AI-POWERED CONTACT CARD SCANNER WITH QR CODE GENERATOR

Ahmed Aseel

BACHELOR OF COMPUTER SCIENCE

UNIVERSITY OF THE WEST OF ENGLAND

ABSTRACT

This project presents the development of an AI-powered application designed to scan business cards, accurately extract contact information, and generate a QR code for easy import into digital contact lists. Leveraging LayoutLMv2, a cutting-edge model fine-tuned on a custom dataset of business cards, the system surpasses traditional Optical Character Recognition (OCR) methods in both accuracy and efficiency. The frontend of the application is built using Vue.js, while the backend is powered by Flask, enabling seamless interaction between the user interface and the AI model hosted on Azure. Key features include the automated generation of QR codes containing structured contact information, facilitating instant integration into smartphone contact lists. The project's effectiveness was evaluated through rigorous testing across diverse business card designs, demonstrating its robust performance in real-world scenarios. This work highlights the potential of AI and machine learning in streamlining the digitization of physical information, offering a significant improvement over existing solutions.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my project supervisor, Ms. Udhuma Abdul Latheef, for her invaluable guidance, support, and encouragement throughout the development of this project. Her expertise and insights have been instrumental in shaping the direction and success of this work.

I am also immensely thankful to my university, Villa College, for providing the resources and environment necessary for the completion of this project. The knowledge and skills I have gained during my time here have been crucial to the development of this application.

Finally, I would like to extend my heartfelt thanks to my family and friends, especially those in my course, for their unwavering support, patience, and encouragement. Their belief in my abilities kept me motivated, and their advice and feedback were essential to overcoming the challenges I faced during this journey.

DECLARATION

I hereby declare that my own work is the outcome of the project described in this study. The work was completed and produced specifically for the final year project.

Student Name	: Ahmed Aseel
Student ID	: S2200334
Email Address	: S220034@villacollege.students.edu.mv
UWE ID	: 22023484
Program	: BSc (Hons) Computer Science
Model Name	: Digital Systems Project
Module Code	: UFCFXK-30-3
Campus	: Villa College, Maldives
Affiliated University	: UWE Bristol, the University of the West of England

TABLE OF CONTENT

Contents

TABLE OF FIGURES

INTRODUCTION

Keeping track of and arranging contact details effectively is a big difficulty in today's increasingly digital environment, especially when working with paper business cards. Traditional methods of manually entering contact details into digital devices are time-consuming and prone to errors. This project addresses this real-world problem by developing an AI-powered application designed to scan business cards, accurately extract contact information, and generate a QR code for seamless integration into digital contact lists.

Problem Statement and Importance

Digitizing traditional business cards presents challenges related to data extraction accuracy and dependability in addition to convenience. The many styles and typefaces found on business cards present a challenge for current optical character recognition (OCR) systems, which frequently results in inaccurate capture of important contact information. In order to ensure that customers can rely on the system to digitize contacts effectively, this research aims to address these constraints by enhancing the accuracy of data extraction through the application of cutting-edge machine learning techniques, notably the LayoutLMv2 multi-model.

Scope

This project's scope includes all steps involved in scanning, extracting, classifying, and digitizing contact details from actual business cards. It entails building a frontend for user interaction, a backend to process the collected data, and a machine learning model that can handle a variety of card designs. The project also includes creating QR codes that, when read by smartphones, instantly add the contact information that has been extracted to the user's contact list. The project is meant to be adaptable enough to function with a variety of business card designs and layouts, guaranteeing its use in practical situations.

Objectives

The specific objectives of the project are as follows:

- **Objective 1:** Develop an AI model, by fine tuning the multi model layoutLMV2, that accurately recognizes and extracts text (including names, phone numbers, emails, etc.) from images of business cards.
- **Objective 2:** Categorize the extracted information into structured contact details using natural language processing techniques.

- **Objective 3:** Generate a QR code containing the structured contact information, allowing for easy scanning and importing into a smartphone's contact list.
- **Objective 4:** Evaluate the system's performance in terms of accuracy, efficiency, and usability across a diverse set of business card designs.

Research Questions

The key research questions guiding this project are:

1. How can AI be leveraged to improve the accuracy of text recognition and extraction from business cards?
2. What natural language processing techniques can be used to categorize and structure contact information effectively?
3. How can QR codes be utilized to facilitate the seamless transfer of contact details from business cards to smartphones?
4. What are the performance metrics of the developed system in terms of accuracy, efficiency, and usability?

Literature Review Summary

A review of the existing literature reveals that while traditional OCR methods are commonly used for text extraction, they are often limited by the variability in business card designs. Advanced multi-models like LayoutLMv2, which incorporate layout-aware techniques , have shown promise in extracting text from complex documents with higher accuracy. Additionally, natural language processing (NLP) methods are increasingly being used to categorize and structure extracted data effectively. The literature also highlights the growing use of QR codes for data transfer, particularly in contactless environments, making them an ideal solution for this project.

Approach

To address the problem, the project employs the LayoutLMv2 model, fine-tuned on a custom-labeled dataset of business cards. The system was developed using a combination of technologies: the backend was built using Flask, the frontend using Vue.js, and the model was deployed on Azure for scalability. The system extracts contact information from scanned business cards, categorizes the data using NLP techniques, and generates a QR code that can be easily scanned to import the contact details into a smartphone.

Outcome

The final product is an AI-powered application that streamlines the process of digitizing business cards with enhanced accuracy. The system has been tested on various business card designs, demonstrating robust performance in extracting and structuring contact information, and effectively generating QR codes for seamless data transfer.

LITERATURE REVIEW

The integration of AI-powered text recognition, natural language processing (NLP), and QR code generation represents a significant advancement in the digitization and automation of information processing. This literature review offers a comprehensive examination of the current state of these technologies, with a specific emphasis on their application in extracting and organizing information from business cards, as well as generating QR codes for easy data sharing. The presentation integrates the latest research findings, including the use of advanced models like LayoutLMv2, and explores the broader implications of these technologies.

AI-Powered Text Recognition

The use of machine learning and artificial intelligence has significantly improved text recognition systems. Traditional Optical Character Recognition (OCR) methods, while often used, can lack accuracy, especially when dealing with complex or atypical designs commonly found on business cards. The research conducted by Xu et al. (2020) demonstrates that LayoutLMv2 represents a notable progression in this particular domain. This model employs a multi-modal approach that combines text, layout, and image data, allowing it to understand and process visually intricate texts more effectively than traditional OCR methods. Xu et al. (2020) found that the model's ability to preserve the structure and visual arrangement of documents results in a significant enhancement in accuracy for text extraction tasks.

Recent research on artificial intelligence (AI) for text recognition emphasizes the importance of training models on diverse datasets to improve performance on different types of texts. In their study, Hussain et al. (2022) developed the PHTI (Pashto Handwritten Text Imagebase), a comprehensive dataset aimed at improving OCR systems, particularly for low-resource languages like Pashto. The work highlights the importance of tailored datasets in enhancing the accuracy of AI models in detecting text across different languages and scripts (Hussain et al., 2022).

Natural Language Processing (NLP)

Natural Language Processing (NLP) is the field of study that focuses on the interaction between computers and human language. It involves developing algorithms and models that enable computers to understand, interpret, and generate human language in a way that is similar to how humans do.
Natural Language Processing (NLP) is crucial for categorizing and structuring gathered material into meaningful and cohesive information. Natural Language Processing (NLP) technologies have evolved from basic text processing techniques to sophisticated models that can handle context, emotions, and complex questions. Transformer-based models have made substantial advancements in enhancing the capabilities of natural language processing (NLP) systems. For instance, researchers have extensively studied the integration of Natural Language Processing (NLP) with AI-assisted systems across various fields, including code generation and text classification (Wong et al., 2023; Kaur & Kaur, 2023). These systems utilize their ability to understand and generate human language, allowing for more accurate extraction and categorization of data (Wong et al., 2023), (Kaur & Kaur, 2023).

In addition, significant advancements have been made in the application of Natural Language Processing (NLP) to extract structured data from unorganized text. Dande and Pund (2023) conducted an assessment that specifically examined the most recent applications of natural language processing (NLP), with a special emphasis on tasks such as named entity recognition (NER) and sentiment analysis. These actions are essential for managing the obtained business card data. These technologies enhance the accuracy and organization of contact information, hence increasing the efficiency of the digitization process (Dande & Pund, 2023).

QR Code Generation

QR codes are now a crucial element of modern information systems, acting as a bridge between the physical and digital worlds. The generation of QR codes for the storage and transfer of contact information not only offers convenience but also enhances the accessibility and use of digital content. In their study, Tambe et al. (2023) investigated the integration of artificial intelligence (AI) in the development of secure and efficient QR codes. The study emphasizes the need of maintaining data integrity during the process of encoding QR codes. This is particularly

important for applications that place emphasis on accuracy and security, such as in the conversion of business cards into digital format (Tambe et al., 2023).
Artificial intelligence is employed in the manufacturing of QR codes to enhance their functionality beyond simple data storage. The integration of artificial intelligence (AI) with QR code technology allows for immediate updates and interactions, hence improving the level of engagement and user-friendliness of the whole experience. The dynamic updating of QR codes is particularly beneficial in corporate environments where contact information may frequently change, as it ensures that the information stays current and relevant.

Integration of Technologies in AI-Powered Contact Card Scanners

The incorporation of AI-powered text recognition, natural language processing (NLP), and QR code generation into a unified system represents a significant advancement in the automation of business processes. The integration is significantly dependent on the LayoutLMv2 model, which acts as a robust solution for extracting structured data from complex and visually varied documents. This system utilizes advanced NLP techniques to accurately categorize and arrange contact details, which are then converted into a QR code for easy sharing and storage.
This comprehensive approach not only streamlines the process of transforming business cards into digital format but also sets a higher standard for accuracy and efficiency in managing information. Using advanced AI models ensures accurate processing of complex and atypical business cards, leading to a significant decrease in the manual work often required by traditional methods.

LayoutLMv2: Multi-Modal

Approach

LayoutLMv2 introduces a multi-modal pre-training approach that simultaneously processes text, layout, and image information, a significant advancement over previous models that primarily focused on text or visual information in isolation. The key innovation of LayoutLMv2 lies in its ability to model the interaction between different modalities during the pre-training phase. This multi-modal framework allows the model to better understand the intricate relationships between the visual layout of documents and their textual content (Xu et al., 2020).

Model Architecture

The architecture of LayoutLMv2 is built on the Transformer framework but with significant enhancements to handle the additional modalities. It employs a two-stream multi-modal Transformer encoder, which processes text and visual features in parallel. This architecture allows the model to capture the nuanced interactions between textual and visual elements in documents, a crucial feature for understanding documents where layout and formatting carry significant meaning. The integration of a spatial-aware self-attention mechanism is particularly notable, as it enables the model to fully leverage the spatial relationships between different text blocks within a document (Xu et al., 2020).

Text, Visual, and Layout Embeddings

LayoutLMv2 uses a sophisticated embedding scheme to encode different aspects of the document:

1. **Text Embedding**: Similar to traditional NLP models, LayoutLMv2 uses embeddings derived from tokenized text. However, these embeddings are enriched with information from other modalities, ensuring that the textual data is interpreted in the context of its visual and spatial characteristics.
2. **Visual Embedding**: The model incorporates visual embeddings using features extracted by a pre-trained ResNet model. These embeddings capture the visual attributes of the document, such as the font style, size, and color of the text, as well as any images or logos present. This allows the model to take into account the visual presentation of the content, which can be crucial for interpreting documents where the appearance of the text influences its meaning (Xu et al., 2020).
3. **Layout Embedding**: LayoutLMv2 introduces layout embeddings that encode the spatial positions of text blocks within the document. These embeddings represent the 2D coordinates of the text blocks, as well as their bounding box dimensions. This spatial information is critical for understanding documents where the layout itself conveys meaning, such as in forms or business cards where the position of a text block (e.g., a name or address) is indicative of its content (Xu et al., 2020).

Pre-training Tasks: Text-Image Matching and Alignment

The pre-training process of LayoutLMv2 includes several innovative tasks designed to improve the model's understanding of the relationship between text and images:

1. **Masked Visual-Language Modeling (MVLM)**: This task extends the masked language modeling approach used in models like BERT by incorporating visual information. In LayoutLMv2, the model is trained to predict masked tokens not just based on their textual context, but also considering the associated visual and layout information. This approach allows the model to learn richer representations that consider the full context of each token within the document (Xu et al., 2020).
2. **Text-Image Alignment (TIA)**: This task requires the model to align textual content with its corresponding visual features. By training the model to match text with its associated visual representation, LayoutLMv2 becomes more adept at understanding how text and images relate to each other within a document. This task is particularly useful for documents that include complex visual layouts, where understanding the relationship between text and images is essential for accurate interpretation (Xu et al., 2020).
3. **Text-Image Matching (TIM)**: TIM further enhances the model's ability to understand multi-modal documents by training it to determine whether a given piece of text matches the visual content it is paired with. This task helps the model develop a deeper understanding of how text and visual elements interact, improving its performance on tasks that require a nuanced interpretation of both modalities (Xu et al., 2020).

