# **Business Requirements Document (BRD)**

# **SNEA: Simple Network Engine Assistant**

### **Document Control**

**Version:** 1.1 **Date:** April 14, 2025  
**Author:** Sneha AJITHKUMAR  
**Status:** Draft **Last Updated:** April 14, 2025

## **1. Executive Summary**

SNEA (Simple Network Engine Assistant) is a conversational AI chatbot designed to provide banking information and assistance through a user-friendly interface. This personal project aims to create a functional chatbot similar to HDFC EVA while leveraging Firebase, Dialogflow, and other modern tools to deliver a seamless user experience.

The project will be developed following an agile methodology with a one-week timeline for the minimum viable product (MVP). This comprehensive BRD outlines the business requirements, technical specifications, architectural design, and implementation approach to guide development. It serves as both a project roadmap and technical documentation for future reference or potential expansion.

By combining cloud-based services with conversational AI capabilities, SNEA will demonstrate how modern tools can be leveraged to create intelligent customer service solutions. The document includes detailed module breakdowns for sprint planning, algorithmic approaches, technical architecture, and a comprehensive tech stack analysis to ensure reusability and extensibility of the developed components.

## 

## **2. Project Overview**

### **2.1 Background**

Banking chatbots like HDFC EVA have revolutionized customer service in the financial sector by providing instant access to information and assistance through conversational interfaces. These virtual assistants help customers with account inquiries, transaction details, product information, and general banking guidance without requiring human intervention for routine matters. This shift has improved customer satisfaction while reducing operational costs for banking institutions.

This project seeks to build a similar capability as a learning exercise and personal project, incorporating practical skills in cloud services, conversation design, and workflow automation. By implementing SNEA, we aim to understand the architectural and technical considerations of building a conversational AI system while creating a portfolio-worthy demonstration of these capabilities.

### **2.2 Objectives**

The primary objectives of the SNEA project are to:

* Create a functional banking chatbot with natural language understanding capabilities that can recognize and respond to common banking queries with human-like conversation.
* Implement core features that demonstrate capabilities similar to banking assistants including account information retrieval, product explanations, and service guidance.
* Learn and apply Firebase, Dialogflow, and n8n technologies in a practical context to understand their integration points and capabilities.
* Follow proper software development lifecycle practices including requirements gathering, design, implementation, testing, and documentation.
* Document the development process for future reference with comprehensive technical specifications and architectural decisions.
* Build a portfolio-worthy project demonstrating conversational AI skills that could be presented to potential employers or clients.
* Create reusable components that could be applied to future projects or extended for more complex implementations.
* Establish a foundation for potential expansion into more advanced banking assistant capabilities.

### 

### **2.3 Success Criteria**

The SNEA project will be considered successful if it achieves the following outcomes:

* SNEA correctly responds to at least 90% of test queries within defined conversation flows, demonstrating appropriate understanding of user intent and context.
* User interface is responsive and accessible across devices with a clean, intuitive interaction model and minimal response latency.
* Chatbot can handle basic banking information requests including account status queries, product information, and service details using simulated data.
* Project is completed within the one-week timeline with all core modules functional and tested.
* Code and documentation are well-structured for future enhancements with clear architectural patterns and development guidelines.
* Integration of at least one workflow automation using n8n that demonstrates potential for backend process integration.
* Clear documentation of architecture and conversation design that explains implementation decisions and technical approaches.
* Project can be demonstrated to showcase technical capabilities with a smoothly functioning end-to-end user experience.
* Reusable components are identified and documented for potential use in future projects.
* Technical implementation follows best practices for security, performance, and maintainability within the constraints of the timeline.

### **2.4 Scope**

#### **In Scope:**

The SNEA project will include the following capabilities and components:

* Development of conversational AI using Firebase and Dialogflow to create a natural language understanding system capable of recognizing common banking queries and extracting relevant entities.
* Implementation of static responses for banking information including account types, loan products, credit cards, fees, and interest rates.
* Basic user interface for chatbot interaction with text input/output, quick reply options, and structured information display.
* Integration with n8n for workflow automation to demonstrate the potential for process automation and external system integration.
* Documentation of development process and architecture including technical specifications, conversation flows, and integration patterns.
* User authentication and session management to personalize interactions and maintain conversation context.
* Simulated account data for demonstration purposes that illustrates how actual banking data would be presented.
* Banking product information (accounts, loans, credit cards) with features, benefits, and application requirements.
* Service location information (branches, ATMs) with basic search capabilities using predefined locations.
* FAQ handling for common banking queries with appropriate responses and follow-up suggestions.
* Simple analytics for conversation tracking to measure interaction effectiveness and popular topics.
* Implementation of reusable components with documentation for future development.
* Architectural foundation for potential extensions and improvements.

#### **Out of Scope:**

The following capabilities are explicitly excluded from the initial SNEA implementation:

* Integration with actual banking systems for real account information or transactions.
* Real user authentication with banking credentials or financial know-your-customer (KYC) verification.
* Production-level security measures beyond basic authentication and data protection.
* Mobile app development (web-based interface only) as a native application would expand the timeline significantly.
* Multi-language support (English-only for initial release) to maintain focus on core functionality.
* Advanced personalization based on user history or preferences beyond basic session context.
* Real-time transaction processing or payment initiation capabilities.
* Integration with payment gateways or financial processing systems.
* Voice interface (text-only for initial release) as speech recognition would add complexity.
* Customer support ticketing system integration for escalation management.
* Regulatory compliance features for financial services beyond basic disclaimers.
* Advanced analytics and reporting beyond basic conversation tracking.
* Machine learning capabilities beyond those provided by Dialogflow.
* Biometric authentication methods.

