BARCODE AND QR CODE GENERATOR – DECODER

ABSTRACT

The Barcode and QR code generator and decoder application is a web-based platform that allows users to generate and decode barcodes and QR codes easily. The application provides a simple user interface, where users can log in with their credentials or sign up for free. The application stores all user information securely in a database. While generating barcodes and QR codes, users can input text in two ways, either by simple text input or by uploading a CSV file of texts. With the ability to input text in two ways, the application caters to users with varying needs. Users can upload multiple texts in one go, saving them time and effort. The application generates barcodes and QR codes for the given input, and users can download, share, or save the images to the database for future reference. The application ensures that the same image is not uploaded to the database twice. Users can view all saved images in the application and can also delete images from the database.

In addition to its user-friendly features, we aimed to containerize the application using Docker and deploy it on the Google Cloud Platform (GCP). The application's containerization using Docker provides a flexible and portable environment that allows for easier deployment and scaling. With Docker, the application can run on any platform, regardless of its underlying infrastructure. The containerization also enhances the application's security and reduces its operational overhead, ensuring that the application runs smoothly and efficiently.

With a secure login authentication process, the application ensures that user information is always protected. The application's database stores all user information and images securely, and users can access and manage their saved images at any time. With the provision of a test user account, users can explore the application's features without the need to sign up. Overall, the Barcode and QR code generator and decoder application provides a powerful tool for businesses, individuals, and organizations to generate and decode barcodes and QR codes easily. With its user-friendly features, secure login authentication, and containerization using Docker, and deployment on the GCP, the application provides a highly available, scalable, and cost-effective solution for all users.

FRAMEWORKS, PROGRAMMING LANGUAGES, & DATABASE

1. Angular

Details:
We have used Angular framework in the client side to develop a single page web application user interface. We used this framework because, it is a popular and widely used front-end web application framework that is used in many modern web applications and this provides a lot of features like routing, component-based architecture, efficient two-way data binding, command line interface and dependency injection.

Angular is an HTML and TypeScript-based platform and architecture for creating single-page applications. TypeScript is used to write Angular. It provides fundamental and additional functionality as a set of TypeScript libraries.

As soon as Angular was introduced, a lot of companies started using it for their applications.
Due to its faster end to end app development, the following companies have been using it for a long time:

- Google
- Gmail
- Microsoft Xbox
- Forbes
- PayPal
- Deutsche Bank
- Upwork
- The Guardian
- Weather.com
- Microsoft Office
- Mixer
- Jet Blue

Alternatives & Reasoning:
However, there are other front-end frameworks and libraries available that could be used instead of Angular. One popular alternative to Angular is React. React is a JavaScript library that is maintained by Facebook and is known for its high performance and flexibility. React uses a virtual DOM (Document Object Model) that allows for efficient updates to the user interface, making it ideal for large and complex applications. However, react can be complex and requires a steep learning curve for developers who are new to it.

Another popular alternative to Angular is Vue.js. Vue.js is a lightweight and easy-to-learn JavaScript framework that is designed for building user interfaces. Vue.js has a small learning curve and is highly flexible, making it ideal for both small and large-scale applications. Vue.js also provides powerful tools for creating reusable components, which can speed up the development process. However, Vue.js does not have the same level of community support and resources as Angular and React.

The reason behind choosing Angular for the Barcode and QR code generator and Decoder application is its robustness, scalability, and community support. Angular provides a comprehensive and well-structured framework for building complex web applications. Angular also has a large community of developers, which provides a wealth of resources, including documentation, tutorials, and plugins. Additionally, Angular has a strong focus on performance, which is essential for a web application that generates and decodes barcodes and QR codes in real-time.

In conclusion, while there are several alternatives to Angular, its robustness, scalability, and community support make it a strong choice for building complex web applications like the Barcode and QR code generator and decoder application.

References:

- “React vs Angular: Which JS Framework to Pick for Front-end Development?” by Nihar Raval, Feb 26, 2023, https://radixweb.com/blog/react-vs-angular
- "Angular vs Vue: A Head-to-Head Comparison" by Shanika Wickramasinghe, March 2021, https://kinsta.com/blog/angular-vs-vue/

2. Python Flask

Details:
We have used Flask framework of python in the server side for creating APIs for our application. Python Flask is a micro web framework that is used for building web applications, including the Barcode and QR code generator and Decoder application. Flask is known for its simplicity, flexibility, and ease of use. However, there are other web frameworks that could be used instead of Flask. Python Flask provides a wide range of features like:

- Built-in development server and fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Jinja2 templating
- support for secure cookies (client-side sessions)
- WSGI 1.0 compliant
- Unicode based.

Alternatives & Reasoning:
One alternative to Flask is Django. Django is a full-stack web framework that is built on Python. It is known for its "batteries included" philosophy, which means that it comes with many built-in features and tools that can speed up the development process. Django is well-suited for building complex web applications that require a lot of customization and scalability. However, Django can be more complex than Flask, and it may not be the best choice for small to medium-sized web applications.

For Python, one alternative is JavaScript. JavaScript is a widely used programming language that is primarily used for front-end web development. It is known for its versatility and its ability to run on multiple platforms. JavaScript is also well-suited for building web applications that require real-time updates and interactivity. However, JavaScript can be challenging for developers who are new to it, and it may not be the best choice for building large-scale back-end applications. And python provides a wide range of libraries to deal with images when compared with JavaScript.

The reason behind choosing Flask for the Barcode and QR code generator and decoder application is its simplicity, flexibility, and ease of use. Flask provides a clean and simple syntax that makes it

easy to read and understand, even for developers who are new to the framework. Flask is also lightweight, which means that it can be easily customized and extended to meet the specific needs of the application. Additionally, Flask has an active community of developers who provide extensive documentation and support, making it easier for developers to troubleshoot and solve problems.

In conclusion, while there are several alternatives to Flask, its simplicity, flexibility, and ease of use make it a strong choice for building web applications like the Barcode and QR code generator and decoder application.

References:

- "Flask vs Django in 2023: Which Framework to Choose and When?" by Ritesh Ranjan, Dec 7, 2022, https://www.netsolutions.com/insights/flask-vs-django/
- "Python vs JavaScript: Most Important Differences" by Vijay Singh Khatri, Apr 17, 2023, https://hackr.io/blog/python-vs-javascript

3. MongoDB

Details:
We have used MongoDB database to store our users and images information in the form of documented records in the database. MongoDB is an open-source document-oriented database that is designed to store a large scale of data and allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables. It is known for its flexibility, scalability, and performance. However, there are other databases that could be used instead of MongoDB. In MongoDB documents, you are allowed to store nested data. This nesting of data allows you to create complex relations between data and store them in the same document which makes the working and fetching of data extremely efficient as compared to SQL.

