Project Requirements Document v2

Project Title: Automated 3 Way Match (tentative)

Team Name: $-flow

Members:

- Millan Batra [Lead]
- Yoon Lee [Scribe]
- Dennis Fong
- Alexander Kang
- Alexander Yuen

Email:

millanbatra@umail.ucsb.edu
yoonlee@ucsb.edu
dfong@ucsb.edu
alexanderkang@ucsb.edu
atyuen@ucsb.edu

Intro

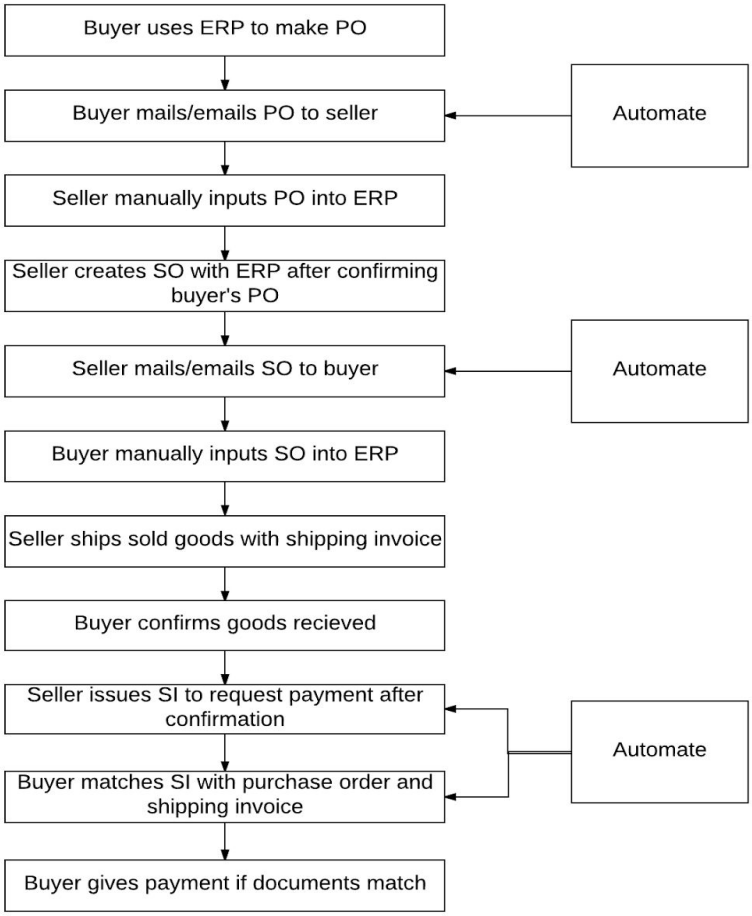
IMPORTANT TERMS

Purchase Order (PO):
Describes the price, quality, and quality of goods to be purchased.

Sales Order (SO):
Confirms the PO details.

Enterprice Resource Planner (ERP):
Creates SOs and POs based on a forecast

Sales Invoice (SI):
Describes price, quality, and quantiity of goods in the shipment and may include PO and/or SO identifiers.



Problem

The current purchase order pipeline imposes up to a 24 hour delay before the data is available online and has the potential for human error. This is due to the large amount of manual input required when attempting to match a purchase order, shipping order, and shipping invoice. This process continues to have many steps which introduce errors and create delays for planners, purchasers, and accountants.

Project Specifics

We will be creating a mobile app that will help reduce the errors and delays associated with the purchasing process. Our mobile application will allow users to take pictures of purchase orders, shipping orders, and shipping invoices and upload the relevant data to our system. Once uploaded, we will match the documents to one another and map this back to the customer's ERP system.

Innovation

This mobile application aims to reduce delays in the purchase order pipeline by providing a clean and efficient platform that automates document matching. Companies are stuck in the past passing around paperwork between companies and scanning documents into their ERP systems. Our service will be the bridge that helps bring companies into the present by automating the most inefficient parts of the purchase order pipeline.

Goals

The goal of $-Flow is to help alleviate the delay between the time an invoice is created and the time for it to be inputted into a company's ERP system. Our app will also have an OCR pipeline which will be able to identify and parse invoices when given an image. Then match related invoices and allow companies to retrieve this information from our database.

