Developer Documentation

for

Copy Print Department Automated Order Printing

Version 20200801

Prepared by Arthur Vardevanyan

Table of Contents

1. Introduction

It is required for a developer to read the customer and operator documentation before reading this document.

1.1 Purpose

The Purpose of this document it to allow a developer to get up to speed on maintaining and modifying this software.

The Purpose of this software is to automate the printing workflow. Due to having a set of high volumes orders where they are all consistent in terms of order structure and option requests, it made sense to develop software to offload that work too.

The Software focus on being a generalizable solution with a focus on a particular customers workflow.

- The Software works best with high order volumes
- Small Individual Orders provide a better return versus few large orders.

The Software allows

- A reduction of wasted time.
- Redundant actions to be automated.
- Inconsistency of orders to be reduced
- Human Error to be reduced.

1.2 References

https://github.com/ArthurVardevanyan/CPD_SO_Automated_Printing
https://www.xerox.com/en-us/digital-printing/digital-presses/xerox-d95-d110-d125
https://github.com/googleapis/google-api-python-client
https://github.com/httplib2/httplib2
https://github.com/mstamy2/PyPDF2
https://github.com/JazzCore/python-pdfkit
https://github.com/adoxa/ansicon/releases
https://github.com/tartley/colorama
https://github.com/hfeeki/termcolor
https://github.com/mysql/mysql-connector-python

1.3 Definitions

Customer is the person who receives the order from the business.
Operator is the person using the software to produce an order.

2. Overall Description

2.1 Product Perspective

This is a total solutions software to automate the printing process from start to finish.
From the moment an order is submitted by a user to the physical print job, nearly all orders can be produced automatically.

For orders that cannot be fully automated, they are still partially automated to the fullest extent that order can be.

2.2 Customer Documentation

Please refer to Customer Documentation for Customer Related information on how to submit an order.

2.3 Operator Documentation

Please refer to Operator Documentation for Operator Related information on how to operate the software.

2.4 Functional Requirements

- Operator Can Enable Order Retrieval
- Operator can Print Tickets for Orders
  - Operator can print specific Tickets
  - Operator can print all Unprinted Tickets
- Operator can Print Orders
  - With Standard Printing & Finishing Options
  - Operator can print orders on White Paper
  - Operator can print orders on Colored Paper / Cardstock
  - Operator can print Tickets with Order
  - Operator can print Saddle Stitched Booklets
  - Operator can choose which printer(s) to print to.
  - Operator can print multiple orders at once.
- The System can determine what options to apply.
- The System can determine how many Sets, and Copies per Set
- The System should verify the integrity of orders.
- The System should revert to Operator Input when Required.

2.5 Product Features

Production Features (Simple Description)

- Set Separation / Job Subdividing, (ex. 5 groups 30 per file)
- Single & Double-Sided Printing.
- Standard Finishing Options
    - Stapling
        - Top Left Portrait
        - Top Left Landscape
        - Double left Portrait
        - Double Top Landscape
    - 3-Hole Punching
- Order Integrity Checking
    - Verifies the orders are valid, and if there is an issue, attempts to recover.
        - (Example: Internet cut during order retrieval)
- Multi Printer Support, and Printer Load Balancing
- Colored Paper and Cardstock
- Saddle Stitched Booklet Printing

Production Features

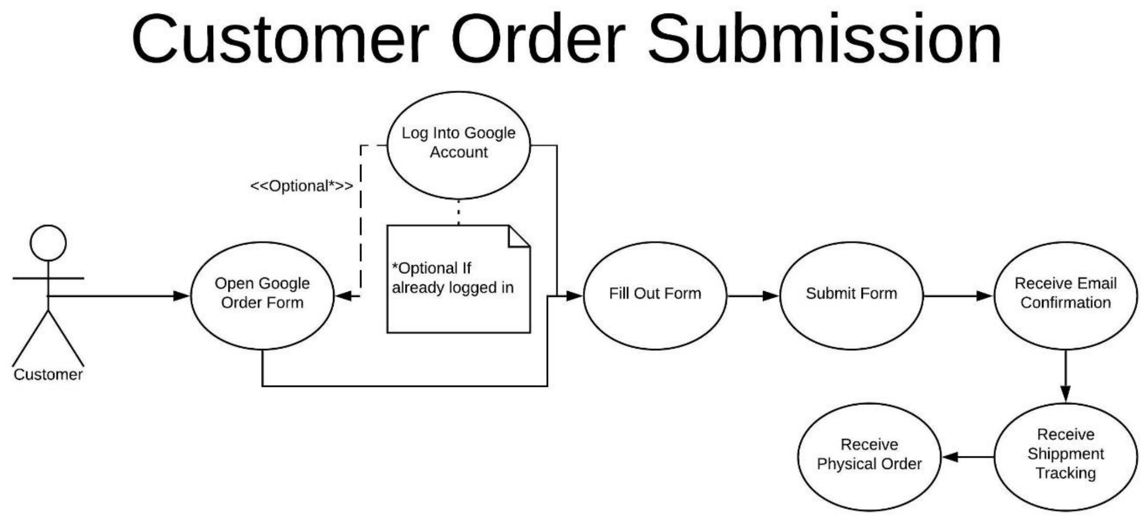
- Order Form (Google Form)
- Generates Confirmation & Order Submission Email
- Downloads Emails
- Downloads Google Drive Files
- Creates and Stores Job Information from the Email
- Contains custom printer "driver"
- Generates Postscript files
- Generates Printable Ticket Form Email
- Custom Separator Banner Sheet's with Job Spec's,
- Job Instruction Error Checking
- Job Integrity Checking
- Bulk Order Printing
- "Natural Language Processing*" to Determine How Many Sets to run.
    - Just some if statements.
- Configures Printer Driver
- Injection of Printer Job Language (PJL) commands
- Printer Load Balancing
- Printer Job Tracker
- Printing of Email/Ticket Out for Billing/Organization
- Run time Logging
- Command Line Interface

