FULL STACK INVOICING WITH IMAGE PROCESSING

Harsh Arora[1]
Department of Information Technology , Maharaja Agrasen Institute Of Technology, Delhi, India

Abstract : As mentioned in a former paper titled 'Automatic Invoicing using Image Processing'[1], most ecommerce companies have their receive to pay process as predominantly manual, leading to non-reliability of payments & delayed visibility for sellers and requirement of additional manpower for scaling up for buyers. With the correct image and pdf processing tools, it is possible to automate this process for more efficient and cost effective results. This has been already demonstrated. This research paper extends the automatic invoicing in a full stack environment to demonstrate how this will work in an environment with multiple components to achieve true end to end automatic invoicing. The idea is to save time, effort, and costs while eliminating human errors from the process. This paper uses the invoicing library discussed in a previous paper titled 'Automatic Invoicing using Image Processing'[1].

Keywords: Image Processing, Electronic invoicing, Android[5] app, React, Node.JS[5].

I. INTRODUCTION

Most ecommerce companies have their receive to pay process as predominantly manual, leading to non-reliability of payments & delayed visibility for sellers and requirement of additional manpower for scaling up for buyers. For a seller this involves high turn around time to know the money that the buyer is going to acknowledge. This in turn leads to reduction in time to respond to any deductions without impact to payment for the invoice and unpredictability in working capital planning.

Manual invoice processing forces a buyer to scale up in operations which in turn require scale up in manpower. Manual processing ensures that the cost of processing per invoice is high while being error prone.

Most ecommerce companies have their receive to pay process as predominantly manual, leading to non-reliability of payments & delayed visibility for sellers and requirement of additional manpower for scaling up for buyers.

This paper uses the invoicing library previously discussed to determine the viability of image processing in a full stack environment.

An android[5] app has been used where a user can click the picture of an invoice. This app sends the scanned invoice to a Node.JS[5] backend where this scanned copy is converted to JSON. This data, when once validated by the user, is saved to a mongoDB[6] database.

Structure of Paper

The paper is organized as follows: In Section 1, the introduction of the paper is provided along with the structure, important terms, objectives and overall description. In Section 2 we discuss related work. In Section 3 we have the complete information about the technology and stack that have been used to demonstrate end to end automated invoicing. Section 4 tells us about the methodology and the process description. Section 5 tells us about the future scope and concludes the paper with acknowledgement and references.

Objectives

The predominant invoice processing systems are either entirely manual or they follow a rigid single template system. Whether an individual is a buyer or a seller, this leads to a lot of inefficiencies and high costs.

This project aims to address some of the problems in current systems by greatly minimizing the human intervention in the process and thus reducing costs and errors. The aim is to ease the task of both the buyer and the seller.

II. RELATED WORK

There are numerous works that have been done related to image processing machine learning algorithms.

Harsh Arora[1] has investigated different image processing techniques and discussed a python library for the processing of invoices and converting it into relevant formats such as JSON, CSV etc.

J.E. Cross[2] has investigated methods to recover the maximum amount of available information from an image. Some radio frequency and optical sensors collect large-scale sets of spatial imagery data whose content is often obscured by fog, clouds, foliage and other intervening structures. Often, the obstruction is such as to render unreliable the definition of underlying images. Various mathematical operations used in image processing to remove obstructions from images and to recover reliable information were investigated, to include Spatial Domain Processing, Frequency Domain Processing, and non-Abelian group operations.

John C. Russ[3] has investigated techniques of image processing. These are operations that start with a grey scale (or color) image and return another grey scale image. The next chapter will deal with some additional techniques that operate on grey-scale images for purposes of locating feature edges in the context of isolating features for measurement.

J. M. White[4] and G. D. Rohrer, "Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction," in IBM Journal of Research and Development have researched on Two new, cost-effective thresholding algorithms for use in extracting binary images of characters from machine- or hand-printed documents.

However there has been little to no work put into the viability of image processing to achieve electronic automated invoicing.

III. TECHNOLOGIES USED

Numerous technologies have been used to develop this end to end system.

Node.JS[5]

Node.js[5] is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js[5] lets developers useJavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js[5] represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts. Node.js[5] distributed development project, governed by the Node.js[5] Foundation, is facilitated by the Linux Foundation's Collaborative Projects program. Corporate users of Node.js[5] software include GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP,Voxer, Walmart, and Yahoo!