Conclusion

The advancements in AI-powered text recognition, NLP, and QR code generation are transforming the way we digitize and manage contact information. The use of models like LayoutLMv2 represents a significant leap forward in accuracy and efficiency, enabling the seamless integration of these technologies into practical applications like contact card scanners. As AI continues to evolve, we can expect further improvements in these systems, leading to even more efficient and user-friendly solutions for managing digital information.

REQUIRMENTS

Functional Requirements

Functional requirements specify what a system must do. These requirements define the specific functions, tasks, or behaviors that the system must perform to meet the user’s needs. They detail the system's intended features and capabilities, providing a clear description of what the system is expected to accomplish.

For example, in the context of software development, functional requirements might include capabilities such as user authentication, data processing, or the ability to generate reports. These requirements focus on the user interactions with the system and the operations that the system must support to fulfill its purpose.

According to Kumar et al. (2022), functional requirements outline the operations and actions that a system must be able to perform to achieve its intended functionality. They are critical for ensuring that the system behaves as expected and delivers the desired outcomes (Kumar et al., 2022).

Non-Functional Requirements

Non-functional requirements (NFRs), on the other hand, describe how the system performs its functions rather than what functions it performs. These requirements define the quality attributes, performance criteria, and operational constraints that the system must adhere to. Non-functional requirements are essential for ensuring the system's reliability, usability, security, and efficiency.

The requirements are derived from the project's objectives and are designed to ensure that the system meets both user needs and technical standards. Below are the key functional and non-functional requirements for this project.

Functional Requirements

1. **Business Card Scanning**
    - The system must be capable of capturing high-quality images of business cards through a mobile device's camera or by uploading an image file.
2. **Text Extraction**

- The system must accurately extract text from the scanned business cards, including names, phone numbers, email addresses, company names, and addresses.

3. **Text Categorization**
   - The system must categorize extracted text into predefined fields (e.g., Name, Email, Phone Number, Company) using Natural Language Processing (NLP) techniques.
4. **QR Code Generation**
   - The system must generate a QR code that contains the categorized contact information in a vCard format, suitable for importing into a smartphone's contact list.
5. **QR Code Scanning**
   - The system must allow users to scan the generated QR code with any standard QR code reader, leading to an interface for adding the contact to their device's contact list.
6. **Error Handling**
   - The system must handle errors in text extraction (e.g., due to poor image quality) and provide users with options to manually correct the extracted information.
7. **Data Storage**
   - The system must securely store scanned business cards and extracted contact information for future retrieval and editing.
8. **User Interface (UI)**
   - The system must provide an intuitive and user-friendly interface for uploading business cards, viewing extracted information, and generating QR codes.
9. **API Integration**
   - The system must integrate with external APIs, such as Azure's AI services, to perform tasks like image processing and text extraction.
10. **Multi-language Support**

- The system must support multiple languages for text extraction and categorization, ensuring it can process business cards from different regions.

Non-Functional Requirements

1. **Performance**
    - The system must process and generate QR codes within 5 seconds of uploading a business card image, ensuring a fast user experience.
2. **Scalability**
    - The system must be scalable to handle multiple users simultaneously, with no degradation in performance.
3. **Accuracy**
    - The system must achieve at least 95% accuracy in text extraction from business cards, with consistent performance across various designs and layouts.
4. **Security**
    - The system must ensure the security of stored contact information, implementing encryption and secure access controls.
5. **Reliability**
    - The system must have an uptime of 99.9%, ensuring that it is available and operational at all times.
6. **Compatibility**
    - The system must be compatible with all major operating systems and devices, including iOS, Android, Windows, and macOS.
7. **Maintainability**
    - The system's codebase must be modular and well-documented to facilitate easy updates and maintenance.
8. **Usability**
    - The system must have an intuitive interface that requires minimal training for end users, with a user satisfaction rating of 90% or higher.

9. **Compliance**
    - The system must comply with relevant data protection regulations, such as GDPR, ensuring that user data is handled lawfully.
10. **Portability**
    - The system must be deployable on multiple cloud platforms (e.g., Azure, AWS) without requiring significant reconfiguration.

Requirement Elicitation Process

Requirement elicitation is the initial step in gathering the necessary information to develop a system that meets the needs of its users. This process is crucial as it ensures that all stakeholders' needs are understood and documented before the design and development phases begin. The process of elicitation involves several steps:

1. **Identifying Stakeholders:** The first step is to identify all the stakeholders who will be affected by the system. Stakeholders include users, clients, developers, managers, and others who have an interest in the system's success (Chung & Leite, 2009).
2. **Gathering Requirements:** Once stakeholders are identified, the next step is to gather their requirements. This can be done using various techniques such as interviews, surveys, observation, and workshops. The goal is to understand the needs, expectations, and constraints of the stakeholders (Ullah et al., 2011).
3. **Clarification and Refinement:** After gathering the initial requirements, the next step is to clarify and refine them. This involves resolving ambiguities, filling in gaps, and ensuring that all requirements are understood and agreed upon by the stakeholders. This step often requires iterative communication and feedback sessions (Devendra Kumar et al., 2022).
4. **Documentation:** The finalized requirements are then documented in a clear and structured manner. This documentation serves as a reference for the design and development phases and is often captured in a Software Requirements Specification (SRS) document (Gonzalez-Huerta et al., 2012).

5. **Validation:** Finally, the documented requirements are validated by the stakeholders to ensure that they accurately reflect their needs and expectations. Validation helps to prevent costly changes and misunderstandings later in the project lifecycle (Ullah et al., 2011).

Methods of Requirement Analysis

Once the requirements have been elicited, the next step is to analyze them. Requirement analysis involves examining, structuring, and prioritizing the elicited requirements to ensure they are clear, complete, and feasible.

1. **Prioritization:** Not all requirements have the same level of importance or urgency. Prioritization involves ranking the requirements based on their significance to the project's goals, resource constraints, and stakeholder needs. Techniques such as the MoSCoW method (Must have, Should have, Could have, and Won't have) are commonly used for this purpose (Mairiza et al., 2010).
2. **Feasibility Analysis:** This step involves evaluating whether the requirements can be realistically implemented within the constraints of the project, such as time, budget, and technology. Feasibility analysis helps in identifying potential challenges early in the development process (Khan et al., 2016).
3. **Modeling:** Requirement modeling involves creating visual representations of the requirements, such as use case diagrams, data flow diagrams, or entity-relationship diagrams. These models help in understanding the relationships between different requirements and in communicating them more effectively to the development team (Nguyen, 2009).
4. **Conflict Resolution:** During the analysis, conflicting requirements from different stakeholders may arise. Conflict resolution techniques, such as negotiation and prioritization, are used to address these conflicts and ensure that the final set of requirements is consistent and agreed upon by all parties involved (Mairiza et al., 2010).
5. **Specification:** The analyzed requirements are then specified in detail, breaking them down into smaller, actionable components. This detailed specification guides the development team during the design and implementation phases (Gonzalez-Huerta et al., 2012).

6. **Verification:** Finally, the requirements are verified to ensure they are complete, consistent, and unambiguous. Verification often involves reviewing the requirements with stakeholders and conducting tests to confirm that they meet the intended objectives (Glinz, 2007).

METHODOLOGY

For the development of the AI-Powered Contact Card Scanner with QR Code Generator I choose the **Agile methodology**. Agile is a flexible and iterative approach to software development that emphasizes collaboration, continuous feedback, and adaptability to change. Given that this project was conducted by a single developer, Agile provided the necessary framework to manage the complexities of integrating AI, front-end, and back-end technologies, while allowing for iterative development and refinement.

Agile Methodology

Agile methodology breaks down the software development process into small, manageable iterations or "sprints." Each sprint typically lasts a few weeks and results in a functional increment of the software. This approach allows for continuous testing, feedback, and improvement throughout the development process (Sommerville, 2016).

Key aspects of the Agile methodology include:

- **Iterative Development:** The project is developed in small increments, allowing for regular adjustments based on feedback and testing.
- **Flexibility:** Agile is designed to accommodate changes in requirements, even late in the development process, making it ideal for projects where the end goals may evolve.
- **Continuous Feedback:** Regular communication and feedback from stakeholders (or in this case, iterative self-review) ensure that the project stays aligned with the desired outcomes.
- **Self-Organization:** Although typically involving teams, Agile’s principles can be adapted to solo development, allowing the individual to manage their own work dynamically.

Critical Evaluation of Agile Methodology

Advantages of Agile in a Solo Development Context:

- **Flexibility and Adaptability:** One of the greatest strengths of Agile is its adaptability to change. In this project, where requirements could evolve as the AI model was refined or as the integration between the front-end and back-end progressed, Agile allowed for adjustments without derailing the entire project (Sommerville, 2016).

- **Incremental Progress:** By working in sprints, the project allowed for the continuous delivery of functional components, such as the image processing feature or QR code generation, which could be tested and validated incrementally. This approach reduced the risk of encountering major issues late in the development process.
- **Continuous Improvement:** Agile's focus on iterative development meant that each component of the project could be refined over time, leading to a higher-quality final product. This was particularly beneficial in optimizing the AI model's accuracy and the user interface's usability.

Challenges of Agile in a Solo Development Context:

- **Lack of Collaboration:** Agile is traditionally designed for teams where continuous collaboration and feedback drive the development process. In my solo project, the absence of a team means that the me the developer must self-manage all aspects of the project, which can be challenging without the input and support that a team provides (Sommerville, 2016).
- **Overhead of Agile Practices:** Agile involves maintaining a disciplined process of planning, testing, and review for each sprint. For a solo developer, this can create additional overhead, as the same person is responsible for all roles typically spread across a team, including developer, tester, and project manager.
- **Difficulty in Self-Feedback:** While Agile benefits from continuous feedback, the lack of external input in a solo project can limit the effectiveness of this feedback loop. Self-review, while useful, may not provide the same level of insight as feedback from others.

Conclusion

In the context of this project, Agile methodology offered significant benefits, particularly in managing the complexities and iterative nature of developing an AI-powered application. The flexibility, iterative progress, and focus on continuous improvement made Agile a suitable choice, despite the challenges of applying it in a solo development context. While some of Agile's collaborative benefits were less relevant, the methodology's adaptability and emphasis on delivering functional increments helped ensure the project's success. However, careful management of the additional overhead and self-reliance was necessary to fully leverage Agile's strengths in this solo development project.

Development Tools

The development of the AI-Powered Contact Card Scanner with QR Code Generator involved a comprehensive set of tools and technologies that were crucial at different stages of the project, from frontend development to backend processing and machine learning model training. Below is a detailed description of each of these tools and their roles in the project.

1. HTML, CSS, and JavaScript

- **HTML (HyperText Markup Language):** HTML was used to structure the content on the frontend of the application. It provided the framework for the user interface, defining the layout and organization of elements such as buttons, forms, and images.
- **CSS (Cascading Style Sheets):** CSS was employed to style the HTML elements, making the user interface visually appealing and ensuring a consistent look and feel across different devices. CSS allowed for customization of colors, fonts, and layouts, enhancing the user experience.
- **JavaScript:** JavaScript was the primary scripting language used on the frontend to add interactivity to the application. It enabled dynamic content updates, form validations, and communication with the backend via AJAX calls, ensuring a smooth user interaction without requiring page reloads.
- **Role in the Project:** Together, HTML, CSS, and JavaScript formed the backbone of the frontend development, allowing users to interact with the

application by uploading images, viewing extracted data, and generating QR codes.

2. Chrome DevTools

- **Chrome DevTools:** Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. It was used extensively during the development and debugging of the frontend. The toolset includes features like the Elements panel for inspecting HTML and CSS, the Console for running JavaScript, and the Network panel for monitoring API calls.
- **Role in the Project:** Chrome DevTools was crucial for debugging and optimizing the frontend code, allowing for real-time testing and troubleshooting of the web application.

3. Python

- **Python:** Python was the primary language used for backend development and machine learning model implementation. Its simplicity and wide range of libraries made it ideal for tasks ranging from data processing to AI model training.
- **Role in the Project:** Python was used to build the backend logic, manage API endpoints, process images, and integrate with machine learning models to perform text extraction and categorization.

4. PyTorch and Transformers

- **PyTorch:** PyTorch is an open-source machine learning library used for developing deep learning models. It was used in this project for training and fine-tuning the LayoutLMv2 model on the custom dataset of business cards.
- **Transformers:** The Transformers library by Hugging Face was used to implement and fine-tune the LayoutLMv2 model. This library provides state-of-the-art machine learning models for natural language processing tasks and made it easier to leverage pre-trained models and adapt them to specific use cases.
- **Role in the Project:** PyTorch and the Transformers library were central to the development of the AI model, enabling the training, fine-tuning, and deployment of a model capable of accurately extracting text from business cards.

5. Python Imaging Library (PIL)/Pillow

- **Pillow:** Pillow, a fork of the Python Imaging Library (PIL), was used for image processing tasks. It provided functions for opening, manipulating, and saving image files, which were essential for preparing images before they were processed by the AI model.
- **Role in the Project:** Pillow was used to handle image preprocessing, such as resizing, cropping, and converting images to the format required by the machine learning model.

