STREAMLINING BUSINESS CHECK WRITING WITH PYTHON AUTOMATION

Ethan Tinsley, University of North Georgia

Tamirat Abegaz, University of North Georgia

Ash Mady, University of North Georgia

Mingyuan Yan, University of North Georgia

ABSTRACT

Many companies still rely heavily on the usage of business checks to pay employees, partners, and invoices. However, the manual process of writing and formatting these checks can be time consuming and burdensome. Although there are existing automated check writing processes, they are mostly designed to writing an individual checks. In this study, we present an alternative approach to automating the business check writing process using Python. Bulk check writing is where multiple checks are printed from a single input file (.csv,. json, .xml). To accomplish this, we utilize the Python programming language and the Python-docx library. The proposed automation solution allows us to write and format business checks that meet MICR E-13B banking specifications. The process also allows multiple output file formats such as MS docx and PDF. With the utilization of python-docx library, we manipulate a template file using a "search and replace" technique to create formatted checks with the desired information. The results of our study demonstrate the capabilities and the efficiency of the automated business check writing process. To execute the process, users only need to execute a Python program and use an ordinary printer to produce a physical copy of the needed checks. Our approach can easily integrate into existing workflows. This allows businesses to save valuable time and resources while increasing productivity. Overall, the presented automation process provides great economic advantage through its efficiency, accuracy, and speed.

Keywords: MICR, E-13B, Business Check, Python, python-docx

INTRODUCTION

Business checks remain a central avenue for conducting business-to-business, and business-to-employee transactions within the modern economy. According to the Federal Reserve Sites, billions of checks are used annually for exchanging money (Board of Governors of The Federal Reserve System: Commercial Checks Collected Through the Federal Reserve--Annual Data). While the number of paper checks issued has declined due to the availability of alternative payment methods, they still account for a staggering 14.5 billion checks written with a value of $25.80 trillion in 2018 (FRS, 2019). Additionally, in 2018 the average value of check payments grew $1,779 compared to previous years (FRS, 2019). Surprisingly, over 60% of consumers reported using paper checks at least once, and most business and government agencies continue to rely on paper checks (Greene, 2020). As business checks still represents a significant portion of business transactions, they must meet specific criteria to be accepted by the bank as a valid form of payment. Each check in circulation must follow the same characteristics to meet these banking standards.

The standard characteristics of a check must include the payee (the recipient of the payment), the payer, a legal money amount (the written amount on the memo line), a digitized amount (the amount in digits), the date the check was printed, the name of the banking institutions, the account routing number, and account number, the payer's signature, and an MICR E-13B line (Bank of America: MICR Specification Sheet). The E-13B font is the standard practice used by U.S banking institutions, and MICR (Magnetic Ink Character Recognition) is the traditional method for reading characters on checks. Each E-13B character consists of a different magnetic signature, which allows banks to quickly process checks' routing and account numbers based on their magnetic signature.

While the banking industry initially relied on MICR to extract the key information from checks, the past three decades have revealed a more desirable system: Optical Character Recognition (OCR) (Van Steenis, 1971). With OCR, the same check structure can be used without the need for specialized printers to print the E-13B font with magnetic signatures. This allows businesses and users to print their own checks without requiring specialized equipment. Furthermore, it leads to the way for development and implementation of private automated check writing programs.

An automated check-writing program would enable the user to directly upload a data file in any typical format, such as .csv, .json, or .html, to a central database. This data can then be used to create formatted checks that adhere to MICR banking standards and are ready for printing. To accomplish this, the program needs to manipulate a standard document to insert the check information then store it. This document can then quickly be selected and printed on a standard business or consumer printer.

The Python programming language (Python: Python Programming) offers many useful and reliable code libraries that support the development of programs with a wide range of advanced functionalities. One such key library for document manipulation is the python-docx library (Python: python-docx). This library enables developers to search or parse through a word document and replace keywords with the appropriate text, making it a crucial tool for document manipulation automation. These keywords serve as indicators for the text fields. Identifying specific text fields where the respective information needs to be placed is essential for the success of the automation process. During the development of check writing automation in Python, this technique is the backbone. By manipulating a properly formatted template that follows MICR compliance, we can generate and print checks. This project aims to implement an automated business process for bulk check writing by leveraging the power of automation in Python.

MICR LINE AND OCR IMPLEMENTATION

The E-13B MICR line is the standard for the banking industry to verify the validity and utilization of a check (Kelechava, 2020). This system was adopted so that check-processing machines can quickly read the magnetic signature contained on the band and process the origin and routing information of the check. The font remains constant despite the presence of more modern forms of check-processing methods, such as OCR (Chin & Wu, 1995). The advantages ushered along with OCR include the ability to produce checks using any printer as long as the proper anti-copying check paper is used. Without requiring specialized printers to produce checks, businesses can drastically reduce expenses on equipment and specialized software.

The recent implementation of OCR for reading MICR lines involves the use of ANN (Artificial Neural Networks) to analyzes each character in the line and determine the respective routing and account numbers. ANN, a technique that trains a computer to learn a key characteristic and create patterns for predicting output based on similar input, is employed in OCR. With OCR, ANN is designed and implemented to predict the character or ASCII value of writing on paper, or in a digitalized format such as a pdf or image file (Zhang, 2008).

The OCR process has significantly reduced the cost of check-reading and writing software, as ordinary printers can now be used to write checks as the magnetic signatures are no longer necessary. Additionally, check readers now can capture images of incoming checks and process them using OCR techniques to identify check amounts, account and routing numbers, and other relevant checking information.

AUTOMATED DOCUMENT WRITING

Automated document writing has been implemented to modify, save, and print research paper abstracts in research journals. The principals of the study involve utilizing Python and the python-docx library to manipulate .docx files. The study found that utilizing the python-docx library could greatly improve the efficiency of adding the journal abstracts to the journal (Ilina & Pelevanyuk, 2020). Implementing the python-docx library to automate the replication of journal papers is just one example of the library's potential, as it could also be utilized to write check-specific information onto a business check template.

The python-docx library has been implemented as a "search and replace" tool for the migration of legacy clinical data (Dunn, Cobb, Levey & Gutman, 2016). In this role, the python-docx library is responsible for searching for keywords within a document and replacing them with the accurate information retrieved from a central database. The search and replace functionality it very useful when maintaining document formatting is involved. By creating a central document with appropriate formatting and keywords, the search and replace utility ensures that the desired information is inserted into the document while maintaining the intended formatting.

Program Requirements

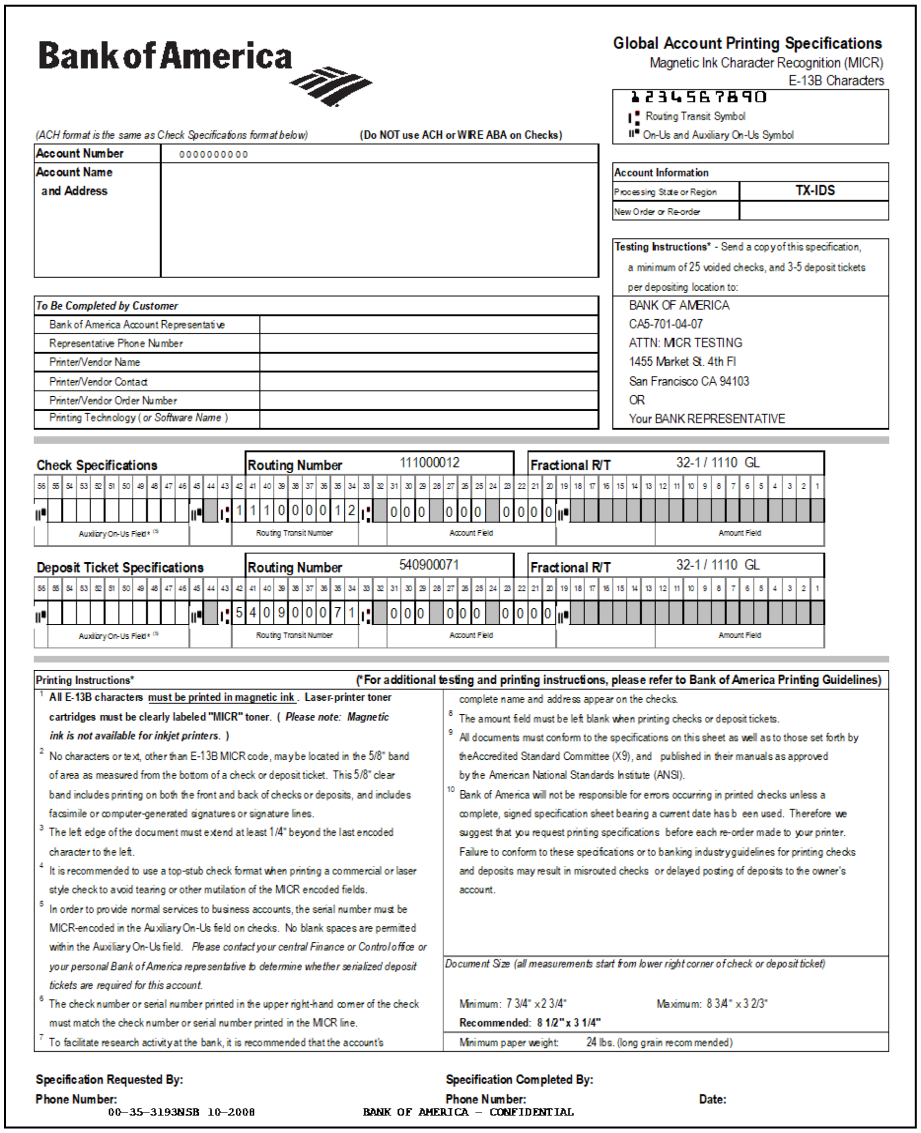
To automate the check writing process in Python, certain output features must be achieved. These requirements shape the design, implementation, and tools utilized in order to automate business check writing. The program should be capable of reading check information (objects) from an input file. The input file can be formatted as a comma-separated values (.csv) file, JavaScript Object Notation (.json) file, or extensible markup language (.xml) file. These file types offer flexibility in creating an input checkbook and do not restrict to a single input parameter.

