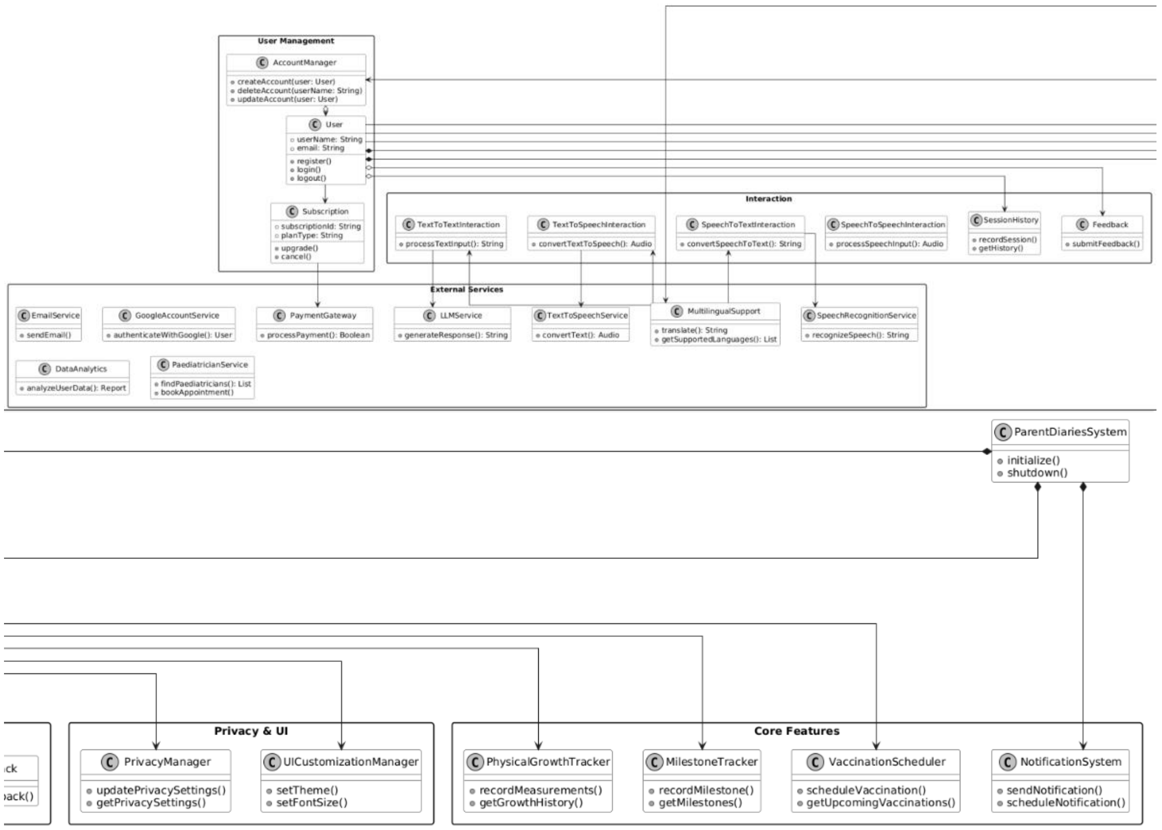
Product Design

Team 49
Wave Diaries
Shivam Gupta, Aditya Gaur, Raunak Seksaria , Manit Roy

Design Model

UML CLASS DIAGRAM (Split into two halves from left to right in two images)