Alternatives & Reasoning:
One alternative to MongoDB is PostgreSQL. PostgreSQL is a powerful, open-source relational database that is known for its scalability and reliability. It is well-suited for handling large amounts of data and complex queries. PostgreSQL also supports many advanced features, such as full-text search, JSON storage, and geospatial queries. However, PostgreSQL can be more complex than MongoDB, and it may not be the best choice for applications that require a lot of data flexibility.

The reason behind choosing MongoDB for the Barcode and QR code generator and decoder application is its flexibility, scalability, and performance. MongoDB provides a flexible data model that makes it easy to store and retrieve complex data structures. It also provides built-in support for horizontal scaling, which makes it easy to handle large amounts of data and high-velocity data streams. Additionally, MongoDB has an active community of developers who provide extensive documentation and support, making it easier for developers to troubleshoot and solve problems.

In conclusion, while there are several alternatives to MongoDB, its flexibility, scalability, and performance make it a strong choice for storing and retrieving data in web applications like the Barcode and QR code generator and Decoder application.

References:
"Comparing MongoDB vs PostgreSQL", from official MongoDB website, https://www.mongodb.com/compare/mongodb-postgresql

PROJECT WORK AMONG TEAM MEMBERS

1. Sardar Thafzil Ahamed (sarda2t)

Client Side:

- I have worked on user interfaces of login, sign-up and home components of the Angular application.
- I am responsible to send the user login and sign-up information from client to the server.
- I have taken care of handling errors in the user authentication.
- I have added features to share images with others and save images to the database.

Server Side:

- I have worked on these API endpoints:
    1. Signup for creating a new user.
    2. Login to the application.
    3. Save an image to the database.
    4. Delete an image from the database.
    5. Delete all images from the database.

Docker:

- I have created a docker compose YAML file to run the application in one go by using the command "docker compose up"
- I have added a local MongoDB containerized image for running the application locally.

Database:

- For the database deployments, I have created a MongoDB database using MongoDB Atlas.
- Added a new project and created a new database called "itc_530".
- Responsible of maintaining the network access of the database deployments.

2. Yaswini Padmaraju (padma1y)

Client Side:

- I have worked on barcode and QR code generating components of the Angular application.
- I have added inputs to take user inputs for generating and decoding texts.

Server Side:

- I have worked on these API endpoints:
    1. Generate barcode for the given text.
    2. Generate QR code for the given text.
    3. Get count of images saved in the database.

Database:

- Created two database records called “user_records” and “images_records” in the created database.
- Responsible of maintaining records in the database.

3. Sai Kiran Debbadi (debba1s)

Client Side:

- I have worked on barcode and QR code decoding components of the Angular application.
- I have taken care of adding and processing a CSV file.

Server Side:

- I have worked on saved images from the database API endpoint.

Docker:

- I have created a Dockerfile for both client and server.
- I am responsible for deploying our application to the Google cloud platform.

4. Sri Srujani Kandula (kandu3s)

Client Side:

- I have handled logout functionality in the UI.
- Added few features like sharing the page with others and taking feedback from others.

Server Side:

- I have worked on these API endpoints:
  1. Generate Barcodes for the given CSV first 100 values.
  2. Generate QR codes for the given CSV first 100 values.
  3. Decode the barcode image.
  4. Decode the QR code image.

Docker

- I have added configuration files to run client application on Docker.
- I am responsible for maintaining the deployed application.

5. Riad Hossain (hossa1r)

Client Side:

- I added a download feature in the application for each image.
- Naming the image file while downloading.
- Handling the download of generated images from the CSV file

Server:

- I am responsible of creating connection between Flask and MongDB using MongoClient library of pymongo.

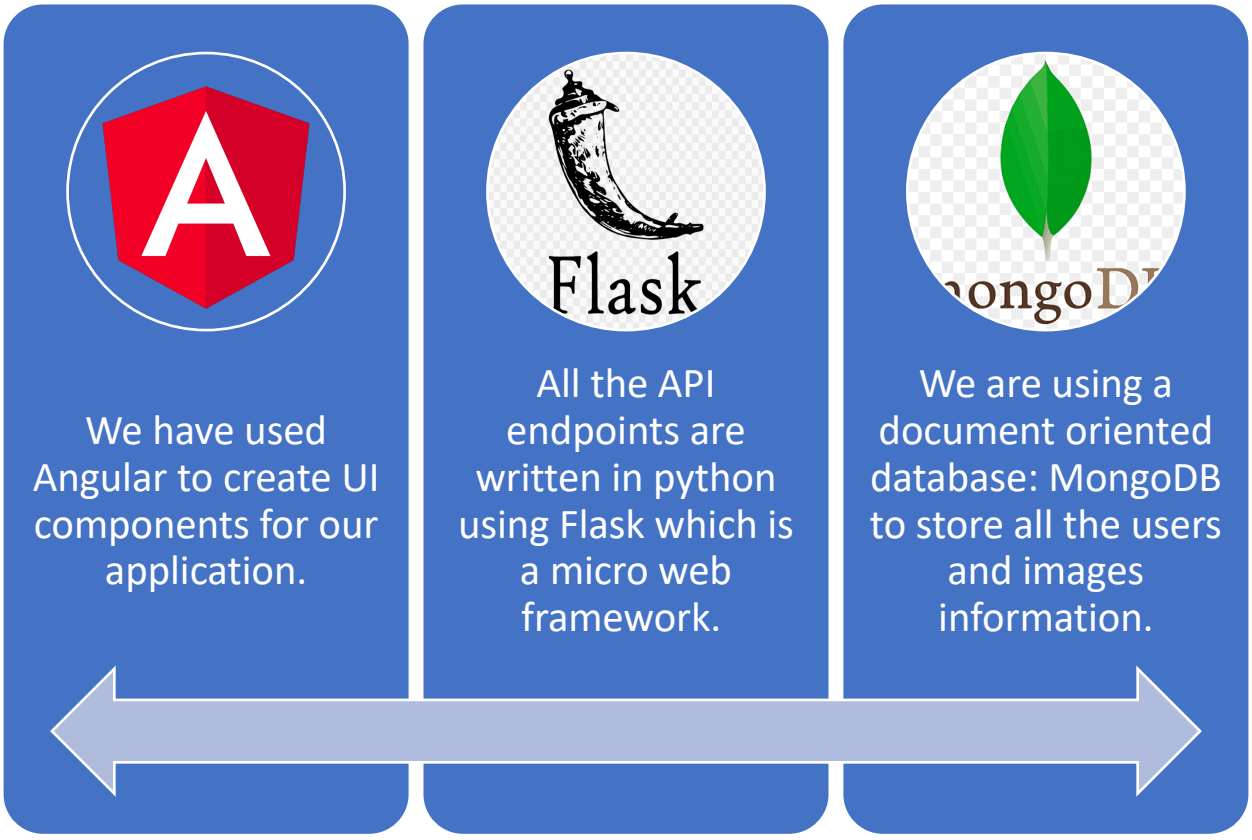
We have shared the work among five of us almost equally and completed the project. However, we have assigned ourselves a Leader to a specific work as below:

Client-side development and application testing: Sardar Thafzil Ahamed
Database deployment: Padmaraju Yaswini
Docker and deployment: Sai Kiran Debbadi
Server-side development: Sri Srujani Kandula
Handling Connections: Riad Hossain

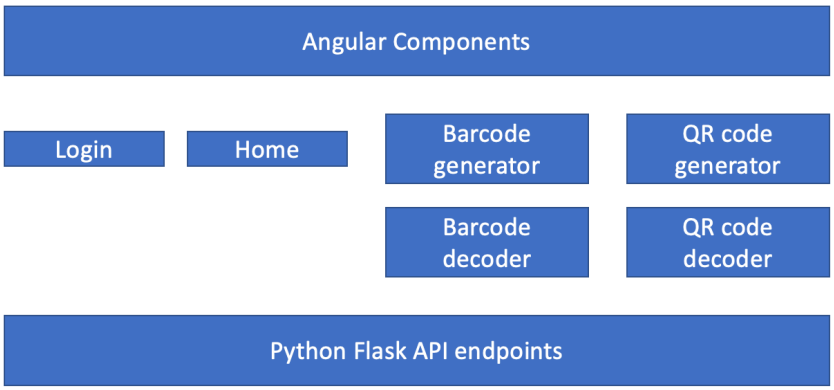
APPLICATION INTERCONNECTION AND DATAFLOW

Here, we are using Angular, Python's Flask and MongoDB.

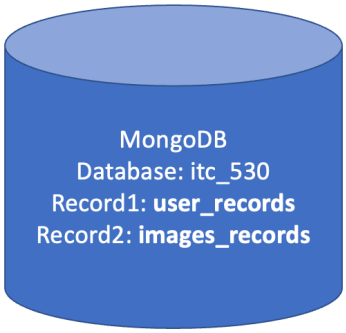
- Angular is a popular front-end web application framework that allows developers to create robust and responsive user interfaces.
- Python is a widely used programming language that will be used for the back-end service to generate barcodes and QR codes.
- MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.



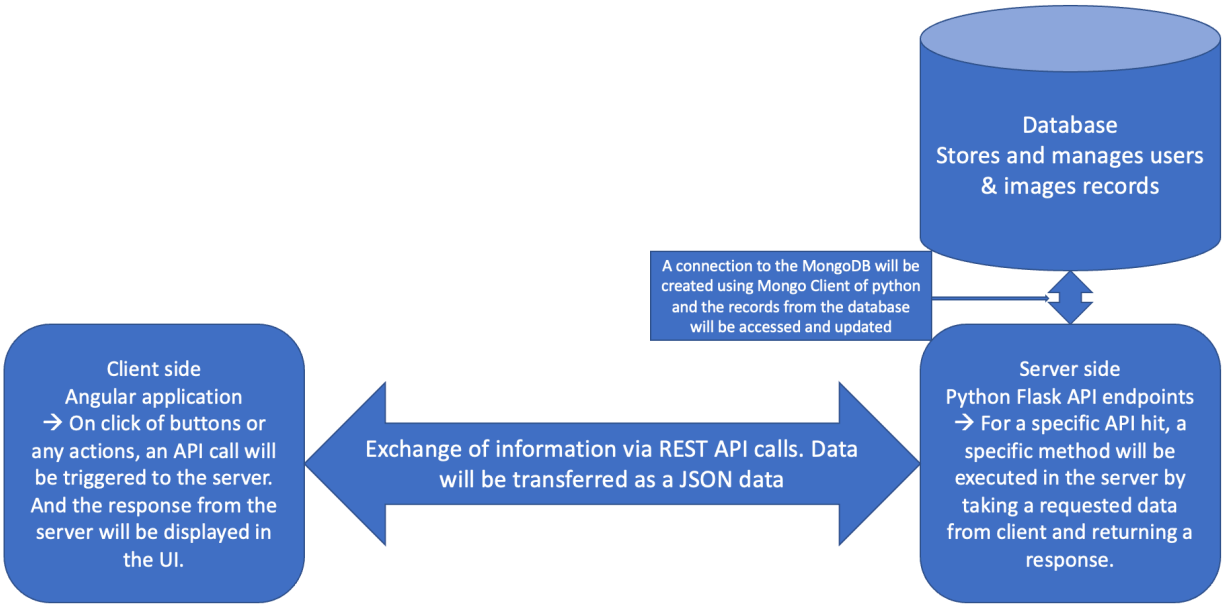
This is the in-detail of components in each section.



- Signup (/) POST
- Login (/login) POST
- Get count of images saved in the database (/get_saved_count) POST
- Save an image to the database (/save_to_db) POST
- Delete an image from the database (/delete_from_db) POST
- Delete all images from the database (/delete_all_images) POST
- Get saved images from the database (/get_from_db) POST
- Generate barcode for the given text (/barcode) POST
- Generate QR code for the given text (/qrcode) POST
- Generate Barcodes for the given CSV first 100 values (/barcode_csv) POST
- Generate QR codes for the given CSV first 100 values (/qrcode_csv) POST
- Decode the barcode image (/decode_barcode) POST
- Decode the QR code image (/decode_qrcode) POST



Interconnection of all these sections and dataflows is shown below.



PROGRESS OVER THE 3 IMPLEMENTATION WEEKS

What worked:

- Integration of Angular and Python Flask frameworks to create a web-based application.
- Implementation of barcode and QR code generation functionality.
- Successful implementation of login authentication system.
- Creation of a database to save and delete generated images.
- Addition of decoder functionality for the barcodes and QR codes.
- Implementation of small features like sharing pages with others, login/logout functionality, and taking feedback from users.

What did not:

- In the second week of the development process, we faced an issue in linking the client and server due to CORS policy. We fixed that by added flask-CORS package where we allowed to access the server from our client application.
- We were not able to implement the addition of style frames for the generated barcode and QR code images before downloading.
- The above functionality idea was not initially proposed however we thought to implement that by end of the project. But we couldn't complete this feature due to time constraint.

What has changed in the final version vs. the original project proposal:

- The final version of the project included several additional features like
    1. Login authentication.
    2. Database integration.
    3. Image saving and deletion.
    4. Sharing page and taking feedback from users.
    5. Decoder functionality.
- These features were not initially planned but were added to improve the functionality of the application.

What tools did you use in the development process for the project:

- **Visual Studio Code** and **PyCharm** IDEs for development
- **MongoDB Atlas** for database management
- **GCP Cloud Console** for deployment and hosting.
- **Angular CLI** for creating components, guards, services, and modules.
- **Docker Desktop** to test the containerized application by running all the images locally.
- **ChatGPT** for rectifying errors that we got during the development process of our application.
- **Google** for knowing the best framework that suits our requirement in the application.

