BORDERLESS.AI

Immigration AI Advisor

Team: Borderless.AI
Sprint 1 Presentation

Agenda

Team Member Roles and Responsibilities

Matt Borkowski

Developer

SCRUM Master

Miles ZhengFerrari

Developer

Team Leader

Nuvorish Paul

Developer

Raj Rahul Parekh

Developer

Santhosh Charudatta

Developer

All team members collaborate as developers, contributing to coding, testing, and feature development.

Improvements

Team Working Agreement

Project Schedule

Personas

Gmail Group: borderlessai@googlegroups.com

Wiki Page

Project Description

Project Name: Immigration AI Advisor

Team: Borderless.AI

Project Description:

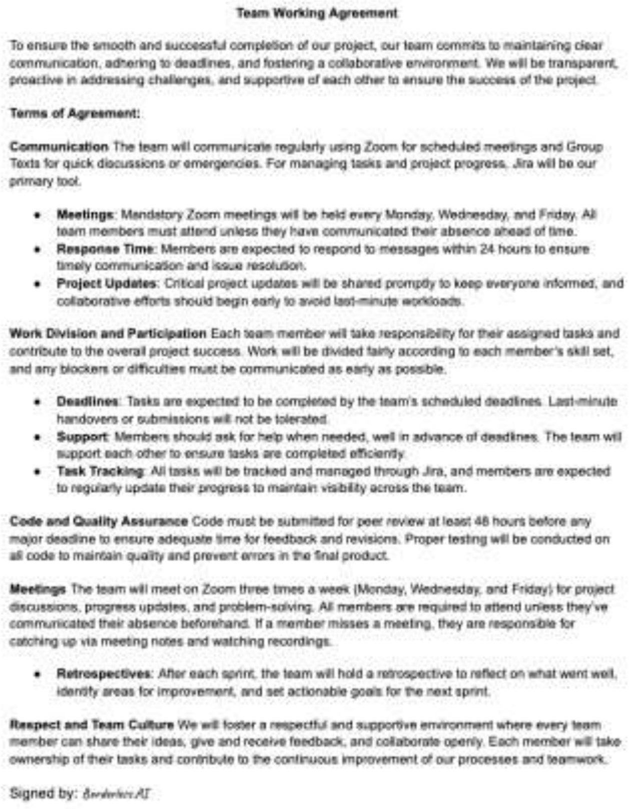
Immigration AI Advisor is a AI-based application designed to assist international students with immigration-related queries by reading documents (I-20, I-94) and providing personalized advice based on immigration rules and policies.

Benefit Outcomes:

- Faster, more accurate immigration advice for international students.
- Reduced burden on international student offices with limited resources.
- Improved efficiency in managing immigration-related queries within budget constraints.

Github Link:https://github.com/htmw/2024F-borderless.ai/wiki

Team Working Agreement



Team Working Agreement

To ensure the smooth and successful completion of our project, our team commits to maintaining clear communication, adhering to deadlines, and fostering a collaborative environment. We will be transparent, proactive in addressing challenges, and supportive of each other to ensure the success of the project.

Terms of Agreement:

Communication The team will communicate regularly using Zoom for scheduled meetings and Group Texts for quick discussions or emergencies. For managing tasks and project progress, Jira will be our primary tool.

- **Meetings**: Mandatory Zoom meetings will be held every Monday, Wednesday, and Friday. All team members must attend unless they have communicated their absence ahead of time.
- **Response Time**: Members are expected to respond to messages within 24 hours to ensure timely communication and issue resolution.
- **Project Updates**: Critical project updates will be shared promptly to keep everyone informed, and collaborative efforts should begin early to avoid last-minute workloads.

Work Division and Participation Each team member will take responsibility for their assigned tasks and contribute to the overall project success. Work will be divided fairly according to each member's skill set, and any blockers or difficulties must be communicated as early as possible.

- **Deadlines**: Tasks are expected to be completed by the team's scheduled deadlines. Last-minute handovers or submissions will not be tolerated.
- **Support**: Members should ask for help when needed, well in advance of deadlines. The team will support each other to ensure tasks are completed efficiently.
- **Task Tracking**: All tasks will be tracked and managed through Jira, and members are expected to regularly update their progress to maintain visibility across the team.

Code and Quality Assurance Code must be submitted for peer review at least 48 hours before any major deadline to ensure adequate time for feedback and revisions. Proper testing will be conducted on all code to maintain quality and prevent errors in the final product.

Meetings The team will meet on Zoom three times a week (Monday, Wednesday, and Friday) for project discussions, progress updates, and problem-solving. All members are required to attend unless they've communicated their absence beforehand. If a member misses a meeting, they are responsible for catching up via meeting notes and watching recordings.

- **Retrospectives**: After each sprint, the team will hold a retrospective to reflect on what went well, identify areas for improvement, and set actionable goals for the next sprint.

Respect and Team Culture We will foster a respectful and supportive environment where every team member can share their ideas, give and receive feedback, and collaborate openly. Each member will take ownership of their tasks and contribute to the continuous improvement of our processes and teamwork.

Signed by: [illegible]

Project Schedule

Sprint 0: Understanding the Project (Pre-Development Phase)

Goal: Gather requirements, define project scope, and identify key user needs.

Activities:

Understand the scope of immigration document handling (I-20, I-94).

Research relevant immigration rules and policies.

Define key features for the Immigration AI Advisor (PDF upload, document parsing, personalized advice).

Set up the development environment and select tech stack (Python, Flask, AI framework).

Initial team roles assignment.

BORDERLESS.AI

Project Schedule

Sprint 1: Basic PDF Upload & Parsing

Goal: Build and demonstrate the basic functionality of the Immigration AI Advisor.

Tasks:

Implement PDF upload functionality via the web UI.

Develop code for reading and extracting basic information from I-20 PDFs (e.g., Name, DoB).

Establish Flask as the backend for handling requests.

Test and verify that the I-20 upload and parsing process is functional.

Demo the working upload and basic data extraction.