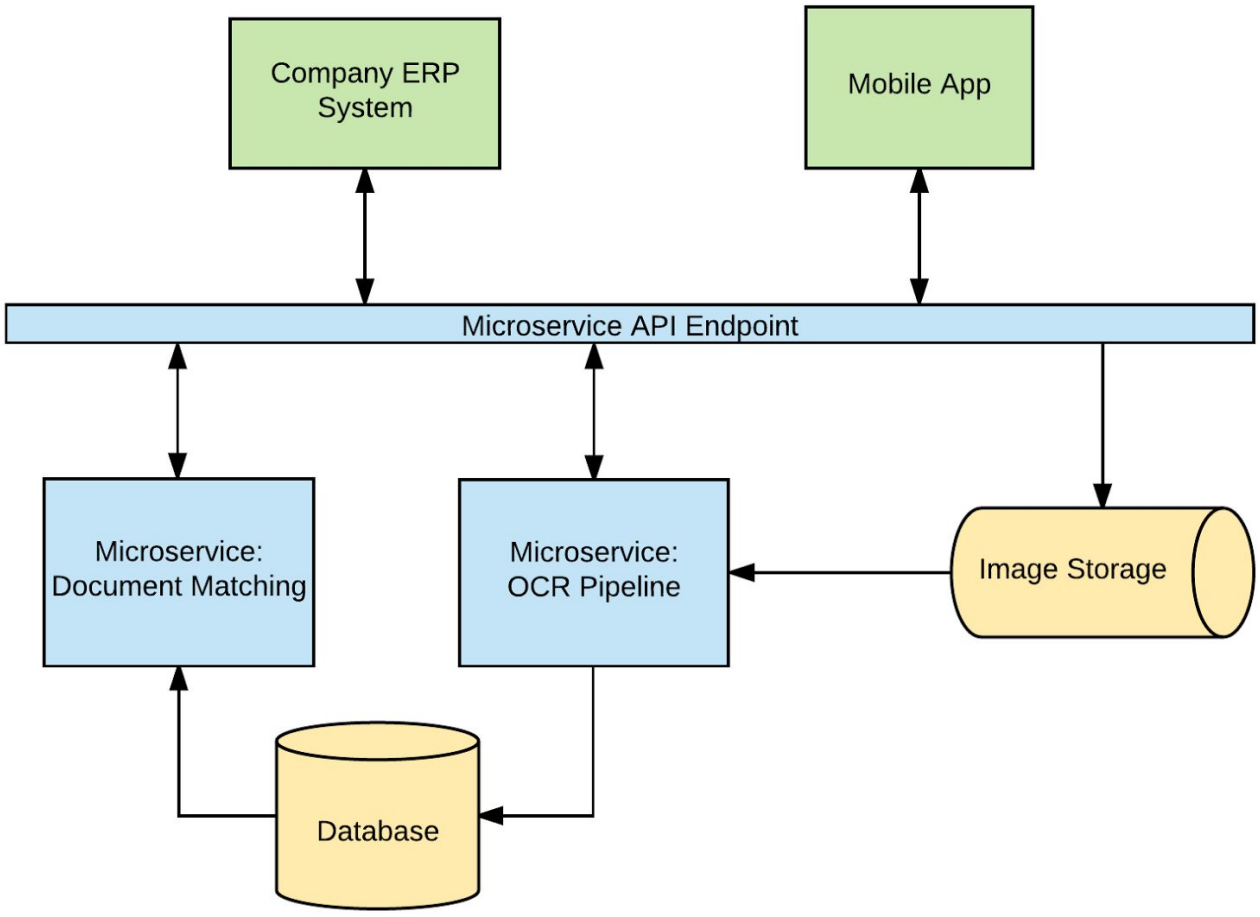
Background

There are no systems currently in place to address these inefficiencies. With this solution, we will help Elementum with their mission to transition the world from one based on excel spreadsheets and emails to one based solely on data.

Assumptions

The documents that we are parsing will contain deviants; each company will have multiple ways of expressing important data. Documents given to the system should be printed and contain no handwriting. We will not be expected to parse handwritten text. Upon opening the app, we assume the user's device will have strong connectivity to the internet and no interference. The user will already have an Elementum account and thus have access to the mobile application's service. Once document matching has completed, it is the responsibility of the company to grab the data from our system as we do not have access to their enterprise resource planners.

System Architecture



Mobile App

The mobile app will be the platform for taking pictures and matching shipping orders, purchase orders, and shipping invoices. Users can register under their respective companies and will be restricted to viewing those company's documents. The app will upload images to an AWS S3 bucket and make API calls to the RESTful web service to notify users whether the document has been matched or not. The mobile app is developed in React-Native allowing us to deploy our application to both Android and IOS.

Microservice: OCR Pipeline

The OCR pipeline will receive requests from the mobile app to parse documents that are stored in AWS S3. Once a request is received, an image is pulled down from AWS S3 and parsed. The pipeline will take care of preprocessing, text identification, and text parsing. Once the text is parsed, we will retrieve the relevant data from the parsed document and insert this into our MySQL database.

