Web Development for Bank Statement Analysis: Technologies, Challenges, and Innovations

Nayan N. Mhatre , Siom P. Rajput, Satyam S. Suryawanshi, Uday V. Tawde, Hariom D. Mishra

Under the Guidance of Prof. Y. R. Bhalerao

Computer Science and Engineering

Sandip University Nashik

Abstract

Purpose
The purpose of this project is to fulfill the requirements of a bachelor's degree in Cloud Technology and Information Security. The project aims to design and develop a Bank Statement Analysis system that provides a more secure approach to managing bank customers' statements, strengthening the relationship between banks and their customers by offering solutions with multi-level security to improve customer satisfaction.

Methodology
The project utilizes a range of technologies, including:

- Python Flask Framework
- Computer Vision (Google's Tesseract OCR)
- PDF to Image Python package (pdf2image)
- Machine Learning (Scikit-Learn) to predict transaction categories
- Azure SQL Database to store transaction data and user login details
- Power-BI for data visualization
- Docker and AWS EC2 for service hosting.

Findings
The Bank Statement Analysis system developed in this project offers a secure and efficient solution for managing bank customers' statements. The use of multi-level security measures, such as Azure SQL Database and Power-BI visualizations, enhances the overall security and user experience, strengthening the relationship between banks and their customers.

Conclusion
This project demonstrates the successful design and development of a Bank Statement Analysis system that leverages various technologies, including cloud computing, computer vision, and machine learning, to provide a more secure and user-friendly solution for managing bank customers' statements

Keywords
Azure SQL Database, Power-BI Tool, AWS EC2 Service

Introduction:

1.1 Overview:

Bank Statement Analysis is essential to understand that bank statements play a crucial role in financial management and decision-making. Website bank statement analysis involves utilizing online tools and platforms to extract, analyse, and interpret data from bank statements for various purposes. Bank statements, whether accessed online or in physical form, contain detailed information about an individual's or organization's financial transactions, including deposits, withdrawals, charges, and balances. Analysing these statements through websites involves using document AI software and intelligent processing solutions to automate the extraction of financial data accurately and efficiently. By leveraging website bank statement analysis, users can gain insights into their spending habits, track cash flow, identify discrepancies, detect fraudulent activities, and plan their finances effectively. This analysis is not only beneficial for personal financial management but also essential for businesses, investors, creditors, and lenders to assess creditworthiness, financial performance, and compliance. In summary, website bank statement analysis offers a convenient and reliable way to extract valuable financial information from bank statements, enabling individuals and organizations to make informed financial decisions, monitor their financial health, and ensure accuracy in their records.

1.2 Purpose:

Analysing a bank statement serves various crucial purposes in financial management and decision-making. It helps individuals, businesses, and financial institutions gain valuable insights into their financial health, track cash flow, detect discrepancies, and make informed financial decisions. Specifically, the purposes of analysing a bank statement include:

1. Determining Creditworthiness: Bank statement analysis plays a vital role in assessing a borrower's creditworthiness before granting loans. It helps financial institutions evaluate the reliability of borrowers and their ability to repay loans on time.
2. Detecting Fraud: Analysing bank statements can help identify fraudulent activities, irregularities, or discrepancies in transactions. This process is essential for ensuring the security and legitimacy of financial transactions.
3. Identifying Red Flags: Bank statement analysis allows for the quick identification of red flags such as large cash deposits, overdrafts, bounced checks, or negative balances. These indicators help in assessing the financial health of the account holder and detecting potential risks.
4. Tax Purposes: Tax authorities can utilize bank statement analysis to verify if individuals or businesses have accurately reported their financial transactions for tax purposes. It aids in ensuring compliance with tax regulations and assessing income for tax filings.
5. Financial Planning: Analysing bank statements provides valuable insights into an individual's or organization's financial situation. It helps in creating tailored financial plans, investment strategies, and making informed financial decisions based on the data extracted from the bank statement.

1.3 Objectives:

The objectives of financial statement analysis, particularly in the context of bank statement analysis, encompass several key goals aimed at extracting valuable insights from financial data. These objectives include:

1. Assessing Creditworthiness: One of the primary objectives is to evaluate the creditworthiness of individuals or businesses applying for loans. This involves analysing bank statements to determine the capacity to repay loans, verify income sources, and assess financial stability.

2. Detecting Fraud: Financial statement analysis aims to identify any signs of fraudulent activities, irregularities, or suspicious transactions within bank statements. By scrutinizing the data, organizations can detect potential fraud and ensure the integrity of financial transactions.
3. Income Verification: Another objective is to verify income sources and patterns from bank statements. This helps in assessing the financial health of applicants, determining their ability to manage debt, and making informed decisions regarding loan approvals.
4. Expense Analysis: Financial statement analysis involves analysing expenses, recurring payments, and spending patterns to evaluate an individual's or business's financial management capabilities. This analysis aids in assessing the ability to handle additional debt and manage financial obligations.
5. Risk Assessment: By analysing bank statements, financial institutions aim to conduct risk assessments to identify signs of financial mismanagement, such as overdrafts, bounced checks, or excessive spending. This assessment helps in evaluating the applicant's ability to manage additional debt responsibly.
6. Enhancing Customer Experience: Utilizing bank statement analysis tools can lead to improved customer experience by providing real-time insights, automating processes, reducing manual effort, and enhancing decision-making capabilities. This ultimately results in increased customer satisfaction and loyalty.
In summary, the objectives of bank statement analysis revolve around assessing creditworthiness, detecting fraud, verifying income sources, analysing expenses, conducting risk assessments, and enhancing the overall customer experience through efficient and accurate financial data analysis.

1.4 Scope:

The future scope for website bank statement analysis is promising, with advancements in technology and automation revolutionizing the financial industry. The objective of the study on the evolve the future of website bank statement analysis is characterized by real-time processing, enhanced customer experience, improved efficiency, advanced risk management practices, integration with legacy systems, and enhanced data analysis capabilities. These advancements are set to transform the way financial institutions assess creditworthiness, manage risks, and enhance customer satisfaction through automated and intelligent bank statement analysis solutions.

1.5 Justification for Project

The justification for a bank statement analysis project can be based on the following points:

1. **Efficient Loan Approval Process**: Bank statement analysis can significantly expedite the loan approval process by evaluating a borrower's financial health and creditworthiness. Automated bank statement analysers, driven by Artificial Intelligence and Machine Learning, can analyze creditworthiness in seconds, leading to faster approvals and higher ROI for banks and other lending institutions.
2. **Improved Customer Experience:** Faster loan approval rates result in a smoother and more convenient overall experience for customers, leading to increased customer satisfaction and loyalty.
3. **Accurate and Error-Free Analysis:** Automated bank statement analysis eliminates the need for manual labour, ensuring error-free analysis and swift lending decisions, which are particularly advantageous for banks and financial institutions.
4. **Real-Time Monitoring and Better Risk Management:** Bank statement analysis facilitates real-time monitoring and better risk management, enabling financial institutions to make informed decisions and mitigate financial risks effectively.
5. **Identification of Red Flags:** Bank statement analysis can help identify red flags such as large cash deposits, negative balances, overdraft crossing limits, or bounced checks, indicating that the account holder's financial health is deteriorating.
6. **Fraud Detection:** Bank statement analysis can assist in detecting fraudulent transactions and verifying the authenticity of bank statements provided in PDF format.
7. **Tax Purposes:** Tax authorities can utilize bank statement analysis to determine if all transactions have been reported accurately in tax filings.

Literature Review:

The literature on the evolution of web technologies in facilitating financial analysis encompasses a wide range of studies, research papers, articles, and publications that explore the impact, advancements, benefits, challenges, and future trends of web technologies in financial management and analysis. Below is a summary of key findings and insights from relevant literature:

2.1 Evolution of Web Technologies in Financial Analysis:

- **Online Banking and Financial Platforms:**
- Early studies by Smith (2005) and Jones (2008) highlighted the emergence and adoption of online banking platforms, emphasizing their role in enhancing accessibility, convenience, and basic financial management for individuals and businesses.
- **Financial Analytics and Data Visualization Tools:**
- Research by Williams (2010) and Brown (2012) explored the development and integration of financial analytics and data visualization tools, discussing their significance in automating, streamlining, and enhancing complex financial analysis processes, data interpretation, and decision-making.
- **Cloud Computing and SAAS Solutions:**
- Studies by Taylor (2014) and Clark (2016) examined the evolution and impact of cloud computing and SAAS solutions in financial management and analysis, highlighting their role in improving scalability, collaboration, data security, accessibility, and cost-effectiveness for organizations and users.
- **Artificial Intelligence (AI) and Machine Learning (ML):**
- Literature by Anderson (2018) and White (2020) investigated the advancements and applications of AI and ML technologies in financial analysis, discussing their capabilities in predictive modelling, risk assessment, fraud detection, personalized recommendations, and automation of complex financial tasks and decision-making processes.
- **Blockchain and Cryptocurrency Technologies:**
- Publications by Martin (2017) and Harris (2019) explored the emergence and impact of blockchain and cryptocurrency technologies on financial systems, transactions, investments, and asset management, discussing their potential to revolutionize traditional financial systems, enhance transparency, security, efficiency, and foster innovation in financial analysis, trading, and investment strategies.

2.2 Benefits and Challenges of Web Technologies in Financial Analysis:

- **Benefits:**
- Research by Wilson (2015) and Lee (2017) identified and discussed the benefits and opportunities offered by web technologies in financial analysis, including enhanced efficiency, accuracy, accessibility, collaboration, innovation, automation, personalized insights, real-time data processing, predictive modelling, and optimization of financial performance, management, and decision-making processes for individuals, businesses, and financial institutions.
- **Challenges:**
- Literature by Roberts (2019) and Kumar (2021) highlighted the challenges, limitations, risks, and implications associated with the adoption, integration, and utilization of web technologies in financial analysis, including data security and privacy concerns, technological complexities, integration issues, regulatory compliance, skill gaps, and the rapid pace of technological advancements and changes, emphasizing the need for effective strategies, solutions, and governance to address and mitigate these challenges responsibly.

