Project Title: Inclusive holiday booking: target group-oriented websites with AI

1. Overview

This project aims to build a **CMS-driven AI-powered web platform** that dynamically adapts website design, data, and user experience based on different user personas with accessibility needs. The CMS acts as a control layer, defining UI parameters, data models, and AI configurations. These parameters are passed to the **frontend rendering engine**, which generates distinct user interfaces and interaction flows tailored to six accessibility personas.

For one persona (**Low Vision**), the system integrates a **3D AI Avatar** for problem-solving, bookings, and guided support. For the other personas, tailored UI adaptations improve ease of navigation, communication, and user experience.

---

2. Objectives

- Provide **personalized, accessible interfaces** for diverse user needs.
- Centralize **content, styling, and component control** via a CMS.
- Enable **AI-driven conversational and interactive experiences** for low vision users.
- Ensure **scalable, modular, and CMS-integrated frontend rendering**.
- Support **AI-based problem resolution and bookings** via backend RAG + avatars.

---

3. Personas & Design Requirements

The system supports six personas:

1. **Wheelchair Users**
    - Blue color scheme

   - Accessibility-focused content

2. **Low Vision (AI Avatar Focus)**
   - High contrast design
   - Larger text
   - Audio descriptions
   - AI Avatar (Azure 3D) for booking, problem-solving, Q&A
3. **Cognitive Impairment**
   - Simple layouts
   - Clear navigation
   - Step-by-step guided processes
4. **Anxiety Support**
   - Calming blue design
   - Flexible options
   - Clear and concise information
5. **Dyslexia**
   - Cream background
   - Arial font
   - Lowercase text
   - Audio support
6. **Hearing Impairment**
   - Visual emphasis
   - Written communication focus

- No dependency on audio

---

4. System Architecture

4.1 High-Level Flow

1. **CMS Layer**
   - Admins configure UI parameters (color, fonts, components, data feeds).
   - Defines AI behavior rules per persona (e.g., avatar for low vision, text focus for hearing impairment).
   - Stores metadata in SQLite.
2. **Backend Layer**
   - Fetches CMS parameters.
   - AI Orchestration with **Agentic RAG** for contextual responses.
   - Integrates **Azure Speech** and **Azure 3D Avatar** for conversational UX.
   - Web scraping via **Web4AI** for real-time external knowledge.
   - Exposes APIs through **Flask**.
3. **Frontend Layer**
   - React + Vite + Tailwind-based rendering.
   - Persona-specific UI generation.
   - Node.js + TypeScript for logic, dynamic component loading.
   - Integration with avatar for low vision persona.

4.2 System Diagram

```
+---------------------+       +-------------------+      +--------------------+
|       CMS           |  ---> |     Backend       | ---> |    Frontend        |
| (Content & Styles)  |       | (Flask + AI APIs) |      | (React + Vite)     |
+---------------------+       +-------------------+      +--------------------+
        |                     |                          |
        |                     |                          |
        v                     v                          v
  SQLite (Config)       OpenAI + Azure Speech + RAG     Persona-specific UI
```

---

5. Technical Stack

5.1 Frontend

- **React** – UI framework
- **Node.js** – runtime environment
- **TypeScript** – type safety
- **Tailwind CSS** – design system
- **Vite** – build tool for fast bundling
- **JavaScript** – dynamic logic

5.2 Backend

- **Python + Flask** – API layer
- **SQLite** – lightweight DB for CMS configurations
- **Pydantic AI** – validation and data models
- **Agentic RAG** – retrieval-augmented generation for contextual answers
- **Web4AI** – web scraping for external data integration

- **Azure Speech Services** – TTS & STT integration
- **Azure 3D Avatar** – interactive avatar for low vision persona
- **OpenAI APIs** – NLP and reasoning layer

---

6. Module Breakdown

6.1 CMS Module

- UI parameter configuration (colors, fonts, layouts, accessibility features).
- Persona rules definition.
- Stores data in SQLite.

6.2 Backend Services

- **Persona Service**: fetch CMS rules per persona.
- **RAG Service**: contextual problem resolution.
- **Avatar Service**: AI avatar integration with Azure APIs.
- **Speech Service**: text-to-speech & speech-to-text.
- **Scraper Service**: external data fetching for FAQs, real-time info.

6.3 Frontend Engine

- **Persona Renderer**: generates persona-specific UI.
- **Dynamic Components Loader**: loads CMS-defined UI components.
- **Accessibility Enhancer**: applies accessibility rules (contrast, font size, etc.).
- **Avatar Interface**: integrates 3D avatar for low vision persona.

---

7. AI & Accessibility Flow

Low Vision Persona Example

1. User logs in → CMS rules applied.
2. High-contrast UI + large fonts rendered.
3. AI Avatar appears for booking/issues.
4. User speaks → **Azure Speech** → **Flask** → **RAG** → **Response**.
5. Avatar replies with speech + animations.

Other Personas Example

1. User logs in → CMS rules applied.
2. Persona-specific UI (dyslexia → cream bg + Arial font).
3. Interaction supported via clear navigation, simplified layouts, or visual focus.

---

8. Security & Compliance

- Role-based access to CMS.
- Secure API authentication between frontend and backend.
- Data encryption for sensitive inputs (bookings, issues).
- Compliance with **WCAG 2.1** accessibility standards.

---

9. Future Enhancements

- Multi-language support.
- Real-time analytics for persona engagement.
- Adaptive AI learning from user behavior.
- Integration with additional accessibility tools (screen readers, haptic feedback).