Microservice: Document Matching

After the documents have been parsed, the parsed data will be matched against a pre-existing document's data stored in our backend database. If the data in the documents is exactly the same, the backend will send an HTTP response to the mobile app to notify the user that the document has been match. Furthermore, companies will be able to hit the api endpoint with document data and receive all referenceable documents which match the search parameters.

System Requirements

1.1. Users can take pictures of invoices and decide whether or not to upload them

1.2.

a. Once a picture is uploaded, we store it in file storage system (AWS S3). Each image will have a corresponding entry, with its metadata, in our database (AWS RDS) to keep track of it.

b. A job will be queued for our OCR pipeline to parse the text from the image.

1.3. Our OCR pipeline will take jobs off our job queue and execute the OCR pipeline.

1.4. Once the text is parsed from the image, we store the parsed data in our database. (AWS RDS)

1.5. After parsing is done, we match the recently parsed image with other referenceable entries.

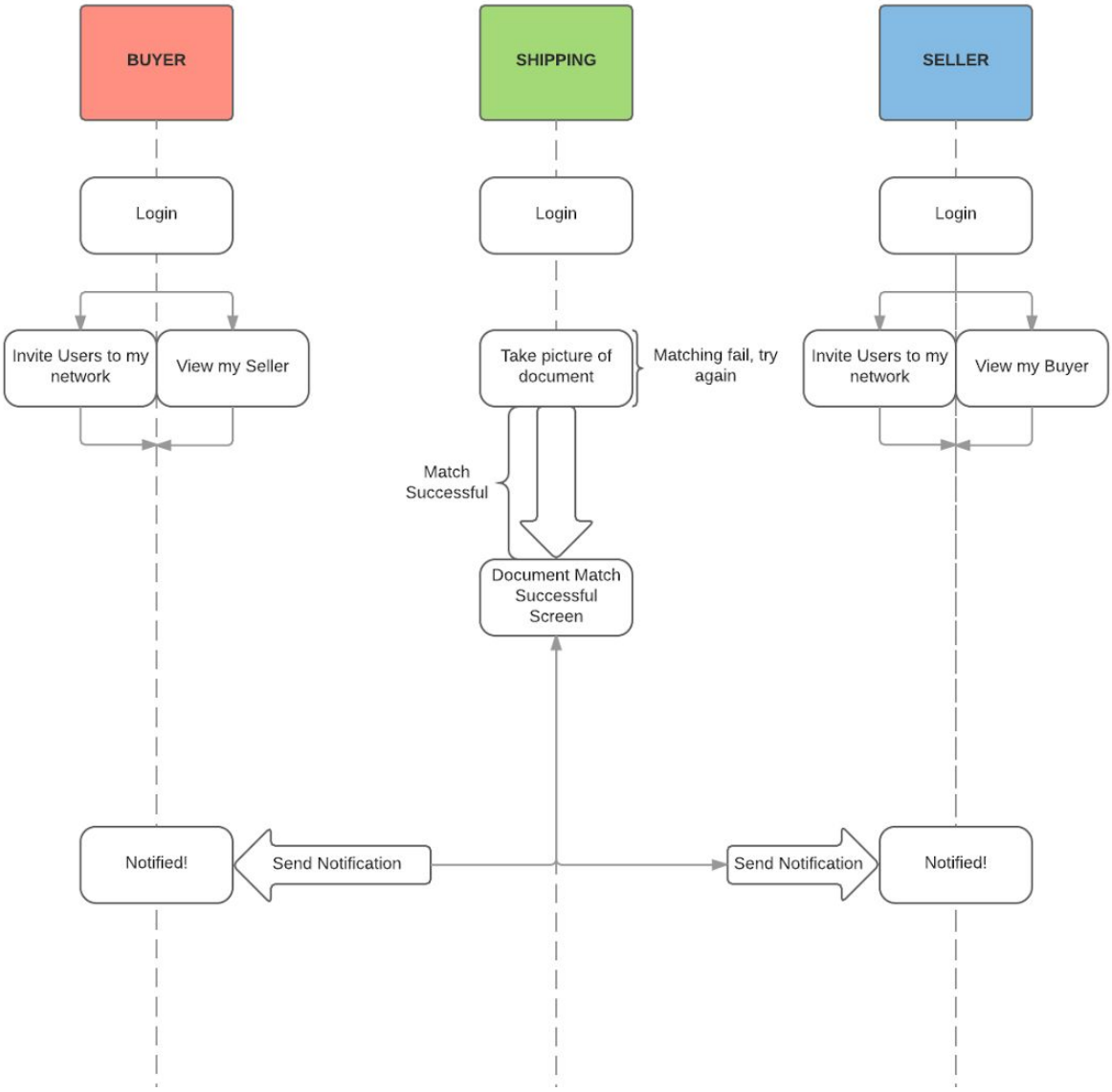
1.6. Waybill and invoice number is recorded and added into the database