The program must format a memo table that is displayed above the check, showing the charges associated with the check. This feature allows each printed check to display where the funds are going via an invoice number, invoice date, and individual charges corresponding to that check number.

Each printed check must comply to the MICR banking specifications, ensuring that the appropriate check fields are displayed on the check, such as amount, payee, signature, bank, and other required information.

Each printed check must include a MICR line with appropriate information (check number, routing number, and account number) in the E-13B font. This is crucial as the MICR line is the key component in check processing with OCR. Formatting the MICR line to meet MICR banking specifications is essential for producing usable and legally compliant checks.

The check specifications are based on the Bank of America check specification sheet (Bank of America: MICR Specification Sheet) . This document outlines the proper formatting for the check. The check specification sheet is provided to check printers so that they produce Bank of America compliant checks. This sheet serves as a central guideline for how the output checks should be displayed (Figure 1-3).



Global Account Printing Specifications
Magnetic Ink Character Recognition (MICR)
E-13B Characters

1234567890
⑆ Routing Transit Symbol
⑈ On-Us and Auxiliary On-Us Symbol

(ACH format is the same as Check Specifications format below) **(Do NOT use ACH or WIRE ABA on Checks)**

Account Number	0000000000
Account Name and Address	

Account Information	
Processing State or Region	**TX-IDS**
New Order or Re-order	

Testing Instructions* - Send a copy of this specification, a minimum of 25 voided checks, and 3-5 deposit tickets per depositing location to:
BANK OF AMERICA
CA5-701-04-07
ATTN: MICR TESTING
1455 Market St. 4th Fl
San Francisco CA 94103
OR
Your BANK REPRESENTATIVE

To Be Completed by Customer	
Bank of America Account Representative	
Representative Phone Number	
Printer/Vendor Name	
Printer/Vendor Contact	
Printer/Vendor Order Number	
Printing Technology (*or Software Name*)	

Check Specifications **Routing Number** 111000012 **Fractional R/T** 32-1 / 1110 GL

Positions 56–1: ⑈ Auxiliary On-Us Field* [5] ⑈ ⑆ 111000012 (Routing Transit Number) ⑆ 000 000 0000 (Account Field) ⑈ Amount Field

Deposit Ticket Specifications **Routing Number** 540900071 **Fractional R/T** 32-1 / 1110 GL

Positions 56–1: ⑈ Auxiliary On-Us Field* [5] ⑈ ⑆ 540900071 (Routing Transit Number) ⑆ 000 000 0000 (Account Field) ⑈ Amount Field

Printing Instructions* **(*For additional testing and printing instructions, please refer to Bank of America Printing Guidelines)**

[1] **All E-13B characters must be printed in magnetic ink . Laser-printer toner cartridges must be clearly labeled "MICR" toner. (*Please note: Magnetic ink is not available for inkjet printers.*)**

[2] No characters or text, other than E-13B MICR code, may be located in the 5/8" band of area as measured from the bottom of a check or deposit ticket. This 5/8" clear band includes printing on both the front and back of checks or deposits, and includes facsimile or computer-generated signatures or signature lines.

[3] The left edge of the document must extend at least 1/4" beyond the last encoded character to the left.

[4] It is recommended to use a top-stub check format when printing a commercial or laser style check to avoid tearing or other mutilation of the MICR encoded fields.

[5] In order to provide normal services to business accounts, the serial number must be MICR-encoded in the Auxiliary On-Us field on checks. No blank spaces are permitted within the Auxiliary On-Us field. *Please contact your central Finance or Control office or your personal Bank of America representative to determine whether serialized deposit tickets are required for this account.*

[6] The check number or serial number printed in the upper right-hand corner of the check must match the check number or serial number printed in the MICR line.

[7] To facilitate research activity at the bank, it is recommended that the account's complete name and address appear on the checks.

[8] The amount field must be left blank when printing checks or deposit tickets.

[9] All documents must conform to the specifications on this sheet as well as to those set forth by theAccredited Standard Committee (X9), and published in their manuals as approved by the American National Standards Institute (ANSI).

[10] Bank of America will not be responsible for errors occurring in printed checks unless a complete, signed specification sheet bearing a current date has b een used. Therefore we suggest that you request printing specifications before each re-order made to your printer. Failure to conform to these specifications or to banking industry guidelines for printing checks and deposits may result in misrouted checks or delayed posting of deposits to the owner's account.

Document Size (all measurements start from lower right corner of check or deposit ticket)

Minimum: 7 3/4" x 2 3/4" Maximum: 8 3/4" x 3 2/3"
Recommended: 8 1/2" x 3 1/4"

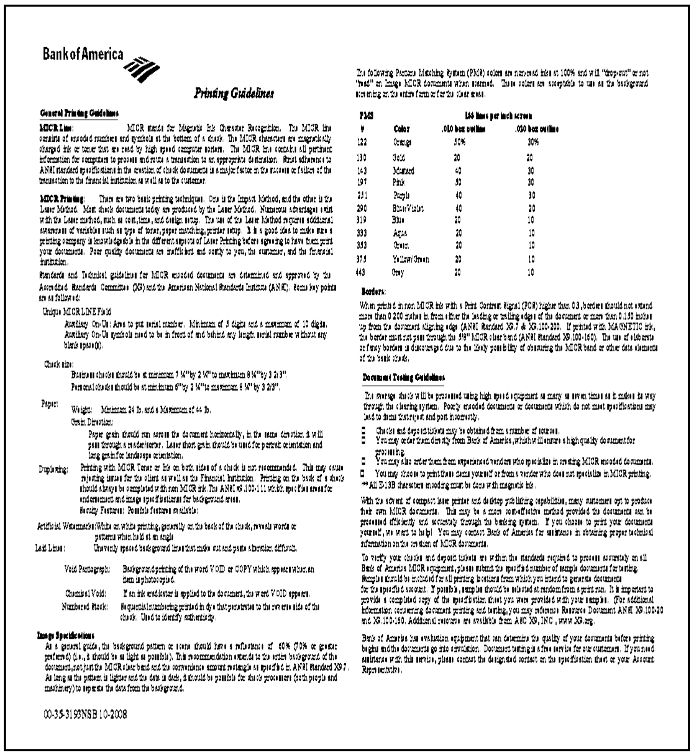
Minimum paper weight: 24 lbs. (long grain recommended)

Specification Requested By:
Phone Number:

Specification Completed By:
Phone Number: **Date:**

00-35-3193NSB 10-2008 BANK OF AMERICA – CONFIDENTIAL

Figure 1. Bank of America Spec Sheet 1



Bank of America

Printing Guidelines

General Printing Guidelines

MICR Line: MICR stands for Magnetic Ink Character Recognition. The MICR line consists of encoded numbers and symbols at the bottom of a check. The MICR characters are magnetically charged ink or toner that are read by high speed computer sorters. The MICR line contains all pertinent information for computers to process and route a transaction to an appropriate destination. Strict adherence to ANSI standard specifications in the creation of check documents is a major factor in the success or failure of the transaction to the financial institution as well as to the customer.