6. Datasets and Labeled Data

- **Datasets:** The project required a labeled dataset of business cards to train the AI model. The dataset included images of business cards with annotations marking relevant fields such as names, phone numbers, and email addresses.
- **Label Studio:** Label Studio was used to create and manage this labeled dataset. It allowed for the manual labeling of business card images, ensuring that the model could learn to accurately recognize and categorize different types of information.
- **Role in the Project:** The labeled dataset was critical for the fine-tuning of the LayoutLMv2 model, enabling it to achieve high accuracy in text extraction tasks. Label Studio facilitated the creation of this dataset, making it possible to train the model effectively.

7. Flask

- **Flask:** Flask is a lightweight Python web framework that was used to build the backend API. It handled requests from the frontend, processed images, interacted with the AI model, and returned the results to be displayed in the user interface.
- **Role in the Project:** Flask served as the bridge between the frontend and the machine learning components, ensuring seamless communication and data flow within the application.

8. Vue.js

- **Vue.js:** Vue.js is a progressive JavaScript framework used for building user interfaces. In this project, Vue.js was utilized to create a responsive and dynamic frontend, allowing users to interact with the application by uploading images and generating QR codes.

- **Role in the Project:** Vue.js powered the user interface, providing a reactive and modular frontend that could easily integrate with the Flask backend and handle user interactions efficiently.

9. Azure

- **Azure:** Microsoft's Azure cloud platform was used to deploy the AI model and host the backend services. Azure provided the necessary infrastructure for scaling the application and ensuring reliable performance.
- **Role in the Project:** Azure was responsible for hosting the AI model and backend API, making the application accessible to users remotely with high availability and scalability.

10. vCard and QRCode Libraries

- **vCard Library:** The vCard library in Python was used to generate vCard files, which are standardized electronic business cards. This was essential for encoding the structured contact information extracted from business cards.
- **QRCode Library:** The QRCode library was used to generate QR codes from the vCard data. These QR codes could be scanned by smartphones, allowing users to easily add the contact information to their device's address book.
- **Role in the Project:** These libraries were crucial for the final step of the application, enabling the conversion of extracted data into a format that could be easily shared and used by end-users.

Conclusion

The successful development of the AI-Powered Contact Card Scanner with QR Code Generator was supported by a diverse set of development tools. From Python and PyTorch for backend processing and AI model training to Vue.js and Flask for frontend and backend integration, each tool played a vital role in the project's completion. Additional tools like Chrome DevTools, Pillow, and various Python libraries (vCard, QRCode) enhanced the development process, ensuring a robust, scalable, and user-friendly application.

DESIGN

The design phase of the AI-Powered Contact Card Scanner with QR Code Generator involved creating a comprehensive blueprint for the system's functionality, user interaction, and overall structure. This phase was critical in ensuring that the system not only met the functional requirements but also provided a seamless and intuitive user experience. The design elements include the use case diagram, system flowchart, user interface (UI) design, and visual design.

Use Case Diagram

A use case diagram is a graphical representation of the interactions between the users (actors) and the system. It helps in identifying the different ways in which the system can be used and the various functionalities that need to be implemented to meet user needs. The use case diagram typically includes actors (e.g., users, administrators) and use cases (e.g., scanning a business card, generating a QR code).

In this project, the use case diagram was essential for mapping out the key interactions between the user and the system. For example, the primary use cases included "Upload Business Card," "Extract Contact Information," "Generate QR Code," and "Download vCard." These use cases helped to clearly define the scope of the system and provided a basis for further design and development (Booch, Rumbaugh, & Jacobson, 2005).

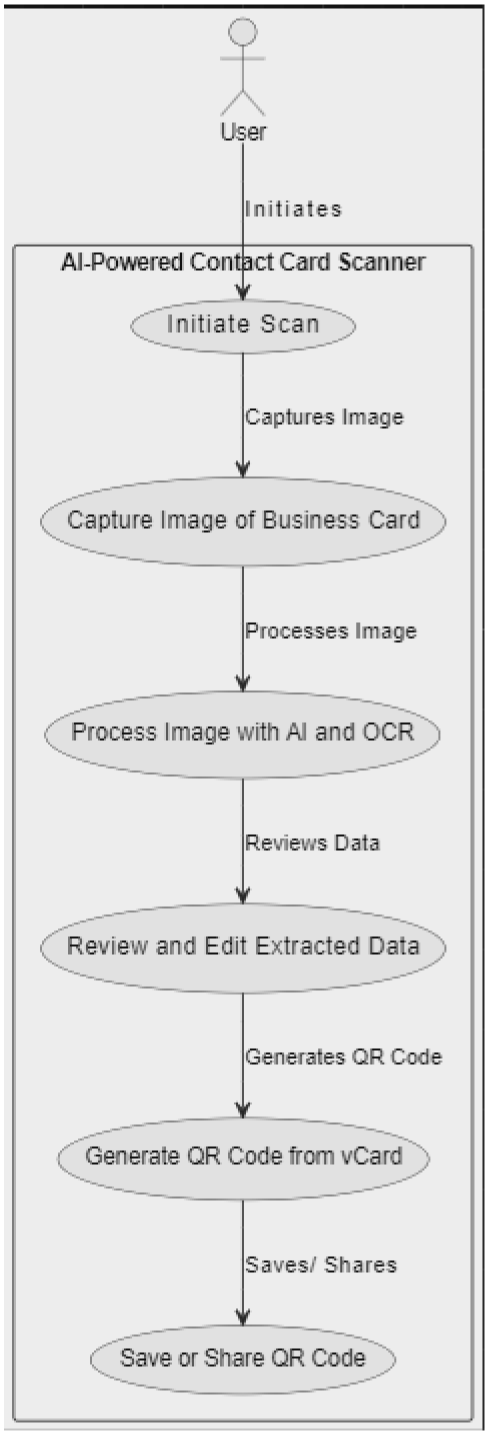

User
Initiates
AI-Powered Contact Card Scanner
Initiate Scan
Captures Image
Capture Image of Business Card
Processes Image
Process Image with AI and OCR
Reviews Data
Review and Edit Extracted Data
Generates QR Code
Generate QR Code from vCard
Saves/ Shares
Save or Share QR Code

Figure 1 Use case

System Flowchart Diagram

A system flowchart diagram visually represents the flow of data and control within the system. It outlines the sequence of operations and the paths that data takes from input to output. Flowcharts are particularly useful for understanding the logic behind the system's processes and identifying potential bottlenecks or inefficiencies.

In this project, the system flowchart diagram detailed the entire process, from the user uploading a business card image to the generation of a QR code. It illustrated the flow of data between the frontend (where the user interacts with the system), the backend (where data processing and AI model inference occur), and external services (such as the Azure-hosted AI model). The flowchart also highlighted key decision points, such as error handling during text extraction or the verification of user input (Sommerville, 2016).

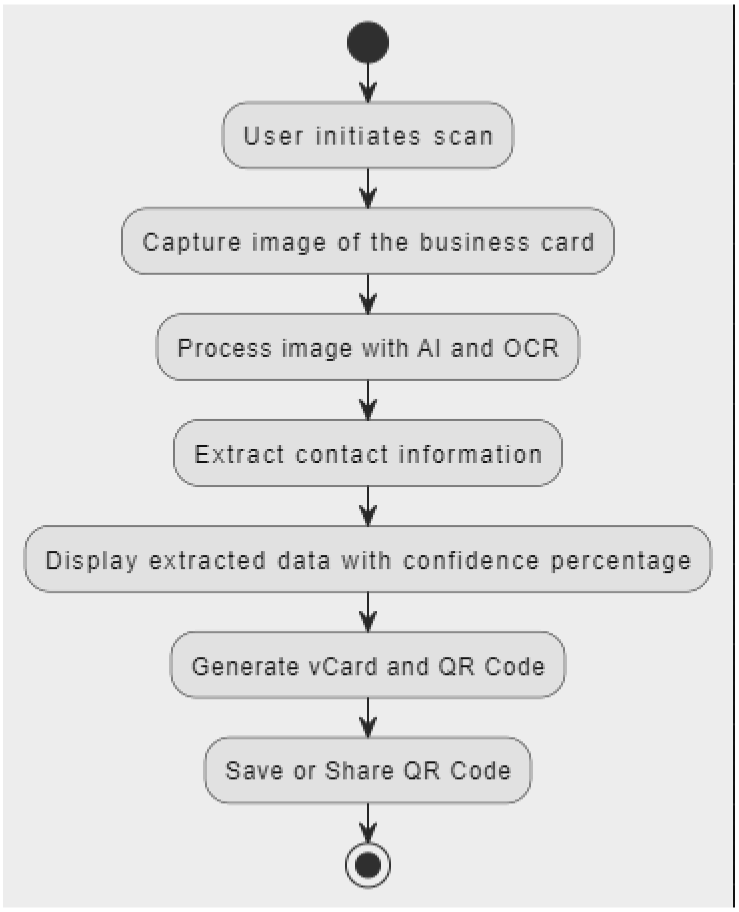

User initiates scan
Capture image of the business card
Process image with AI and OCR
Extract contact information
Display extracted data with confidence percentage
Generate vCard and QR Code
Save or Share QR Code

Figure 2 FlowChart

User Interface Design

User Interface (UI) design focuses on the layout and design of the user interface elements, ensuring that they are intuitive and easy to use. Good UI design is crucial for providing a positive user experience, particularly in applications that involve complex tasks like image processing and data extraction.

For this project, the UI design was centered around simplicity and usability. The main interface allowed users to easily upload business card images, view extracted contact information, and generate QR codes. Key principles of UI design, such as consistency, visibility, and feedback, were applied to ensure that users could navigate the application without confusion. For instance, the design included clear labels, buttons, and progress indicators to guide users through the process (Nielsen, 1994).

Visual Design

Visual design refers to the overall aesthetics of the application, including the choice of colors, typography, imagery, and layout. While UI design focuses on the functional aspects of the interface, visual design ensures that the application is visually appealing and aligns with the intended brand or style.

In this project, the visual design was crafted to be clean and modern, reflecting the advanced technology behind the AI-powered system. A neutral color palette was chosen to create a professional look, while the use of contrasting colors helped to highlight important actions, such as the "Generate QR Code" button. Typography was selected for readability, ensuring that text information, such as extracted contact details, was easy to read on all devices. Additionally, the layout was designed to be responsive, adapting seamlessly to different screen sizes and orientations (Tidwell, 2010).

Following are the ui for the system:

Home Page:

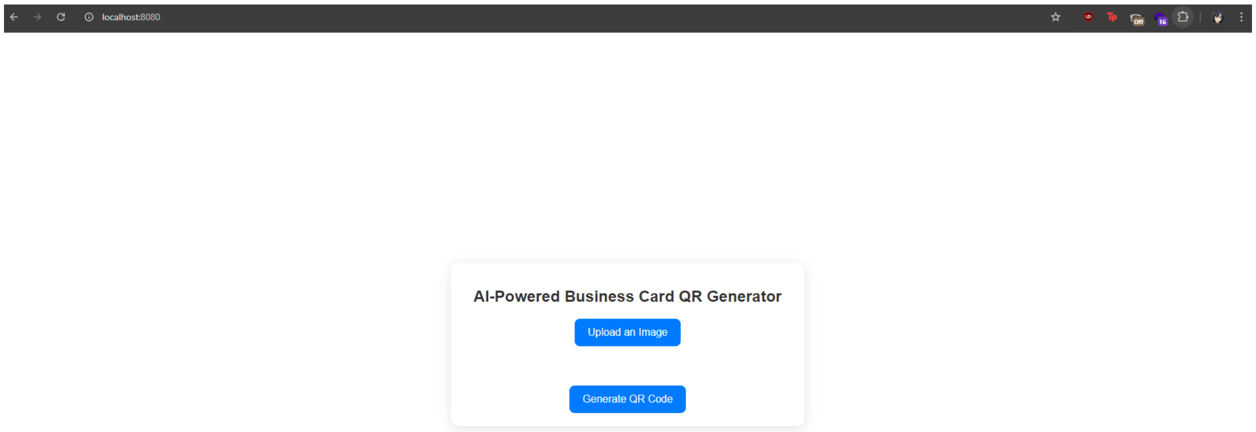


Figure 3 HomePage

Upload Image Page:

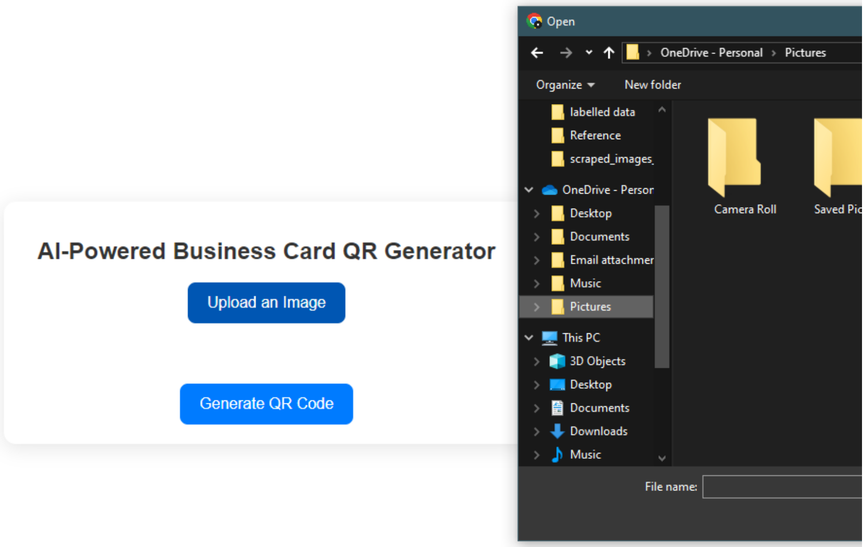


Figure 4 Upload Page

After Image Uploaded Page:

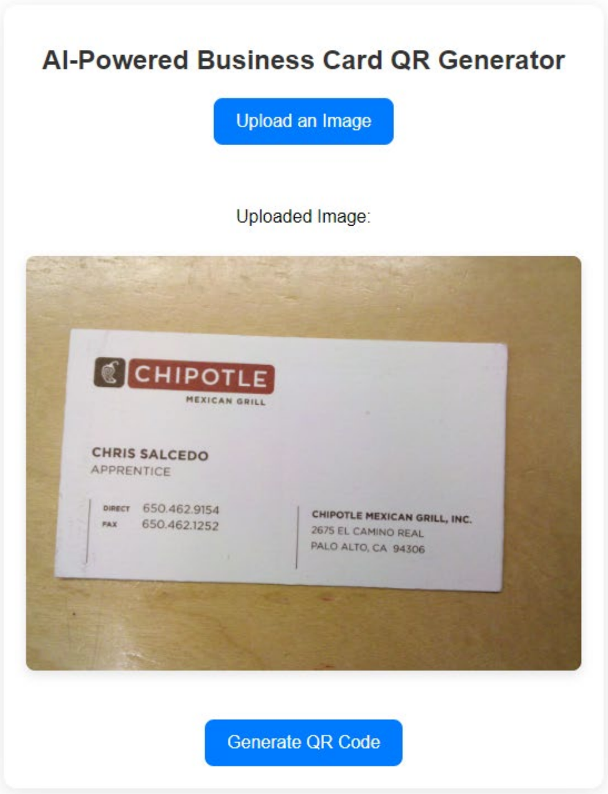


Figure 5 after image uploaded page

Generating QR Code Page:

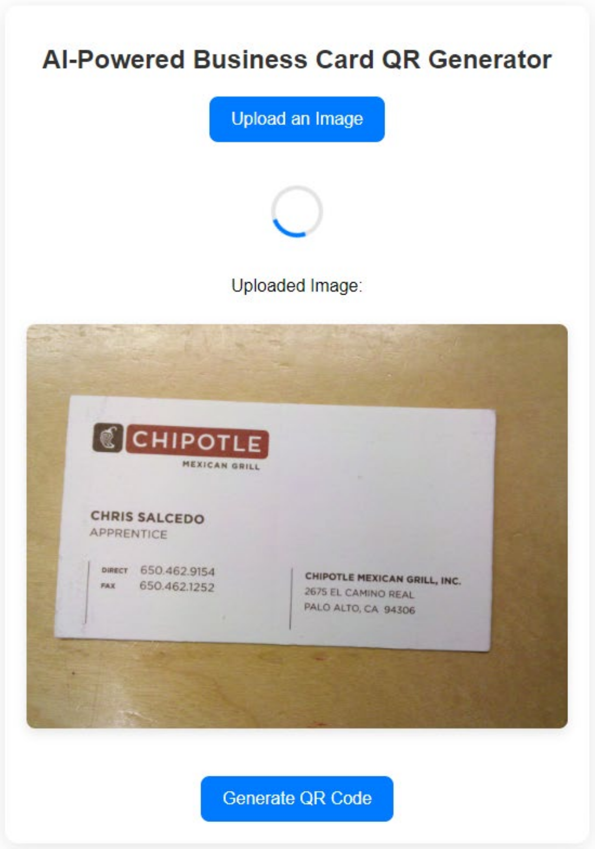
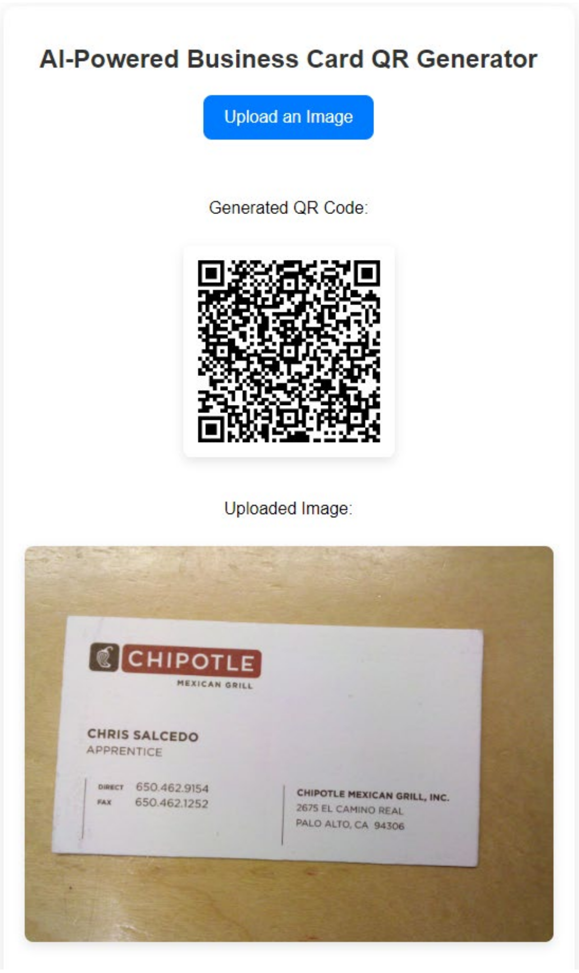
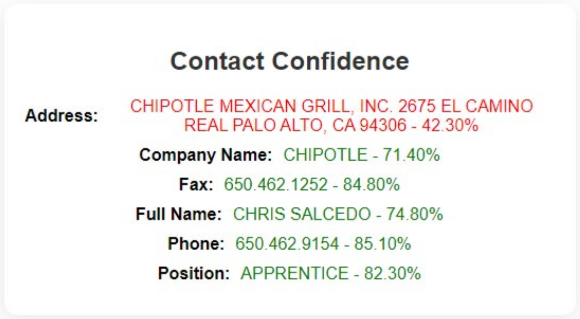


Figure 6 qr code

After QR code Generated Page:





Contact Confidence

Address:	CHIPOTLE MEXICAN GRILL, INC. 2675 EL CAMINO REAL PALO ALTO, CA 94306 - 42.30%
Company Name:	CHIPOTLE - 71.40%
Fax:	650.462.1252 - 84.80%
Full Name:	CHRIS SALCEDO - 74.80%
Phone:	650.462.9154 - 85.10%
Position:	APPRENTICE - 82.30%

Figure 7 after qr code generated

Conclusion

The design phase of this project was crucial in laying the foundation for a functional, user-friendly, and visually appealing application. The use case diagram provided a clear roadmap of user interactions, while the system flowchart ensured that the internal processes were logically structured and efficient. The UI and visual design focused on creating an intuitive and attractive user experience, ensuring that the application not only met its functional requirements but also delighted its users. By carefully considering each of these design elements, the project was able to deliver a well-rounded and effective solution for digitizing business cards and generating QR codes.

IMPLEMENTATION

The implementation of the AI-Powered Contact Card Scanner with QR Code Generator involved a series of well-coordinated steps, leveraging various technologies and development tools to achieve the project's goals. This chapter details the technical implementation process, evaluates how well the project’s aims and objectives were met, and reflects on the results of software testing.

Implementation Process

1. **Backend Development with Flask and Python:**
    - The backend was developed using **Flask**, a lightweight Python web framework that provided the necessary infrastructure for handling HTTP requests and serving as the bridge between the frontend and the AI model. Flask was chosen for its simplicity and flexibility, allowing for rapid development and easy integration with other Python libraries.
2. **AI Model Training and Fine-Tuning on Google Colab:**
    - **Model Training with PyTorch and Transformers:** The core of the project involved developing and fine-tuning an AI model based on **LayoutLMv2**, using the **Transformers** library by Hugging Face and **PyTorch**. This model was crucial for accurately extracting and categorizing text from business card images.
    - **Challenges with Local Environment Setup:** my initial attempts were made to set up the model training environment on a local Integrated Development Environment (IDE). However, the process involved configuring multiple environmental variables, dependencies, and tools like CUDA for GPU acceleration, which proved to be complex and time-consuming. These challenges were compounded by the need for a Linux environment to get the detectron2 working,

something that my local development setup could not easily accommodate.

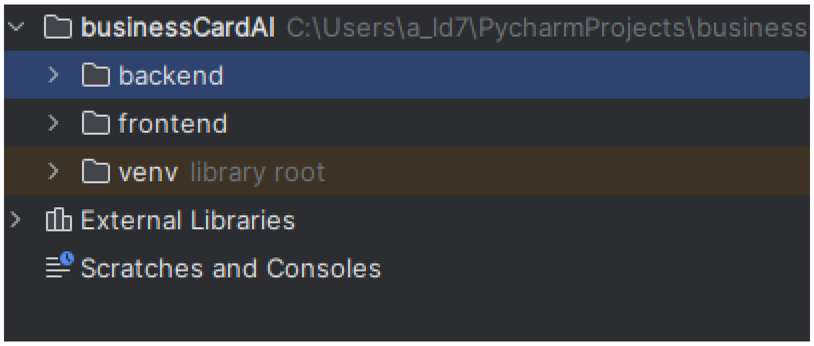
- **Solution with Google Colab:** To overcome these challenges, the model was fine-tuned on **Google Colab**, a cloud-based platform that provides a pre-configured environment with all necessary tools, including GPU support. Colab significantly simplified the process by eliminating the need to manually configure environmental variables and install dependencies. This allowed for a smooth and efficient training process, enabling the model to be fine-tuned on a custom-labeled dataset of business cards.
- **Advantages of Colab:** Google Colab provided the flexibility of a Linux environment without the overhead of setting up a local development environment, making it an ideal solution for managing the complexities of AI model training. The platform's built-in support for libraries like PyTorch and TensorFlow, along with its access to powerful GPUs, accelerated the training process and ensured that the model could be fine-tuned efficiently.

3. **Frontend Development with Vue.js:**
    - The frontend was developed using **Vue.js**, a progressive JavaScript framework known for its ease of use and flexibility. Vue.js was used to create a responsive, user-friendly interface where users could upload images of business cards, view extracted contact information, and generate QR codes.
    - **User Interface Design:** The interface was designed with a focus on simplicity and usability, ensuring that users could easily navigate through the application’s features. Key interface components included image upload functionality, a display area for extracted text, and a button to generate QR codes.
4. **Image Processing with Pillow and Tesseract:**
    - **Pillow:** The **Pillow** library was utilized for image preprocessing tasks, such as resizing, cropping, and converting images to the format required by the AI model. This step was crucial to ensure that the images fed into the model were optimized for accurate text extraction.

- **Tesseract: Tesseract**, an open-source OCR engine, was used as a supplementary tool for text extraction. Although the primary text extraction was handled by the AI model, Tesseract provided a baseline comparison and helped validate the model's outputs.

5. **QR Code Generation and vCard Creation:**
   - The extracted contact information was converted into a vCard format using the **vCard** library in Python. This structured format allowed the data to be easily shared and imported into digital contact lists.
   - **QRCode Library:** The **QRCode** library was used to generate QR codes from the vCard data, enabling users to quickly scan and add the contact information to their smartphones.
6. **Azure for Deployment:**
   - The backend services and AI model were deployed on **Azure**, providing a reliable and scalable platform for the application. Azure's cloud infrastructure ensured that the application was accessible to users with high availability and performance.

Code Structure:



```
businessCardAI C:\Users\a_ld7\PycharmProjects\business
  backend
  frontend
  venv library root
External Libraries
Scratches and Consoles
```

Figure 8 code structure

The code is structured so that the backend and frontend is separated but using the same codebase.