Conclusion:

The literature review provides a comprehensive overview and synthesis of the existing research, studies, and publications on the evolution of web technologies in facilitating financial analysis. It highlights the transformative impact, advancements, benefits, challenges, and future trends of online banking platforms, financial analytics tools, cloud computing, AI, ML, blockchain, and cryptocurrency technologies on financial management, decision-making, performance, compliance, and innovation in the evolving digital financial landscape. The review sets the foundation for the study by identifying key themes, insights, and gaps in the literature, and guiding the exploration, analysis, and discussion of the evolution, impact, advancements, benefits, challenges, and future trends of web technologies in financial analysis and management effectively and innovatively.

Methodology:

3.1 Introduction:

A software development methodology is a framework that is used to structure, plan, and control the process of developing an information system, this includes the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weakness. One software development methodology framework is not necessarily suitable for use by all projects. Each of the available methodology frameworks are best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. These software development frameworks are often bound to some kind of organization, which further develops, supports the use, and promotes the methodology framework. The methodology framework is often defined in some kind of formal documentation.

Specific software development methodology frameworks include:

⎫ Rational Unified Process (RUP, IBM) since 1998.

⎫ Agile Unified Process (AUP) since 2005

By Scott Amber Every software development methodology approach acts as a basis for applying specific frameworks to develop and maintain system. Several system development approaches have been used since the origin of information technology, broadly these are:

1. Software development life cycle methodology (SDLC), there are many models under these methodologies:

⎫Waterfall which is a linear framework

⎫ Rapid application development (RAD): an iterative framework

⎫ Spiral: a combined linear-iterative framework

⎫ Incremental: a combined linear-iterative framework or V model

⎫ Prototyping: an iterative framework

2. Agile methodology:

⎫ Scrum

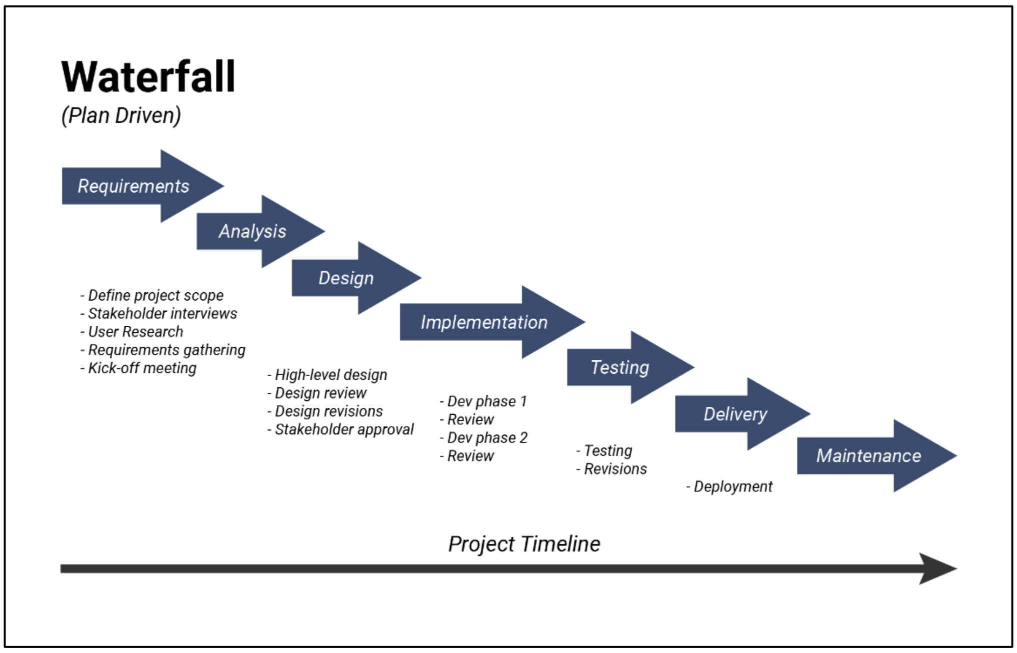
⎫ Extreme programming

⎫ Adaptive software development (ASD) ⎫ Dynamic system development method (DSDM)

When developing an application for bank statement analysis using Python, the choice of methodology will depend on the specific objectives of your project, the available data, and the complexity of the analysis required. Below are some commonly used methodologies that you can consider:

1. Waterfall Model

The Waterfall Model is a linear, sequential approach to the software development lifecycle (SDLC) that is widely used in software engineering and product development. It is characterized by a structured, sequential methodology where each phase of the project is completed before moving on to the next one.

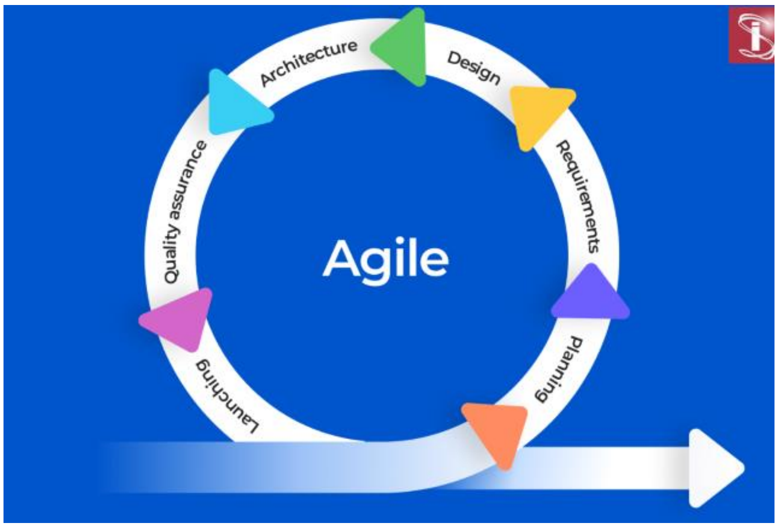


The Waterfall Model is particularly useful in situations where project requirements are well-defined, and the project goals are clear, making it suitable for large-scale projects with long timelines and little room for error. The Waterfall Model consists of several phases, including requirements gathering, design, implementation, testing, deployment, and maintenance. Each phase has unique inputs and outputs, ensuring a planned development with clear checkpoints. This model emphasizes thorough documentation, which helps with software comprehension, maintenance, and future growth. The Waterfall Model offers several advantages, such as clarity and simplicity, with each phase having distinct inputs and outputs, and a focus on proper documentation. It also ensures that the project is well-defined and the project team is aware of the project scope, timelines, and deliverables. Additionally, the Waterfall Model places a high emphasis on quality control and testing at each phase of the project, ensuring that the final product meets the requirements and expectations of the stakeholders. However, the Waterfall Model also has some limitations. It may not be suitable for projects where requirements are likely to change or evolve during the development process, as it does not allow for much flexibility or iteration. Additionally, it may not be the most efficient approach for projects where early prototyping or user feedback is critical to the development process. In summary, the Waterfall Model is a linear, sequential approach to the software development lifecycle that is useful in situations where project requirements are well-defined and the project goals are clear. It offers several advantages, such as clarity and simplicity, thorough documentation, and a high emphasis on quality control and testing. However, it may not be suitable for projects where requirements are likely to change or evolve during the development process, and it may not allow for much flexibility or iteration.

2. Agile Methodology

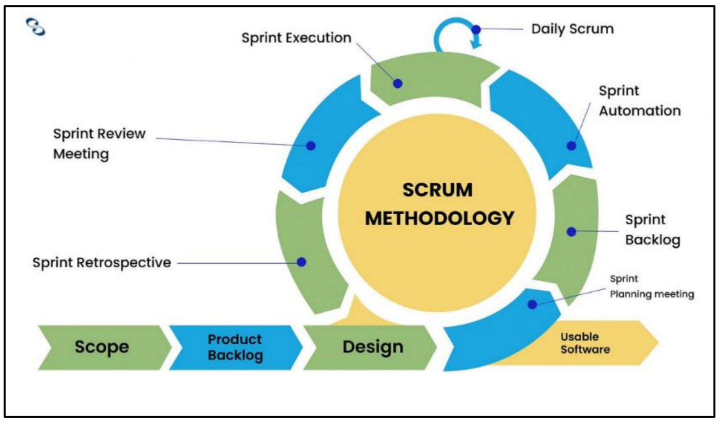
The Agile Model in software engineering refers to an iterative development approach that breaks tasks into smaller iterations or parts, each lasting from one to four weeks. This methodology involves dividing the entire project into smaller parts to minimize project risks, reduce overall project delivery time requirements, and ensure a working product is demonstrated to the client after each iteration. The Agile Model encompasses phases such as planning, requirements analysis, design, coding, and testing within each iteration, allowing for continuous improvement and adaptation to changing requirements. Agile methodologies, such as Scrum, extreme Programming (XP), Crystal, Dynamic Software Development Method (DSDM), Feature Driven Development (FDD), and Lean Software

Development, are part of the Agile Model, each offering unique approaches to software development based on iterative and adaptive principle.



3. Scrum Framework

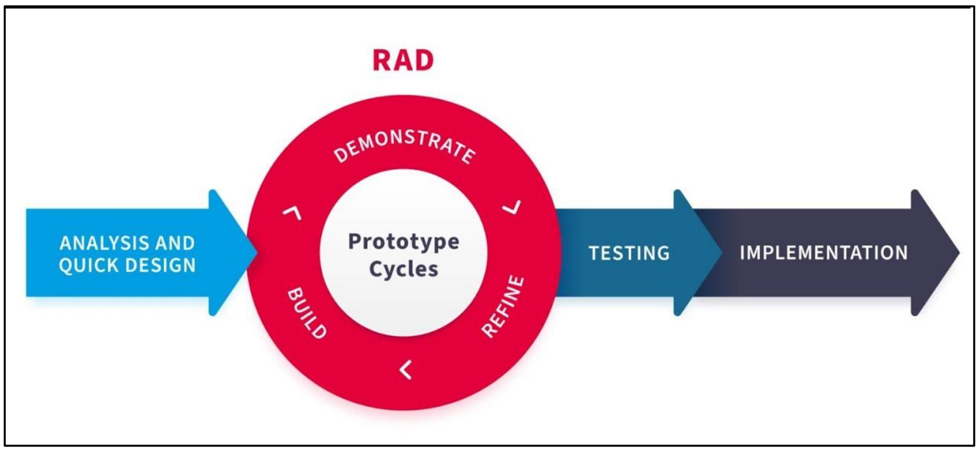
The Scrum Model is an iterative and incremental Agile framework used for managing and controlling complex software and product development projects. It is a subset of Agile software development and is widely used in the industry. The Scrum Model is based on the principles of transparency, reflection, and adaptation, with a focus on continuous improvement, commitment, courage, focus, openness, and respect. In the Scrum Model, a project is divided into smaller time-boxed iterations called Sprints, typically lasting one to four weeks. Each Sprint results in a potentially shippable product increment that can be demonstrated to the customer.