1.7. We store the matched entries in our database.

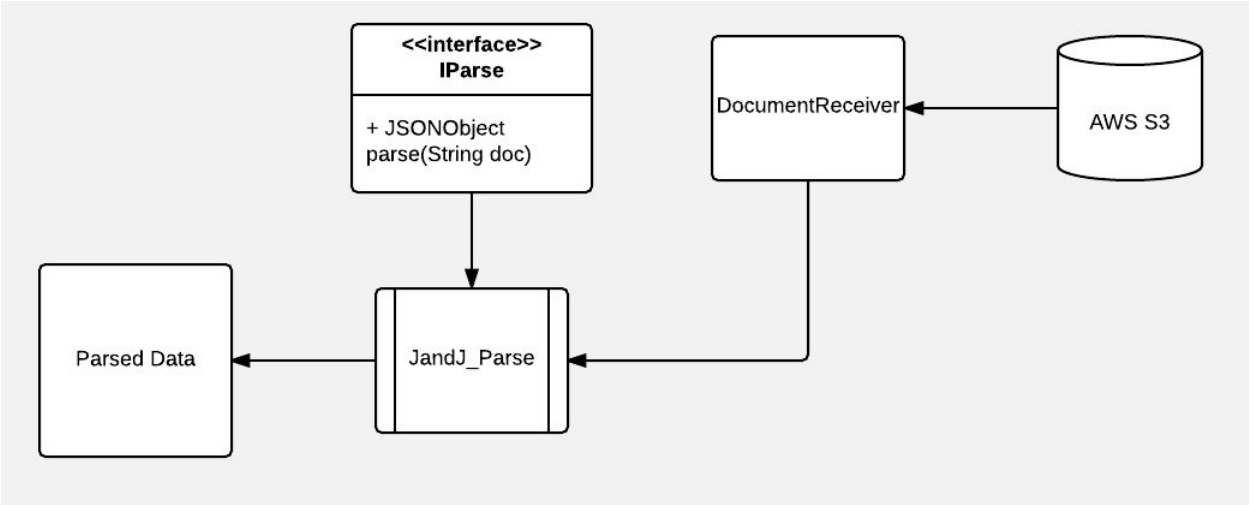
2.1. Users can request document matches for documents give a unique id.