Not-Included Beta / Testing Features:

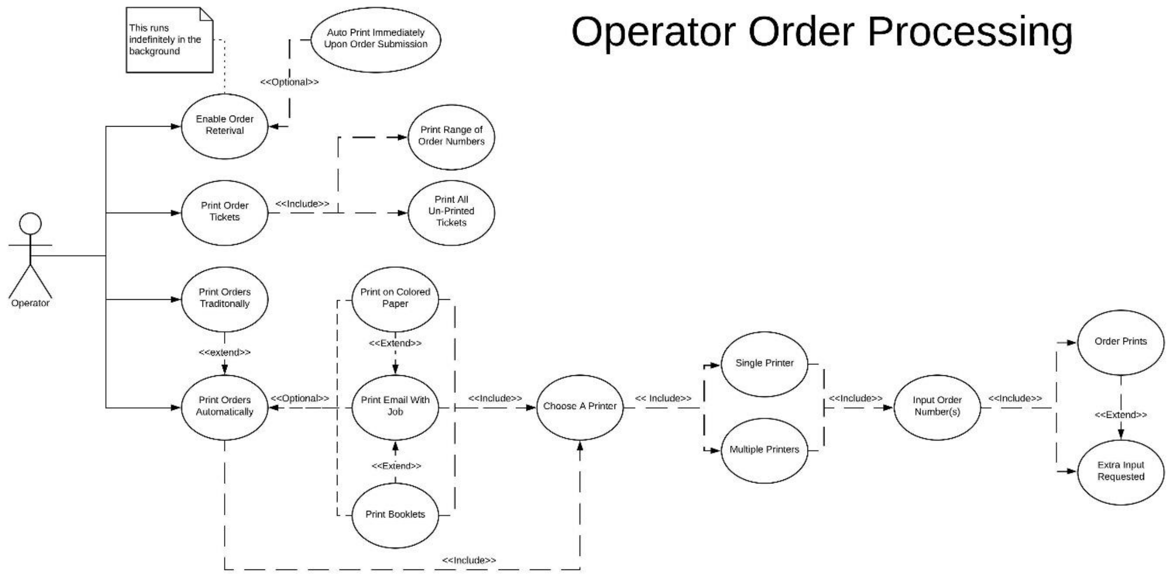
- Multi-Up Printing
- Database Storage
- Web Order Status Tracking
    - Not Started
    - Printing (Including which printer)
    - Waiting for Finishing
    - Waiting to be Shipped
- Shipping Label Creation
- Invoice Generation
- Front/Back Cover Printing

2.6 Use Case Diagrams

Customers Place their orders through the use of a Google Form, they will then receive an email confirmation that their order has been usefully placed, along with the confirmation they also receive a ticket with all the options that they have chosen and an order number.



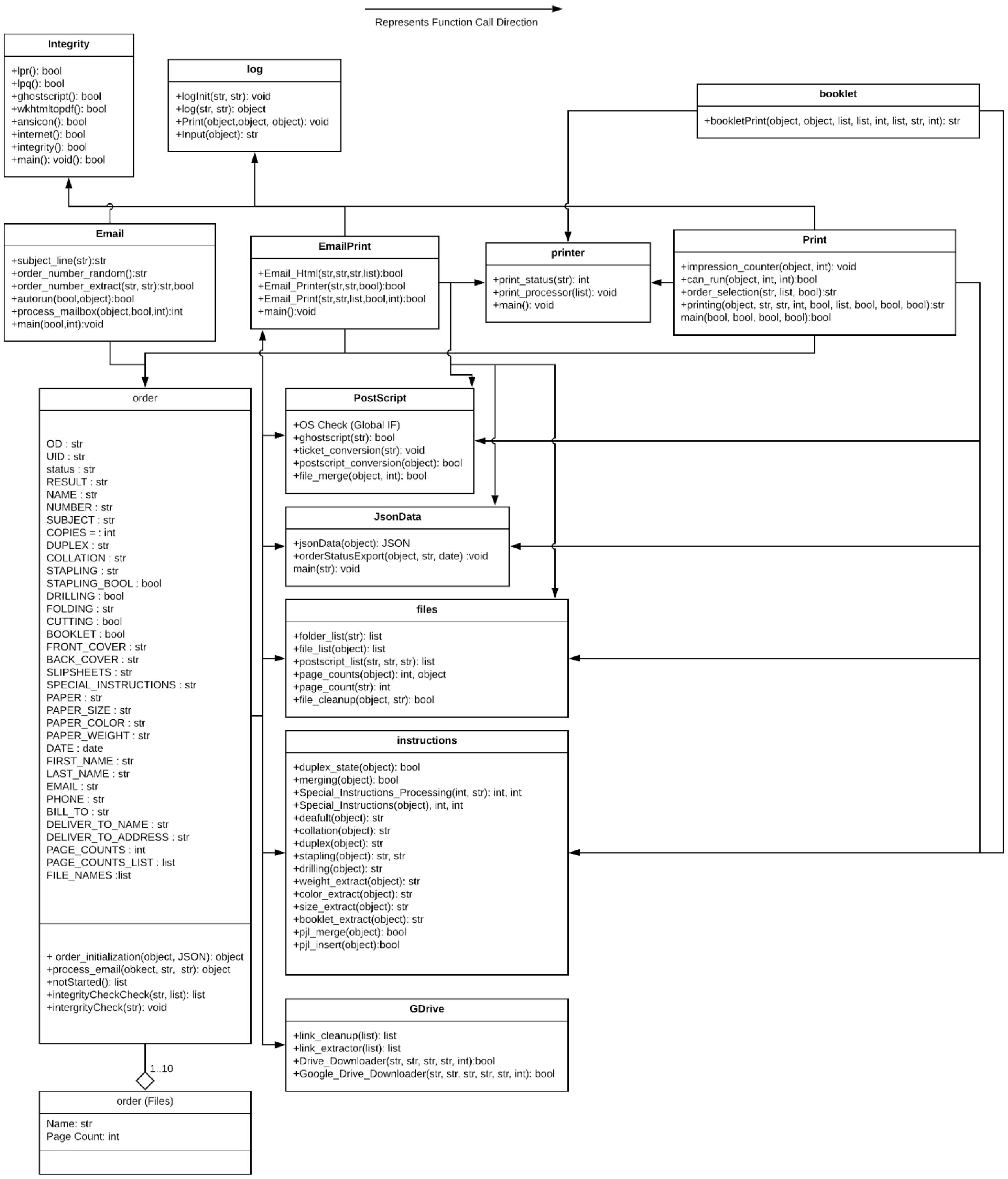
Operator's enable the ability to receive new customer orders.
They then print out tickets for these orders when they are ready to produce them.
The Operator skims the physical tickets for an issue that may occur by using the software.
The Operator will run the orders through the automated system, or if required, traditionally.