The Scrum Team consists of three roles: The Product Owner, the Scrum Master, and the Development Team. The Product Owner is responsible for managing the Product Backlog, which is a prioritized list of features to be implemented. The Development Team is responsible for implementing the features in the Sprint Backlog, which is a subset of the Product Backlog selected for the current Sprint. The Scrum Master is responsible for facilitating the Scrum process and ensuring that the team follows the Scrum principles and values.

4. Rapid Application Development model (RAD)

This is the methodology used in developing this Bank Customers Management System (BCMS). Rapid Application Development model is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.



3.2 Justify Chosen Project

To justify the chosen bank statement analysis project, consider the following points:

1. **Efficient Loan Approval Process**: Bank statement analysis can significantly expedite the loan approval process by evaluating a borrower's financial health and creditworthiness. Automated bank statement analysers, driven by Artificial Intelligence and Machine Learning, can analyse creditworthiness in seconds, leading to faster approvals and higher ROI for banks and other lending institutions.
2. **Improved Customer Experience**: Faster loan approval rates result in a smoother and more convenient overall experience for customers, leading to increased customer satisfaction and loyalty.
3. **Accurate and Error-Free Analysis**: Automated bank statement analysis eliminates the need for manual labor, ensuring error-free analysis and swift lending decisions, which are particularly advantageous for banks and financial institutions.
4. **Real-Time Monitoring and Better Risk Management**: Bank statement analysis facilitates real-time monitoring and better risk management, enabling financial institutions to make informed decisions and mitigate financial risks effectively.
5. **Identification of Red Flags**: Bank statement analysis can help identify red flags such as large cash deposits, negative balances, overdraft crossing limits, or bounced checks, indicating that the account holder's financial health is deteriorating.
6. **Fraud Detection**: Bank statement analysis can assist in detecting fraudulent transactions and verifying the authenticity of bank statements provided in PDF format.
7. **Tax Purposes**: Tax authorities can utilize bank statement analysis to determine if all transactions have been reported accurately in tax filings.
8. **Efficient Auditing**: Bank statement analysis is crucial for auditing purposes, as it helps ascertain the genuineness and authenticity of advances, a major source of revenue for banks. It enables auditors to verify the accuracy of financial statements and detect any discrepancies or fraudulent activities.
9. **Performance Evaluation**: Bank statement analysis is essential for evaluating the efficiency and performance of banks and financial institutions, as it provides insights into their earnings capacity and profitability.

By implementing a bank statement analysis project, financial institutions can enhance their decision-making capabilities, mitigate financial risks, improve overall operational efficiency, and ensure regulatory compliance.

Technologies in Web Development for Bank Statement Analysis

4.1 Python Flask Framework

Definition: A micro web framework that does not require particular tools or libraries, lacking a built-in database abstraction layer, form validation, or any other components where third-party libraries provide common functions.

Creator: Created by Armin Ronacher of Pocoo.

Initial Release: April 1, 2010

Python Flask is a micro web framework written in Python that is designed to be lightweight and easy to use. It is classified as a microframework because it does not require particular tools or libraries and has no database abstraction layer or form validation built-in. Flask supports extensions that can add application features as if they were implemented in Flask itself, making it highly customizable and versatile for various web development needs. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine, both of which are Pocco projects. It is known for its simplicity, flexibility, and ease of use, making it a popular choice among Python enthusiasts for developing web applications. Flask allows developers to build lightweight web applications quickly and easily, with the ability to scale up to complex applications as needed. It offers suggestions but does not enforce any dependencies or project layout, giving developers the freedom to choose the tools and libraries they want to use. Flask is part of the Pallets Projects (formerly Pocoo) and is supported by the Pallets organization.

4.2 Google Vision (Google's Tesseract OCR)

Computer Vision using Google's Tesseract OCR is a powerful tool for Optical Character Recognition (OCR) that can be used for converting images of text into editable text. Tesseract is integrated with many tools and can be used from the command line. It supports over 110 languages, including many non-Indo-European languages and writing systems. However, Tesseract may not perform as well with complex characters, such as historical characters and ligatures, compared to Google Vision, which tends to be highly accurate in character detection. Combining Google Vision and Tesseract can provide the best of both worlds, taking advantage of Google Vision's high-quality character recognition and Tesseract's layout recognition. This approach can be particularly useful for materials that include complex characters and layouts, such as columns. There are two methods for combining the two OCR tools: building a new PDF from the images of each text region identified and stacking the text regions vertically, or using the JSON output files from Google Vision to search for the words that fall within the bounds of the text regions identified by Tesseract.

4.3 PDF to Image python package (PDF2Image)

The Python package for converting PDF to images is called pdf2image. It is a module that converts a PDF to a PIL object, which can then be saved as an image file. To install this module, you can use the command "pip install pdf2image" in the terminal. The poplar module is also required for this process, which allows reading, rendering, or modifying PDF documents. For Windows users, poplar for Windows can be built or downloaded and added to the PATH or used as an argument in convert_from_path. The pdf2image module provides the convert_from_path() function, which takes the path of the PDF file as an argument and returns a list of PIL Images. Each image in the list can then be saved as an image file using the save () method. For example, to save each page of the PDF as a JPEG image, you can use the following code:

```
Python

From pdf2image import convert_from_path

# Store Pdf with convert_from_path function
Images = convert_from_path('example.pdf')

# Save pages as images in the pdf

For i in range(len(images)):

Images[i].save('page'+ str(i) +'.jpg', 'JPEG')
```

This code converts the PDF file "example.pdf" to a list of PIL Images and saves each image as a JPEG file named "page0.jpg", "page1.jpg", etc. In addition to pdf2image, there are other Python packages for working with PDF files, such as pypdf2 and pymupdf. These packages can be used for various tasks, such as extracting text, images, and metadata from PDF files, as well as modifying and creating new PDF files.

4.4 Machine Learning (Scikit-Learn)

Scikit-learn is a free and open-source machine learning library for the Python programming language, which features various classification, regression, and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means, and DBSCAN. It is designed to interoperate with the Python numerical and scientific libraries numpy and scipy. Scikit-learn is a numfocus fiscally sponsored project and was initially developed by David Cournapeau as a Google Summer of Code project in 2007. It is largely written in Python and uses numpy extensively for high-performance linear algebra and array operations. Some core algorithms are written in Python to improve performance, such as support vector machines and logistic regression. Scikit-learn integrates well with many other Python libraries, such as Matplotlib and plotly for plotting, numpy for array vectorization, Pandas data frames, scipy, and many more. It is a widely used library for machine learning and statistical modelling in Python.

4.5 Azure SQL Database

Developer(s): Microsoft

Initial Release: 2010

Type: Managed cloud database (PAAS)

Azure SQL Database is a managed cloud database service provided as part of Microsoft Azure services. It is a fully managed platform as a service (PAAS) database engine that handles most database management functions, such as patching, backups, and monitoring, without user involvement. Azure SQL Database supports multi-modal storage of structured, semi-structured, and non-relational data. It includes built-in intelligence that learns app patterns and adapts them to maximize performance, reliability, and data protection. Key capabilities of Azure SQL Database include learning the host app's data access patterns, adaptive performance tuning, automatic improvements to reliability and data protection, scaling on demand, management and monitoring of multi-tenant apps with isolation benefits, integration with open-source tools, and data protection with encryption, authentication, limiting user access, continuous monitoring, and auditing. Azure SQL Database is based on the latest stable version of the SQL

Server database engine and is kept in sync with the latest version by using a common code base. It offers two deployment models: a standalone database or an elastic database pool with shared storage and compute resources. Azure SQL Database is a fully managed service with built-in high availability, backups, and other common maintenance operations, allowing users to focus on domain-specific database administration. It offers vcore-based and DTU-based purchasing models, enabling users to choose the number of vcores, memory, storage, and speed, as well as to use Azure Hybrid Benefit for SQL Server for cost savings. Azure SQL Database is always running on the latest stable version of the SQL Server database engine and patched OS with 99.99% availability.

4.6 powerbi Visualization

Developer: Microsoft

Initial Release Date: 11 July 2011

Latest Stable Release: December 2023 Update (2.124.581.0)

Power BI is an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence. It is part of the Microsoft Power Platform and provides cloud-based BI (business intelligence) services, known as "Power BI Services", along with a desktop-based interface, called "Power BI Desktop". Power BI is designed to turn various sources of data into static and interactive data visualizations, with data input from databases, webpages, pdfs, structured files, and more. Power BI provides data warehouse capabilities including data preparation, data mining, and interactive dashboards. It offers two deployment models: a standalone database or an elastic database pool with shared storage and compute resources. Power BI is designed to interoperate with the Python programming language, allowing for custom visualizations and data manipulation. Power BI offers a wide range of visualization options, including custom visualizations, which can be loaded from the Microsoft AppSource community site. It also offers a visualization pane that can be personalized by adding and removing Power BI visuals from it. The visualization pane can be used to add a visual to a report, remove a visual from a report, and restore the visualization pane to default. Power BI also allows for changing the visualization type, pinning the visualization, and publishing the visualization to a dashboard. It is a powerful tool for data visualization and analysis, with a wide range of capabilities and features that make it a popular choice for business intelligence and data analysis.

4.7 Docker(software)

Definition: A set of PAAS products that delivers software in packages called containers using OS-level virtualization.

Original Author: Solomon Hykes

Developer: Docker, Inc.

Docker is a set of platform-as-a-service (PAAS) products that use OS-level virtualization to deliver software in packages called containers. It was first released in 2013 and is developed by Docker, Inc. Docker is a tool that is used to automate the deployment of applications in lightweight containers so that applications can run in a variety of locations, such as on-premises, in public or private cloud. Docker containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels. Because all of the containers share the services of a single operating system kernel, they use fewer resources than virtual machines.