DEMONSTRATION OF THE PROJECT IN ACTION

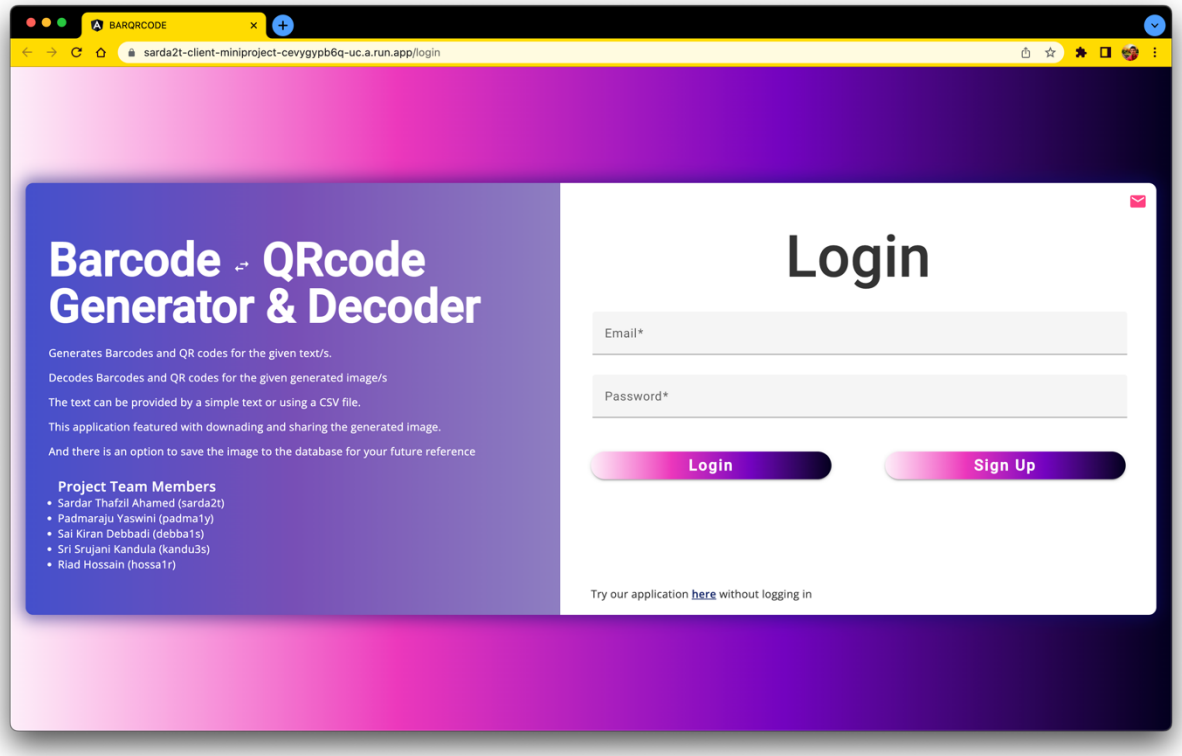
Short video demo:

A short video demo is uploaded in the OneDrive. Anyone with these links can access the video demo. ITC530 OneDrive link and ITC530 Google drive link

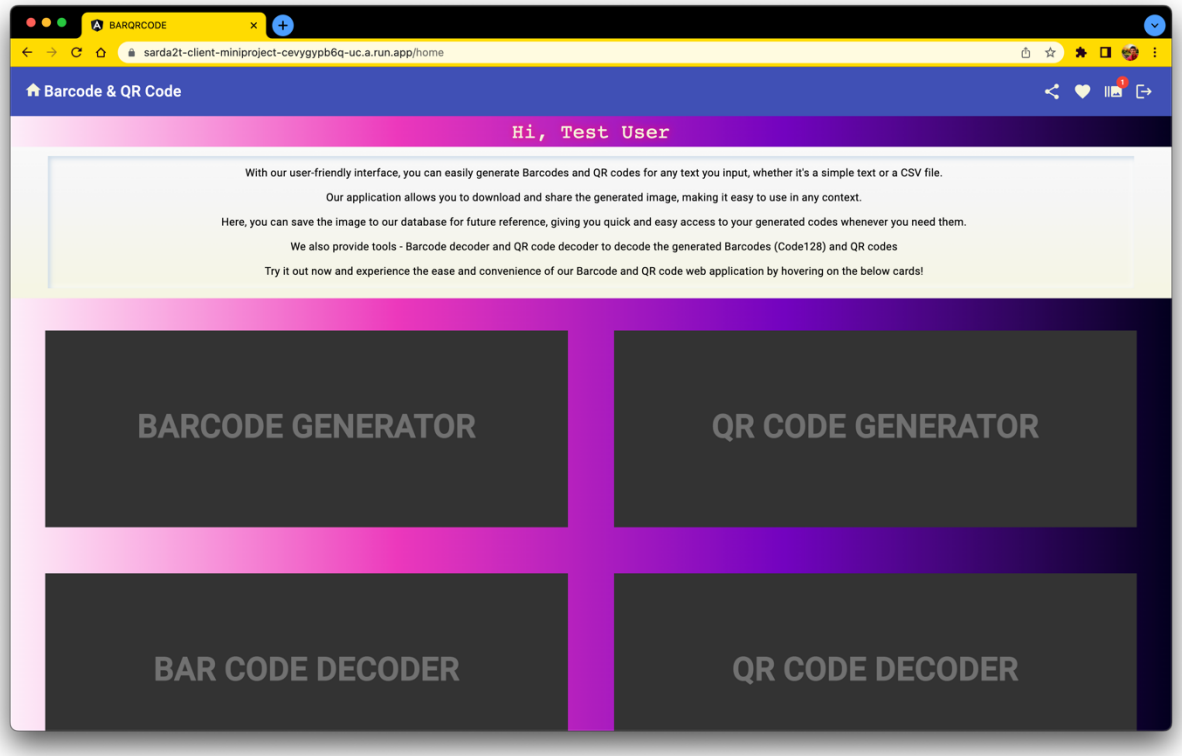
In the demo video, we have explained the application features.

Screenshots of our project in action:

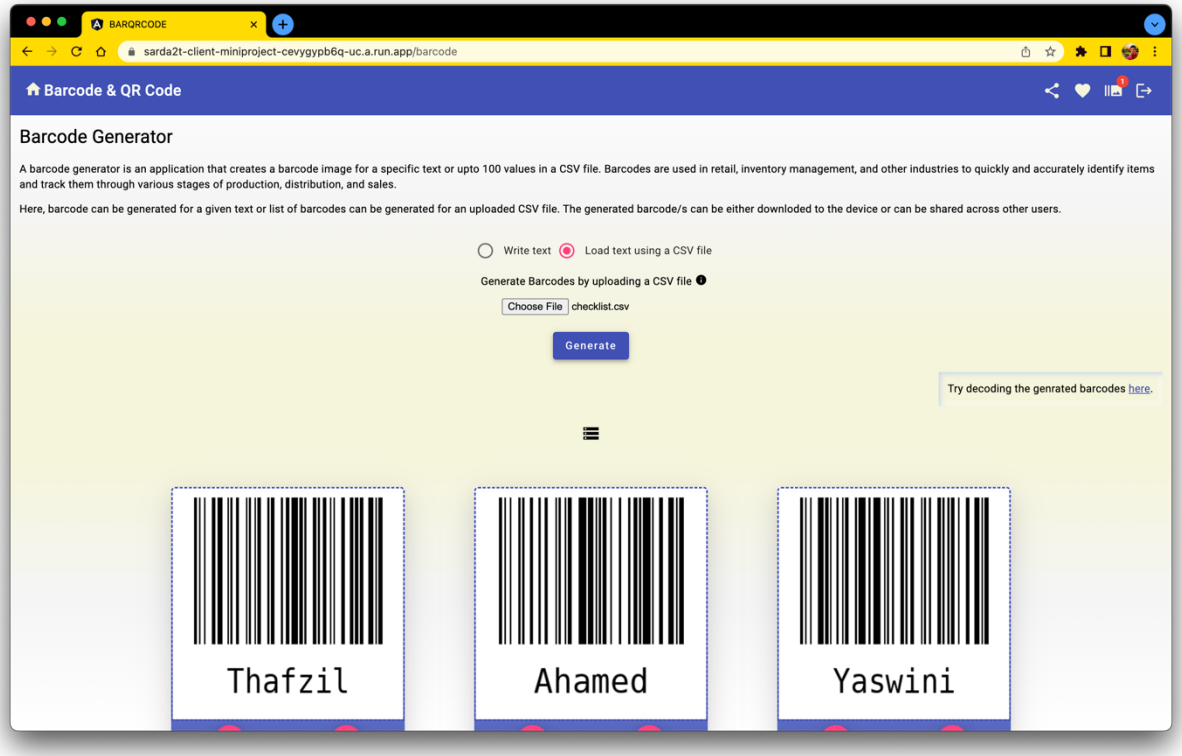
Login Page of our application where user can enter his login credentials or sign up into our application for free.



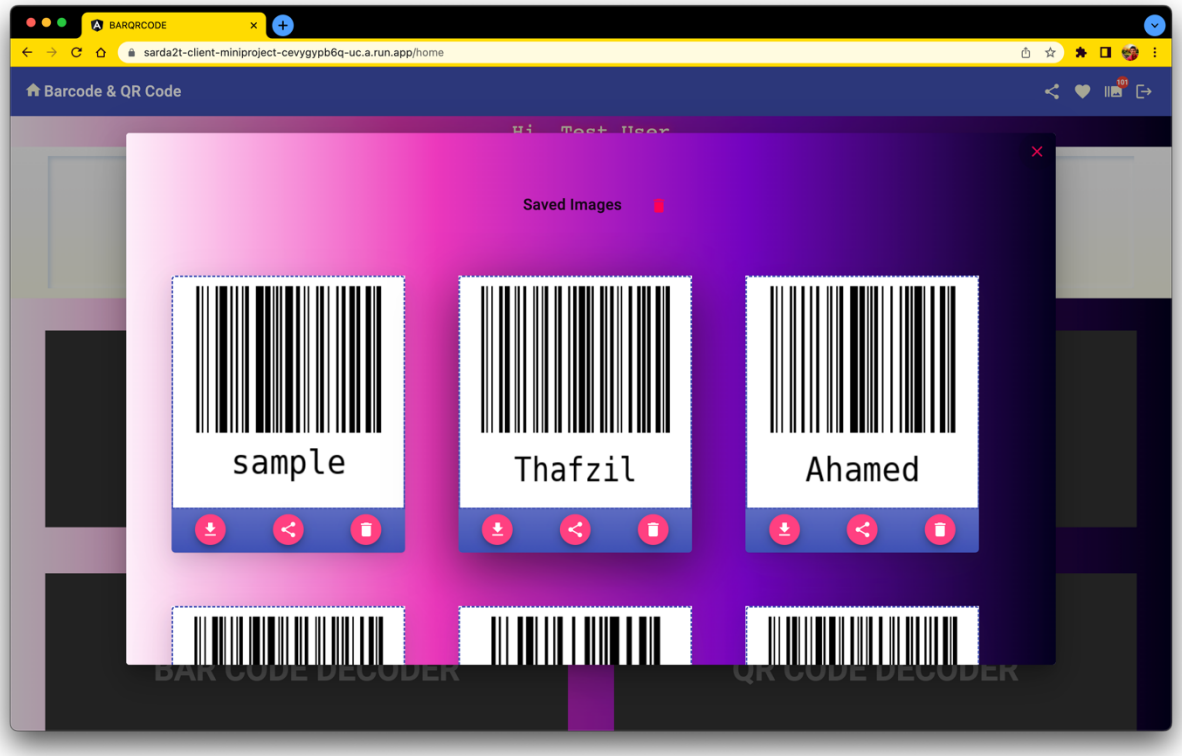
After successful login, the home page will be displayed as below.



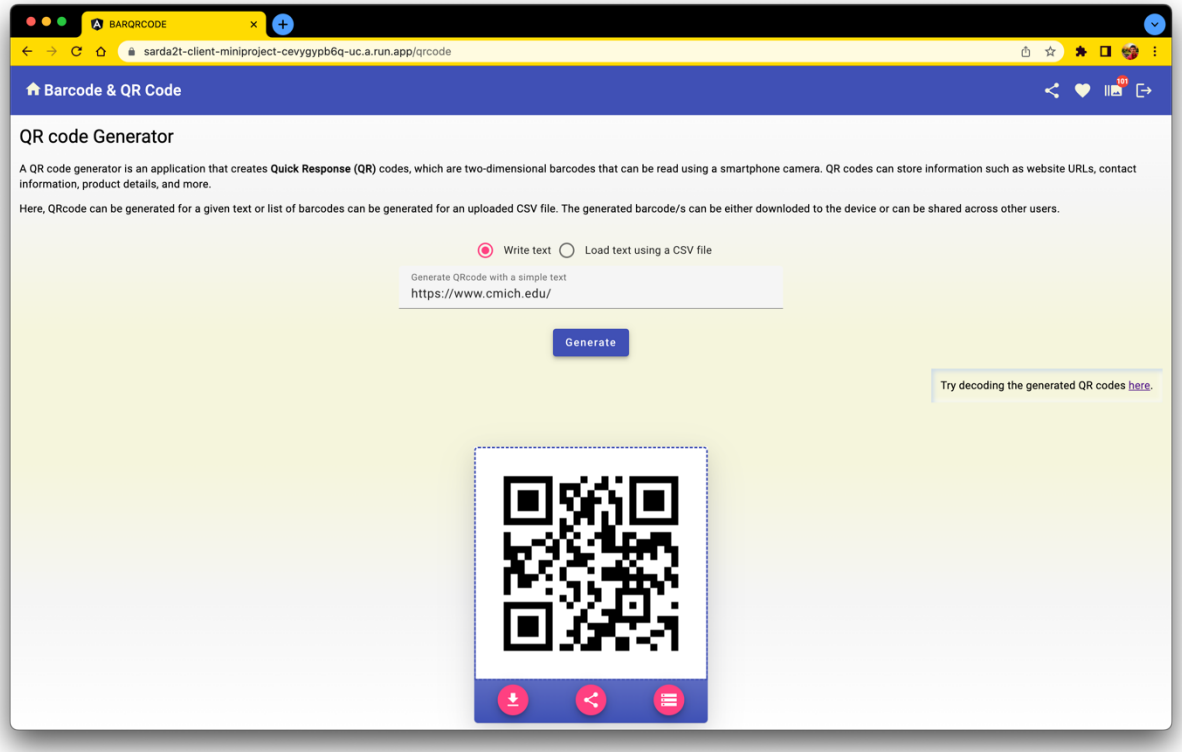
Generation of barcodes from a CSV file with more than 100 values