MongoDB[6]

MongoDB[6] is an open source database management system (DBMS) that uses a document-oriented database model which supports various forms of data. It is one of numerous nonrelational database technologies which arose in the mid-2000s under the NoSQL banner for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB[6] architecture is made up of collections and documents.

Android SDK[7]

Android SDK provides tools to create, compile, test and publish applications that run on the Android operating system. Android apps can be written using Kotlin and Java with help of android sdk.

React[7]

React[7] (also known as React.js or ReactJS) is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React[7] can be used as a base in the development of single-page or mobile applications, as it's optimal only for its intended use of being the quickest method to fetch rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex React[7] applications usually require the use of additional libraries for state management, routing, and interaction with an API. Reac[7]t was created by Jordan Walke, a software engineer at Facebook. He was influenced by XHP, an HTML component framework for PHP. It was first deployed on Facebook's newsfeed in 2011 and later on Instagram.com in 2012. It was open-sourced at JSConf US in May 2013. React[7] Native, which enables native Android[5], iOS, and UWP development with React, was announced at Facebook's React.js Conf in February 2015 and open-sourced in March 2015. On April 18, 2017, Facebook announced React[7] Fiber, a new core algorithm of React[7] framework library for building user interfaces. React[7] Fiber was to become the foundation of any future improvements and feature development of the React[7] framework. On April 19, 2017, react[7] 360 V1.0.0 was released to the public. This allowed developers with experience using react[7] to jump into VR development.

V. METHODOLOGY

The objective of this project is to allow an efficient, cost effective and a convenient invoicing system which can adapt to different types of invoice templates.

The idea is to get the vendor to click a picture of the invoice using the android[5] app. This image is then uploaded to a cdn and the link is sent to our backend server. Then in the invoice processing library, raw text from the invoice can be extracted. The data is then converted to JSON.

The backend server sends the image as a request to the invoicing library and receives the JSON as response.

This data is validated by the vendor and then saved to a database.

Once the digitised data has been saved to the database, it can be accessed via the react[7] web-panel.

The project is aimed at building a flexible invoicing system which can precisely match content PDF files, easily match line items and tables and automate the entire invoicing process for any major organization. The proposed system will have the ability to obtain all relevant information from the document with 100% accuracy while ensuring speed and reliability.

Process Description

The following high level diagram makes it easier to understand how we proceed.

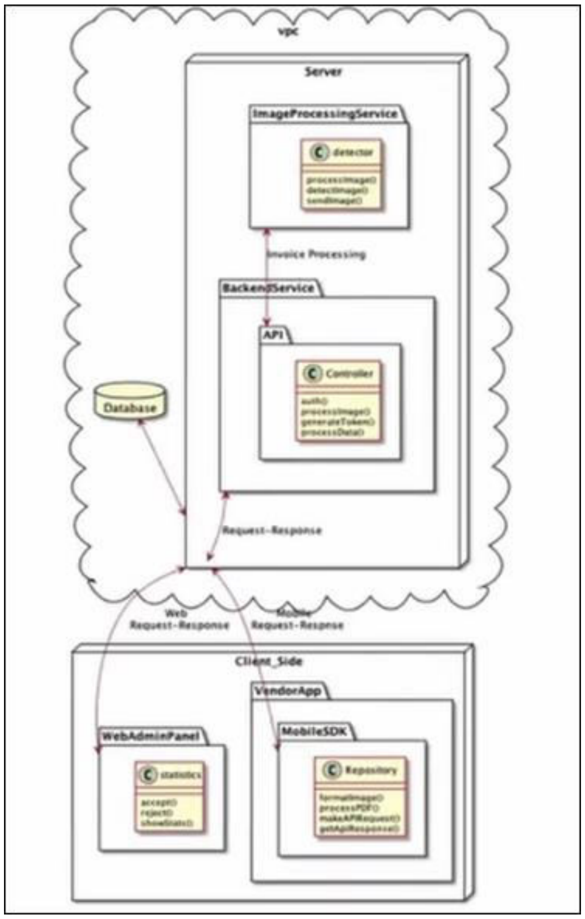

Fig 1 High Level Diagram

- The vendor is required to login to the app and click a picture of an invoice.
- The invoice is uploaded to a cdn and the link is sent to the backend.
- The backend authenticates the vendor and sends the invoice link to invoicing library.
- The invoice is then processed and the raw text received is searched for regex and data is converted to JSON.
- The obtained JSON is sent back to the backend, which in turn sends the

generated data for the vendor to validate.