Requirements

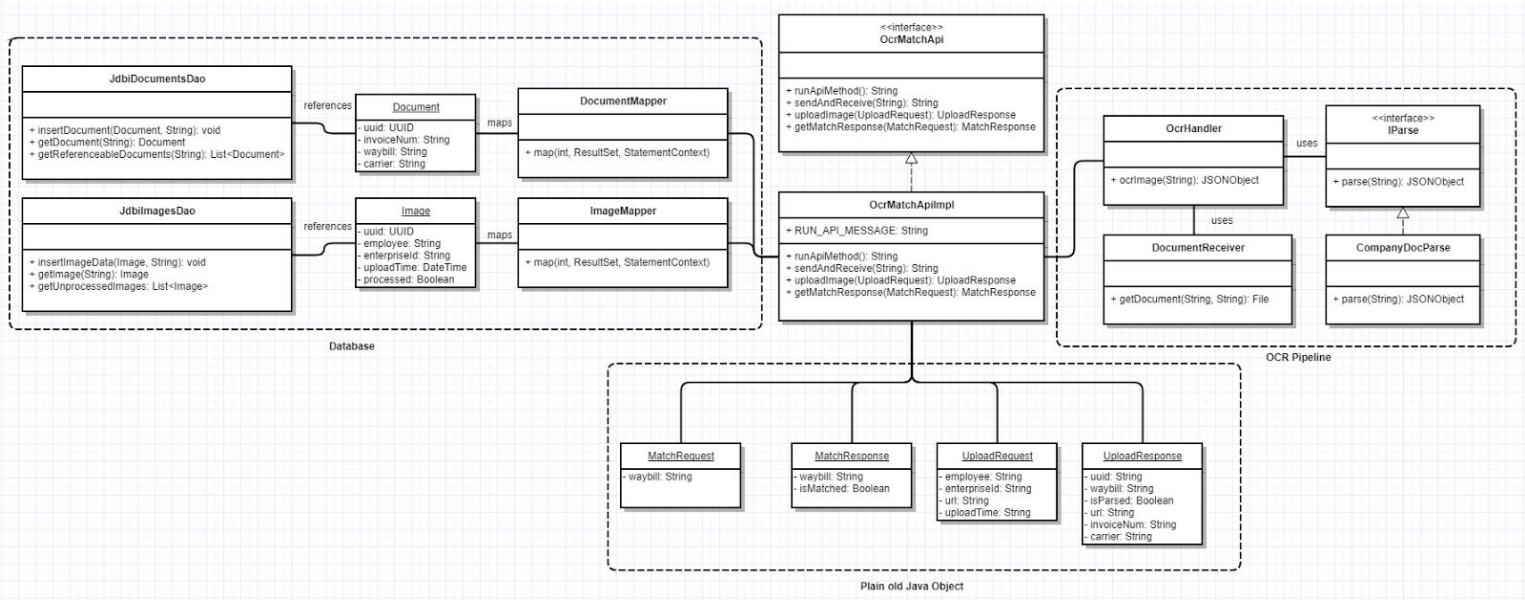
User Interaction and Designs



OCR Pipeline Interaction

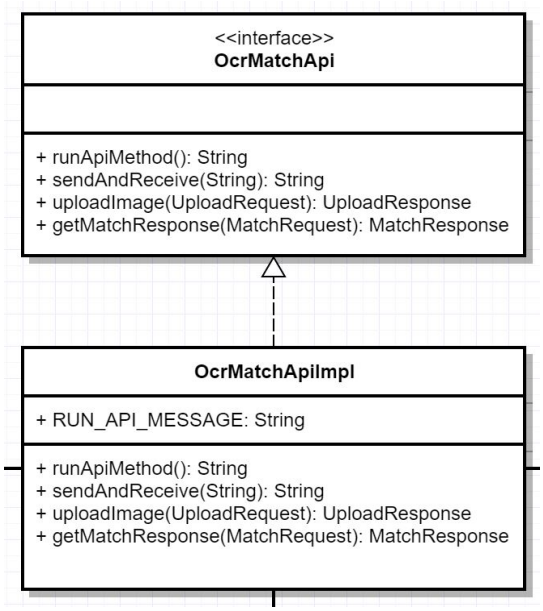


UML Class Diagram

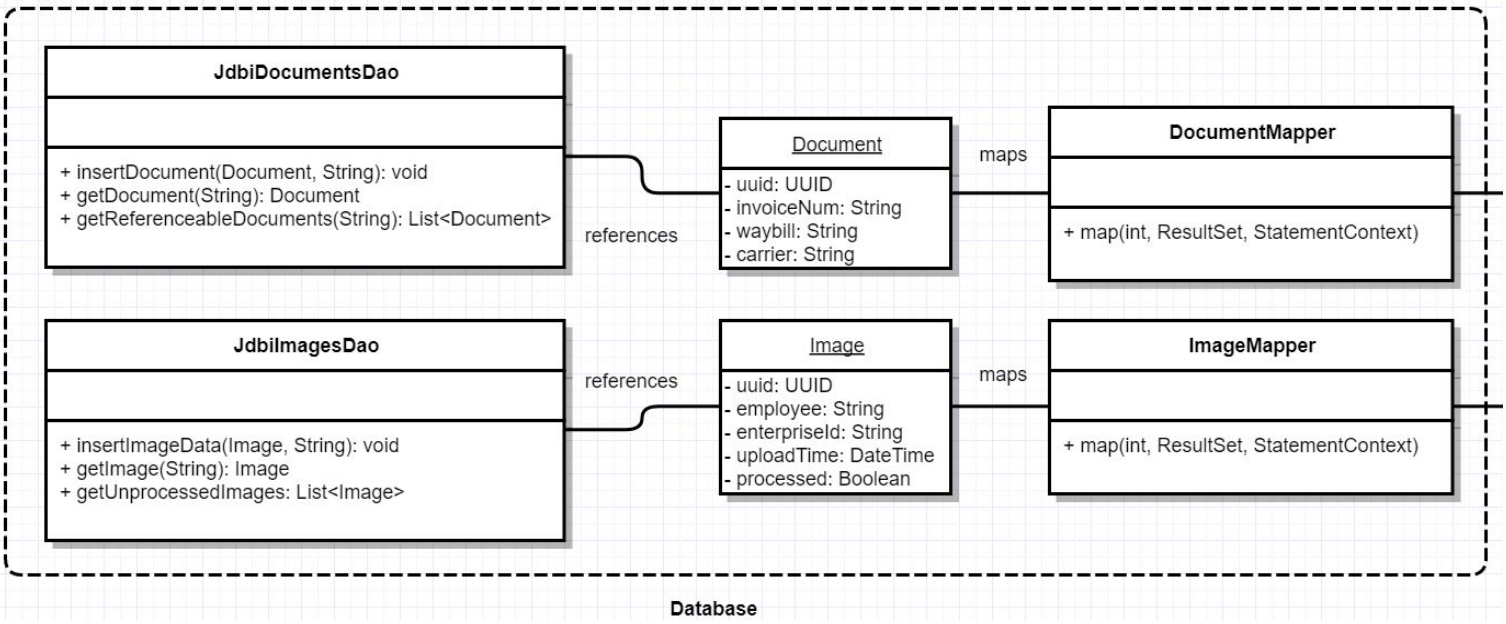


UML Class Diagram Broken Down

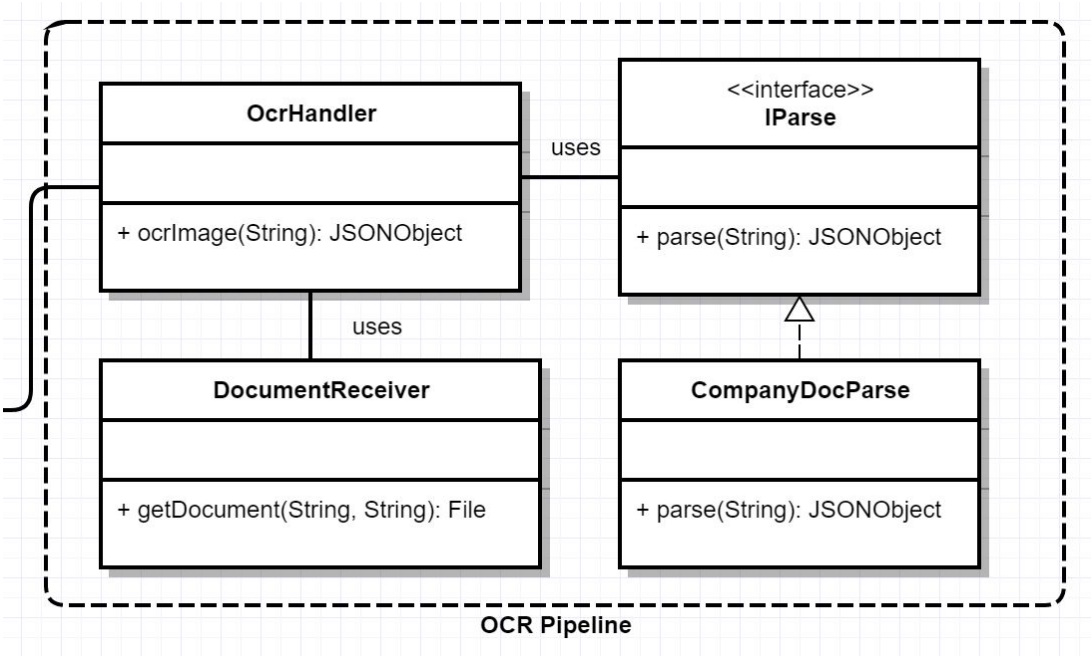
API



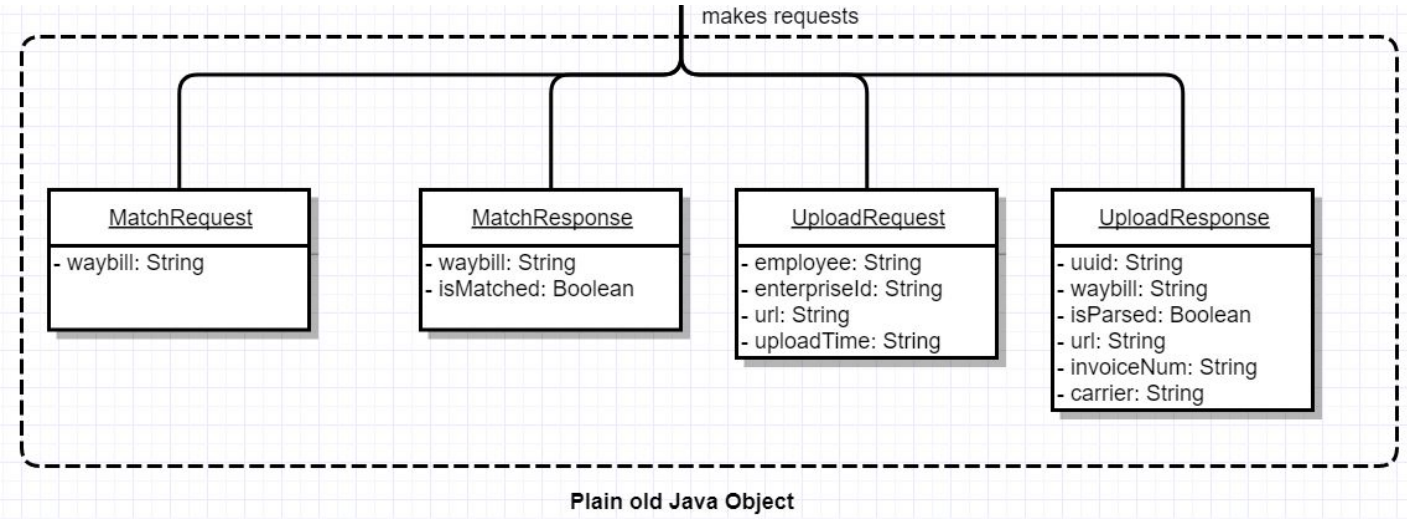
Database



OCR Pipeline



Plain Old Java Object



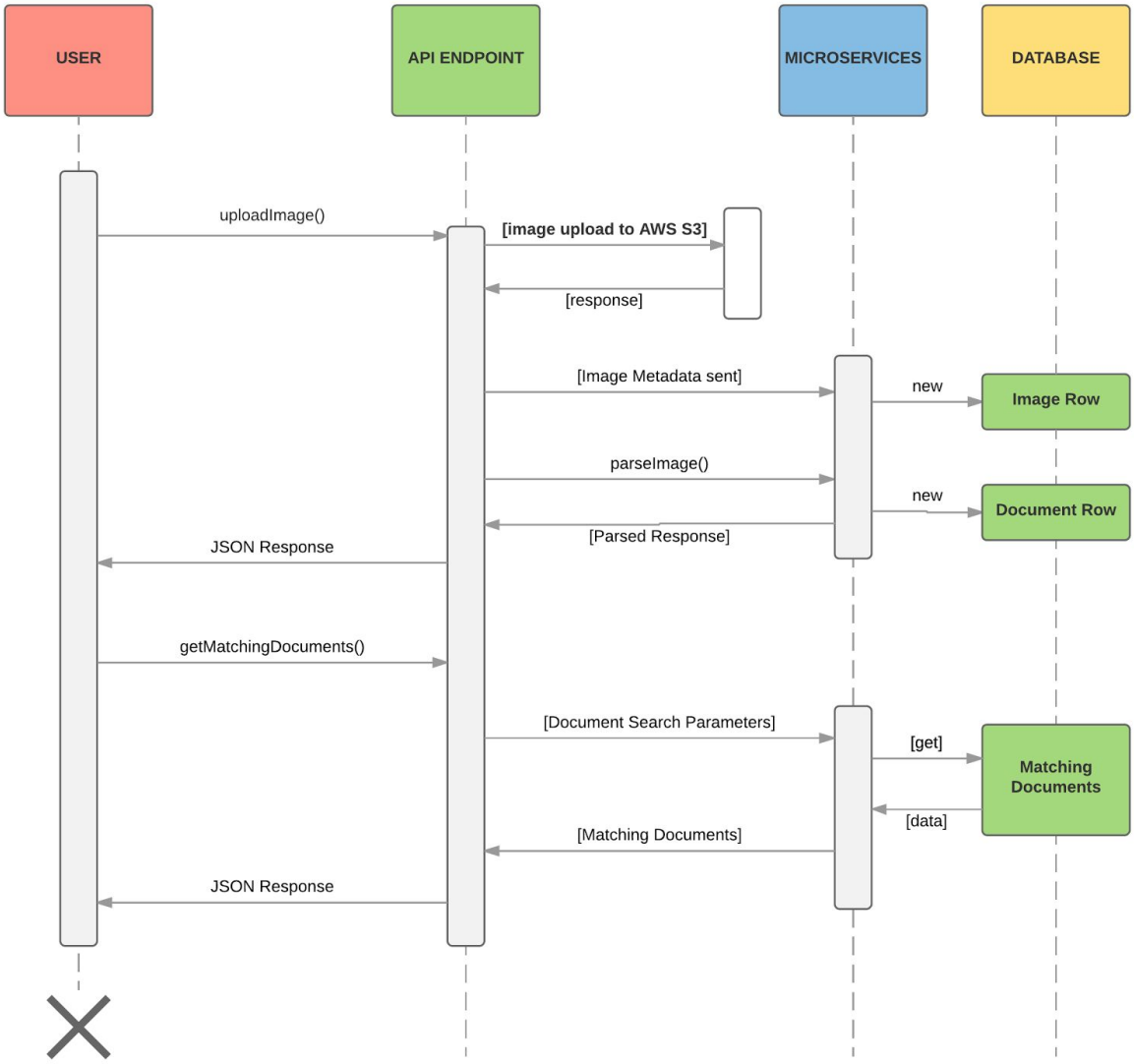
User Stories

	User Story	Acceptance Criteria
1.	As a user, I want to be able to navigate efficiently from the home screen to the settings or camera screen and back.	Buttons on the screen for navigation works as expected. It can also go back to the screen it was previously on.
2.	As a user, I want to be able to take a picture inside the app and view it.	Take a picture of a document given a screen where a user can take it. When the picture is taken, then the user can view it and delete it if the user doesn't like it.
3.	As a user, I want to be able to upload pictures taken.	Upload a picture given a picture and a user clicking upload. When a user wants to upload a picture, then we want to store it in a filesystem and store its metadata in a database.
4.	As a user, I want to be able to log in to the app.	Log in to the app. Given a login screen and a user giving username and password. On button click, user is logged in and routed to another page.
5.	As a company, I want to be able to match referenceable invoices so I can save time.	Many documents in the database. Given many invoices. When the picture is uploaded, the backend will handle matching the data within the picture to other data in the database.