Explanation of various classes

Parent Diaries System	**Class state:** • No explicit attributes but manages overall system configuration **Class behavior:** • **initialize():** Starts up the system and initializes all components • **shutdown():** Properly closes all connections and terminates the

	system
User	**Class state:** • **userName:** The unique identifier for a user • **email:** The user's email address for communication **Class behavior:** • **register():** Creates a new user account • **login():** Authenticates a user and grants access • **logout():** Ends a user session
Subscription	**Class state:** • **subscriptionId:** Unique identifier for a subscription • **planType:** The type of subscription plan • **startDate:** When the subscription begins • **expiryDate:** When the subscription ends **Class behavior:** • **upgrade():** Changes to a higher-tier subscription • **cancel():** Terminates a subscription
Vaccination Scheduler	**Class state:** • No explicit attributes but maintains vaccination records internally **Class behavior:** • **scheduleVaccination():** Creates a new vaccination appointment • **getUpcomingVaccinations():** Retrieves planned vaccinations
PhysicalGrowthTracker	**Class state:** • No explicit attributes but stores growth measurements internally **Class behavior:** • **recordMeasurements():** Stores new height/weight measurements • **getGrowthHistory():** Retrieves historical growth data
MilestoneTracker	**Class state:** • No explicit attributes but tracks developmental milestones internally **Class behavior:** • **recordMilestone():** Logs a new developmental achievement • **getMilestones():** Retrieves developmental milestone history

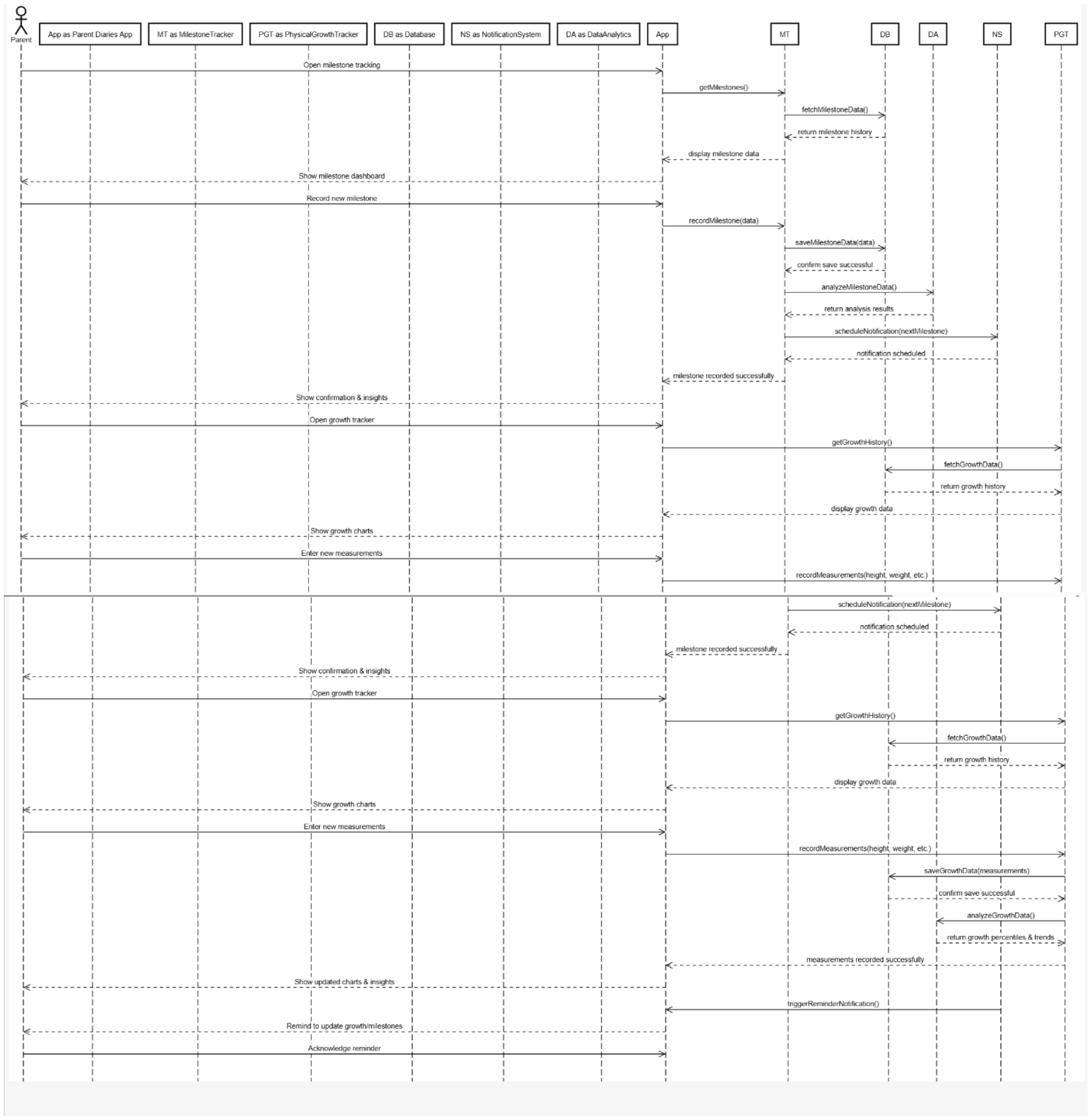
NotificationSystem	**Class state:** • No explicit attributes but manages notification queue internally **Class behavior:** • **sendNotification():** Delivers immediate notifications • **scheduleNotification():** Sets up future notifications
TextToTextInteraction	**Class state:** • No explicit attributes but maintains conversation context **Class behavior:** • **processTextInput():** Handles text input and returns text responses
TextToSpeechInteraction	**Class state:** • No explicit attributes but maintains voice settings **Class behavior:** • **convertTextToSpeech():** Transforms text into spoken audio
SpeechToTextInteraction	**Class state:** • No explicit attributes but manages speech recognition settings **Class behavior:** • **convertSpeechToText():** Transcribes spoken audio to text
SpeechToSpeechInteraction	**Class state:** • No explicit attributes but maintains voice interaction context **Class behavior:** • **processSpeechInput():** Processes spoken input and returns spoken output
SessionHistory	**Class state:** • No explicit attributes but stores interaction records **Class behavior:** • **recordSession():** Saves a new interaction session • **getHistory():** Retrieves past interaction sessions
Feedback	**Class state:** • No explicit attributes but stores user feedback data