Project Schedule

Sprint 2: **Data Fields Expansion & AI Component**

Goal: Extend the document parsing capabilities and introduce basic AI-driven advice.

Tasks:

Integrate additional data fields from I-20 (e.g., School Name, Program Dates)

Implement basic AI functionality to begin providing personalized advice

(e.g., identifying common visa statuses).

Ensure the system is flexible enough to handle more complex parsing as needed.

Test for accuracy across multiple types of documents (I-20, I-94).

Project Schedule

Sprint 3: AI Component Enhancement

Goal: Improve the AI’s capability to provide more complex immigration advice.

Tasks:

Refine the AI logic to handle more nuanced queries based on immigration policies.

Add more advanced decision-making capabilities to the AI component.

Enhance the AI model's ability to analyze and cross-reference multiple document types.

Test the AI’s performance and ensure it scales well with increased complexity.

Prepare the Immigration AI Advisor for final testing and potential deployment.

Personas

Name: Sarah Johnson

Age: 35

Role: International Student Adviser

Background:

Sarah has been an adviser for 8 years, handling hundreds of students each semester. She's responsible for guiding them through immigration processes, ensuring compliance with visa regulations, and managing I-20 and I-94 forms.

Challenges:

Overwhelmed by repetitive tasks (e.g., checking I-20, I-94 documents). Struggles to keep up with changing immigration rules. Limited time for personalized student interactions due to workload.

Goals:

Automate routine tasks to reduce workload. Improve accuracy and efficiency in advising. Free up time for more complex student cases.

How Immigration AI Advisor Helps:

Automates document review (I-20, I-94).

Provides preliminary, personalized immigration advice.

Reduces manual workload, enabling Sarah to focus on complex cases.

BORDERLESS.AI

Personas

Name: Aakken Lee

Age: 22

Role: International Student, 3rd-year undergraduate

Background:

Aakken is studying Computer Science and is on an F-1 visa. He regularly deals with immigration paperwork like I-20s and I-94s, and needs to stay compliant with visa regulations to maintain his legal status in the U.S.

Challenges:

Needs quick, accurate advice on maintaining visa status. Often faces delays due to overburdened advisers. Uncertainty about document requirements or policy changes causes stress.

Goals:

Access immediate immigration information and advice. Avoid potential issues with immigration status.

How Immigration AI Advisor Helps:

Provides instant, accurate responses to common immigration queries.

Offers clear explanations of I-20 and I-94 documents.

Reduces reliance on advice availability for basic information.

BORDERLESS.AI

Personas

Name: Dr. Maria Rodriguez

Age: 50

Role: VP of Office of Global Services

Background:

Dr. Rodriguez's role focuses on ensuring compliance with immigration regulations while enhancing student support and satisfaction. She's responsible for strategic planning, resource allocation, and policy implementation.

Challenges:

Ensuring the institution stays compliant with evolving immigration laws. Balancing limited resources with the growing needs of international students.

Goals:

Improve efficiency to better serve students. Use data and automation to ensure compliance with immigration rules while respect the limited budget.

How Immigration AI Advisor Helps:

Streamlines immigration document processing, freeing up adviser resources.

Enhances support for international students, improving satisfaction and retention.

MVP

Description:

The MVP of the Immigration AI Advisor focuses on delivering essential functionality to assist international students with their immigration-related needs. The MVP allows users to upload their I-20 document, and the system extracts key information such as their Name and Date of Birth. This basic document parsing provides immediate value by automating a critical task, helping students quickly access and verify important information from their immigration paperwork.

Key Features of the MVP:

PDF Upload: Users can upload an I-20 form through a web interface.

Data Extraction: The system automatically extracts and displays basic information (e.g., Name, Date of Birth).

Usability: Users can interact with the system, receive valuable information instantly, and ensure their immigration documents are processed accurately.

Value to Users:

The MVP solves a practical problem for students by saving time and reducing the complexity of manually reading their immigration documents. It provides immediate utility while laying the groundwork for future enhancements, such as more detailed document parsing and AI-based immigration advice.

Technologies

Languages & Frameworks:

Python, Flask, FastAPI, Pytest, ReactJS, HTML, CSS

Artificial Intelligence:

TensorFlow, PyTorch, Chat GPT4o mini, BART, Claude

Any additional technology we may discover and add as we move on to each sprints.

Algorithms

Regular Expressions/Pattern Matching

To recognize structure of, and extract information from, I-20 formatted forms

ChatGPT 4o mini/ BART/ Claude

LLM's (Large Language Models), minimally trained language models, to provide the basis of a custom-trained, immigration-oriented "knowledge base"

TensorFlow/ PyTorch:

Might be used for statistical/quantitative analysis and transformation, in order to augment LLMs' capabilities

