Banking Chatbot - Complete Use Cases & Flows Documentation

Table of Contents

---

1. User Authentication Flows

1.1 Complete Authentication Flow (New Session)

Description: Full authentication process for new users with flexible input support.

Trigger: User sends greeting or starts conversation

Flow:

```
Step 1: Initial Greeting
User: "Hi" / "Hello" / "Hey"
→ Bot: Welcome message + CNIC request

Step 2: Flexible CNIC Verification
User: "my cnic is 42501-5440926-9" / "this is my cnic 42501-5440926-9" / "42501-5440926-9"
→ System: Extracts CNIC using regex pattern
→ Backend: Verifies CNIC in database
→ If valid: Bot requests OTP
→ If invalid: Bot asks for correct format

Step 3: OTP Verification
User: "1234" (any 1-5 digit number)
→ System: Validates OTP format
→ If valid: Bot shows available accounts
→ If invalid: Bot requests valid OTP format

Step 4: Smart Account Selection
User: "my usd account" / "first account" / "1234" (last 4 digits)
→ System: Uses smart selection algorithm
→ Backend: Verifies account ownership
→ If valid: Full authentication complete
→ If invalid: Bot provides guidance

Step 5: Ready for Banking
→ Bot: Confirmation message + available services
```

Example Conversation:

```
User: "Hi there"
Bot: "Hello! I'm Sage, your banking assistant. To get started securely, could you please provide your CNIC in the format 12345-1234567-1?"

User: "sure, my cnic is 42501-5440926-9"
Bot: "Great! For additional security, please provide an OTP. You can enter any number between 1-5 digits."

User: "123"
Bot: "OTP verified! I found 2 accounts for you. Please select your account by saying 'my USD account', 'my PKR account', 'first account', or entering the last 4 digits."

User: "my usd account"
Bot: "Perfect! Your USD account ending in 7890 is now selected. You're all set for banking! How can I help you today?"
```

1.2 CNIC Verification Sub-Flows

1.2.1 Valid CNIC Formats Accepted:

- Direct: `"42501-5440926-9"`
- Natural: `"my cnic is 42501-5440926-9"`
- Casual: `"this is my cnic 42501-5440926-9"`
- Formal: `"here is my cnic: 42501-5440926-9"`

1.2.2 Invalid CNIC Handling:

```
User: "my cnic is 123456"
→ Bot: "Please provide your CNIC in the correct format: 12345-1234567-1 (5 digits - 7 digits - 1 digit)"

User: "I don't know my cnic"
→ Bot: "Your CNIC is your national identification number. It should be in the format 12345-1234567-1. Please check your ID card."
```

1.3 Smart Account Selection Sub-Flows

1.3.1 Currency-Based Selection:

```
User: "my usd account" → Selects first USD account
User: "my pkr account" → Selects first PKR account
User: "pakistani rupee account" → Selects first PKR account
```

1.3.2 Position-Based Selection:

```
User: "first account" → Selects account[0]
User: "second account" → Selects account[1]
User: "third account" → Selects account[2]
```

1.3.3 Traditional Selection (Backward Compatibility):

```
User: "1234" → Finds account ending with 1234
```

1.3.4 Type-Based Selection:

```
User: "savings account" → Prefers PKR accounts
User: "current account" → Selects first available
```

---

2. Balance Inquiry Flows

2.1 Simple Balance Check

Description: User wants to check current account balance.

Trigger: Keywords like "balance", "money", "funds", "how much"

Flow:

```
User: "What's my balance?" / "How much money do I have?"
→ System: Queries latest transaction for account_balance
→ Backend: Returns current balance with currency
→ AI Agent: Generates natural response with balance info
```

Example Conversation:

```
User: "what's my balance?"
Bot: "Your current balance is $2,450.75 USD. Your account is looking healthy!"
```

2.2 Savings Goal / Affordability Check

Description: User wants to know if they can afford something or reach a savings goal.

Trigger: Keywords like "afford", "target", "goal", "can I buy"

Flow:

```
User: "Can I afford a car that costs 1000000?" / "Do I have enough for 500000?"
→ System: Extracts target amount using regex
→ System: Compares current balance vs target
→ AI Agent: Provides affordability analysis + savings advice
```

Example Conversation:

```
User: "can I afford a house that costs 5000000 PKR?"
Bot: "You currently have 2,340,000 PKR. To reach your target of 5,000,000 PKR, you'd need an additional 2,660,000 PKR. Would you like me to help you create a savings plan?"
```

2.3 Financial Planning Query

Description: User asks for financial advice or planning.

