Architectural Design for AI-Powered Appointment Booking System

DISCLAIMER:

This document presents a high-level implementation and architectural overview excerpted from a live, enterprise-grade project. Code snippets and design components included herein are illustrative and do not represent the complete source code or full system architecture. Confidentiality and privacy considerations have been strictly observed to ensure no proprietary or sensitive information is disclosed.

1. Overview and Objectives

The system is an AI-driven appointment booking platform that enables users to book, cancel, reschedule, or check doctor availability via voice interactions (Twilio) and receive SMS confirmations. It integrates with Calendly for scheduling, ElevenLabs for TTS, and AWS S3 for audio storage, with a knowledge base extracted from PDFs. The architecture aims to:

- Scalability: Handle thousands of concurrent voice calls and SMS messages.

- Reliability: Ensure high availability with fault tolerance.

- Security: Protect sensitive data (e.g., phone numbers, API keys) and prevent injection attacks.

- Maintainability: Use modular microservices with CI/CD pipelines.

- AI Integration: Leverage TTS and potential NLP for enhanced user interaction.

2. System Components

The architecture is divided into microservices, each corresponding to a script's functionality, deployed as containerized services in a Kubernetes cluster on AWS EKS. Below is a breakdown of components, their roles, and technologies.

2.1. Microservices

1. Twilio Handler Service (`twilio_handler.py`)

- Purpose: Entry point for incoming Twilio calls, providing a voice-based menu for user interactions (book, cancel, reschedule, check availability, or connect to a human agent).
- Tech Stack:
  - Framework: Flask 3.0.3 (REST API for Twilio webhooks).
  - Twilio SDK: twilio 9.3.0 (TwiML for voice responses).
  - TTS: ElevenLabs via `tts.py` (elevenlabs 1.7.0, model: `eleven_multilingual_v2`, voice ID: `3UFZ7Pkyx3hNTropzBlS`).
  - Validation: Input sanitization (`_sanitize_input`).
- Endpoints:
  - `POST /voice`: Handles incoming calls, prompts DTMF input, redirects to `/process-selection`.
- Dependencies: `Flask`, `twilio`, `elevenlabs` (via `tts.py`), `pydantic` (via `receptionist_agent.py`).

2. Intent Handler Service (`intent_handler.py`)

- Purpose: Processes DTMF inputs (1–5) to redirect to appropriate actions (e.g., booking, cancellation).

- Tech Stack:

  - Framework: Flask 3.0.3.

  - Twilio SDK: twilio 9.3.0.

  - TTS: ElevenLabs via `tts.py`.

  - Validation: Input sanitization (`_sanitize_input`), environment variable for human support number.

- Endpoints:

  - `POST /process-selection`: Maps DTMF digits to actions (e.g., `/book-appointment`, `/cancel-appointment`).

- Dependencies: `Flask`, `twilio`, `elevenlabs`, `pydantic`.

3. Booking Handler Service (`booking_handler.py`)

- Purpose: Manages voice-based appointment booking, capturing doctor name, time, user details, and creating Calendly appointments.

- Tech Stack:

  - Framework: Flask 3.0.3.

  - Twilio SDK: twilio 9.3.0 (voice responses, speech recognition via `Gather`).

  - TTS: ElevenLabs via `tts.py`.

  - Validation: `ReceptionistAgent` (pydantic 2.9.2) for guest data, input sanitization.

  - Session Management: In-memory `secure_session_data` (to be replaced with Redis).

- Endpoints:

  - `POST /book-appointment`: Initiates booking.

- `POST /capture-doctor-name`, `/capture-appointment-time`, `/capture-user-name`, `/capture-user-phone`, `/capture-user-address`: Sequential steps for data capture.

- Dependencies: `Flask`, `twilio`, `elevenlabs`, `pydantic`, `requests` (via `calendly.py`), `boto3` (via `tts.py`).

4. Calendly Service (`calendly.py`)

- Purpose: Integrates with Calendly API to check availability and create appointments.

- Tech Stack:

  - HTTP Client: requests 2.32.3.