Diagrams

Architecture Diagrams

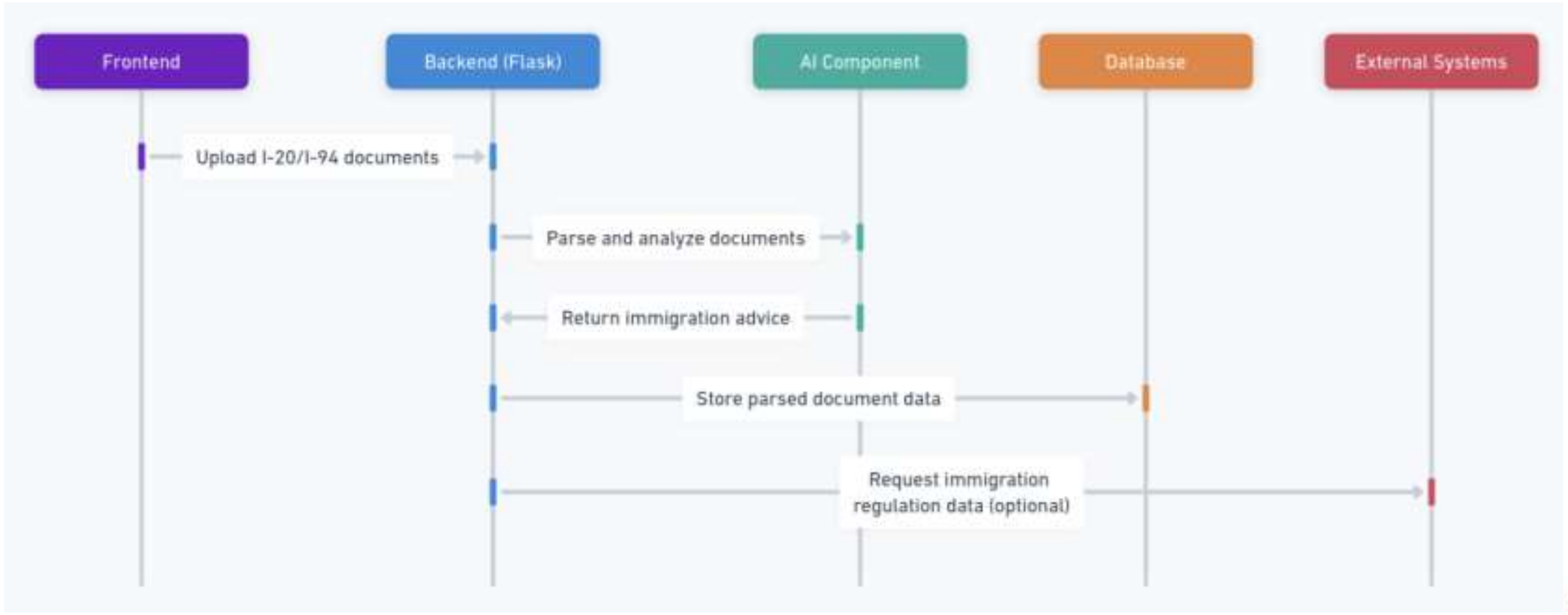
Context Diagram

ER Diagram

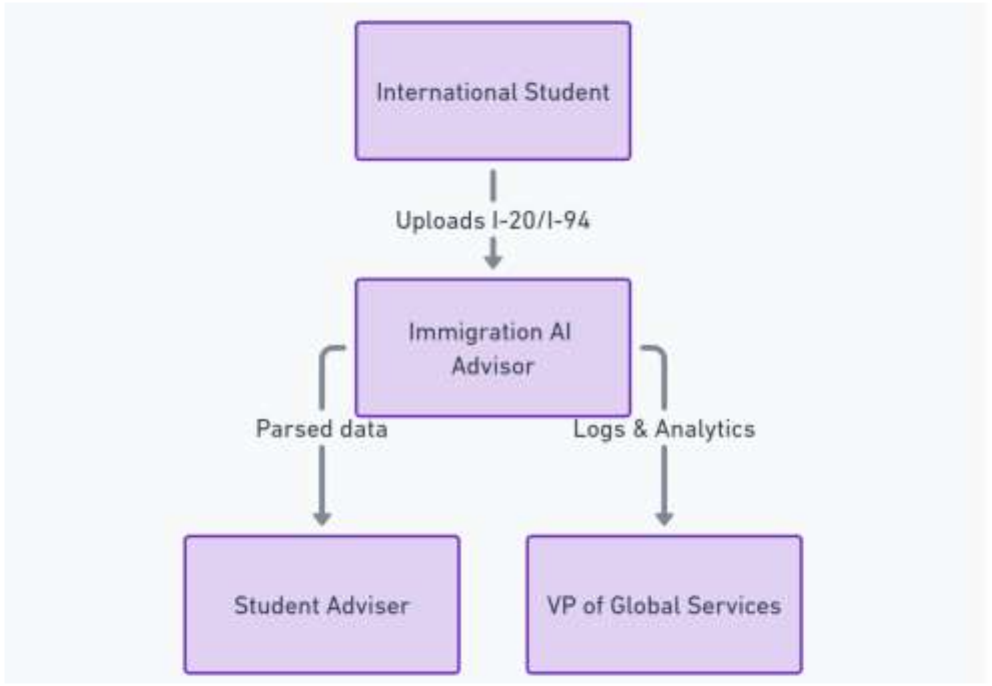
Sequence Diagram

State Diagrams

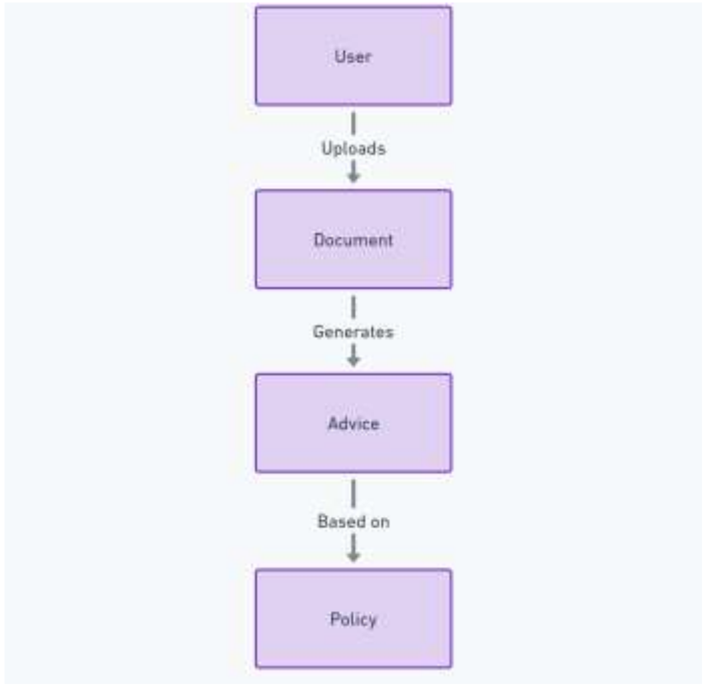
Architecture Diagram


Frontend
Backend (Flask)
AI Component
Database
External Systems
Upload I-20/I-94 documents
Parse and analyze documents
Return immigration advice
Store parsed document data
Request immigration regulation data (optional)