**3. Stakeholders**

### **3.1 Primary Stakeholder**

The primary stakeholder for the SNEA project is:

* **Project Owner/Developer: Sneha AJITHKUMAR**  
   As the project owner and developer, Sneha will be responsible for all aspects of the project from requirements gathering through implementation and documentation. She will make all key decisions regarding technical approach, design choices, and feature prioritization. The project serves as both a learning opportunity and portfolio piece for her.

### **3.2 Secondary Stakeholders**

Secondary stakeholders who have an interest in the project outcomes include:

* **Ajithkumar KUMARAN** As a collaborator or advisor, Ajithkumar may provide input on technical decisions, review progress, and offer feedback throughout the development process.
* **Future users of the chatbot** Though this is primarily a personal project, potential users represent a stakeholder group whose needs should be considered in the design and implementation of the chatbot interface and functionality.
* **Developers interested in similar projects** The documentation and code may serve as a reference for other developers undertaking similar projects, making the technical choices and architecture important for educational purposes.
* **Potential portfolio reviewers** Employers, clients, or other professionals who might review the project as part of a portfolio assessment represent a stakeholder group with interest in the quality of implementation, documentation, and problem-solving approaches demonstrated.

## 

## **4. Functional Requirements**

### **4.1 Conversation Capabilities**

#### **4.1.1 Basic Conversation**

The foundation of SNEA's functionality is its ability to engage in natural conversation with users. This capability includes:

**Greeting and Introduction**

SNEA will welcome users with a friendly, professional greeting that establishes the chatbot's identity and purpose. The introduction will briefly explain SNEA's capabilities and suggest ways the user might begin the conversation. The greeting will adapt to recognize returning users versus new interactions, and support various greeting inputs (hello, hi, hey, etc.) with appropriate responses.

The conversation initiation will include:

* Welcome message with SNEA identification
* Brief overview of capabilities
* Suggestion chips for common starting points
* Option to type a free-form question

**Small Talk Management**

Beyond task-oriented conversation, SNEA will handle casual conversation elements that humanize the interaction. This includes appropriate responses to pleasantries, gratitude expressions, and casual questions. While not the primary focus, these capabilities maintain conversation flow and prevent frustration when users employ natural conversation patterns.

**Small talk capabilities include:**

* Responding to greetings and farewells
* Acknowledging gratitude expressions
* Brief responses to personal questions about the chatbot
* Professional handling of casual comments

**Help and Guidance**

When users are unsure how to proceed or what capabilities are available, SNEA will provide clear guidance on available features and how to interact effectively. This guidance system serves as an in-conversation user manual that adapts to the user's current context and previous interactions.

**Help functionality includes:**

* Explanation of available features and services
* Example questions the chatbot can answer
* Navigation assistance through menu options
* Topic suggestions based on popular queries
* Guidance back to main topics when conversation diverges

**Context Management**

To create a cohesive conversation experience, SNEA will maintain contextual awareness across multiple interaction turns. This allows users to ask follow-up questions without repeating all details and enables the chatbot to provide more relevant responses based on the conversation history.

**Context management features include:**

* Maintaining active topics across multiple messages
* Remembering referenced entities (accounts, products)
* Supporting pronoun resolution for follow-up questions
* Allowing context switching while preserving previous context
* Managing context lifespan appropriately

#### **4.1.2 Banking Information**

A core function of SNEA is providing accurate information about banking products and services. This capability forms the knowledge base that powers most user interactions.

**Account Types Information**

SNEA will provide comprehensive information about various account types offered by banks. This information will be structured to address common questions about features, benefits, requirements, and comparison points between account types.

For each account type, SNEA can explain:

* Key features and benefits
* Minimum balance requirements
* Fee structures and conditions
* Interest rates and calculation methods
* Documentation requirements for opening
* Eligibility criteria
* Digital banking capabilities
* Comparison with other account types

**Loan Information**

Loan products represent complex financial instruments that often require detailed explanation. SNEA will provide information about common loan types, their features, application processes, and requirements.

**Loan information capabilities include:**

* Personal, home, vehicle, and education loan details
* Interest rate information and calculation methods
* Repayment terms and options
* Eligibility criteria and documentation
* Application processes and timelines
* EMI calculation explanation
* Prepayment and foreclosure terms
* Collateral requirements where applicable

**Credit Card Information**

Credit cards have distinct features, benefits, and terms that consumers need to understand. SNEA will provide detailed information about credit card options, rewards programs, and usage considerations.

**Credit card information includes:**

* Card types and their target demographics
* Annual fees and charges
* Reward points systems and redemption options
* Cashback programs and special offers
* Interest rates and billing cycles
* Credit limits and enhancement processes
* International usage policies
* Insurance and protection benefits

**Fees and Charges**

Banking services often come with various fees and charges that customers need to understand. SNEA will provide clear information about common banking fees, when they apply, and how they might be avoided.

**Fee information includes:**

* Account maintenance charges
* Transaction fees (ATM, IMPS, RTGS, NEFT)
* Minimum balance non-maintenance penalties
* Debit/credit card annual fees
* Loan processing fees
* Checking charges and statement fees
* Online/mobile banking charges
* Late payment penalties

**Interest Rates**

Interest rates are a critical factor in financial decision-making. SNEA will provide current interest rate information for various banking products and explain how interest calculations work.

**Interest rate information includes:**

* Savings account interest rates
* Fixed deposit rates for various tenures
* Loan interest rates (floating and fixed)
* Credit card interest calculation
* Compound interest explanations
* Tax implications of interest income
* Rate comparison between products
* Historical rate trends (simplified)

#### **4.1.3 Account Information (Simulated)**

To demonstrate how a production banking chatbot would handle personal financial information, SNEA will include simulation capabilities for account-related queries.

**Balance Inquiries**

SNEA will respond to balance inquiries with simulated data that demonstrates how actual account information would be presented. This simulation will include appropriate security acknowledgments and data presentation formats.