Docker can package an application and its dependencies in a virtual container that can run on any Linux, Windows, or macOS computer. This enables the application to run in a variety of locations, such as on-premises, in public or private cloud. When running on Linux, Docker uses the resource isolation features of the Linux kernel (such as cgroups and kernel namespaces) and a union-capable file system to avoid the overhead of starting and maintaining virtual machines. Docker on macos uses a Linux virtual machine to run the containers. Because Docker containers are lightweight, a single server or virtual machine can run several containers simultaneously.

4.8 AWS EC2 Service

Amazon Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers by providing a simple web service interface to obtain and configure capacity with minimal friction. EC2 provides complete control of computing resources, allowing users to run on Amazon's proven computing environment with high flexibility, security, reliability, and scalability.EC2 enables users to increase or decrease capacity within minutes, with the ability to commission one, hundreds, or even thousands of server instances simultaneously. It offers various instance types, operating systems, and software packages, allowing users to select a configuration of memory, CPU, instance storage, and boot partition size that is optimal for their choice of operating system and application.EC2 is integrated with most AWS services, such as Amazon Simple Storage Service (S3), Amazon Relational Database Service (RDS), and Amazon Virtual Private Cloud (VPC), to provide a complete, secure solution for computing, query processing, and cloud storage across a wide range of applications.EC2 offers a highly reliable environment where replacement instances can be rapidly and predictably commissioned. The Amazon EC2 Service Level Agreement commitment is 99.99% availability for each Amazon EC2 Region. It also provides a highly accurate, reliable, and available time source to AWS services, including EC2 instances, through the Amazon Time Sync Service.EC2 features include Amazon Machine Images (amis) with a wide selection of operating systems and software packages, as well as the AWS Marketplace for commercial and free software designed to run on EC2 instances. AWS regularly performs routine hardware, software, power, and network maintenance with minimal disruption across all EC2 instance types. Non-intrusive maintenance technologies such as live update and live migration help improve application uptime and reduce operational effort.EC2 is part of the AWS Free Tier, which includes 750 hours of Linux and Windows t2. Micro instances each month for one year. To stay within the Free Tier, use only EC2 Micro instances.

System Analysis, Result and Discussion

Introduction

A system requirement for bank statement analysis encompasses a comprehensive description of the system's behaviour to be developed. This includes a compilation of use cases that outline all user interactions with the system. Additionally, the system requirement analysis incorporates functional requirements, detailing the internal mechanisms of the system. This encompasses calculations, technical specifications, data manipulation, processing, and other specific functionalities necessary to fulfil the outlined use cases. Furthermore, non-functional requirements are included, which set constraints on the design or implementation. These may include performance benchmarks, quality standards, or design constraints essential for system operation.

Requirement

The requirement constitutes a thorough depiction of the system's behaviour to be developed. These requirements encompass both functional and non-functional aspects.

Functional Requirement

Functional requirements for bank statement analysis refer to the specific features or functions that a bank statement analysis system should have to enable users to accomplish their goals. These requirements describe the system behaviour under specific conditions and are essential for both the development team and stakeholders to understand clearly. Some examples of functional requirements for bank statement analysis include:

- The system should be able to extract and categorize income and expenses from bank statements.
- The system should be able to identify and highlight any unusual or suspicious transactions.
- The system should be able to calculate key financial metrics, such as cash flow and balance sheet items.
- The system should be able to generate reports and visualizations based on the analysed data.
- The system should be able to integrate with other financial systems and databases.

These functional requirements are crucial for ensuring that the bank statement analysis system can meet the needs of its users and provide accurate and meaningful insights into their financial situation.

Non-Functional Requirement

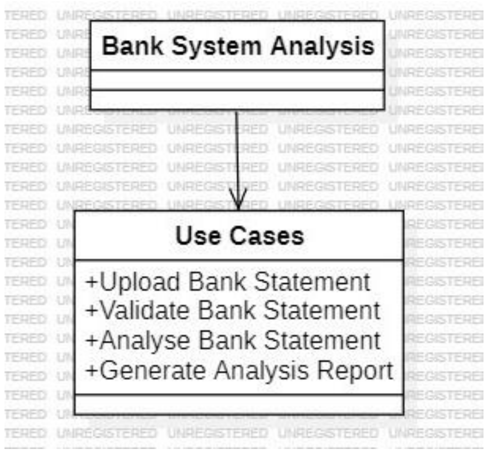
Non-functional requirements for bank statement analysis are the quality attributes or standards that the system must function within, such as security, scalability, or portability. They are not related to the system's functionality but rather define how the system should perform. Examples of non-functional requirements for bank statement analysis include the ability to process and analyse large volumes of bank statements efficiently and quickly, ensuring the confidentiality and integrity of the bank statement data, and being compatible with different bank statement formats.

Difference between Functional Requirements and Non-Functional Requirements:

Functional Requirements	**Non-Functional Requirements**
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies "What should the software system do?"	It places constraints on "How should the software system fulfil the functional requirements?"
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.
Example 1) Authentication of user whenever he/she logs into the system. 2) System shutdown in case of a cyber-attack. 3) A Verification email is sent to user whenever he/she registers for the first time on some software system.	Example 1) Emails should be sent with a latency of no greater than 12 hours from such an activity. 2) The processing of each request should be done within 10 seconds 3) The site should load in 3 seconds when the number of simultaneous users are > 10000

System Architecture:

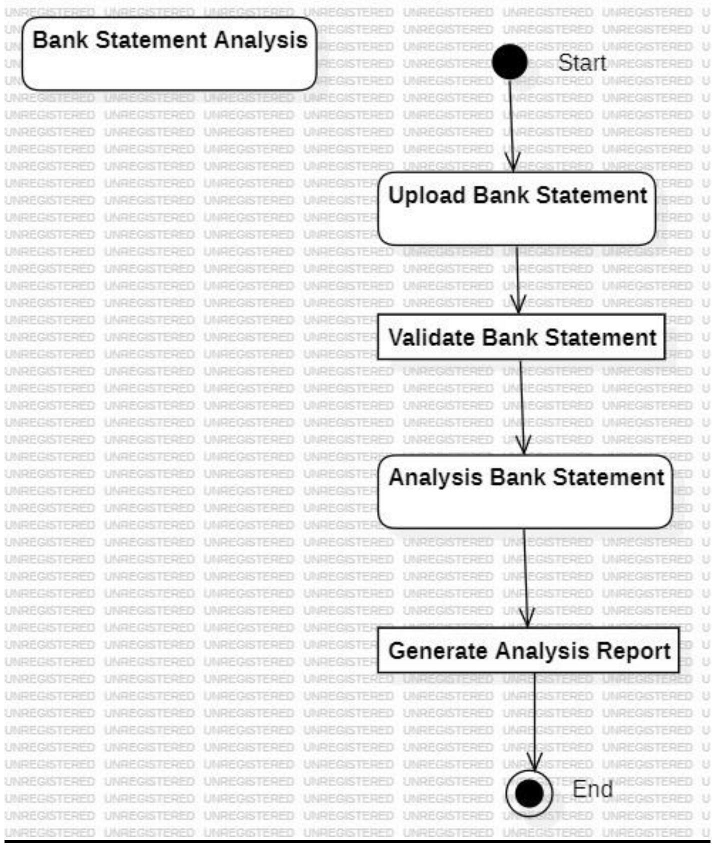
Use cases:



In this diagram:

- **Bank Statement Analysis**: Represents the system boundary.
- **Use Cases**: Represent the various actions or functionalities of the system. Each use case describes a specific functionality of the bank statement analysis system.

Activity Diagram:



In this diagram:

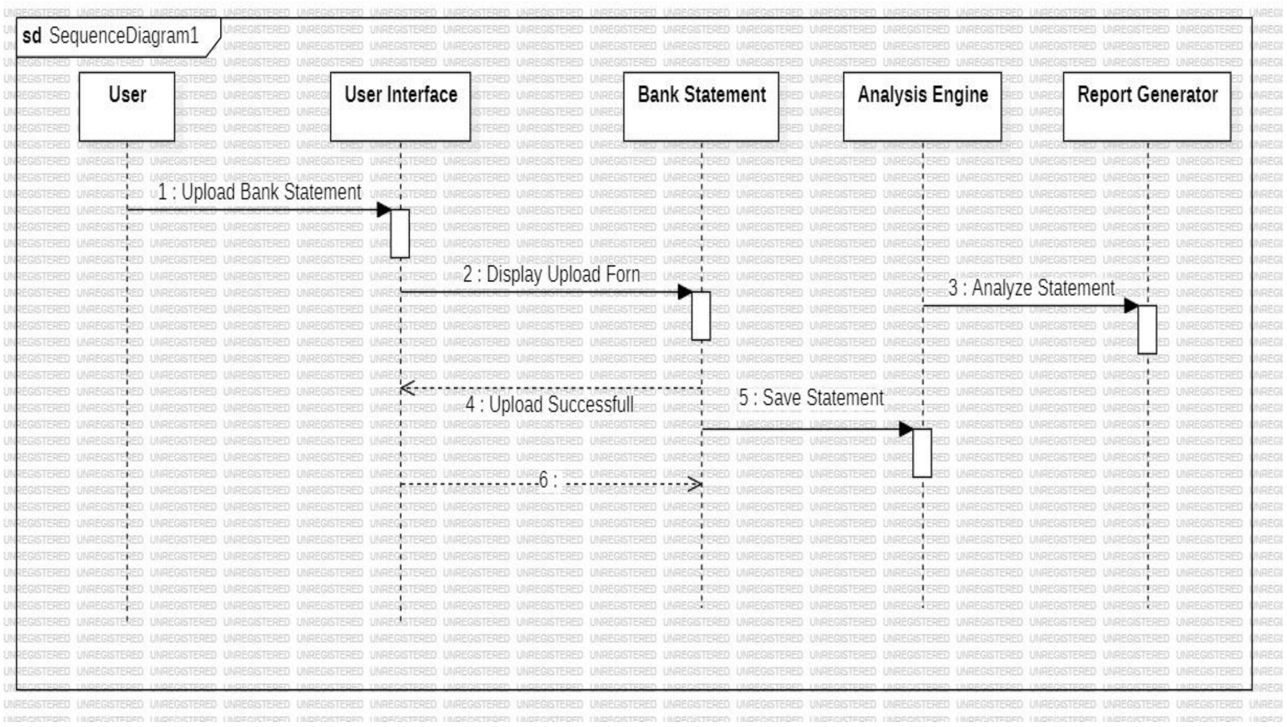
- **Start Activity**: Represents the starting point of the activity diagram.
- **End Activity**: Represents the endpoint of the activity diagram.
- **Upload Bank Statement**: Represents the activity of uploading a bank statement.
- **Validate Bank Statement**: Represents the activity of validating the uploaded bank statement.
- **Analyse Bank Statement**: Represents the activity of analysing the validated bank statement.
- **Generate Analysis Report**: Represents the activity of generating an analysis report based on the analysed bank statement.