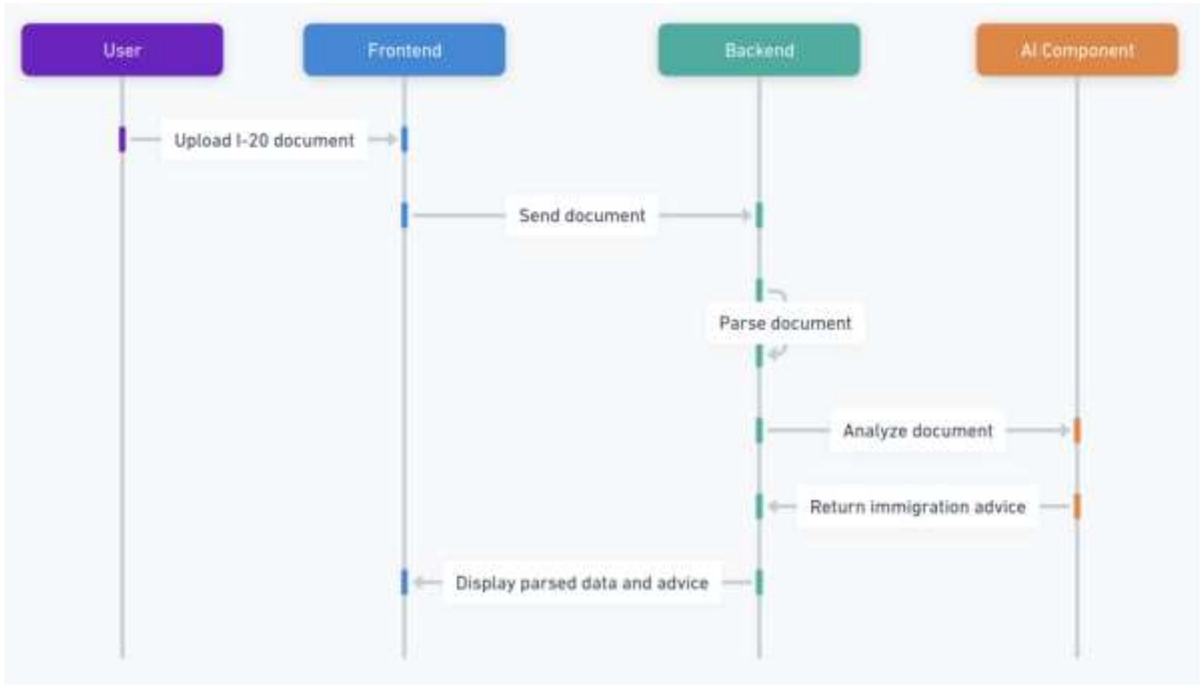
Context Diagram



ER Diagram (Entity-Relationship Diagram)

