**Component Report**

**Components and Functions: Free Whatsapp Bulker**

**1. GUI Components**

* **Libraries Used:**
  + **tkinter:** For the GUI components.
  + **scrolledtext:** For the multi-line message input.
  + **messagebox, filedialog, simpledialog:** For user interaction dialogs.
* **Widgets**:
  + Labels, Entry widgets, Buttons for file selection, and text input.
  + A text box for direct message entry.
  + Buttons for starting the messaging process and clearing inputs.
* **Layout**:
  + The GUI uses frames and organized widget placement for intuitive usage.

**2. File Handling**

* **Supported File Types:**
  + Phone Numbers: .csv and .txt.
  + Messages: .txt.
  + Media Attachments: .jpg, .jpeg, .png, .pdf, .docx.
* **File Dialogs:**
  + Utilizes filedialog.askopenfilename() for file selection.
  + Google Sheets integration using gspread.

3. **Message Delivery Mechanism**

* **WhatsApp Web Automation:**
  + Uses Selenium WebDriver to interact with WhatsApp Web.
  + Loads a user profile for preserving login session (--user-data-dir option).
  + Sends messages via the WhatsApp Web URL API with pre-filled messages.
  + XPath is used to locate and interact with the "Send" button.
* **Media Attachment:**
  + URLs for sending messages are dynamically generated using urllib.parse.quote() for encoding the message text.

4. **Threading**

* Messaging operations run in a separate thread using the threading module to keep the GUI responsive.

5. **Error Handling**

* Basic error messages using messagebox.showerror() for GUI errors.
* Exceptions during messaging are caught and displayed in the status label.

6. **Dependencies**

* **selenium**: For browser automation.
* **webdriver\_manager.chrome:** For managing ChromeDriver installation.
* **gspread:** For integrating Google Sheets.
* **google.oauth2.service\_account:** For authenticating Google Sheets access.

7. **Additional Features**

* **Google Sheets Integration:**
  + Fetches phone numbers from a specified Google Sheet.
  + Requires a JSON service account key for authentication.
* **Real-time Status Updates:**
  + Status label displays progress, errors, or completion messages.
* **Input Clearing:**
  + Clears all input fields with the "Clear" button.

**Components and Functions: Whatsapp Bulker using Official API**

**1. API Integration**

* **Library Used**: requests for HTTP communication with the WhatsApp API.
* **Endpoint**: https://graph.facebook.com/v17.0/{phone\_number\_id}/messages.
* **Authentication**: Utilizes an access\_token for API requests via an Authorization header.
* **Message Payload**:
  + **Template Messages**: Sends pre-approved templates using template.name and language configurations.
  + **Media Messages**:
    - **Image**: Sends image URLs using the image.link field.
    - **Document**: Sends document URLs using the document.link field.
* **Error Handling**: Logs errors or failed requests in a log file (message\_log.txt).

**2. GUI Components**

* **Libraries Used**:
  + tkinter: For the graphical user interface.
  + Dialogs: messagebox, filedialog, and simpledialog for user interactions.
* **Widgets**:
  + Input fields for phone\_number\_id, access\_token, template\_name, and file selection.
  + Radio buttons for selecting message type (text, image, or document).
  + Checkbox for enabling "Dry Run" mode (test without actual message delivery).
  + Buttons for starting the messaging process, selecting files, and fetching from Google Sheets.
* **Layout**:
  + Organized into frames for logical grouping of options and inputs.
  + Customizable look and feel with color themes and fonts.

**3. File Handling**

* **Supported Formats**:
  + Phone Numbers: .csv and .txt for recipient numbers.
  + Media Files: External URLs for images/documents.
* **Google Sheets Integration**:
  + Uses gspread to authenticate and fetch numbers from Google Sheets.
  + Requires a JSON credentials file for authentication.

**4. Messaging Process**

* **Dry Run Mode**:
  + Logs the payload without sending messages when dryrun is enabled.
  + Useful for verifying configurations without API calls.
* **Bulk Processing**:
  + Iterates through recipient numbers and sends messages based on the selected type.
  + Logs success or failure for each recipient.
* **Logging**:
  + Logs each operation (success, failure, or errors) in message\_log.txt.

**5. Error Handling**

* Displays user-friendly error messages using messagebox.
* Catches exceptions during file operations or API calls and logs them for troubleshooting.

**6. Dependencies**

* **External Libraries**:
  + requests: For API requests.
  + gspread: For Google Sheets integration.
  + tkinter: For GUI components.
* **Logging**:
  + Logs activity and errors in a structured file (message\_log.txt).