MICR Printing: There are two basic printing techniques. One is the Impact Method, and the other is the Laser Method. Most check documents today are produced by the Laser Method. Numerous advantages exist with the Laser method, such as cost, time, and design setup. The use of the Laser Method requires additional awareness of variables such as type of toner, paper matching, printer setup. It is a good idea to make sure a printing company is knowledgeable in the different aspects of Laser Printing before agreeing to have them print your documents. Poor quality documents are inefficient and costly to you, the customer, and the financial institution.

Standards and Technical guidelines for MICR encoded documents are determined and approved by the Accredited Standards Committee (X9) and the American National Standards Institute (ANSI). Some key points are as followed:

Unique MICR LINE Field

Auxiliary On-Us: Area to put serial number. Minimum of 5 digits and a maximum of 10 digits. Auxiliary On-Us symbols need to be in front of and behind any length serial number without any blank space(s).

Check size:

Business checks should be at minimum 7 ¾" by 2 ¾" to maximum 8 ¾" by 3 2/3".

Personal checks should be at minimum 6" by 2 ¾" to maximum 8 ¾" by 3 2/3".

Paper:

Weight: Minimum 24 lb. and a Maximum of 44 lb.

Grain Direction:

Paper grain should run across the document horizontally, in the same direction it will pass through a reader/sorter. Laser short grain should be used for portrait orientation and long grain for landscape orientation.

Duplexing: Printing with MICR Toner or Ink on both sides of a check is not recommended. This may cause rejecting issues for the client as well as the Financial Institution. Printing on the back of a check should always be completed with non MICR ink. The ANSI x9.100-111 which specifies areas for endorsement and image specifications for background area.

Security Features: Possible features available:

Artificial Watermarks: White on white printing, generally on the back of the check, reveals words or patterns when held at an angle

Laid Lines: Unevenly spaced background lines that make cut and paste alteration difficult.

Void Pantograph: Background printing of the word VOID or COPY which appears when an item is photocopied.

Chemical Void: If an ink eradicator is applied to the document, the word VOID appears.

Numbered Stock: Sequential numbering printed in dye that penetrates to the reverse side of the check. Used to identify authenticity.

Image Specifications

As a general guide, the background pattern or scene should have a reflectance of 60% (70% or greater preferred) (i.e., it should be as light as possible). This recommendation extends to the entire background of the document, not just the MICR clear band and the convenience amount rectangle as specified in ANSI Standard X9.7. As long as the pattern is lighter and the data is dark, it should be possible for check processors (both people and machinery) to separate the data from the background.

00-35-3193NSB 10-2008

The following Pantone Matching System (PMS) colors are non-read inks at 100% and will "drop-out" or not "read" on Image MICR documents when scanned. These colors are acceptable to use as the background screening on the entire form or for the clear areas.

PMS		133 lines per inch screen	
#	Color	.010 box outline	.020 box outline
122	Orange	50%	30%
130	Gold	20	20
143	Mustard	40	30
197	Pink	50	30
251	Purple	40	30
290	Blue/Violet	40	20
319	Blue	20	10
333	Aqua	20	10
353	Green	20	10
375	Yellow/Green	20	10
443	Grey	20	10

Borders:

When printed in non MICR ink with a Print Contrast Signal (PCS) higher than 0.3, borders should not extend more than 0.200 inches in from either the leading or trailing edges of the document or more than 0.150 inches up from the document aligning edge (ANSI Standard X9.7 & X9.100-200. If printed with MAGNETIC ink, the border must not pass through the 5/8" MICR clear band (ANSI Standard X9.100-160). The use of elaborate or fancy borders is discouraged due to the likely possibility of obscuring the MICR band or other data elements of the basic check.

Document Testing Guidelines

The average check will be processed using high speed equipment as many as seven times as it makes its way through the clearing system. Poorly encoded documents or documents which do not meet specifications may lead to items that reject and post incorrectly.

- ☐ Checks and deposit tickets may be obtained from a number of sources.
- ☐ You may order them directly from Bank of America, which will ensure a high quality document for processing.
- ☐ You may also order them from experienced vendors who specialize in creating MICR encoded documents.
- ☐ You may choose to print these items yourself or from a vendor who does not specialize in MICR printing.

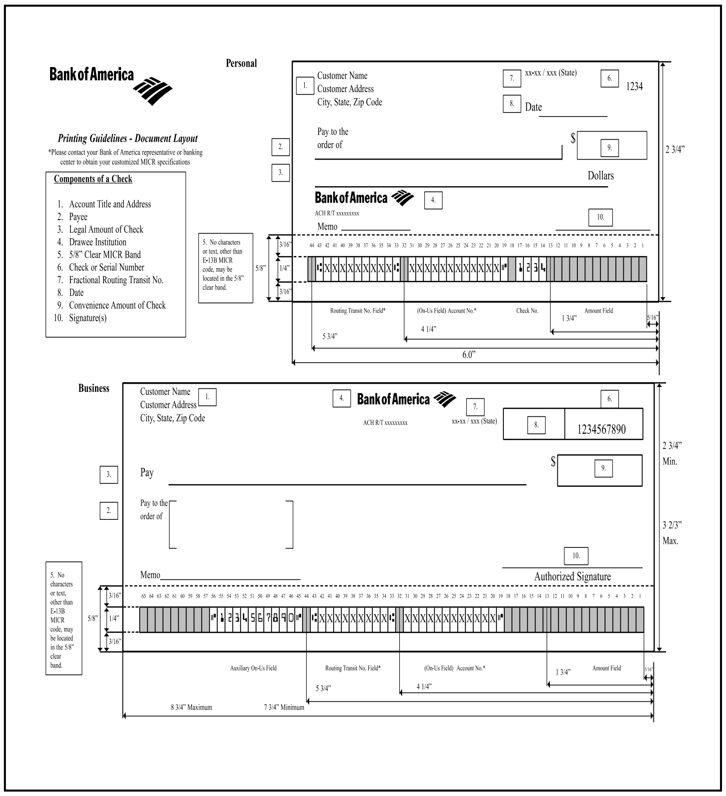
*** All E-13B characters encoding must be done with magnetic ink.

With the advent of compact laser printer and desktop publishing capabilities, many customers opt to produce their own MICR documents. This may be a more cost-effective method provided the documents can be processed efficiently and accurately through the banking system. If you choose to print your documents yourself, we want to help! You may contact Bank of America for assistance in obtaining proper technical information on the creation of MICR documents.

To verify your checks and deposit tickets are within the standards required to process accurately on all Bank of America MICR equipment, please submit the specified number of sample documents for testing. Samples should be included for all printing locations from which you intend to generate documents for the specified account. If possible, samples should be selected at random from a print run. It is important to provide a completed copy of the specification sheet you were provided with your samples. (For additional information concerning document printing and testing, you may reference Resource Document ANSI X9.100-20 and X9.100-160. Additional resource are available from ASC X9, INC, www.X9.org.

Bank of America has evaluation equipment that can determine the quality of your documents before printing begins and the documents go into circulation. Document testing is a free service for our customers. If you need assistance with this service, please contact the designated contact on the specification sheet or your Account Representative.

Figure 2. Bank of America Spec Sheet 2



Bank of America

Printing Guidelines - Document Layout

*Please contact your Bank of America representative or banking center to obtain your customized MICR specifications

Components of a Check

1. Account Title and Address
2. Payee
3. Legal Amount of Check
4. Drawee Institution
5. 5/8" Clear MICR Band
6. Check or Serial Number
7. Fractional Routing Transit No.
8. Date
9. Convenience Amount of Check
10. Signature(s)

5. No characters or text, other than E-13B MICR code, may be located in the 5/8" clear band.

Figure 3. Bank of America Spec Sheet 3

Architectural Design

The program's architectural design was implemented using object-oriented programming, which involved creating classes for convenient variable storage objects (checking, banking, account information) that can be reused throughout the program. As shown in Figure 4, The backend design of the application focuses on creating key objects used throughout the application. Additionally, the backend design provides classes that compose the main functionality of the program. The core objects include: the Check object, which stores relevant check information; the Bank object, which stores information about the bank for which the check is formatted; the Company object, which stores information related to the company printing the check; the Signature object, utilized for storing an image of an authorized signee to be displayed on the check; and finally, the Account object, which stores the account number and routing number to be displayed in the MICR line of the check.