  - Validation: `ReceptionistAgent` for user data, input sanitization (`_sanitize_input`, `_parse_datetime`).

  - API Key: Stored in environment variable (`CALENDLY_TOKEN`).

- Functions:

  - `is_time_available`: Checks slot availability via Calendly API.

  - `create_calendly_appointment`: Creates a one-time scheduling link.

- Dependencies: `requests`, `pydantic`.

5. Knowledge Base Service (`knowledge_base.py`)

- Purpose: Extracts doctor availability from PDFs and provides availability checks.

- Tech Stack:

  - PDF Parser: PyPDF2 3.0.1.

- Validation: `ReceptionistAgent` for doctor names, input sanitization (`_sanitize_path`, `_sanitize_doctor_name`).

- Storage: In-memory `DOCTOR_AVAILABILITY` (to be replaced with a database).

- Functions:

- `extract_doctor_availability`: Parses PDFs to populate availability.

- `is_doctor_available`: Checks availability for a given doctor and time.

- Dependencies: `PyPDF2`, `pydantic`.

6. Messaging Service (`messaging.py`)

- Purpose: Sends SMS confirmations for appointments using Twilio.

- Tech Stack:

- Twilio SDK: twilio 9.3.0.

- Validation: `ReceptionistAgent` for phone numbers, input sanitization (`_sanitize_input`, `_sanitize_url`).

- Credentials: Environment variables (`TWILIO_SID`, `TWILIO_AUTH_TOKEN`, `TWILIO_FROM_NUMBER`).

- Functions:

- `send_confirmation_sms`: Sends SMS with appointment details.

- Dependencies: `twilio`, `pydantic`.

7. TTS Service (`tts.py`)

- Purpose: Generates TTS audio using ElevenLabs and stores it in AWS S3.

- Tech Stack:

  - TTS: elevenlabs 1.7.0.

  - Storage: boto3 1.35.24 (AWS S3).

  - Validation: `ReceptionistAgent` for text, input sanitization (`_sanitize_text`).

  - Credentials: Environment variables (`ELEVENLABS_API_KEY`, `S3_BUCKET`, `AWS_REGION`).

- Functions:

  - `generate_audio`: Generates audio and returns an S3 URL.

- Dependencies: `elevenlabs`, `boto3`, `pydantic`.

8. Receptionist Agent Service (`receptionist_agent.py`)

- Purpose: Provides data validation using Pydantic models (assumed `GuestInfo`).

- Tech Stack:

  - Validation: pydantic 2.9.2.

- Functions:

  - `validate_input`: Validates guest data (name, email, phone, purpose).

- Dependencies: `pydantic`.

2.2. External Services

- Twilio: Handles voice calls and SMS, integrated via webhooks and SDK.

- Calendly: Manages appointment scheduling and availability.

- ElevenLabs: Provides TTS with the `eleven_multilingual_v2` model.

- AWS S3: Stores TTS audio files for Twilio playback.

- AWS Secrets Manager: Stores sensitive credentials (API keys, Twilio tokens).

2.3. Data Storage

- Redis: Replaces in-memory `secure_session_data` (in `booking_handler.py`) for session management, with a 24-hour TTL.

- PostgreSQL: Stores extracted doctor availability (from `knowledge_base.py`) and user data, replacing in-memory `DOCTOR_AVAILABILITY`.

- AWS S3: Stores PDFs (`knowledge_base.py`) and TTS audio (`tts.py`).

2.4. Infrastructure

- AWS EKS: Kubernetes cluster for deploying microservices as Docker containers.

- AWS ALB: Application Load Balancer for routing HTTP requests to Flask endpoints.

- AWS CloudWatch: Centralized logging and monitoring.

- AWS SNS/SQS: Asynchronous messaging for inter-service communication (e.g., SMS notifications).

- GitHub Actions: CI/CD pipeline for automated testing and deployment.

3. System Architecture Diagram

https://www.mermaidchart.com/raw/362708c3-43eb-4e58-92ad-cc988ae000ff?theme=light&version=v0.1&format=svg

4. Data Flow

1. User Interaction:

- A user calls a Twilio number, triggering a webhook to the `Twilio Handler Service` (`/voice`).