**Balance inquiry features include:**

* Available balance display for simulated accounts
* Multiple account type support (savings, current, credit card)
* Balance breakdown (available vs. holding)
* Mini statement option with recent transactions
* Clear indication that data is simulated
* Balance trend visualization (textual)

**Transaction History**

Transaction history provides insight into account activity over time. SNEA will simulate transaction history responses with realistic transaction types, descriptions, and amounts.

**Transaction history features include:**

* Recent transaction listing with dates and amounts
* Transaction categorization (spending categories)
* Search/filter capability by date range or type
* Transaction detail expansion
* Spending pattern observations
* Downloadable statement explanation (simulated)

**Statement Information**

Account statements are formal records of financial activity. SNEA will explain statement features, access methods, and interpretation guidelines.

**Statement information includes:**

* Statement generation process explanation
* Statement period and delivery options
* Statement format explanation and sample
* E-statement registration process
* Statement archival and retrieval information
* Understanding statement components guidance

#### **4.1.4 Service Information**

Beyond product and account information, banking customers often need information about operational aspects of banking services.

**Product Application Procedures**

SNEA will provide step-by-step guidance on application procedures for various banking products and services, explaining the process, requirements, and timelines.

Application procedure information includes:

* Account opening procedures (online and branch)
* Loan application processes and stages
* Credit card application methods
* Required documentation explanation
* Processing timelines and expectations
* Application status checking methods
* Common application issues and resolutions

**Documentation Requirements**

Banking services typically require identity verification and supporting documentation. SNEA will provide detailed information about documentation requirements for various services.

**Documentation information includes:**

* **I**dentity proof requirements (acceptable documents)
* Address proof documentation options
* Income proof requirements for credit products
* KYC process explanation
* Document verification procedures
* Special requirements for specific products
* Digital document submission guidelines

**Location Services**

Physical banking locations remain important for many services. SNEA will provide information about branch and ATM locations using simulated location data.

Location service information includes:

* Branch locations with addresses and contact numbers
* ATM locations and available features
* Branch operating hours and special timings
* Services available at specific locations
* Accessibility information
* Appointment booking guidance
* Location search by area/pincode (simulated)

**Customer Service Details**

When users need assistance beyond what the chatbot can provide, SNEA will offer appropriate customer service contact information and escalation paths.

Customer service information includes:

* Contact numbers for various departments
* Email addresses for specific queries
* Social media support channels
* Complaint registration procedures
* Feedback submission methods
* Escalation matrix for unresolved issues
* Expected response times

### **4.2 User Experience**

#### **4.2.1 Interface Requirements**

The user interface is critical to providing an engaging and effective chatbot experience. SNEA's interface will balance simplicity with functionality to create an intuitive user experience.

**Chat Interface Design**

The core of the user experience is the chat interface itself, which must facilitate natural conversation while providing appropriate visual structure and feedback.

**The chat interface will feature:**

A clean, minimalist design that focuses attention on the conversation itself while providing necessary context and controls. Messages will be clearly attributed with visual distinction between user and bot responses, including timestamps for reference and conversation flow understanding. Typing indicators will display when SNEA is preparing responses to set appropriate user expectations, particularly for complex queries requiring longer processing times.

The interface will support rich content display within the conversation stream, allowing for formatted text, tables, images, and interactive elements while maintaining a cohesive conversation experience. Message grouping will combine related information for readability while avoiding overwhelming message blocks.

The conversation history will be persistent within a session with smooth scrolling capabilities and a "load more" function for longer conversations. Users will have the ability to reference previous parts of the conversation easily, supporting the natural flow of human dialogue which often refers back to earlier statements.

**Quick Reply Options**

To streamline common interactions and guide users toward successful outcomes, SNEA will implement contextual quick reply options throughout conversations.

**Quick reply options will include:**

Contextually relevant suggestion buttons that appear after bot responses, offering logical next steps based on the current conversation state. These buttons will accelerate the conversation flow by minimizing typing for common responses while still allowing free-form input for complex queries.

Common query shortcuts will provide one-click access to frequently asked questions within the current context, such as "Check balance" or "See transactions" when discussing accounts. Category navigation buttons will help users explore major service areas quickly, functioning as a visual menu system embedded within the conversation.

For confirmation queries, simple yes/no options will streamline decision points while more complex choices will present clearly labeled options that balance brevity with clarity. All quick reply options will be designed for touch interaction on mobile devices while remaining keyboard-accessible for desktop users.

**Visual Information Display**

While conversation is the primary interaction mode, visual elements will enhance information clarity for complex or structured data.

**Visual information elements will include:**

Rich cards for product information that combine images, key details, and action buttons in a compact, scannable format. These cards will help users quickly grasp product features and benefits while maintaining the conversation context.

Tabular data will present fee structures, interest rates, and comparison information in a structured format that facilitates understanding of complex relationships between data points. For multiple options or products, carousel displays will allow horizontal scrolling through alternatives with consistent presentation.

Transaction history and account information will use structured list views with appropriate typography and color-coding to highlight important information. Visual hierarchy within these displays will guide attention to key information while providing access to details on demand.

All visual elements will be designed for responsive display across device sizes with appropriate scaling and reorganization based on available screen space.

**Responsive Design**

SNEA's interface must function effectively across a range of devices from mobile phones to desktop computers.

The responsive design approach includes:

A fluid layout that adapts to screen sizes from 320px to 2560px width, ensuring all content and functionality remains accessible regardless of device. The interface will prioritize content and controls differently based on available space, ensuring the most important elements are always readily accessible.

Touch-friendly design elements will accommodate finger interaction on mobile devices with appropriate target sizes and spacing, while still supporting precise mouse **interaction on** desktops. Font sizes and line heights will adjust across devices to maintain readability without requiring horizontal scrolling.

Media and visual elements will load appropriately for the user's connection speed and device capabilities, with optimized assets for different screen resolutions. The layout will work effectively in both portrait and landscape orientations on mobile devices, adapting the conversation display and input methods accordingly.

