STAGE 2: REQUIREMENTS ANALYSIS AND SYSTEM DESIGN

BU Chatbot – A Personalized Chatbot System for Bugema University

2.1 Introduction

This document presents a comprehensive requirements analysis and system design for the BU Chatbot, a personalized conversational AI system developed to enhance information accessibility and communication at Bugema University. The system aims to provide 24/7 automated assistance to students, staff, and visitors by delivering accurate, timely responses to inquiries about admissions, academic programs, fees, schedules, campus services, and university policies.

The BU Chatbot addresses the challenge of information overload and limited accessibility of university resources, particularly outside office hours. By implementing natural language processing and a robust knowledge management system, the chatbot serves as an intelligent virtual assistant that reduces the administrative burden on staff while improving the overall user experience for the university community.

This stage encompasses the complete requirements gathering, analysis, and system design phases, including functional and non-functional requirements, system architecture, database design, and interface specifications.

2.2 User Requirements

2.2.1 Students

- Access instant information about course registration, academic calendar, and examination schedules
- Inquire about tuition fees, payment plans, and financial aid opportunities
- Get guidance on admission requirements and application procedures
- Retrieve information about campus facilities, accommodation, and student services
- Receive personalized recommendations based on their program and academic level
- View conversation history and previous interactions with the chatbot
- Provide feedback on chatbot responses to improve service quality

2.2.2 Lecturers and Staff

- Obtain quick answers about academic policies, grading procedures, and administrative guidelines
- Access information about faculty resources, meeting schedules, and departmental contacts
- Retrieve details about research opportunities and professional development programs
- Get support for common technical and administrative queries
- Access relevant university documentation and policy manuals

2.2.3 Visitors and Prospective Students

- Learn about available academic programs and their requirements
- Understand the admission process and application deadlines
- Explore campus facilities, accommodation options, and student life
- Access general information about Bugema University without registration
- Find contact information for specific departments and offices

2.2.4 Administrators

- Manage and update the chatbot knowledge base with current information
- Monitor system performance and user interaction metrics
- View comprehensive analytics on query types, user satisfaction, and system usage
- Manage user accounts and access permissions
- Configure chatbot behavior, response templates, and escalation rules
- Export chat logs and reports for analysis and compliance purposes

2.3 System Requirements

2.3.1 Functional Requirements

FR1: User Authentication and Authorization

- FR1.1: The system shall allow users to register with email and password
- FR1.2: The system shall authenticate users using secure login credentials
- FR1.3: The system shall support role-based access control (Student, Staff, Admin, Visitor)
- FR1.4: The system shall maintain user sessions securely using JWT tokens
- FR1.5: The system shall allow password reset functionality via email

FR2: Query Processing and Response Generation

- FR2.1: The system shall accept natural language queries from users
- FR2.2: The system shall process queries using NLP to extract intent and keywords
- FR2.3: The system shall search the knowledge base for relevant information
- FR2.4: The system shall generate contextually appropriate responses
- FR2.5: The system shall provide response confidence scores
- FR2.6: The system shall handle multiple concurrent user queries

FR3: Knowledge Base Management

- FR3.1: Administrators shall be able to add, update, and delete knowledge entries
- FR3.2: The system shall organize knowledge by categories (Admissions, Academics, Fees, Campus Life, etc.)
- FR3.3: The system shall support tagging of knowledge entries for improved search

- FR3.4: The system shall track usage frequency of knowledge entries
- FR3.5: The system shall version control knowledge base updates

FR4: Conversation Management

- FR4.1: The system shall store complete conversation histories for authenticated users
- FR4.2: The system shall allow users to view past conversations
- FR4.3: The system shall maintain conversation context within a session
- FR4.4: The system shall support conversation export in common formats (PDF, TXT)

FR5: Feedback and Rating System

- FR5.1: Users shall be able to rate chatbot responses (thumbs up/down or 1-5 stars)
- FR5.2: Users shall be able to provide text comments on responses
- FR5.3: The system shall collect and store feedback for analysis
- FR5.4: The system shall flag low-rated responses for review

