NU CRON DEVELOPMENT PROJECT
CONTENTS

NU CRON

INTRODUCTION

NU CRON is an innovative AI-driven Contact Schedule Manager designed to optimize lead generation and communication journey events. Inspired by the Greek mythological figure Cronus, NU CRON embodies the role of a timekeeper and event scheduler, making it a unique and powerful tool for managing contact communications.

PRIMARY FUNCTIONS AND NICHE ROLES

1. **Timekeeper Role**
   - **Function**: Design optimal contact schedules for leads.
   - **Purpose**: Determine the best days and times to contact leads based on enriched data.
   - **Outcome**: Maximizes the likelihood of successful contact by reaching leads at optimal times.
2. **Communication Scheduler Role**
   - **Function**: Schedule initial and follow-up contact events.
   - **Purpose**: Automate communication scheduling based on journey outcomes.
   - **Outcome**: Ensures timely follow-ups and optimizes the communication journey.
3. **Task Maker Role**

   - **Function**: Decision-making based on communication outcomes.
   - **Purpose**: Use AI suggestions to determine next steps in the communication journey.
   - **Outcome**: Enhances the effectiveness of communication strategies.
4. **Data Keeper Role**
   - **Function**: Maintain logs, activities, and outcomes.
   - **Purpose**: Provide feedback and updates to other roles.
   - **Outcome**: Ensures accurate record-keeping and data-driven decision-making.

AI-DRIVEN COMMUNICATION

- **Generative Communication**: Powered by AI, NU CRON engages with leads through various channels (SMS, email, voice, chat), providing personalized and timely responses.
- **System Mediators**: Agents and operational team members monitor and assist in the communication process, stepping in when necessary to ensure optimal outcomes.

INTEGRATION WITH NU GUI AND OTHER SOLUTIONS

- **NU GUI**: Develops the user interface and user experience for NU CRON, ensuring an intuitive and efficient interaction.
- **NU SIP**: Provides advanced VoIP services for clear and reliable voice communications.
- **NU CCS**: Enhances call control with advanced VoIP management capabilities.
- **NU SMS**: Facilitates comprehensive messaging services for effective communication.
- **NU DATA**: Enriches contact data, providing essential information for scheduling and communication.

KEY FEATURES OF NU GUI SOLUTIONS

1. **NU SIP (VoIP Services)**
   - **Advanced Proxy Interconnect**: Efficient routing and security.
   - **Enhanced RTP Media Interconnect**: Effective media management.
   - **Advanced DID Database Management**: Robust DID/ANI control.
   - **Smart CLI Number Management**: Improved call routing control.
   - **Comprehensive Reporting**: In-depth call analytics.
2. **NU CCS (Call Control Server)**
   - **Advanced Proxy Interconnect**: Consistent voice traffic control.
   - **Enhanced Traffic Filtering**: Optimized network performance.
   - **Fixed IP Security**: Enhanced security with fixed IPs.
   - **Local Media Server Utilization**: Superior call quality.
   - **Webhook API Integration**: Enhanced data retrieval and display.
3. **NU SMS (Direct Messaging Services)**
   - **API Features**: Easy integration.
   - **Bulk SMS and Automated Messaging**: Streamlined communication.
   - **Real-Time Delivery Tracking**: Accurate message delivery.
   - **Comprehensive Reporting**: Insights into messaging performance.
4. **NU DATA (Data Enrichment Services)**
   - **Data Cleansing**: Removes inaccuracies.
   - **Data Validation**: Ensures reliability.
   - **Data Augmentation**: Adds valuable insights.
   - **Real-Time Processing**: Up-to-date information.
   - **Secure and Compliant**: Adheres to industry standards.

BUSINESS CASE FOR NU CRON

INTRODUCTION

NU CRON is a Contact Schedule Manager designed to automate communication tasks (cron jobs) at set intervals or triggered events. It supports multiple communication channels such as SMS, email, voice, and instant messaging. Leveraging AI generative communication solutions, it integrates with various open-source technologies to streamline interactions, improve lead management, and enhance customer engagement.

USE CASES

1. SALES AND MARKETING AUTOMATION

- Lead Nurturing: Automate follow-up communications with cold leads, moving them through the sales funnel using personalized messages.
- Campaign Management: Use Mailtrain and Mautic to run email campaigns and track their performance.
- Segmented Marketing: Utilize AI to analyze interactions and segment leads based on their behavior and responses.

