coditi

DIGITALIZE YOUR VISION

Technical Document for Twilio Robocall/SMS Application

1. System Overview

Purpose:
The Twilio robocall/SMS application automates the process of making calls and sending SMS based on data provided in a CSV file. The system allows for both automated and manual operation and calls/SMS with retry logic if messages or calls go unanswered. It also ensures that duplicate records within the CSV file are handled efficiently by processing each record only once.

Key Features:

- Daily CSV upload for scheduling calls and SMS.
- Automatic retry logic for unanswered/undelivered calls and SMS.
- Duplicate record handling (one call/SMS per unique record).
- Option to trigger calls/SMS via command line or front end.
- Admin panel for management and monitoring.

---

2. Flow Overview

2.1 CSV Upload Process

Purpose:
Handles the daily automated or manual upload of a CSV file, which triggers the calling and SMS processes.

- **CSV File Location**: A CSV file is placed in a designated folder (`csv/`) inside the codebase.
- **Automatic Upload**: The system automatically processes the file daily at 7:00 PM via a command line job.

- **Manual Upload**: Admins can upload the CSV manually through the command line or frontend interface.
- **Duplicate Handling**: The system identifies duplicate records in the CSV and ensures that only one call/SMS is sent per unique record.

2.2 Outgoing Call and SMS Process

Purpose:
Initiates calls and SMS using Twilio APIs, based on the data from the CSV file.

- **Call and SMS Execution**: Sends both calls and SMS when the CSV is processed.
- **Twilio Integration**: The system uses Twilio APIs to initiate calls and send SMS based on the phone numbers in the CSV.
- **Logging**: Logs are maintained for each call and SMS, recording their status (answered, failed, etc.).

2.3 Retry Logic

Purpose:
Implements retry logic to resend calls or SMS if the initial attempt is unsuccessful.

- **First Attempt (7:00 PM)**: Calls and SMS are sent at 7:00 PM based on the CSV.
- **First Retry (7:45 PM)**: If a call or SMS is unanswered or undelivered, the system automatically retries sending them at 7:45 PM.
- **Final Retry (Next Day, 9:00 AM)**: If the call/SMS fails again at 7:45 PM, the system makes a final retry the next day at 9:00 AM.

---

3. Use Cases

3.1 Daily Notification System

Purpose:
Automates daily calls and SMS to recipients based on information from the CSV file.

- **Flow**: Each day at 7:00 PM, a CSV file with recipient information is processed to send automated calls and SMS.
- **Logic**: If a recipient doesn’t respond by 7:45 PM, the system retries. If there’s still no response, the system retries again at 9:00 AM the following morning.

3.2 Manual Upload

Purpose:
Allows administrators to manually upload the CSV file and trigger the call/SMS process immediately.

- **Flow**: Admins may manually upload the CSV via the command line or front end.
- **Logic**: The system processes the CSV immediately, sending the messages/calls and following the same retry logic when the next job is dispatched.