2.7 Class Diagram

The software is not OOP, the only object is the Order Class, with the sub class Files. Their may be between 1 and 10 files per order.

The arrows on the diagrams represent which “modules” call other “modules”; modules being groups of code. The functions that each “module” contains are listed in the diagram as well.

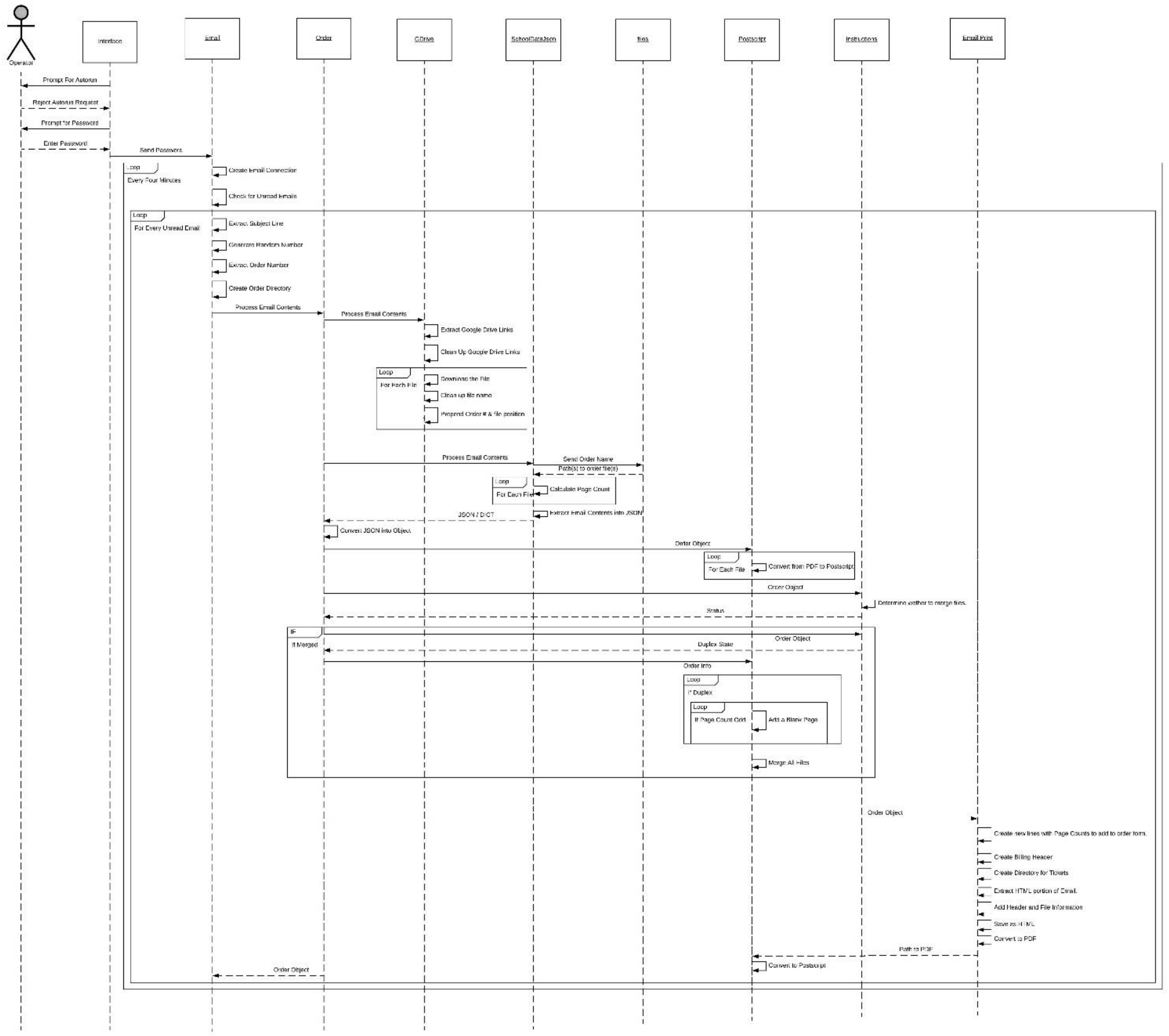

Represents Function Call Direction
Integrity
+lpr(): bool
+lpq(): bool
+ghostscript(): bool
+wkhtmltopdf(): bool
+ansicon(): bool
+internet(): bool
+integrity(): bool
+main(): void(): bool
log
+logInit(str, str): void
+log(str, str): object
+Print(object,object, object): void
+Input(object): str
booklet
+bookletPrint(object, object, list, list, int, list, str, int): str
Email
+subject_line(str):str
+order_number_random():str
+order_number_extract(str, str):str,bool
+autorun(bool,object):bool
+process_mailbox(object,bool,int):int
+main(bool,int):void
EmailPrint
+Email_Html(str,str,str,list):bool
+Email_Printer(str,str,bool):bool
+Email_Print(str,str,list,bool,int):bool
+main():void
printer
+print_status(str): int
+print_processor(list): void
+main(): void
Print
+impression_counter(object, int): void
+can_run(object, int, int):bool
+order_selection(str, list, bool):str
+printing(object, str, str, int, bool, list, bool, bool, bool):str
main(bool, bool, bool, bool):bool
order
OD : str
UID : str
status : str
RESULT : str
NAME : str
NUMBER : str
SUBJECT : str
COPIES = : int
DUPLEX : str
COLLATION : str
STAPLING : str
STAPLING_BOOL : bool
DRILLING : bool
FOLDING : str
CUTTING : bool
BOOKLET : bool
FRONT_COVER : str
BACK_COVER : str
SLIPSHEETS : str
SPECIAL_INSTRUCTIONS : str
PAPER : str
PAPER_SIZE : str
PAPER_COLOR : str
PAPER_WEIGHT : str
DATE : date
FIRST_NAME : str
LAST_NAME : str
EMAIL : str
PHONE : str
BILL_TO : str
DELIVER_TO_NAME : str
DELIVER_TO_ADDRESS : str
PAGE_COUNTS : int
PAGE_COUNTS_LIST : list
FILE_NAMES :list
+ order_initialization(object, JSON): object
+process_email(obkect, str, str): object
+notStarted(): list
+integrityCheckCheck(str, list): list
+intergrityCheck(str): void
PostScript
+OS Check (Global IF)
+ghostscript(str): bool
+ticket_conversion(str): void
+postscript_conversion(object): bool
+file_merge(object, int): bool
JsonData
+jsonData(object): JSON
+orderStatusExport(object, str, date) :void
main(str): void
files
+folder_list(str): list
+file_list(object): list
+postscript_list(str, str, str): list
+page_counts(object): int, object
+page_count(str): int
+file_cleanup(object, str): bool
instructions
+duplex_state(object): bool
+merging(object): bool
+Special_Instructions_Processing(int, str): int, int
+Special_Instructions(object), int, int
+deafult(object): str
+collation(object): str
+duplex(object): str
+stapling(object): str, str
+drilling(object): str
+weight_extract(object): str
+color_extract(object): str
+size_extract(object): str
+booklet_extract(object): str
+pjl_merge(object): bool
+pjl_insert(object):bool
GDrive
+link_cleanup(list): list
+link_extractor(list): list
+Drive_Downloader(str, str, str, str, int):bool
+Google_Drive_Downloader(str, str, str, str, str, int): bool
1..10
order (Files)
Name: str
Page Count: int

2.8 Sequence Diagrams

There are separate PDF’s with just the diagrams.