#### **4.2.2 Conversation Flow**

The conversation flow defines how interactions progress and how context is maintained throughout a user session.

**Natural Dialogue Management**

SNEA will strive to create conversations that feel natural and human-like while still operating within the constraints of a rule-based system.

**Natural dialogue management includes:**

Coherent conversation progression that builds logically from one exchange to the next without awkward transitions or repetitive phrasing. The system will acknowledge user inputs appropriately before providing new information, creating a sense of listening and understanding similar to human conversation.

Language patterns will avoid overly robotic or formulaic phrasing while maintaining a consistent brand voice that balances professionalism with approachability. The conversation will include appropriate confirmation and clarification requests when needed, but will avoid redundant verification when the intent is clear.

SNEA will demonstrate personality consistency throughout interactions with a defined character that remains stable across conversation topics and functions. This includes consistent formality level, terminology usage, and response styles.

**Context Retention**

Context awareness is essential for multi-turn conversations that feel cohesive rather than disconnected.

**Context retention capabilities include:**

Session memory that remembers referenced accounts, products, or topics within the current conversation, allowing users to ask follow-up questions without repeating identifiers. For example, after asking about savings accounts, a user could simply ask "What's the interest rate?" without needing to specify the account type again.

The system will maintain topic context across multiple question-answer pairs, recognizing when new questions relate to the previously established subject. This includes support for anaphora resolution, handling pronouns like "it" or "them" by connecting them to previously mentioned entities.

Context decay will be implemented thoughtfully, gradually reducing the influence of older conversation topics as new ones are introduced. This prevents context confusion while still allowing natural conversation flow. Users will have the option to explicitly change topics or clear context when desired.

**Fallback Strategies**

Even with well-designed conversation flows, users may input queries that the system doesn't understand or can't handle appropriately. Effective fallback strategies maintain a positive user experience in these scenarios.

**Fallback strategies include:**

Graceful handling of unrecognized inputs that acknowledges the limitation without breaking the conversational experience. Instead of generic "I don't understand" responses, SNEA will offer targeted suggestions based on partial understanding or conversation history.

When the intent is unclear but similar to known patterns, the system will offer suggestions that might match the user's goal, providing quick-access buttons to common intents. For example, "Did you want to know about our savings account interest rates or checking account fees?"

After multiple consecutive fallbacks, the system will implement escalation options such as suggesting more structured queries or offering alternative contact methods for complex issues. Throughout the fallback process, SNEA will maintain a helpful, non-frustrating tone that acknowledges limitations without creating user frustration.

**Conversation Control**

Users should feel in control of the conversation direction and flow, with clear options to navigate, restart, or refocus the interaction.

**Conversation control features include:**

Clear options to restart the conversation entirely when the user wants to begin fresh. This will reset all context and state information, effectively beginning a new session.

Users will have the ability to clear specific context while maintaining the overall conversation, allowing them to switch topics cleanly without starting over completely. For multi-step processes like applications or calculations, cancel options will be available at each stage.

A help command will be accessible throughout the conversation, providing contextual assistance based on the current conversation state. This creates a safety net for users who become confused or need guidance on available options.

## **5. Technical Requirements**

### **5.1 Platform Requirements**

#### **5.1.1 Development Technologies**

The SNEA implementation will utilize a stack of complementary technologies to deliver a complete conversational experience. These technologies have been selected based on their capabilities, integration potential, and alignment with project objectives.

Firebase Services

Firebase will serve as the primary backend platform for SNEA, providing a comprehensive suite of services that work together seamlessly. The Firebase ecosystem offers significant advantages for rapid development of cloud-connected applications with minimal backend configuration.

Firebase Hosting will deploy and serve the web application frontend, providing global CDN distribution, automatic SSL certification, and optimized asset delivery. This service will handle all static asset delivery including HTML, CSS, JavaScript, and media files, with versioned deployments that support rollback if needed. The hosting service integrates directly with other Firebase offerings for seamless security and authentication.

Firebase Authentication will manage user identity and session security, supporting email/password authentication with the option to add social login providers in the future. The authentication system will generate and validate JWT tokens for secure API access and maintain session state across page refreshes. User profile information will be minimally stored, focusing on essential identification rather than personal details.

Firestore will provide the document-oriented database backend for SNEA, storing conversation history, user preferences, and banking information. Its real-time capabilities will enable live updates to conversation interfaces without explicit polling, while offline support ensures the application remains functional during temporary connectivity issues. The flexible schema accommodates the varied data structures needed for different aspects of the application.

Firebase Cloud Functions will implement serverless backend logic, particularly for Dialogflow webhook fulfillment and data processing tasks. These functions will execute in response to specific triggers including HTTP requests, database changes, and authentication events. The serverless architecture eliminates the need for server provisioning and management while scaling automatically with demand.

Firebase Analytics will track usage patterns and conversation metrics to inform future improvements. Event logging will capture key user interactions such as conversation starts, intent triggers, and feature usage, while conversion tracking measures successful task completions.

**Conversational AI**

Dialogflow ES (Essentials) will provide the natural language understanding capabilities that power SNEA's conversational interface. As a managed NLU platform, it offers sophisticated intent recognition and entity extraction with minimal training data requirements.

The Dialogflow agent will be structured around intents that map user utterances to specific functionality, with training phrases that capture various ways users might express the same need. Entity recognition will extract structured data from natural language input, identifying account types, amounts, dates, and other parameters needed for processing.

Context management within Dialogflow will maintain conversation state across multiple turns, with carefully designed context lifespans to balance persistence against confusion. The platform's built-in small talk capabilities will be customized to match SNEA's personality and domain focus.

Webhook fulfillment implemented through Firebase Cloud Functions will connect Dialogflow's NLU capabilities to dynamic data sources and business logic. These webhooks will receive intent and entity information from Dialogflow, execute appropriate backend operations, and return structured responses that Dialogflow can deliver to the user.

