USSD Architecture Plan

For implementing **USSD** in India, you'll need a provider or platform that works with local Indian mobile network operators (MNOs). Here are some practical options for using USSD in India:

1. Partnering Directly with Indian Mobile Network Operators (MNOs)

Description:

- Major telecom operators in India like **Airtel**, **Jio**, **Vodafone Idea**, and **BSNL** provide direct USSD gateway services.
- You can integrate directly with their USSD gateways for production-level deployments.

Steps:

1. Contact the operator's enterprise/business services division.
2. Negotiate terms and pricing for USSD sessions (usually charged per session or per query).
3. Integrate your application with their USSD gateway using the APIs or protocols they provide.

Pros:

- Reliable and production-grade.
- Direct access to the telecom network ensures high performance and low latency.
- Access to nationwide USSD services.

Cons:

- Typically requires significant setup fees and commercial agreements.
- Best suited for medium to large-scale projects with a clear business use case.

---

2. Using Aggregators/USSD Gateway Providers

Description:

Aggregators act as intermediaries between your application and telecom operators. They provide APIs to access USSD services.

Popular Options in India:

1. **Karix Mobile (formerly mGage):**
    - Offers enterprise-grade USSD solutions.
    - Works with multiple telecom operators in India.
2. **Gupshup:**
    - Primarily an SMS/Chatbot platform but can arrange USSD services through operator partnerships.
3. **Route Mobile:**
    - Provides USSD gateways for businesses.

Steps:

1. Choose an aggregator and create an account.
2. Work with their team to set up your USSD application.
3. Use their API documentation to integrate the USSD gateway into your backend (e.g., Java Spring Boot).

Pros:

- Easier setup than dealing directly with operators.
- Aggregators handle the complexities of operator integration.

Cons:

- Additional costs (aggregators charge a markup on operator fees).
- Dependency on the aggregator for support and service quality.

---

3. Open Source Solutions with a Local GSM Modem

Description:

- Use open-source platforms like **Kannel** or **Gammu** with a GSM modem and a local SIM card.
- This is ideal for small-scale or testing purposes.

Steps:

1. Install and configure **Kannel** or **Gammu** on your Linux server.
2. Connect a GSM modem to your system with a local SIM card (ensure the SIM supports USSD).
3. Use AT commands to send and receive USSD requests through the modem.

Pros:

- No dependency on third-party providers.
- Cost-effective for development and testing.

Cons:

- Limited scalability.
- Requires technical expertise to set up and maintain.
- Dependent on the GSM modem's capabilities.

4. Hosted Solutions for USSD

If you're looking for a **plug-and-play solution**, consider platforms that offer USSD hosting for applications.

Option: Africa's Talking (for testing)

- **Availability in India**: Africa's Talking doesn't support live USSD in India but offers a sandbox environment for free USSD flow testing.
- Use their platform to prototype and later migrate to an Indian provider for production.

Steps:

1. Sign up for Africa's Talking.

2. Test your USSD flows in their sandbox.
3. Replace the sandbox API with an Indian provider when you go live.

Pros:

- Easy to use for development and testing.
- No setup cost for the sandbox.

Cons:

- Can't be used in production in India.

---

5. API-Based Platforms with Global Reach

Some platforms offer limited USSD services in India but might require custom arrangements:

1. **Vonage (Nexmo)**:
    - Supports USSD in specific regions. Check with them for India-specific capabilities.
2. **Infobip**:
    - Enterprise-focused communication platform that supports USSD in some markets, including India on request.

Steps:

1. Contact their sales/support to verify coverage.
2. Integrate their USSD API into your backend.

---

6. High-Level Architecture:

User (via USSD/SMS) ↔ Telecom Network ↔ Kannel Gateway ↔ Spring Boot Backend ↔ MQTT Broker ↔ Other Services.

Flow Breakdown:

1. **User (via USSD/SMS)**:

- The user interacts with the application using **USSD** or **SMS** on their mobile device. The input could be a request for information, a service initiation (like applying for a loan), or something else based on the application flow.

2. **Telecom Network**:
    - The request from the user is transmitted over the **telecom network** (cellular network). USSD requests are typically processed via the GSM network, while SMS is sent over the mobile network.
3. **Kannel Gateway**:
    - The **Kannel Gateway** handles the **USSD/SMS** requests from the telecom network. It acts as a bridge between the mobile network and your backend system. It translates USSD or SMS requests into HTTP API calls that the backend can understand.
4. **Spring Boot Backend**:
    - Your **Spring Boot application** processes the incoming request. It could query a database, perform business logic, or interact with external services. Based on the user's request, it prepares a response.
5. **MQTT Broker**:
    - The **MQTT Broker** is used for real-time message exchange. Once the Spring Boot backend has processed the request, it publishes a message (e.g., a loan approval message or status update) to the MQTT broker, which will distribute it to any subscribed clients, such as mobile apps or other services.
6. **Other Services**:
    - The **Other Services** component could be anything from a notification service (email/SMS) to external integrations (like a loan processing service, external APIs, or data analytics). These services can subscribe to the MQTT messages for further action.

Comparison of Options:

Provider	Best For	Cost	Scalability	Ease of Use
Direct with MNOs	Large-scale production	High	High	Moderate
Aggregators	Medium-to-large businesses	Medium	High	High
Kannel/Gammu	Small-scale/testing	Low	Low	Low
Africa's Talking	Testing/development	Free (sandbox)	NA in India	High
Vonage/Infobip	Enterprise/global projects	Medium to High	High	High

---

Recommendation :

1. **For Production in India**:
   - Contact a local telecom operator or aggregator like Karix Mobile or Route Mobile.
   - These are reliable, scalable, and tailored for Indian telecom networks.
2. **For Testing/Prototyping**:
   - Use Africa's Talking sandbox to develop USSD workflows.
   - Or, set up **Kannel** with a GSM modem for cost-effective testing.
3. **For Long-Term Projects**:
   - Partner with an aggregator like Gupshup or Infobip for easier scaling.