6.	As a user, I want to be notified whether the picture is matched or not.	Prompt me to retake picture with tips on better picture e.g. better lighting, angle it perpendicular to the invoice. Picture is of good quality but there was no match - "contact supervisor"? Otherwise, display "Match approved" screen.
7.	As a user, I want all participating parties to be notified of the document match.	After a 'document match' event in the app, participating parties associated with the user will receive an e-mail notification with a timestamp and location.

8.	As a company, I want to be able to invite users to join my network.	Company can login to the app and on the 'invite' page, can fill out a form (e-mail address?) to grant authentication to a user signed up under that e-mail address.
9.	As a user, I want to be able to select between PDF and camera upload options for images.	The application successfully uploads an image/pdf/file from the camera/file system to AWS S3 bucket.
10.	As a buyer, I want to be able to select my sellers.	Given a screen that prompts the user to enter in the seller, the app will make a GET request for authenticating permission.
11.	As a seller, I want to be able to view my buyers (multiple).	A page in the app will show a list of buyers that the user can navigate through easily.
12.	As a developer, I want to be able to parse data from uploaded documents.	The OCR pipeline should parse the waybill number, invoice number(s), and any other relevant information from the documents uploaded.
13	As a developer, I want the OCR pipeline to activate when there's a new entry in the AWS database	The OCR pipeline is activated as soon as a new image is uploaded