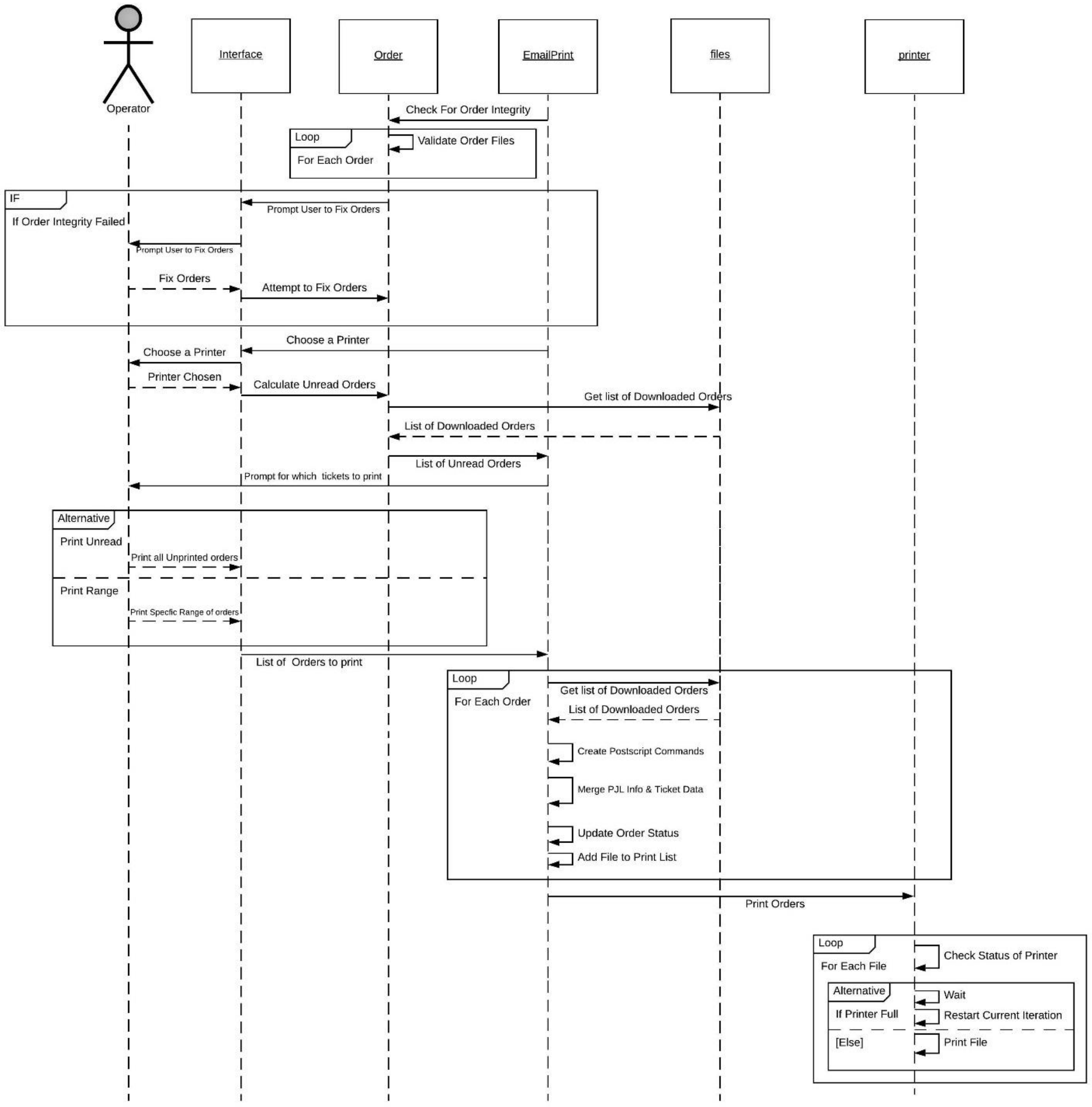
2.8.1 Retrieving Orders (Simplified)

This is a simplified diagram showing the process of orders being retrieved by the software.



