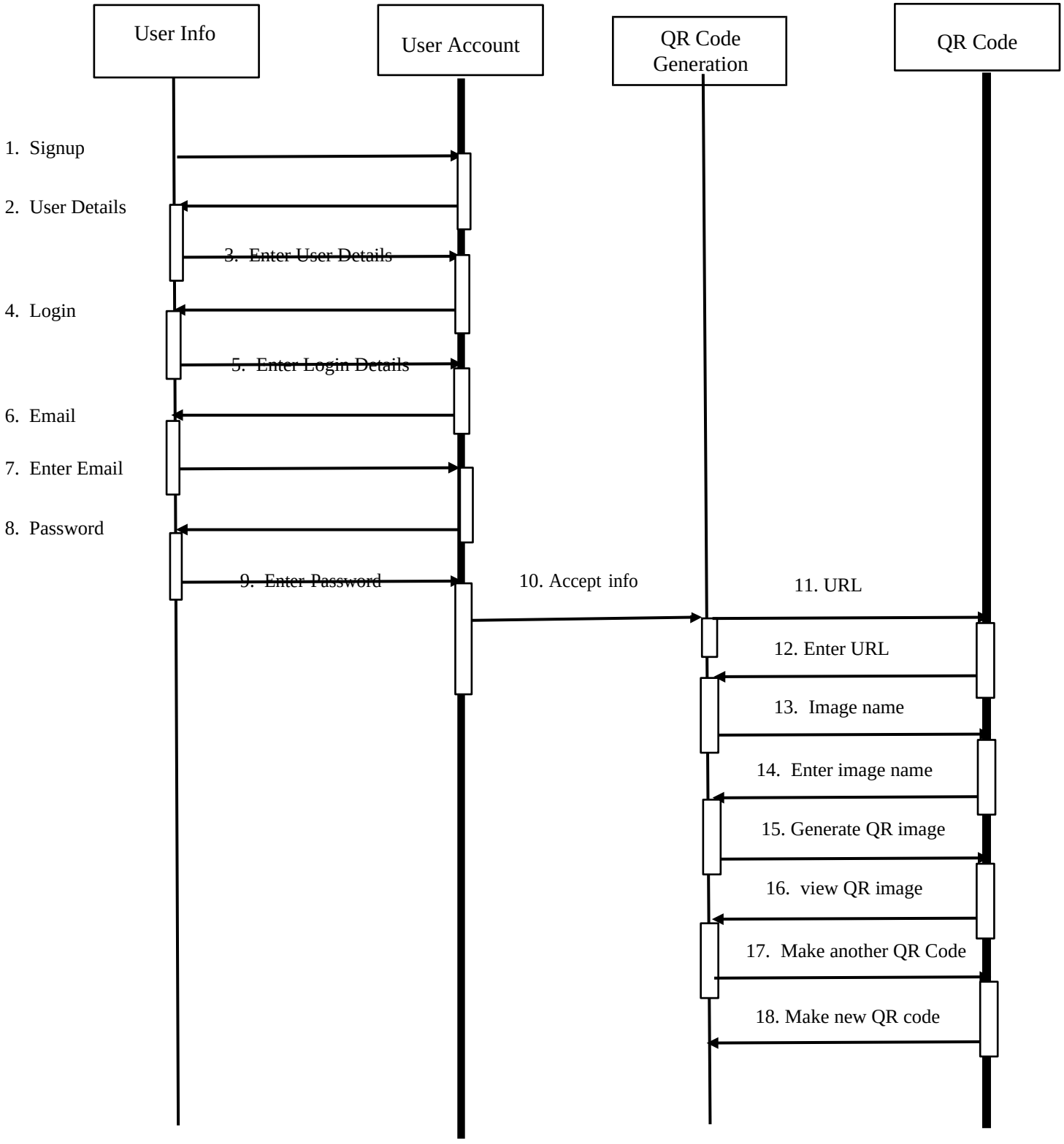

Fig 3.18 Sequence Diagram

3.3.9 State transition Diagram

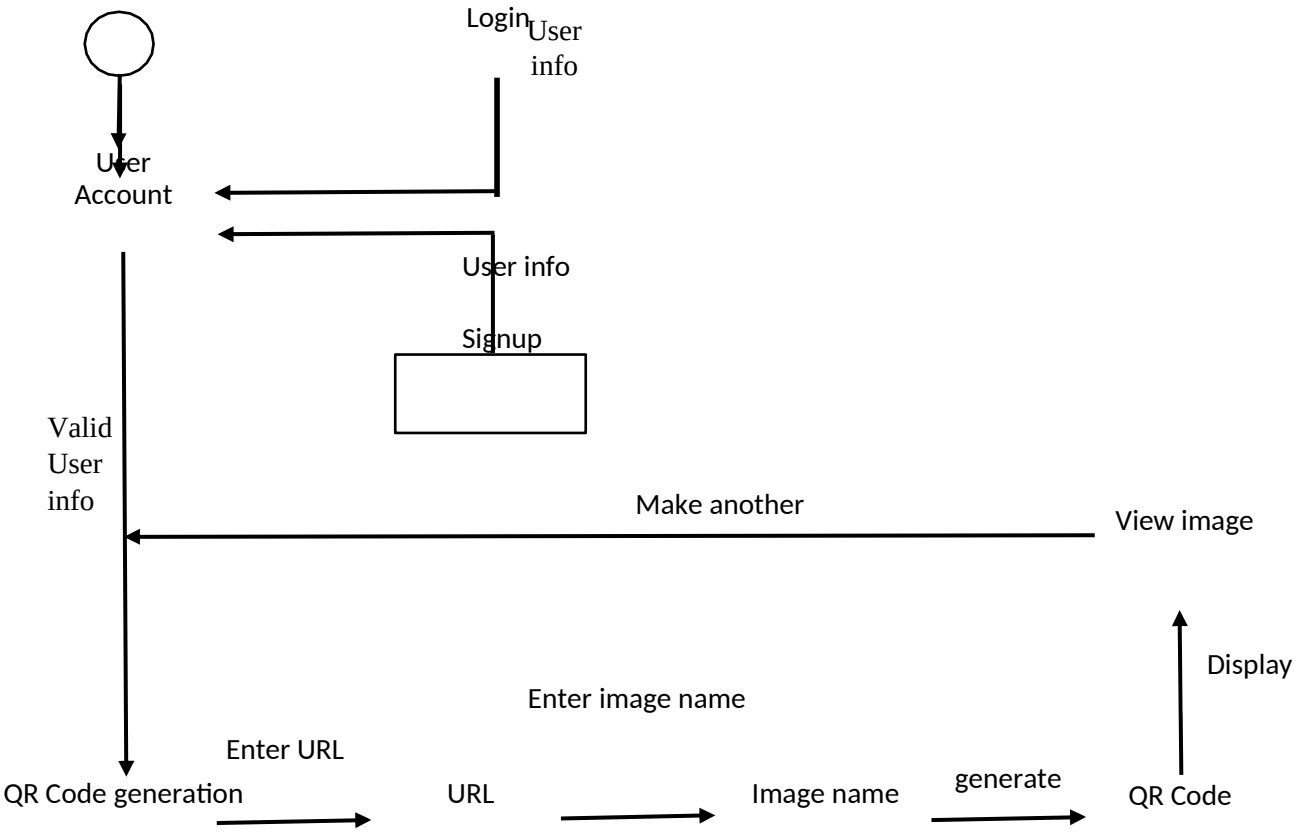

Fig 3.19 State transition Diagram

3.3.10 Use Case Diagram

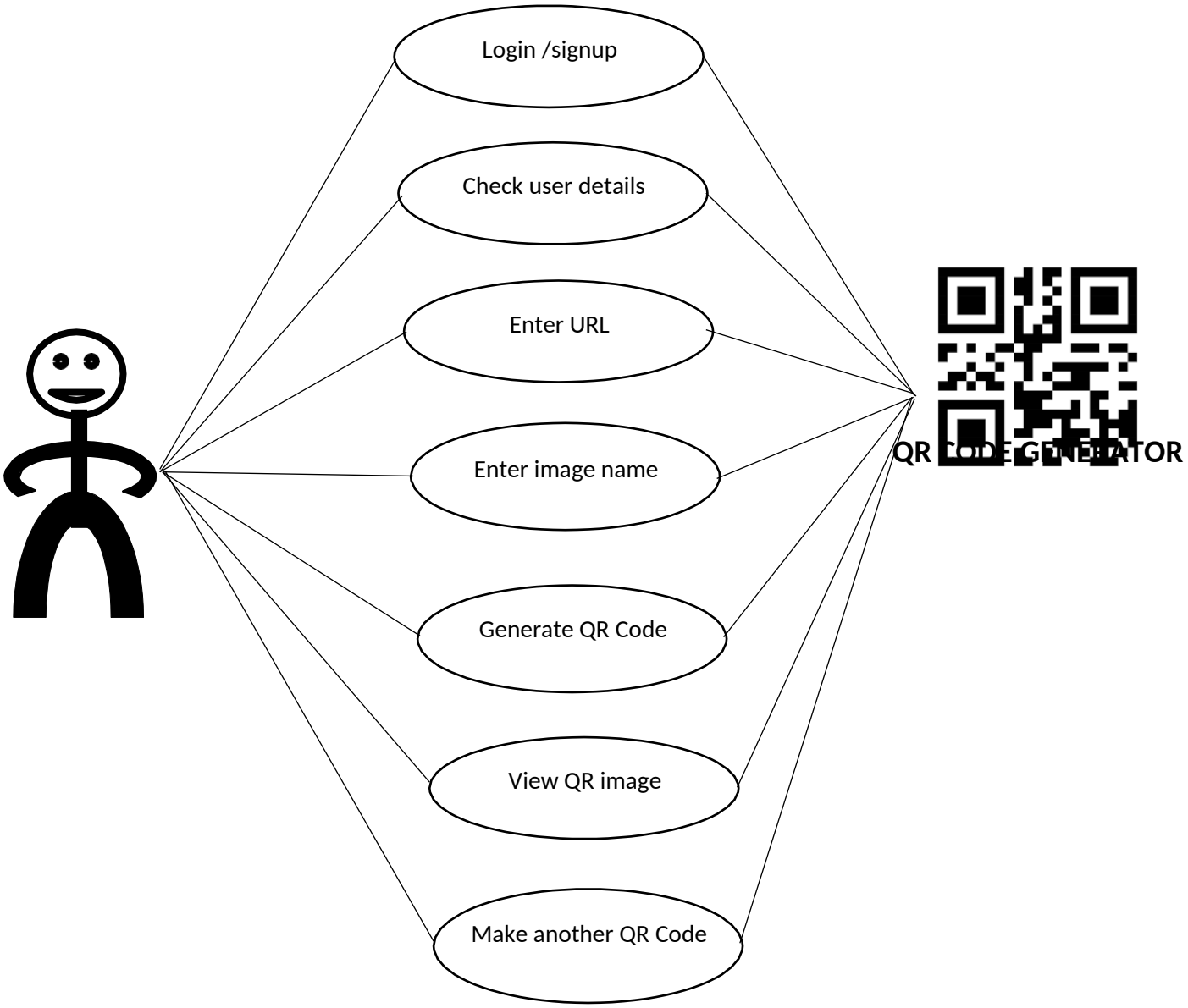

Fig 3.20 Use Case Diagram

Chapter 4

Implementation and Testing

4.1 Coding

4.1.1 Code

- **Login and Signup Screen**

```
from kivy.lang import Builder
from kivymd.app import MDApp
from kivy.uix.screenmanager import ScreenManager, Screen
import json
from kivymd.uix.button import MDFlatButton
from kivymd.uix.dialog import MDDialog
import requests
from kivy.core.window import Window
Window.size = 360,640


help_str = '''
ScreenManager:
    WelcomeScreen:
    MainScreen:
    LoginScreen:
    SignupScreen:

<WelcomeScreen>:
    name:'welcomescreen'
    MDLabel:
        text:'Login'
        font_style:'H2'
        halign:'center'
        pos_hint: {'center_y':0.9}
    MDLabel:
        text:'&'

        font_style:'H2'
        halign:'center'
```

```
        pos_hint: {'center_y':0.7}
    MDLabel:
        text:'Signup'
        font_style:'H2'
        halign:'center'
        pos_hint: {'center_y':0.5}

    MDRaisedButton:
        text:'Login'
        pos_hint : {'center_x':0.4,'center_y':0.3}
        size_hint: (0.13,0.1)
        on_press:
            root.manager.current = 'loginscreen'
            root.manager.transition.direction = 'left'
    MDRaisedButton:
        text:'Signup'
        pos_hint : {'center_x':0.6,'center_y':0.3}
        size_hint: (0.13,0.1)
        on_press:
            root.manager.current = 'signupscreen'
            root.manager.transition.direction = 'left'

<LoginScreen>:
    name:'loginscreen'
    MDLabel:
        text:'Login'
        font_style:'H2'
        halign:'center'
        pos_hint: {'center_y':0.9}

    MDTextField:

        id:login_email
        pos_hint: {'center_y':0.6,'center_x':0.5}
        size_hint : (0.7,0.1)
        hint_text: 'Email'
        helper_text:'Required'
        helper_text_mode:  'on_error'
        icon_right: 'account'
        icon_right_color: app.theme_cls.primary_color
        required: True
        mode: "rectangle"
```

```
    MDTextField:
        id:login_password
        pos_hint: {'center_y':0.4,'center_x':0.5}
        size_hint : (0.7,0.1)
        hint_text: 'Password'
        helper_text:'Required'
        helper_text_mode:  'on_error'
        icon_right: 'account'
        icon_right_color: app.theme_cls.primary_color
        required: True
        mode: "rectangle"

    MDRaisedButton:
        text:'Login'
        size_hint: (0.13,0.07)
        pos_hint: {'center_x':0.5,'center_y':0.2}
        on_press:
            app.login()
            app.username_changer()



    MDTextButton:
        text: 'Create an account'
        pos_hint: {'center_x':0.5,'center_y':0.1}
        on_press:
            root.manager.current = 'signupscreen'
            root.manager.transition.direction = 'up'


<SignupScreen>:
    name:'signupscreen'
    MDLabel:
        text:'Signup'
        font_style:'H2'
        halign:'center'
        pos_hint: {'center_y':0.9}

    MDTextField:
        id:signup_email
        pos_hint: {'center_y':0.6,'center_x':0.5}
        size_hint : (0.7,0.1)
```

```
    hint_text: 'Email'
    helper_text:'Required'
    helper_text_mode:  'on_error'
    icon_right: 'account'
    icon_right_color: app.theme_cls.primary_color
    required: True
    mode: "rectangle"
MDTextField:
    id:signup_username
    pos_hint: {'center_y':0.75,'center_x':0.5}
    size_hint : (0.7,0.1)
    hint_text: 'Username'
    helper_text:'Required'
    helper_text_mode:  'on_error'
    icon_right: 'account'
    icon_right_color: app.theme_cls.primary_color
    required: True
MDTextField:
    id:signup_password
    pos_hint: {'center_y':0.4,'center_x':0.5}
    size_hint : (0.7,0.1)
    hint_text: 'Password'
    helper_text:'Required'
    helper_text_mode:  'on_error'
    icon_right: 'account'
    icon_right_color: app.theme_cls.primary_color