14.	As a user, I want to be able to retrieve my documents' parsed data from the OCR pipeline.	After parsing the uploaded image, the OCR pipeline should send the parsed data back to the user. The user should confirm whether or not the data is correct.
15.	As a developer, I want to be able unit test our data access objects and mappers.	When any unit testing for the data layer occurs, a reusable setup should be used to set up the testing environment.
16.	As a developer, I want to be able to hit the backend endpoints with HTTP requests.	Given endpoints, a user should be able to send HTTP requests and receive responses.
17	As a developer, I want to be able to easily map data	Call a method which retrieves data from the database. After retrieving data from the database, the data is place

	from the database back to Java.	
18.	As a developer, I want to be able to store image metadata in Java.	After the API endpoint is hit with a POST request. Given the post request, a java object should be created containing the relevant data.
19.	As a developer, I want to be able to store parsed document data in a Java object.	After relevant data is parsed from the document, a java object should be created containing the parsed data. For example, it should contain waybill, invoice#, and carrier.
20.	As a developer, I want to be able to map data stored in Java classes to the database.	After a SQL query has been executed and the data has been returned, the data should be mapped into a java object.
21.	As a developer, I want to be able to abstract data into different entities.	For any data stored in a Image or Document class, there is a related table which stores the data in the database

Sequence Diagram



Appendix

Purchase Order (PO) - Document issued by the seller to the buyer that indicates types, quantities, and prices for purchased products/services