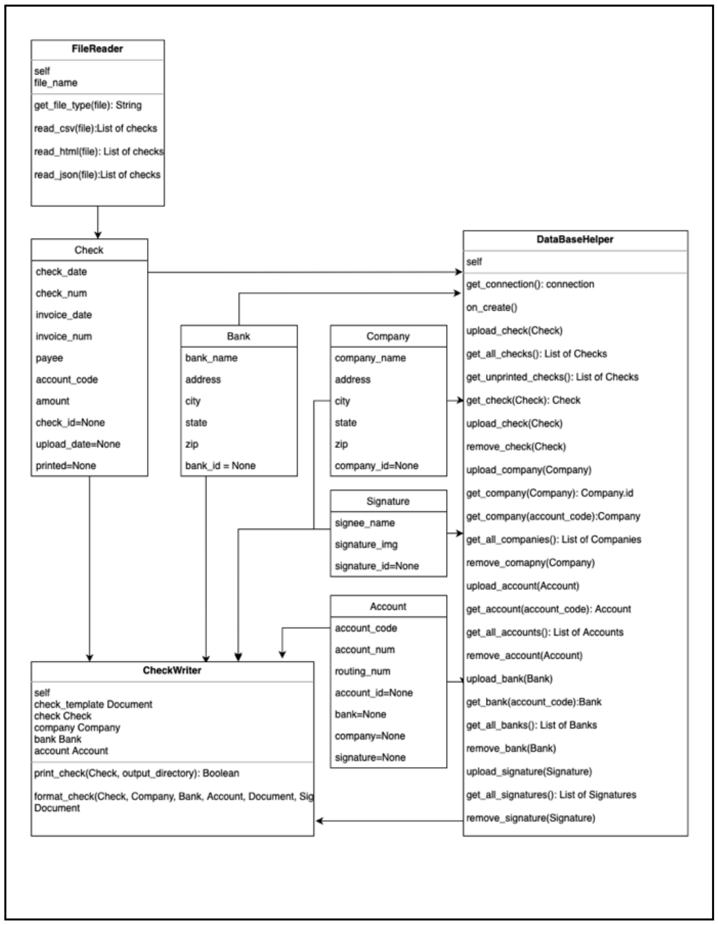


Figure 4. Backend UML

The FileReader, CheckWriter, and DatabaseHelper classes serve as the core functionality of the program. The FileReader class is responsible for parsing an input file and extracting the respective business checks (Check objects) from the file. The ChekWriter class is responsible for implementing the 'search and replace' technique to format and output the business check accordingly. Finally, the DatabaseHelper class contains all the methods used in the database interactions, including uploading, updating, and retrieving data.

The frontend design of the program (Figure 5) revolves around designing the classes used to create the user interface and implement the backend to the UI. The fronted implements objects and methods from the Tkinter user interface module (Python documentation: Tkinter). The Tkinter module enables the creation of the application window, content and navigational frames, as well as widgets such as buttons and labels, and provides formatting options for these objects. The frontend design focuses on the display of the user interface and provides an environment for the user to interact with the backend functionality.

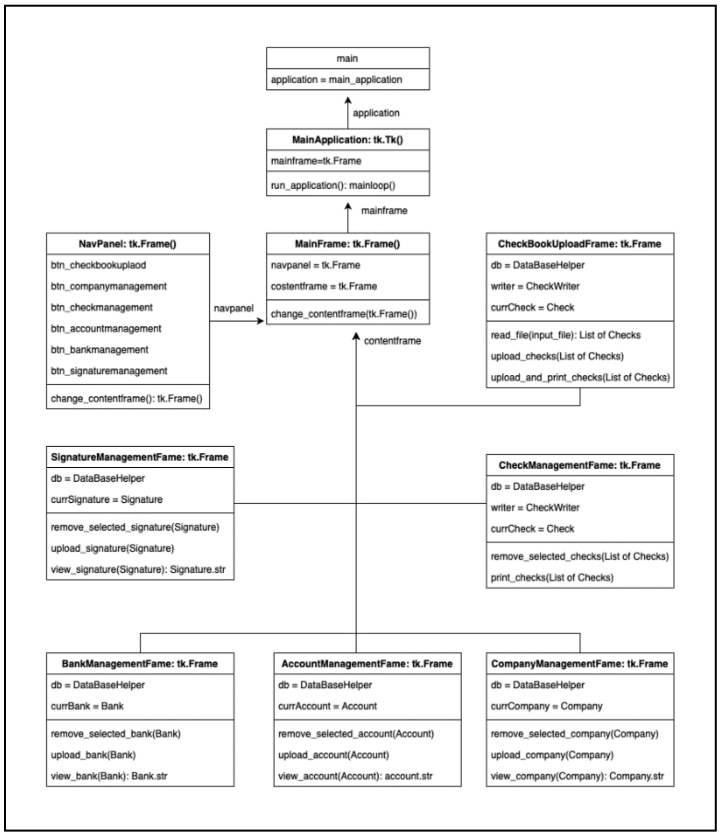

main
application = main_application
application
MainApplication: tk.Tk()
mainframe=tk.Frame
run_application(): mainloop()
mainframe
NavPanel: tk.Frame()
btn_checkbookuplaod
btn_companymanagement
btn_checkmanagement
btn_accountmanagement
btn_bankmanagement
btn_signaturemanagement
change_contentframe(): tk.Frame()
navpanel
MainFrame: tk.Frame()
navpanel = tk.Frame
costentframe = tk.Frame
change_contentframe(tk.Frame())
contentframe
CheckBookUploadFrame: tk.Frame
db = DataBaseHelper
writer = CheckWriter
currCheck = Check
read_file(input_file): List of Checks
upload_checks(List of Checks)
upload_and_print_checks(List of Checks)
SignatureManagementFame: tk.Frame
db = DataBaseHelper
currSignature = Signature
remove_selected_signature(Signature)
upload_signature(Signature)
view_signature(Signature): Signature.str
CheckManagementFame: tk.Frame
db = DataBaseHelper
writer = CheckWriter
currCheck = Check
remove_selected_checks(List of Checks)
print_checks(List of Checks)
BankManagementFame: tk.Frame
db = DataBaseHelper
currBank = Bank
remove_selected_bank(Bank)
upload_bank(Bank)
view_bank(Bank): Bank.str
AccountManagementFame: tk.Frame
db = DataBaseHelper
currAccount = Account
remove_selected_account(Account)
upload_account(Account)
view_account(Account): account.str
CompanyManagementFame: tk.Frame
db = DataBaseHelper
currCompany = Company
remove_selected_company(Company)
upload_company(Company)
view_company(Company): Company.str

Figure 5. Frontend UML

The tk.Tk class is a class provided by the Tkinter library in Python. The Tk object is the Tkinter object responsible for displaying the main application window and set up the basic structure for creating GUI. The Tk object can display tk.Frame objects within them and thus, serve as a background from which the content is pasted upon.

The tk.Frame class is another class provided by the Tkinter library in Python. Instances of the tk.Frame class can create sections within the main window where we can place widgets as needed, such as buttons, labels, list boxes, and entry forms. These classes are called and displayed within the Tk() class.

The Full UML design, as shown Figure 6, illustrates the integration of the fronted architecture with the backend architecture. This diagram provides insight into the flow of the application. The backend classes from Figure 4 are shown at the bottom of the full UML as they supplement the functionality and objects utilized within the frontend of the application. The frontend design classes from Figure 5 are positioned above the backend classes as they inherit the backend classes. The frontend classes are further inherited by the main application, which executes the application using the Tkinter.mainloop() function. This function displays the window where the content is displayed, making it the main function for running the program.