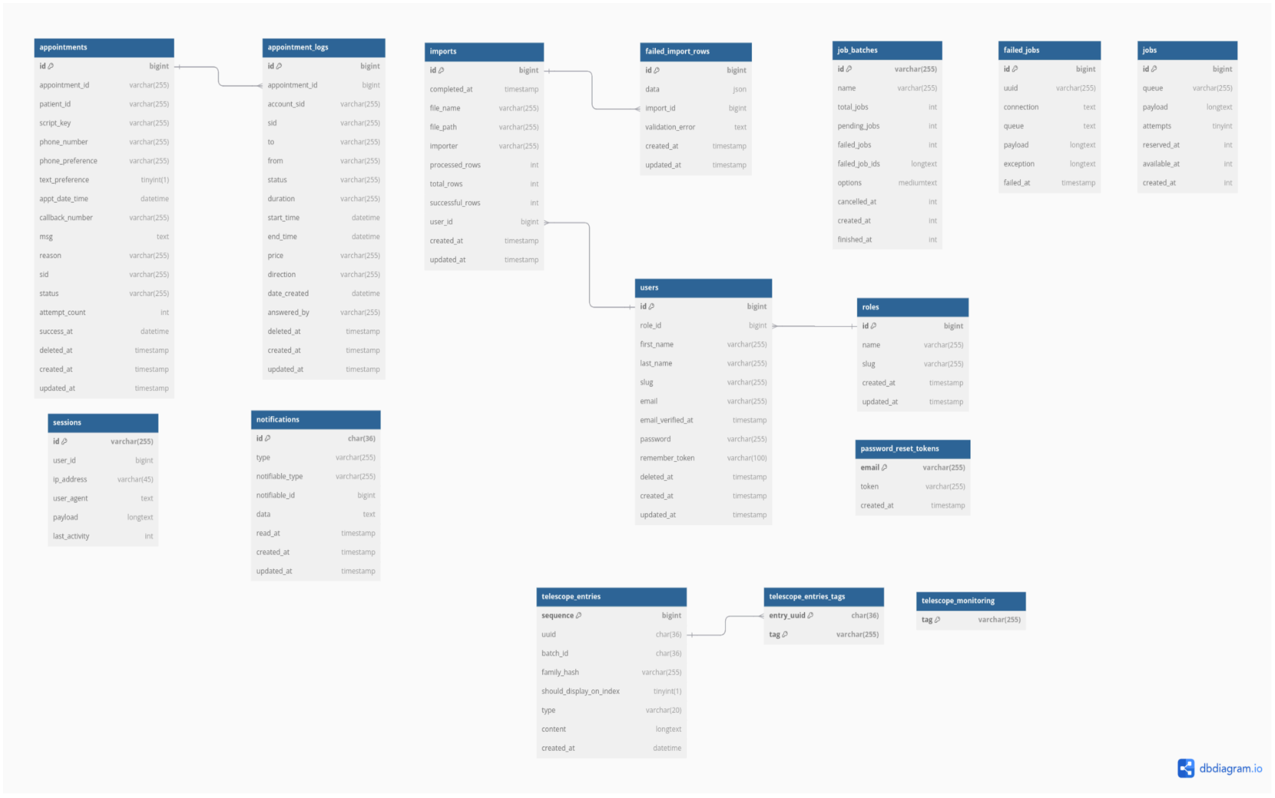
3.3 Duplicate Record Handling

Purpose:
Ensures that duplicate records in the CSV file are handled efficiently, avoiding repeated calls/SMS.

- **Scenario**: If duplicate records are found in a CSV (e.g., the same appointment ID, Patient ID, and Appointment date/time), the system ensures that only one call/SMS is sent per unique record.

---

4. Database Tables



4.1 Appointments Table

Purpose:
Stores all the details related to appointments, including patient information, communication preferences, and appointment status.

Key Fields:

- `id`: Unique identifier for the appointment (primary key).
- `appointment_id`: Comes from the uploaded CSV file.
- `patient_id`: Comes from the uploaded CSV file.
- `phone_number`: Comes from the CSV file.
- `phone_preference`: Comes from the CSV file.
- `text_preference`: Comes from the CSV file.
- `appt_date_time`: Comes from the CSV file.
- `attempt_count`: Tracks how many attempts were made to contact the patient.
- `success_at`: Records when the appointment was successfully confirmed or completed.
- **Timestamps**: `created_at`, `updated_at`, `deleted_at` track creation, modification, and soft deletion.

4.2 Appointment Logs Table

Purpose:
Logs detailed information about phone calls and communications related to appointments.

Key Fields:

- `id`: Unique identifier for each log entry.
- `appointment_id`: Links to the appointments table (foreign key) connecting each log entry to a specific appointment.
- `sid`, `to`, `from`, `status`, `duration`: Stores communication details.
- `start_time`, `end_time`: The start and end time of the calls.
- **Foreign Key**: `appointment_id` references the `id` in the appointments table.
- **Timestamps**: Tracks creation, updates, and soft deletion.

4.3 Failed Import Rows Table

Purpose:
Stores records of data rows that failed during an import process.

Key Fields:

- `id`: Unique identifier for each failed row.
- `data`: JSON object containing the failed row data.
- `import_id`: Foreign key linking to the imports table.