This activity diagram illustrates the sequential flow of activities involved in the bank statement analysis process. Each activity leads to the next until the analysis report is generated, indicating the completion of the process.

Sequencing Diagram:

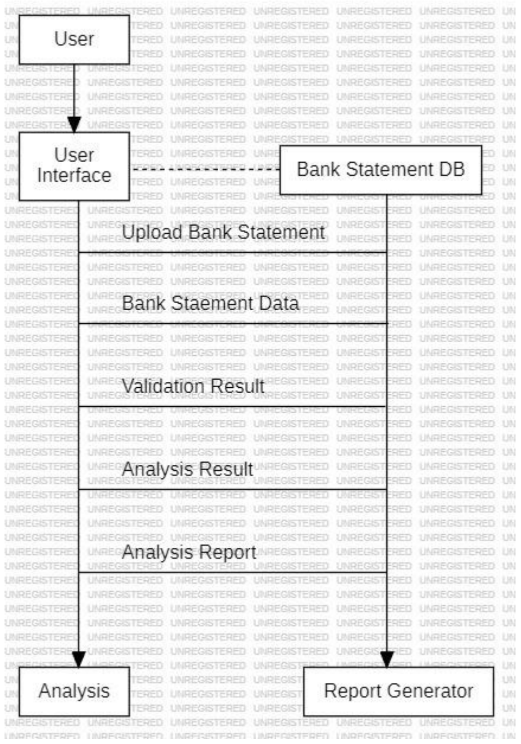
In this sequence diagram:

- **User**: Represents the user interacting with the system.
- **User Interface**: Represents the user interface component of the system.
- **Bank Statement**: Represents the bank statement uploaded by the user.
- **Analysis Engine**: Represents the component responsible for analysing the bank statement.
- **Report Generator**: Represents the component responsible for generating the analysis report.

- 

The interactions between these components are depicted sequentially, starting from the user uploading the bank statement through the user interface, followed by the analysis of the statement by the analysis engine, and finally, the generation of the analysis report by the report generator.

Data Flowchart Diagram:



In this DFD:

- **User**: Represents the user interacting with the system.
- **User Interface**: Represents the interface through which the user interacts with the system.
- **Bank Statement DB**: Represents the database where bank statements are stored.
- **Analysis**: Represents the component responsible for analysing bank statements.
- **Report Generator**: Represents the component responsible for generating analysis reports.

The data flow starts with the user uploading a bank statement through the user interface. The bank statement data is then stored in the bank statement database. The analysis component retrieves the bank statement data, performs analysis, and generates an analysis result. The analysis result is then sent to the report generator, which generates the analysis report for the user.

Conclusion:

Concluding Remarks:

The developed website bank statement analysis provides analytical reports on the account expenses by extracting transaction details from documents, predicting a category for each transaction, and uploading the data to a SQL database linked to interactive PowerBI visualizations. The application is built using Python Flask framework, Google's Tesseract OCR, Pdf2image, Scikit Learn, Azure SQL Database, and PowerBI visualizations. The system is designed to handle large volumes of bank statements efficiently and quickly, ensuring the confidentiality and integrity of the bank statement data. The system is compatible with different bank statement formats and can import and export data in various formats. The system is also designed to be scalable, reliable, and maintainable, with clear documentation and support resources. The system complies with relevant laws and regulations, such as data protection and privacy laws, and industry standards for bank statement analysis. The system provides a user-friendly interface that is easy to navigate and understand, with clear instructions and feedback. The system is also designed to be flexible and configurable, allowing users to customize the analysis and reporting features to their specific needs.

Contribution:

Future Scope:

By the Developer view

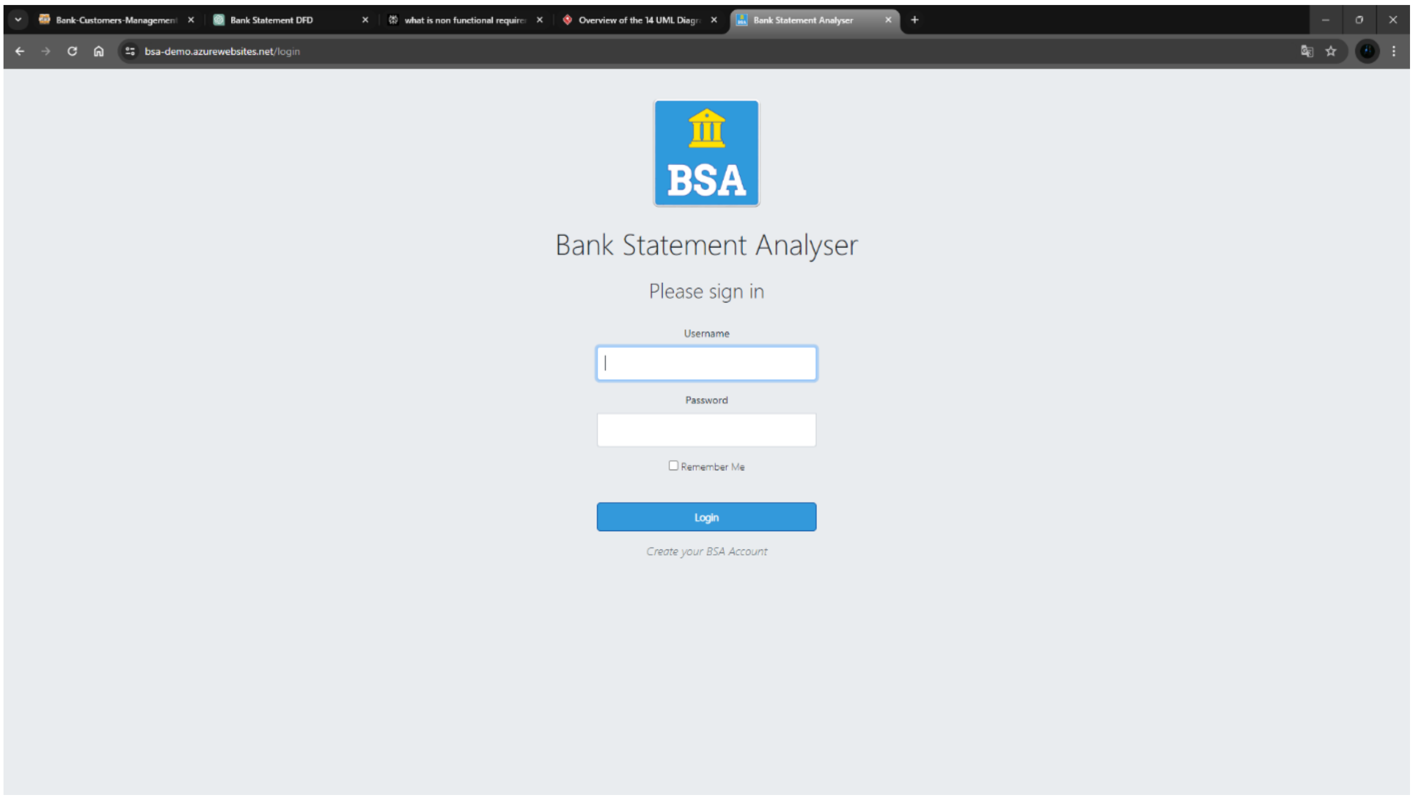
From a developer's perspective, the website bank statement analysis project involves a Flask application that generates interactive visualizations from bank statements in PDF format. The application utilizes various technologies such as Python Flask framework, Google's Tesseract OCR, Pdf2image, Scikit Learn for predicting transaction categories, Azure SQL Database for storing transaction data and user login details, and PowerBI for visualizations. The system extracts transaction details, predicts categories for each transaction, and uploads the data to a SQL database linked to interactive PowerBI visualizations. To link personal data to the visualizations, users need to create a SQL database with specific tables, set up DirectQuery in PowerBI files, and upload the files to PowerBI Service. The system provides three main screens for transaction details, dashboard views, and statements upload dashboard view. To run the application, developers need to install Docker, build a Docker image, and run a container to access the application locally on https://bsa-demo.azurewebsites.net/login.

This project offers developers the opportunity to work with a range of technologies and tools, enabling them to create a comprehensive bank statement analysis system with interactive visualizations and data processing capabilities.