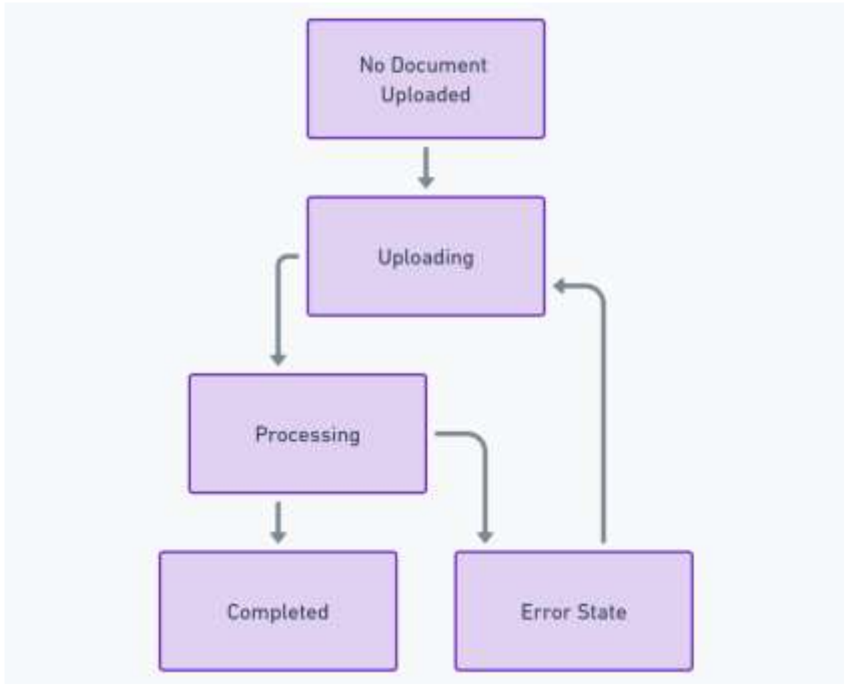


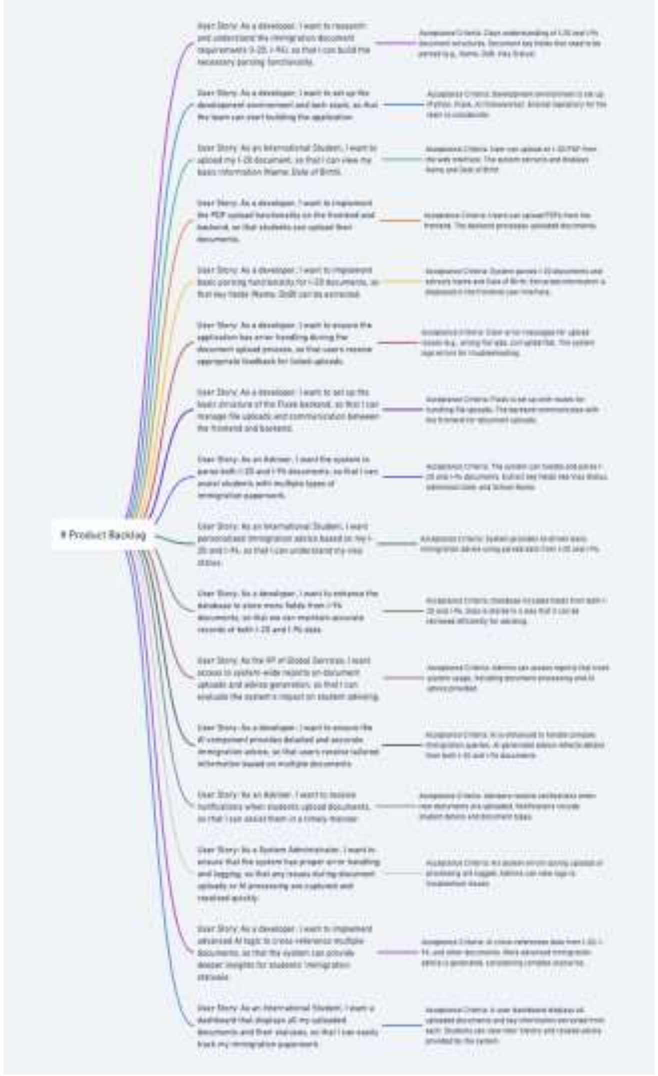
Sequence Diagram



BORDERLESS.AI

State Diagram





Product Backlog

- User Story: As a developer, I want to research and understand the immigration document requirements (I-20, I-94), so that I can build the necessary parsing functionality.
  - Acceptance Criteria: Clear understanding of I-20 and I-94 document structures. Document key fields that need to be parsed (e.g., Name, DoB, Visa Status).
- User Story: As a developer, I want to set up the development environment and tech stack, so that the team can start building the application.
  - Acceptance Criteria: Development environment is set up (Python, Flask, AI Frameworks). [illegible] repository for the team to collaborate.
- User Story: As an International Student, I want to upload my I-20 document, so that I can view my basic information (Name, Date of Birth).
  - Acceptance Criteria: User can upload an I-20 PDF from the web interface. The system extracts and displays Name and Date of Birth.
- User Story: As a developer, I want to implement the PDF upload functionality on the frontend and backend, so that students can upload their documents.
  - Acceptance Criteria: Users can upload PDFs from the frontend. The backend processes uploaded documents.
- User Story: As a developer, I want to implement basic parsing functionality for I-20 documents, so that key fields (Name, DoB) can be extracted.
  - Acceptance Criteria: System parses I-20 documents and extracts Name and Date of Birth. Extracted information is displayed in the frontend user interface.
- User Story: As a developer, I want to ensure the application has error handling during the document upload process, so that users receive appropriate feedback for failed uploads.
  - Acceptance Criteria: Clear error messages for upload issues (e.g., wrong file type, corrupted file). The system logs errors for troubleshooting.
- User Story: As a developer, I want to set up the basic structure of the Flask backend, so that I can manage file uploads and communication between the frontend and backend.
  - Acceptance Criteria: Flask is set up with routes for handling file uploads. The backend communicates with the frontend for document uploads.
- User Story: As an Advisor, I want the system to parse both I-20 and I-94 documents, so that I can assist students with multiple types of immigration paperwork.
  - Acceptance Criteria: The system can handle and parse I-20 and I-94 documents. Extract key fields like Visa Status, Admission Date, and School Name.
- User Story: As an International Student, I want personalized immigration advice based on my I-20 and I-94, so that I can understand my visa status.
  - Acceptance Criteria: System provides AI-driven basic immigration advice using parsed data from I-20 and I-94.
- User Story: As a developer, I want to enhance the database to store more fields from I-94 documents, so that we can maintain accurate records of both I-20 and I-94 data.
  - Acceptance Criteria: Database includes fields from both I-20 and I-94. Data is stored in a way that it can be retrieved efficiently for advising.
- User Story: As the VP of Global Services, I want access to system-wide reports on document uploads and advice generation, so that I can evaluate the system's impact on student advising.
  - Acceptance Criteria: Admins can access reports that track system usage, including documents processing and AI advice provided.
- User Story: As a developer, I want to ensure the AI component provides detailed and accurate immigration advice, so that users receive tailored information based on multiple documents.
  - Acceptance Criteria: AI is enhanced to handle complex immigration queries. AI-generated advice reflects details from both I-20 and I-94 documents.
- User Story: As an Advisor, I want to receive notifications when students upload documents, so that I can assist them in a timely manner.
  - Acceptance Criteria: Advisors receive notifications when new documents are uploaded. Notifications include student details and document types.
- User Story: As a System Administrator, I want to ensure that the system has proper error handling and logging, so that any issues during document uploads or AI processing are captured and resolved quickly.
  - Acceptance Criteria: All system errors during uploads or processing are logged. Admins can view logs to troubleshoot issues.
- User Story: As a developer, I want to implement advanced AI logic to cross-reference multiple documents, so that the system can provide deeper insights for students' immigration statuses.
  - Acceptance Criteria: AI cross-references data from I-20, I-94, and other documents. More advanced immigration advice is generated, considering complex scenarios.
- User Story: As an International Student, I want a dashboard that displays all my uploaded documents and their statuses, so that I can easily track my immigration paperwork.
  - Acceptance Criteria: A user dashboard displays all uploaded documents and key information extracted from each. Students can view their history and related advice provided by the system.

Sprint 1 Backlog
Stories or Tasks committed for this Sprint

User Story: As an International Student, I want to upload my I-20 document, so that I can view my basic information (Name, Date of Birth).
Acceptance Criteria:
User can upload an I-20 PDF from the web interface.
The system extracts and displays Name and Date of Birth.
Story Points: 3

User Story: As a developer, I want to implement the PDF upload functionality on the frontend and backend, so that students can upload their documents.
Acceptance Criteria:
Users can upload PDFs from the frontend.
The backend processes uploaded documents.
Story Points: 5

User Story: As a developer, I want to implement basic parsing functionality for I-20 documents, so that key fields (Name, DoB) can be extracted.
Acceptance Criteria:
System parses I-20 documents and extracts Name and Date of Birth.
Extracted information is displayed in the user interface.
Story Points: 8

BORDERLESS.AI

Sprint 1 Backlog
Stories or Tasks committed for this Sprint

User Story: As a developer, I want to ensure the application has error handling during the document upload process, so that users receive appropriate feedback for failed uploads.
Acceptance Criteria:
Clear error messages for upload issues (e.g., wrong file type, corrupted file).
The system logs errors for troubleshooting.
Story Points: 3

