**Ufone Bulk Payments Posting Via**

**Robotic Process Automation (RPA)**

**Developed by: Iman Imtiaz**

**July-Aug 2023**

1. **Project Overview:**
   1. **Project Title:**

Ufone Bulk Payments Posting Via Robotic Process Automation (RPA)

* 1. **Project Purpose:**

This project develops an automated system for Ufone to manage customer Bulk Payments Posting and balance information via Robotic Process Automation(RPA).

1. **Project Charter:** 
   1. **Problem Statement:**

The current billing system at Ufone for Bulk Payment posting is manual and prone to errors, leading to inefficiencies in the billing process. It requires significant human effort and is time-consuming. Additionally, the absence of automation leads to delays in generating bills and increases the risk of inaccuracies in customer billing. Ufone requires a more efficient and accurate billing system to streamline operations and improve customer satisfaction. Moreover, once the payment is posted the customers don’t get any intimation via email or SMS.

* 1. **Project Scope:**

The project mainly covers the following areas.

1. Development of the Ufone Bulk Posting System through web application.
2. Database design and implementation for storing customer information, billing data, and payment records.
3. Billing calculation algorithm to accurately calculate bills based on customer usage, and pricing plans.
4. Customer balance tracking functionality to monitor and update customer balances in real time.
5. Reporting module to generate various reports, including billing statements, and outstanding balances.
6. Integrations to facilitate online payments.
7. SMS feedback for successful billing activation.
8. User documentation for PTCL.
   1. **To be Solution:**

The objective of this project is to implement an automated billing system that enhances operational efficiency, reduces manual effort, minimizes errors, and improves overall customer billing experience. The goal is to provide Ufone with a modern Bulk Posting System that is reliable, scalable, and capable of handling a large customer base.

1. **Project Objectives:**
   1. **Automate Billing Process:**

The project aims to automate the billing process for PTCL by developing a system that can calculate and generate accurate bills for customers.

* 1. **Track Customer Balances:**

The system enables PTCL to track and manage customer balances effectively, ensuring accurate and up-to-date balance information.

* 1. **Improve Customer Experience:**

The billing system enhances the overall customer experience by enabling online payment options.

* 1. **Increase Operational Efficiency:**

The project aims to improve operational efficiency within PTCL by automating the billing process, reducing manual effort, and minimizing human errors.

* 1. **Generate Reports:**

The project aims to improve operational efficiency within PTCL by automating the billing process, reducing manual effort, minimizing human errors, and reducing the need for additional staffing or outsourcing, resulting in cost savings on fees or salaries.

1. **Project Deliverables:**
2. **Billing System Application**

A web-based application is developed to handle the billing process, customer information, and balance tracking. It will provide a user-friendly interface for PTCL staff to manage billing operations and for customers to add balance online.

1. **Database Management:**

The project includes the development and implementation of a database system to store customer data, billing information, and payment records securely using MySQL and Xamp resources.

1. **Billing Calculation Algorithm:**

An algorithm is developed to automatically calculate bills accurately based on customer usage.

1. **Customer Balance Tracking:**

The system enables PTCL to track customer balances in real time, updating them with each payment or billing cycle.

1. **Reporting Module:**

A reporting module is integrated into the system to generate various reports, including billing statements, SMS, and outstanding balances.