- `validation_error`: Describes the reason for validation failure.
- **Foreign Key**: `import_id` references the imports table.

4.4 Failed Jobs Table

Purpose:
Tracks jobs (tasks) that failed during execution.

Key Fields:

- `id`: Unique identifier for the failed job.
- `uuid`: A universally unique identifier for the failed job.
- `payload`: The job data or task that failed.
- `exception`: Stores the exception message explaining the failure.
- `failed_at`: The timestamp when the job failed.

4.5 Imports Table

Purpose:
Records data about file imports, including status and progress.

Key Fields:

- `id`: Unique identifier for each import.
- `file_name`, `file_path`: Details about the imported file.
- `processed_rows`, `successful_rows`: Tracks how many rows were processed successfully.
- `user_id`: The user who initiated the import (foreign key to users table).
- **Foreign Key**: `user_id` references the users table.

4.6 Job Batches Table

Purpose:
Tracks groups of jobs that are executed together.

Key Fields:

- `id`: Unique identifier for each job batch.
- `total_jobs`, `pending_jobs`, `failed_jobs`: Counters for the total, pending, and failed jobs in the batch.
- `failed_job_ids`: Stores IDs of failed jobs.
- `created_at`, `finished_at`: Timestamps indicating the start and end of the batch.

4.7 Jobs Table

Purpose:
Stores queued jobs for background processing.

Key Fields:

- `id`: Unique identifier for each job.
- `queue`: The name of the queue where the job is stored.
- `payload`: The serialized job data.
- `attempts`: Number of attempts made to complete the job.
- `available_at`, `reserved_at`: Timestamps for when the job is available for processing or reserved for execution.

4.8 Migrations Table

Purpose:
Stores queued jobs for background processing.

Key Fields:

- `id`: Unique identifier for each migration.
- `migration`: The name of the migration file.
- `batch`: The batch number, used to group migrations that were applied together.

4.9 Notifications Table

Purpose:
Stores notifications that are sent to users.

Key Fields:

- `id`: Unique identifier for each notification.
- `type`: The type of notification (e.g., email, SMS).
- `notifiable_type`, `notifiable_id`: Polymorphic fields to link notifications to users or other entities.
- `data`: JSON data containing the notification content.
- `read_at`: Timestamp for when the notification was read.

4.10 Password_reset_tokens Table

Purpose:
Stores tokens for password reset functionality.

Key Fields:

- `email`: The email address of the user requesting a password reset.
- `token`: A secure token used to verify the password reset request.
- `created_at`: The timestamp when the token was generated.

4.11 Roles Table

Purpose:
Stores user roles for access control.

Key Fields:

- `id`: Unique identifier for each role.
- `name`: The name of the role (e.g., admin, user).
- `slug`: A URL-friendly version of the role name.

4.12 Sessions Table

Purpose:
Tracks user sessions for authentication purposes.

Key Fields:

- `id`: Unique session identifier.
- `user_id`: The user associated with the session.
- `ip_address`, `user_agent`: Stores the user's IP address and browser details.
- `payload`: Serialized session data.
- `last_activity`: Timestamp for the last activity in the session.

4.13 Telescope_entries Table

Purpose:
Tracks various requests and logs captured by Laravel Telescope for debugging and monitoring.

Key Fields:

- `sequence`: Unique identifier for each entry.
- `uuid`: Universally unique identifier for the entry.
- `type`: The type of entry (e.g., request, log, exception).
- `content`: The detailed content of the entry (e.g., request payload or log message).
- `created_at`: The timestamp for when the entry was created.
- **Indexes**: Efficiently track entries by type, UUID, and creation date.

4.14 telescope_entries_tags Table

Purpose:
Links tags to `telescope_entries` for easier filtering of Telescope logs.

Key Fields:

- `entry_uuid`: The UUID of the Telescope entry.
- `tag`: The tag used for filtering entries (e.g., request type).
- **Foreign Key**: Links `entry_uuid` to the `telescope_entries` table.

4.15 telescope_monitoring Table

Purpose:
Stores monitoring tags used by Laravel Telescope to track specific application events or issues.