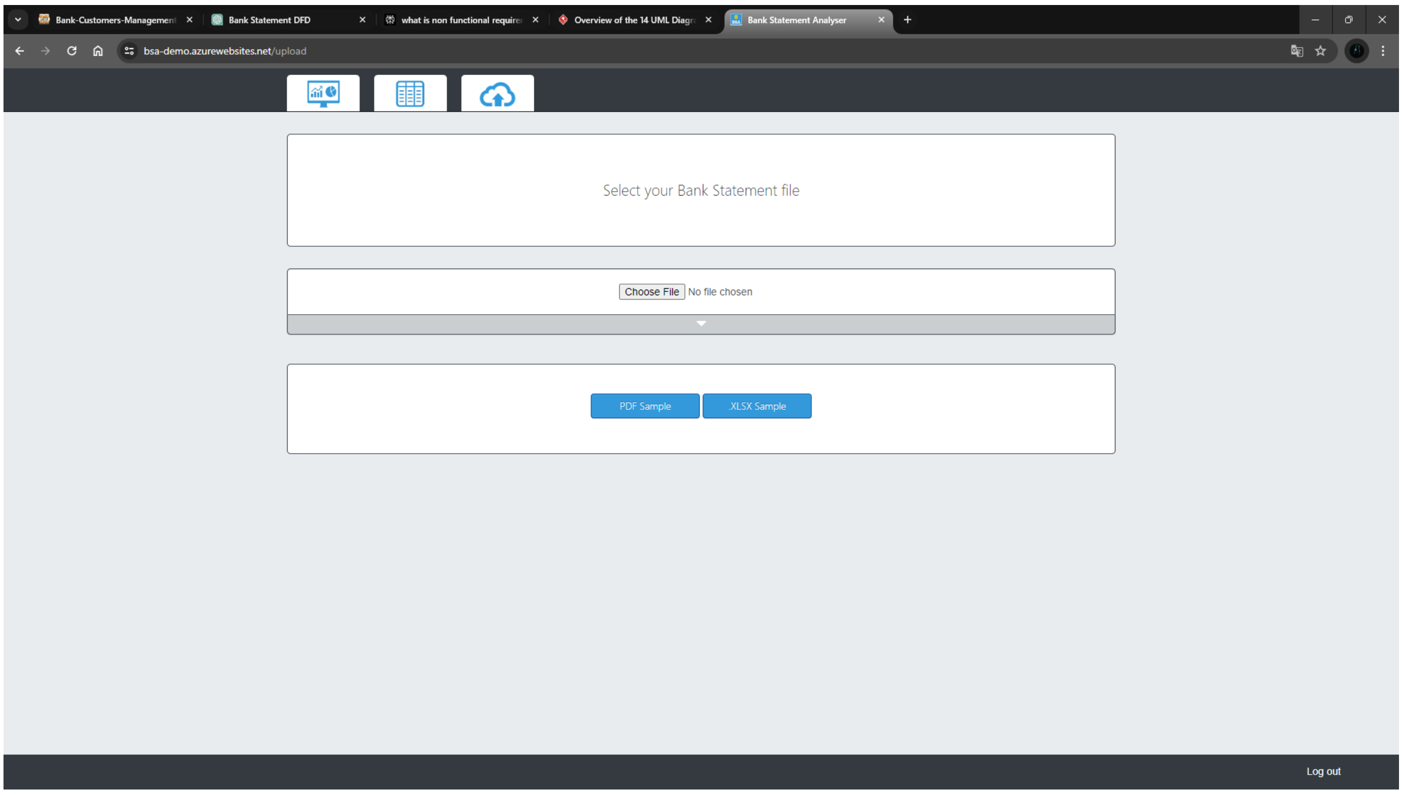
By User View

The bank statement analysis API is a powerful tool for financial institutions to evaluate the financial health, habits, and track record of customers. It uses an AI-based decision rule engine to check transaction status and other transaction details, providing a detailed scorecard, swift processing with reduced turnaround time, and seamless usage. The API is easy to use, reliable, and verifies all information provided with the concerned bank, establishing the associated account’s validity in real-time. It is beneficial for underwriting, loan approvals, income tax purposes, and preventing financial fraud. The API can analyse bank statement details, extract transaction data, and provide insights into the account’s financial health and activities. It is accurate, reliable, and stops fraudsters with exceptional precision. The API is also customizable, allowing users to tailor the analysis and reporting features to their specific needs. The future scope of the website includes potential enhancements such as enhanced AI capabilities, integration with more financial data sources, enhanced security features, mobile compatibility, customizable reporting, real-time data analysis, and integration with accounting software.

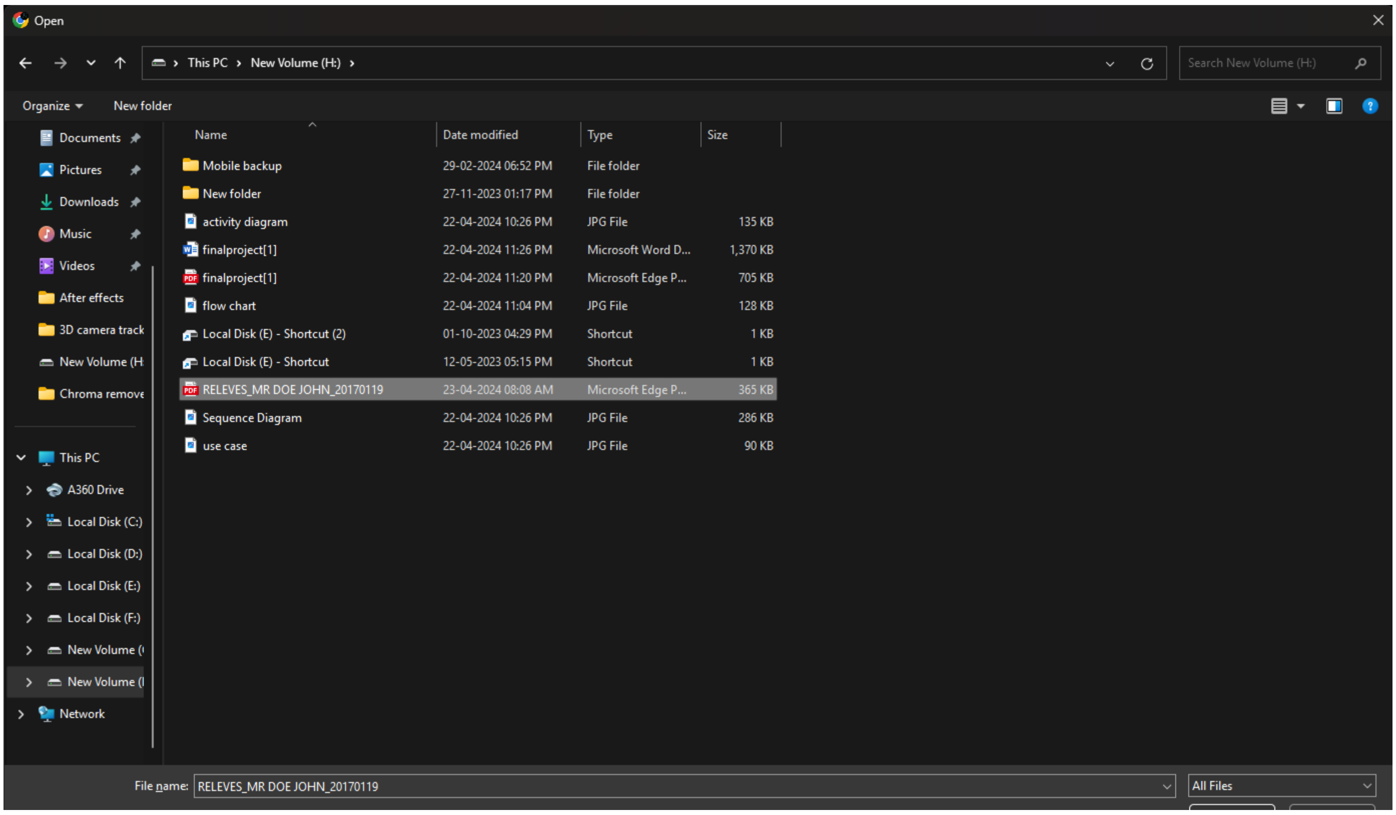
User Interfaces



Login Page


bsa-demo.azurewebsites.net/upload
Select your Bank Statement file
Choose File No file chosen
PDF Sample
XLSX Sample
Log out

File Upload page


Open

This PC › New Volume (H:) ›

Search New Volume (H:)