**Frontend Development**

The frontend implementation will focus on creating a responsive, accessible chatbot interface using standard web technologies without heavy framework dependencies.

HTML5 will provide the structural foundation with semantic markup that enhances accessibility and SEO performance. The structure will separate conversation elements from application controls and use appropriate ARIA attributes to support screen readers and other assistive technologies.

CSS3 will implement the visual design with a mobile-first responsive approach using Flexbox and Grid for layout management. CSS variables will define a consistent color scheme, typography, and spacing system that can be easily modified for future theming. Media queries will adapt the interface to different device sizes while maintaining functionality and readability.

JavaScript (ES6+) will handle client-side interactivity including user input processing, API communication, and dynamic interface updates. The implementation will use modern JavaScript features while ensuring compatibility with current browsers. Fetch API will manage asynchronous communication with backend services, with appropriate error handling and loading state management.

**Workflow Automation**

n8n will provide workflow automation capabilities that demonstrate integration potential beyond the core conversational interface. As an open-source workflow automation tool, n8n offers visual workflow creation with extensive integration options.

Workflows will be created for scenarios such as scheduled data processing, multi-step transaction analysis, and notification generation. These workflows will connect to Firebase services through webhook triggers and actions, demonstrating how external processes can integrate with the conversational system.

The n8n implementation will initially run locally for development purposes but could be deployed to a cloud environment for production use. Workflows will be documented as both visual diagrams and configuration exports to facilitate understanding and potential reimplementation.

Version Control and Collaboration

Git will provide version control throughout the development process, tracking code changes with descriptive commit messages that document evolution of the codebase. Branch management will separate feature development from the stable main branch, with merges occurring after feature completion and testing.

GitHub will host the repository with issue tracking for feature development and bug management. Documentation will be maintained in Markdown format directly within the repository, ensuring documentation and code remain synchronized. The repository structure will organize code logically by function with clear README files explaining each component.

#### **5.1.2 Integration Points**

The integration architecture defines how the various system components communicate and work together to create a cohesive application.

**Dialogflow-Firebase Integration**

The integration between Dialogflow and Firebase forms the core backend architecture of SNEA, connecting natural language understanding to data storage and processing.

Firebase Cloud Functions will serve as webhook endpoints for Dialogflow, receiving intent information and entities extracted from user queries and generating appropriate responses. These functions will follow a consistent pattern of intent validation, entity extraction, data retrieval or processing, and response formatting.

Authentication between systems will use service account credentials stored securely as environment variables in Cloud Functions. Session identification will maintain conversation continuity across multiple interactions, with appropriate session timeout handling.

Context information will pass between systems using Dialogflow's context mechanism, with Cloud Functions both reading from and writing to this context store. Additional context information may be maintained in Firestore for more persistent state management across longer time periods.

**Frontend-Backend Integration**

The web frontend will connect to backend services to create a seamless user experience that combines local interface responsiveness with cloud-based processing.

The Firebase SDK will integrate directly with the web frontend, providing authentication services, real-time database connections, and analytics tracking. This direct integration minimizes custom API development and leverages Firebase's security model for data access control.

User authentication state will synchronize between client and server, with appropriate handling of sign-in, sign-out, and session expiration events. The interface will adapt dynamically based on authentication state, showing appropriate options for authenticated versus anonymous users.

Real-time updates will flow from Firestore to the interface using listeners that update the display when underlying data changes. This enables features like conversation history updates and status changes without explicit polling.

API calls to Cloud Functions will follow RESTful patterns with consistent error handling and loading state management. These calls will be authenticated using Firebase Auth tokens to ensure secure access to user-specific data and operations.

**n8n Workflow Integration**

n8n workflows will demonstrate how external systems can integrate with the conversational platform for automation and data processing.

Webhook endpoints will trigger workflows based on specific events within the SNEA system, such as new conversation starts, particular intent detections, or scheduled maintenance tasks. Webhooks will pass contextual data to workflows for processing and receive results for storage or presentation.

Data processing workflows will analyze conversation patterns, extract insights from transaction data, and generate reports that could inform future system improvements. These demonstrations will simulate how production banking systems might integrate with customer-facing conversational interfaces.

Notification workflows will generate alerts based on specific triggers, demonstrating how a production system might notify users about account activity, security events, or marketing opportunities. While notifications will be simulated rather than delivered to actual channels, the workflows will include the necessary formatting and channel-specific adaptations.

**Development Environment**

* **Local Server**: For frontend development with hot reload.
* **Firebase Emulator Suite**: For offline testing of Auth, Functions, and Firestore.
* **Postman**: API testing tool for simulating webhook payloads.
* **Environment Management**: .env files and Firebase config for switching between dev/prod setups.
* **Deployment Pipelines**: Optionally automated using GitHub Actions.

### **5.2 Security Requirements**

The security architecture for SNEA prioritizes privacy, integrity, and access control, even in a demonstration context. Although the application operates on simulated data and does not interface with real banking systems, it is designed with production-grade security principles to foster best practices and ensure secure interactions.

##### **5.2.1 Data Protection**

SNEA adheres to data minimization and least privilege principles. It only collects the information strictly required to demonstrate its functionality. No actual user financial data is captured or stored. All user-identifying information is limited to basic credentials such as email addresses. These are securely stored using Firebase Authentication and are accessible only through authorized roles.

Simulated data used within the chatbot (such as balances, account statements, or transaction details) is clearly marked and isolated from any real user input. Firestore security rules are rigorously defined to allow access solely to authenticated users and limit their scope strictly to their own session data. User data such as conversation history and session metadata are encrypted at rest, while data in transit is protected using HTTPS and SSL encryption protocols.

Users are provided options to delete their chat history at any point, and these deletions are processed instantly. The data retention policy is configurable, and although the demonstration system does not persist data indefinitely, it lays a foundation for production-level governance. Additionally, conversation logs and analytics intentionally exclude personally identifiable information, instead focusing on aggregate metrics such as top intents and usage frequency.