2. CUSTOMER SUPPORT

- Omnichannel Support: Manage customer queries across multiple channels (SMS, email, voice, chat) from a single platform.
- Automated Responses: Implement Rasa chatbots to provide instant answers to common questions.
- Call Transcription: Use Whisper to transcribe customer calls, making it easier to review and improve support interactions.

3. SERVICE DELIVERY

- Appointment Scheduling: Automate reminders and confirmations for appointments via SMS, email, and other channels.
- Follow-Up Communications: Automatically send follow-up messages after service delivery to ensure customer satisfaction.

4. INTERNAL COMMUNICATIONS

- Task Management: Schedule and automate internal communications and reminders for team members.
- Meeting Summaries: Transcribe and summarize meeting recordings to ensure all team members are on the same page.

5. ANALYTICS AND REPORTING

- Performance Analysis: Use Metabase to visualize data and generate reports on communication effectiveness and lead conversion rates.
- Feedback Loop: Continuously gather and analyze customer feedback to improve communication strategies.

BENEFITS

- Efficiency: Automate repetitive communication tasks, freeing up time for more strategic activities.
- Scalability: Easily scale to handle a growing number of leads and interactions across multiple channels.
- Personalization: Use AI to tailor communications based on individual lead behavior and preferences.
- Improved Decision-Making: Gain insights from detailed analytics and reports to optimize marketing and support strategies.

- Enhanced Customer Experience: Provide timely, consistent, and personalized communications across all touchpoints.

DEVELOPMENT PROJECT PLAN FOR NU CRON

PROJECT OVERVIEW

NU CRON is a robust, scalable, and efficient Contact Schedule Manager that automates communication tasks (cron jobs) at set intervals or triggered events. The platform will support multiple communication channels, including SMS, email, voice, and instant messaging, leveraging AI generative communication solutions, and integrating with various open-source technologies.

PROJECT PHASES

1. PROJECT INITIATION

- Define project scope and objectives.
- Identify key stakeholders.
- Develop project charter.
- Assemble project team.

2. PLANNING

- Requirements gathering.
  - Conduct workshops with stakeholders to gather functional and non-functional requirements.
  - Identify all communication channels and integrations needed.
- Define project timeline and milestones.
- Develop detailed project plan.
  - Create a work breakdown structure (WBS) and Gantt chart.
- Risk assessment and mitigation planning.
  - Conduct a SWOT analysis.

3. DESIGN

- System architecture design.
  - Create high-level and detailed architecture diagrams.
  - Define data flow and user journey maps.
- API design and documentation.
  - Utilize OpenAPI Specification for documenting APIs.
- Database schema design.
  - Design ER diagrams and data models.
- Security and compliance planning.
  - Develop a security policy and compliance checklist.

4. DEVELOPMENT

- Set up development environment.
  - Utilize Docker for containerization.
  - Set up CI/CD pipelines.
- Implement core functionalities.
  - Develop the core scheduling and cron job engine.
  - Implement lead funneling and workflow management systems.
- Develop integrations for each component.

- Implement AI generative communication solutions.
  - Integrate GPT-Neo/GPT-J for AI responses.
  - Develop UI/UX for the central management hub.
  - Create wireframes and prototypes.
- Implement audio-to-text transcription and summary outcome features.
  - Integrate Whisper for speech-to-text conversion.
  - Develop NLP pipelines using SpaCy for transcription analysis.
  - Train and deploy summarization models using TensorFlow/PyTorch.
  - Store and manage transcriptions and summaries in PostgreSQL.

5. TESTING

- Unit testing.
- Integration testing.
  - Ensure all integrations work seamlessly.
- System testing.
  - Conduct end-to-end testing.
- User acceptance testing (UAT).
  - Involve key stakeholders and end-users.

6. DEPLOYMENT

- Prepare deployment environment.
  - Set up staging and production environments.
- Deploy platform components.
  - Use Kubernetes for orchestration.
- Conduct final testing in production.
- Go-live.
  - Plan a phased rollout.

7. MAINTENANCE AND SUPPORT

- Monitor system performance.
  - Use monitoring tools like Prometheus and Grafana.
- Address bugs and issues.
  - Implement a ticketing system.
- Provide user support and training.
  - Develop a knowledge base and training materials.
- Plan for future enhancements.
  - Establish a feedback loop.

