FINAL PROJECT DOCUMENTATION

FOR VERIPAY

(A Third party payment confirmation App)

Prepared for

Group 62

TechyNerds

CONTENTS

Executive Summary..2

Problem Statement..3

Methodology...4

System Design…..6

1. **Process and Iterations...6**
2. **User Personas..6**
3. **Empathy Maps..7**
4. **Storyboards..7**
5. **User Flow..7**
6. **Final Design Solution...7**

Implementation...8

Results and Evaluation...9

1. **Product Designs Results and Key Take-away…...9**

Conclusion...10

References...11

Appendices...12

Supporting Links………………………………………………………………………….12

Technical Report……………………………………...13

EXECUTIVE SUMMARY

The need for a better means of confirming mobile transfer payment during business transactions has become a demand that must be met among small and medium scale enterprises. From recent research conducted by our team, the means of confirming payment during business transactions with customers is through phone calls to the business owners or employers, customer's debit alerts, emails, and through alerts on company phones. Largely, payments were confirmed through phone calls. In other words, transaction alerts for businesses are majorly received by the business owners or employers and only few employees have access to the privilege of receiving transaction alerts. According to the survey, the means of payment confirmation through phone calls to business owners/employers is inefficient as it causes delay, frustration, and trust issues between customers and employees and between business owner/employer and employees and sometimes customers show doctored transaction details to employees or business owner in an attempt to scam them thereby making the process of showing the cashier or employee the transaction detail inefficient.

In a situation where a kidney patient at a dialysis clinic have to make payment before commencement of treatment and such patient is not with cash to make payment and only left with the option of making a mobile app transfer but was delayed due to the clinics standing policy of confirming payment before treatment and the account officer is not around or in a pharmacy store or a petty trader shop that the salesperson has to one way or the other confirm payment which they usually rely on taking customers number or a bank debit alert and this often leads to delay in business process or unsatisfied customers or maybe loss of customer, our product offers to provide payment solution that enables cashier and sales person to instantly confirm these payments without depending on the business owner or accountant in order to carry out business without delay and also reducing the risk of being shown fake debit alerts by customers.

This will be done by partnering with Okra.ng which is an API product that can be outsourced to access financial information from banks. These financial api providers will be asked to tweak our service to just send transaction information without sending the total balance of the account to the employee or cashier, it will be configured on the server to fetch transaction information every 30 seconds and to maintain the traffic.

This will help business owners to get real time payment details that will be made available to the cashiers/employees to confirm their customers payment faster and easier thereby making the business process seamless and to generate more funds and revenue and to retain and attract more customers.

PROBLEM STATEMENT

Based on the user research and pain points, the team developed a problem statement that summarized the key issue faced by businesses. The problem statement was "How might we help businesses to efficiently confirm payments made by customers during business transactions, without delaying the customer or leading to confusion and disputes?"

The need for a better option for a sales person or cashier in a small or medium business enterprise to confirm customer's payment without depending on the business owner is ever increasing and has become a bottleneck in the business process due to other inefficient method of calling the business owner to confirm, take down customer's number or checking the customer's debit alert to confirm the customer's payment.

The main objective of the project is to:

- Build a platform that will provide users with instant access to customer's payment confirmation so as to eliminate waste of time and resolve business issues in cases of emergency.
- Simplify the bank transfer option for businesses and customers of small and medium enterprises and make payment via transfer, the number one channel for payment when conducting business.
- Help cashiers/employees of SMEs to efficiently confirm payments made by customers during business transactions, without delaying the customer or leading to confusion and disputes.

METHODOLOGY

This project was carried out by conducting research through interaction with business owners and employees. This was carried out to find out their business challenges and how technology can be applied to help in providing solutions. A few challenges were cited during our interactive phase. After multiple brainstorming sessions by the members of the capstone group, a peculiar problem and solution was identified.

A need for a secure and efficient payment confirmation was identified as a challenge by most businesses. The team members came up with an idea to build an app to confirm payments transactions for businesses. A problem statement and name were assigned to the project. The required tasks and skill sets for the completion of the project were summarized. Tasks were assigned out to each member of the team based on their skill set and learning track.