Trigger: Keywords like "save", "planning", "goal", "what can I do"

Flow:

```
User: "What can I do to hit my target of 1 million?" / "How should I save?"
→ System: Gets current balance
→ AI Agent: Provides personalized financial guidance
```

---

3. Transaction History Flows

3.1 General Transaction History

Description: User wants to see recent transactions.

Trigger: Keywords like "transactions", "history", "show", "list"

Flow:

```
User: "Show me my transactions" / "Transaction history"
→ System: Defaults to last 20 transactions
→ MongoDB: Queries with sort by date descending
→ AI Agent: Formats transactions naturally with dates, amounts, descriptions
```

Example Conversation:

```
User: "show me my recent transactions"
Bot: "Here are your recent transactions:

1. Jul 24, 2025 - Netflix Subscription - $15.99 USD (Entertainment)
2. Jul 23, 2025 - Uber Ride - $12.50 USD (Transport)
3. Jul 22, 2025 - Amazon Purchase - $89.99 USD (Shopping)
..."
```

3.2 Limited Transaction History

Description: User wants specific number of transactions.

Trigger: "last X transactions", "recent Y transactions"

Flow:

```
User: "Last 5 transactions" / "Show me 10 recent transactions"
→ LLM: Extracts number from query
→ System: Applies limit to MongoDB query
→ AI Agent: Shows exact number requested
```

3.3 Monthly Transaction History

Description: User wants transactions for specific month.

Trigger: Month names like "May", "June", "July"

Flow:

```
User: "Transactions in May" / "Show me June transactions"
→ System: Detects month using keyword matching
→ MongoDB: Queries with date range for full month
→ AI Agent: Shows all transactions for that month
```

Example Conversation:

```
User: "transactions in june"
Bot: "Here are all your transactions from June 2025 (total: 28 transactions):

June 30 - Grocery Store - $45.67 PKR (Food)
June 29 - Gas Station - $30.00 PKR (Transport)
..."
```

4. Spending Analysis Flows

4.1 Simple Spending Query

Description: User wants to know spending on specific merchant/service.

Trigger: "How much did I spend on X"

Flow:

```
User: "How much did I spend on Netflix?" / "Netflix spending"
→ LLM: Extracts description="netflix", type="debit"
→ MongoDB: Aggregates all Netflix transactions
→ AI Agent: Returns total amount with context
```

Example Conversation:

```
User: "how much did I spend on uber this month?"
Bot: "You spent $67.50 USD on Uber rides this month across 5 trips. Your average ride cost was $13.50."
```

4.2 Category Spending Analysis

Description: User wants spending breakdown by category.

Trigger: "How much on food", "entertainment expenses"

Flow:

```
User: "How much did I spend on food last month?"
→ LLM: Extracts category="Food", month="june", type="debit"
→ MongoDB: Groups by category and sums amounts
→ AI Agent: Provides category breakdown with insights
```

Example Conversation:

```
User: "how much did i spend on food in may?"
Bot: "You spent $450.75 USD on food in May. This included $280 on restaurants, $120 on groceries, and $50.75 on food delivery services."
```

4.3 Comparative Spending Analysis

Description: User wants to compare spending across time periods.

Trigger: "more than", "compared to", "vs", "versus"

Flow:

```
User: "Did I spend more in May than April?" / "Am I spending more right now than in April?"
→ LLM: Generates $facet pipeline for both periods
→ MongoDB: Parallel aggregation for comparison periods
→ AI Agent: Provides comparative analysis with insights
```

Example Conversation:

```
User: "am I spending more right now than I did in april?"
Bot: "You've spent $1,240 USD so far in July compared to $1,890 USD in April. You're actually spending $650 less this month! This is mainly due to reduced shopping expenses."
```

4.4 Spending Breakdown Analysis

Description: User wants to understand what caused high spending.

Trigger: "What caused my high spending", "spending breakdown"

Flow:

```
User: "What caused my high spending in May?"
→ System: Analyzes all May debit transactions
→ System: Groups by category, finds largest transactions
→ AI Agent: Provides detailed breakdown with top categories
```

Example Conversation:

```
User: "what drove my high spending last month?"
Bot: "Your spending in June totaled $2,340 USD. The main drivers were:
1. Shopping (45%) - $1,053, mainly from Amazon purchases
2. Food (25%) - $585, mostly restaurant dining
3. Transport (20%) - $468, primarily Uber rides
4. Entertainment (10%) - $234, Netflix and streaming services"
```

4.5 Spending Patterns Analysis

Description: User wants general spending insights and patterns.