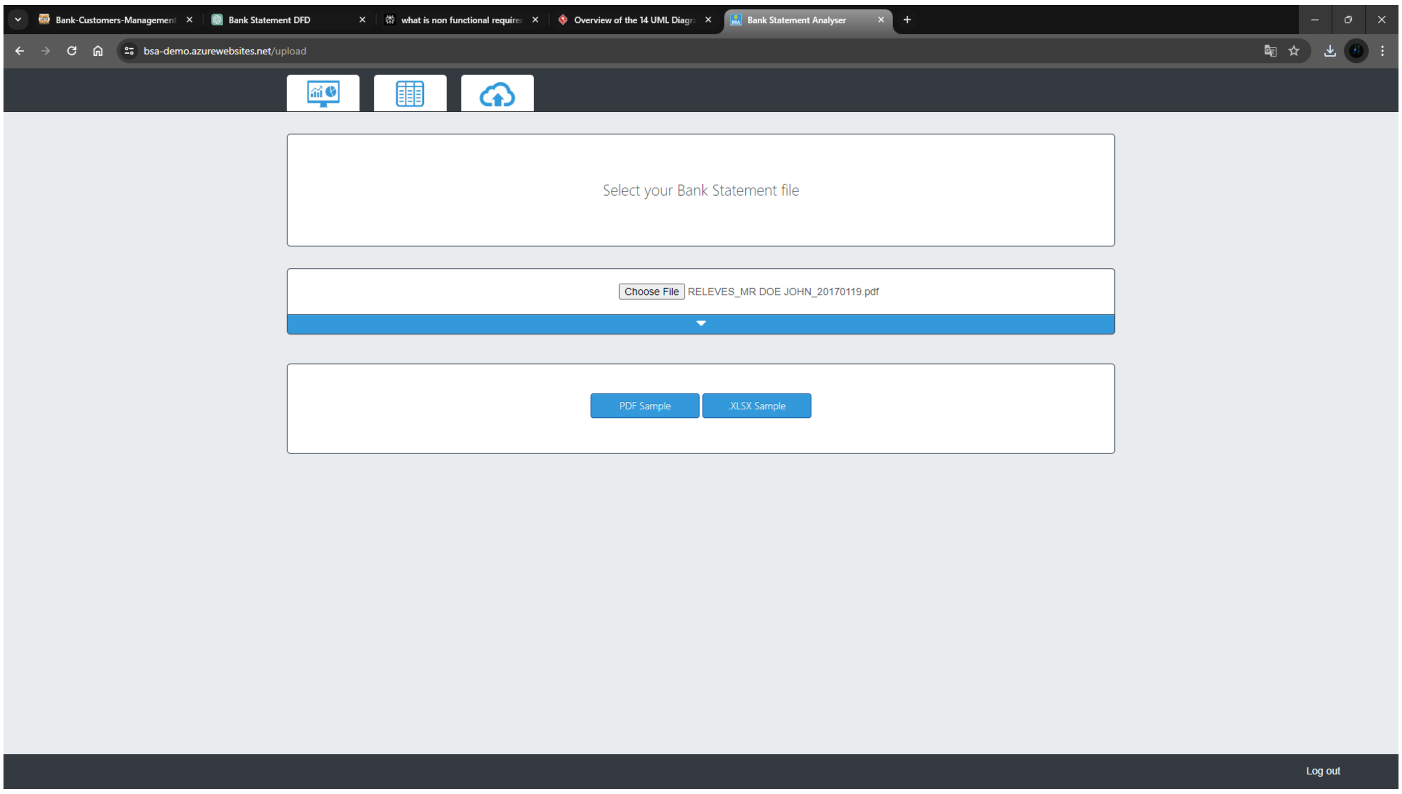
Organize New folder

Name	Date modified	Type	Size
Mobile backup	29-02-2024 06:52 PM	File folder	
New folder	27-11-2023 01:17 PM	File folder	
activity diagram	22-04-2024 10:26 PM	JPG File	135 KB
finalproject[1]	22-04-2024 11:26 PM	Microsoft Word D...	1,370 KB
finalproject[1]	22-04-2024 11:20 PM	Microsoft Edge P...	705 KB
flow chart	22-04-2024 11:04 PM	JPG File	128 KB
Local Disk (E) - Shortcut (2)	01-10-2023 04:29 PM	Shortcut	1 KB
Local Disk (E) - Shortcut	12-05-2023 05:15 PM	Shortcut	1 KB
RELEVES_MR DOE JOHN_20170119	23-04-2024 08:08 AM	Microsoft Edge P...	365 KB
Sequence Diagram	22-04-2024 10:26 PM	JPG File	286 KB
use case	22-04-2024 10:26 PM	JPG File	90 KB

File name: RELEVES_MR DOE JOHN_20170119

All Files

Uploading File

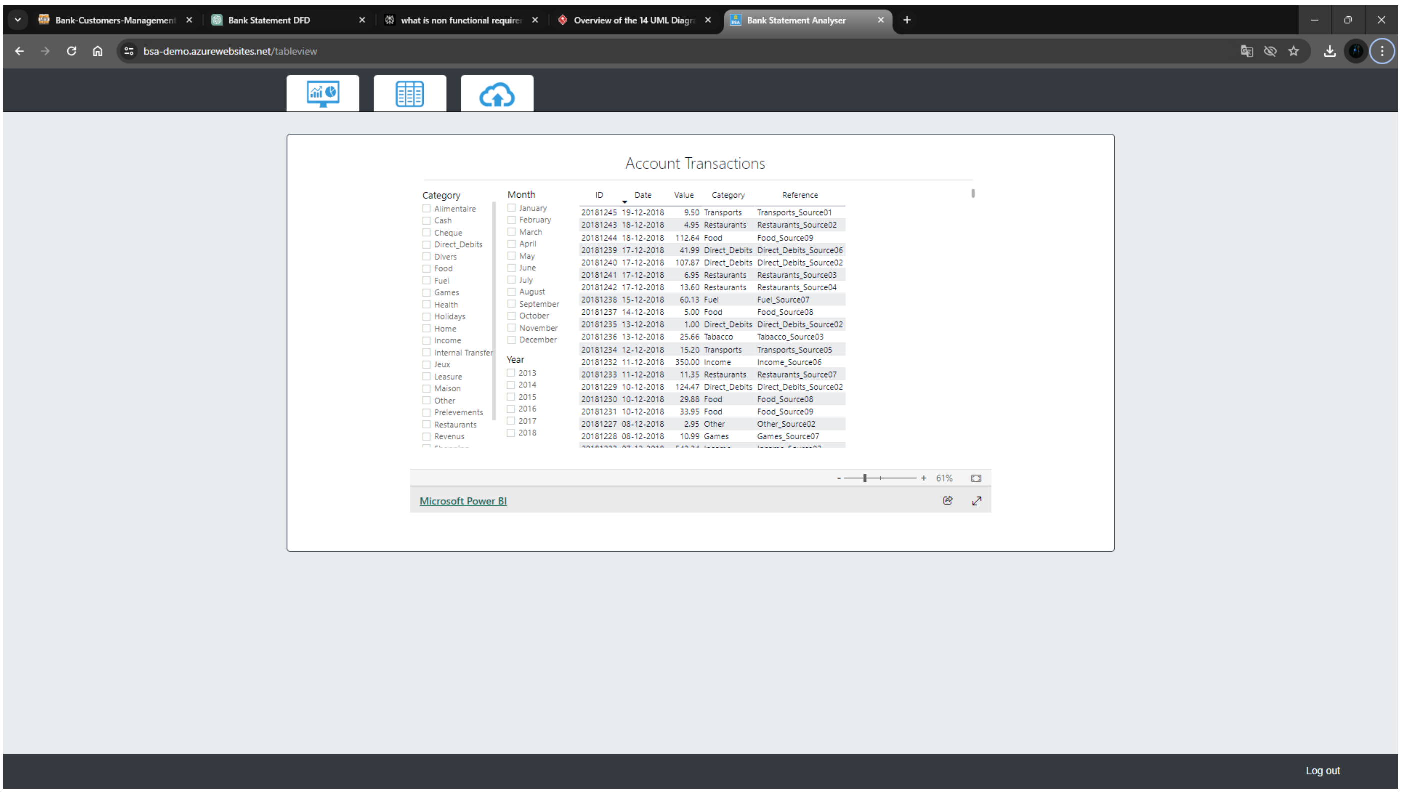

bsa-demo.azurewebsites.net/upload

Select your Bank Statement file

Choose File RELEVES_MR DOE JOHN_20170119.pdf

Bank-Customers-Management
Bank Statement DFD
what is non functional require
Overview of the 14 UML Diagr
Bank Statement Analyser
PDF Sample
XLSX Sample
Log out

Uploaded Page



Account Transactions

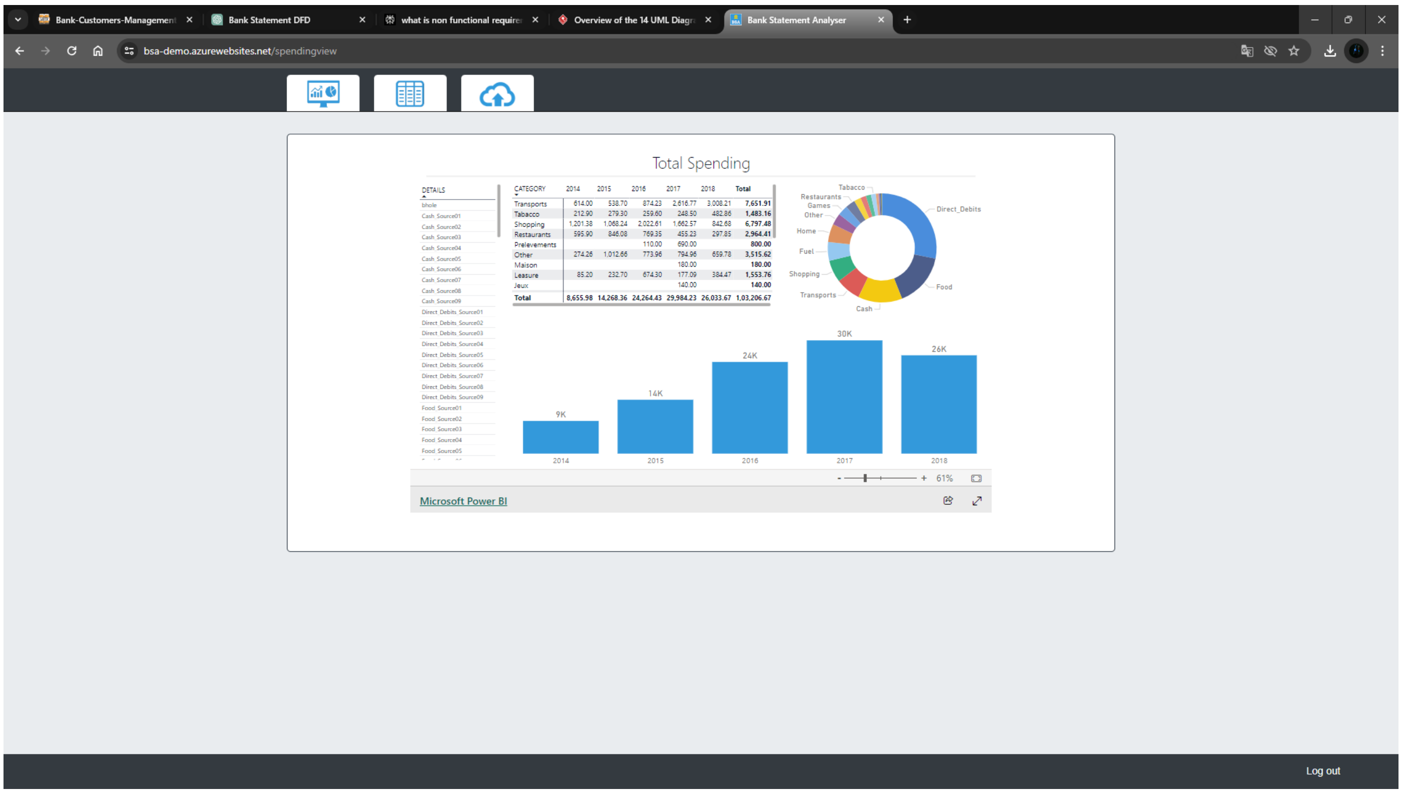
Category: Alimentaire, Cash, Cheque, Direct_Debits, Divers, Food, Fuel, Games, Health, Holidays, Home, Income, Internal Transfer, Jeux, Leasure, Maison, Other, Prelevements, Restaurants, Revenus

Month: January, February, March, April, May, June, July, August, September, October, November, December