The data scientists proceeded to develop questionnaires for a survey that will guide in developing the product and satisfying our intended users. The survey was administered and data was obtained within a period of five days. The obtained data was cleaned and prepared for exploratory data analysis and visualization. The data analytic tools employed include Microsoft Excel, Python and Power BI.

From the visualized data, meaningful insights were generated and communicated to all members of the team. The insights served as a guide in product development, optimization and satisfying our consumers.

Furthermore, appropriate documentation was carried out throughout the capstone Project.

The design process for the platform involved several stages, including user research, creating user personas, identifying pain points, developing problem statements, setting goal statements, and determining the potential impacts of the solution. Also, it involved several iterations to ensure that the platform met the needs of both business owners and customers. The team used feedback from user testing to refine the platform's design, creating a user-friendly interface that streamlined the payment confirmation process.

Frontend Setup and installation.

Node was downloaded to enable the use of React JavaScript Framework. Afterwhich a React App was created and it was named VERIPAY. The text editor used to write the codes is VS code . React app was opened on the VScode created and then proceeded to install and configure

- Tailwind CSS to use for the styling.
- Daisy UI was installed to help with styling as well.
- React Router was also installed to enable navigating from one page of our app to another.

Challenges encountered was because tailwind CSS was use for styling, there was a bit of difficulty in using the same colors the product designers used on figma.

Backend Setup and Installation

The Language used is Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Firstly a new version of node was downloaded and installed from the browser

Direct Link: https://nodejs.org/en/download

Bearing in mind the system specifications before downloading it. Latest LTS Version of Node.js includes npm, so there is no need to download npm again.

To confirm that node.js has been installed, the following command was runned.

prompt(cmd): node --version

When creating a node project, the need to have package.json. npm init is a convenient way of scaffolding your package.json; and there's a need to run it every time you are starting a new project.

Therefore to start a new project run: npm init

To install dependencies into node_modules/directory for the node project run : npm install ....(package name)

To Run Server: Say, the application source is in a file inside a folder, example:

/WTF□PROJECT/index.js

RUN : node /WTF-PROJECT/index.js

OR

RUN: node index.js
You would then be able to run the server using npm by running: npm start (or npm run start).

SYSTEM DESIGN

The design process for the platform involved several stages, including user research, creating user personas, identifying pain points, developing problem statements, setting goal statements, and determining the potential impacts of the solution. Also, it involved several iterations to ensure that the platform met the needs of both business owners and customers. The team used feedback from user testing to refine the platform's design, creating a user-friendly interface that streamlined the payment confirmation process.

Process and iterations

To design the platform, the team started by conducting research to understand the pain points of business owners and customers. User testing was conducted at each iteration to gather feedback and improve the platform. The team used a combination of interviews, surveys, and market research to identify the key issues.

Based on the research, the team created a set of user personas that represented the needs and goals of different types of users. They then developed a set of user stories to capture the key tasks that the platform needed to support.

Next, the team created wireframes and prototypes of the platform. They conducted several rounds of user testing to gather feedback and make improvements. Some of the key iterations included simplifying the payment confirmation process and adding more intuitive features that made the platform more user-friendly.

User Personas

Based on the research, the design team created user personas using Figma that represented different types of users (Business Owners/Employers and Employee). The

personas included information such as user demographics, goals, challenges, and behaviors. The personas helped the team to design the platform with specific users in mind.
https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=57%3A352&t=sQByqDHnXE76oizK-1

Empathy Maps

Empathy maps were created to understand the emotions, thoughts, and behaviors of users during the payment confirmation process. Empathy maps helped the team to design solutions that would address user pain points and create a better user experience.