    required: True
    mode: "rectangle"
MDRaisedButton:
    text:'Signup'
    size_hint: (0.13,0.07)
    pos_hint: {'center_x':0.5,'center_y':0.2}
    on_press: app.signup()

MDTextButton:
    text: 'Already have an account'
    pos_hint: {'center_x':0.5,'center_y':0.1}
    on_press:
        root.manager.current = 'loginscreen'
        root.manager.transition.direction = 'down'
```

```
<MainScreen>:
    name: 'mainscreen'
    MDLabel:
        id:username_info
        text:'Hello Main'
        font_style:'H1'
        halign:'center'

'''


class WelcomeScreen(Screen):
    pass


class MainScreen(Screen):
    pass


class LoginScreen(Screen):
    pass


class SignupScreen(Screen):
    pass


sm = ScreenManager()
sm.add_widget(WelcomeScreen(name='loginscreen'))
sm.add_widget(MainScreen(name='mainscreen'))
sm.add_widget(LoginScreen(name='loginscreen'))
sm.add_widget(SignupScreen(name='signupscreen'))


class LoginApp(MDApp):
    def build(self):
        self.strng = Builder.load_string(help_str)
        self.url = "https://qrlogin-8fb80-default-rtdb.firebaseio.com/.json"
        return self.strng
```

```
    def signup(self):
        signupEmail = self.strng.get_screen('signupscreen').ids.signup_email.text
        signupPassword = self.strng.get_screen('signupscreen').ids.signup_password.text
        signupUsername = self.strng.get_screen('signupscreen').ids.signup_username.text
        if signupEmail.split() == [] or signupPassword.split() == [] or signupUsername.split() == []:
            cancel_btn_username_dialogue = MDFlatButton(text='Retry', on_release=self.close_username_dialog)
            self.dialog = MDDialog(title='Invalid Input', text='Please Enter a valid Input', size_hint=(0.7, 0.2),
                                   buttons=[cancel_btn_username_dialogue])
            self.dialog.open()
        if len(signupUsername.split()) > 1:
            cancel_btn_username_dialogue = MDFlatButton(text='Retry', on_release=self.close_username_dialog)
            self.dialog = MDDialog(title='Invalid Username', text='Please enter username without space',
                                   size_hint=(0.7, 0.2), buttons=[cancel_btn_username_dialogue])
            self.dialog.open()
        else:
            print(signupEmail, signupPassword)
            signup_info = str(
                {f'\"{signupEmail}\":{{"Password":\"{signupPassword}\","Username":\"{signupUsername}\"}}'})
            signup_info = signup_info.replace(".", "-")
            signup_info = signup_info.replace("\'", "")
            to_database = json.loads(signup_info)
            print((to_database))
            requests.patch(url=self.url, json=to_database)
            self.strng.get_screen('loginscreen').manager.current = 'loginscreen'

    auth = 'DYmN81piotyN8VZWgyCeKyIK7Z6SFBn71MilAG9Z'

    def login(self):
        loginEmail = self.strng.get_screen('loginscreen').ids.login_email.text
        loginPassword = self.strng.get_screen('loginscreen').ids.login_password.text

        self.login_check = False
        supported_loginEmail = loginEmail.replace('.', '-')
        supported_loginPassword = loginPassword.replace('.', '-')
        request = requests.get(self.url + '?auth=' + self.auth)
        data = request.json()
        emails = set()
        for key, value in data.items():
            emails.add(key)
        if supported_loginEmail in emails and supported_loginPassword == data[supported_loginEmail]
['Password']:
            self.username = data[supported_loginEmail]['Username']
```

```
            self.login_check = True
            self.strng.get_screen('mainscreen').manager.current = 'mainscreen'
        else:
            print("user no longer exists")

    def close_username_dialog(self, obj):
        self.dialog.dismiss()

    def username_changer(self):
        if self.login_check:
            self.strng.get_screen('mainscreen').ids.username_info.text = f"welcome {self.username}"


LoginApp().run()
```