ROLES AND RESPONSIBILITIES

1. Project Manager: Oversee the entire project, ensuring timelines and objectives are met.
2. Developers: Implement integrations, develop core functionalities, and handle API connections.
3. QA Team: Conduct testing at all levels (unit, integration, system, UAT).
4. DevOps Engineer: Set up and manage deployment environments.
5. UI/UX Designer: Design intuitive interfaces for Chatwoot.
6. Security Specialist: Ensure all security measures are implemented and compliance requirements are met.
7. Support Team: Provide ongoing support and training for users.
8. Data Scientist: Develop and refine AI models for generative communication and summarization.

TECHNOLOGIES AND THEIR FUNCTIONS

CORE COMMUNICATION TECHNOLOGIES

1. CHATWOOT
- Function: Central hub for omnichannel communications.
- Purpose: Integrate and manage all communication channels.
- Git Repository: [Chatwoot GitHub](https://github.com/chatwoot/chatwoot)

2. KAMAILIO
- Function: SIP server for call control.
- Purpose: Manage SIP signaling for voice calls.
- Git Repository: [Kamailio GitHub](https://github.com/kamailio/kamailio)

3. FREESWITCH
- Function: Media handling.
- Purpose: Handle media streams for voice calls, conferencing, and IVR.
- Git Repository: [FreeSwitch GitHub](https://github.com/signalwire/freeswitch)

4. JASMIN SMS GATEWAY
- Function: SMS management.
- Purpose: Handle inbound and outbound SMS, support for high throughput.
- Git Repository: [Jasmin SMS Gateway GitHub](https://github.com/jookies/jasmin)

5. MAILTRAIN
- Function: Email campaign management.
- Purpose: Manage and automate email campaigns.
- Git Repository: [Mailtrain GitHub](https://github.com/Mailtrain-org/mailtrain)

6. JITSI
- Function: Video conferencing and RCS support.
- Purpose: Extend communication capabilities to include video and Rich Communication Services (RCS).
- Git Repository: [Jitsi GitHub](https://github.com/jitsi/jitsi-meet)

7. WHATSAPP BUSINESS API
- Function: Messaging via WhatsApp.
- Purpose: Enable communication with users on WhatsApp.
- Documentation: [WhatsApp Business API Documentation](https://developers.facebook.com/docs/whatsapp/getting-started/)

8. TELETHON
- Function: Telegram API library.
- Purpose: Manage Telegram communications.
- Git Repository: [Telethon GitHub](https://github.com/LonamiWebs/Telethon)

9. RASA
- Function: Chatbot framework.
- Purpose: Implement AI-driven chatbots for automated responses.

- Git Repository: [Rasa GitHub](https://github.com/RasaHQ/rasa)

10. MAUTIC

- Function: Marketing automation.
- Purpose: Manage and automate marketing campaigns.
- Git Repository: [Mautic GitHub](https://github.com/mautic/mautic)

11. SUITECRM

- Function: Customer relationship management.
- Purpose: Manage leads and customer interactions.
- Git Repository: [SuiteCRM GitHub](https://github.com/salesagility/SuiteCRM)

12. METABASE

- Function: Business intelligence and analytics.
- Purpose: Analyze and visualize data for insights.
- Git Repository: [Metabase GitHub](https://github.com/metabase/metabase)

13. GPT-NEO/GPT-J

- Function: AI generative communication.
- Purpose: Generate AI-driven responses for communication.
- Git Repository: [GPT-Neo GitHub](https://github.com/EleutherAI/gpt-neo) / [GPT-J GitHub](https://github.com/kingoflolz/mesh-transformer-jax)

AUDIO-TO-TEXT AND DATA PROCESSING TECHNOLOGIES

14. WHISPER

- Function: Speech recognition.
- Purpose: Convert audio recordings of phone calls or chats into text, with robust support for multiple languages and accents, including African languages.
- Git Repository: [Whisper GitHub](https://github.com/openai/whisper)

15. POSTGRESQL

- Function: Database management.
- Purpose: Store transcriptions, summary reports, and categorized leads.
- Git Repository: [PostgreSQL GitHub](https://github.com/postgres/postgres)