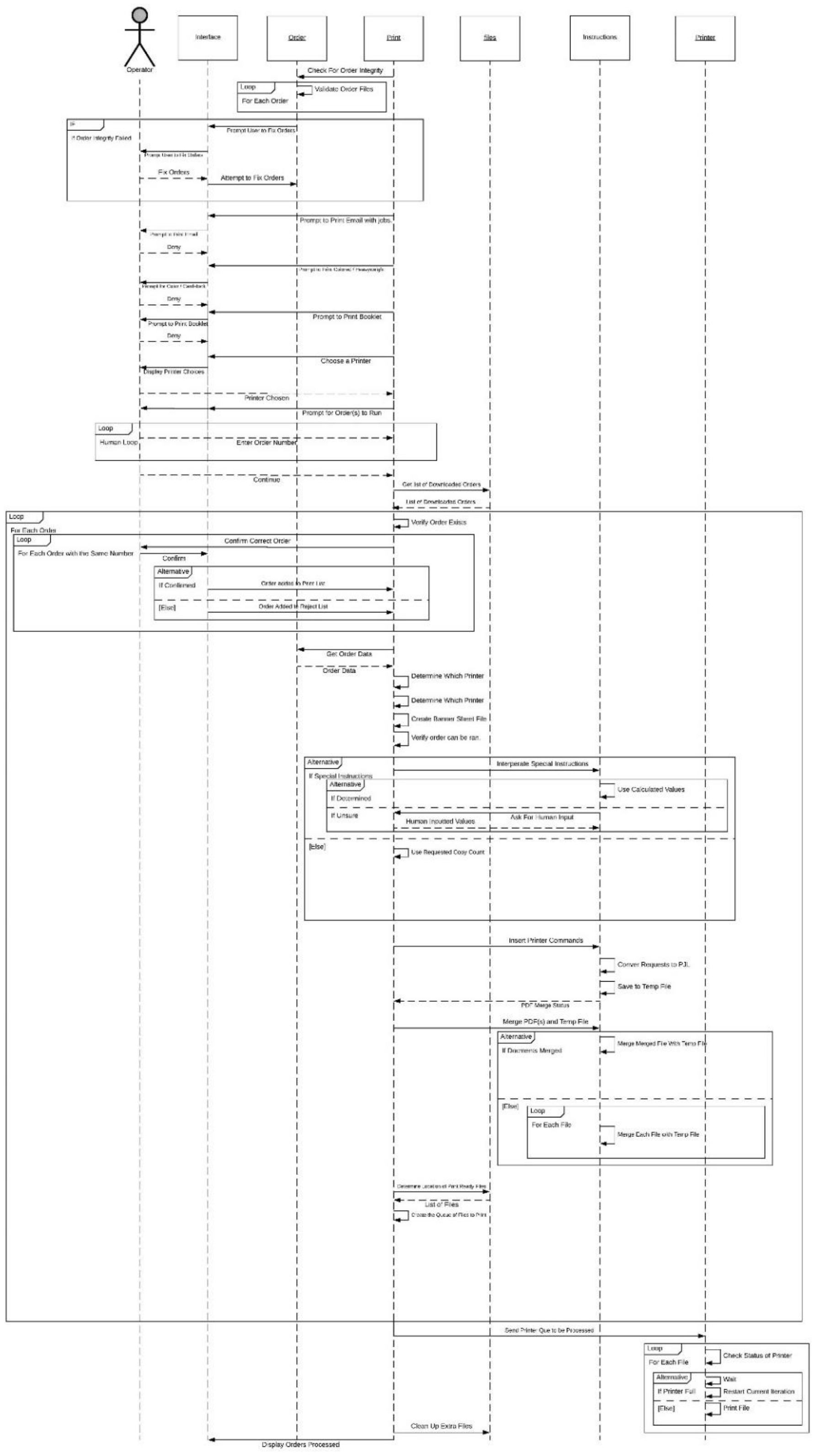
2.8.2 Printing Tickets (Simplified)

This is a simplified diagram showing the process of tickets being printed for orders.