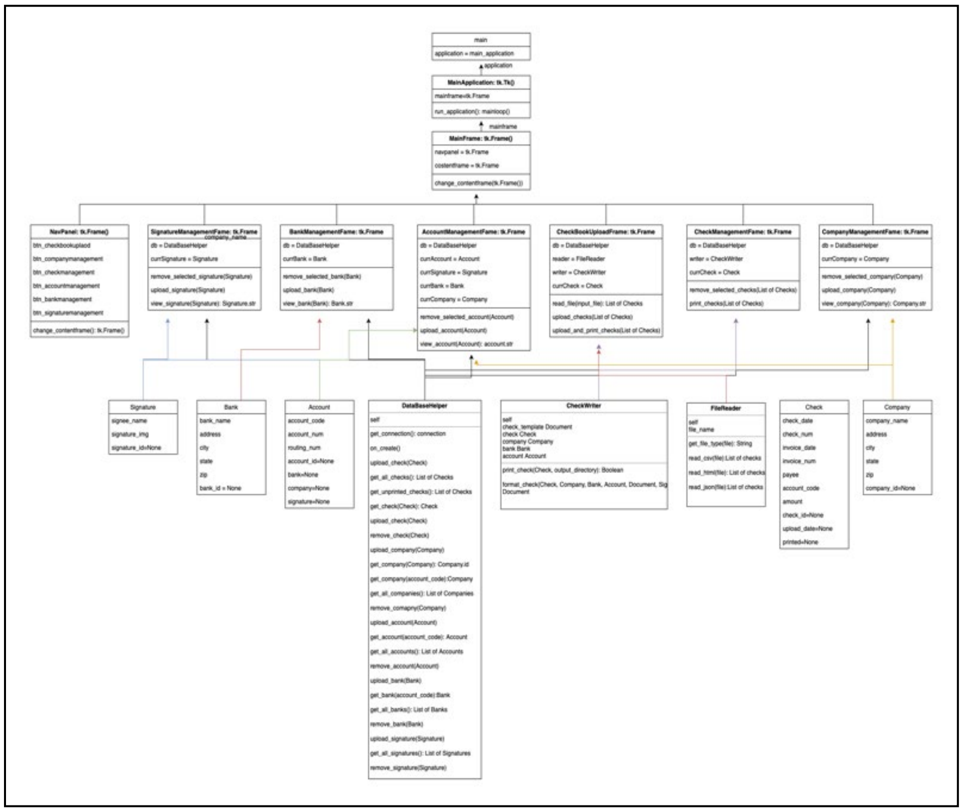

main
application = main_application
application
MainApplication: tk.Tk()
mainframe=tk.Frame
run_application(): mainloop()
mainframe
MainFrame: tk.Frame()
navpanel = tk.Frame
costentframe = tk.Frame
change_contentframe(tk.Frame())
NavPanel: tk.Frame()
btn_checkbookuplaod
btn_companymanagement
btn_checkmanagement
btn_accountmanagement
btn_bankmanagement
btn_signaturemanagement
change_contentframe(): tk.Frame()
SignatureManagementFame: tk.Frame
company_name
db = DataBaseHelper
currSignature = Signature
remove_selected_signature(Signature)
upload_signature(Signature)
view_signature(Signature): Signature.str
BankManagementFame: tk.Frame
db = DataBaseHelper
currBank = Bank
remove_selected_bank(Bank)
upload_bank(Bank)
view_bank(Bank): Bank.str
AccountManagementFame: tk.Frame
db = DataBaseHelper
currAccount = Account
currSignature = Signature
currBank = Bank
currCompany = Company
remove_selected_account(Account)
upload_account(Account)
view_account(Account): account.str
CheckBookUploadFrame: tk.Frame
db = DataBaseHelper
reader = FileReader
writer = CheckWriter
currCheck = Check
read_file(input_file): List of Checks
upload_checks(List of Checks)
upload_and_print_checks(List of Checks)
CheckManagementFame: tk.Frame
db = DataBaseHelper
writer = CheckWriter
currCheck = Check
remove_selected_checks(List of Checks)
print_checks(List of Checks)
CompanyManagementFame: tk.Frame
db = DataBaseHelper
currCompany = Company
remove_selected_company(Company)
upload_company(Company)
view_company(Company): Company.str
Signature
signee_name
signature_img
signature_id=None
Bank
bank_name
address
city
state
zip
bank_id = None
Account
account_code
account_num
routing_num
account_id=None
bank=None
company=None
signature=None
DataBaseHelper
self
get_connection(): connection
on_create()
upload_check(Check)
get_all_checks(): List of Checks
get_unprinted_checks(): List of Checks
get_check(Check): Check
upload_check(Check)
remove_check(Check)
upload_company(Company)
get_company(Company): Company.id
get_company(account_code):Company
get_all_companies(): List of Companies
remove_comapny(Company)
upload_account(Account)
get_account(account_code): Account
get_all_accounts(): List of Accounts
remove_account(Account)
upload_bank(Bank)
get_bank(account_code):Bank
get_all_banks(): List of Banks
remove_bank(Bank)
upload_signature(Signature)
get_all_signatures(): List of Signatures
remove_signature(Signature)
CheckWriter
self
check_template Document
check Check
company Company
bank Bank
account Account
print_check(Check, output_directory): Boolean
format_check(Check, Company, Bank, Account, Document, Sig
Document
FileReader
self
file_name
get_file_type(file): String
read_csv(file):List of checks
read_html(file): List of checks
read_json(file):List of checks
Check
check_date
check_num
invoice_date
invoice_num
payee
account_code
amount
check_id=None
upload_date=None
printed=None
Company
company_name
address
city
state
zip
company_id=None

Figure 6. Full UML

Database Design

The database is responsible for storing essential information used throughout the application. The information stored within the database closely correlates with the class objects present in the backend of the application (as shown in Figure 4).

The database is built using a SQL (Structured Query Language) database, which is a key component for designing and creating relational databases (SQLCourse, 2000). Relation databases establish tables or entities and relationships between them. The ERD (Entity Relationship Diagram) shown in Figure 7 illustrates the database design and connections between working components of the application.

The database stores and manages entities such as company, signature, bank, and check, which align with the objects used in the backend of the application. This connection allows the program to seamlessly upload and retrieve relevant information from the database. As a result, we can obtain required information from the database to print checks.

The company entity table stores company information to be displayed on the check. Each entry in the table is assigned a unique company id as primary key that maintains the separation and uniqueness of the entries. The company table also includes the company name and address information, which are needed on the printed check.

The signature entity table stores the necessary information for displaying a signature on the check. These fields include a signature id as the primary key to ensure distinct entries. Additionally, the signature table includes the signee's name, and a file path to an image file of their signature, which is used on the printed check.

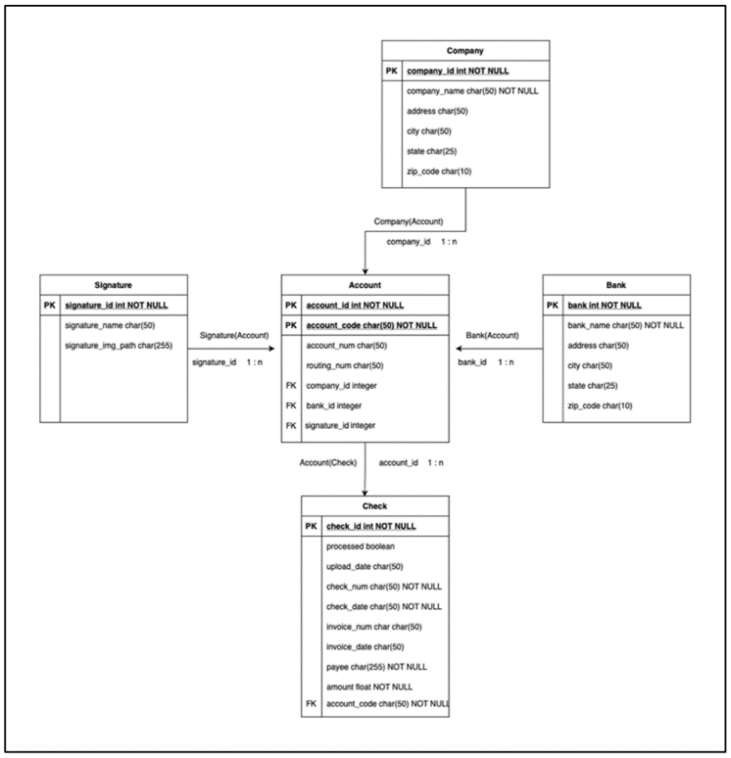


Figure 7. ERD Design

The bank entity table stores banking information used to display bank information on the check. This table includes a unique bank id, the bank name, and the address of its headquarters.