FR6: Analytics and Reporting

- FR6.1: The system shall track total queries, unique users, and response times
- FR6.2: The system shall calculate user satisfaction rates based on feedback
- FR6.3: The system shall identify top query categories and trending topics
- FR6.4: The system shall generate daily, weekly, and monthly reports
- FR6.5: Administrators shall be able to export analytics data

FR7: Personalization

- FR7.1: The system shall provide personalized greetings using user information
- FR7.2: The system shall recommend relevant information based on user profile (program, level)
- FR7.3: The system shall remember user preferences and frequently asked topics
- FR7.4: The system shall adjust response complexity based on user type

FR8: Escalation and Human Support

- FR8.1: The system shall provide options to contact human support for unresolved queries
- FR8.2: The system shall collect contact requests with query context
- FR8.3: The system shall notify administrators of escalation requests

2.3.2 Non-Functional Requirements

NFR1: Performance

- NFR1.1: The system shall respond to queries within 500ms under normal load
- NFR1.2: The system shall support at least 100 concurrent users
- NFR1.3: The system shall achieve 99% uptime availability

- NFR1.4: Database queries shall execute in less than 200ms

NFR2: Security

- NFR2.1: All data transmission shall use HTTPS encryption
- NFR2.2: User passwords shall be hashed using bcrypt with salt
- NFR2.3: The system shall implement JWT-based authentication
- NFR2.4: The system shall prevent SQL injection and XSS attacks
- NFR2.5: The system shall implement rate limiting to prevent abuse

NFR3: Usability

- NFR3.1: The interface shall be intuitive and require no training
- NFR3.2: The system shall provide clear error messages and guidance
- NFR3.3: The chatbot shall maintain a friendly, professional tone
- NFR3.4: The interface shall be accessible to users with disabilities (WCAG 2.1 Level AA)

NFR4: Reliability

- NFR4.1: The system shall handle errors gracefully without crashing
- NFR4.2: The system shall log all errors for debugging purposes
- NFR4.3: The system shall maintain data integrity during failures
- NFR4.4: The system shall implement automatic backup of critical data

NFR5: Scalability

- NFR5.1: The system architecture shall support horizontal scaling
- NFR5.2: The database shall handle growing data volume efficiently
- NFR5.3: The knowledge base shall accommodate expanding content

NFR6: Maintainability

- NFR6.1: The code shall follow consistent style guidelines and best practices
- NFR6.2: The system shall be modular to facilitate updates
- NFR6.3: The system shall include comprehensive documentation
- NFR6.4: The system shall support version control and rollback capabilities

NFR7: Compatibility

- NFR7.1: The frontend shall be responsive and work on desktop, tablet, and mobile devices
- NFR7.2: The system shall support modern browsers (Chrome, Firefox, Safari, Edge)
- NFR7.3: The system shall maintain backward compatibility with previous API versions

NFR8: Accuracy

- NFR8.1: The system shall provide correct information with at least 90% accuracy
- NFR8.2: The system shall clearly indicate when information may be outdated
- NFR8.3: The system shall avoid providing speculative or unverified information

2.4 System Design

2.4.1 Architecture Design

The BU Chatbot employs a three-tier architecture consisting of:

1. Presentation Layer (Client-Side)

- Built with React.js for dynamic, responsive user interfaces
- Implements component-based architecture for reusability
- Uses React Hooks for state management (useState, useEffect, useContext)
- Integrates real-time updates for chat functionality
- Deployed on Vercel for optimal performance and CDN distribution

2. Application Layer (Server-Side)

- Developed using Express.js on Node.js runtime
- RESTful API design for standardized communication
- Implements middleware for authentication, validation, and error handling
- Contains business logic for NLP processing and response generation
- Deployed on Render with environment-based configuration

3. Data Layer

- MongoDB Atlas cloud database for scalable data storage
- NoSQL document model for flexible schema design
- Implements indexing for optimized query performance
- Provides automated backups and high availability

Key Architectural Patterns:

- **MVC Pattern**: Separation of concerns between models, views, and controllers
- **Middleware Pattern**: Chainable request processing pipeline
- **Repository Pattern**: Abstraction layer for data access operations
- **Strategy Pattern**: Pluggable NLP processing algorithms

2.4.2 System Modules

Module 1: Authentication Module

- User registration and login
- Password encryption and validation

- JWT token generation and verification
- Session management and logout

Module 2: Query Processing Module

- Natural language understanding
- Intent classification
- Entity extraction
- Context management

Module 3: Knowledge Base Module

- CRUD operations for knowledge entries
- Category and tag management
- Search and retrieval algorithms
- Relevance ranking

Module 4: Response Generation Module

- Template-based responses
- Dynamic content insertion
- Response formatting and styling
- Confidence scoring

Module 5: Conversation Management Module

- Chat history storage and retrieval
- Multi-turn conversation handling
- Context preservation
- Export functionality

Module 6: Analytics Module

- Query logging and metrics collection
- User behavior tracking
- Performance monitoring
- Report generation

Module 7: Admin Dashboard Module

- Knowledge base management interface
- User management
- Analytics visualization
- System configuration

2.5 Database Design

2.5.1 Database Schema Overview

The BU Chatbot uses MongoDB, a NoSQL document database, chosen for its flexibility, scalability, and ease of integration with Node.js. The database consists of the following collections:

2.5.2 Collection Structures

Users Collection

```
{
  _id: ObjectId,
  userId: String (unique, indexed),
  email: String (unique, indexed),
  password: String (hashed),
  fullName: String,
  userType: String (enum: ['student', 'staff', 'admin', 'visitor']),
  studentId: String (optional),
  department: String (optional),
  registrationDate: Date,
  lastLogin: Date,
  isActive: Boolean,
  preferences: {
    language: String,
    notificationsEnabled: Boolean
  },
  createdAt: Date,
  updatedAt: Date
}
```

Conversations Collection

```
{
  _id: ObjectId,
  conversationId: String (unique, indexed),
  userId: String (indexed, foreign key),
  startTime: Date,
  lastUpdated: Date,
  status: String (enum: ['active', 'closed', 'archived']),
  messageCount: Number,
  metadata: {
    deviceType: String,
    browserInfo: String,
    ipAddress: String
  },
  createdAt: Date,
  updatedAt: Date
}
```

Messages Collection

```
{
  _id: ObjectId,
  messageId: String (unique, indexed),
  conversationId: String (indexed, foreign key),
  content: String,
  sender: String (enum: ['user', 'bot']),
  sentAt: Date,
  isBot: Boolean,
  intent: String,
  confidence: Number,
  entities: Array,
  responseSource: String,
  processingTime: Number,
  createdAt: Date
}
```

KnowledgeBase Collection

```
{
  _id: ObjectId,
  knowledgeId: String (unique, indexed),
  category: String (indexed),
  subcategory: String,
  question: String,
  answer: String,
  alternativeQuestions: Array,
  tags: Array (indexed),
  priority: Number,
  usageCount: Number,
  lastUsed: Date,
  isActive: Boolean,
  createdBy: String (admin userId),
  createdAt: Date,
  updatedAt: Date,
  metadata: {
    source: String,
    lastReviewedBy: String,
    lastReviewedAt: Date
  }
}
```

Feedback Collection

```
{
  _id: ObjectId,
  feedbackId: String (unique),
  messageId: String (indexed, foreign key),
  userId: String (indexed),
  conversationId: String,
  rating: Number (1-5),
  comment: String,
  feedbackType: String (enum: ['helpful', 'unhelpful', 'incorrect',
'other']),
  submittedAt: Date,
  reviewedBy: String (optional),
  reviewStatus: String (enum: ['pending', 'reviewed', 'actioned']),
  createdAt: Date
}
```