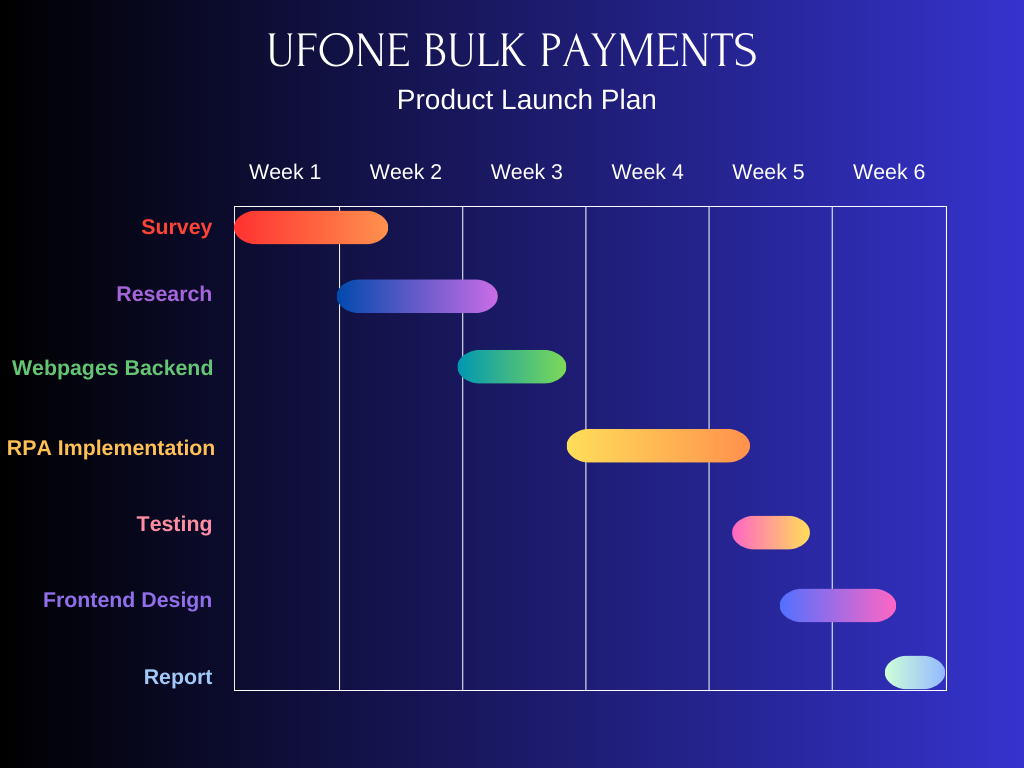
1. **Online Payment Integration:**

The system provides integration to facilitate online payment options for customers, improving convenience and efficiency.

1. **User Documentation:**

Comprehensive documentation is created to guide PTCL on using the billing system effectively.

**Project Implementation Plan (PIP)**



**User Interface (UI) Design:**

I have designed the user interface for the billing system project. The UI consists of several pages, each serving a specific purpose.

1. **Billing System Page (Home Page):**

The "Billing System" page will provide an overview of customer data in a tabular format. It will display customer information such as the **CUSTOMER ID, NAME, EMAIL, PHONENUMBER, PURPOSE, BALANCE, PERIOD(Months), STATUS, ACTION, HISTORY** For customers who are "In Active" and have a balance greater than 500, an "Activate Billing" button will be shown. Clicking this button will trigger the activation process, updating the customer's record in the database. By clicking the “Show History” button agent can see the billing history of any customer at any time.

|  |
| --- |
|  |

1. **Activate Billing Page:**

I have created an "Activate Billing" page where agents can activate billing for customers who are currently "In Active." The page will feature a form with fields for entering the customer's name, phone number, ID, balance, and billing period. Agents will submit the form to initiate the activation process. Upon successful activation, a success message would be sent via SMS notification to the customer's phone number.

|  |
| --- |
|  |

1. **Pay Bill Page:**

To enable customers to pay bills of their accounts, I have design an "Pay Bill" page. This page will include a form with fields for entering the customer's phone number and the desired bill amount. Users will submit the form to update the customer's balance in database. Feedback messages will be displayed to indicate the success or failure of the balance update.

|  |
| --- |
|  |

The UI design will focus on simplicity and usability. I have ensured that the layout is intuitive, with clear labels and input fields for data entry. Styling is applied using CSS to enhance the visual appeal of the pages. The Ufone logo is prominently displayed at the top of each page for branding purposes.

**Database Design:**

In the future, I will design the database schema for the billing system project. The database will be named "ptcl" and will include two tables called "customers." and “billing\_history”.

1. **Customers**

The "customers" table has the following columns:

* ***customerId:*** A unique identifier for each customer.
* ***name:*** The customer's name.
* ***email:*** The customer's email address.
* ***phoneNumber:*** The customer's phone number.
* ***purpose:*** Information on whether it's a personal or business-related account.
* ***balance:*** The customer's account balance.
* ***billing\_Period:*** The billing period in months.
* ***status:*** The customer's status, which can be either "In Active" or "Active."

|  |
| --- |
|  |

The "customer’s" table is an essential component of the database, providing comprehensive information about each customer's profile, financial status, billing period, account status, and related actions.