Trigger: "spending patterns", "spending habits", "analyze my spending"

Flow:

```
User: "Analyze my spending patterns" / "What are my spending habits?"
→ System: Comprehensive analysis across multiple months
→ System: Identifies trends, averages, and patterns
→ AI Agent: Provides insights and recommendations
```

---

5. Money Transfer Flows

5.1 Complete Transfer Flow (with Confirmation)

Description: User wants to transfer money with enhanced security.

Trigger: "transfer", "send money", "pay"

Flow:

```
Step 1: Transfer Request
User: "Transfer 500 PKR to John" / "Send 100 USD to Alice"
→ LLM: Extracts amount, currency, recipient
→ System: Validates completeness
→ If complete: Proceed to OTP

Step 2: Transfer OTP Request
→ System: Sets TRANSFER_OTP_PENDING state
→ Bot: Requests OTP for security

Step 3: Transfer OTP Verification
User: "1234" (OTP)
→ System: Validates OTP format
→ If valid: Move to confirmation step

Step 4: Transfer Confirmation (NEW)
→ System: Sets TRANSFER_CONFIRMATION_PENDING state
→ Bot: "Do you want to confirm transfer of 500 PKR to John?"

Step 5: User Confirmation
User: "yes" / "confirm" → Proceed with transfer
User: "no" / "cancel" → Cancel transfer
User: unclear → Ask for clarification

Step 6: Transfer Execution
→ Backend: Executes transfer, updates balance
→ AI Agent: Confirms successful transfer with details
```

Example Conversation:

```
User: "transfer 500 pkr to ahmad"
Bot: "To complete the transfer of 500 PKR to Ahmad, please provide an OTP (any number between 1-5 digits)."

User: "12345"
Bot: "OTP verified! Do you want to confirm the transfer of 500 PKR to Ahmad? Please reply with 'yes' to confirm or 'no' to cancel."

User: "yes confirm it"
Bot: "Transfer completed successfully! 500 PKR has been sent to Ahmad. Your new balance is 4,500 PKR. Transaction ID: TXN123456789"
```

5.2 Transfer Cancellation Flow

Description: User cancels transfer during confirmation.

Flow:

```
User: "no" / "cancel" / "don't send it"
→ System: Clears pending transfer
→ System: Returns to ACCOUNT_SELECTED state
→ Bot: Confirms cancellation + account security
```

Example:

```
User: "no, cancel it"
Bot: "Transfer cancelled, Sarah. The transfer of 500 PKR to Ahmad has been stopped and your account is secure. Is there anything else I can help you with?"
```

5.3 Incomplete Transfer Information

Description: User provides incomplete transfer details.

Flow:

```
User: "transfer money to john" (missing amount)
→ LLM: Detects missing information
→ AI Agent: Asks for missing details naturally
```

Example:

```
User: "send money to sara"
Bot: "I'd be happy to help you send money to Sara! How much would you like to transfer and in which currency (PKR or USD)?"
```

5.4 Transfer Error Handling

Description: Transfer fails due to insufficient funds or other errors.

Flow:

```
User confirms transfer → Backend processes → Insufficient funds
→ Bot: "Sorry, you don't have enough balance. You have 300 PKR but need 500 PKR for this transfer."
```

---

6. Session Management Flows

6.1 Session Start Flow

Description: User initiates new banking session.

Flow:

```
User: Greeting → Bot: Welcome + Authentication request
→ Complete authentication flow → Ready for banking
```

6.2 Session End Flow

Description: User terminates banking session securely.

Trigger: "exit", "quit", "logout", "end"

Flow:

```
User: "exit"
→ System: Clears all user state and memory
→ System: Security cleanup
→ AI Agent: Personalized goodbye message
→ User must re-authenticate for new session
```

Example:

```
User: "exit"
Bot: "Thanks for banking with us today, Sarah! Your session has been securely closed and all temporary data cleared.
To start a new session, just say 'hi' and I'll help you get authenticated again. Have a great day!"
```

6.3 Session Timeout/Cleanup

Description: Automatic cleanup of inactive sessions.