	Class behavior: • **submitFeedback():** Collects and stores user opinions and ratings
PrivacyManager	**Class state:** • No explicit attributes but maintains privacy configuration **Class behavior:** • **updatePrivacySettings():** Changes privacy configurations • **getPrivacySettings():** Retrieves current privacy settings
EmailService	**Class state:** • No explicit attributes but maintains email service configuration **Class behavior:** • **sendEmail():** Delivers emails to specified recipients
GoogleAccountService	**Class state:** • No explicit attributes but maintains OAuth connections **Class behavior:** • **authenticateWithGoogle():** Handles Google sign-in authentication
TextToSpeechService	**Class state:** • No explicit attributes but maintains voice synthesis settings **Class behavior:** • **convertText():** Transforms text into spoken audio via external service
SpeechRecognitionService	**Class state:** • No explicit attributes but maintains speech recognition configurations **Class behavior:** • **recognizeSpeech():** Transcribes audio to text via external service
LLM Service	**Class state:** • No explicit attributes but maintains language model configurations

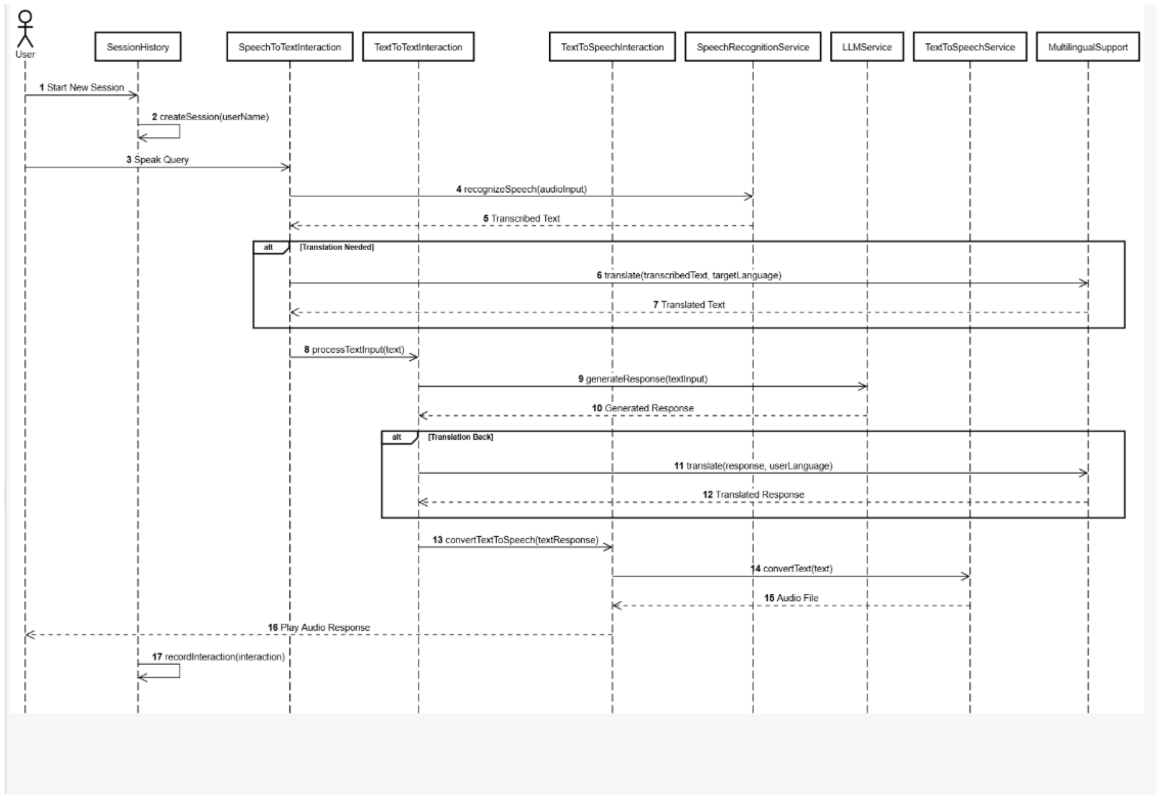
	Class behavior: • **generateResponse():** Creates AI-generated text responses
DataAnalytics	**Class state:** • No explicit attributes but maintains analytics datasets **Class behavior:** • **analyzeUserData():** Processes user data to generate insights and reports
PaediatricianService	**Class state:** • No explicit attributes but maintains doctor directory **Class behavior:** • **findPaediatricians():** Locates nearby pediatricians • **bookAppointment():** Schedules doctor visits
PaymentGateway	**Class state:** • No explicit attributes but maintains payment processor connections **Class behavior:** • **processPayment():** Handles financial transactions
MultilingualSupport	**Class state:** • No explicit attributes but maintains language configurations **Class behavior:** • **translate():** Converts text between languages • **getSupportedLanguages():** Lists available languages

Sequence Diagram(s)