- The service plays a TTS greeting (via `TTS Service`) and prompts for DTMF input using `Gather`.

2. DTMF Processing:

- DTMF input is sent to the `Intent Handler Service` (`/process-selection`), which redirects to endpoints like `/book-appointment` (handled by `Booking Handler Service`).

3. Booking Process:

- The `Booking Handler Service` captures voice inputs (doctor name, time, user details) via `Gather`, using `TTS Service` for prompts.

- It validates inputs with `Receptionist Agent Service` and checks availability via `Calendly Service` or `Knowledge Base Service`.

- Upon successful booking, it creates a Calendly appointment and sends an SMS via `Messaging Service`.

4. Availability Check:

- The `Calendly Service` queries the Calendly API for real-time availability.

- The `Knowledge Base Service` provides a fallback by checking PDF-extracted availability in PostgreSQL.

5. TTS Generation:

- The `TTS Service` generates audio using ElevenLabs, uploads it to S3, and returns URLs for Twilio playback.

- Audio is cached in S3 with a lifecycle policy (e.g., 7-day expiration).

6. Session Management:

- Session data is stored in Redis, accessed by `Booking Handler Service` for multi-step interactions.

7. Logging and Monitoring:

- All services log to `.log` files, aggregated in CloudWatch.

- Metrics (e.g., call duration, API latency) are monitored via CloudWatch.

5. Technical Specifications

5.1. Microservice Deployment

- Containerization: Each service is packaged as a Docker container using a `Dockerfile`:

```dockerfile
```

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["gunicorn", "--bind", "0.0.0.0:8080", "app:app"]
```

- Kubernetes: Deployed in EKS with:

  - Pods: One pod per service, with 2 replicas for high availability.

  - Services: ClusterIP for internal communication, LoadBalancer for ALB integration.

  - HPA: Horizontal Pod Autoscaling based on CPU/memory (80% threshold).

  - ConfigMaps/Secrets: Store environment variables (e.g., `TWILIO_SID`, `ELEVENLABS_API_KEY`) in Secrets.

5.2. Security Measures

- Authentication:

  - API keys (Twilio, Calendly, ElevenLabs) stored in AWS Secrets Manager, accessed via IAM roles.

  - Twilio webhooks validated using Twilio's `X-Twilio-Signature`.

- Encryption:

  - Data in transit: TLS 1.3 for all HTTP requests (ALB, API calls).

  - Data at rest: S3 buckets encrypted with AES-256, PostgreSQL with AWS RDS encryption.

- Input Validation:

  - Sanitization functions (`_sanitize_input`, `_sanitize_text`, `_sanitize_url`, `_sanitize_path`) across services.

  - Pydantic validation via `Receptionist Agent Service`.

- Network Security:

  - VPC with private subnets for EKS, RDS, and Redis.

  - Security Groups restrict ingress/egress (e.g., ALB only exposes port 443).

  - WAF (Web Application Firewall) on ALB to block SQL injection and XSS.

5.3. Data Models

- PostgreSQL Schema:

  ```sql
  CREATE TABLE doctors (
      id SERIAL PRIMARY KEY,
      name VARCHAR(100) NOT NULL,
      department VARCHAR(50),
      availability JSONB  -- Stores day, start_time, end_time
  );
  CREATE TABLE appointments (
      id SERIAL PRIMARY KEY,
      user_name VARCHAR(100),
      phone VARCHAR(20),
      email VARCHAR(100),
  ```

```
    doctor_id INT REFERENCES doctors(id),
    appointment_time TIMESTAMP,
    calendly_link TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- Redis Keys:
  - `session:<call_sid>`: Stores session data (e.g., doctor, time) with 24-hour TTL.