- **QR Code Generator with SQL database connectivity**

```
from kivy.metrics import dp
from qrcode import QRCode, ERROR_CORRECT_L
from kivymd.uix.label import MDLabel
from kivymd.uix.textfield import MDTextField
from kivymd.uix.button import MDRectangleFlatButton
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.image import Image
from kivy.lang import Builder
import qrcode
from kivymd.app import MDApp
from kivymd.uix.screen import Screen
import sqlite3
from kivy.core.window import Window

Window.size = (360, 640)

screen_helper = """
ScreenManager:
    QRCodeScreen:
    GeneratedQRCodeScreen:

<QRCodeScreen>:
    name: 'qrcode'
```

```
    MDLabel:
        text: "QR Code Generator"
        halign: 'center'
        theme_text_color: 'Secondary'
        font_style: 'H2'
        pos_hint: {'center_x':0.5, 'center_y':0.9}

    MDTextField:
        id: qr_text
        hint_text: "Enter text for QR code"
        size_hint_x: None
        width: 250
        pos_hint: {'center_x':0.5, 'center_y':0.6}

    MDRaisedButton:
        text: "Generate QR Code"
        size_hint_x: None
        width: 150
        pos_hint: {'center_x':0.5, 'center_y':0.4}
        on_press: app.generate_qr_code(qr_text.text)

    MDRaisedButton:
        text: "Go to Generated QR Code Screen"
        size_hint_x: None
        width: 250
        pos_hint: {'center_x':0.5, 'center_y':0.2}
        on_press: root.manager.current = 'generated_qrcode'

    MDRaisedButton:
        text: "Generate QR Code"
        size_hint_x: None
        width: 150
        pos_hint: {'center_x':0.5, 'center_y':0.4}
        on_press: app.generate_qr_code(root, qr_text.text)


<GeneratedQRCodeScreen>:
    name: 'generated_qrcode'

    MDLabel:
        text: "Generated QR Code"
```

```
        halign: 'center'
        theme_text_color: 'Secondary'
        font_style: 'H2'
        pos_hint: {'center_x':0.5, 'center_y':0.9}

    Image:
        id: qrcode_image
        source: ''
        size_hint: None, None
        size: 300, 300
        pos_hint: {'center_x':0.5, 'center_y':0.5}

    MDRaisedButton:
        text: "Generate Another QR Code"
        size_hint_x: None
        width: 250
        pos_hint: {'center_x':0.5, 'center_y':0.2}
        on_press: root.manager.current = 'qrcode'
"""


class QRCodeScreen(Screen):
    pass


class GeneratedQRCodeScreen(Screen):
    pass


class QRCodeApp(MDApp):

    def build(self):
        self.theme_cls.primary_palette = 'Blue'
        screen_manager = Builder.load_string(screen_helper)
        return screen_manager

    def on_start(self):
        self.conn = sqlite3.connect("qrcodes.db")
        self.cursor = self.conn.cursor()
        self.cursor.execute(
            "CREATE TABLE IF NOT EXISTS qrcodes (id INTEGER PRIMARY KEY AUTOINCREMENT,
data TEXT, image BLOB)"
```

```
        )
        self.conn.commit()

        self.theme_cls.primary_palette = "LightBlue"
        self.theme_cls.primary_hue = "A700"
        self.theme_cls.theme_style = "Light"
        Window.size = (360, 640)

        screen = Screen()

    def generate_qr_code(self, instance, qr_text):
        qr = qrcode.QRCode(version=1, box_size=10, border=5)
        qr.add_data(qr_text)
        qr.make(fit=True)
        img = qr.make_image(fill_color='black', back_color='white')
        img_path = 'qr_code.png'
        img.save(img_path)
        self.root.get_screen('generated_qrcode').ids.qrcode_image.source = img_path

        # Insert QR code into database
        with open("temp_qrcode.png", "rb") as f:
            image_data = f.read()
        self.cursor.execute(
            "INSERT INTO qrcodes (data, image) VALUES (?, ?)", (qr_text, image_data)
        )
        self.conn.commit()


if __name__ == '__main__':
    QRCodeApp().run()
```

- **SQLite Database Query**

```
SELECT * FROM qrcodes;
SELECT id,
data,
image
    FROM qrcodes;
```

- **kivy to apk command**

```
!pip install buildozer

!pip install cython==0.29.21

!sudo apt-get install -y \
    python3-pip \
    build-essential \
    git \
    python3 \
    python3-dev \
    ffmpeg \
    libsdl2-dev \
    libsdl2-image-dev \
    libsdl2-mixer-dev \
    libsdl2-ttf-dev \
    libportmidi-dev \
    libswscale-dev \
    libavformat-dev \
    libavcodec-dev \
    zlib1g-dev

!sudo apt-get install -y \
    libgstreamer1.0 \
    gstreamer1.0-plugins-base \
    gstreamer1.0-plugins-good

!sudo apt-get install build-essential libsqlite3-dev sqlite3 bzip2 libbz2-dev zlib1g-dev libssl-dev openssl libgdbm-dev libgdbm-compat-dev liblzma-dev libreadline-dev libncursesw5-dev libffi-dev uuid-dev libffi6



!sudo apt-get install libffi-dev

!buildozer init

!buildozer -v android debug

!buildozer android clean
```

- **Google Colab Buildozer Spec File**

```
[app]

# (str) Title of your application
title = QR Code Generator

# (str) Package name
package.name = qrcodegenerator

# (str) Package domain (needed for android/ios packaging)
package.domain = com.qrcodegenerator

# (str) Source code where the main.py live
source.dir = .

# (str) The main .py file to use
source.main = main.py

# (list) Source files to include (let empty to include all the files)
source.include_exts = py,png,jpg,kv,atlas

# (list) List of inclusions using pattern matching
#source.include_patterns = assets/*,images/*.png

# (list) Source files to exclude (let empty to not exclude anything)
#source.exclude_exts = spec

# (list) List of directory to exclude (let empty to not exclude anything)
#source.exclude_dirs = tests, bin, venv

# (list) List of exclusions using pattern matching
# Do not prefix with './'
#source.exclude_patterns = license,images/*/*.jpg

# (str) Application versioning (method 1)
version = 1.0.0

# (str) Application versioning (method 2)
# version.regex = __version__ = ['"](.*)['"]
# version.filename = %(source.dir)s/main.py
```

```
# (list) Application requirements
# comma separated e.g. requirements = sqlite3,kivy
requirements = python3, kivy, kivymd, qrcode, sqlite3, pillow

# (str) Custom source folders for requirements
# Sets custom source for any requirements with recipes
# requirements.source.kivy = ../../kivy

# (str) Presplash of the application
#presplash.filename = %(source.dir)s/data/content/qrcode.png

# (str) Icon of the application
icon.filename = content/qrcode.png.

# (list) Supported orientations
# Valid options are: landscape, portrait, portrait-reverse or landscape-reverse
orientation = portrait

# (list) List of service to declare
#services = NAME:ENTRYPOINT_TO_PY,NAME2:ENTRYPOINT2_TO_PY

#
# OSX Specific
#

# author = © Copyright Info

# change the major version of python used by the app
osx.python_version = 3

# Kivy version to use
osx.kivy_version = 1.9.1

#
# Android specific
#

# (bool) Indicate if the application should be fullscreen or not
fullscreen = 0

# (string) Presplash background color (for android toolchain)
```