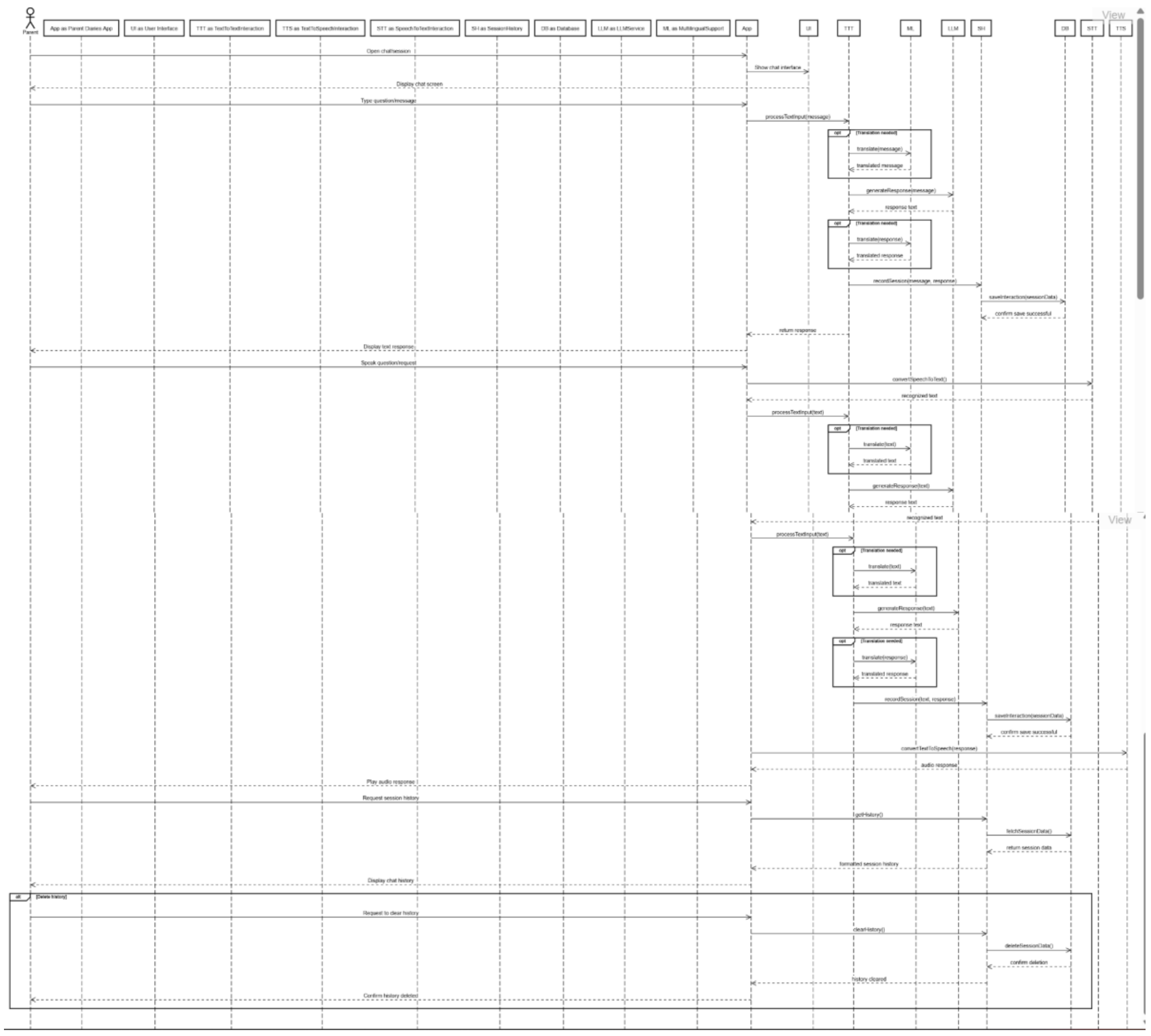
1) Milestone Tracking and Physical Growth Tracker