Backend Structure:

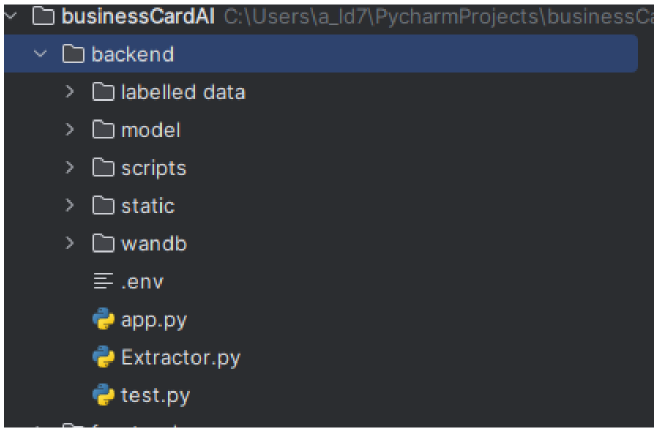
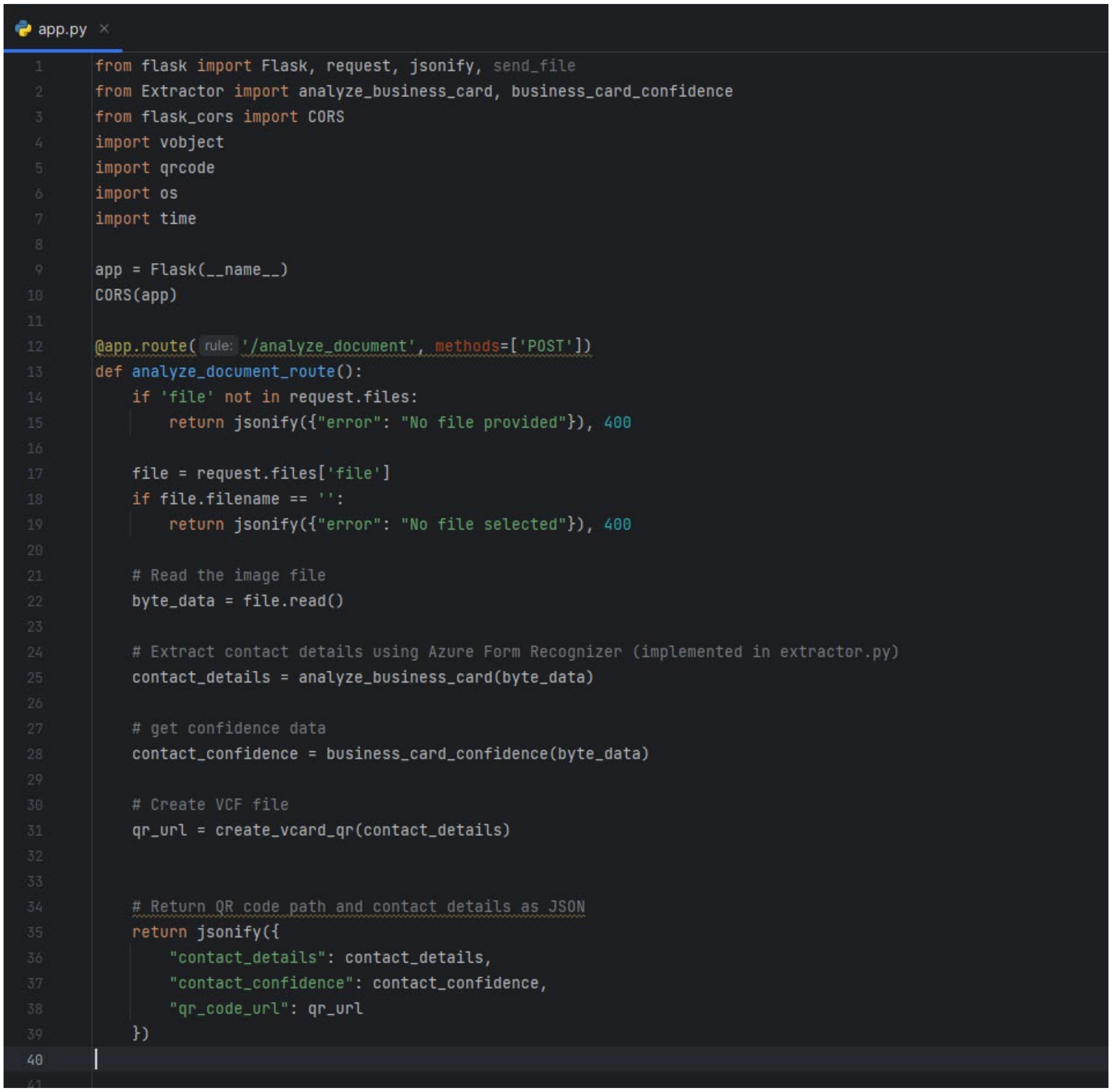

```
businessCardAI C:\Users\a_ld7\PycharmProjects\businessCa
  backend
    labelled data
    model
    scripts
    static
    wandb
    .env
    app.py
    Extractor.py
    test.py
```

Figure 9 backend structure

App.py: Main method for the flask backend



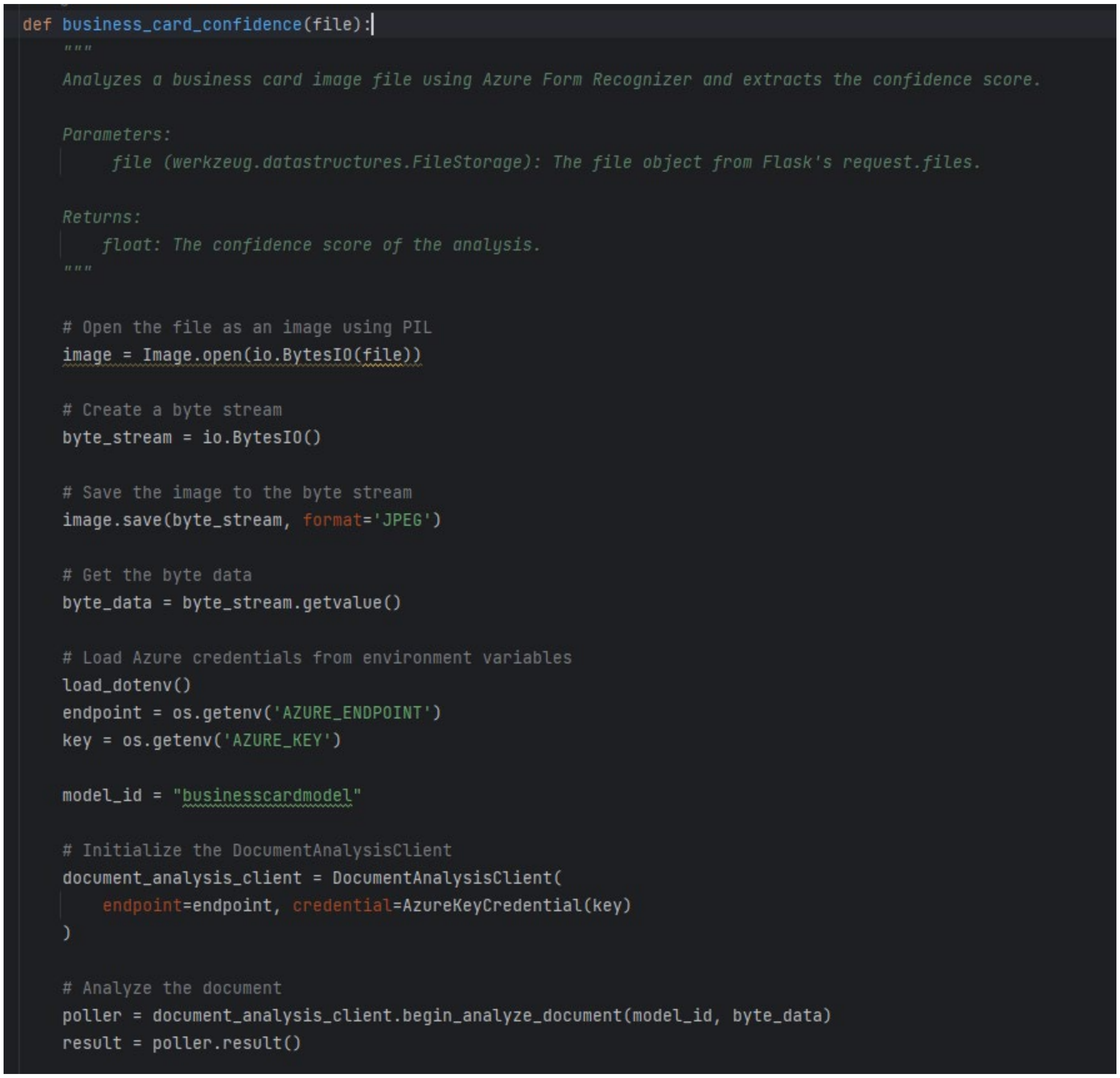
app.py ×

```python
from flask import Flask, request, jsonify, send_file
from Extractor import analyze_business_card, business_card_confidence
from flask_cors import CORS
import vobject
import qrcode
import os
import time

app = Flask(__name__)
CORS(app)

@app.route( rule: '/analyze_document', methods=['POST'])
def analyze_document_route():
    if 'file' not in request.files:
        return jsonify({"error": "No file provided"}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No file selected"}), 400

    # Read the image file
    byte_data = file.read()

    # Extract contact details using Azure Form Recognizer (implemented in extractor.py)
    contact_details = analyze_business_card(byte_data)

    # get confidence data
    contact_confidence = business_card_confidence(byte_data)

    # Create VCF file
    qr_url = create_vcard_qr(contact_details)


    # Return QR code path and contact details as JSON
    return jsonify({
        "contact_details": contact_details,
        "contact_confidence": contact_confidence,
        "qr_code_url": qr_url
    })

```

Figure 10 app.py

Shows the first api endpoint to analyze the uploaded document (image)

Method that process the image using the AI model:

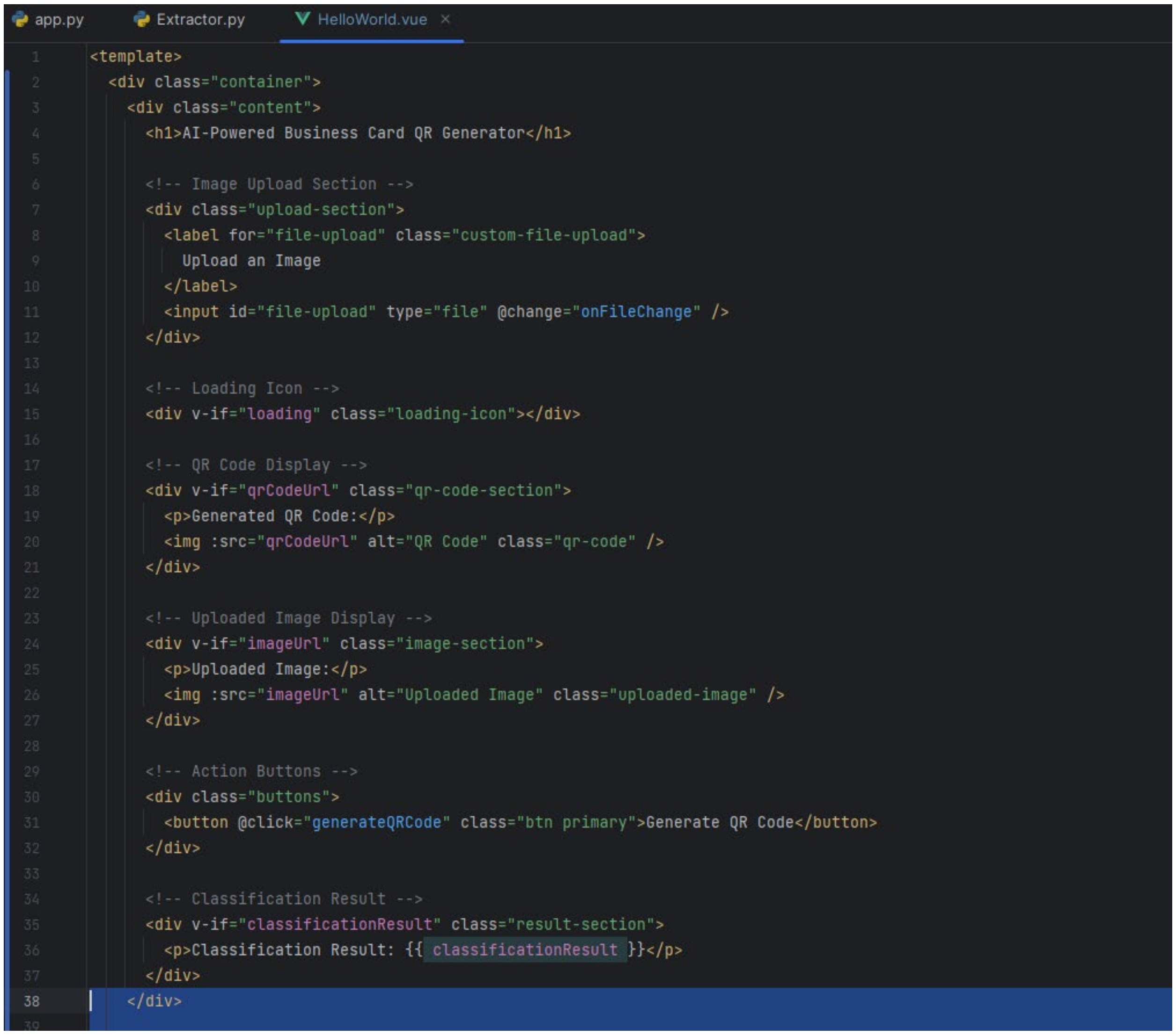


```
def business_card_confidence(file):
    """
    Analyzes a business card image file using Azure Form Recognizer and extracts the confidence score.

    Parameters:
        file (werkzeug.datastructures.FileStorage): The file object from Flask's request.files.

    Returns:
        float: The confidence score of the analysis.
    """

    # Open the file as an image using PIL
    image = Image.open(io.BytesIO(file))

    # Create a byte stream
    byte_stream = io.BytesIO()

    # Save the image to the byte stream
    image.save(byte_stream, format='JPEG')

    # Get the byte data
    byte_data = byte_stream.getvalue()

    # Load Azure credentials from environment variables
    load_dotenv()
    endpoint = os.getenv('AZURE_ENDPOINT')
    key = os.getenv('AZURE_KEY')

    model_id = "businesscardmodel"

    # Initialize the DocumentAnalysisClient
    document_analysis_client = DocumentAnalysisClient(
        endpoint=endpoint, credential=AzureKeyCredential(key)
    )

    # Analyze the document
    poller = document_analysis_client.begin_analyze_document(model_id, byte_data)
    result = poller.result()
```