To further protect conversation history, sensitive content is excluded from logs and analytics. Inactive sessions are purged automatically after a defined timeout window, ensuring stale data is removed unless explicitly retained. The Firestore access model ensures isolation of user documents, preventing cross-user access at the document level.

API access and cloud function invocations are authenticated via Firebase Auth tokens, with server-side verification before performing any operation. Upload functionality, if included (such as for KYC simulation), is safeguarded with file size and MIME type restrictions to mitigate injection and upload risks. Rate limiting middleware will be introduced to prevent brute-force or denial-of-service style attacks on sensitive endpoints.

##### **5.2.2 Authentication**

SNEA uses Firebase Authentication as its identity provider, offering robust and flexible authentication strategies. For the MVP version, email/password sign-in is enabled, with enforced password strength rules and email verification. The system does not allow unauthenticated access to protected resources.

Session continuity is achieved using JWTs (JSON Web Tokens) provided by Firebase. These tokens are securely stored on the client side and refreshed as needed to maintain active sessions without constant re-authentication. Tokens are short-lived by design and are accompanied by refresh tokens that are periodically validated.

Social login integration via providers like Google is available for expansion and will be added as an optional feature. Firebase’s built-in user management tools simplify password resets and multi-device session control. Session logs can be made available to users to view active sessions and revoke them manually.

For admin-level access, separate authentication is enforced. Admin routes are secured using both role-based access control (RBAC) and function-level guards. Administrative capabilities are limited to analytics viewing, intent management, and session monitoring in this phase. Every admin interaction is logged with timestamps and user IDs to ensure accountability.

Security testing procedures are integrated into the development cycle. Manual and automated tests simulate common web vulnerabilities such as XSS, CSRF, and injection attacks. Authorization boundaries are strictly enforced to ensure users cannot escalate privileges or access resources outside their role.

All these practices, even in a demonstration context, serve to mirror the standards required in real-world banking applications, making SNEA a secure and scalable foundation for future enhancement.

### **6. Implementation Approach**

The development of SNEA will follow an **Agile methodology** optimized for solo or small-team execution, with a strong focus on iterative delivery, rapid feedback, and continuous documentation. The implementation approach is designed to balance structured software engineering principles with the flexibility required for exploratory development, enabling both the completion of a minimum viable product (MVP) and the incorporation of emerging ideas.

#### **6.1 Agile Development Methodology**

SNEA will be developed using short, focused sprints with clearly defined goals. Each sprint represents a logical module of the system—frontend UI, backend integration, NLP agent design, automation workflows, or analytics—and concludes with internal testing and documentation.

A typical sprint will consist of the following phases:

* **Sprint Planning:** Definition of goals, deliverables, and technical scope.
* **Design & Prototyping:** Rapid mockup and architecture alignment.
* **Implementation:** Functional coding and integration of assigned components.
* **Verification:** Manual testing, validation, and documentation.
* **Review:** Sprint retrospective to assess quality, blockers, and improvements.

Each feature or module will be developed independently, with code merged into the main branch only after it passes validation checks. This modular approach allows for parallel refinement, ease of debugging, and rapid iteration.

#### **6.2 Milestone Planning and MVP Roadmap**

The roadmap for delivering the MVP is structured into manageable milestones, aligned with Agile sprints and tracked via version control and sprint documentation.

| **Milestone** | **Deliverable** |
| --- | --- |
| Sprint 1 | Firebase & Dialogflow Setup |
| Sprint 2 | Frontend UI Development (HTML/CSS/JS) |
| Sprint 3 | Dialogflow Intent & Entity Modeling |
| Sprint 4 | Cloud Function Fulfillment & Firestore Integration |
| Sprint 5 | Authentication + Session Management |
| Sprint 6 | n8n Workflow Automation |
| Sprint 7 | Testing, QA, and Final Deployment |

While each sprint has predefined outputs, the project maintains flexibility to reallocate efforts based on unexpected complexity or emerging ideas. Daily check-ins or reflections will guide this realignment process.

#### **6.3 Modular Development Strategy**

SNEA’s codebase will be architected to support modularity and future extensibility. Every feature—from the chatbot interface to backend logic—is isolated in its own module or component, governed by interface contracts and versioning rules.

For instance:

* **UI modules** (chat window, input handler, message bubbles) are built as reusable blocks.
* **Dialogflow intents** are grouped by domain (Accounts, Loans, Support).
* **Cloud Functions** are structured per use-case (e.g., balance lookup, KYC verification).
* **n8n workflows** are independent, event-triggered services with minimal shared state.

This structure allows new functionality to be added without rewriting existing code, fostering long-term scalability and clean maintainability.

#### **6.4 Code Management and CI/CD**

All source code and configurations will be managed in a **Git repository** hosted on GitHub. Branching strategies such as main, dev, and feature/\* will separate stable, experimental, and in-progress work. Each commit will be annotated with descriptive messages reflecting the work completed or issues resolved.

Code reviews (self or peer) will be conducted before merging major features into the main branch. Pull requests will include testing notes and relevant documentation updates.

Continuous Integration/Deployment (CI/CD) may be introduced using GitHub Actions to automate:

* Code linting and style enforcement
* Firebase deployment with preview links
* Dialogflow agent exports for backup/versioning
* Post-deployment test runs

While not mandatory for the MVP, this CI/CD framework can be scaled into a production pipeline.

#### 

#### **6.5 Development and Testing Environments**

Local development will be supported through:

* **Live-reload server** for frontend changes
* **Firebase Emulator Suite** for Firestore, Authentication, and Functions
* **n8n local server** for automation workflows
* **Mock API endpoints** for integration testing

Test cases and manual testing scripts will be defined for each module, with outcomes tracked in a test results document. Development logs and error reports will be tagged with timestamps, module origin, and resolution notes.