5.4. AI/ML Integration

- TTS: ElevenLabs `eleven_multilingual_v2` model provides natural voice responses, enhancing user experience.
- Future NLP: Integrate an NLP model (e.g., BERT-based intent classifier on AWS SageMaker) to improve speech recognition accuracy in `booking_handler.py`, replacing Twilio's `command_and_search` speech model.
- Availability Prediction: Train a time-series model (e.g., Prophet on SageMaker) to predict doctor availability based on historical Calendly data, supplementing `knowledge_base.py`.

5.5. CI/CD Pipeline

- GitHub Actions Workflow:

```yaml
name: CI/CD
on:
```

```
  push:
    branches: [main]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v5
        with: { python-version: '3.11' }
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest tests/
      - name: Build Docker image
        run: docker build -t appointment-system .
      - name: Push to ECR
        run: |
          aws ecr get-login-password | docker login --username AWS --password-stdin <ecr-repo>
          docker tag appointment-system <ecr-repo>:latest
          docker push <ecr-repo>:latest
      - name: Deploy to EKS
        run: |
```

```
kubectl apply -f k8s/deployment.yaml
kubectl apply -f k8s/service.yaml
```

- Testing: Unit tests for sanitization, validation, and API calls using `pytest`.

5.6. Monitoring and Alerts

- CloudWatch:
  - Logs: Aggregate `.log` files from all services.
  - Metrics: Track API latency, error rates, Twilio call duration, and S3 upload times.
  - Alarms: Alert on 5xx errors, high latency (>1s), or low availability.
- Sentry: Real-time error tracking for unhandled exceptions.
- Health Checks: Kubernetes liveness/readiness probes for each service.

6. Performance and Scalability

- Load Handling:
  - EKS Autoscaling: Cluster autoscaler adds nodes based on pod demand.
  - HPA: Scales pods based on CPU/memory usage.
  - Redis: Handles session data for thousands of concurrent calls.
- Latency Optimization:

- TTS Caching: Cache common audio responses in S3 with CloudFront CDN.

- API Calls: Use connection pooling for Twilio, Calendly, and ElevenLabs APIs.

- Throughput:

- Twilio can handle thousands of concurrent calls with proper webhook scaling.

- ALB distributes load across EKS pods.

7. Security and Compliance

- Data Privacy:

- Complies with HIPAA for handling user data (phone, name, email).

- Encrypts PII in PostgreSQL and S3.

- Access Control:

- IAM roles for EKS pods to access S3, Secrets Manager, and RDS.

- Least privilege principle for API keys.

- Audit Logging:

- CloudWatch logs track all API calls and user interactions.

- Enable AWS CloudTrail for auditing Secrets Manager access.

8. Future Enhancements

- NLP for Speech Recognition: Replace Twilio’s speech model with a custom NLP model (e.g., Whisper on SageMaker) for better accuracy in `booking_handler.py`.

- Multi-Channel Support: Add WhatsApp and email notifications via `messaging.py`.

- Analytics Dashboard: Build a dashboard (e.g., using AWS QuickSight) to visualize appointment trends and doctor availability.

- Multi-Language Support: Extend ElevenLabs language parameter in `tts.py` for multilingual TTS.

9. Deployment Plan

1. Setup:
   - Provision EKS cluster, RDS, Redis, and S3 using Terraform.
   - Configure Secrets Manager for API keys.
   - Set up ALB and WAF.

2. Build:
   - Create Docker images for each service.
   - Push to AWS ECR.

3. Deploy:
   - Apply Kubernetes manifests (`deployment.yaml`, `service.yaml`).
   - Configure HPA and cluster autoscaler.

4. Test:

- Run load tests using Locust to simulate 1,000 concurrent calls.
- Validate end-to-end flow (call → booking → SMS).

5. Monitor:

- Set up CloudWatch dashboards and Sentry alerts.
- Monitor for 99.9% uptime.

10. Conclusion

This architectural design leverages microservices, Kubernetes, and AI-driven TTS to deliver a scalable, secure, and user-friendly appointment booking system. The modular design ensures maintainability, while integrations with Twilio, Calendly, and ElevenLabs provide robust functionality. Security measures (encryption, validation, WAF) and monitoring (CloudWatch, Sentry) ensure enterprise-grade reliability.