Key Fields:

- `tag`: A unique identifier for the monitored tag.

4.16 Users Table

Purpose:
Stores information about the users of the system.

Key Fields:

- `id`: Unique identifier for the user.
- `role_id`: Foreign key linking to the `roles` table to determine the user's role.
- `first_name`, `last_name`, `email`: Personal details of the user.

---

5. Code Files

5.1 Scheduler File

Path: `im-appointment-reminders.ca/bootstrap/app.php`
Purpose:
Contains the scheduling time of the command in the `withSchedule` method/function, where the scheduling timings for automatic execution can be modified.

5.2 Command File

Path: `im-appointment-reminders.ca/app/Console/Commands/ImportCsv.php`
Purpose:
This file is called when the commands `php artisan import:csv` and `php artisan import:csv --now` are executed. It handles the import of the CSV file from the `csv/` folder and triggers the call/SMS jobs. Here in the handle method/function, we can change the csv folder name inside base_path('csv') and after changing this folder we also have to change or make a folder in the project so that our command will pick the csv file without any error.

5.3 Reminder Jobs File

Path: `im-appointment-reminders.ca/app/Jobs/CallAndSmsReminderJob.php`
Purpose:
Handles retrieving the pending or failed appointments that have had fewer than 3 attempts and triggers the call or SMS job based on the user's text_preference in the CSV.

5.4 Call Job File

Path: `im-appointment-reminders.ca/app/Jobs/MakeCallJob.php`
Purpose:
This file contains the logic to make a call to the patient, it gets the Twilio credentials, Twilio phone number, and appointment phone_number and makes a call to the appointment phone_number and updates the appointment status to scheduled, If the call fails for several reasons it will update the appointment status to failed. Also when this job runs it will update the appointment attempt count.

5.5 SMS Job File

Path: `im-appointment-reminders.ca/app/Jobs/SentSmsJob.php`
Purpose:
Contains the logic to send SMS to recipients using Twilio credentials. It updates the status and attempt count for each SMS.

5.6 Call Log File

Path:
`im-appointment-reminders.ca/app/Http/Controllers/Api/CallLogApiController.php`
Purpose:
Handles the request generated by Twilio after a call, logging details such as call duration, status, start time, and end time. Logs will be created in the appointment_logs table.

5.7 SMS Log File

Path:
`im-appointment-reminders.ca/app/Http/Controllers/Api/MessageLogApiController.php`
Purpose:
Handles requests from Twilio after an SMS, logging the delivery details, status, and timestamps for tracking. Logs will be created in the appointment_logs table.

---

6. Frontend Files

For the Front end, we use the Filament package whose files you can find in the folder im-appointment-reminders.ca/app/Filament
Here are some folders inside im-appointment-reminders.ca/app/Filament :

6.1 Resource Folder

- It contains the front end for an appointment inside im-appointment-reminders.ca/app/Filament/Resources/AppointmentResource.php file and appointment logs front end inside im-appointment-reminders.ca/app/Filament/Resources/AppointmentLogResource.php file

6.2 Imports Folder

- It contains the appointment import file in which we map the CSV rows and save it in the database.

---

7. Technical Requirements

7.1 Infrastructure

- **Cloud Provider**: AWS (Amazon Web Services)
- **Server**: The backend application is hosted on AWS using a Lightsail instance.
- **Database**: RDS (Relational Database Service) stores call/SMS logs and retry status.
- **Storage**: CSV files are temporarily stored in the `csv/` folder, after importing the file it goes into the archive folder - storage/app/public/archive

7.2 API Integration

- **Twilio API**: The system uses Twilio APIs for both SMS and call functionality.
- Use an XML file for voice response according to the appointment_msg field.

---

8. Commands

- **Restart the job**: If there are any changes inside the Job files then we need to restart the jobs using
- **Refresh the database**: If you ever need to empty the database then use the command

---

9. Conclusion

The Twilio robocall/SMS application efficiently automates communication processes with flexible scheduling, retry logic, and built-in duplicate handling. Its ability to run calls and SMS and retry based on delivery status ensures reliable and timely notifications.