Environment variables and configuration files will manage development credentials and prevent hard coded secrets in the repository. .env files will be excluded from version control and documented for reproducible setup.

### **7. Testing Strategy**

The testing strategy for SNEA is designed to ensure that each component—whether backend, frontend, or conversational logic—functions as intended and delivers a reliable user experience. Even though SNEA is a demonstration project, the testing approach reflects real-world software development practices and includes both automated and manual validation methods.

#### **7.1 Testing Objectives**

The primary objectives of the testing phase are to:

* Validate correctness of chatbot responses across different conversation flows.
* Ensure secure and predictable behavior of backend services.
* Detect UI issues across devices and screen sizes.
* Identify and mitigate regressions as features evolve.
* Simulate edge cases such as missing parameters, incorrect input formats, or interrupted sessions.

#### **7.2 Types of Testing**

**Unit Testing** All backend cloud functions will include unit tests using Jest or a similar framework. These tests will cover:

* Input validation
* Data fetching and transformation
* Intent fulfillment logic
* Response formatting

**Integration Testing** Integration testing will verify the end-to-end flow from user input to system response. These tests ensure proper communication between:

* Dialogflow and Firebase Functions
* Firebase Functions and Firestore
* n8n Webhooks and data pipelines

**Manual Testing of Conversational Flows** Defined conversation flows for each intent (e.g., “check account balance,” “loan eligibility”) will be tested using a standard set of test cases. These test cases will verify:

* Accurate intent recognition
* Correct entity extraction
* Valid follow-up prompts
* Meaningful fallback handling
* User experience continuity

**UI and Compatibility Testing** The chatbot interface will be tested across:

* Multiple screen sizes (mobile, tablet, desktop)
* Modern browsers (Chrome, Firefox, Safari, Edge)
* Accessibility tools (screen readers)

**Security Testing** Security validations will include:

* Unauthorized access attempts
* Token tampering
* Session hijacking scenarios
* CSRF and XSS simulations

**User Acceptance Testing (UAT)** Before final deployment, the system will undergo a round of user acceptance testing with representative queries and simulated user flows. Feedback will be gathered to verify the usability and reliability of the assistant under real conditions.

#### **7.3 Issue Tracking**

All discovered issues will be recorded in the GitHub repository under the Issues tab. Each issue will be tagged with its type (bug, enhancement, question), severity, and status. Fixes will be tied to specific commits for traceability.

### **8. Algorithmic Approaches**

SNEA’s intelligence layer is built on a set of natural language processing (NLP) and dialog management algorithms, primarily implemented via Dialogflow ES and extended using custom backend logic.

#### **8.1 Natural Language Processing Techniques**

Dialogflow ES utilizes machine learning to detect user intents from natural language input. Intent classification is based on training phrases supplied during agent creation, enabling the model to generalize beyond exact matches. Behind the scenes, Dialogflow uses Google's BERT-based models and sequence classification techniques to improve accuracy.

The system also incorporates synonym matching, fuzzy logic, and phrase weighting to identify user intentions even when phrased differently. Entity recognition leverages built-in models for numbers, currencies, dates, and also supports custom-trained entities for domain-specific terms like “savings account” or “personal loan.”

#### **8.2 Context Management Algorithms**

Context in Dialogflow acts like a scoped memory store, enabling multi-turn conversations. The system uses a state machine model to manage conversational flow, with entry, exit, and lifespan control for each context.

Cloud Functions use this context to determine how to process the next input. Lifespan values dictate whether the conversation is in the middle of a transaction or should reset.

#### **8.3 Entity Extraction and Parameter Validation**

Entities are automatically extracted by Dialogflow and passed into the webhook request. The backend then applies additional validation logic to ensure:

* Required entities are present
* Data formats match expectations (e.g., numbers, emails)
* Semantic checks are satisfied (e.g., loan amount below threshold)

If validation fails, the conversation is redirected to a clarification or fallback intent.

#### **8.4 Response Selection and Generation**

Responses are selected based on:

* Matched intent
* Confidence score
* Active context and extracted parameters

Dialogflow supports multiple variations of static responses, which are randomly selected for naturalness. Dynamic responses from the webhook use templates populated with retrieved data, such as:

“Your simulated balance is ₹12,000 in your savings account.”

### **9. Technical Architecture Details**

#### **9.1 Component Diagram**

The architecture is designed as a modular, event-driven system composed of loosely coupled services:

[Frontend UI] <--> [Firebase SDK] <--> [Firestore + Auth + Analytics]

|

v

[Cloud Functions Webhook]

|

v

[Dialogflow NLP Engine]

|

v

[n8n Workflows]

Each component communicates via secure HTTPs APIs or real-time connections, minimizing dependencies and enabling independent testing.

#### **9.2 Data Flow Diagram**

1. User enters message in UI.
2. Message sent to Dialogflow via REST or embedded client.
3. Dialogflow classifies intent and extracts parameters.
4. If webhook is defined, Dialogflow triggers Cloud Function with payload.
5. Cloud Function processes request, fetches data (simulated), and returns response.
6. Response displayed in chat window.
7. Simultaneously, analytics and logs are written to Firestore.

#### **9.3 State Diagrams**

Example: **Loan Application Flow**

* Start → Ask for loan type → Ask for amount → Confirm eligibility → Provide simulated result → Offer options → End

States and transitions are defined in Dialogflow using input/output contexts with clearly scoped lifespan.