Figure 11 method to process image

Image is converted into bytestream, and then document analysis is done on the model

Frontend Home Page:

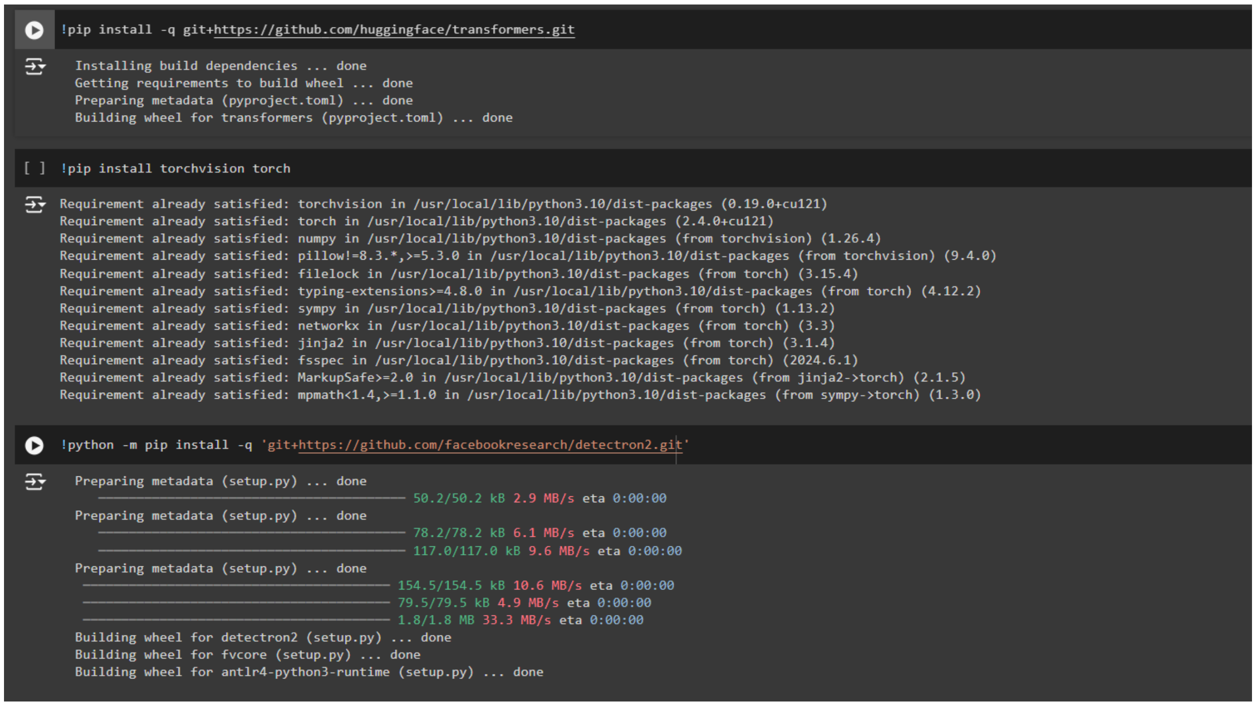


```
<template>
  <div class="container">
    <div class="content">
      <h1>AI-Powered Business Card QR Generator</h1>

      <!-- Image Upload Section -->
      <div class="upload-section">
        <label for="file-upload" class="custom-file-upload">
          Upload an Image
        </label>
        <input id="file-upload" type="file" @change="onFileChange" />
      </div>

      <!-- Loading Icon -->
      <div v-if="loading" class="loading-icon"></div>

      <!-- QR Code Display -->
      <div v-if="qrCodeUrl" class="qr-code-section">
        <p>Generated QR Code:</p>
        <img :src="qrCodeUrl" alt="QR Code" class="qr-code" />
      </div>

      <!-- Uploaded Image Display -->
      <div v-if="imageUrl" class="image-section">
        <p>Uploaded Image:</p>
        <img :src="imageUrl" alt="Uploaded Image" class="uploaded-image" />
      </div>

      <!-- Action Buttons -->
      <div class="buttons">
        <button @click="generateQRCode" class="btn primary">Generate QR Code</button>
      </div>

      <!-- Classification Result -->
      <div v-if="classificationResult" class="result-section">
        <p>Classification Result: {{ classificationResult }}</p>
      </div>
    </div>
```

Figure 12 frontend home page

Google Colab for Model Fine-Tuning:

Link: https://colab.research.google.com/drive/1nNMJBeazi_EHx9cdnC4M-YU4kO0aXTE_?usp=sharing



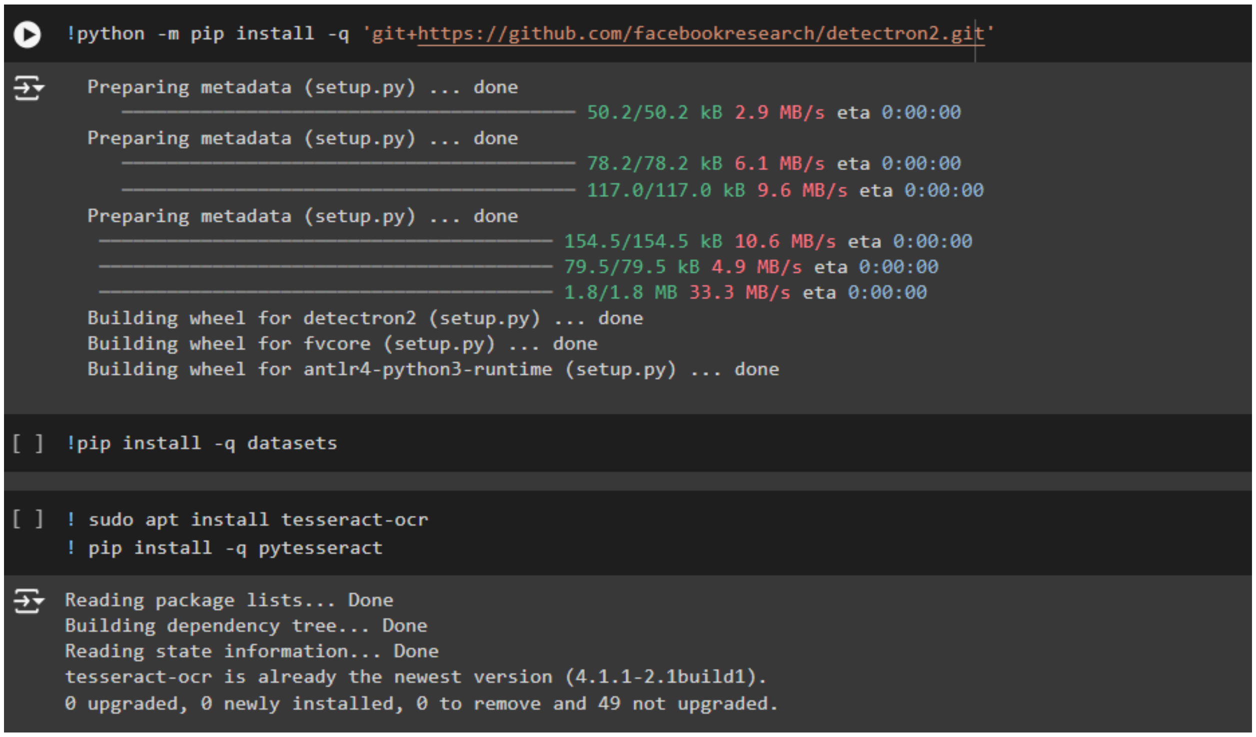
```
!pip install -q git+https://github.com/huggingface/transformers.git

  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for transformers (pyproject.toml) ... done
```

```
[ ] !pip install torchvision torch

Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.19.0+cu121)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.4.0+cu121)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (9.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.15.4)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
```

```
!python -m pip install -q 'git+https://github.com/facebookresearch/detectron2.git'

  Preparing metadata (setup.py) ... done
     ──────────────────────────────────────── 50.2/50.2 kB 2.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ──────────────────────────────────────── 78.2/78.2 kB 6.1 MB/s eta 0:00:00
     ──────────────────────────────────────── 117.0/117.0 kB 9.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
   ──────────────────────────────────────── 154.5/154.5 kB 10.6 MB/s eta 0:00:00
   ──────────────────────────────────────── 79.5/79.5 kB 4.9 MB/s eta 0:00:00
   ──────────────────────────────────────── 1.8/1.8 MB 33.3 MB/s eta 0:00:00
  Building wheel for detectron2 (setup.py) ... done
  Building wheel for fvcore (setup.py) ... done
  Building wheel for antlr4-python3-runtime (setup.py) ... done
```

Figure 13 colab environment

Installing necessary environment variables

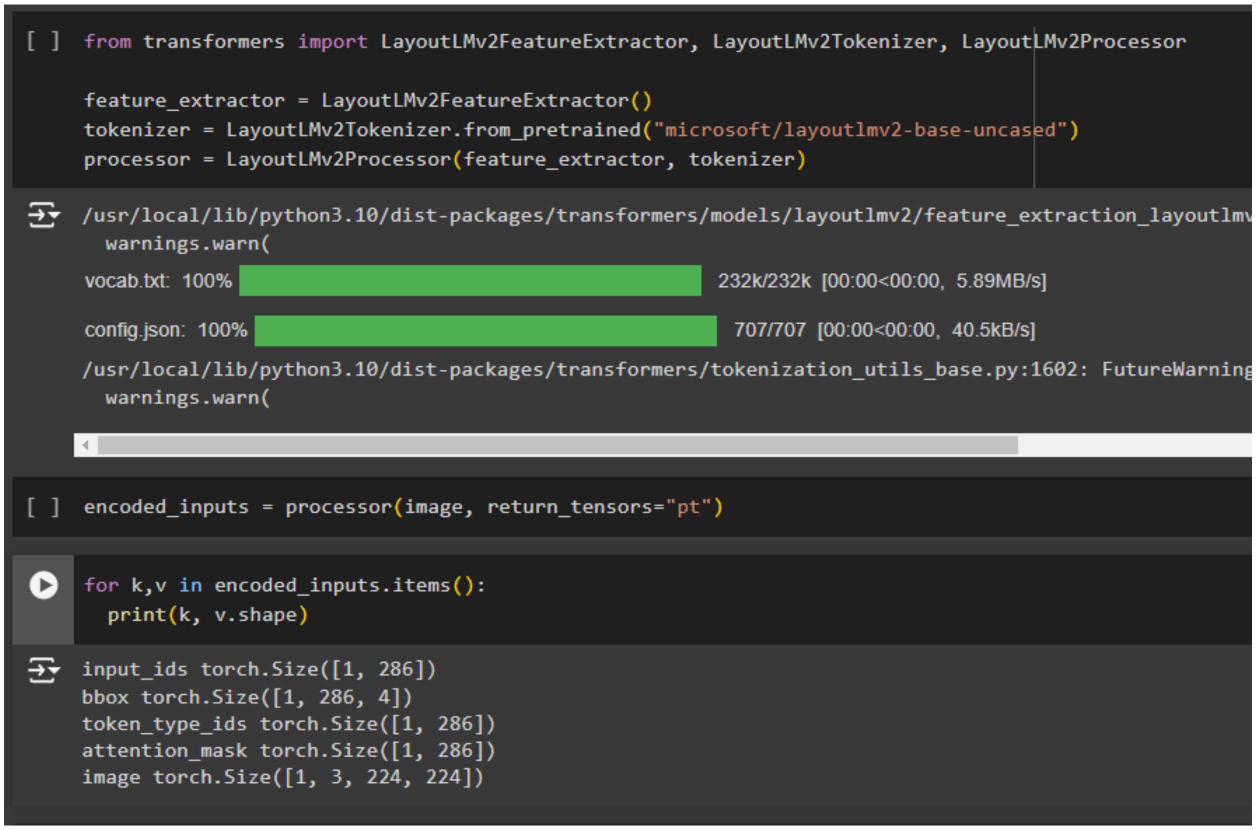


```
!python -m pip install -q 'git+https://github.com/facebookresearch/detectron2.git'

  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.2/50.2 kB 2.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.2/78.2 kB 6.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.0/117.0 kB 9.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.5/154.5 kB 10.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.5/79.5 kB 4.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 33.3 MB/s eta 0:00:00
  Building wheel for detectron2 (setup.py) ... done
  Building wheel for fvcore (setup.py) ... done
  Building wheel for antlr4-python3-runtime (setup.py) ... done
```

```
[ ]  !pip install -q datasets
```

```
[ ]  ! sudo apt install tesseract-ocr
     ! pip install -q pytesseract

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
```