Storyboards

The design team created storyboards using Figma to visualize the user journey and the potential user flow for the platform. Storyboards helped the team to identify potential user pain points and to create solutions that would address these challenges.
https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=430%3A3310&t=hgjhWIiz3iwW5K7x-1

User Flow

The user flow was created using Figjam to visualize the steps involved in the payment confirmation process. The user flow helped the team identify potential pain points and design solutions to streamline the process of designing a platform for both business owners and customers. So, it shows the users' flow to confirm payments quickly and efficiently.
https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=430%3A11893&t=LqGajeQfxn59zjle-1

Final design solution

The final design solution is a web app service that allows business owners to confirm payments made by customers in real-time. The platform is intuitive and easy to use, allowing business owners to quickly and efficiently confirm payments without any

delays. Customers receive instant confirmation of their payments, reducing their frustration and increasing their satisfaction. The platform also includes additional features, such as the ability to generate invoices and track payments.
The business owner registers on the website and provides bank details and gives or denies access to cashiers/employees in form of cashpoints.
Link to the figma prototype-
https://www.figma.com/proto/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=218%3A3342&scaling=scale-down&page-id=52%3A50&starting-point-node-id=59%3A709&show-proto-sidebar=1

IMPLEMENTATION

An API (REST Api) will be plugged into the Enterprise service bus (ESB) using a wrapper service (url) because of the code language of ESB which is written in Java,C/C++ this will be used to communicate to finacle which is the Core banking solution (CBS).

Our API will directly connect with the Enterprise service bus (ESB) as this feeds into the Commercial banking application (CBA) and this is where the database is that houses the customer’s bank account details, summary and transaction information.

Another alternative can be by

Partnering with Okra.ng which is an API product that can be outsourced to access financial information of customers' bank details which include credit transaction details and statement of account. These financial API providers will be asked to tweak our service to just send transaction information particularly credit alerts without sending the total balance of the account to the employee or cashier, it will be configured on the server to fetch transaction information and to maintain the traffic flow every 30 seconds

RESULTS AND EVALUATION

Product Design Results and key takeaways

Since the launch of the platform, business owners have reported significant improvements in their ability to confirm payments in real-time. This has led to increased revenue and customer satisfaction, as well as reduced manual workloads. Customers have reported increased satisfaction with the platform's ease of use and the ability to receive instant confirmation of their payments. Key takeaways from the project include the importance of conducting thorough user research and testing, simplifying complex processes, and designing for ease of use and user satisfaction.

Threat Model.

A threat model is a structured representation of all the information that affects the security of an application. In essence, it is a view of the application and its environment through the lens of security.

Threat Model Information.

1. Application Name: VERIPAY
2. Application Version: 1.0
3. Description: A third party API to confirm transactions made by clients (between employer and employee). As this is the first version, the functionality will be limited. There will be three users of the application:
    a. Employer
    b. Employee
    c. Bank

Employers will be able to confirm transactions made by clients in the absence of the employer. The banks will use the API as a means for customers to get real-time transaction details.

Assets.

ID: A unique ID is assigned to identify each asset. This will be used to cross-reference the asset with any threats or vulnerabilities that are identified e.g 1, 2

Name: A descriptive name that clearly identifies the asset e.g login info

Description: A textual description of what the asset is and why it needs to be protected

Trust Levels: The level of access required to access the entry point is documented here e.g user with valid login credentials.

Entry Points.

ID: 1, 1.1, 1.2

Name: HTTPS port, Login page

Description: The web page will only be accessible through TLS(Standard Encryption)

Trust Level: Anonymous web user, user with valid login credentials

Trust Levels.

Trust levels represent the access rights that the application will grant to external entities. The trust levels are cross-referenced with the entry points and assets. This allows us to define the access rights or privileges required at each entry point, and those required to interact with each asset.

Anonymous web user- A user who is connected but has not provided valid credentials.
User with valid login credentials- A user who has logged in with valid login credentials.
User with invalid login credentials- A user who has provided invalid login credentials

CONCLUSION

The design team considered the potential impacts of the solution, including the potential for increased revenue, reduced workload for business owners, improved customer satisfaction, and reduced errors and disputes. Also we considered the potential for increased efficiency and streamlined processes for both business owners and customers. We believed that the platform would have a significant positive impact on businesses and their customers. We intend to include confirmation of payment made with cryptocurrency in the next iteration of the product to allow customers make payment using cryptocurrency and also allow the business owners to generate invoices and pay bills.