2.8.3 Printing Orders (Simplified)

This is a simplified diagram showing the process of printing orders.



2.9 Operating Environment

Hardware Requirements:

- A Xerox D-Series **Printer (Please note, a physical printer is not required for development, however will be ideal to do final testing and validation of any changes.)**
    - https://www.xerox.com/en-us/digital-printing/digital-presses/xerox-d95-d110-d125
    - Required Attachments
        - Booklet Maker Finisher
        - 2 Tray High Capacity Feeder (HCF-2 Tray)
- An Internet Connection
    - ~10mbps minimum
- A Modern Computer
    - Minimum
        - CPU: i3-2130
        - RAM: 4GB
        - Ethernet Port
    - Recommended
        - CPU: i5-6600
        - RAM: 8GB
        - SSD

Software Requirements:

- Windows Features: LPR (Line printer Daemon) Enabled
- Python 3 and Dependences (Please refer to the requirments.txt)
    - Python
    - term color
    - pdfkit
    - httplib2
    - colorama
    - PyPDF2
    - pyinstaller
    - oauth2client
    - google-api-python-client==1.8.0
        - There is a bug in the newer versions that prevents pyinstaller from compiling.
    - google-auth-httplib2
    - google-auth-oauthlib

- Latest Version of : CPD_SO_Automated_Printing Code
- ansi183 : https://github.com/adoxa/ansicon/releases
- wkhtmltopdf : https://wkhtmltopdf.org/downloads.html
- ghostscript : https://www.ghostscript.com/download.html
- Adobe Reader
    - Adobe Acrobat (Preferred)

Other Requirements, refer to Operator Documentation on how to setup Google Drive API (Ideally a separate account from the one used in production, however the same should be fine):

- Gmail account
- Google Form
- Google Drive Storage
    - (100GB minimum, or use a G-Suite Account with unlimited Storage)
- Gmail API – Insecure App Should be Enabled
- Google Drive API

2.10 Design and Implementation Constraints

- Limited to the features that printers offer
- Must account for edge cases from customer
- Customers are human, they make mistakes, software and/or operator must catch these.
- Operators are human, software should be able to catch some of their mistakes.

Current Network Situation Design and Implementation Constraints
(Please Note: In a normal scenario, these network constraints can be ignored.)

- The Software cannot run on corporate machines, operator must have admin access.
- The machine that the system is operated on cannot use the internal corporate network
- The System must be able to access the internet on one network interface and interact with the printer on-another
- The Devices must all be configured to use the switch as the gateway, and have static IP addresses set.
    - Subnet mask should also be same across all devices.
- Network Structure
    - Ethernet (Layer 2 Switched Network)
        - Network Switch (IP 10.56.54.1 <- Used as Gateway IP for all other devices)
            - Computer (IP 10.56.54.X)
            - Printer 1 (IP 10.56.54.162)
            - Printer 2 (IP 10.56.54.156)
    - Wireless
        - Internet

3. System Features

3.1 Order Form (Google Form)

The ordering system for our customers is built using Google Forms, it is done this way, because it is the best solution, and it would be unreasonable to develop this entire system from scratch.

3.2 Generates Confirmation & Order Submission Email

There is a JavaScript file located in /googleform/emailsender.js. This is used to send the confirmation email to the customer and to us after an order is placed. The Email contains all the relevant job information.

Google Forms, does not contain a feature to send emails on order submissions. While there are paid third party add-ons, this feature is not complex enough to warrant paying for it. It also allows for more control by developing it in-house.

3.3 Downloads Emails

- Unread emails from our Gmail account are fetched approximately every four minutes.
- The Order Number is Extracted from the Email
- Emails are parsed for Order Number, Subject Line
- A random number (current hour/minute) is added to the provided order number
  - Order Form Plug In occasionally repeats order numbers

3.4 Creates and Stores Job Information from the Email

- Each Job is Stored into a folder with the structure:
  - OrderNumber-Randomness Customer Name – Subject
  - Ex: 11349-0311 First Last - Test 3
- Files are stored with order numbers, and which file it is.
  - 11349-0311.03 First Last - Test File.pdf
- Email and Configuration are also Stored with a similar naming scheme
  - 11349-0311 First Last - Test 3.txt / .json
- Temp files are also stored within respective job folders.

3.5 Downloads Google Drive Files

- Google Drive Links are parsed out from the email.
- Google Drive files are downloaded using the API.
- File Names are cleaned up for the Windows Operating System and readability.
- Order Number is added on to the file name(s).
  - Helps identify orders when the printed
- In Addition, files are also numbered sequentially within the order.

3.6 JSON File / Object Creation

- Email is Parsed Further
- All requirements are extracted and stored into a dictionary (map) and JSON file
- Page Counts are either extracted or calculated and stored for each file as well.
- During Runtime, Job Specs are stored in an Object.