The account entity table stores critical account information. Each account entry has a unique account id, the account's routing and account number, and a unique account code identifier. Additionally, it establishes connections with the Signature, Bank, and Company tables.

The account entity establishes a one-to-many relationship with the bank entity. That means that an account can be linked to only one bank, while a bank can be linked to multiple accounts. The relationship is established via making bank id as a foreign key of the account entity.

The account entity also establishes a connection to the company entity by introducing the company id as a foreign key identifier. This relationship is also a one-to-many relationship where a single company can have multiple accounts linked to it, while an account can only be linked to a single account.

The account entity establishes a many-to-one relationship with the signature entity by including the signature id as its foreign key field. This relationship allows one signature image to provide a signature for many accounts while a single account can only have a single signature tied to it.

The check entity table stores the relevant information of the check object while establishes a one-to-many relationship with the account entity table. The check table uses a check id as its unique primary key. Besides that, it stores the invoice number, invoice date, check number, check date, payee, amount , and account code. The account code field serves as a foreign key. A single check connects to a single account (and therefore inherits the accounts connections), whereas an account can be linked to many Checks.

This design ensures that any check recorded in the database can be associated with its corresponding bank, company, and signature through the established relationships.

User Interface Design

The program's user interface is designed to offer users with all the essential information and handle the required tasks in a user-friendly manner. The UI window design allows users to interact with a content frame that displays key information, while a navigation panel enables them to seamless navigate throughout the application.

The checkbook upload frame (shown in Figure 8) allows users to upload a file and carry out any necessary actions on the checkbook. Once the file is uploaded, the extracted checks populate the label (white space) for user review. Users can choose to either upload the checkbook or upload and print it.

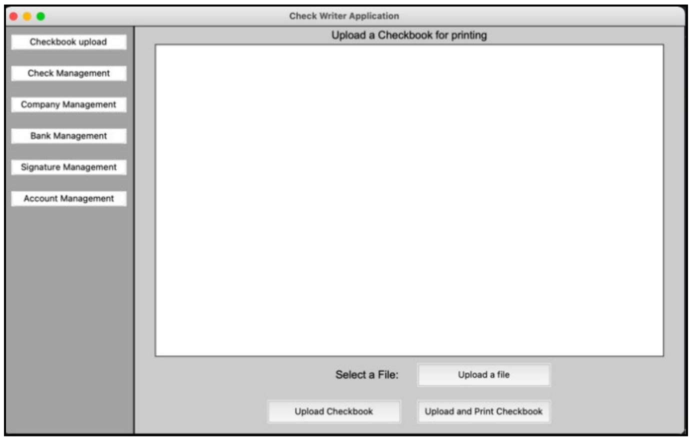
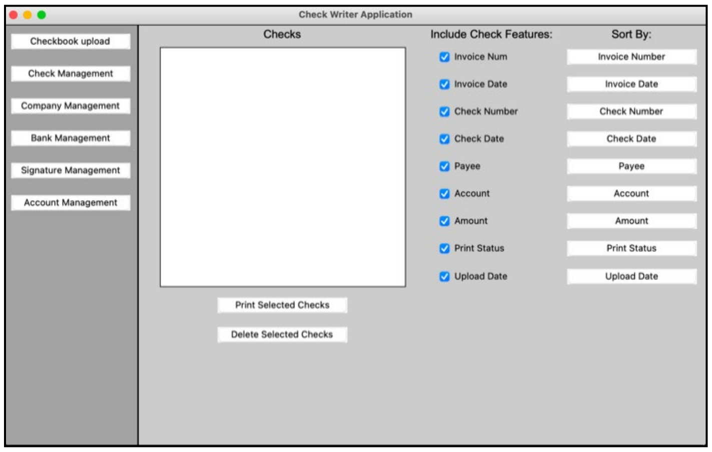


Figure 8. Checkbook Upload Frame

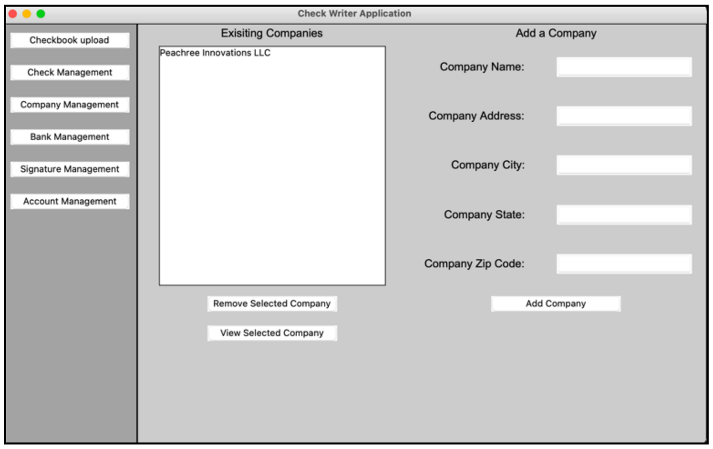
The check management frame (shown in Figure 9) enables users to navigate through all the checks stored in the database. This frame allows users to sort the checks by specific information and customize which components are included in the sorted checks using checkboxes. Users can then select checks from the list-box (white space) and choose to print those checks or delete them accordingly.

The company management frame (shown in Figure 10) allows users to view, add and remove companies from the database. The user can add new companies using the entry fields on the right side of the panel. Additionally, companies that have already been uploaded are displayed on the left side. Users can view their information or remove them from the database permanently.



✓ Invoice Num
✓ Invoice Date
✓ Check Number
✓ Check Date
✓ Payee
✓ Account
✓ Amount
✓ Print Status
✓ Upload Date

Figure 9. Check Management Frame



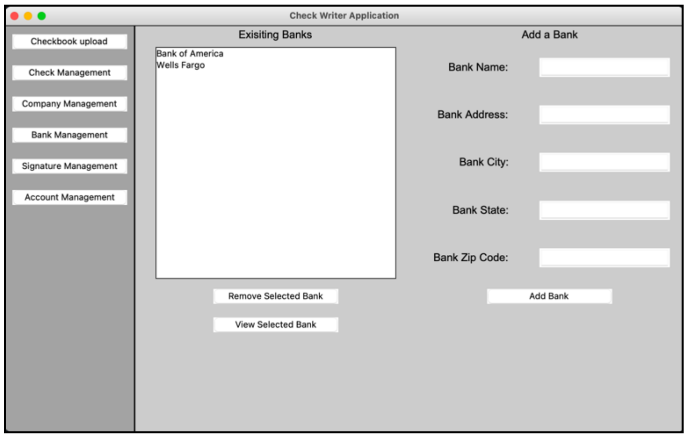
Peachree Innovations LLC

Figure 10. Company Management Frame

The bank management frame (shown in Figure 11) allows the user to interact with the bank objects stored in the database. The right side of the frame allows the user to add a new bank to the database. While the left side of the frame allows the user to view, or remove existing banks form the database.

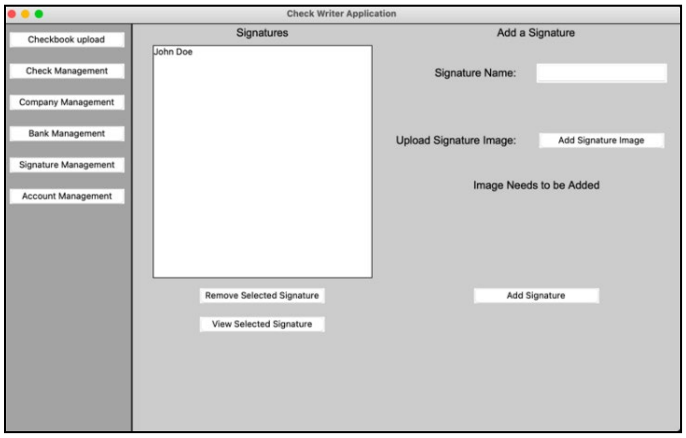
The signature management frame (shown in Figure 12) allows users to upload a signature image, and view or remove stored signatures. The signature objects stored in the database are stored within the list-box for display.

The account management frame (shown in Figure 13) allows users to manage accounts stored in the database, including creating, viewing, and removing accounts. To create additional accounts, users input the required information in the entry fields and select the corresponding bank, company , and signature objects from the dropdown selection menu. Existing accounts are displayed using their account code, but users can expand them to view their full information or remove them from the database.