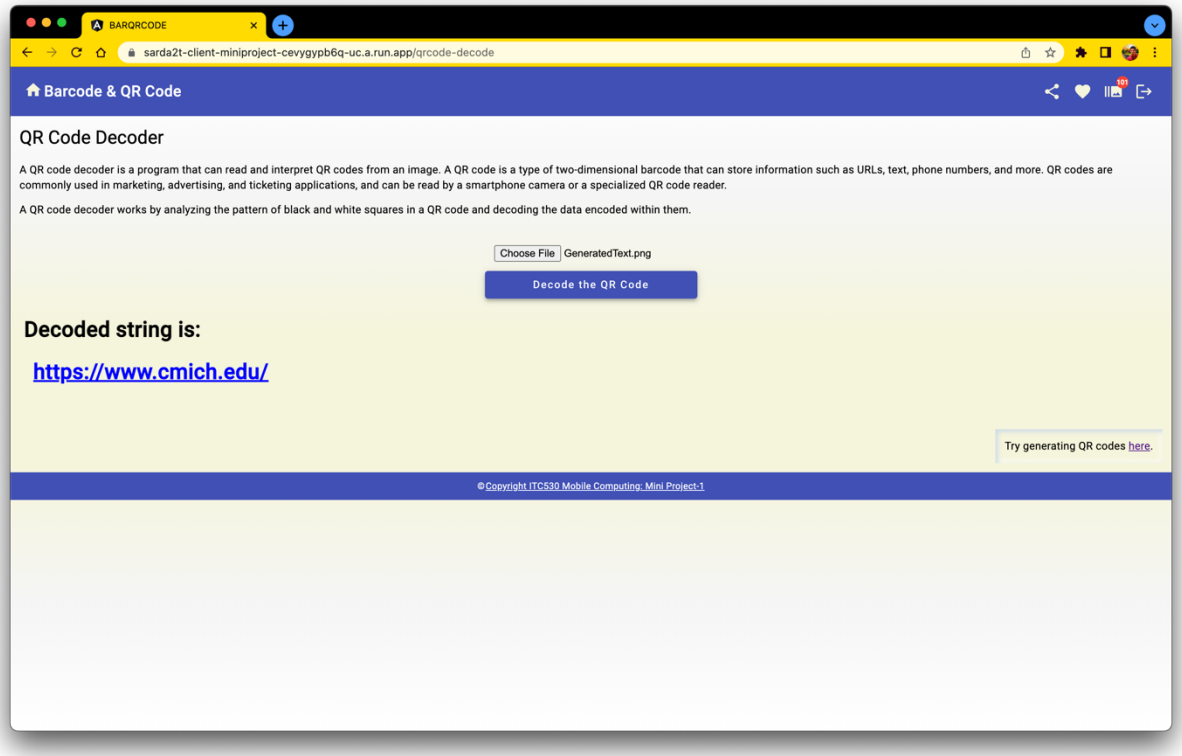
Displaying the saved images that were uploaded to the database.



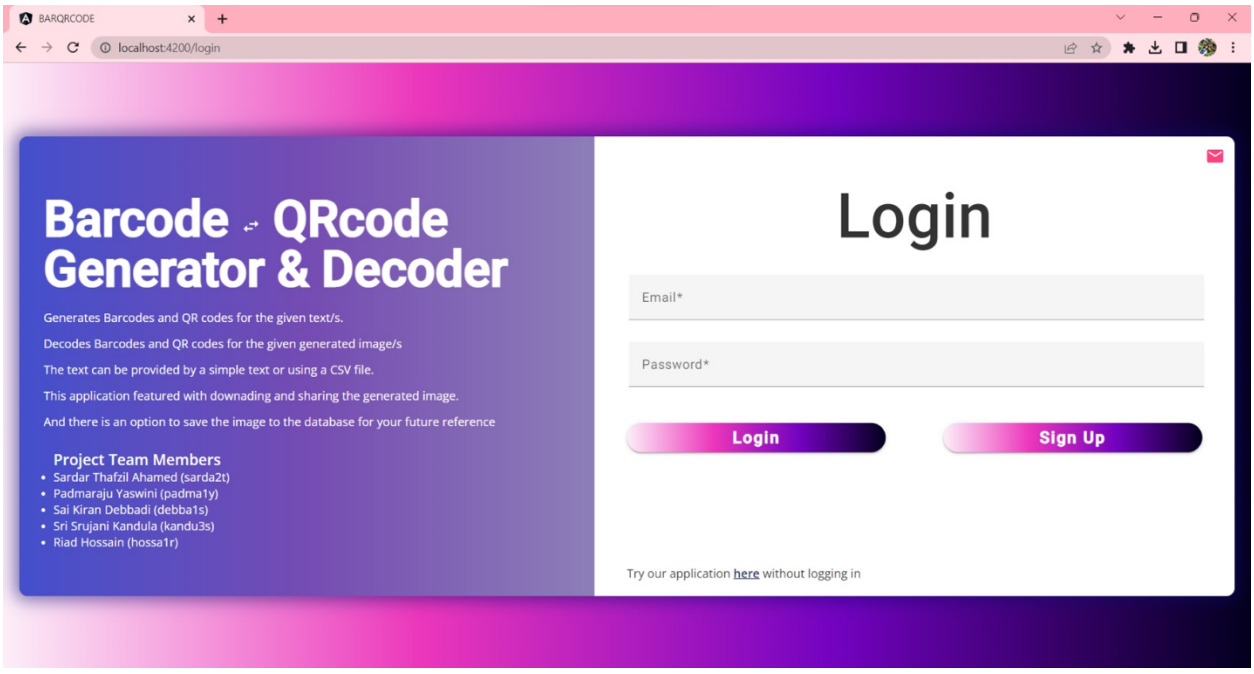
Creating QR code for the link https://www.cmich.edu