3.7 Colored paper & Booklets

- Togglable Options
- Booklets
  - Has different interface workflow due to increase job complexity.
  - Order processing differs from normal jobs

3.8 Contains custom printer "driver"

- Contains all commands necessary to control the printer.
- All Printer commands were reverse engineered and stored
- Printer commands are prepended to each file before being sent to the printer

3.9 Generates Postscript files

- Files are converted from PDF to Postscript
- This conversion is done after the order if finished downloading.

3.10 Generates Printable Ticket Form Email

- Using the HTML version of the Email, a PDF is constructed.
- The previously calculated page counts are also added to this Ticket.
- Two Identical Pages are created for the document.
- Second Page is configured to be printed on yellow.
- Print Ready File is generated.

3.11 Custom Separator Banner Sheet's with Job Spec's

- Create a custom Postscript file with the job's specs
- Insert file before every job when printing on "Purple" Paper
- The Order Number is Prominent on this sheet for easy visibility.

3.12 Job Integrity & Instruction Checking

- Make sure all files have been downloaded and/or created correctly, otherwise attempt to retry.
- Recover Job if failed.
  - Example, Internet goes out while downloading a file.
- Fix incorrect instructions
  - Within reason
- Prevent Running In-Compatible Orders

3.13 Bulk Order Printing

- Print Multiple Orders at once
- Print Orders as they are submitted

3.14 “Natural Language Processing*” to Determine How Many Sets to run

Using keywords inside the sentence and some math comparing the number of total copies requested to the numbers included in the sentence, the amount of sets and copies per set, can be auto determined.

- Read Comments Section to Determine Sets / Copies Per Set
- Key Word Search
- Number Search
- Mathematical Comparisons with Total Copies and recognized numbers, as well as some common sense parameters.
    - Example, nobody would reasonably order 30 Sets of 2. It should probably be 2 sets of 30.
- Defaults to Human Input if Unsure

3.15 Injection of Printer Job Language (PJL) commands

- Take Requested Job Specs and convert to Xerox printer language
- Automatically Configures the custom internal printer “driver”
- Merges the Printer Command Postscript in front of the document postscript files to create print ready postscript files.

3.16 Printer Load Balancing / Printer Job Tracker

- Automatically Distribute Load between multiple printers
- Check Status of job(s) on printer(s)
- Prevent overloading a printer with too many jobs.
    - Printer only holds approximately 40 files in its queue
    - Check How many files are on the Printer
    - Only Send more jobs if printer can support it.
- Auto Determining Which Printer to Print
    - Keep Track of Impressions for each printer
    - Send to Printer with lower impression count

3.17 Run time Logging

- Command Line Interface is Replicated to Log File, as well any additional information, including exceptions.
- Only Critical Information is displayed to the user
- All processes are recorded into a file

- All error and recover attempts are also stored into the log file

3.18 Software Integrity Checking

- Checks the environment to make sure it is suitable to run the software.
    - Dependencies
    - Internet Connection
    - Printers
    - Etc.

3.19 Command Line Interface

- The Interface that is used to interact with the software.

4. Software Architecture & Design

- The software should be designed to reduce the amount of operator input.
- The code is designed in the procedural coding style. Due to the code automating the same task over and over again, the procedure in what needs to be done is very is consistent.
- There is no GUI to provide an event-based flow of events. There however is one object, made up of two classes. There is the “Order” class with multiple “File” classes per order.
- Functions are broken down to their simplest form to increase the ability to test the program and make future modifications.
- Heavy Operations should run during Order Retrieval over Order Production

4.1 File/Code Breakdown

For specifics on functions within in each file, refer to the function headers in the code.

4.1.1 Email.py

Fetches the Emails and calls other functions to setup and preprocess jobs.

- Fetches Emails
- Extracts Subject Lines
- Extracts the Order Number
    - Appends custom order number (To prevent duplicate order numbers)

4.1.2 Print.py

This is the Printing Utility. It prints mostly autonomously for the inputted orders, main aspects it still asks for are which printer to use and if the "Special Instructions" field cannot be automatically determined, it has an operator verify and input information.

- It determines which order is being referenced by operator input
- It checks to see if the order can be run.
- It keeps track of how many copies are sent to each printer.

4.1.3 EmailPrint.py (Tickets)

Prints the Emails with Page Counts and a Duplicate sheet on a different color.

- Generates an HTML Email Ticket
- Generates a PDF version of this Ticket

4.1.4 order.py

Contains the object the contains all the information regarding the current order that is being processed.

- The list of steps to process an incoming email are here. (Refer to Sequence Diagrams)
- Initializations the order object from the local json file.
- Can check the Integrity of Orders.
- Can check whether an order is started or not.

4.1.5 printer.py

It processes the final commands that are sent to the printer, and also checks the status of the printer to ensure it doesn't get overloaded.

- Can Check the status of the printer.
- Processes the Queue of files that need to get printed.

4.1.6 files.py

Grabs list of files and folders

- Can Grab a list of Files
- Can Grab a list of Folders
- Can Grab Page Counts
- Can Cleanup extra files from completed orders.

4.1.7 PostScript.py

Converts the PDFS to PS Files, also merges postscript files when needed.

- Runs the Requested Ghostscript Command
- Converts files to Postscript
- Merges files together.

4.1.8 jsonData.py

- Converts the Email text into a JSON file.
- Exports that status of the order to the JSON file

4.1.9 GDrive.py

Downloads the files for the order from Google Drive
- Extracts and Cleans up the Links from the Email
- Downloads and Renames the Google Drive Files.

