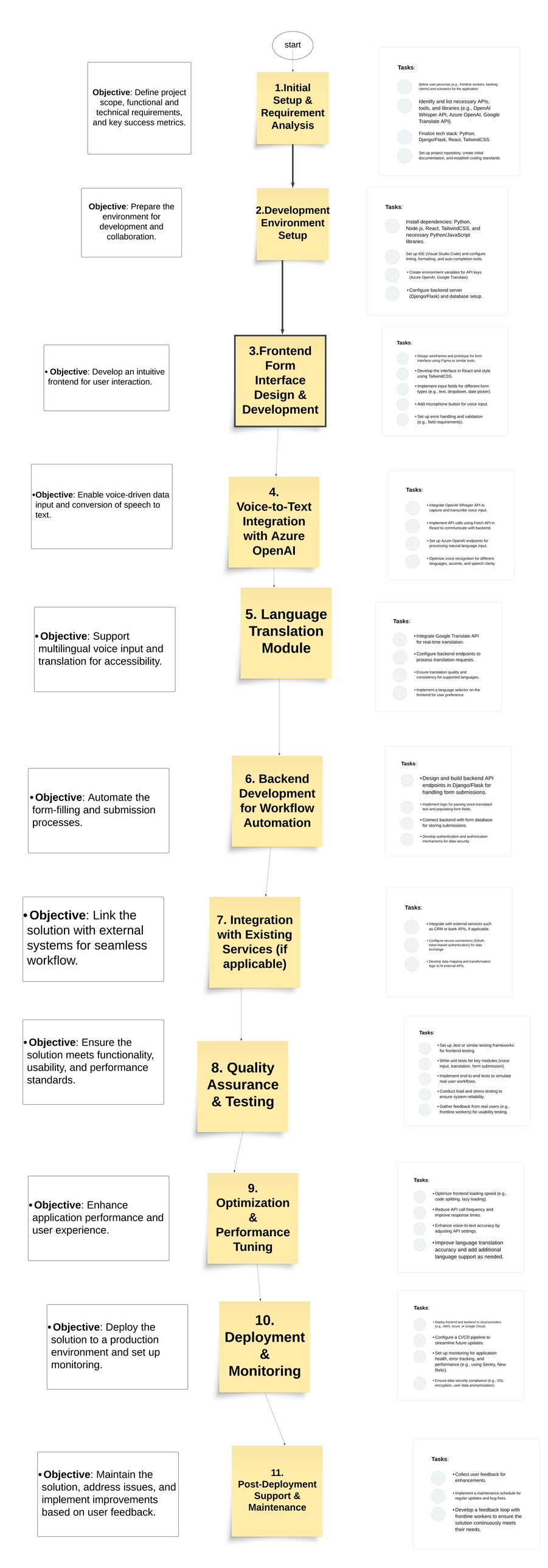
start
Objective: Define project scope, functional and technical requirements, and key success metrics.
1.Initial Setup & Requirement Analysis
Tasks:
Define user personas (e.g., frontline workers, banking clients) and scenarios for the application
Identify and list necessary APIs, tools, and libraries (e.g., OpenAI Whisper API, Azure OpenAI, Google Translate API).
Finalize tech stack: Python, Django/Flask, React, TailwindCSS.
Set up project repository, create initial documentation, and establish coding standards.
Objective: Prepare the environment for development and collaboration.
2.Development Environment Setup
Tasks:
Install dependencies: Python, Node.js, React, TailwindCSS, and necessary Python/JavaScript libraries.
Set up IDE (Visual Studio Code) and configure linting, formatting, and auto-completion tools.
• Create environment variables for API keys (Azure OpenAI, Google Translate).
• Configure backend server (Django/Flask) and database setup.
• Objective: Develop an intuitive frontend for user interaction.
3.Frontend Form Interface Design & Development
Tasks:
• Design wireframes and prototype for form interface using Figma or similar tools.
• Develop the interface in React and style using TailwindCSS.
• Implement input fields for different form types (e.g., text, dropdown, date picker).
• Add microphone button for voice input.
• Set up error handling and validation (e.g., field requirements).
•Objective: Enable voice-driven data input and conversion of speech to text.
4. Voice-to-Text Integration with Azure OpenAI
Tasks:
• Integrate OpenAI Whisper API to capture and transcribe voice input.
• Implement API calls using Fetch API in React to communicate with backend.
• Set up Azure OpenAI endpoints for processing natural language input.
• Optimize voice recognition for different languages, accents, and speech clarity.
• Objective: Support multilingual voice input and translation for accessibility.
5. Language Translation Module
Tasks:
• Integrate Google Translate API for real-time translation.
• Configure backend endpoints to process translation requests.
• Ensure translation quality and consistency for supported languages.
• Implement a language selector on the frontend for user preference.
• Objective: Automate the form-filling and submission processes.
6. Backend Development for Workflow Automation
Tasks:
• Design and build backend API endpoints in Django/Flask for handling form submissions.
• Implement logic for parsing voice-translated text and populating form fields.
• Connect backend with form database for storing submissions.
• Develop authentication and authorization mechanisms for data security.
• Objective: Link the solution with external systems for seamless workflow.
7. Integration with Existing Services (if applicable)
Tasks:
• Integrate with external services such as CRM or bank APIs, if applicable.
• Configure secure connections (OAuth, token-based authentication) for data exchange.
• Develop data mapping and transformation logic to fit external APIs.
• Objective: Ensure the solution meets functionality, usability, and performance standards.
8. Quality Assurance & Testing
Tasks:
• Set up Jest or similar testing frameworks for frontend testing.
• Write unit tests for key modules (voice input, translation, form submission).
• Implement end-to-end tests to simulate real user workflows.
• Conduct load and stress testing to ensure system reliability.
• Gather feedback from real users (e.g., frontline workers) for usability testing.
• Objective: Enhance application performance and user experience.
9. Optimization & Performance Tuning
Tasks:
• Optimize frontend loading speed (e.g., code splitting, lazy loading).
• Reduce API call frequency and improve response times.
• Enhance voice-to-text accuracy by adjusting API settings.
• Improve language translation accuracy and add additional language support as needed.
• Objective: Deploy the solution to a production environment and set up monitoring.
10. Deployment & Monitoring
Tasks:
• Deploy frontend and backend to cloud providers (e.g., AWS, Azure, or Google Cloud).
• Configure a CI/CD pipeline to streamline future updates.
• Set up monitoring for application health, error tracking, and performance (e.g., using Sentry, New Relic).
• Ensure data security compliance (e.g., SSL encryption, user data anonymization).
• Objective: Maintain the solution, address issues, and implement improvements based on user feedback.
11. Post-Deployment Support & Maintenance
Tasks:
• Collect user feedback for enhancements.
• Implement a maintenance schedule for regular updates and bug fixes.
• Develop a feedback loop with frontline workers to ensure the solution continuously meets their needs.