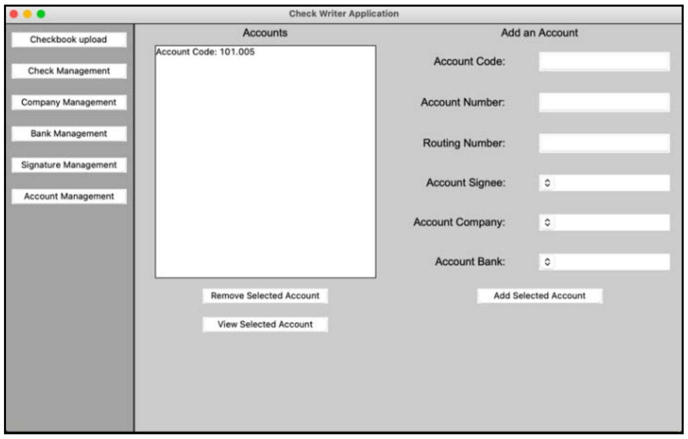
Bank of America
Wells Fargo

Figure 11. Bank Management Frame



John Doe

Figure 12. Signature Management Frame



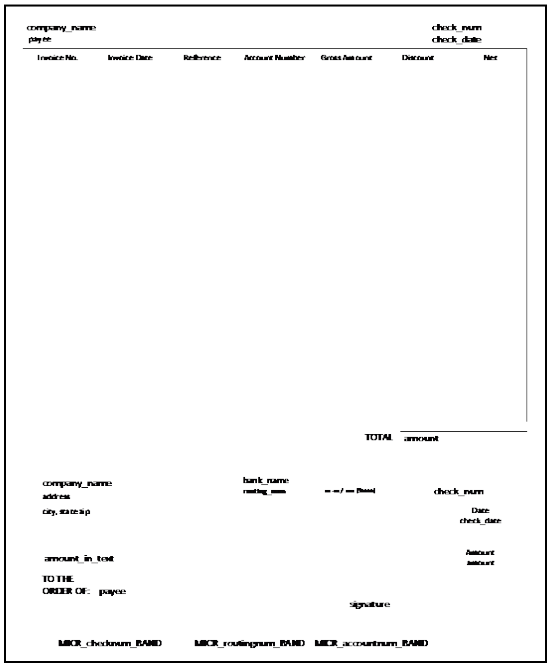
Account Code: 101.005

Figure 13. Account Management Frame

Utilized Tools and Libraries

To Implement the program, a number of development tools were utilized including Python 3.9.5, Microsoft Word, SQLite3, and Tkinter. The program was developed using the Python programming language (version 3.9.5). Several Python libraries were utilized. Python-docx was utilized for manipulating the check template and formatting the checks for printing. Python-docx's .save() method enabled the finished check to be output as a .docx file. The python-num2words library was used for converting the digitized amount into the amount in written format. The python-docx2pdf library allowed for the conversion of output .docx files to .pdf files. The python-json library enabled the program to read .json input checkbook files. The python-csv library allowed the reading of .csv input checkbook files. The python-os library provides a support to work with various operating system modules, such as saving image files, renaming files, and creating directories. The python xml.etree.ElementTree API module was used for reading .xml input checkbook files.

The check template (shown in Figure 14), which is central to the search and replace functionality of the application, was created using Microsoft Word. The template was created using Word's formatting tools and saved as .docx file for manipulation with python-docx. In addition to the creation of the check template, Word also plays a central role in the file manipulation process.


company_name
payee

check_num
check_date

Invoice No.	Invoice Date	Reference	Account Number	Gross Amount	Discount	Net

TOTAL amount

company_name
address
city, state zip

bank_name
routing_num

[illegible]

check_num

Date
check_date

amount_in_text

Amount
amount

TO THE
ORDER OF: payee

signature

MICR_checknum_BAND MICR_routingnum_BAND MICR_accountnum_BAND

Figure 14. CheckTemplate.docx

The check template is formatted to include all the necessary check components identified by their respective keywords. This template is also formatted to ensure proper alignment of the MICR band on the printer paper and within the printing range of commercial printers. Tkinter is a user interface builder that allows the creation of application windows using widgets and various placement methods (Python documentation: Tkinter). It provides a grid feature for organizing widgets on the frame/window, allowing them to maintain proper spacing and structure. Tkinter is bundled with Python 3.

SQlite3 is a built-in database included with Python 3. It allows for the creation of relational databases that are housed ‘on-board’ or at a specific file location and accessed via file location rather than an internet connection (SQLite). SQLite3 was responsible for the creation of the database and store relevant database information.

IMPLEMENTATION

The FileReader class is a central component in the program as it parses information from an input file and extracts embedded check objects. The FileReader class is initialized as an object and can be declared without any input parameters. It utilizes a get_file_type method to determine the input file type. The file type can be determined based on file’s extension, which is obtained using a python-os method called ‘endswith’. The extension is then compared to a list of supported extensions to find a match.

Checks being printed by the print_check() method need to be evaluated to determine whether it is a multi-charge check or singular charge check. A multi-charge check consists of a list of embedded check objects within the checklist (Ex. Check, Check, [Check, Check = 3 Checks]) to represent multiple charges or invoices being paid by a single check. The isinstance() method is used to determine if a check is a multi-charge check or a singularly charged check. If the isinstance() method returns true, it indicates that the check is multi-charged and requires a different formatting method.

Each check in the checkbook has its corresponding account, bank, company, and signature information retrieved from the database using the check’s account code attribute. With this information, the method formats the python-docx document using a series of sub-methods in format_check() function. Once a complete and formatted check being returned from format_check(), the document is then saved to the desired output.

Each check needs to contain a written format amount. To automate this process, the program uses the format_amount_text() method to add the written amount to the check. This method replaces the written amount keyword with a string representation of the check amount in words.
To convert the amount of the check into words requires the utilization of python-num2words library. This library can generate the written representation of numbers passed into the num2words() method. To ensure precise conversion, the method isolates the dollar amount from the total amount. This is achieved by converting the amount from a float to an integer (ex. float(23.33) = int(23)). To separate the decimal or cent amount of the check (which is displayed as a fraction on the check (.33 = 33/100)), the method subtracts the dollar amount from the total amount. The cent amount is then obtained by multiplying the decimal by 100 to convert it to a whole number for display. Once the dollar and cent amounts are isolated, the dollar amount is converted to words and the cent amount is set as a fraction out of 100. These two strings are then combined to generate the amount line in words accordingly.

The check signee’s signature is displayed on the check’s signature line via an uploaded image of their signature. The format_signature() method is responsible for replacing the signature keyword with a signature image. To achieve this, the method removes the signature keyword and utilizes the python-docx run object to insert the signature image into the document. When the image is added to the document, its height is set to half an inch to occupy an appropriate amount of space, and the width will auto-fill based on the image’s height.

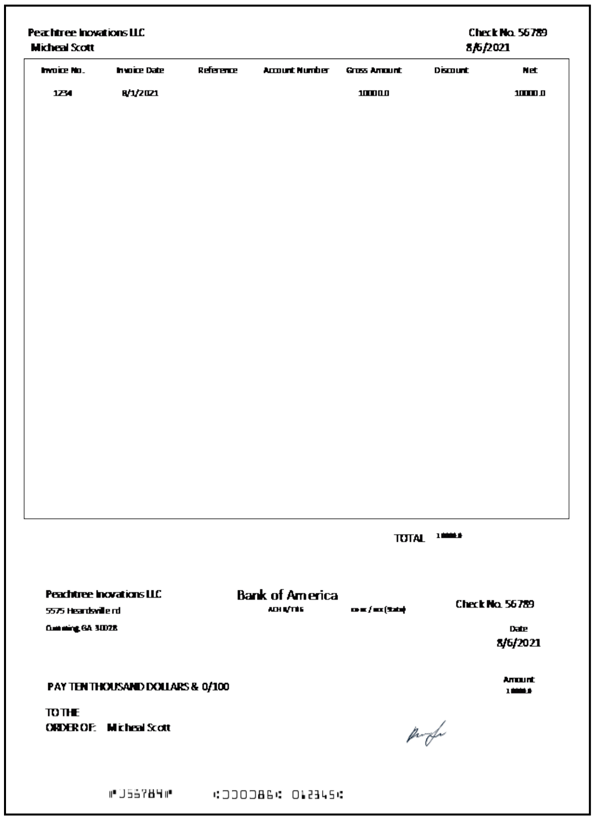
To format the MICR band, the appropriate fields (check number, routing number, account number) use methods to replace MICR keywords with the E-13B font and corresponding symbols. These methods iterate through each digit in the respective number. For each digit, a python-docx run object inserts an image of the E-13B character representing that digit (e.g., ‘1’ inserts the E-13B ‘1’ character). This allows the MICR band to be formatted with the E-13B font and can be placed on the check.