Supported formats are: #RRGGBB #AARRGGBB or one of the following names:
red, blue, green, black, white, gray, cyan, magenta, yellow, lightgray,
darkgray, grey, lightgrey, darkgrey, aqua, fuchsia, lime, maroon, navy,
olive, purple, silver, teal.
#android.presplash_color = #FFFFFF

(string) Presplash animation using Lottie format.
see https://lottiefiles.com/ for examples and https://airbnb.design/lottie/
for general documentation.
Lottie files can be created using various tools, like Adobe After Effect or Synfig.
#android.presplash_lottie = "path/to/lottie/file.json"

(str) Adaptive icon of the application (used if Android API level is 26+ at runtime)
#icon.adaptive_foreground.filename = %(source.dir)s/data/icon_fg.png
#icon.adaptive_background.filename = %(source.dir)s/data/icon_bg.png

(list) Permissions
(See https://python-for-android.readthedocs.io/en/latest/buildoptions/#build-options-1 for all the supported syntaxes and properties)
#android.permissions = android.permission.INTERNET, (name=android.permission.WRITE_EXTERNAL_STORAGE;maxSdkVersion=18)

(list) features (adds uses-feature -tags to manifest)
#android.features = android.hardware.usb.host

(int) Target Android API, should be as high as possible.
#android.api = 31

(int) Minimum API your APK / AAB will support.
#android.minapi = 21

(int) Android SDK version to use
#android.sdk = 20

(str) Android NDK version to use
#android.ndk = 23b

(int) Android NDK API to use. This is the minimum API your app will support, it should usually match android.minapi.
#android.ndk_api = 21

(bool) Use --private data storage (True) or --dir public storage (False)

```
#android.private_storage = True

# (str) Android NDK directory (if empty, it will be automatically downloaded.)
#android.ndk_path =

# (str) Android SDK directory (if empty, it will be automatically downloaded.)
#android.sdk_path =

# (str) ANT directory (if empty, it will be automatically downloaded.)
#android.ant_path =

# (bool) If True, then skip trying to update the Android sdk
# This can be useful to avoid excess Internet downloads or save time
# when an update is due and you just want to test/build your package
# android.skip_update = False

# (bool) If True, then automatically accept SDK license
# agreements. This is intended for automation only. If set to False,
# the default, you will be shown the license when first running
# buildozer.
# android.accept_sdk_license = False

# (str) Android entry point, default is ok for Kivy-based app
#android.entrypoint = org.kivy.android.PythonActivity

# (str) Full name including package path of the Java class that implements Android Activity
# use that parameter together with android.entrypoint to set custom Java class instead of PythonActivity
#android.activity_class_name = org.kivy.android.PythonActivity

# (str) Extra xml to write directly inside the <manifest> element of AndroidManifest.xml
# use that parameter to provide a filename from where to load your custom XML code
#android.extra_manifest_xml = ./src/android/extra_manifest.xml

# (str) Extra xml to write directly inside the <manifest><application> tag of AndroidManifest.xml
# use that parameter to provide a filename from where to load your custom XML arguments:
#android.extra_manifest_application_arguments = ./src/android/extra_manifest_application_arguments.xml

# (str) Full name including package path of the Java class that implements Python Service
# use that parameter to set custom Java class which extends PythonService
#android.service_class_name = org.kivy.android.PythonService

# (str) Android app theme, default is ok for Kivy-based app
```

```
# android.apptheme = "@android:style/Theme.NoTitleBar"

# (list) Pattern to whitelist for the whole project
#android.whitelist =

# (str) Path to a custom whitelist file
#android.whitelist_src =

# (str) Path to a custom blacklist file
#android.blacklist_src =

# (list) List of Java .jar files to add to the libs so that pyjnius can access
# their classes. Don't add jars that you do not need, since extra jars can slow
# down the build process. Allows wildcards matching, for example:
# OUYA-ODK/libs/*.jar
#android.add_jars = foo.jar,bar.jar,path/to/more/*.jar

# (list) List of Java files to add to the android project (can be java or a
# directory containing the files)
#android.add_src =

# (list) Android AAR archives to add
#android.add_aars =

# (list) Put these files or directories in the apk assets directory.
# Either form may be used, and assets need not be in 'source.include_exts'.
# 1) android.add_assets = source_asset_relative_path
# 2) android.add_assets = source_asset_path:destination_asset_relative_path
#android.add_assets =

# (list) Put these files or directories in the apk res directory.
# The option may be used in three ways, the value may contain one or zero ':'
# Some examples:
# 1) A file to add to resources, legal resource names contain ['a-z','0-9','_']
# android.add_resources = my_icons/all-inclusive.png:drawable/all_inclusive.png
# 2) A directory, here  'legal_icons' must contain resources of one kind
# android.add_resources = legal_icons:drawable
# 3) A directory, here 'legal_resources' must contain one or more directories,
# each of a resource kind:  drawable, xml, etc...
# android.add_resources = legal_resources
#android.add_resources =
```

```
# (list) Gradle dependencies to add
#android.gradle_dependencies =

# (bool) Enable AndroidX support. Enable when 'android.gradle_dependencies'
# contains an 'androidx' package, or any package from Kotlin source.
# android.enable_androidx requires android.api >= 28
#android.enable_androidx = True

# (list) add java compile options
# this can for example be necessary when importing certain java libraries using the 'android.gradle_dependencies' option
# see https://developer.android.com/studio/write/java8-support for further information
# android.add_compile_options = "sourceCompatibility = 1.8", "targetCompatibility = 1.8"

# (list) Gradle repositories to add {can be necessary for some android.gradle_dependencies}
# please enclose in double quotes
# e.g. android.gradle_repositories = "maven { url 'https://kotlin.bintray.com/ktor' }"
#android.add_gradle_repositories =

# (list) packaging options to add
# see https://google.github.io/android-gradle-dsl/current/com.android.build.gradle.internal.dsl.PackagingOptions.html
# can be necessary to solve conflicts in gradle_dependencies
# please enclose in double quotes
# e.g. android.add_packaging_options = "exclude 'META-INF/common.kotlin_module'", "exclude 'META-INF/*.kotlin_module'"
#android.add_packaging_options =

# (list) Java classes to add as activities to the manifest.
#android.add_activities = com.example.ExampleActivity

# (str) OUYA Console category. Should be one of GAME or APP
# If you leave this blank, OUYA support will not be enabled
#android.ouya.category = GAME

# (str) Filename of OUYA Console icon. It must be a 732x412 png image.
#android.ouya.icon.filename = %(source.dir)s/data/ouya_icon.png

# (str) XML file to include as an intent filters in <activity> tag
#android.manifest.intent_filters =

# (list) Copy these files to src/main/res/xml/ (used for example with intent-filters)
```

```
#android.res_xml = PATH_TO_FILE,

# (str) launchMode to set for the main activity
#android.manifest.launch_mode = standard

# (str) screenOrientation to set for the main activity.
# Valid values can be found at https://developer.android.com/guide/topics/manifest/activity-element
#android.manifest.orientation = fullSensor

# (list) Android additional libraries to copy into libs/armeabi
#android.add_libs_armeabi = libs/android/*.so
#android.add_libs_armeabi_v7a = libs/android-v7/*.so
#android.add_libs_arm64_v8a = libs/android-v8/*.so
#android.add_libs_x86 = libs/android-x86/*.so
#android.add_libs_mips = libs/android-mips/*.so

# (bool) Indicate whether the screen should stay on
# Don't forget to add the WAKE_LOCK permission if you set this to True
#android.wakelock = False

# (list) Android application meta-data to set (key=value format)
#android.meta_data =

# (list) Android library project to add (will be added in the
# project.properties automatically.)
#android.library_references =

# (list) Android shared libraries which will be added to AndroidManifest.xml using <uses-library> tag
#android.uses_library =

# (str) Android logcat filters to use
#android.logcat_filters = *:S python:D

# (bool) Android logcat only display log for activity's pid
#android.logcat_pid_only = False

# (str) Android additional adb arguments
#android.adb_args = -H host.docker.internal

# (bool) Copy library instead of making a libpymodules.so
#android.copy_libs = 1
```

```
# (list) The Android archs to build for, choices: armeabi-v7a, arm64-v8a, x86, x86_64
# In past, was `android.arch` as we weren't supporting builds for multiple archs at the same time.
android.archs = arm64-v8a, armeabi-v7a

# (int) overrides automatic versionCode computation (used in build.gradle)
# this is not the same as app version and should only be edited if you know what you're doing
# android.numeric_version = 1

# (bool) enables Android auto backup feature (Android API >=23)
android.allow_backup = True

# (str) XML file for custom backup rules (see official auto backup documentation)
# android.backup_rules =

# (str) If you need to insert variables into your AndroidManifest.xml file,
# you can do so with the manifestPlaceholders property.
# This property takes a map of key-value pairs. (via a string)
# Usage example : android.manifest_placeholders = [myCustomUrl:\"org.kivy.customurl\"]
# android.manifest_placeholders = [:]

# (bool) Skip byte compile for .py files
# android.no-byte-compile-python = False

# (str) The format used to package the app for release mode (aab or apk or aar).
# android.release_artifact = aab

# (str) The format used to package the app for debug mode (apk or aar).
# android.debug_artifact = apk

#
# Python for android (p4a) specific
#

# (str) python-for-android URL to use for checkout
#p4a.url =

# (str) python-for-android fork to use in case if p4a.url is not specified, defaults to upstream (kivy)
#p4a.fork = kivy

# (str) python-for-android branch to use, defaults to master
#p4a.branch = master
```

```
# (str) python-for-android specific commit to use, defaults to HEAD, must be within p4a.branch
#p4a.commit = HEAD

# (str) python-for-android git clone directory (if empty, it will be automatically cloned from github)
#p4a.source_dir =

# (str) The directory in which python-for-android should look for your own build recipes (if any)
#p4a.local_recipes =

# (str) Filename to the hook for p4a
#p4a.hook =

# (str) Bootstrap to use for android builds
# p4a.bootstrap = sdl2

# (int) port number to specify an explicit --port= p4a argument (eg for bootstrap flask)
#p4a.port =

# Control passing the --use-setup-py vs --ignore-setup-py to p4a
# "in the future" --use-setup-py is going to be the default behaviour in p4a, right now it is not
# Setting this to false will pass --ignore-setup-py, true will pass --use-setup-py
# NOTE: this is general setuptools integration, having pyproject.toml is enough, no need to generate
# setup.py if you're using Poetry, but you need to add "toml" to source.include_exts.
#p4a.setup_py = false

# (str) extra command line arguments to pass when invoking pythonforandroid.toolchain
#p4a.extra_args =



#
# iOS specific
#

# (str) Path to a custom kivy-ios folder
#ios.kivy_ios_dir = ../kivy-ios
# Alternately, specify the URL and branch of a git checkout:
ios.kivy_ios_url = https://github.com/kivy/kivy-ios
ios.kivy_ios_branch = master

# Another platform dependency: ios-deploy
# Uncomment to use a custom checkout
#ios.ios_deploy_dir = ../ios_deploy
```

```
# Or specify URL and branch
ios.ios_deploy_url = https://github.com/phonegap/ios-deploy
ios.ios_deploy_branch = 1.10.0

# (bool) Whether or not to sign the code
ios.codesign.allowed = false

# (str) Name of the certificate to use for signing the debug version
# Get a list of available identities: buildozer ios list_identities
#ios.codesign.debug = "iPhone Developer: <lastname> <firstname> (<hexstring>)"

# (str) The development team to use for signing the debug version
#ios.codesign.development_team.debug = <hexstring>

# (str) Name of the certificate to use for signing the release version
#ios.codesign.release = %(ios.codesign.debug)s

# (str) The development team to use for signing the release version
#ios.codesign.development_team.release = <hexstring>

# (str) URL pointing to .ipa file to be installed
# This option should be defined along with `display_image_url` and `full_size_image_url` options.
#ios.manifest.app_url =

# (str) URL pointing to an icon (57x57px) to be displayed during download
# This option should be defined along with `app_url` and `full_size_image_url` options.
#ios.manifest.display_image_url =

# (str) URL pointing to a large icon (512x512px) to be used by iTunes
# This option should be defined along with `app_url` and `display_image_url` options.
#ios.manifest.full_size_image_url =


[buildozer]

# (int) Log level (0 = error only, 1 = info, 2 = debug (with command output))
log_level = 2

# (int) Display warning if buildozer is run as root (0 = False, 1 = True)
warn_on_root = 1

# (str) Path to build artifact storage, absolute or relative to spec file
```

```
# build_dir = ./.buildozer

# (str) Path to build output (i.e. .apk, .aab, .ipa) storage
# bin_dir = ./bin

#    ---------------------------------------------------------------------------
#    List as sections
#
#    You can define all the "list" as [section:key].
#    Each line will be considered as a option to the list.
#    Let's take [app] / source.exclude_patterns.
#    Instead of doing:
#
#[app]
#source.exclude_patterns = license,data/audio/*.wav,data/images/original/*
#
#    This can be translated into:
#
#[app:source.exclude_patterns]
#license
#data/audio/*.wav
#data/images/original/*
#

#    ---------------------------------------------------------------------------
#    Profiles
#
#    You can extend section / key with a profile
#    For example, you want to deploy a demo version of your application without
#    HD content. You could first change the title to add "(demo)" in the name
#    and extend the excluded directories to remove the HD content.
#
#[app@demo]
#title = My Application (demo)
#
#[app:source.exclude_patterns@demo]
#images/hd/*
#
#    Then, invoke the command line with the "demo" profile:
#
#buildozer --profile demo android debug
```