**7. Additional Features**

* **Customization**:
  + Supports different message types via the WhatsApp Business API.
  + Enables flexible number input from local files or Google Sheets.
* **Test Mode**:
  + Provides a dry-run feature for safe testing without sending real messages.

**Components and Functions: Web Scraping Application**

1. **API Integration**
   * **Library Used**:
     + serpapi for Google Maps API interaction.
   * **API Endpoint**:
     + Uses the SerpApi Google Maps engine with search queries.
   * **Authentication**:
     + Utilizes an API key for authentication.
   * **Data Retrieval**:
     + Extracts local search results (e.g., hotels) from Google Maps.
   * **Data Normalization**:
     + Converts API responses into a normalized DataFrame.
   * **Columns Extracted**:
     + Position, title, address, phone, website, description, amenities, rating, reviews, price, type\_id, type\_ids, place\_id, data\_id, reviews\_link, provider\_id.
2. **GUI Components**
   * **Libraries Used**:
     + tkinter: For the graphical user interface (GUI).
     + ttk: Enhanced widgets for the GUI.
     + messagebox: Provides error and info messages for the user.
   * **Widgets**:
     + Input fields for search term, Google Sheets ID, sheet name, country, state, and city.
     + Dropdown menus for country, state, and city selection.
     + Buttons for starting the scraping process.
   * **Dynamic Dropdowns**:
     + The state and city dropdowns dynamically update based on the selected country.
   * **Data Selection**:
     + Allows the user to select countries, states, and cities to focus the scraping process.
3. **File Handling**
   * **File Support**:
     + CSV file (Zip Code.csv) for country, state, and city data.
     + Outputs data to CSV and JSON files.
   * **Data Handling**:
     + The Google Sheets data is uploaded using gspread.
     + Extracted data is stored locally as places.csv and places.json in the scraping\_results folder.
   * **Data Cleansing**:
     + Removes missing rows before uploading to Google Sheets.
     + Clears previous data in the Google Sheets worksheet before inserting new data.
   * **Google Sheets Integration**:
     + Uses gspread to authenticate and access Google Sheets.
     + Data is uploaded to the specified Google Sheets worksheet.
4. **Web Scraping Process**
   * **Libraries Used**:
     + selenium: Used to control and interact with a Chrome web browser.
     + webdriver\_manager: Automates the setup of ChromeDriver.
   * **Scraping Configuration**:
     + **Browser Setup**:
       - Incognito mode for privacy.
       - Headless mode for running without a visible browser.
       - Window size of 1920x1080 to ensure full page load.
   * **Data Extraction**:
     + Uses Selenium WebDriver to scrape data from Google Maps.
     + Extracts the following data from each location:
       - Title, rating, reviews, category, address, website, phone number.
     + Data is stored in both CSV and JSON formats.
   * **Scroll Automation**:
     + Scrolls down Google Maps pages until all items are loaded.
     + Waits for end-of-list messages before stopping.
   * **Error Handling**:
     + Logs errors encountered during scraping.
     + If scraping is interrupted, it resumes from the last link.
5. **Google Sheets Integration**
   * **Libraries Used**:
     + gspread: For Google Sheets API interaction.
     + gspread\_dataframe: For reading and writing DataFrames to Google Sheets.
   * **Authentication**:
     + Uses a service account JSON file for Google Sheets API authentication.
   * **Data Upload**:
     + Uploads data from CSV files to specified Google Sheets worksheets.
     + Clears existing content in the sheet before adding new data.
     + Ensures that empty DataFrames are properly managed (i.e., adds a blank row if DataFrame is empty).
6. **Logging and Error Handling**
   * **Error Logging**:
     + Captures errors during web scraping and API calls.
     + If errors occur during scraping, the item is skipped, and the program continues.
   * **User Feedback**:
     + Uses messagebox to provide feedback on success or failure.
     + Displays specific error messages for missing inputs (like country, state, search term, etc.).
   * **Error Handling During File Operations**:
     + If data extraction fails, the program notifies the user.
     + If Google Sheets upload fails, it displays a descriptive error message.
7. **Core Functions**
   * **chrome\_setup()**:
     + Configures Chrome browser for scraping (headless, incognito, etc.).
   * **scroll\_panel\_to\_bottom()**:
     + Scrolls the Google Maps page to load additional results.
   * **extract\_data()**:
     + Extracts data from item links and writes to CSV and JSON files.
   * **get\_element\_text()**:
     + Retrieves text from elements located by XPath.
   * **get\_element\_attribute()**:
     + Extracts attributes (like aria-label) from elements located by XPath.
   * **update\_states()**:
     + Updates the states dropdown dynamically when a country is selected.
   * **update\_cities()**:
     + Updates the cities dropdown dynamically when a state is selected.
   * **start\_scraping()**:
     + Main function to start the scraping process based on the selected country, state, and city.
8. **Dependencies**
   * **External Libraries**:
     + tkinter: For GUI components.
     + gspread: For Google Sheets API integration.
     + gspread\_dataframe: For manipulating DataFrames in Google Sheets.
     + serpapi: For Google Maps API search.
     + pandas: For DataFrame manipulation.
     + selenium: For web scraping.
     + webdriver\_manager: For managing ChromeDriver automatically.
   * **Logging and Alerts**:
     + Logs all extraction steps.
     + Alerts the user if any errors are encountered during scraping or Google Sheets upload.
9. **Additional Features**
   * **Customization**:
     + User can select search terms, countries, states, and cities.
     + Outputs CSV and JSON files for review.
   * **Data Upload**:
     + Integrates with Google Sheets for seamless data uploads.
     + Supports automatic upload to Google Sheets using service account authentication.
   * **Data Cleaning**:
     + Ensures only valid data is uploaded to Google Sheets.
     + Filters out empty rows before uploading to the sheet.
   * **Data Persistence**:
     + Extracted data is saved locally as CSV and JSON.
     + If an error occurs, data is preserved for later inspection.