Analytics Collection

```
{
  _id: ObjectId,
  analyticsId: String (unique),
  recordDate: Date (indexed),
  period: String (enum: ['daily', 'weekly', 'monthly']),
  metrics: {
    totalQueries: Number,
    uniqueUsers: Number,
    newUsers: Number,
    avgResponseTime: Number,
    avgConfidence: Number,
    satisfactionRate: Number,
    escalationRate: Number
  },
  topCategories: Array,
  topQueries: Array,
  userDistribution: Object,
  performanceMetrics: {
    serverUptime: Number,
    errorRate: Number,
    apiLatency: Number
  },
  createdAt: Date
}
```

IntentCategories Collection

```
{
  _id: ObjectId,
  categoryId: String (unique, indexed),
  categoryName: String,
  description: String,
  keywords: Array,
  parentCategory: String (optional),
  priority: Number,
  isActive: Boolean,
  responseTemplate: String,
  escalationThreshold: Number,
  createdAt: Date,
  updatedAt: Date
}
```

AdminLogs Collection

```
{
  _id: ObjectId,
  logId: String (unique),
  adminId: String (indexed),
  action: String,
  targetEntity: String,
  targetId: String,
  changes: Object,
  ipAddress: String,
  timestamp: Date,
  status: String (enum: ['success', 'failed']),
  createdAt: Date
}
```

2.5.3 Database Indexes

To optimize query performance, the following indexes are implemented:

- Users: `userId`, `email`, `userType`
- Conversations: `conversationId`, `userId`, `status`
- Messages: `messageId`, `conversationId`, `sentAt`
- KnowledgeBase: `knowledgeId`, `category`, `tags`
- Feedback: `messageId`, `userId`, `rating`
- Analytics: `recordDate`, `period`

2.5.4 Data Relationships

- One User can have Many Conversations (1:N)
- One Conversation can have Many Messages (1:N)
- One Message can have One Feedback (1:1)
- One KnowledgeBase entry can generate Many Messages (1:N)
- One Admin can manage Many KnowledgeBase entries (1:N)

2.6 Interface Design

2.6.1 User Interface Principles

The BU Chatbot interface adheres to the following design principles:

Simplicity: Clean, uncluttered layout focusing on the conversation **Consistency**: Uniform design patterns across all pages **Accessibility**: WCAG 2.1 compliant with keyboard navigation and screen reader support **Responsiveness**: Adaptive layouts for desktop, tablet, and mobile devices **Feedback**: Clear visual indicators for system status and user actions

2.6.2 Key Interface Components

1. Landing Page

- University branding (logo, colors)
- Brief description of chatbot capabilities
- Quick access buttons for common queries
- Login/Register options
- Guest access option for visitors

2. Chat Interface

- Header: University logo, user profile, logout button
- Main chat area: Message bubbles (user in blue, bot in gray)
- Input field: Text box with send button
- Quick reply suggestions below input
- Sidebar (collapsible): Conversation history, settings, help

3. Message Components

- Timestamp for each message
- Bot avatar icon for system messages
- User avatar/initial for user messages
- Typing indicator when bot is processing
- Feedback buttons (thumbs up/down) for bot messages
- Copy and share options for useful responses

4. Admin Dashboard

- Navigation menu: Dashboard, Knowledge Base, Analytics, Users, Settings
- Overview cards: Total queries, active users, satisfaction rate, recent activity
- Knowledge base management: Add/edit/delete entries, search, filter by category
- Analytics charts: Query trends, user growth, category distribution, performance metrics
- User management table: View users, edit roles, activate/deactivate accounts

5. Mobile Interface

- Bottom navigation bar for key features
- Swipe gestures for conversation history
- Floating action button for new conversation
- Optimized touch targets (minimum 44x44px)

2.6.3 Wireframes Description

Main Chat Screen Wireframe:

```
+-------------------------------------------+
|  [≡]  BU Chatbot        [@] [Settings]  |
+-------------------------------------------+
|                                           |
|  [Bot Avatar] Hello! How can I help     |
|              you today?                 |
|              [10:30 AM]                 |
|                                           |
|                  What are the tuition   |
|                  fees for BSc CS?       |
|                  [10:31 AM] [You]       |
|                                           |
|  [Bot Avatar] The tuition fees for      |
|              Bachelor of Science in     |
|              Computer Science are...    |
|              [10:31 AM]                  |
|              [👍] [👎]                   |
|                                          |
+-----------------------------------------+
| [Quick: Admissions] [Fees] [Academic]   |
+-----------------------------------------+
| Type your message...           [Send]  |
+-----------------------------------------+
```