REFERENCES

https://www.contractscounsel.com/t/us/business-transaction

https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=57%3A352&t=jlkILBmaEmbHLjML-1

https://aws.amazon.com/what-is/restful-api/#:~:text=RESTful%20API%20is%20an%20interface,applications%20to%20perform%20various%20tasks.

https://en.wikipedia.org/wiki/Representational_state_transfer

https://it.ucla.edu/news/what-esb#:~:text=An%20Enterprise%20Service%20Bus%20(ESB,and%20delivers%20it%20to%20another.

https://en.wikipedia.org/wiki/Enterprise_service_bus

https://www.techtarget.com/searchapparchitecture/definition/Enterprise-Service-Bus-ESB

https://www.google.com/search?client=opera&q=wrapper+servicein+api&sourceid=opera&ie=UTF-8&oe=UTF-8

https://www.inspirisys.com/blog-details/Everything-you-need-to-know-about-Finacle-in-Banking/82

APPENDICES

Source code

https://github.com/laradipupo/VERIPAY

SUPPORTING DOCUMENTS, LINKS.

VERIPAY PRD

VERIPAY Project Charter

Veripay competitive analysis.docx

PROBLEM STATEMENT FOR VERIPAY APP

VERIPAY Business Model Canvas.pdf

VERIPAY Lean model canvas poster.pdf

- User persona
  https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=57%3A352&t=sQByqDHnXE76oizK-1
- User Flow
  https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=430%3A11893&t=LqGajeQfxn59zjle-1
- Storyborad
  https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=430%3A3310&t=hgjhWIiz3iwW5K7x-1
- Wireframes
  https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=430%3A12690&t=WqdcLvLfJ7iJThOW-1
- Hi-Fidelty Designs
  https://www.figma.com/file/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=52%3A50&t=4FZXaDdwiwRLsCBL-1
- Link to Prototype
  https://www.figma.com/proto/gh54ESZd6W8id8FpngQsTs/VERIPAY?node-id=218%3A3342&scaling=scale-down&page-id=52%3A50&starting-point-node-id=59%3A709&show-proto-sidebar=1

TECHNICAL REPORT

What is an API?

An application programming interface (API) defines the rules that you must follow to communicate with other software systems. Developers expose or create APIs so that other applications can communicate with their applications programmatically. For example, the timesheet application exposes an API that asks for an employee's full name and a range of dates. When it receives this information, it internally processes the employee's timesheet and returns the number of hours worked in that date range.

You can think of a web API as a gateway between clients and resources on the web.

RESTful API?

RESTful API is an interface that two computer systems use to exchange information securely over the internet. Most business applications have to communicate with other internal and third-party applications to perform various tasks. For example, to generate monthly pay slips, your internal accounts system has to share data with your customer's banking system to automate invoicing and communicate with an internal timesheet application. RESTful APIs support this information exchange because they follow secure, reliable, and efficient software communication standards.

What is REST?

Representational State Transfer (REST) is a software architecture that imposes conditions on how an API should work. REST was initially created as a guideline to manage communication on a complex network like the internet. You can use REST-based architecture to support high-performing and reliable communication at scale. You can easily implement and modify it, bringing visibility and cross-platform portability to any API system.

API developers can design APIs using several different architectures. APIs that follow the REST architectural style are called REST APIs. Web services that implement REST architecture are called RESTful web services. The term RESTful API generally refers to RESTful web APIs. However, you can use the terms REST API and RESTful API interchangeably.

https://aws.amazon.com/what-is/restful-api/#:~:text=RESTful%20API%20is%20an%20interface,applications%20to%20perform%20various%20tasks.