- Once the vendor validates the data, the backend saves it in a mongoDB[6] database.
- In-case there are minor changes in data, the vendor can specify them and the changes get saved to the database.
- The react[7] web panel is then used to access all the invoices that have been uploaded along with the data generated from them for convenient access by authorized individuals.

The following sequence diagram makes it easier to understand how the sequence of events occurs.

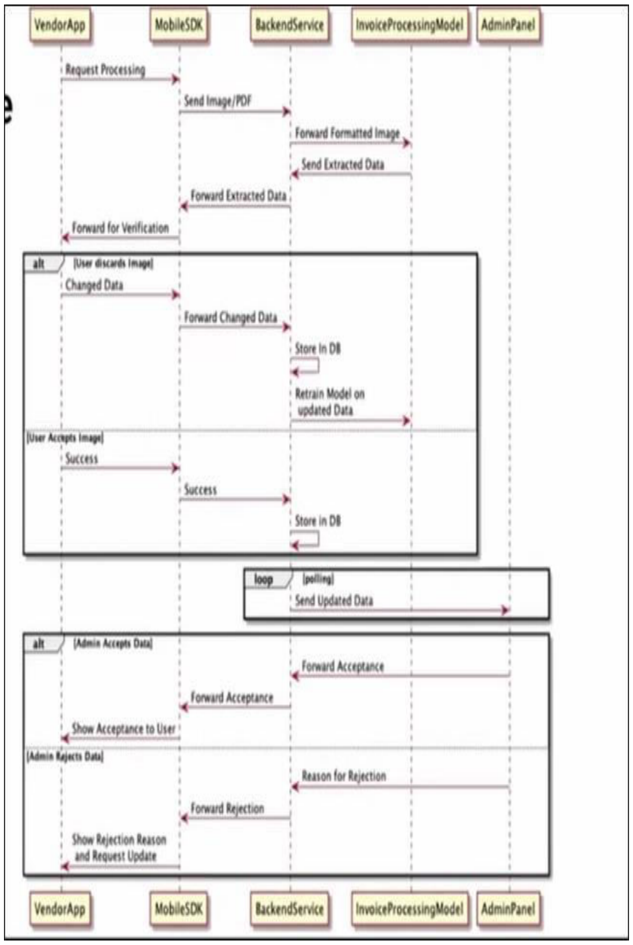

Fig 2: Low Level Sequence Diagram

This flexible invoicing system helps minimize human interaction, which in turn increases efficiency and reduces costs.

VI. FUTURE SCOPE AND CONCLUSION

The project had the sole goal of building a flexible invoicing system which can precisely match content PDF files, easily match line items and tables and automate the entire invoicing process for any major organization.

The demonstrated system has the ability to obtain all relevant information from the document accurately while ensuring speed and reliability.

This project has a huge potential for further development. While the problem focuses on digitization of invoices, this could be extended to digitizing any document for processing, thereby removing any manual efforts, errors and management of document processing within companies.

References

[1] Harsh Arora,"Electronic Invoicing using Image Processing", International Journal for Modern Trends in Science and Technology, 6(12): 520-523,2020.

[2] J. Luo and J. Cross, "Advanced Image Processing Techniques for Maximum Information Recovery," 2007 Thirty-Ninth Southeastern Symposium on System

Theory, Macon, GA, 2007, pp. 58-62, doi: 10.1109/SSST.2007.352317.

[3] Russ J.C. (1990) Image Processing. In: Computer-Assisted Microscopy. Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-0563-7_3

[4] J. M. White and G. D. Rohrer, "Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction," in IBM Journal of Research and Development, vol. 27, no. 4, pp. 400-411, July 1983, doi: 10.1147/rd.274.0400.

[5] Node.JS https://nodejs.org/en/

[6] MongoDB https://www.mongodb.com/

[7] Android sdk and developer tools https://developer.android.com/

[8] React https://reactjs.org/