**Table Structure:**

The structure of the given MySQL “customer” table is as below:

1. **customerId:**
   * **Data Type:** int(11)
   * **Description:** This column serves as the primary key and a unique identifier for each customer in the table.
   * **Properties:** Primary Key
   * **Additional Notes:** The "customerId" column plays a critical role in ensuring each customer has a unique identifier, making it a suitable candidate for primary key usage.
2. **name:**
   * **Data Type:** varchar(50)
   * **Description:** The "name" column stores the full name of the customer.
   * **Additional Notes:** The length limit of 50 characters ensures that names are adequately accommodated while preventing excessive data storage.
3. **email:**
   * **Data Type:** varchar(100)
   * **Description:** The "email" column holds the email address associated with the customer's account.
   * **Additional Notes:** The varchar(100) data type accommodates typical email addresses and supports international character sets.
4. **phoneNumber:**
   * **Data Type:** varchar(15)
   * **Description:** The "phoneNumber" column contains the customer's contact phone number.
   * **Additional Notes:** The varchar(15) data type allows for various phone number formats while keeping the length manageable.
5. **purpose:**
   * **Data Type:** varchar(255)
   * **Description:** The "purpose" column indicates the purpose of the customer's account, providing information on whether it's a personal or business-related account.
   * **Additional Notes:** The varchar(255) data type allows for flexible storage of descriptive purposes and can handle a wide range of account types.
6. **balance:**
   * **Data Type:** int(11)
   * **Description:** The "balance" column reflects the account balance of the customer.
   * **Additional Notes:** The int(11) data type is suitable for storing integer values representing account balances.
7. **billingPeriod:**
   * **Data Type:** int(11)
   * **Description:** The "billingPeriod" column specifies the billing period in months for which the customer has been using the service.
   * **Additional Notes:** The int(11) data type stores the number of billing months as integers.
8. **status:**
   * **Data Type:** varchar(10)
   * **Description:** The "status" column indicates the customer's account status, which can be either "In Active" or "Active."
   * **Additional Notes:** The varchar(10) data type accommodates short status descriptions and supports various character sets.

Each column's data type and length have been chosen appropriately to ensure efficient storage and representation of the respective data. The primary key designation for the "customerId" column ensures the uniqueness and integrity of each customer's record in the table. The table's design appears to be well-structured to accommodate customer information effectively.

|  |
| --- |
|  |

1. **Billing\_history**

The "billing\_history" table, table has the following columns:

* ***customerId:*** A foreign key referring to the customer for whom the billing is recorded.
* ***billingId:*** A primary key representing a unique identifier for each billing record.
* ***billingDate:*** The date and time when the billing was recorded.
* ***billingAmount:*** The amount associated with the billing record.
* ***billingType:*** The type of billing, indicating whether it's conducted online, in-person, or other.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

**Table Structure:**

The "billing\_history" table records billing details associated with customers. It is structured with the following columns:

**1. billingId:**

* + - **Data Type:** int(11)
    - **Description:** This column serves as the primary key and represents a unique identifier for each billing record.
    - **Properties:** Primary Key, AUTO\_INCREMENT

2. **customerId:**

* + - **Data Type:** int(11)
    - **Description:** The "customerId" column serves as a foreign key that references the customer for whom the billing is recorded. It relates to the "customerId" primary key in the "customers" table.
    - **Properties:** Primary Key, Foreign Key

3. **billingDate:**

* + - **Data Type:** datetime
    - **Description:** The "billingDate" column records the date and time when the billing was conducted.
    - **Additional Notes:** The datetime data type ensures accurate storage of date and time information.

4. **billingAmount:**

* + - **Data Type:** decimal(10,2)
    - **Description:** The "billingAmount" column stores the monetary amount associated with the billing record.
    - **Additional Notes:** The decimal(10,2) data type accommodates currency values with precision up to two decimal places.

**5. billingType:**

* + - **Data Type:** varchar(50)
    - **Description:** The "billingType" column indicates the type of billing, such as "Online," "In-Person," or other descriptions.
    - **Additional Notes:** The varchar(50) data type allows for descriptive billing type entries.

The "billing\_history" table maintains a relationship with the "customers" table using the "customerId" foreign key, linking billing records to their respective customers. The primary key for the "billing\_history" table consists of the combination of "billingId" and "customerId," ensuring uniqueness for each billing record per customer. This table structure facilitates the tracking of billing transactions, their associated amounts, types, and dates, offering valuable insights into customer financial interactions.

|  |
| --- |
|  |

The database schema has a straightforward structure, consisting of two tables to store customer information. The customerId column will serve as the primary key for uniquely identifying each customer. I have defined appropriate data types and constraints for each column to ensure data integrity.