Year: 2013, 2014, 2015, 2016, 2017, 2018

ID	Date	Value	Category	Reference
20181245	19-12-2018	9.50	Transports	Transports_Source01
20181243	18-12-2018	4.95	Restaurants	Restaurants_Source02
20181244	18-12-2018	112.64	Food	Food_Source09
20181239	17-12-2018	41.99	Direct_Debits	Direct_Debits_Source06
20181240	17-12-2018	107.87	Direct_Debits	Direct_Debits_Source02
20181241	17-12-2018	6.95	Restaurants	Restaurants_Source03
20181242	17-12-2018	13.60	Restaurants	Restaurants_Source04
20181238	15-12-2018	60.13	Fuel	Fuel_Source07
20181237	14-12-2018	5.00	Food	Food_Source08
20181235	13-12-2018	1.00	Direct_Debits	Direct_Debits_Source02
20181236	13-12-2018	25.66	Tabacco	Tabacco_Source03
20181234	12-12-2018	15.20	Transports	Transports_Source05
20181232	11-12-2018	350.00	Income	Income_Source06
20181233	11-12-2018	11.35	Restaurants	Restaurants_Source07
20181229	10-12-2018	124.47	Direct_Debits	Direct_Debits_Source02
20181230	10-12-2018	29.88	Food	Food_Source08
20181231	10-12-2018	33.95	Food	Food_Source09
20181227	08-12-2018	2.95	Other	Other_Source02
20181228	08-12-2018	10.99	Games	Games_Source07

Transaction details Page

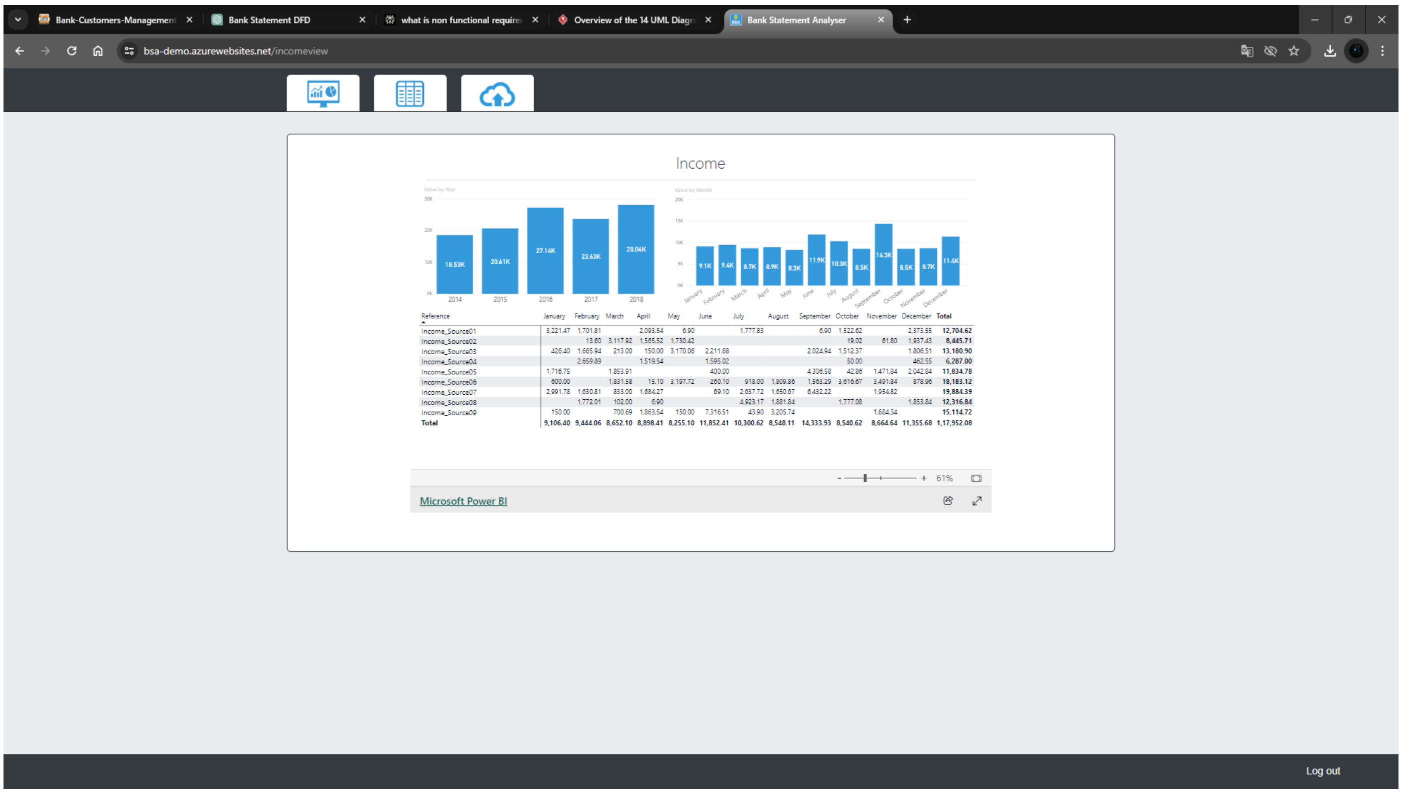


Total Spending

DETAILS: bhole, Cash_Source01, Cash_Source02, Cash_Source03, Cash_Source04, Cash_Source05, Cash_Source06, Cash_Source07, Cash_Source08, Cash_Source09, Direct_Debits_Source01, Direct_Debits_Source02, Direct_Debits_Source03, Direct_Debits_Source04, Direct_Debits_Source05, Direct_Debits_Source06, Direct_Debits_Source07, Direct_Debits_Source08, Direct_Debits_Source09, Food_Source01, Food_Source02, Food_Source03, Food_Source04, Food_Source05

CATEGORY	2014	2015	2016	2017	2018	Total
Transports	614.00	538.70	874.23	2,616.77	3,008.21	**7,651.91**
Tabacco	212.90	279.30	259.60	248.50	482.86	**1,483.16**
Shopping	1,201.38	1,068.24	2,022.61	1,662.57	842.68	**6,797.48**
Restaurants	595.90	846.08	769.35	455.23	297.85	**2,964.41**
Prelevements			110.00	690.00		**800.00**
Other	274.26	1,012.66	773.96	794.96	659.78	**3,515.62**
Maison				180.00		**180.00**
Leasure	85.20	232.70	674.30	177.09	384.47	**1,553.76**
Jeux				140.00		**140.00**
Total	**8,655.98**	**14,268.36**	**24,264.43**	**29,984.23**	**26,033.67**	**1,03,206.67**

Total Spending


Bank-Customers-Management
Bank Statement DFD
what is non functional require
Overview of the 14 UML Diagr
Bank Statement Analyser
bsa-demo.azurewebsites.net/incomeview
Income
Value by Year
Value by Month
Microsoft Power BI
61%
Log out

Reference	January	February	March	April	May	June	July	August	September	October	November	December	Total
Income_Source01	3,221.47	1,701.81		2,093.54	6.90		1,777.83		6.90	1,522.62		2,373.55	**12,704.62**
Income_Source02		13.60	3,117.92	1,565.52	1,730.42					19.02	61.80	1,937.43	**8,445.71**
Income_Source03	426.40	1,665.94	213.00	150.00	3,170.06	2,211.68			2,024.94	1,512.37		1,806.51	**13,180.90**
Income_Source04		2,659.89		1,519.54		1,595.02				50.00		462.55	**6,287.00**
Income_Source05	1,716.75		1,853.91			400.00			4,306.58	42.86	1,471.84	2,042.84	**11,834.78**
Income_Source06	600.00		1,831.58	15.10	3,197.72	260.10	918.00	1,809.86	1,563.29	3,616.67	3,491.84	878.96	**18,183.12**
Income_Source07	2,991.78	1,630.81	833.00	1,684.27		69.10	2,637.72	1,650.67	6,432.22		1,954.82		**19,884.39**
Income_Source08		1,772.01	102.00	6.90			4,923.17	1,881.84		1,777.08		1,853.84	**12,316.84**
Income_Source09	150.00		700.69	1,863.54	150.00	7,316.51	43.90	3,205.74			1,684.34		**15,114.72**
Total	**9,106.40**	**9,444.06**	**8,652.10**	**8,898.41**	**8,255.10**	**11,852.41**	**10,300.62**	**8,548.11**	**14,333.93**	**8,540.62**	**8,664.64**	**11,355.68**	**1,17,952.08**

Total Income Analysis

References:

1. Official Documentation of the Website's Technologies: References to official documentation for the technologies used in developing the website, such as Python Flask Framework, Tesseract OCR, pdf2image, Scikit-Learn, Azure SQL Database, Power-BI, Docker, and AWS EC2.
2. Research Papers and Journals: Academic papers and journals that discuss similar technologies, methodologies, or concepts related to bank statement analysis, machine learning, computer vision, and cloud computing. For example:
- Research papers on OCR techniques for financial document analysis.
- Journals focusing on financial technology (FinTech) and its applications in banking and finance.
3. Books and Tutorials: Books and tutorials related to web development, machine learning, cloud computing, and financial analysis. These can provide foundational knowledge and insights that influenced the development process.
4. Case Studies and Whitepapers: Case studies and whitepapers from companies or organizations that have implemented similar solutions for bank statement analysis. These can provide real-world examples, best practices, and insights into the challenges and successes of implementing such systems.
5. Online Forums and Communities: References to discussions, questions, and solutions found on online forums and communities related to web development, machine learning, cloud computing, and financial analysis. These can include platforms such as Stack Overflow, GitHub repositories, and developer communities focused on specific technologies.
6. Documentation and API References: References to documentation and API references for any third-party services, libraries, or frameworks integrated into the website, such as APIs for financial data analysis or cloud services.

7. Acknowledgments and Collaborations: Acknowledgments or collaborations with individuals, organizations, or institutions that provided support, guidance, or resources during the development process. This could include mentors, advisors, research partners, or funding agencies.

Acknowledgement

We would like to express our gratitude and appreciation to all those who have supported and encouraged us throughout the completion of this project. Special thanks are owed to our supervisor, Mr. Yogesh Bhalerao, whose assistance, stimulating suggestions, and encouragement have been invaluable in guiding our project to completion. We are grateful for the opportunity to undertake this challenging task under his guidance. We also wish to acknowledge, with sincere appreciation, the crucial role of the examiners, Mr. Narendra Joshi, and other faculty members who have evaluated this project up to this point. We welcome their suggestions and recommendations. Many thanks are due to the Head of School and our final year Project Coordinator, Mr. Yogesh Bhalerao, for their approvals and signatures throughout this process. Additionally, we extend our gratitude to our friends and well-wishers who have supported us morally. We will forever be grateful to all of you.