**Component Report: AI Calling Feature**

**1. Components and Functions:**

**1.1 API Integration**

* **Libraries Used:**
  + **Twilio API**:
    - twilio.rest.Client: For making phone calls via the Twilio API.
  + **Ngrok Integration**:
    - requests: For sending HTTP requests to the Ngrok API.
  + **Google Sheets Integration**:
    - gspread: For interacting with Google Sheets.
    - gspread\_dataframe: For converting and handling Google Sheets as DataFrames.
  + **Environment Variable Management**:
    - dotenv: To manage environment variables from the .env file.
* **Data Transmission**:
  + Sends data to Twilio via an API call using client.calls.create().
  + Sends user inputs via Ngrok's public URL.
  + Fetches phone numbers from Google Sheets and populates the call list in the app UI.

**1.2 Voice Interaction**

* **Twilio Voice Route**:
  + **Prompt**: Plays an initial message to users:

"Hello, we are Chef Connect. Are you interested in our services?"

* + **User Response**: Captures DTMF (keypad input) and speech inputs using Twilio's Gather.
  + **Service Selection**:
    - If the user presses 1 or says "yes", they choose from three service segments: Pharmacy, Industrial, or Food.
    - If the user presses 2 or says "no", the call ends with a "Thank you" message.
    - If no input is detected, the call replays the menu.
  + **User Interaction Options**:
    - Users can select options using voice or DTMF (keypad) inputs.
* **Call Flow**:
  + User choices are processed through /handle\_response and /process\_selection endpoints.
  + Redirects to /voice if no input is received.
* **Data Storage**:
  + User responses, Twilio responses, and the option selected are stored in a responses.json file.

**1.3 Call Handling**

* **Call Initiation**:
  + Calls are initiated via Twilio’s client.calls.create() method.
  + Calls use the endpoint NGROK\_URL/voice.
  + Calls are made from a Twilio phone number (set in the .env variable TWLIO\_NUMBER) to the recipient's phone number (MY\_NUMBER).
* **Environment Variables**:
  + **TWILIO\_ACCOUNT\_SID**: Twilio account identifier.
  + **TWILIO\_AUTH**: Twilio API authentication key.
  + **NGROK\_URL**: Public URL from Ngrok for call redirection.
  + **MY\_NUMBER**: The recipient's phone number.
  + **TWLIO\_NUMBER**: The Twilio phone number used for outbound calls.

**2. User Interface (GUI)**

* **Libraries Used**:
  + **tkinter**: For creating GUI components.
  + **threading**: To run services in the background without blocking the UI.
* **Widgets**:
  + **Input Fields**:
    - For entering the Google Sheets ID, sheet name, and column containing phone numbers.
    - For entering phone numbers manually (one per line) for calling.
  + **Labels**: Display the Ngrok URL and system status.
  + **Buttons**:
    - **Fetch Data**: Fetches phone numbers from the Google Sheet.
    - **Make Calls**: Initiates calls to all listed phone numbers.
* **User Actions**:
  + Users can enter phone numbers manually or fetch them directly from Google Sheets.
  + Users can initiate calls to multiple phone numbers via the "Make Calls" button.
  + The status label shows the current system state (e.g., checking services, errors, or ongoing calls).