4.2 Test Cases

4.2.1 Test Case 1

- Test Case ID : QCG001
- Test Case Description : Login requires the e-mail and password of the user
- Initial State : Working
- Input : Enter the E-mail and Password

- Actual Output :

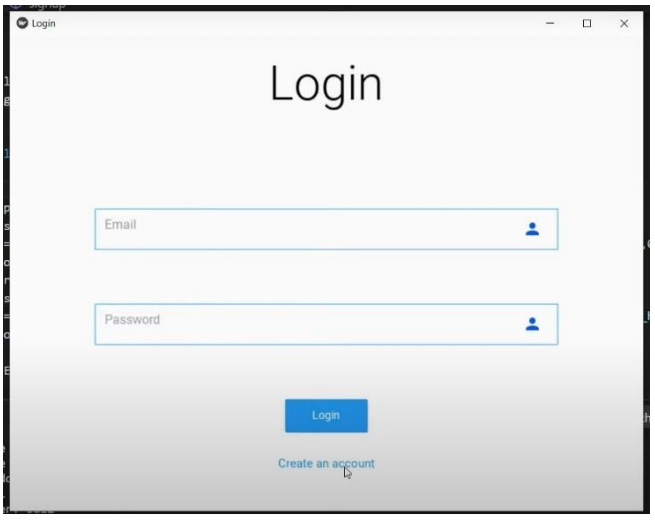

Fig 4.1 Login Page 1 (Actual Output)

- Expected Output
  :

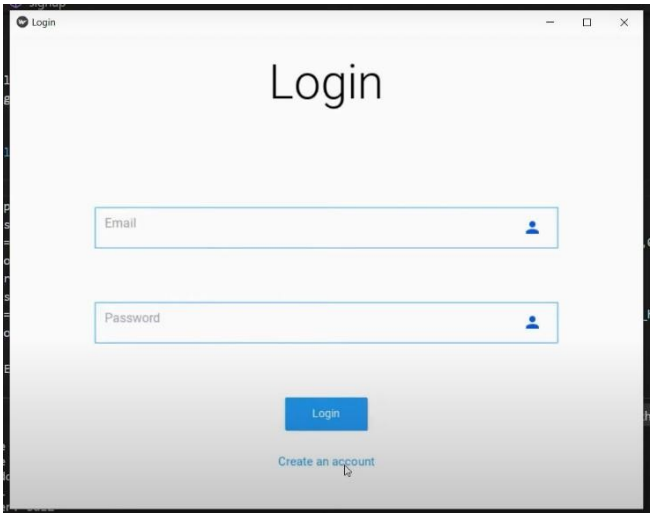

Fig 4.2 Login Page 1 (Expected Output)

- Remark : Actual output is equal to expected output. Hence, the system is working properly

4.2.2 Test Case 2

- Test Case ID : QCG002
- Test Case Description : User should enter all details to Signup
- Initial State : Working
- Input : Enter valid User Name, E-Mail, Password

- Actual Output :

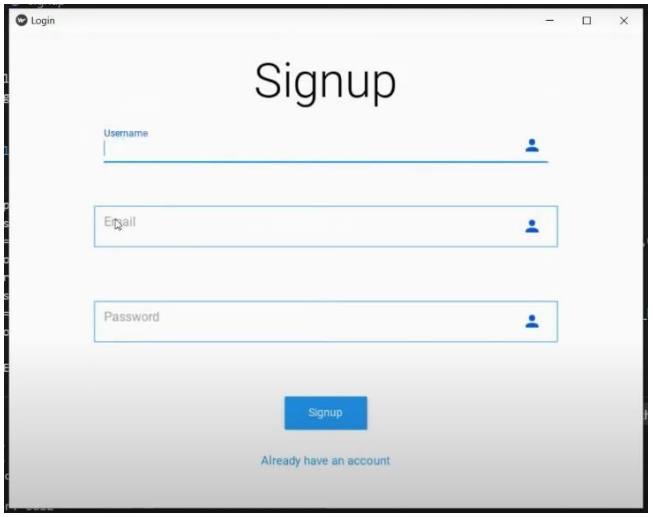

Fig 4.3 Signup Page 1 (Actual Output)

