Project Documentation: QR Code Generator & Decoder

1. Project Overview

1.1 Problem Statement

In the digital age, sharing complex information (such as long URLs, Wi-Fi credentials, or contact details) between devices quickly and accurately is a common challenge. Manual data entry is prone to errors, and dependency on online tools for generating or reading QR codes poses privacy and security risks. Users require a lightweight, offline, and reliable local solution to bridge the gap between physical and digital data transfer.

1.2 Objectives

The objective of this project is to develop a Python-based command-line utility that allows users to:

- **Encode:** Convert text strings into standard QR code images (PNG format) for easy sharing.
- **Decode:** Read and extract information from existing QR code images.
- **Manage:** Provide a user-friendly interface to handle file inputs, outputs, and error checking locally without internet dependency.

2. Scope & Requirements

2.1 Functional Requirements

The system implements the following core functional modules:

1. **QR Code Generation (Encoding Module):**
    - Accepts user text input.
    - Generates a QR code image using standard versioning and error correction.
    - Saves the output as a .png file with automatic or custom naming.
    - Visualizes the generated image immediately.
2. **QR Code Reading (Decoding Module):**
    - Accepts a file path to an image.
    - Validates the existence of the file.
    - Scans the image for QR data using computer vision libraries (pyzbar).
    - Decodes and prints the textual content to the console.
3. **User Interface & Workflow Management:**
    - Provides a continuous menu loop.
    - Handles user inputs for navigation (Encode, Decode, Exit).

    - Manages program termination and navigation logic.

2.2 Non-Functional Requirements

1. **Usability:** The system utilizes a clear Command Line Interface (CLI) with distinct prompts, banners, and instructions, making it accessible to users with basic computer knowledge.
2. **Reliability & Error Handling:** The system implements robust try-except blocks to catch file permission errors, invalid image paths, or decoding failures, preventing the application from crashing unexpectedly.
3. **Performance:** The application utilizes optimized libraries (PIL and pyzbar) to ensure image generation and decoding occur in near real-time (milliseconds).
4. **Maintainability:** The code is structured into distinct functions (encode_qr, decode_qr, main), enabling easy updates or debugging of specific features without affecting the whole system.

This is a complete documentation structure tailored specifically to your code and the **VITyarthi Project Guidelines**. Since I cannot generate a downloadable .docx file directly, I have formatted the content below so you can simply **copy and paste** it into Microsoft Word or Google Docs.

I have ensured all specific requirements (3 functional modules, 4 non-functional requirements, correct diagram descriptions) are met.

Project Documentation: QR Code Generator & Decoder

1. Project Overview

1.1 Problem Statement

In the digital age, sharing complex information (such as long URLs, Wi-Fi credentials, or contact details) between devices quickly and accurately is a common challenge. Manual data entry is prone to errors, and dependency on online tools for generating or reading QR codes poses privacy and security risks. Users require a lightweight, offline, and reliable local solution to bridge the gap between physical and digital data transfer.

1.2 Objectives

The objective of this project is to develop a Python-based command-line utility that allows users to:

- **Encode:** Convert text strings into standard QR code images (PNG format) for easy sharing.
- **Decode:** Read and extract information from existing QR code images.
- **Manage:** Provide a user-friendly interface to handle file inputs, outputs, and error checking locally without internet dependency.

---

2. Scope & Requirements

2.1 Functional Requirements

The system implements the following core functional modules:

1. **QR Code Generation (Encoding Module):**
   - Accepts user text input.
   - Generates a QR code image using standard versioning and error correction.
   - Saves the output as a .png file with automatic or custom naming.
   - Visualizes the generated image immediately.
2. **QR Code Reading (Decoding Module):**
   - Accepts a file path to an image.
   - Validates the existence of the file.
   - Scans the image for QR data using computer vision libraries (pyzbar).
   - Decodes and prints the textual content to the console.
3. **User Interface & Workflow Management:**
   - Provides a continuous menu loop.
   - Handles user inputs for navigation (Encode, Decode, Exit).
   - Manages program termination and navigation logic.

2.2 Non-Functional Requirements

1. **Usability:** The system utilizes a clear Command Line Interface (CLI) with distinct prompts, banners, and instructions, making it accessible to users with basic computer knowledge.
2. **Reliability & Error Handling:** The system implements robust try-except blocks to catch file permission errors, invalid image paths, or decoding failures, preventing the application from crashing unexpectedly.
3. **Performance:** The application utilizes optimized libraries (PIL and pyzbar) to ensure image generation and decoding occur in near real-time (milliseconds).
4. **Maintainability:** The code is structured into distinct functions (encode_qr, decode_qr, main), enabling easy updates or debugging of specific features without affecting the whole system.

---

3. Technical Implementation

3.1 Technology Stack

- **Language:** Python 3.x
- **Core Libraries:**

- qrcode: For complying with standard QR code generation logic.
- Pillow (PIL): For image manipulation and file saving.
- pyzbar: For decoding barcodes/QR codes from images.
- os, sys, datetime: For file system operations and timestamping.

3.2 Architectural Design

The project follows a **Modular Procedural Architecture**. The main() function acts as the controller, routing user intent to specific service functions (encode_qr or decode_qr). Data flows from the user input Logic Module External Library File System.

3.3 Process Flow (Workflow)

1. **Start Application**
2. **Display Menu:** (1. Encode, 2. Decode, 3. Exit)
3. **If 1 (Encode):**
    - Input Text Input Filename Generate QR Save to Disk Show Image.
4. **If 2 (Decode):**
    - Input Path Check File Exists Load Image Decode Data Print Text.
5. **Loop:** Return to Menu unless Exit is selected.

3.4 Sequence Diagram (Example: Decoding)

1. **User enters option '2'.**
2. **System prompts for "Image Path".**
3. **User inputs path.**
4. **System calls os.path.exists().**
    - ***Alt:* If False, return error message.**
5. **System calls pyzbar.decode().**
6. **System extracts text data.**
7. **System displays "Decoded Text" to User.**