Figure 14 colab environment



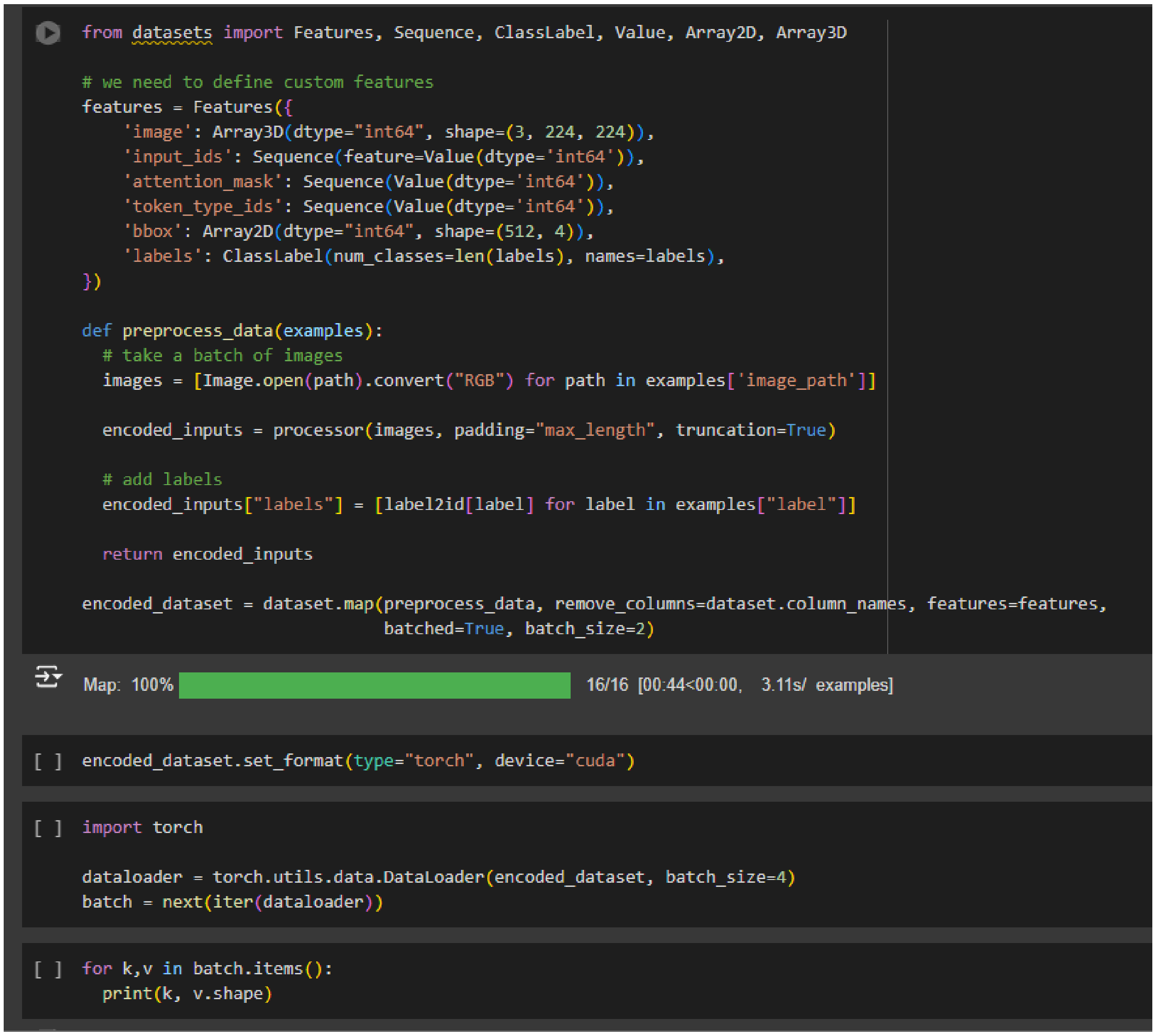
```
[ ]  from transformers import LayoutLMv2FeatureExtractor, LayoutLMv2Tokenizer, LayoutLMv2Processor

     feature_extractor = LayoutLMv2FeatureExtractor()
     tokenizer = LayoutLMv2Tokenizer.from_pretrained("microsoft/layoutlmv2-base-uncased")
     processor = LayoutLMv2Processor(feature_extractor, tokenizer)

/usr/local/lib/python3.10/dist-packages/transformers/models/layoutlmv2/feature_extraction_layoutlmv
  warnings.warn(
vocab.txt: 100%   232k/232k [00:00<00:00, 5.89MB/s]
config.json: 100%   707/707 [00:00<00:00, 40.5kB/s]
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1602: FutureWarning
  warnings.warn(
```

```
[ ]  encoded_inputs = processor(image, return_tensors="pt")
```

```
for k,v in encoded_inputs.items():
  print(k, v.shape)

input_ids torch.Size([1, 286])
bbox torch.Size([1, 286, 4])
token_type_ids torch.Size([1, 286])
attention_mask torch.Size([1, 286])
image torch.Size([1, 3, 224, 224])
```

Figure 15 model importing

Importing the model and encoding the inputs



```
from datasets import Features, Sequence, ClassLabel, Value, Array2D, Array3D

# we need to define custom features
features = Features({
    'image': Array3D(dtype="int64", shape=(3, 224, 224)),
    'input_ids': Sequence(feature=Value(dtype='int64')),
    'attention_mask': Sequence(Value(dtype='int64')),
    'token_type_ids': Sequence(Value(dtype='int64')),
    'bbox': Array2D(dtype="int64", shape=(512, 4)),
    'labels': ClassLabel(num_classes=len(labels), names=labels),
})

def preprocess_data(examples):
  # take a batch of images
  images = [Image.open(path).convert("RGB") for path in examples['image_path']]

  encoded_inputs = processor(images, padding="max_length", truncation=True)

  # add labels
  encoded_inputs["labels"] = [label2id[label] for label in examples["label"]]

  return encoded_inputs

encoded_dataset = dataset.map(preprocess_data, remove_columns=dataset.column_names, features=features,
                              batched=True, batch_size=2)
```

```
Map: 100% 16/16 [00:44<00:00,  3.11s/ examples]
```

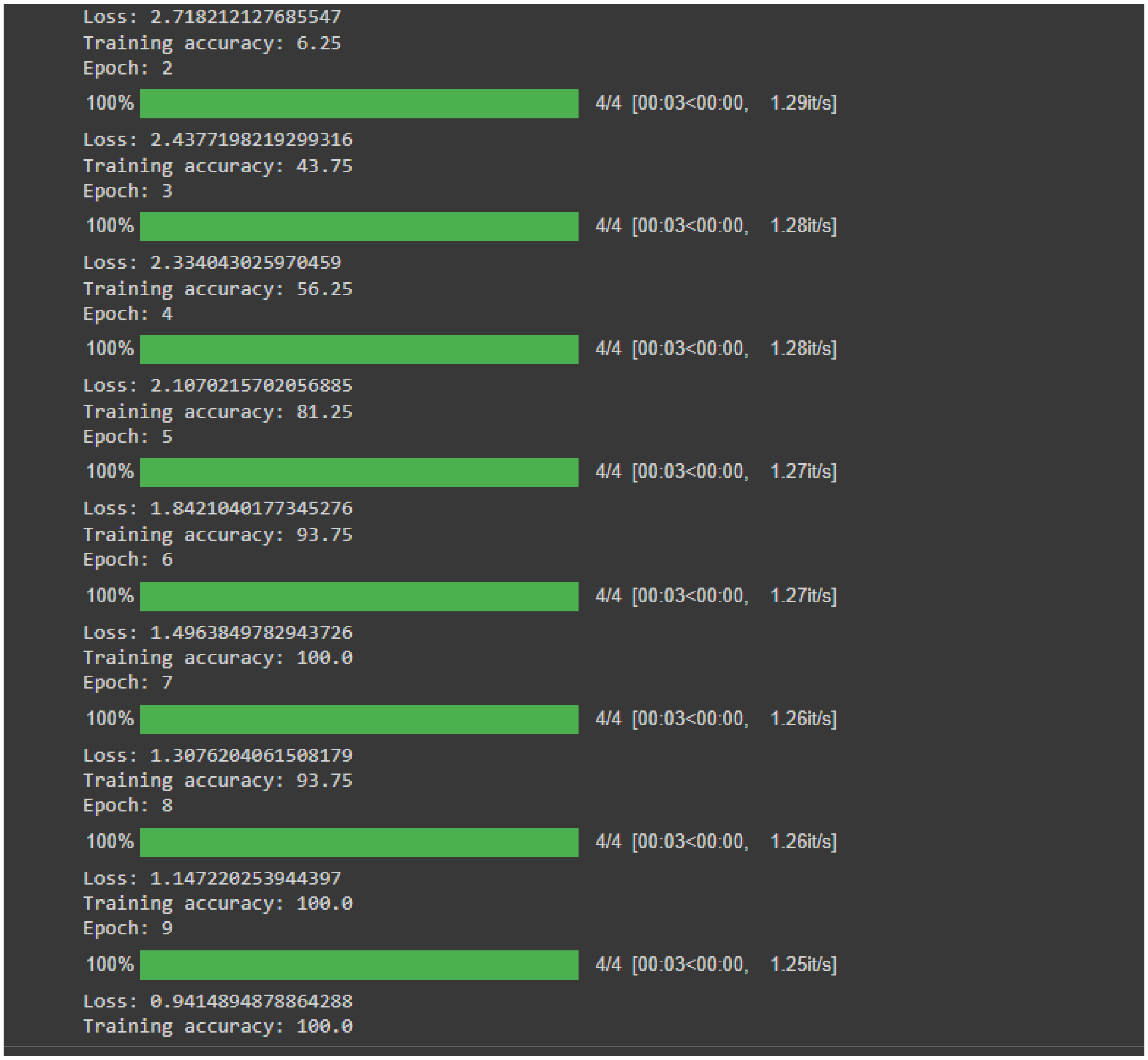
```
[ ] encoded_dataset.set_format(type="torch", device="cuda")
```

```
[ ] import torch

    dataloader = torch.utils.data.DataLoader(encoded_dataset, batch_size=4)
    batch = next(iter(dataloader))
```

```
[ ] for k,v in batch.items():
      print(k, v.shape)
```

Figure 16 processing data

Preprocessing the data

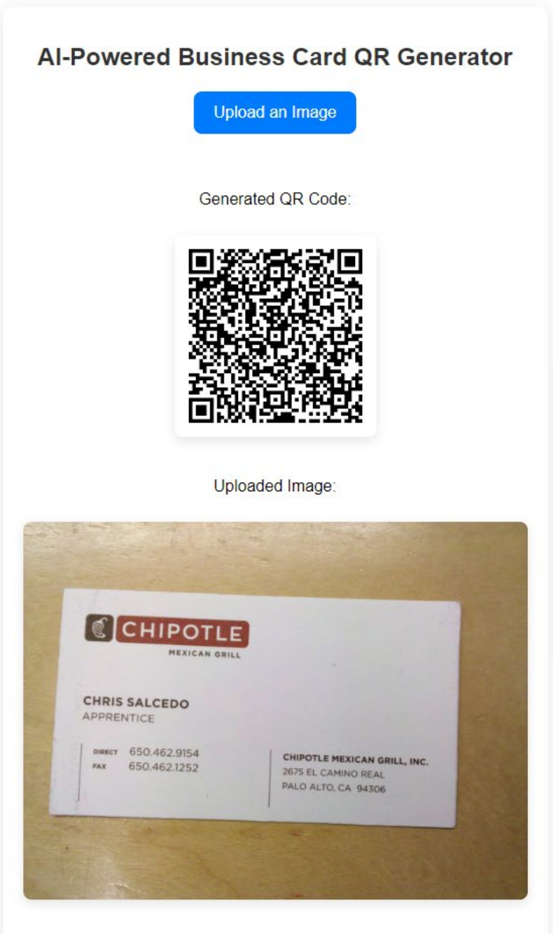


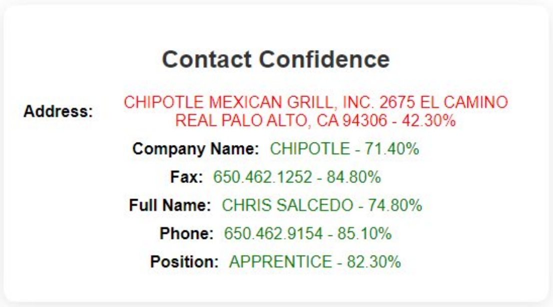
```
Loss: 2.718212127685547
Training accuracy: 6.25
Epoch: 2
100% 4/4 [00:03<00:00,  1.29it/s]
Loss: 2.4377198219299316
Training accuracy: 43.75
Epoch: 3
100% 4/4 [00:03<00:00,  1.28it/s]
Loss: 2.334043025970459
Training accuracy: 56.25
Epoch: 4
100% 4/4 [00:03<00:00,  1.28it/s]
Loss: 2.1070215702056885
Training accuracy: 81.25
Epoch: 5
100% 4/4 [00:03<00:00,  1.27it/s]
Loss: 1.8421040177345276
Training accuracy: 93.75
Epoch: 6
100% 4/4 [00:03<00:00,  1.27it/s]
Loss: 1.4963849782943726
Training accuracy: 100.0
Epoch: 7
100% 4/4 [00:03<00:00,  1.26it/s]
Loss: 1.3076204061508179
Training accuracy: 93.75
Epoch: 8
100% 4/4 [00:03<00:00,  1.26it/s]
Loss: 1.147220253944397
Training accuracy: 100.0
Epoch: 9
100% 4/4 [00:03<00:00,  1.25it/s]
Loss: 0.9414894878864288
Training accuracy: 100.0
```