- Expected Output :

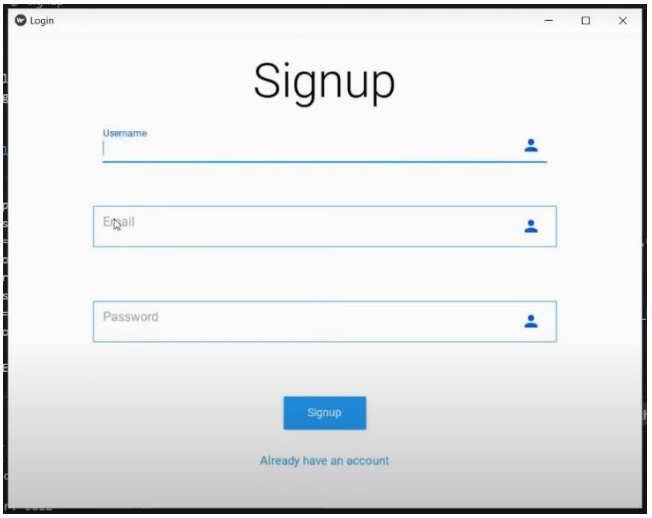

Fig 4.4 Signup Page 1 (Expected Output)

- Remark : Actual output is equal to expected output. Hence, the system is working properly.

4.2.3 Test Case 3

- Test Case ID : QCG003
- Test Case Description : Login requires the e-mail and password of the user
- Initial State : Not working
- Input : Enter the E-mail and Password

- Actual Output :

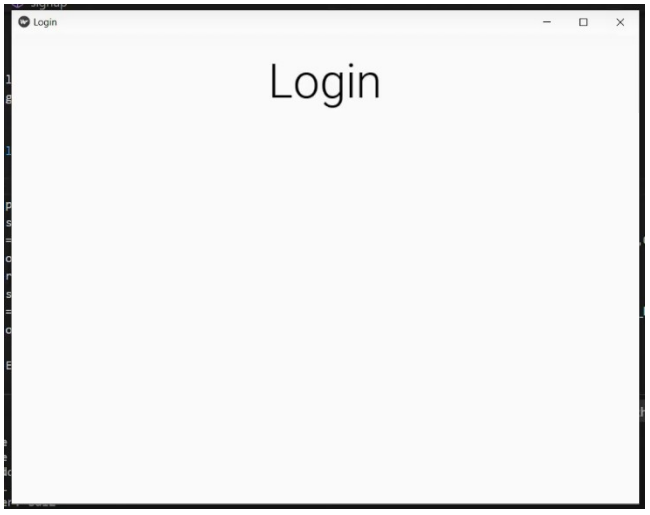

Fig 4.5 Login Page 2 (Actual Output)

- Expected Output :

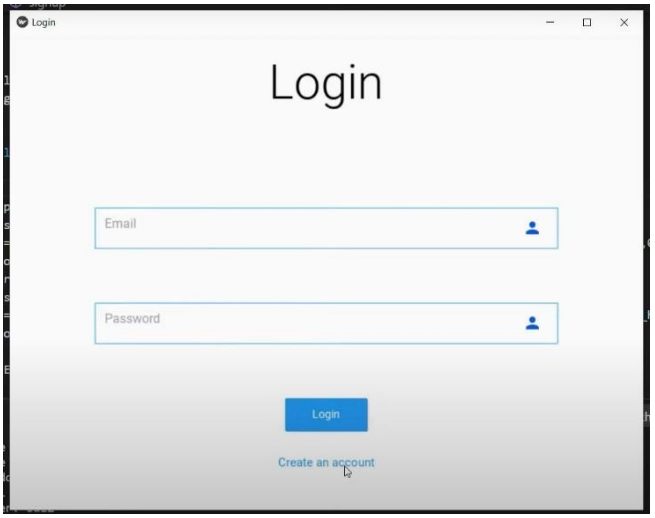

Fig 4.6 Login Page 2 (Expected Output)

- Remark : Actual output is not equal to expected output. Hence, the system is not working properly.

4.2.4 Test Case 4

- Test Case ID : QCG004
- Test Case Description : User should enter all details to Signup
- Initial State : Not working
- Input : Enter valid User Name, E-Mail, Password

- Actual Output :

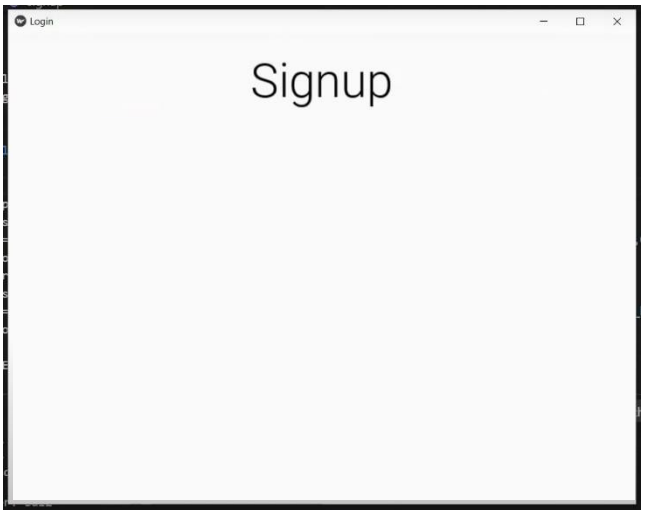

Fig 4.7 Signup Page 2 (Actual Output)

- Expected Output :

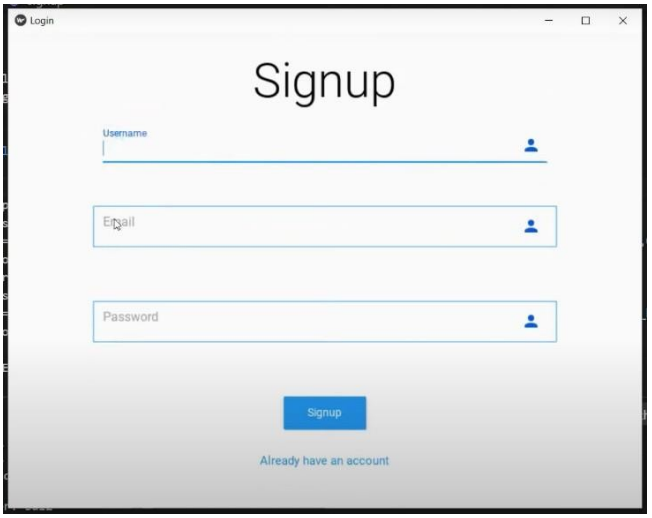

Fig 4.8 Signup Page 2 (Expected Output)

- Remark : Actual output is not equal to expected output. Hence, the system is not working properly.

4.2.5 Test Case 5

- Test Case ID : QCG005
- Test Case Description : QR Code Generator Home Page 1
- Initial State : Working
- Input : Enter text to generate QR Code

- Actual Output :

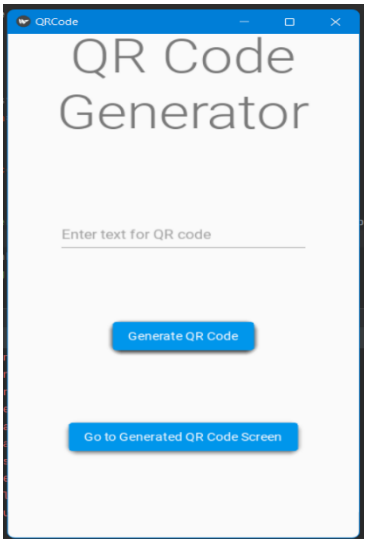

Fig 4.9 QR Code Generator Home Page 1 (Actual Output)

- Expected Output :

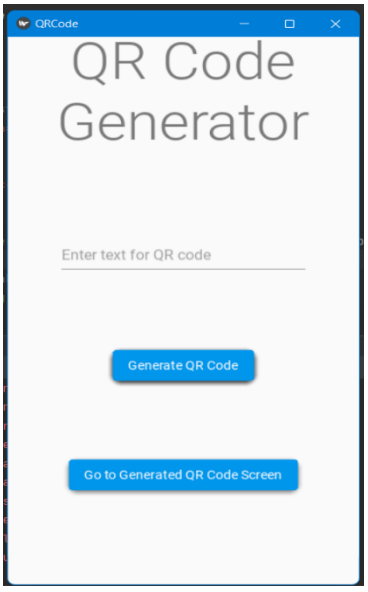

Fig 4.10 QR Code Generator Home Page 1 (Expected Output)

- Remark : Actual output is equal to expected output. Hence, system is working properly.