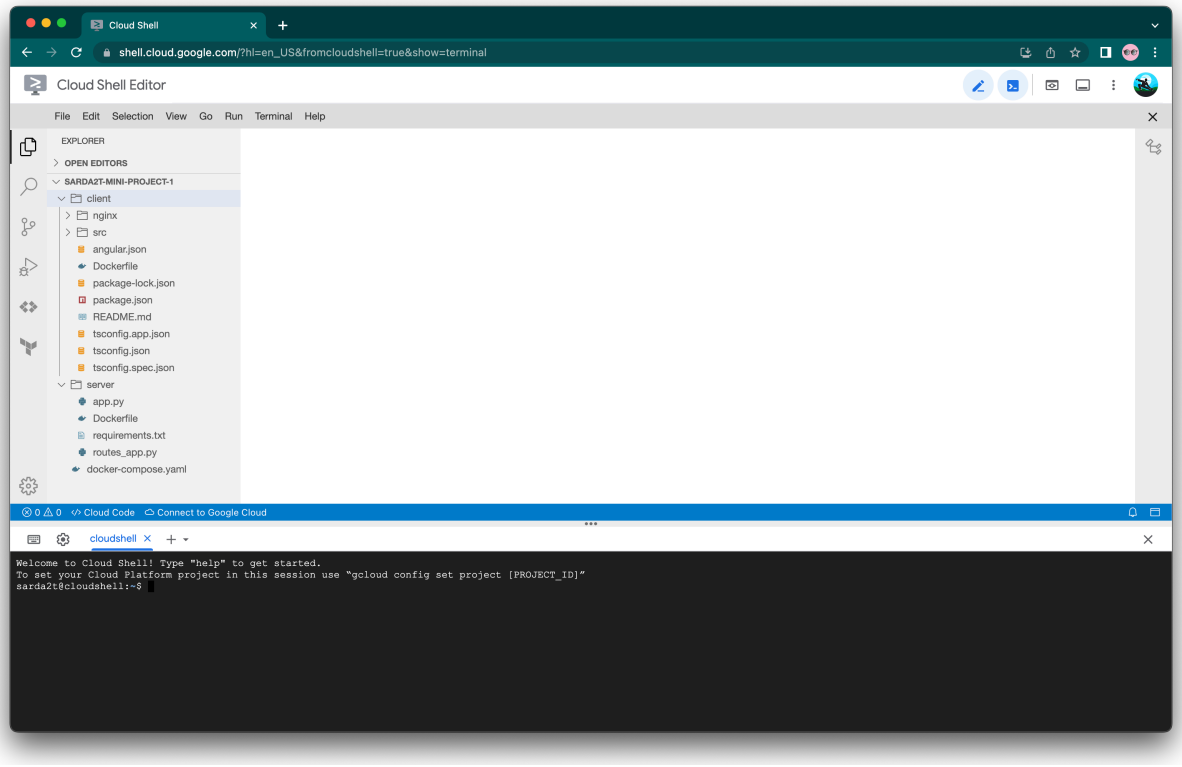
Decoding the QR code that is generated above.



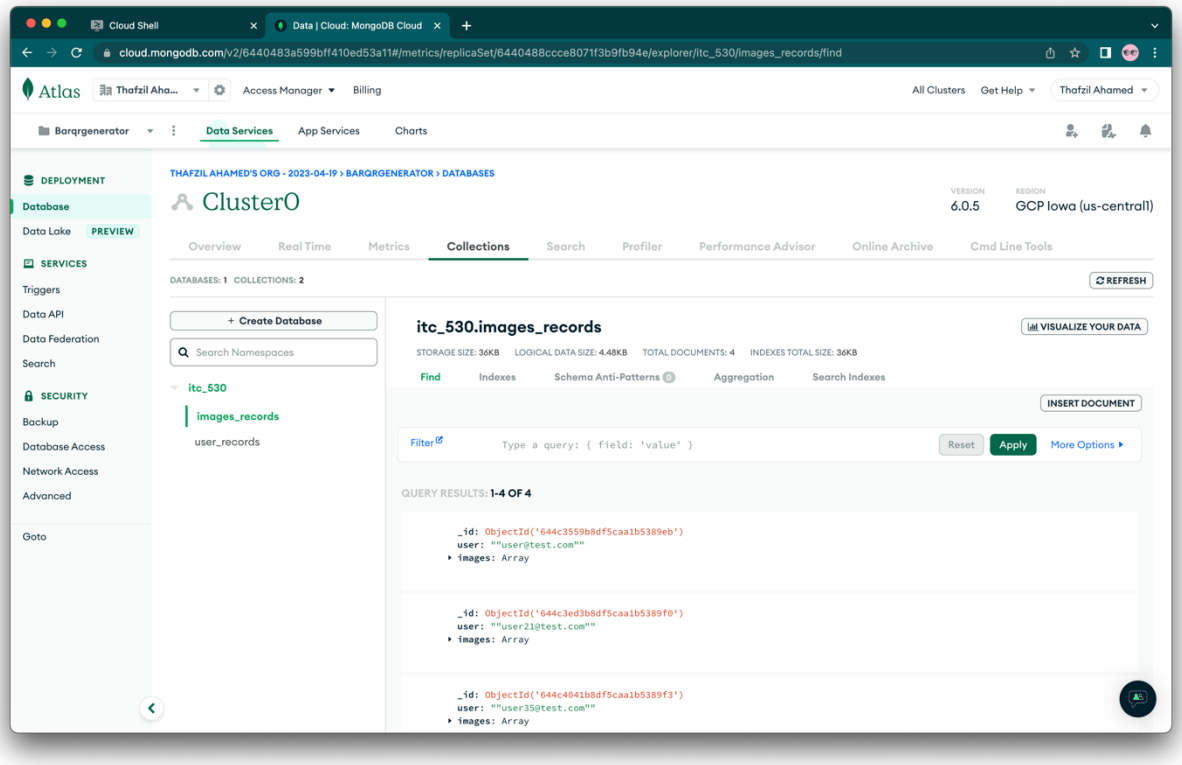
Snapshot of a local version of containerized application running on another machine by just running docker compose up. This machine is not having Angular, python and Mongo installed locally. Everything is containerized here and running on Docker using Docker Desktop.



Code structure for client and server. Each folder has separate Dockerfile with its own configuration.



Database deployment snapshot of our MongoDB database used in our project.



PROJECT FILES FOR REPRODUCIBILITY

Client-Side code is deployed in: https://sarda2t-client-miniproject-cevygypb6q-uc.a.run.app
Server-Side code is deployed in: https://sarda2t-server-miniproject-cevygypb6q-uc.a.run.app

All files are attached in the project files section and the configurations and step by step implementation details are displayed below.

Configuration details:

docker-compose.yaml file:
This Docker Compose YAML file is defining three services for a web-based application that includes a client, a server, and a MongoDB database. Here's a breakdown of the configuration details:

- The version of the Compose file is "0.1".
- The client service is built from the Dockerfile located in the ./client directory and maps port 4200 on the host to port 8080 inside the container.
- The server service is built from the Dockerfile located in the ./server directory and maps port 8083 on the host to port 8080 inside the container. It also has a dependency on the mongodb service.
- The mongodb service is using the latest version of the official MongoDB image available on Docker Hub and sets the hostname to "test_mongodb". It also sets some environment variables to initialize a new database with the name "itc_530" and a root user with the username "miniproject1" and password "pass".