Flow:

```
Inactive for 1 hour → System: Automatic cleanup
Next message → Bot: "Session expired, please restart"
```

---

7. Error Handling & Edge Cases

7.1 Invalid Input Handling

7.1.1 Invalid CNIC Format:

```
User: "my cnic is 123456"
Bot: "Please provide your CNIC in the correct format: 12345-1234567-1"
```

7.1.2 Invalid OTP Format:

```
User: "abc123"
Bot: "Please enter a valid OTP (any number between 1-5 digits)."
```

7.1.3 Invalid Account Selection:

```
User: "my savings account" (when no savings account exists)
Bot: "I found these accounts for you: USD account ending in 7890, PKR account ending in 1234. Please select one by saying 'my USD account' or 'my PKR account'."
```

7.2 Database/Backend Errors

Flow:

```
User query → Backend error → AI Agent: Natural error message + retry suggestion
```

Example:

```
User: "show my balance"
Backend: Connection timeout
Bot: "I'm having trouble accessing your account information right now. Please try again in a moment."
```

7.3 Rate Limiting

Flow:

```
User sends multiple messages quickly (< 2 seconds apart)
→ Bot: "I appreciate your enthusiasm! Please give me just a moment to process your previous message."
```

---

8. Contextual Query Handling

8.1 Follow-up Questions

Description: User asks follow-up questions based on previous context.

Flow:

```
Previous: "How much did I spend in May?"
User: "What about June?" / "And June?"
→ System: Uses conversation memory to understand context
→ System: Applies same query type to June
```

Example:

```
Bot: "You spent $1,200 USD in May on various expenses."
User: "what about june?"
Bot: "In June, you spent $1,450 USD, which is $250 more than May."
```

8.2 Referential Queries

Description: User refers to previous results using "this", "that", "it".

Flow:

```
Previous: Shows transaction list
User: "Can you analyze this?" / "What category is this?"
→ System: References previous context
```

8.3 Conversation Memory

Description: System maintains context across conversation.

Features:

- Remembers last 4 messages for context
- Stores query context for references
- Clears memory on session end for security

---

9. Non-Banking Query Handling

9.1 Clearly Non-Banking Queries

Description: User asks about non-banking topics.

Trigger: Weather, sports, politics, entertainment, etc.

Flow:

```
User: "What's the weather today?" / "Who won the football game?"
→ System: Keyword detection + LLM filtering
→ Bot: Polite decline + redirect to banking services
```

Example:

```
User: "what's the weather like today?"
Bot: "I'm sorry, but I can only help with banking and financial questions. I don't have information about weather in my database. Is there anything related to your banking needs I can help you with instead?"
```

9.2 Ambiguous Queries

Description: Queries that could be banking or non-banking.

Flow:

```
User: "check 3" / "what about last month?"
→ System: Errs on side of allowing (could be banking context)
→ Processes as potential banking query
```

9.3 General Help Queries

Description: User asks about bot capabilities.

Flow:

```
User: "What can you do?" / "Help me"
→ AI Agent: Explains available banking services
```

Example:

```
User: "what can you do?"
Bot: "I'm your personal banking assistant! I can help you with:
- Check your account balance
- View transaction history
- Analyze your spending patterns
- Transfer money securely
- Answer questions about your finances

What would you like to do today?"
```

Testing Scenarios

Quick Test Cases for Each Flow:

1. **Authentication**: "Hi" → "my cnic is 42501-5440926-9" → "123" → "my usd account"
2. **Balance**: "what's my balance?"
3. **Transactions**: "show last 5 transactions"
4. **Spending**: "how much did I spend on netflix in june?"
5. **Transfer**: "transfer 100 pkr to john" → "1234" → "yes"
6. **Exit**: "exit"
7. **Non-banking**: "what's the weather?"
8. **Error**: Invalid inputs at each step

Edge Cases to Test:

1. **Flexible CNIC**: Try various formats
2. **Smart Account Selection**: Try all selection methods
3. **Transfer Confirmation**: Test yes/no/unclear responses
4. **Contextual Queries**: Follow-up questions
5. **Rate Limiting**: Send messages quickly
6. **Session Timeout**: Wait and try to use expired session

Flow State Diagram

```
NOT_VERIFIED → CNIC_VERIFIED → OTP_VERIFIED → ACCOUNT_SELECTED
                             ↓
Banking Queries ← → TRANSFER_OTP_PENDING → TRANSFER_CONFIRMATION_PENDING
                             ↓
                        Transfer Execution
```

This document covers all use cases and flows in the banking chatbot system. Each flow is designed to be user-friendly, secure, and handle edge cases gracefully.