Representational state transfer (REST) is a software architectural style that describes the architecture of the Web. It was derived from the following constraints:

- client-server communication
- stateless communication
- caching
- uniform interface
- layered system
- code on demand

The uniform interface itself creates four interface constraints:

- Identification of resources
- Manipulation of resources through representations
- Self-descriptive messages
- Hypermedia as the engine of application state (HATEOAS)

Although REST is the architecture of the Web, it has not been widely employed throughout the software industry as the architecture for Web services APIs.

https://en.wikipedia.org/wiki/Representational_state_transfer

What is an ESB

An Enterprise Service Bus (ESB) is a type of software platform known as middleware, which works behind the scenes to aid application-to-application communication. Think of an ESB as a “bus” that picks up information from one system and delivers it to another.

https://it.ucla.edu/news/what-esb#:~:text=An%20Enterprise%20Service%20Bus%20(ESB,and%20delivers%20it%20to%20another.

An enterprise service bus (ESB) implements a communication system between mutually interacting software applications in a service oriented architecture (SOA). It represents a software architecture for distributed computing, and is a special variant of the more general client server model, wherein any application may behave as server or client. ESB promotes agility and flexibility with regard to high-level protocol communication between applications. Its primary use is in enterprise application integration (EAI) of heterogeneous and complex service landscapes.

An ESB applies the design concept of a modern operating system to independent services running within networks of disparate and independent computers. Like concurrent operating systems, an ESB provides commodity services in addition to adoption, translation and routing of client requests to appropriate answering services.

The primary duties of an ESB are:

- Route messages between services
- Monitor and control routing of message exchange between services
- Resolve contention between communicating service components
- Control deployment and versioning of services
- Marshal use of redundant services
- Provide commodity services like event handling, data transformation and mapping, message and event queuing and sequencing, security or exception handling protocol conversion and enforcing proper quality of communication service.

ESB as software

The ESB is implemented in software that operates between the business applications, and enables communication among them. Ideally, the ESB should be able to replace all direct contact with the applications on the bus, so that all communication takes place via the ESB. To achieve this objective, the ESB must encapsulate the functionality offered by its component applications in a meaningful way. This typically occurs through the use of an enterprise message model. The message model defines a standard set of messages that the ESB transmits and receives. When the ESB receives a message, it routes the message to the appropriate application. Often, because that application evolved without the same message model, the ESB has to transform the message into a format that the application can interpret. A software adapter fulfills the task of effecting these transformations, analogously to a physical adapter.

ESBs rely on accurately constructing the enterprise message model and properly designing the functionality offered by applications. If the message model does not completely encapsulate the application functionality, then other applications that desire that functionality may have to bypass the bus, and invoke the mismatched applications directly. Doing so violates the principles of the ESB model, and negates many of the advantages of using this architecture.

The beauty of the ESB lies in its platform-agnostic nature and the ability to integrate with anything at any condition. It is important that application lifecycle management vendors truly apply all the ESB capabilities in their integration products while adopting SOA. Therefore, the challenges and opportunities for enterprise application integration EAI vendors are to provide an integration solution that is low-cost, easily configurable, intuitive, user-friendly, and open to any tools customers choose.

https://en.wikipedia.org/wiki/Enterprise_service_bus

What is an enterprise service bus (ESB)?

An enterprise service bus (ESB) is a software platform used to distribute work among connected components of an application. It is designed to provide a uniform means of moving work, offering applications the ability to connect to the ESB and subscribe to messages based on simple structural and business policy rules.

As such, it's a tool that has use in both distributed computing and component integration. The best way to think of this tool is to visualize it as a set of switches that can direct a message along a specific route between application components based on message contents and implementation of business policies.

https://www.techtarget.com/searchapparchitecture/definition/Enterprise-Service-Bus-ESB

In that line, an API wrapper is a language-specific package or kit that encapsulates multiple API calls to make complicated functions easy to use. It helps developers call various APIs without the need for their real-time interaction. As such, wrappers can be used to automate API-reliant processes.

https://www.google.com/search?client=opera&q=wrapper+servicein+api&sourceid=opera&ie=UTF-8&oe=UTF-8

What is Finacle?