Admin Dashboard Wireframe:

```
+-----------------------------------------+
| BU Chatbot Admin      [Profile] [Logout] |
+-----------------------------------------+
| [Dashboard] [Knowledge] [Analytics] [...] |
+-----------------------------------------+
|  +----------+  +-----------+  +----------+ |
|  | Queries  |  | Users     |  | Satisfied| |
|  | 1,245    |  | 342       |  | 87%      | |
|  +----------+  +-----------+  +----------+ |
|                                          |
|  Recent Activity                         |
|  +-----------------------------------+   |
|  | [📊] Chart: Queries over time     |   |
|  |                                    |   |
|  +-----------------------------------+   |
|                                          |
|  Top Categories                          |
|  1. Admissions (35%)                     |
|  2. Fees (28%)                          |
|  3. Academic Programs (22%)             |
+-----------------------------------------+
```

2.6.4 Color Scheme and Branding

- **Primary Color**: #003366 (Bugema University Blue)
- **Secondary Color**: #FFD700 (Gold/Yellow for accents)
- **Success**: #28A745 (Green for positive actions)
- **Warning**: #FFC107 (Amber for notifications)
- **Error**: #DC3545 (Red for errors)
- **Background**: #F8F9FA (Light gray)
- **Text**: #212529 (Dark gray)
- **Bot Messages**: #E9ECEF (Light gray background)
- **User Messages**: #007BFF (Blue background)

2.6.5 Typography

- **Headings**: Inter, sans-serif (Bold, 24-32px)
- **Body Text**: Inter, sans-serif (Regular, 16px)

- **Chat Messages**: Inter, sans-serif (Regular, 14-16px)
- **Buttons**: Inter, sans-serif (Medium, 14px)

2.7 Technology Stack

2.7.1 Frontend Technologies

- **React.js** (v18.x): UI library for building interactive components
- **React Router**: Client-side routing
- **Axios**: HTTP client for API requests
- **CSS3/Tailwind CSS**: Styling and responsive design
- **React Context API**: State management
- **Socket.io-client** (optional): Real-time communication

2.7.2 Backend Technologies

- **Node.js** (v18.x): JavaScript runtime
- **Express.js** (v4.x): Web application framework
- **Mongoose**: MongoDB object modeling
- **JWT (jsonwebtoken)**: Authentication tokens
- **Bcrypt**: Password hashing
- **Cors**: Cross-origin resource sharing
- **Dotenv**: Environment variable management
- **Express-validator**: Input validation
- **Helmet**: Security headers
- **Morgan**: HTTP request logger

2.7.3 Database

- **MongoDB Atlas**: Cloud-hosted NoSQL database
- **Mongoose**: ODM for MongoDB

2.7.4 Deployment Platforms

- **Vercel**: Frontend hosting with automatic deployments
- **Render**: Backend hosting with continuous deployment
- **MongoDB Atlas**: Database hosting

2.7.5 Development Tools

- **Git**: Version control
- **GitHub**: Repository hosting and collaboration
- **VS Code**: Code editor
- **Postman**: API testing
- **ESLint**: Code linting

- **Prettier**: Code formatting

2.7.6 Optional Enhancement Technologies

- **Natural.js / Compromise.js**: NLP processing
- **TensorFlow.js**: Machine learning (future enhancement)
- **Redis**: Caching layer (future scalability)
- **Socket.io**: Real-time features (typing indicators, live updates)