16. SPACY

- Function: Natural Language Processing (NLP).
- Purpose: Analyze and summarize transcriptions to determine the outcome of conversations.
- Git Repository: [SpaCy GitHub](https://github.com/explosion/spaCy)

17. TENSORFLOW/PYTORCH

- Function: Machine learning frameworks.
- Purpose: Develop custom models for transcription analysis and outcome summary generation.
- Git Repositories:
  - [TensorFlow GitHub](https://github.com/tensorflow/tensorflow)
  - [PyTorch GitHub](https://github.com/pytorch/pytorch)

INTEGRATION STEPS AND API CONNECTIONS

CORE COMMUNICATION TECHNOLOGIES

1. Set up and configure Chatwootas the central communication hub.
2. Integrate Kamailiofor SIP call control.
3. Connect FreeSwitchto Kamailio for handling media streams.
4. Set up and configure Jasmin SMS Gatewayfor SMS management.
5. Configure Mailtrainfor email campaigns.
6. Set up Jitsifor video conferencing and RCS support.
7. Configure WhatsApp Business APIfor WhatsApp messaging.
8. Set up Telethonfor Telegram communications.
9. Implement Rasafor chatbot interactions.
10. Configure Mauticfor marketing automation.
11. Set up SuiteCRMfor managing leads and customer interactions.
12. Configure Metabasefor data analysis and visualization.
13. Integrate GPT-Neo/GPT-Jfor AI generative communication.

AUDIO-TO-TEXT TRANSCRIPTION AND SUMMARIZATION

1. Install and configure Whisperfor speech recognition.
2. Set up PostgreSQLto store transcriptions, summaries, and lead data.
3. Configure SpaCyfor NLP tasks.
4. Develop and train models with TensorFlow/PyTorchfor transcription analysis and summary generation.

INTEGRATION FLOW AND API CONNECTIONS

1. Initiate Communication
   - Communication via multiple channels (SMS, email, voice, chat) is captured by Chatwoot.
   - Kamailio and FreeSwitch handle SIP call control and media streaming.
   - Jasmin SMS Gateway manages SMS interactions.
   - Mailtrain handles email campaigns.
   - Jitsi manages video conferencing and RCS.
   - WhatsApp Business API handles WhatsApp messaging.
   - Telethon manages Telegram communications.
   - Rasa chatbots provide automated responses.
   - GPT-Neo/GPT-J generates AI-driven responses.

2. Audio-to-Text Transcription
   - Audio recordings from calls and chats are sent to Whisper for transcription.
   - Transcriptions are stored in PostgreSQL.

3. NLP and Summarization
   - Transcriptions are analyzed by SpaCy to determine conversation outcomes.
   - Summarization models in TensorFlow/PyTorch generate detailed reports.
   - Summaries and classified outcomes are stored in PostgreSQL.

4. Lead Data Management
   - Lead information is updated in SuiteCRM based on conversation outcomes and segmentation.

- Metabase analyzes and visualizes data for insights.

Conclusion
By following this comprehensive technical document, the development team can effectively implement and integrate the various technologies required for NU CRON. Each technology is chosen to handle specific functionalities, ensuring a robust and scalable Contact Schedule Manager capable of automating communication tasks, transcription analysis, and lead management. This structured approach will ensure seamless interoperability and optimal performance of the NU CRON application.

NU CRON PROJECT DIRECTORY STRUCTURE (REACT FRAMEWORK)