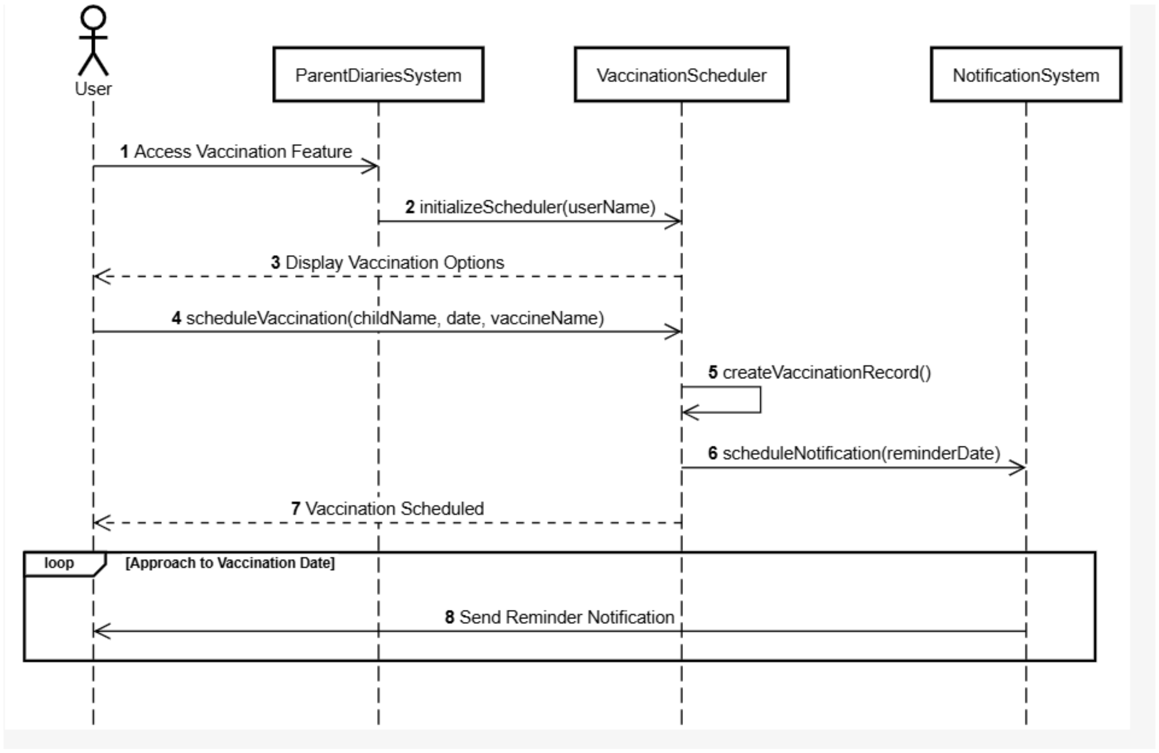
2) Speech Interaction Sequence Diagram



3) Session History Sequence Diagram



4) Vaccination Scheduling and Charting Sequence Diagram


User
ParentDiariesSystem
VaccinationScheduler
NotificationSystem
1 Access Vaccination Feature
2 initializeScheduler(userName)
3 Display Vaccination Options
4 scheduleVaccination(childName, date, vaccineName)
5 createVaccinationRecord()
6 scheduleNotification(reminderDate)
7 Vaccination Scheduled
loop
[Approach to Vaccination Date]
8 Send Reminder Notification

Design Rationale

Design Rationale for Parenting Application

1. User Registration System

Chosen Design
Multi-step registration process with email verification, separate User and AccountManager classes, and optional Google account integration.

Alternatives Considered

1. **Single-Step Registration**: Simple form with minimal information collection.
    a. Pros: Reduced friction, higher completion rates, faster onboarding
    b. Cons: Less initial user data, potential for low-quality accounts, limited personalization
2. **Mandatory Social Authentication**: Requiring registration through Google or other social platforms.
    a. Pros: Simplified authentication, reduced password management, verified identities
    b. Cons: Excludes users without social accounts, privacy concerns, dependency on third-party services
3. **Progressive Registration**: Basic registration first, with incremental profile completion.
    a. Pros: Lower initial barrier, better engagement metrics, adaptable to user interest
    b. Cons: Incomplete user profiles, complex state management, potential abandonment

Rationale for Final Decision

We implemented a multi-step registration with email verification because:

1. Parenting applications require trustworthy accounts due to sensitive child data
2. Email verification reduces spam accounts and enables critical communication
3. The separate AccountManager class provides specialized account lifecycle management
4. Google account integration offers convenience while maintaining direct registration options
5. This approach balances security concerns with reasonable user experience

2. Privacy and Security Architecture

Chosen Design
Dedicated PrivacyManager class with user-controlled settings, coupled with a comprehensive security approach including data encryption and access controls.