Finacle online banking is an advanced, cloud-native internet banking solution that helps onboard, sell, service and engage better. It offers comprehensive functionality and a tailored user interface to retail, SME and corporate customers. The architecture promotes openness and collaboration through a number of APIs.

Finacle is a Core Banking Solution (CBS) that enables banks with digital banking functionalities. The software product, written in Java, C/C++, was released in the year 1999 (Stable version released in 2000). Finacle is widely used by banks across 100 countries, serving over 1.05 billion customers and 1.3 billion accounts. It is the platform of choice for well-established financial institutions, financial technology organizations (Fintech), digital-only banks, and non-financial companies.

Key features of finacle

- Seamless Products Launch-Today's banking customers consume innovation and personalized products as well as services. Finacle software supports banks to take new banking products to market in a quick time. It can be leveraged by various types of banks including micro-finance institutions, private & public sector banks, co-operative banks, fintech companies and non-banking providers.
- Third-party products configuration-Flexibility for banks to set up rule-based third-party products.
- Optimal Mix of Componentization-Componentized structure and API based integration of components.
- Real-Time Processing-Processes and posts transactions in real-time on own and third-party channels of origination.

https://www.inspirisys.com/blog-details/Everything-you-need-to-know-about-Finacle-in-Banking/82

Implementation in Summary

An API (REST Api) will be plugged into the Enterprise service bus (ESB) using a wrapper service (url) because of the code language of ESB which is written in

Java,C/C++ this will be used to communicate to finacle which is the Core banking solution (CBS) Finacle.

Our API will directly connect with the Enterprise service bus (ESB) as this feeds into the Core banking solution (CBS) Finacle and this is where the database is that houses the customer’s bank account details, summary and transaction information.

Another alternative can be by partnering with Okra.ng which is an API product that can be outsourced to access financial information of customers' bank details which include credit transaction details and statement of account. These financial API providers will be asked to tweak our service to just send transaction information particularly credit alerts without sending the total balance of the account to the employee or cashier, it will be configured on the server to fetch transaction information and to maintain the traffic flow every 30 seconds

PROJECT TEAM MEMBERS

First Name	Last Name	Email Address	Student ID Number	Learning Track	Status
Winner	Blessing	Winner.Blessing@womentechsters.org	WTF/23/BT/052	Blockchain Technology	Active
Precious	Olajide	Precious.Olajide@womentechsters.org	WTF/23/CS/B/037	Cyber Security B	Active
Chikosolu	Njoku	Chikosolu.Njoku@womentechsters.org	WTF/23/DS/A/074	Data Science A	Active
Grace	Emeruwa	Grace.Emeruwa@womentechsters.org	WTF/23/DS/B/063	Data Science B	Active
Nkechinyere	Ariom	Nkechinyere.Ariom@womentechsters.org	WTF/23/DS/C/069	Data Science C	Active
Temilade	Olajide	Temilade.Olajide@womentechsters.org	WTF/23/DS/D/075	Data Science D	Active
Blessing	Sunday	Blessing.Sunday@womentechsters.org	WTF/23/PD/A/038	Product Design A	Partially active
Matilda	Anashie	Matilda.Anashie@womentechsters.org	WTF/23/PD/B/073	Product Design B	Partially Active
Reham	Beltagy	Reham.Beltagy@womentechsters.org	WTF/23/PD/C/087	Product Design C	Active
Joy	Ologun	Joy.Ologun@womentechsters.org	WTF/23/PM/A/105	Product Management A	Active
Victoria	Ajayi	Victoria.Ajayi@womentechsters.org	WTF/23/PM/B/100	Product Management B	Active
Stella	Onyekwelu	Stella.Onyekwelu@womentechsters.org	WTF/23/SDB/043	Software Development (Back End)	Partially active
Omolara	Oladipupo	Omolara.Oladipupo@womentechsters.org	WTF/23/SDF/084	Software Development (Front End)	Active