4.1.10 BannerSheet.py

Generates a custom banner sheet for running these jobs. Outputted in front of each job before all the files get outputted.

4.1.11 instructions.py

Determines the Job Specs, and Reads the Special Instructions, and determines information based on that. Also Translates Human Instructions into Printer Instructions.
- Converts the Job Specs to the PJL (Print Job Language) Commands.
- Create the PJL header file.
- Merges the PJL header with the job files.

4.1.12 booklets.py

Contains the custom workflow for running Saddle-Stitched Booklets.

4.1.13 log.py

Allows the software to log its actions.
- The default Input and Print functions are overloaded for logging purposes.

4.1.14 integrity.py

Checks the environment to make sure it is suitable to run the software.

4.1.15 google form/emailsender.js

This allows Google Sheets to send formatted emails for incoming orders.

4.1.16 shortcuts/*.bat

For Colored Text to work in windows python must be executed from within command prompt.

4.1.17 buildexe.bat

Generates the executable files for Windows, and cleans up the extra files generated by the executable generator.

4.1.18 PJL_Commands

This folder contains the resources for the Printer Job Language (PJL) Commands needed to output the postscript files on the printer.
In our environment these PJL commands are for a Xerox D-Series (D110) Printer with a

- High Capacity Stacker
- 3 Hole Punch Unit
- Booklet Maker Finisher
- BannerSheetPS.py
- This file is a PJL template file for how the banner sheets print. This gets inserted as is* on to the file generated by BannerSheet.py
    - The color can be changed based on the setting in BannerSheet.py*

4.1.19 PJL_PS.py

This file is a PJL template file for how the job files print. The file gets modified by on the job parameters and then inserting onto each postscript file.

5. Testing

This folder contains all the test code and test files for running tests.
All of the tests are self-contained expect the test that checks for:

- The ability to download Google Drive Files.
- The ability to read and generate Specs from an Email File.

Those pull from a constant file we use in production, and checks against a known.
Both files will need to be adjusted for your environment.

6. Issues List

- No Known Bugs

6.1 TODO

- Refactor
- More Tests
    - Adjust the couple of tests that are not self-contained.
- Remove hardcoded values.
- Optimization.
- Feature Idea, run final printing steps preemptively, improves performance if nothing changes when actually going to print the order, this idea needs more analysis on repercussions and implementation details.

7. External Interface Requirements

7.1 User Interfaces

- There is no user interface, the system operates inside a command prompt / terminal interface.
- Very basic questions are passed to the operator when needed, otherwise, as much is abstracted as possible.

7.2 Hardware Interfaces

- The Xerox Production Printers.
- The Host Computer

7.3 Software Interfaces

- Windows 10
- Latest Version of CPD_SO_Automated_Printing Code
- ansi183 : https://github.com/adoxa/ansicon/releases
- wkhtmltopdf : https://wkhtmltopdf.org/downloads.html
- ghostscript : https://www.ghostscript.com/download.html
- Adobe Reader, (Adobe Acrobat Preferred)

7.4 Communications Interfaces

- Customer Software is hosted on Google Forms, and is communicated using HTTPS.

- Operator Software runs using http to communicate with the internet to retrieve orders.
- It also uses TCP/IP to communicate to the production printers.
- A keyboard for using the software is the primary human interface communication device.
- A mouse is used for navigating the host operating system.

8. Other Nonfunctional Requirements

8.1 Performance Requirements

- As many operations as possible should be ran during order retrieval
- Placement of slow operations are placed at the end of the operator sessions.
- Operations that can be ran Asynchronously, should be done so.
- Operations that can be threaded should be.

8.2 Safety & Security Requirements

- Customer Data should remain protected.
- Orders should print correctly
- Checks should be in place to prevent Operator Error where reasonable.

8.3 Software Quality Attributes

8.3.1 Usability

- Minimum amount of operator interaction
- Minimal options presented to the Operator
- Guiding Operator when issues or interaction is required.

8.3.2 Interoperability

The Software has to interact with:

- Gmail
- G-Drive
- Printers
- Database(s)
- Other local machine dependencies
    - The Operating System
    - Ghostscript (Postscript converter)
    - A html to pdf converter software
- The Local Network

8.3.3 Reliability / Availability

- Overall, the Software should run fairly bug free
- Edge case will happen that neither an operator nor the software catches, and should fall on the operator to handle correctly handle.
- Customer interaction is through Google Forms, which has almost 0 down time.
- The negative down time for production related to the software is negligible, due to the intake volume and speed that it can be caught up at.
- Occasionally when all the printers are down for maintenance that is an issue, however same as above, the speed at which back log can be processed is generally not an issue.

8.3.4 Maintainability / Modifiability / Reusability

- Features Components should be able to be added and removed without breaking total functionality.
- Components should be adaptable towards other projects.

8.3.5 Testability

- The software should be written such that features can easily be tested.

8.3.6 Scalability

- The software should be able to run smoothly as order volume increases.

- Weakest Link is Physical Machines
- High Volume of orders could limit:
    - Internet connection
    - Processing from PDF to PS (Ghost Script converter is also limited to being single threaded)
        - Multiple Contract, High Volume Order options are possible through virtualization, but that currently is not necessary.
    - Printer print speed and QTY
    - Operator ability to Box and Ship

8.3.7 Supportability

- The Software should have Run Time Logging
- The Software should have Error Tracking and Recovery
- The Software should have Order Error Checking
- The Software should have Job Integrity Checking
- The Software should have Input Validation