User Story: As a developer, I want to set up the basic structure of the Flask backend, so that I can manage file uploads and communication between the frontend and backend.
Acceptance Criteria:
Flask is set up with routes to handle file uploads.
The backend communicates with the frontend for document uploads.
Story Points: 5

Total Story Points for Sprint 1: 24 Story Points

Sprint 1 Backlog Test Cases

User Story: As an International Student, I want to upload my I-20 document, so that I can view my basic information (Name, Date of Birth).
Story Points: 3
Test Cases:
Test that a valid I-20 PDF is successfully uploaded.
Verify that the system correctly extracts the Name and Date of Birth from the document.
Ensure that the extracted data is accurately displayed in the user interface.
Test how the system behaves when an invalid document is uploaded (e.g., incorrect format, corrupted file).

User Story: As a developer, I want to implement the PDF upload functionality on the frontend and backend, so that students can upload their documents.
Story Points: 5
Test Cases:
Test the PDF upload functionality from the frontend.
Verify that the backend successfully receives and processes uploaded files.
Test for the correct response when an unsupported file type is uploaded.
Check that files larger than the size limit trigger an appropriate error message.

BORDERLESS.AI

Sprint 1 Backlog
Test Cases

User Story: As a developer, I want to implement basic parsing functionality for I-20 documents, so that key fields (Name, DoB) can be extracted.
Story Points: 8
Test Cases:
Test the system's ability to parse different I-20 PDFs (valid variations).
Verify the accuracy of the extracted Name and Date of Birth.
Ensure that extracted data is displayed correctly in the UI.
Test how the parser handles incomplete or incorrectly formatted I-20s.

User Story: As a developer, I want to ensure the application has error handling during the document upload process, so that users receive appropriate feedback for failed uploads.
Story Points: 3
Test Cases:
Test the system's response to uploading unsupported file types.
Verify that corrupted or invalid files trigger appropriate error messages.
Ensure that error messages are user-friendly and informative.
Check that the system logs errors appropriately for further debugging.

Sprint 1 Backlog
Test Cases

User Story: As a developer, I want to set up the basic structure of the Flask backend, so that I can manage file uploads and communication between the frontend and backend.
Story Points: 5
Test Cases:
Test that Flask is properly set up and handles HTTP requests correctly.
Verify that the backend routes handle document upload requests successfully.
Check that the backend returns appropriate success or error messages to the frontend.
Test communication between the frontend and backend during the file upload process.

Sprint 1 Backlog Stories completed

User Story: As an International Student, I want to upload my I-20 document, so that I can view my basic information (Name, Date of Birth).
Acceptance Criteria Completed:
User can successfully upload an I-20 PDF from the web interface.
The system extracts and displays the Name and Date of Birth from the uploaded document.
Status: Completed

User Story: As a developer, I want to implement the PDF upload functionality on the frontend and backend, so that students can upload their documents.
Acceptance Criteria Completed:
Users can upload PDFs from the frontend interface.
Backend successfully processes uploaded documents, handling the files appropriately.
Status: Completed

User Story: As a developer, I want to implement basic parsing functionality for I-20 documents, so that key fields (Name, DoB) can be extracted.
Acceptance Criteria Completed:
System parses I-20 documents and correctly extracts Name and Date of Birth.
Extracted information is displayed in the frontend user interface.
Status: Completed

Sprint 1 Backlog
Stories completed

User Story: As a developer, I want to ensure the application has error handling during the document upload process, so that users receive appropriate feedback for failed uploads.
Acceptance Criteria Completed:
Clear error messages are displayed for invalid uploads (e.g., wrong file types, corrupted files).
The system logs errors for troubleshooting purposes.
Status: Completed

User Story: As a developer, I want to set up the basic structure of the Flask backend, so that I can manage file uploads and communication between the frontend and backend.
Acceptance Criteria Completed:
Flask is fully set up with routes for handling file uploads.
Backend and frontend communicate successfully, with files uploading and processing properly.
Status: Completed

Total Completed Stories: 5 Stories (24 Story Points)
Total Stories not completed: 0

Metrics

Team Velocity - Sprint 1

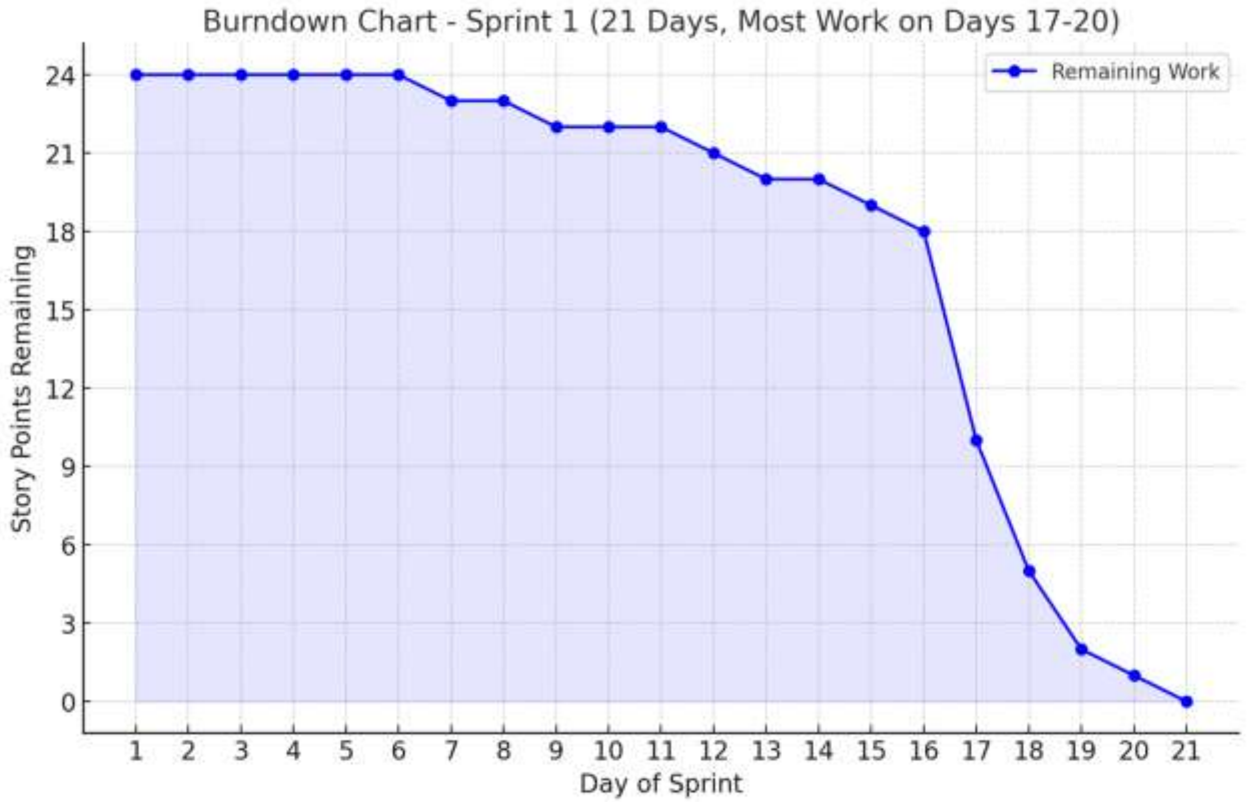
Details:

The team successfully completed 5 user stories, focusing on core functionalities like PDF uploads, document parsing, and error handling.

Story points are based on the complexity and effort required for each story, not the number of issues.

Total Story Points: 24 Story Points

Metrics



BORDERLESS.AI

Metrics

Completed/Committed Ratio:

24/24 = 100%

Retrospective

What Went Well?

Completed all tasks successfully.

On time for all scheduled meetings.

Gained a solid understanding of agile methodology.

Each team member delivered quality work.

Successfully designed and planned an MVP.

What Can Be Improved?

Overestimated tasks; could have accomplished more.

Overstated effort due to lack of capacity planning experience.

Incomplete velocity data, expected for a new team.

Could improve respecting deadlines.

Enhance workload distribution and collaboration.

Actionable Items:

Start recording demo videos 48 hours prior to deadlines.

Develop a structure and mechanisms for team accountability.

Improve capacity planning to better estimate task effort and timelines.

Continue refining workload distribution to ensure balanced collaboration.

Sprint 2 Planning and Commitment

Sprint 2: I-94 Parsing & AI Component

User Story: As an Adviser, I want the system to parse both I-20 and I-94 documents, so that I can assist students with multiple types of immigration paperwork.

Acceptance Criteria:

The system can handle and parse I-20 and I-94 documents.

Extract key fields like Visa Status, Admission Date, and School Name.

User Story: As an International Student, I want personalized immigration advice based on my I-20 and I-94, so that I can understand my visa status.

Acceptance Criteria:

System provides AI-driven basic immigration advice using parsed data from I-20 and I-94.

User Story: As a developer, I want to enhance the database to store more fields from I-94 documents, so that we can maintain accurate records of both I-20 and I-94 data.

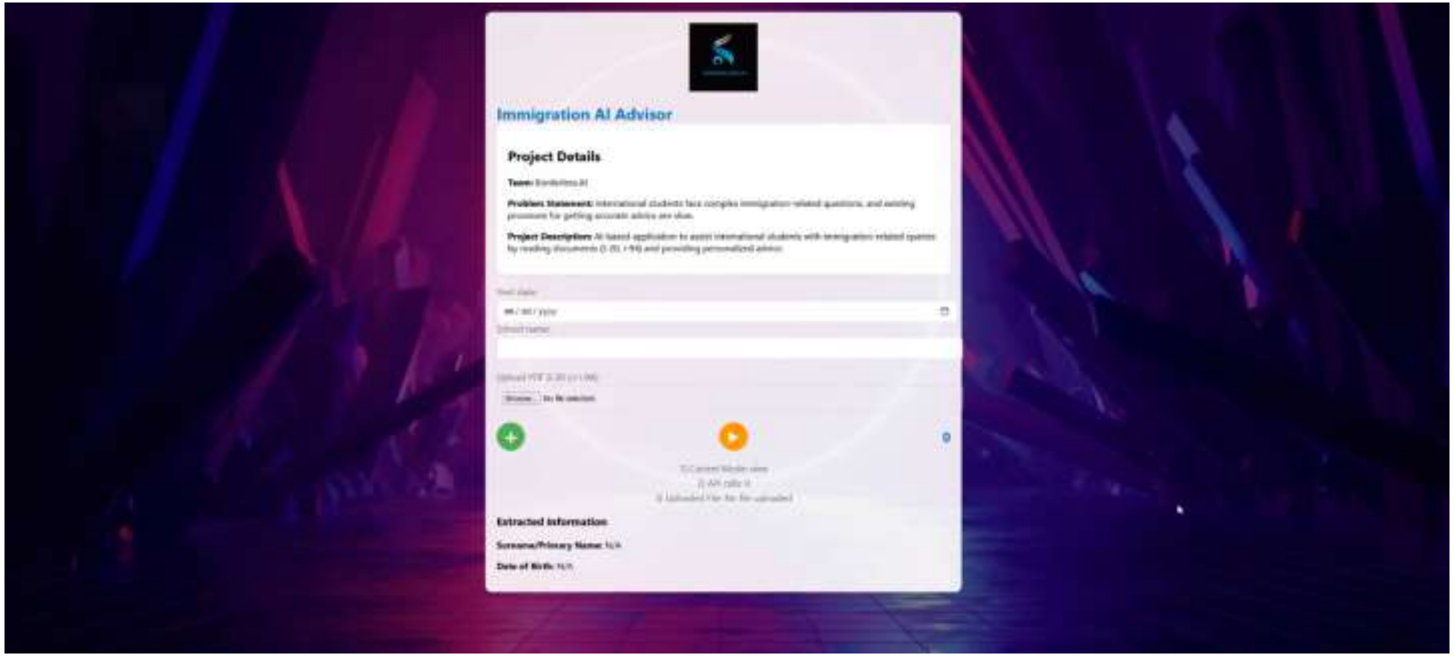
Acceptance Criteria:

Database includes fields from both I-20 and I-94.

