# AI-Powered Chatbot: Project Outline

This is a step-by-step guide to building, deploying, and improving your AI-powered social interaction chatbot, based on the provided project specification.

### Phase 1: Foundation & Basic Chat (Estimated: Weeks 1-3)

**Goal:** To build the core, functional skeleton of the application. By the end of this phase, you will have a simple, working chatbot.

* **Step 1: Set Up Your Development Environment**
  + Install necessary tools: Node.js (for React), Python, Docker, and Git.
  + Create a project folder and initialize a Git repository.
* **Step 2: Build the Backend Foundation**
  + Set up a simple Python backend using the **Flask** framework.
  + Create a basic API endpoint (e.g., /chat) that can receive and respond with JSON data.
* **Step 3: Create the Frontend Interface**
  + Initialize a **React.js** application.
  + Build a simple chat interface with a message display area and a text input field.
  + Connect the frontend to the backend API, allowing users to send a message and see a hardcoded response.
* **Step 4: Set Up the Database**
  + Create a new project in **Google Cloud Firestore**.
  + Connect your Python backend to Firestore.
* **Step 5: Implement User Authentication & Security**
  + Create a users collection in Firestore.
  + Build signup and login functionality.
  + Implement **JSON Web Token (JWT)** authentication to secure your API endpoints.
* **Step 6: Develop the Basic AI**
  + Create a simple, **rule-based dialogue engine** in your backend. It should look for specific keywords in the user's message and return a pre-defined response.
  + Implement basic **conversation logging**, saving each message to a conversations collection in Firestore.

### Phase 2: Refinement & Initial Deployment (Estimated: Weeks 3-8)

**Goal:** To containerize the application, set up an automated deployment pipeline, and make the first version publicly accessible.

* **Step 1: Containerize the Application with Docker**
  + Write a Dockerfile for your React frontend.
  + Write a Dockerfile for your Python backend.
  + Create a docker-compose.yml file to easily run both services together for local development.
* **Step 2: Set Up CI/CD (Continuous Integration/Continuous Deployment)**
  + Push your code to a **GitHub** repository.
  + Create a **GitHub Actions** workflow (.github/workflows/main.yml).
  + Configure the workflow to:
    1. Trigger on pushes to the main branch.
    2. Build the Docker images for the frontend and backend.
    3. Run any automated tests you've written.
* **Step 3: Deploy the Application**
  + Deploy your backend Docker container to a service like **Google Cloud Run**.
  + Deploy your frontend to a static hosting service like **Firebase Hosting**.
  + Configure your GitHub Actions workflow to automatically deploy new versions when tests pass.
* **Step 4: Refine the User Experience**
  + Improve the styling and usability of the chat interface.
  + Refine the AI's rules based on initial testing and feedback.

### Phase 3: Analytics & Feedback (Estimated: Weeks 3-12)

**Goal:** To integrate analytics and feedback mechanisms to understand user interaction and guide future improvements.

* **Step 1: Integrate Web Analytics**
  + Add **Google Analytics** to your React application to track user events like conversation\_start and message\_sent.
* **Step 2: Implement a User Feedback System**
  + Add a "Send Feedback" or rating button to the chat interface.
  + Create a new API endpoint to receive this feedback.
  + Store the feedback in a feedback collection in Firestore.
* **Step 3: Introduce Basic NLP**
  + Begin exploring NLP libraries like **scikit-learn** or **spaCy**.
  + Start building a simple **intent recognition** model to categorize user messages (e.g., "greeting," "question," "goodbye") instead of relying purely on keywords.
* **Step 4: Develop Analytics Scripts**
  + Write Python scripts to query your Firestore data and generate simple reports on:
    - Most common user questions.
    - Conversation length.
    - User satisfaction based on feedback.

### Phase 4: Future Enhancements & Polish (Ongoing)

**Goal:** To evolve the chatbot into a more intelligent and capable conversational agent.

* **Step 1: Enhance AI Context**
  + Modify the dialogue engine to retain **context** from previous messages in a conversation.
* **Step 2: Improve AI Robustness**
  + Implement **clarification capabilities**, allowing the AI to ask for more information when it doesn't understand a query.
* **Step 3: Explore Actionable Responses**
  + Investigate how the AI could perform simple actions for the user (e.g., fetching weather data from an external API).
* **Step 4: Strengthen Security**
  + Implement an **audit trail** for important security events (e.g., failed logins).
  + Develop a strategy for detecting and **anonymizing** any sensitive personal data that might appear in conversation logs.
* **Step 5: Iterate and Improve**
  + Continuously use the analytics and feedback from Phase 3 to refine and retrain your AI models.