2.8 System Workflow

2.8.1 User Authentication Flow

1. User accesses the BU Chatbot landing page
2. User selects "Get Started"
3. User redirected to a temporary chat
4. User selects “Login” or “Signup”
5. For Signup: User provides email, password, name
6. System validates input and checks for existing email
7. System hashes password and stores user data in database
8. System generates JWT token and sends to client
9. For login: User provides credentials
10. System verifies credentials against database
11. System generates and returns JWT token
12. Client stores token and redirects to chat interface

2.8.2 Query Processing Flow

1. User types a question in the chat interface
2. Frontend sends query to backend API with user token
3. Backend verifies authentication and extracts user info
4. NLP module processes query to extract intent and keywords
5. System searches knowledge base for matching entries
6. System ranks results by relevance and confidence
7. Response generation module creates appropriate answer
8. System stores message in conversation history
9. System logs query for analytics
10. Backend returns response to frontend
11. Frontend displays response in chat interface
12. User can provide feedback on response

2.8.3 Admin Knowledge Management Flow

1. Admin logs into dashboard
2. Admin navigates to Knowledge Base section

3. Admin clicks "Add New Entry"
4. Admin fills in category, question and answer
5. System validates input data
6. System stores entry in KnowledgeBase collection
7. System logs admin action
8. System displays success confirmation
9. Knowledge base is immediately available for queries

2.9 Security Measures

2.9.1 Authentication Security

- Password strength requirements (minimum 8 characters, mix of letters, numbers, symbols)
- Bcrypt hashing with salt rounds (10+)
- JWT tokens with expiration (24 hours)
- Secure token storage (httpOnly cookies or localStorage with encryption)
- Password reset via email with time-limited tokens

2.9.2 Data Security

- HTTPS/TLS encryption for all data transmission
- MongoDB encryption at rest
- Environment variables for sensitive configuration
- Input sanitization to prevent NoSQL injection
- XSS protection through content security policy
- CSRF protection for state-changing operations

2.9.3 Access Control

- Role-based access control (RBAC)
- API endpoint protection with middleware
- Admin-only routes for sensitive operations
- User-specific data access restrictions
- Rate limiting to prevent abuse (e.g., 100 requests/15 minutes)

2.9.4 Monitoring and Logging

- Error logging with Winston or Morgan
- Admin action audit trail
- Failed login attempt tracking
- Suspicious activity detection
- Regular security audits

2.10 Performance Optimization Strategies

2.10.1 Frontend Optimization

- Code splitting and lazy loading
- Image optimization and compression
- Caching strategies for static assets
- Minification of CSS and JavaScript
- CDN usage via Vercel

2.10.2 Backend Optimization

- Database query optimization with indexes
- Response caching for frequent queries
- Connection pooling for database
- Compression of API responses (gzip)
- Asynchronous processing for heavy operations

2.10.3 Database Optimization

- Proper indexing on frequently queried fields
- Aggregation pipelines for complex queries
- Regular database maintenance and cleanup
- Archiving of old conversations
- Query result caching

2.11 Limitations and Mitigations

2.11.1 Limitations

- **Limited Context Understanding**: Basic NLP may not grasp complex, multi-part questions
- **Knowledge Base Dependency**: Accuracy relies on completeness and currency of knowledge base
- **Language Limitation**: Currently supports English only
- **No Voice Interface**: Text-only interaction
- **Limited Integration**: No direct integration with university systems (student portals, LMS)

2.11.2 Mitigation Strategies

- Implement clarifying questions for ambiguous queries
- Regular knowledge base updates and reviews
- Plan for future multi-language support
- Provide escalation to human support for complex issues
- Design API for future system integrations
- Continuous improvement based on user feedback and analytics

2.13 Conclusion

Stage 2 has established a comprehensive foundation for the BU Chatbot system through detailed requirements analysis and system design. The functional and non-functional requirements clearly define what the system should accomplish and how it should perform. The three-tier architecture using React, Express.js, and MongoDB provides a scalable, maintainable solution suitable for Bugema University's needs.

The database design with eight collections ensures efficient data management, while the interface designs prioritize user experience and accessibility. Security measures and performance optimization strategies have been integrated into the design to ensure a robust, reliable system.