#### **9.4 Database Schema (Firestore)**

/users/{userId}

- email

- created\_at

- preferences

- auth\_provider

/conversations/{sessionId}

- user\_id

- start\_time

- end\_time

- intents\_triggered[]

- messages[]

- feedback

### **10. API Specifications**

#### **10.1 REST Endpoints**

| **Endpoint** | **Method** | **Purpose** |
| --- | --- | --- |
| /balance | POST | Return simulated balance based on account type |
| /transactions | GET | Fetch recent simulated transactions |
| /faq | GET | Return common banking questions and answers |
| /submit-feedback | POST | Store user feedback |

#### **10.2 Authentication Flow**

* Firebase Auth issues JWT after login.
* JWT sent as bearer token in API headers.
* Backend verifies token before serving request.

#### **10.3 Rate Limiting and Error Handling**

* API requests are rate-limited per user/session/IP.
* Error responses follow structured format:

json

CopyEdit

{

"error": {

"code": 403,

"message": "Unauthorized access"

}

}

#### **10.4 Webhook Payload Format (Dialogflow to Firebase)**

json

CopyEdit

{

"session": "projects/snea/agent/sessions/abc-123",

"queryResult": {

"intent": { "displayName": "Check Balance" },

"parameters": { "account\_type": "savings" }

}

}

### **11. Reusability Framework**

The codebase is designed to support future reuse and modular extensions.

* **Reusable Dialogflow Modules**: Intents grouped into domains with shared follow-up prompts and templates.
* **Shared UI Components**: Chat bubble, input bar, loading indicators, error messages.
* **Configurable Theme System**: CSS variables allow easy rebranding.
* **Abstracted Backend Services**: Firestore operations and API logic centralized into utility files.
* **Templated Conversations**: JSON-based conversation flow definitions for easy portability.

### **12. Integration Patterns**

SNEA employs modern integration patterns that support flexibility, scalability, and minimal coupling.

* **Webhook-Based Communication**: Dialogflow triggers Firebase functions on matched intents.
* **Event-Driven Workflows**: n8n handles asynchronous jobs like report generation or feedback alerts.
* **Protocol Abstraction**: All services communicate over REST/HTTPS, avoiding platform lock-in.
* **Secure Token Exchange**: Authentication tokens are validated on every request.
* **Extensible Design**: Future integrations with CRM, email marketing tools, or core banking APIs are achievable via webhook listeners or service adapters.

### **13. Code Structure and Standards**

The codebase follows clear, conventional naming and layout to improve readability and maintainability.

#### **13.1 Folder Structure**

pgsql

CopyEdit

/functions

└── index.js

└── services/

└── validators/

/public

└── index.html

└── styles.css

└── app.js

/src

└── components/

└── utils/

└── config/

#### **13.2 Naming Conventions**

* Functions and variables use camelCase.
* Components and classes use PascalCase.
* Files and folders use kebab-case.

#### **13.3 Documentation Practices**

* Every exported function includes JSDoc-style comments.
* Each module contains a README.md explaining purpose, setup, and usage.
* Dialogflow intent documentation is auto-exported via CLI for version control.

#### **13.4 Testing and Coverage**

* Unit tests for backend logic written in Jest.
* Manual test cases documented for conversational paths.
* Code coverage tracked using testing libraries (optional).
* Firebase emulator enables full-stack local integration testing.

### **14. Appendix**

The appendix provides definitions, references, and supporting material relevant to the SNEA project. It serves as a reference section for readers unfamiliar with some of the terminology or tools used throughout this document and supports traceability, clarity, and continuity of knowledge.

#### **14.1 Glossary of Terms**

| **Term** | **Definition** |
| --- | --- |
| **SNEA** | *Simple Network Engine Assistant* – The project name for the banking-focused conversational chatbot. |
| **NLP** | *Natural Language Processing* – A field of AI that enables computers to understand, interpret, and respond to human language. |
| **Intent** | A predefined purpose or action the user wants to perform (e.g., "Check Account Balance"). |
| **Entity** | Structured data extracted from user input that helps refine the intent (e.g., "₹5,000", "Savings Account"). |
| **Context** | Temporary memory used in a conversation to retain the state across multiple interactions. |
| **Webhook** | An HTTP callback used to pass data between Dialogflow and backend services such as Firebase Functions. |
| **Firebase** | A Backend-as-a-Service platform by Google offering authentication, hosting, Firestore, analytics, and cloud functions. |
| **Dialogflow ES** | A Google-powered NLU engine that maps user input to intents and supports context and entity handling. |
| **n8n** | A source-available automation platform that enables workflow creation through node-based visual programming. |
| **Firestore** | A scalable NoSQL database provided by Firebase for real-time data storage and syncing. |
| **JWT** | *JSON Web Token* – A secure token format used for authentication and information exchange. |
| **CI/CD** | *Continuous Integration/Continuous Deployment* – Practices for automating code integration and delivery pipelines. |
| **MVP** | *Minimum Viable Product* – A version of the product with just enough features to be functional and demonstrable. |

#### **14.2 References**

* **Firebase Documentation** https://firebase.google.com/docs
* **Dialogflow ES Documentation** https://cloud.google.com/dialogflow/es/docs
* **n8n Documentation** https://docs.n8n.io
* **Google Cloud Functions** https://cloud.google.com/functions
* **Jest Testing Framework** https://jestjs.io/docs/getting-started
* **Draw.io (Diagrams)** https://app.diagrams.net

#### **14.3 Supporting Materials**

The following documents and artifacts will be maintained alongside the BRD:

* Technical Design Document (TDD)
* Dialogflow Agent Export (.zip)
* API Reference Manual
* Test Case Spreadsheet
* Architecture Diagrams (Component, Data Flow, State)
* Development Environment Setup Guide
* README.md for Project Repository

#### **14.4 Document History**

| **Version** | **Date** | **Author** | **Changes** |
| --- | --- | --- | --- |
| 1.0 | April 14, 2025 | Sneha AJITHKUMAR | Initial version drafted and approved |
| 1.1 | [TBD] | Sneha AJITHKUMAR | Added Algorithmic Approaches, API Specs, Architecture |
| 1.2 | [TBD] | Sneha AJITHKUMAR | Final QA and formatting edits |