**SMS Integration:**

This code focuses on activating billing for customers and sending an SMS notification using the Twilio service. Here's a explanation of the SMS integration part:

1. **Libraries Used:**

The code requires the Twilio PHP SDK to interact with the Twilio API. The **require \_\_DIR\_\_ . '/vendor/autoload.php';** line imports the required libraries.

1. **Twilio Account Configuration:**

The Twilio account SID, authentication token, and Twilio phone number are configured. These credentials are used to authenticate and connect to the Twilio service.

1. **Sending SMS Function (sendSms):**

This function takes the recipient's phone number, the SMS message, and the Twilio account credentials as parameters. It uses the Twilio REST Client to send an SMS.

1. **Customer Details Retrieval:**
   * The code retrieves customer details from the MySQL database based on the provided customer ID (**customerId**).
   * It constructs a SQL query to fetch the customer details from the "customers" table.
2. **Activation Process:**
   * If customer details are found, the code checks if the customer is "In Active."
   * If the customer is "In Active," and a form has been submitted (**POST** request), it validates the entered customer information.
   * It ensures that the entered name, phone number, and ID match the customer's record. If not, it displays an error message.
   * If the activation is successful, an SMS is sent to the customer's phone number to notify them of the successful activation.
3. **SMS Sending:**
   * The SMS message is constructed using the customer's name and a predefined message.
   * The Twilio SDK's **messages->create** method is used to send the SMS to the customer's phone number.
   * The **sendSms** function is called with the recipient's phone number, message, and Twilio account credentials.
4. **HTML Form:**
   * If the customer is "In Active," a form is displayed to collect activation details (ID, name, phone number, balance, and billing period).
   * The form's data is submitted to the same page (**action=""**) using the **POST** method.

**Note:** The code assumes that the provided customer ID is a valid identifier in the "customers" table.

This code demonstrates how to integrate SMS notifications using the Twilio service within a billing activation process. It retrieves customer details, validates user input, calculates billing requirements, updates the database, and sends SMS notifications to customers. Make sure to replace placeholders like Twilio credentials and adjust the SMS content as needed for your specific application.

|  |
| --- |
|  |

**Cron Job and Billing Period Update Script:**

**Cron Job:**

A cron job is a scheduled task on operating systems that allows users to execute specific commands or scripts at predetermined intervals. It's commonly used for automating repetitive tasks, such as data backups, updates, and maintenance, without requiring manual intervention.

**Purpose of the Script:**

The provided PHP script serves the purpose of updating the billing period for customers in a MySQL database. It is intended to be executed as a cron job to automate the process of decrementing the billing period for active customers and handling status changes based on the billing period.

**Overview of the Script:**

The script connects to a MySQL database, retrieves customer data, and updates the billing period for each customer. It also checks whether the billing period has reached zero for active customers and updates their status accordingly.

Here's a breakdown of the script's functionality:

**a. MySQL Database Connection:**

* The script establishes a connection to the MySQL database using provided credentials (**$servername**, **$username**, **$password**, **$dbname**).

**b. Retrieve Customer Data:**

* The script queries the "customers" table to retrieve customer data using the SQL query **$sql**.
* The results are stored in the **$result** variable.

**c. Update Billing Period:**

* If there are customers in the result set (**$result->num\_rows > 0**), the script iterates through each customer's data using a **while** loop.
* For each customer:
  + The customer's ID, billing period, and status are retrieved.
  + The billing period is decremented by 1 if it's greater than 0 (**$billingPeriod > 0**).
  + The script then updates the billing period in the database using the **$updateSql** query.

**d. Update Status (Optional):**

* If the billing period reaches 0 and the customer's status is 'Active,' the script updates the customer's status to 'In Active' (**$newBillingPeriod == 0 && $status == 'Active'**).

**e. Close Database Connection:**

* After updating all customer data, the script closes the database connection using **$conn->close()**.

**Integration as a Cron Job:**

To automate the execution of this script, it is set up a cron job on my server. The cron job can be configured to run this script at specific intervals, such as daily, weekly, or monthly.

**Note:** Before integrating this script as a cron job, I have ensured about appropriate permissions and that the script paths, database credentials, and scheduling intervals are correctly configured.

|  |
| --- |
|  |

**Summary:**

This PHP script, when used as a cron job, automates the process of updating billing periods for customers in a MySQL database. It's designed to decrement billing periods, update status as needed, and facilitate the smooth management of customer billing cycles. The script helps maintain accurate customer billing information and manage their account statuses based on predefined rules. This can be used by ptcl for automatic billing period completion on monthly and weekly basis without keeping any record or doing the task manually.