Sales Order (SO) - Document generated when a buyer purchases a product/service to confirm the purchase order details
Sales Invoice - A bill sent to the customer after a sale requesting for payment
Shipping invoice - Document that accompanies the sold goods when shipped. Lists price, quantity, and quality of goods and can also include PO and/or SO identifiers
Enterprise Resource Planner (ERP) - An ERP management information system integrates areas such as planning, purchasing, inventory, sales, marketing, finance and human resources.
Tesseract - Library used to perform OCR. Comes with packages for many languages but we will only be using english
OCR - Object Character Recognition - Process of recognizing characters in a given image
React Native - Framework that allows users to create universal mobile apps using Javascript
POJO (Plain Old Java Object) - Java object which can compiled under the jdk without any extensions
Continuous Integration - Process of frequently merging in developer work into a shared mainline
Continuous Deployment - Series of processes which allow code to be quickly and safely be deployed to production
AWS S3 - Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data
AWS RDS - Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud
Docker - an application build and deployment tool which allows code to be packaged and deployed in containers.
Jenkins - Server which helps automate the CI/CD pipeline.

Technologies Employed

Communications and Work Tools - JIRA, BitBucket, Slack
Document Parsing and OCR - Tesseract
Cloud Storage Tools - AWS S3, AWS RDS
Backend Development - Java
Mobile Development - React Native
CI/CD - Docker, Jenkins