4.2.6 Test Case 6

- Test Case ID : QCG006
- Test Case Description : QR Code Generator Home Page 2
- Initial State : Not working
- Input : Enter text to generate QR Code

- Actual Output :

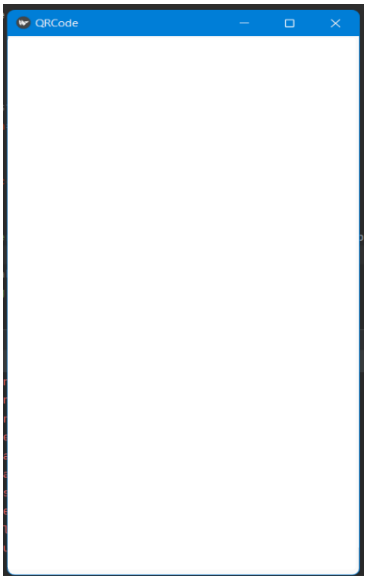

Fig 4.11 QR Code Generator Home Page 2 (Actual Output)

- Expected Output :

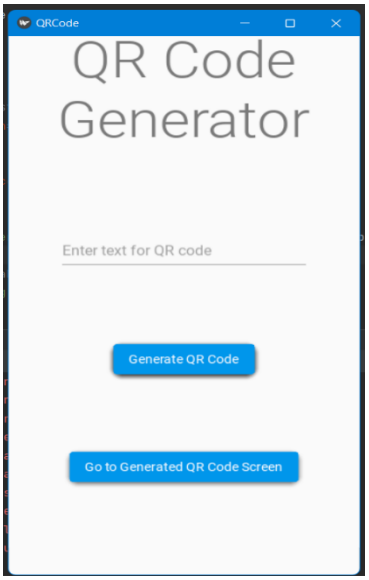

Fig 4.12 QR Code Generator Home Page 2 (Expected Output)

- Remark : Actual output is not equal to expected output. Hence, the system is not working properly.

4.2.7 Test Case 7

- Test Case ID : QCG007
- Test Case Description : View QR image
- Initial State : Working
- Input : Click on go to generated QR Code Screen

- Actual Output :

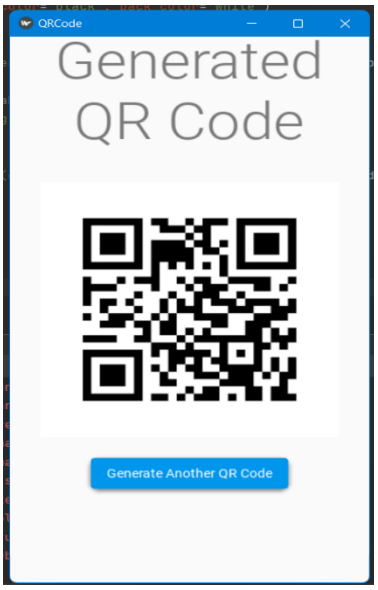

Fig 4.13 View QR image 1 (Actual Output)

- Expected Output :

Fig 4.14 View QR image 1 (Expected Output)

- Remark : Actual output is equal to expected output. Hence, system is working properly.

4.2.8 Test Case 8

- Test Case ID : QCG008
- Test Case Description : View QR image
- Initial State : Not working
- Input : Click on go to generated QR Code Screen

- Actual Output :

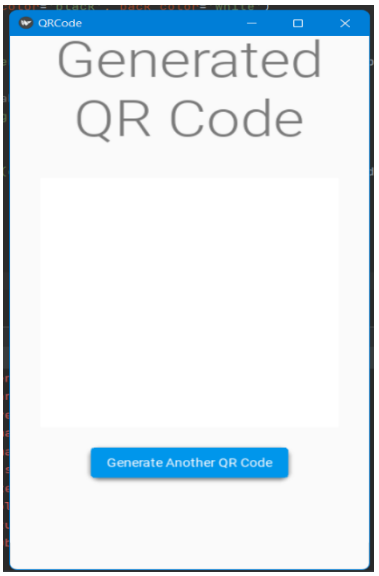

Fig 4.15 View QR image 2 (Actual Output)

- Expected Output :

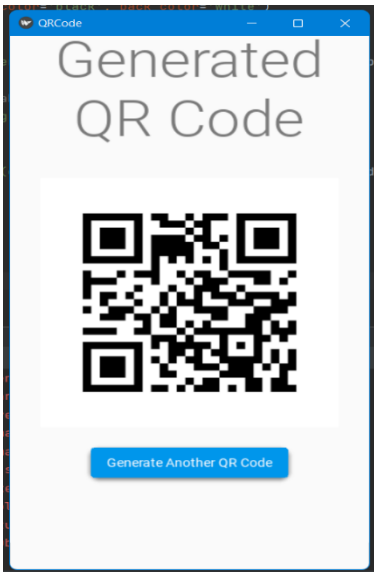

Fig 4.16 View QR image 2 (Expected Output)

- Remark : Actual output is not equal to expected output. Hence, the system is not working properly.

4.2.9 Test Case 9

- Test Case ID : QCG009
- Test Case Description : Generate QR Code
- Initial State :Not Working
- Input : Click on Generate QR Code

- Actual Output :

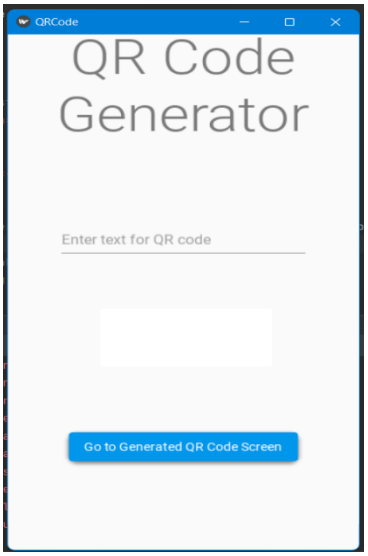

Fig 4.17 Generate QR Code (Actual Output)

- Expected Output :

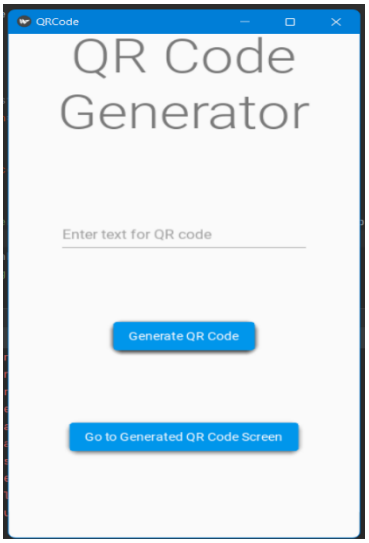

Fig 4.18 Generate QR Code (Expected Output)

- Remark : Actual output is equal to expected output. Hence, system is not working properly.

4.2.10 Test Case 10

- Test Case ID : QCG010
- Test Case Description : Go to generated QR Code Screen
- Initial State : Not working
- Input : Click on go to generated QR Code screen

- Actual Output :

Fig 4.19 Go to generated QR Code Screen (Actual Output)

- Expected Output :

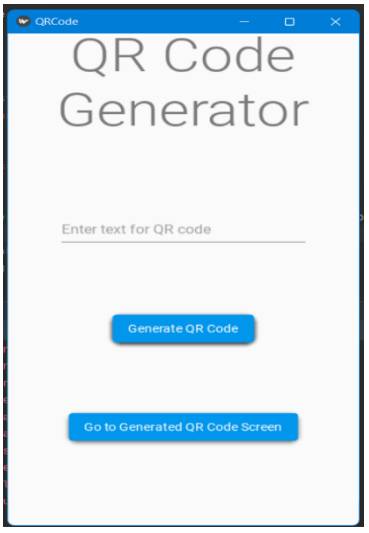

Fig 4.20 Go to generated QR Code Screen (Expected Output)

- Remark : Actual output is not equal to expected output. Hence, the system is not working properly.

