| **KeepSake Documentation**  Current Date: Dec 7, 2022 |
| --- |

**Product Overview**

Our product is a business card scanner named KeepSake. Its job is to scan business cards and save them in the user's contacts. The primary purpose behind KeepSake is to keep a record of all business cards the user has. It would allow the user to recycle business cards cluttering up their environment. No more paper contact info is needed. Customers can now have everything in a digital form at their fingertips.

KeepSake is a web application that can also run on mobile devices. It is an application that scans business cards and saves them for the user's contacts. The application is a new product written in Python and JavaScript with a user interface extension that allows the customer to use the product.

**Product Objectives**

The Keepsake application is for those possessing unorganized business cards that need an easy way to store the information. Keepsake is used for managing advertisements and organizing connections, allowing the consumer to catalog contact info from a possible employer and business opportunity.

**Team Members**

| **Resource Name** | **Role** |
| --- | --- |
| Kyle McGinnis | Software Developer, Co-Project Leader, UI Designer, Tester |
| Kenneth Russell | Software Developer, GitHub Expert, QA Manager Tester |
| Benjamin Ryan | Software Developer, UI Designer, QA Director, Tester |
| Ryan Johnson | Software Developer, Co-Project Leader, Scrum Master, Tester |

**1 Outline**

* 1. Break Down of Skeleton

Our group built the web application to have only two pages, the main page, and a scanner page. The main page has the web application's title and a description of how to use the app. Then a button allows the user to switch to the scanner page. The scanner page uses the machine's camera to visually see the business card, which then uses the algorithm to read once the scan button is clicked.

* 1. Break Down of Algorithms

The algorithms in the KeepSake application are designed to break down the business card information into four parts: full name and title, address, phone number, and email address. Each piece of the business card will be digitally visible for the user to use in their own contacts.

**2 Libraries and Systems**

When developing KeepSake, the packages used are RegEx, Tesseract, and Next.js. The packages will be explained in depth in the sections below.

2.1 Python

The first package used was Tesseract. The version used is Python-tesseract, which allows the developers to create an optical character recognition(OCR) program to extract text from images.

Then the second package, RegEx, is a regular expression program that allows the developers to teach the computer to recognize different characteristics of how text is formed within an image. The image of a business card is broken down into full name and title, address, phone number, and email address. Each part of the business card is matched based on the different attributes associated with the outline of the text.

2.2 JavaScript

The front end is built with web technologies and utilizes Next.js, a server-side rendering (SSR) framework. This allows end users to access KeepSake on any device with a browser and a webcam or phone camera. Keepsake requests camera access on the front end that pipes to a sandboxed assembly version of Tesseract in the browser before running our algorithms.

**3 User Application**

3.1 Start

When using KeepSake, the user will need to pick a business card to be selected for the program to operate. The user can scan the business card in multiple ways; the user doesn't have to have the card right side up to make KeepSake work.

Ex.



3.2 Scanning

The scanner will then read the information line by line on the card and store the data “locally” on the user’s machine.

**Resources**

*Deployment | Next.js*. (n.d.). <https://nextjs.org/docs/deployment>

GeeksforGeeks. (2020, December 26). *How to Extract Text from Images with Python?* <https://www.geeksforgeeks.org/how-to-extract-text-from-images-with-python/>

*Getting Started | Next.js*. (n.d.). <https://nextjs.org/docs>

*Learn | Next.js*. (n.d.). <https://nextjs.org/learn/foundations/about-nextjs>

*Python - Extract Emails from Text*. (n.d.). <https://www.tutorialspoint.com/python_text_processing/python_extract_emails_from_text.htm>

*Python RegEx*. (n.d.). <https://www.w3schools.com/python/python_regex.asp>

*re — Regular expression operations — Python 3.11.1 documentation*. (n.d.). <https://docs.python.org/3/library/re.html>

Tao, C. (2022, January 7). *7 Useful Tricks for Python Regex to Learn - Towards Data Science*. Medium. <https://towardsdatascience.com/7-useful-tricks-for-python-regex-you-should-know-ec20381e22f2>

Waked, G. (2021, December 14). *How to extract text from a photo of a business card using Python*. Medium. <https://medium.com/@gabrielwaked/how-to-extract-text-from-a-photo-of-a-business-card-using-python-df0e1cfa0a75>