The MICR formatting methods follow the same basic structures for inserting MICR character images. To obtain the MICR image, the formatting methods utilize the get_MICR_char_path() method, which returns the image of the E-13B character based on the character passed to the method using a switch statement.

RESULTS

The FileReader class is used to parse an input file and extract check objects from it. It determines the file type based on the file extension and uses different formatting methods for multi-charge checks and singular charge checks. The program uses the python-num2words library to convert the check amount to words for the written amount on the check. The check signer's signature is added as an image using the format_signature() method. The MICR band is formatted using methods that replace MICR keywords with E-13B font and corresponding symbols. The formatting methods use the get_MICR_char_path() method to obtain images of the E-13B characters.

The program successfully reads data from an input file and outputs the checks from the file as a .docx or .pdf file. These output files contain checks that are properly formatted. Figure 15 displays a formatted check example produced by the program. The output file displays the paying company and payee in the top left corner. The check number and check date are displayed in the top right-hand corner. The memo table is displayed in the center showing the check amount.



Peachtree Inovations LLC
Micheal Scott

Check No. 56789
8/6/2021

Invoice No.	Invoice Date	Reference	Account Number	Gross Amount	Discount	Net
1234	8/1/2021			10000.0		10000.0

TOTAL 10000.0

Peachtree Inovations LLC
5575 Heardsville rd
Cumming, GA 30028

Bank of America
ACH R/T [illegible] [illegible] / [illegible] (State)

Check No. 56789

Date
8/6/2021

PAY TEN THOUSAND DOLLARS & 0/100

Amount
10000.0

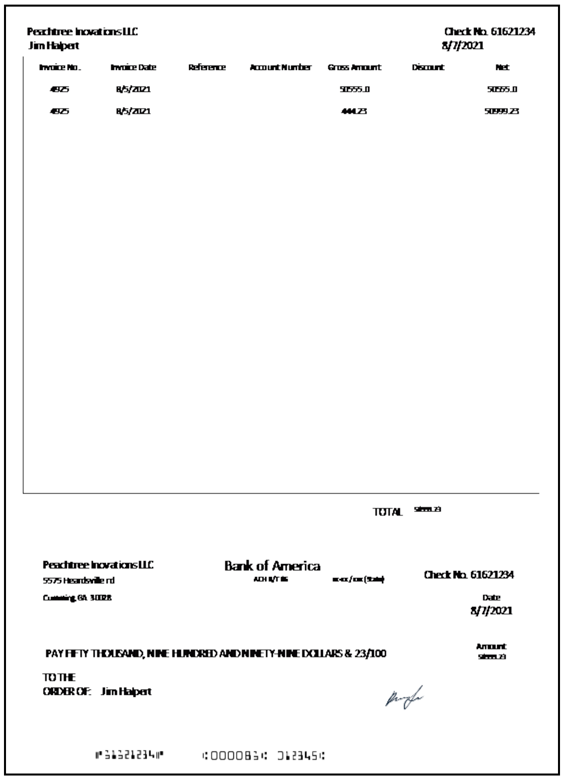
TO THE
ORDER OF: Micheal Scott

[illegible]

⑈355784⑈ ⑆330386⑆ 012345⑆

Figure 15. Output File of a Singular Charge Check

The check on the output file contains all necessary information, and all the required elements are presented and properly formatted. Furthermore, the MICR line is properly aligned at the bottom of the check, with the appropriate symbols (‘on-us’ & ‘transit’) in place.
The output file, as shown in Figure 16, displays a printed multi-charge check example. It differs from the output of a singular charge check (shown in Figure 15), as it displays multiple items within the memo table. These charges amounts are summed to display the total amount.



Peachtree Inovations LLC
Jim Halpert

Check No. 61621234
8/7/2021

Invoice No.	Invoice Date	Reference	Account Number	Gross Amount	Discount	Net
4925	8/5/2021			50555.0		50555.0
4925	8/5/2021			444.23		50999.23

TOTAL [illegible]

Peachtree Inovations LLC
5575 Heardsville rd
Cumming, GA 30028

Bank of America
ACH R/T [illegible] [illegible]

Check No. 61621234

Date
8/7/2021

PAY FIFTY THOUSAND, NINE HUNDRED AND NINETY-NINE DOLLARS & 23/100

Amount
[illegible]

TO THE
ORDER OF: Jim Halpert

[illegible]

⑈61621234⑈ ⑆0000831⑆ 0123451⑆

Figure 16. Output File of a Multi-Charge Check

CONCLUSION

The automation of the business check writing process using Python brings the advanced business check writing capabilities to end users. With the ability to execute a Python file and an ordinary printer, users can automate the check writing process and easily print physical copies of check. The automation allows companies to significantly save resources and time, greatly speeding up the check writing process.

This paper presents an automated process for bulk check writing using the Python programming language and Python-docx library. This process provides a more efficient way of writing and formatting multiple business checks from a single input file, such as .csv, .json, or .xml, and supports output in various file formats. The Python-docx library is used to manipulate a template file using a search and replace technique to create formatted checks with the desired information, meeting MICR E-13B banking specifications. The automated process provides a more efficient, accurate, and speedy way of writing and printing business checks, requiring only a Python program and an ordinary printer.

REFERENCES

Bank of America (2008), MICR Specification Sheet, *Bank of America*. Retrieved from http://www.internationalpcg.com/documents/MICRSpecs.pdf

Chin, F. & Wu, F. (1995), A Microprocessor-Based Optical Character Recognition Check

Reader, *Proceeding of 3rd International Conference on Document Analysis and Recognition*, Montreal,QC, Canada. Retrieved from https://ieeexplore.ieee.org/abstract/document/602066

Commercial Checks Collected Through the Federal Reserve--Annual Data (n.d.), *Board of Governors of The Federal Reserve System*. Retrieved from https://www.federalreserve.gov/paymentsystems/check_commcheckcolannual.htm

Dunn W., Cobb J., Levey A. & Gutman D. (2016), Redletr: Workflow and Tools to Support the Migration of Legacy Clinical Data Capture Systems to Redcap, *International Journal of Medical Informatics*, vol. 93, 2016, pp. 103–110. Retrieved from: https://www.sciencedirect.com/science/article/abs/pii/S1386505616301447

Ilina A. & Pelevanyuk I (2020), Automated Generation of A Book Of Abstracts For Conferences That Use Indico Platform, *Proceeding Of International Conference On Data Analytics And Management In Data Intensive Domains*, Voronezh, Russia. Retrieved from https://ceur-ws.org/Vol-2790/paper28.pdf, Central Russia

Kelechava, B. (2020, June 8), MICR Specifications For Checks In ASC X9 Standards, The ANSI. Retrieved from https://blog.ansi.org/2020/01/micr-specifications-checks-ansi-x9-standards/#gref

Python Progamming (n.d), *Python.org*, Retrieved from https://www.python.org/

Python-Docx (n.d.), *Python-docx*. Retrieved from https://python-docx.readthedocs.io/en/latest/

SQLCourse (2000), *SQLCourse.com*. Retrieved from: http://www.sqlcourse.com/intro.html.

SQLite (n.d.), SQLite.org. Retrieved from: https://www.sqlite.org/index.html.
Tkinter - Python interface to Tcl/Tk (n.d.), *Python 3.9.6 documentation*. Retrieved from: https://docs.python.org/3/library/tkinter.html

Van Steenis H. (1971), The IBM 1275 Recognition System and Its Development, *International Conference on Pattern Recognition in Biological and Technical Systems*, Springer, Berlin, Heidelberg. Retrieved From https://link.springer.com/chapter/10.1007/978-3-642-65175-5_24

Zhang, P. (2008), Ensemble Classifier and Its Application to Image Based MICR Character Recognition, *International Conference on Machine Learning and Cybernetics*, KuMing, China. Retrieved from: https://ieeexplore.ieee.org/document/4620375

QRBD

Quarterly Review of Business Disciplines

May 2023

Volume 10
Number 1

A Journal of International Academy of Business Disciplines
Sponsored by University of North Florida
ISSN 2334-0169 (print)
ISSN 2329-5163 (online)