Data is stored in a way that it can be retrieved efficiently for advising.

BORDERLESS.AI

Project Demo



Immigration AI Advisor

Project Details

Team: BorderlessAI

Problem Statement: International students face complex immigration-related questions, and existing processes for getting accurate advice are slow.

Project Description: AI-based application to assist international students with immigration-related queries by reading documents (I-20, I-94) and providing personalized advice.

[illegible] date:

mm / dd / yyyy

School name:

Upload PDF (I-20 or I-94):

Browse... No file selected.

1) Current Mode: view

2) API calls: 0

3) Uploaded File: No file uploaded

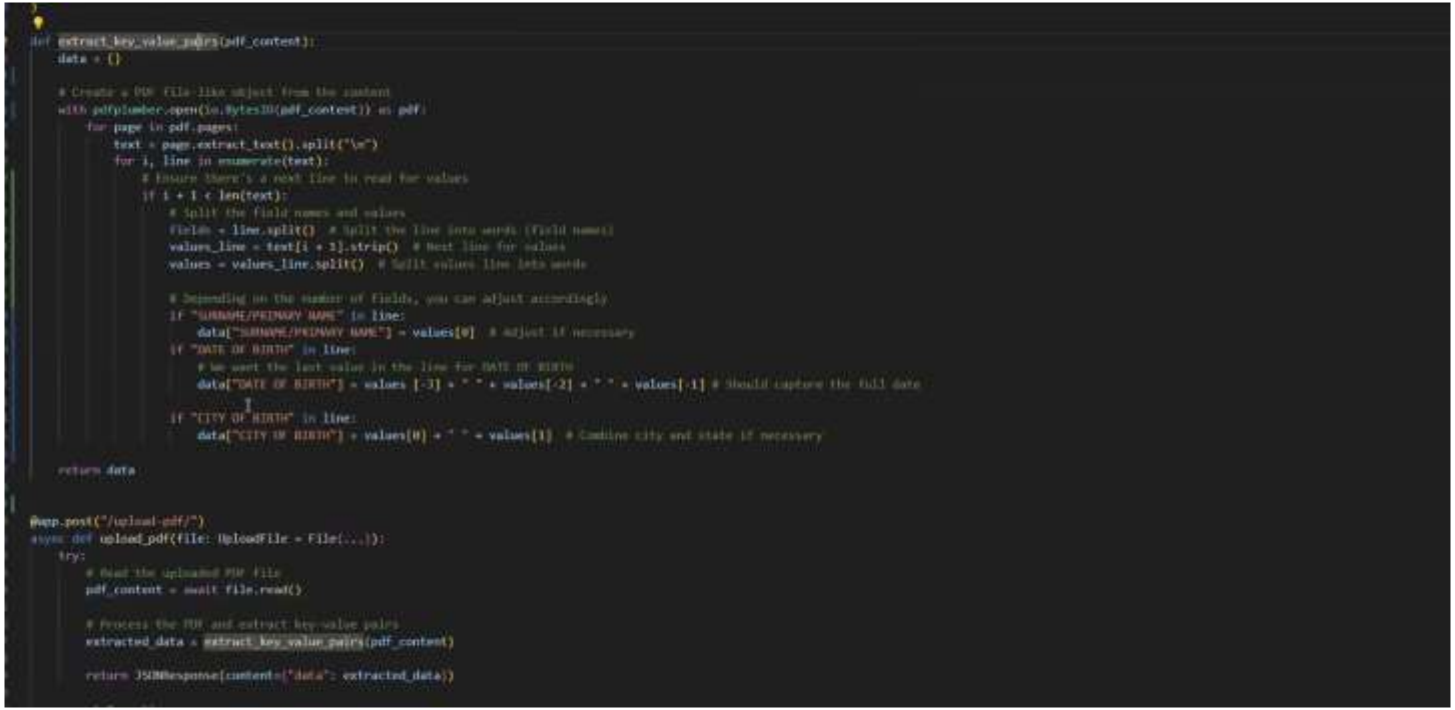
Extracted Information

Surname/Primary Name: N/A

Date of Birth: N/A

BORDERLESS.AI

Project Demo



```python
def extract_key_value_pairs(pdf_content):
    data = {}

    # Create a PDF file-like object from the content
    with pdfplumber.open(io.BytesIO(pdf_content)) as pdf:
        for page in pdf.pages:
            text = page.extract_text().split("\n")
            for i, line in enumerate(text):
                # Ensure there's a next line to read for values
                if i + 1 < len(text):
                    # Split the field names and values
                    fields = line.split()  # Split the line into words (field names)
                    values_line = text[i + 1].strip()  # Next line for values
                    values = values_line.split()  # Split values line into words

                    # Depending on the number of fields, you can adjust accordingly
                    if "SURNAME/PRIMARY NAME" in line:
                        data["SURNAME/PRIMARY NAME"] = values[0]  # Adjust if necessary
                    if "DATE OF BIRTH" in line:
                        # We want the last value in the line for DATE OF BIRTH
                        data["DATE OF BIRTH"] = values [-3] + " " + values[-2] + " " + values[-1] # Should capture the full date

                    if "CITY OF BIRTH" in line:
                        data["CITY OF BIRTH"] = values[0] + " " + values[1]  # Combine city and state if necessary

    return data


@app.post("/upload-pdf/")
async def upload_pdf(file: UploadFile = File(...)):
    try:
        # Read the uploaded PDF file
        pdf_content = await file.read()

        # Process the PDF and extract key-value pairs
        extracted_data = extract_key_value_pairs(pdf_content)

        return JSONResponse(content={"data": extracted_data})
```

Github Link

https://github.com/htmw/2024F-borderless.ai/wiki

Live Application Demo

THANK YOU!