QR Vault

QR Vault is a secure web-based application that allows users to generate, encrypt, and manage QR codes for files and text, with user authentication, password protection, OTP-based operations, and an admin dashboard.

Features

- User and Admin Login/Registration with OTP verification
- Text & File encryption into QR codes
- QR Code decryption with optional password
- Download and manage QR entries
- Admin panel with OTP-secured access
- Profile view and secure password update via OTP
- Role-based access: mother admin can manage other admins
- OTPs and notifications via email using HTML templates

Tech Stack

- **Frontend:** HTML, Tailwind CSS, JavaScript
- **Backend:** Flask (Python)
- **Database:** MongoDB (Atlas)
- **Email Service:** Flask-Mail + SMTP
- **QR Generator:** `qrcode`, `pyzbar`, `filetype`
- **Deployment:** Local/Cloud
- **Security:** OTP verification, hashed passwords, `.env` config

---

Folder Structure

```
QR-Vault/
│
├── static/
│   ├── css/            # Tailwind styles
│   ├── js/             # Frontend logic
│   ├── images/         # Logos, icons
│   └── data/           # Downloads, temp files
│
├── templates/          # HTML pages (Jinja2 templating)
├── .env                # Secret keys and credentials (excluded via .gitignore)
├── .gitignore          # Git ignored files
```

```
├── app.py               # Main Flask application
├── auth.py              # Auth-related routes and logic
├── models.py            # MongoDB schema handlers
├── qr_utils.py          # QR operations (encode/decode/encrypt)
├── requirements.txt     # Project dependencies
├── LICENSE              # Project license
├── Readme.pdf           # PDF version of README
```

Setup Instructions (Local)

1. **Clone the Repository**

```
git clone https://github.com/atribiswas03/qr-vault.git
cd qr-vault
```

1. **Create a Virtual Environment**

```
python -m venv venv
venv\Scripts\activate   # Windows
# OR
source venv/bin/activate  # macOS/Linux
```

1. **Install Dependencies**

```
pip install -r requirements.txt
```

1. **Create `` File**

```
SECRET_KEY=your_secret_key
MONGO_URI=your_mongodb_connection_string
MAIL_SERVER=smtp.gmail.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=your_email@gmail.com
MAIL_PASSWORD=your_app_password
```

NOTE: Don't share this file. It's already excluded in `.gitignore`.

1. **Run the App**

```
python app.py
```

Then open your browser and go to: http://localhost:5000

Admin Roles

Step 1: Add Mother Admin

- Open your MongoDB collection (admin collection).
- Manually insert one entry with:
- `username`: admin
- `email`: your email id
- `role`: `admin`
- `is_mother_admin`: `true`
- `password`: hashed password using the same algorithm used in your app.

  ✅To generate password: hashed password using the same algorithm used in your app (e.g., `werkzeug.security.generate_password_hash('your_password')`)

This becomes the **first mother admin**.

Step 2: Login as Mother Admin

- Login using the above credentials on the Admin Login page.
- Once logged in, the **Mother Admin** has access to:
- Add new **Normal Admins** or **Mother Admins** (via OTP verification).
- View the list of existing admins.
- Delete any existing admin.
- Promote or demote between Normal and Mother admin roles.

Normal Admin Permissions

- A Normal Admin **cannot** add or delete other admins.
- A Normal Admin **can**:
- View all users and their QR data.
- View and respond to Contact Us queries.
- Send bulk mails or replies to users.

This role separation ensures administrative security and maintains controlled access.

Contributing

1. Fork the repository
2. Create a new branch: `git checkout -b feature-name`
3. Commit changes: `git commit -m "Add feature"`
4. Push to branch: `git push origin feature-name`
5. Submit a pull request

Contact

Atri Biswas\ Matribiswas2003@gmail.com\ GitHub

License