The following directory structure is a suggested tailored for the NU CRON project, utilizing the React framework for the frontend. This structure adheres to best development practices, ensuring modularity, maintainability, and scalability.

```root directory
NU_CRON/
├── config/
│   ├── config.yml
│   ├── database.yml
│   ├── environment/
│   │   ├── development.yml
│   │   ├── production.yml
│   │   └── test.yml
│   └── secrets.yml
├── docs/
│   ├── API/
│   │   ├── OpenAI/
│   │   │   ├── OpenAI_getting_started.txt
│   │   │   ├── Endpoints_API_References.txt
│   │   │   └── OpenAI_API_Docs.txt
│   │   ├── Beta_Assistants_API_References.txt
│   │   └── Other_APIs/
│   │       └── API_Documentation.txt
│   ├── UIUX/
│   │   └── Fundamentals_of_creating_a_great_UIUX.pdf
│   ├── architecture_diagrams/
│   ├── requirements/
│   └── user_manual/
├── src/
│   ├── backend/
│   │   ├── app/
│   │   │   ├── controllers/
│   │   │   ├── models/
│   │   │   ├── views/
│   │   │   └── services/
│   │   ├── config/
│   │   ├── db/
│   │   ├── lib/
│   │   ├── public/
│   │   ├── scripts/
│   │   ├── test/
```

```
│   │   ├── tmp/
│   │   └── log/
│   ├── frontend/
│   │   ├── public/
│   │   │   ├── assets/
│   │   │   ├── css/
│   │   │   ├── js/
│   │   │   └── images/
│   │   ├── src/
│   │   │   ├── components/
│   │   │   ├── pages/
│   │   │   ├── services/
│   │   │   ├── store/
│   │   │   ├── utils/
│   │   │   ├── App.js
│   │   │   ├── index.js
│   │   │   └── setupTests.js
│   │   ├── config/
│   │   ├── tests/
│   │   └── index.html
│   ├── integration/
│   │   ├── chatwoot/
│   │   ├── kamailio/
│   │   ├── freeswitch/
│   │   ├── jasmin/
│   │   ├── mailtrain/
│   │   ├── jitsi/
│   │   ├── whatsapp/
│   │   ├── telethon/
│   │   ├── rasa/
│   │   ├── mautic/
│   │   ├── suitecrm/
│   │   ├── metabase/
│   │   ├── gpt-neo/
│   │   ├── whisper/
│   │   ├── postgresql/
│   │   ├── spacy/
│   │   ├── tensorflow/
│   │   └── pytorch/
│   ├── middleware/
│   ├── tests/
│   ├── utils/
│   └── main.py
├── deployments/
│   ├── docker/
│   │   ├── Dockerfile
│   │   └── docker-compose.yml
│   ├── kubernetes/
│   │   ├── deployment.yml
│   │   ├── service.yml
│   │   ├── ingress.yml
│   │   └── configmap.yml
│   ├── scripts/
```

```
│   │   ├── deploy.sh
│   │   ├── start.sh
│   │   └── stop.sh
│   └── CI_CD/
│       ├── .github/
│       │   └── workflows/
│       │       └── ci_cd_pipeline.yml
│       └── jenkins/
│           └── Jenkinsfile
├── logs/
│   ├── backend.log
│   ├── frontend.log
│   ├── integration.log
│   └── error.log
├── scripts/
│   ├── setup/
│   ├── maintenance/
│   └── backup/
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── .gitignore
├── README.md
└── LICENSE
```

EXPLANATION OF DIRECTORY STRUCTURE

`config/`
- Contains configuration files for the entire project, including environment-specific settings.

`docs/`
- Holds all documentation, including API references, UI/UX guidelines, architecture diagrams, requirements, and user manuals.

`src/`
- The main source code directory, divided into backend, frontend, integration, middleware, tests, and utilities.

`backend/`
- Contains backend codebase including controllers, models, views, services, and configurations.

`frontend/`
- Contains frontend codebase using React, including components, pages, services, store, and utilities.
  - public/: Static assets, CSS, JS, and images.
  - src/: Source code for the React application.
    - components/: Reusable React components.
    - pages/: React components for different pages/views.
    - services/: Services for API calls and business logic.
    - store/: State management (e.g., Redux).
    - utils/: Utility functions and helpers.
    - App.js: Main App component.

- index.js: Entry point for the React application.
- setupTests.js: Configuration for testing.

`integration/`
- Contains subdirectories for each integrated service or application, such as Chatwoot, Kamailio, FreeSwitch, etc.

`middleware/`
- Contains code for middleware functionalities.

`tests/`
- Contains unit tests, integration tests, and end-to-end (e2e) tests.

`utils/`
- Contains utility scripts and helpers.

`deployments/`
- Contains Docker and Kubernetes deployment configurations and scripts for CI/CD pipelines.

`logs/`
- Directory for storing log files for backend, frontend, integration, and errors.

`scripts/`
- Contains scripts for setup, maintenance, and backups.

`.gitignore`
- Specifies files and directories to be ignored by Git.

`README.md`
- Provides an overview and instructions for the project.

`LICENSE`
- Contains the project's licensing information.

This structure ensures clear separation of concerns, facilitates easy maintenance, and supports scalable growth as the project evolves.