Chapter 5

Result and Discussion

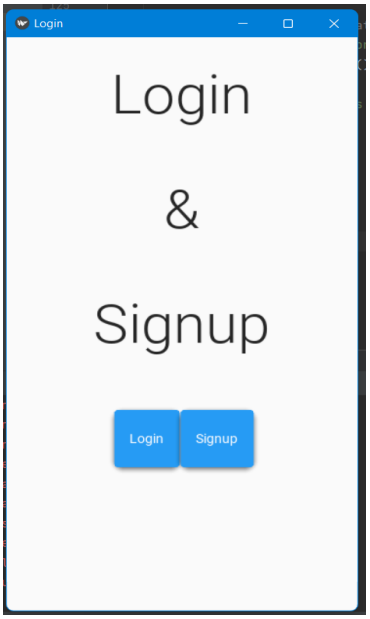

Fig 5.1 Login & Signup

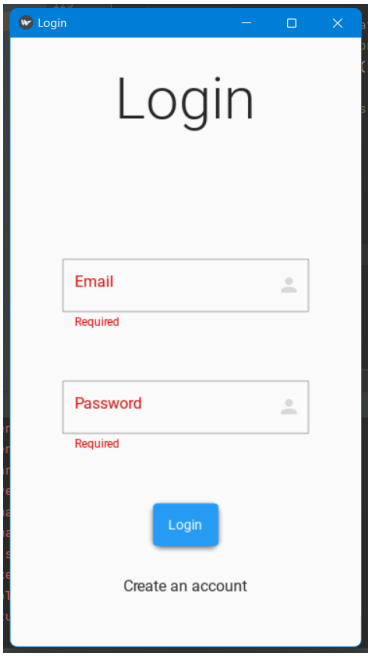

Fig 5.2 User Login

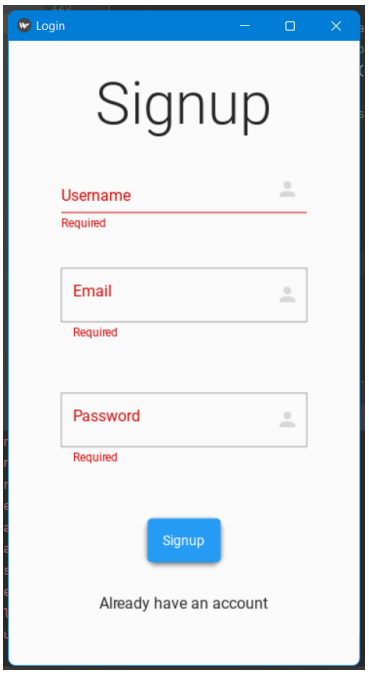

Fig 5.3 User Signup

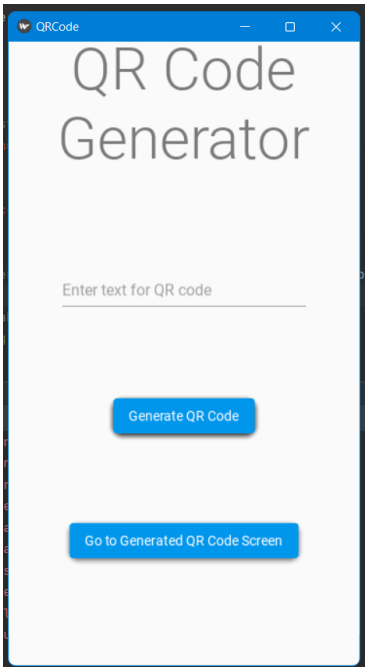

Fig 5.4 Home page

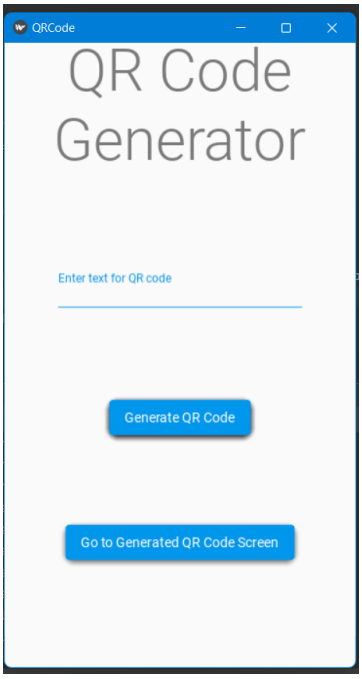

Fig 5.5 Enter text for QR Code

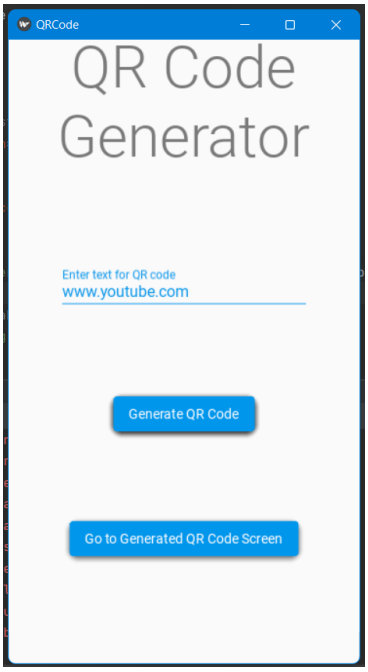

Fig 5.6 Generate QR Code

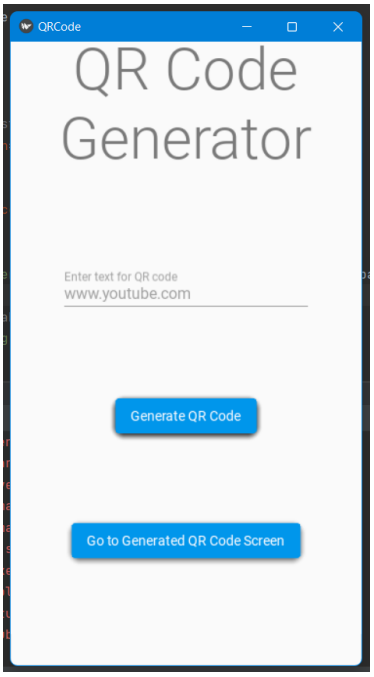

Fig 5.7 Go to Generated QR Code Screen

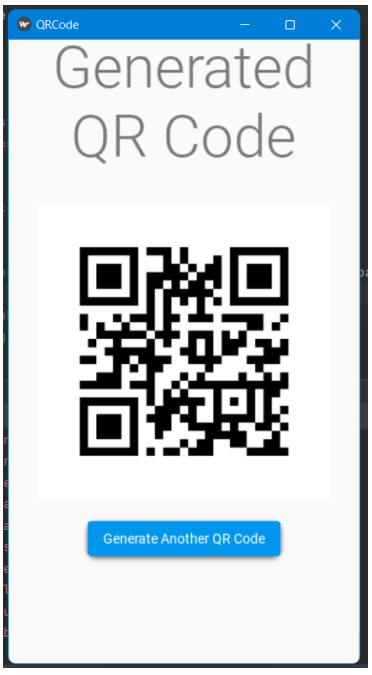

Fig 5.8 View QR Code

Fig 5.9 Generate another QR Code

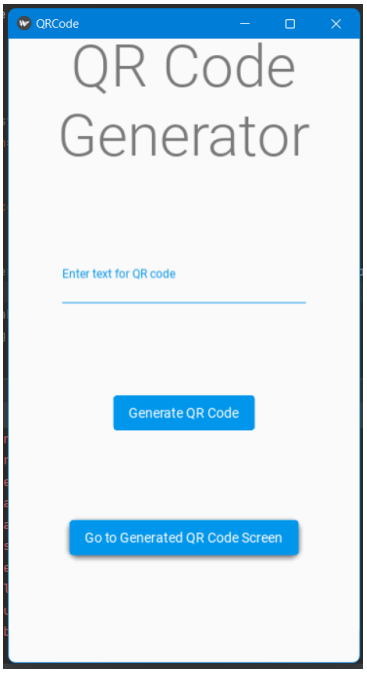

Fig 5.10 Home Page

Chapter 6

Conclusion and Future Work

The application of QR Codes is diversifying with their growing application in numerous industries. From contactless payments to promotions, QR Coded will be seen everywhere and if you're planning on running a campaign, you must ponder overriding the wave of contactless technology with QR Codes too. Today, nothing engages users better than an interactive experience. And with the QR Codes' ability to connect the offline world to online content, you can create interactive experiences for your audience. This could be to hand out coupons, get them to visit your website, or even show them a movie trailer.

Considering the benefits of QR Codes, their applications are going to grow manifold in the upcoming years. Many industries are using QR Codes already. QR Codes are everywhere. You might have seen them on the product packaging, billboards, business cards, museums, and restaurants. They are a form of contactless technology that bridges the gap between physical and digital experiences. Hence, many businesses and city admins are actively leveraging QR Codes for promotional and operational use cases. Today many businesses are looking forward to implementing advanced technologies to engage their audiences. The QR Codes can be used in every industry for different use cases, from sharing website links and text to multimedia, they can do it all.

In this QR code generator I will add some features like customizing QR code in different colors and different shapes, to look QR code more attractive. Also add some security password to each QR code so if user generates a QR code for a particular person or a group of persons, to read or see information or data they have to enter the security password.

Instead of make a universal QR code generator , I will make a QR code generator app for particular industries like smart menu card for restaurant industry, smart brochure for fashion ,advertising and marketing, smart QR code ID for companies, schools, colleges, also make a QR shopping app for shopping grocery, vegetables, fruits and so on.

Chapter 7

References

1. https://en.wikipedia.org/wiki/QR_code#History
2. https://www.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan
3. https://scanova.io/blog/what-is-a-qr-code-generator/
4. https://www.geeksforgeeks.org/difference-between-barcode-and-qr-code/
5. https://youtu.be/5l6YCXnXcb0
6. https://www.jetbrains.com/pycharm/
7. https://digitalfellows.commons.gc.c
8. https://digitalfellows.commons.gc.cuny.edu/2016/04/08/fun-times-with-sqlite-or-a-beginners-tutorial-to-data-management-and-databases-with-sql/