**3. Web Server Integration**

* **Libraries Used**:
  + **Flask**: For HTTP requests from Twilio.
  + **flask\_sock**: For WebSocket support.
* **Flask Routes**:
  + /voice: Handles the initial voice interaction.
  + /handle\_response: Handles user input (digits or speech).
  + /process\_selection: Processes user selection for Pharmacy, Industrial, or Food.
* **System Integration**:
  + Starts a local Flask server on port 5000.
  + Ngrok exposes the local server to the internet.
* **Ngrok Setup**:
  + **Ngrok Authentication**: Relies on NGROK\_AUTHTOKEN from the .env file.
  + **Ngrok Tunnel**: The URL is retrieved from the local Ngrok API and updated in the .env file.

**4. Service Automation**

* **Ngrok Service**:
  + **start\_ngrok()**: Starts a public Ngrok tunnel.
  + The Ngrok URL is stored in the .env file for call redirection.
* **Flask Server**:
  + Flask runs on port 5000.
  + On startup, it verifies if Ngrok and Flask are active.
* **Google Sheets Integration**:
  + **download\_data\_from\_spreadsheet()**: Downloads phone numbers from a Google Sheets file and populates the call list in the app UI.
  + **upload\_data\_to\_spreadsheet()**: Uploads data back to Google Sheets.

**5. Error Handling**

* **Call Errors**:
  + If a call fails, the system shows an error message.
  + Exceptions are caught and displayed via messagebox.showerror.
* **Ngrok and Flask Status**:
  + If either service is down, the system prompts the user to restart.
  + If Flask and Ngrok are not running after 60 seconds, an error is displayed.
* **File and Environment Errors**:
  + Errors in file I/O, API requests, and HTTP calls are caught and logged.
  + Errors in Ngrok URL retrieval are logged.

**6. Google Sheets Integration**

* **Libraries Used**:
  + **gspread**: For interacting with Google Sheets.
  + **gspread\_dataframe**: For converting data into Google Sheets DataFrames.
* **Functions**:
  + **download\_data\_from\_spreadsheet()**:
    - Downloads data from a specified Google Sheet.
    - Extracts a column (e.g., "phone numbers") and populates the text box with numbers.
  + **upload\_data\_to\_spreadsheet()**:
    - Uploads data to a specified Google Sheet.
* **Data Handling**:
  + The system fetches phone numbers from Google Sheets.
  + Phone numbers are displayed in the GUI for manual verification before calling.
  + Data is handled as a DataFrame for better manipulation.

**7. Core Functions**

* **update\_env\_variable(key, value)**: Updates or adds a key-value pair to the .env file.
* **store\_response(iteration, user\_response, twilio\_response, selection)**: Stores user responses, Twilio responses, and selection data into responses.json.
* **start\_ngrok(port)**: Starts the Ngrok tunnel and retrieves the public URL.
* **disconnect\_and\_exit()**: Disconnects services and terminates the Flask and Ngrok processes.
* **make\_call(number)**: Initiates a call to a phone number using Twilio.
* **check\_flask\_and\_ngrok()**: Checks if Flask and Ngrok services are running.
* **download\_data\_from\_spreadsheet()**: Downloads data from a Google Sheet.
* **upload\_data\_to\_spreadsheet()**: Uploads data back to a Google Sheet.
* **start\_app\_script()**: Runs the main application script in a background subprocess.

**8. Dependencies**

* **External Libraries**:
  + **Twilio**: For voice calls.
  + **Ngrok**: To expose the local Flask server to the internet.
  + **Flask**: For the web server to handle Twilio callbacks.
  + **gspread**: For Google Sheets data interaction.
  + **dotenv**: For environment variable handling.
  + **gspread\_dataframe**: For DataFrame support with Google Sheets.
  + **requests**: For HTTP requests.
  + **tkinter**: For the graphical user interface.

**9. Additional Features**

* **Voice Recognition**: Users can interact with the system via voice commands (e.g., "yes", "no") or DTMF keypad input.
* **Automated Services**: Flask and Ngrok start automatically at launch.
* **Dynamic Call List**: Users can enter phone numbers manually or fetch them from Google Sheets.
* **Data Logging**: User interactions and system responses are saved in responses.json.
* **Call Status Tracking**: The system tracks and displays call status (e.g., failed, busy, no answer).