Figure 17 training model

Training the Model

Implementation Result:





Contact Confidence

Address: CHIPOTLE MEXICAN GRILL, INC. 2675 EL CAMINO REAL PALO ALTO, CA 94306 - 42.30%

Company Name: CHIPOTLE - 71.40%

Fax: 650.462.1252 - 84.80%

Full Name: CHRIS SALCEDO - 74.80%

Phone: 650.462.9154 - 85.10%

Position: APPRENTICE - 82.30%

Figure 18 Result

Perfectly Generates the QR code based on the image provided, with correct names and addresses, further more on the right we can see what the model confidence is as well showing the users that name and phone has high confidence

Meeting Aims and Objectives

The primary aim of this project was to develop an AI-powered application that could accurately extract contact information from business cards and generate a QR code for seamless digital transfer. The implementation effectively met these aims and objectives:

- **Accurate Text Extraction:** The fine-tuning of the LayoutLMv2 model on Google Colab resulted in a highly accurate text extraction system, capable of handling various business card designs.
- **Efficient Process Integration:** The seamless integration of the backend API with the AI model and the Vue.js frontend ensured a smooth user experience, from image upload to QR code generation.
- **User-Friendly Interface:** The responsive and intuitive user interface made it easy for users to interact with the application and manage their contact information.

Software Testing

Comprehensive testing was conducted to ensure that the application functioned correctly and met the specified requirements:

1. **Unit Testing:**
    - Unit tests were written for key components of the backend, including API endpoints and the AI model's inference functions. These tests ensured that individual components operated as expected.
2. **Integration Testing:**
    - Integration tests were performed to verify that the backend, frontend, and AI model worked together seamlessly. This included testing the full workflow from image upload to QR code generation, ensuring that data was correctly processed and displayed.
3. **Usability Testing:**

    - Usability tests assessed the user experience, focusing on the ease of use of the application. Feedback from users was used to refine the interface, ensuring that it was intuitive and easy to navigate.
4. **Performance Testing:**
    - The system was tested under varying loads to ensure that it could handle multiple simultaneous requests without significant delays, particularly during image processing and AI inference.

Reflection on Test Results

The testing phase confirmed that the application met its functional requirements and performed reliably under normal operating conditions. Key reflections include:

- **High Accuracy in Text Extraction:** The AI model demonstrated high accuracy in extracting text across different business card designs, validating the effectiveness of the fine-tuning process on Google Colab.
- **Efficient QR Code Generation:** The application was able to generate QR codes quickly, allowing users to add contact details to their phones with minimal effort.
- **Positive User Experience:** Usability testing indicated that the interface was intuitive and well-received by users. The ability to process images quickly and generate accurate QR codes was particularly appreciated.
- **Areas for Improvement:** Some edge cases in text extraction, particularly with highly stylized or non-standard business cards, highlighted opportunities for further improvement in the AI model or additional preprocessing steps.

Conclusion

The implementation of the AI-Powered Contact Card Scanner with QR Code Generator successfully achieved its objectives, leveraging a range of development tools and technologies to deliver a reliable and user-friendly application. The decision to fine-tune the AI model on Google Colab proved to be crucial, overcoming the challenges of a complex local setup and providing a powerful, flexible environment for model training. The positive results from the testing phase confirm that the project met its goals and provided a strong foundation for potential future enhancements.

EVALUATION

The development of the AI-Powered Contact Card Scanner with QR Code Generator was a challenging yet rewarding experience that provided valuable insights into the complexities of software development, particularly in integrating AI technologies with web-based applications. This chapter reflects on the project's outcomes, the difficulties encountered, the strategies used to overcome them, and considerations for future improvements.

Project Outcome

The primary objective of this project was to create a user-friendly application capable of scanning business cards, extracting contact information using AI, and generating QR codes for easy digital transfer. The project successfully met these objectives, resulting in a functional system that can accurately process business cards and deliver a seamless user experience. The integration of technologies such as LayoutLMv2, Flask, Vue.js, and Azure demonstrated a strong technical implementation that aligned with the project's goals.

Difficulties Faced and Solutions

Throughout the project, several challenges were encountered, particularly in the areas of AI model training, system integration, and environment setup.

1. **AI Model Training Challenges:**
    - **Complexity of Environment Setup:** One of the significant challenges was the complexity of setting up the AI model training environment on a local IDE. The process required configuring multiple environmental variables, dependencies, and GPU support, which proved to be difficult due to the need for a Linux environment.
    - **Solution:** The decision to use **Google Colab** for model fine-tuning was a pivotal solution. Colab provided a pre-configured, Linux-based environment with built-in support for GPUs and necessary libraries, streamlining the training process and eliminating the need for complex local configurations.
2. **System Integration:**
    - **Data Flow and API Integration:** Integrating the frontend with the AI model via the Flask backend presented challenges in ensuring smooth

data flow and consistent API responses. Coordinating the different components and managing asynchronous operations required careful attention to detail.

   - **Solution:** Rigorous testing and iterative development allowed for the identification and resolution of integration issues early on. Continuous debugging and the use of tools like Chrome DevTools helped to ensure that the frontend and backend communicated effectively.

3. **User Interface and Experience:**
   - **Designing for Usability:** Creating an intuitive and user-friendly interface was critical, yet challenging, especially when balancing functionality with simplicity. Ensuring that users could easily upload images, view extracted data, and generate QR codes without confusion required multiple iterations of the UI design.
   - **Solution:** Usability testing played a crucial role in refining the interface. Feedback from users was used to make adjustments, ensuring that the final design was both functional and easy to navigate.

Limitations and Test Reflections

While the project achieved its primary objectives, several limitations were identified during the testing and evaluation phases:

1. **Text Extraction Accuracy:**
   - **Limitation:** Although the AI model performed well on most business cards, there were instances where text extraction accuracy declined, particularly with highly stylized or unconventional designs
   - **Reflection:** The limitations in text extraction highlight the need for further training of the model on a more diverse dataset ,this is mainly due to my limited training data as I was only able to get 10 labelled images of cards, with further data it’s possible to make the model even have higher accuracy
2. **System Performance:**

- **Limitation:** the main limitation was when training the model, I used the colab GPU model but was limited due to its poor availability, when training the model using torch CUDA
- **Reflection:** This can be improved if we have a powerful GPU to train the model

Room for Improvement and Further Work

The project, while successful, offers several opportunities for enhancement and future development:

1. **Model Enhancement:**
   - **Further Training:** To improve text extraction accuracy, the AI model could be trained on a more extensive and varied dataset, including business cards with diverse designs, fonts, and languages. Additionally, incorporating advanced preprocessing techniques could help the model better handle complex images.
2. **Scalability and Performance Optimization:**
   - **Optimization:** future work could focus on either leveraging Cloud GPU or running the model training locally on a powerful graphics cards and leveraging NVidia cuda framework with its tensor cores, this will train the model even faster with more accuracy
3. **User Experience Enhancements:**
   - **Advanced User Features:** Adding features such as batch processing of multiple business cards, or the ability to manually correct extracted data before generating a QR code, could enhance the user experience.
   - **Mobile Optimization:** This is probably the biggest improvement this project can have, having a mobile application allows the users to just take a picture or even scan the business card, generate the QR code either share the QR code or show it to others via the display.

Reflection on Project Components

1. **Research:**
   - The research phase provided a solid foundation for understanding the complexities of AI-based text extraction and QR code generation. The literature review guided the selection of tools and methodologies, ensuring that the project was grounded in current best practices.
2. **Requirements Analysis:**
   - The requirements analysis was thorough and well-structured, identifying both functional and non-functional needs. This clarity in requirements was crucial for guiding the development process and ensuring that all project goals were met.
3. **Implementation:**
   - The implementation phase demonstrated the successful integration of various technologies, from AI model training to frontend development. The use of Google Colab for model training was a key decision that enabled the project to overcome significant technical challenges.
4. **Testing:**
   - Comprehensive testing ensured that the system was robust and met the defined requirements. However, the testing phase also revealed areas for potential improvement, particularly in handling edge cases and optimizing performance.

CONCLUSION

This report includes the detailed development and implementation of the ai powered contact card scanner with QR code generator system, the project aimed at creating a user friendly application which allows user to digitalize their contact information on the fly this project successfully integrates variety of tools such as flask,vue.js and the multi-modal LayoutLMv2.

The primary outcome of this project is a fully functional application that allows users to easily scan business cards and extract its contact detail as a QR code and when scanned it immediately goes to the contact details page in any type of phone regardless of its OS or type as it's in a universally accepted format called vCards.

Despite the challenges encountered, such as setting up the AI model training environment and ensuring smooth integration between components, the project met its objectives effectively. The use of Google Colab for model fine-tuning provided a practical solution to overcome environmental setup issues, enabling the successful training and deployment of the AI model.

In summary, this project not only met the intended goals but also provided valuable insights into the integration of AI with web-based applications. The final product is a robust, scalable, and user-centric application that fulfills its purpose of making contact management easier and more efficient. The project lays a solid foundation for future enhancements and further exploration into more advanced features and optimizations.

REFERENCES

Booch, G. (2017) *Unified Modeling Language User Guide.*

Chung, L. and do Prado Leite, J.C.S. (2009) On Non-Functional Requirements in Software Engineering. *Conceptual Modeling: Foundations and Applications* [online]. pp. 363–379.

Glinz, M. (2007) On Non-Functional Requirements. *15th IEEE International Requirements Engineering Conference (RE 2007)* [online]. Available from: https://ieeexplore.ieee.org/abstract/document/4384163.

Hussain, I., Ahmad, R., Muhammad, S., Ullah, K., Shah, H. and Abdallah Namoun (2022) PHTI: Pashto Handwritten Text Imagebase for Deep Learning Applications. *IEEE Access* [online]. 10, pp. 113149–113157. [Accessed 17 September 2023].

Jézéquel, J.-M. (2012) Model-Driven Engineering for Software Product Lines. *ISRN Software Engineering* [online]. 2012, pp. 1–24. [Accessed 30 October 2019].

Kaur, R. and Kaur, A. (2023) Text Generator using Natural Language Processing Methods. [online]. [Accessed 31 August 2024].

Khan, F., Jan, S.R., Tahir, M., Khan, S. and Ullah, F. (2016) Survey: Dealing Non-Functional Requirements at Architecture Level. *VFAST Transactions on Software Engineering* [online]. 9 (2), p. 7. [Accessed 27 April 2020].

Kumar, D., Kumar, A. and Singh, L. (2022) Non-functional Requirements Elicitation in Agile Base Models. *Webology* [online]. 19 (1), pp. 1992–2018.

Mairiza, D., Zowghi, D. and Nurmuliani, N. (2010) An investigation into the notion of non-functional requirements. *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10* [online].

Nielsen, J. (1993) *Usability Engineering*. Amsterdam, Morgan Kaufmann.

Sommerville, I. (2016) *Software engineering* 10th edition. Harlow, Pearson Education.

Tambe, T. *et al*. (2023) A 16-nm SoC for Noise-Robust Speech and NLP Edge AI Inference With Bayesian Sound Source Separation and Attention-Based DNNs. *IEEE Journal of Solid-State Circuits* [online]. 58 (2), pp. 569–581. [Accessed 31 August 2024].

Tidwell (2006) *Designing interfaces : [patterns for effective interaction design]*. Sebastopol, Ca, O'reilly.

Ullah, S., Iqbal, M. and Khan, A.M. (2011) A survey on issues in non-functional requirements elicitation. *International Conference on Computer Networks and Information Technology* [online].

Wong, M.-F., Guo, S., Hang, C.-N., Ho, S.-W. and Tan, C.-W. (2023) Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review. *Entropy* [online]. 25 (6), p. 888. Available from: https://www.mdpi.com/1099-4300/25/6/888.

Xu, Y. *et al*. (2021) LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding. [online].