Overall, this configuration creates a set of services that can work together to provide a functional web-based application. The client service will be accessible on port 4200 of the host, while the server service will be accessible on port 8083 of the host. The MongoDB service will be used by the server service to store and retrieve data.

Dockerfile for client**:**
This Dockerfile is used to configure the container for a web-based application that generates and decodes barcodes and QR codes.

- FROM node:lts-alpine3.16 as build: This sets the base image to use for building the container, which is node:lts-alpine3.16. This is a lightweight version of Node.js that includes the Long-Term Support (LTS) version of Node.js.
- WORKDIR /app: This sets the working directory inside the container to /app.
- COPY package*.json ./: This copies the package.json and package-lock.json files from the local file system to the /app directory inside the container.
- RUN npm install: This runs the npm install command inside the container to install all the dependencies required for the application.
- COPY . .: This copies all the files from the local file system to the /app directory inside the container.
- RUN npm run build --prod: This runs the npm run build --prod command inside the container to build the production version of the Angular application.

- FROM nginx:1.19: This sets the base image to use for the final container, which is nginx:1.19. This is a lightweight version of the Nginx web server.
- COPY ./nginx/nginx.conf /etc/nginx/nginx.conf: This copies the nginx.conf file from the local file system to the /etc/nginx directory inside the container.
- COPY --from=build /app/dist/barqrcode/ /usr/share/nginx/html: This copies the built application from the build stage to the /usr/share/nginx/html directory inside the final container.
- EXPOSE 8080: This exposes the port 8080 to allow incoming connections to the container.

Overall, this Dockerfile sets up a container for a web-based application built with Angular. It installs all the required dependencies and builds the application in the first stage, and then copies the built application to the Nginx web server in the second stage.

Dockerfile for Server
This Dockerfile describes the configuration and setup for a containerized Python application. Here are the configuration details:

- The Dockerfile starts with a base image of Python version 3.9 that is based on the slim version of the Debian Bullseye distribution.
- The working directory is set to the root directory of the container.
- The requirements.txt file is copied from the local directory to the /tmp directory inside the container.
- The pip package manager is upgraded to the latest version.
- The packages listed in the requirements.txt file is installed using pip. The --no-cache-dir option is used to prevent caching of downloaded packages.
- The libopencv-dev and libzbar-dev packages are installed using the apt-get package manager. These packages are required for the barcode and QR code generation and decoding functionality of the application.
- The OpenCV and pyzbar packages are installed separately using pip for the decoding part of the project.
- The app.py and routes_app.py files are copied from the local directory to the root directory of the container.
- Port 8080 is exposed for communication with the container.
- The entrypoint command is set to start the Gunicorn web server to run the app on port 8080.

Overall, this Dockerfile sets up a containerized Python application that uses various packages for barcode and QR code generation and decoding. It also installs the necessary dependencies for OpenCV and pyzbar packages and runs the application using Gunicorn web server.

STEP-BY-STEP instructions on how to configure the components while implementing.

To run the web application completely using the local docker containerized applications, you need to follow below steps.

1. Extract the project files.

2. The lines at the below mentioned files need to be commented and uncommented. This will avoid remote server calls and enable local container calls and enable local containerized database to come in action instead of original MongoDB Atlas.
3. **Uncomment these lines: Line 9** in /client/src/app/app.service.ts, **Line 65** in /server/routes_app.py
4. **Comment these lines: Line 6 in** /client/src/app/app.service.ts, Line **62** in /server/routes_app.py
5. Enter the command "docker compose up" in the main project folder where you can see the docker-compose.yaml file.
6. Observe the output of the above command and once the composing is done, you can access the docker containerized application by navigating to http://localhost:4200

To run this application by uploading to Google cloud platform, you need to follow these steps.

1. Extract the project files.
2. Go to https://console.cloud.google.com/
3. Create a new Project.
4. Create a folder in the Artifact Registry. This is the place where you will be pushing all your built images.
5. Open cloud shell and create a new folder to upload all the project files. You can upload a file by right click on folder and clicking upload files or by drag and drop the files.
6. By changing directory to server enter the below commands to push the image to the registry
7. docker build -t <artifact_registry_path>/image_name_with_version . for building the image.
8. docker push <artifact_registry_path>/image_name_with_version> for pushing image to artifact registry.
9. Once images are pushed to registry, you can use cloud run to create a service for each image and deploy them.
10. Open cloud run, create a service, and select the server image that is pushed to the registry one image per service. Leave the port configuration to 8080.
11. Once the image is deployed to the GCP, copy the deployed service URL, and paste it in the below file at specified line.
12. In line 6 of /client/src/app/app.service.ts, replace the **remote_api_url** with the URL that is generated above.
13. Now, do step 7 and step 8 by changing directory to client to push the client image to the repository.
14. Once image is pushed, open cloud run, again create a service, and select the client image that is pushed to the registry. Leave the port configuration to 8080.
15. After deploying to the cloud run, the URL will be generated where you can access the application.
16. For the database deployment, we can either use MongoDB Atlas by creating a free tier account or creating a containerized version of MongoDB in local.
17. Configuration details of the database will be available in this file /server/routes_app.py.

Features of our application

User registration: The application allows users to register for an account with a valid email address and password.
User authentication: The application includes a user authentication feature that requires users to log in with a username and password to access the functionality. In addition, there is an option on the login page to explore the application without logging in as a test user.
User-friendly interface: The application provides a simple and intuitive interface for users to easily navigate and use the application.
Barcode and QR code generation: Users can easily generate various types of barcodes and QR codes by entering the required information such as text, CSV file.
Barcode and QR code decoding: The application allows users to scan and decode barcodes and QR codes for the given Barcode and QR code.
Database integration: The application integrates with a Mongo DB database to store and retrieve generated barcodes and QR codes.
Sharing the page: The application includes a share button that allows users to share the generated barcode or QR code page via email or social media platforms such as Facebook, Twitter, or LinkedIn.
Saving images to a database: The application allows users to save generated barcodes and QR codes as images to a database for future retrieval and use.
User logout: The application includes a logout button that allows users to securely log out of the application.
User privacy: The application is designed with user privacy in mind, ensuring that user information is not shared or used without their explicit consent.
Error handling: The application includes error handling mechanisms to prevent user errors and provide helpful feedback.
Cross-platform compatibility: The application can be accessed from any device with a web browser, including desktops, laptops, tablets, and smartphones.
Docker containerization: The application is containerized using Docker, which allows for easy deployment and scaling.
Cloud deployment: The application is deployed on the Google Cloud Platform (GCP), which provides high availability, scalability, and reliability.

Project Team Members

- Sardar Thafzil Ahamed (sarda2t)
- Padmaraju Yaswini (padma1y)
- Sai Kiran Debbadi (debba1s)
- Sri Srujani Kandula (kandu3s)
- Riad Hossain (hossa1r)