Alternatives Considered

1. **Privacy Settings in User Profile**: Embedding privacy controls in User Profile.
    a. Pros: Simplified data model, reduced complexity, direct user association
    b. Cons: Limited granularity, harder to audit, mixed responsibility

2. **Global Privacy Settings**: System-wide privacy settings applied to all users.
    a. Pros: Consistent implementation, easier maintenance, simplified interface
    b. Cons: Limited personalization, potential regulatory issues across regions, reduced user control
3. **Third-Party Privacy Management**: Integration with specialized privacy management services.
    a. Pros: Expert implementation, compliance guarantees, offloaded maintenance
    b. Cons: Increased costs, external dependencies, potential integration challenges

Rationale for Final Decision

We implemented a dedicated PrivacyManager class because:

1. Child-related data requires specialized, granular privacy controls
2. Separating privacy logic allows for focused security audits and compliance checks
3. Different regions have varying privacy regulations affecting parenting applications
4. Users need transparent control over sensitive information sharing
5. The dedicated class provides a foundation for future privacy enhancements and compliance features

The system also implements comprehensive security measures including:

- End-to-end encryption for all health-related data
- Role-based access controls for data sharing
- Regular security audits and penetration testing
- Secure data storage with minimal retention periods
- Transparent data usage policies with user consent management

3. Core Features Architecture

Chosen Design
Modular feature classes (VaccinationScheduler, PhysicalGrowthTracker, MilestoneTracker) connected to the central User entity with a shared NotificationSystem.

Alternatives Considered

1. **Feature-as-Service Approach**: Implementing each feature as an independent microservice.
    a. Pros: Maximum isolation, independent scaling, specialized teams per feature
    b. Cons: More complex integration, increased operational overhead, potential consistency issues
2. **Monolithic Feature Manager**: Single class managing all parenting tracking features.
    a. Pros: Simplified implementation, centralized business logic, easier data sharing
    b. Cons: Limited maintainability, difficult to extend, harder to test individual components
3. **Event-Based Feature Communication**: Features communicating through an event bus rather than direct relationships.
    a. Pros: Lower coupling, easier to add new features, more resilient architecture
    b. Cons: More complex to implement, potential for missed events, harder to debug

Rationale for Final Decision

We chose the modular feature class approach with direct user relationships because:

1. It balances modularity with implementation simplicity, appropriate for a mobile application
2. Each feature has distinct tracking needs but benefits from shared user context

3. The centralized NotificationSystem provides consistent alerting while maintaining feature independence
4. Direct relationships make data flow more transparent and easier to reason about
5. This approach supports incremental development of features without major architectural changes

4. AI Interaction Models

Chosen Design
Multi-modal interaction system with specialized classes for different communication patterns (TextToText, TextToSpeech, SpeechToText, SpeechToSpeech) supported by external AI services.

Alternatives Considered

1. **Single Interaction Controller**: Unified interaction class handling all input/output modalities.
   a. Pros: Simplified interface, consistent handling, easier integration
   b. Cons: Complex internal logic, difficult to extend, potential performance bottlenecks
2. **Client-Side Voice Processing**: Moving speech processing to the mobile device.
   a. Pros: Works offline, reduced server costs, faster response for simple queries
   b. Cons: Inconsistent quality across devices, larger app footprint, limited AI capabilities
3. **Third-Party Conversational Platform**: Using a complete solution like Dialogflow or Lex.
   a. Pros: Faster time-to-market, specialized conversation design tools, built-in analytics
   b. Cons: Less control over experience, potential vendor lock-in, recurring costs

Rationale for Final Decision

We implemented separate interaction classes with external service integration because:

1. Parents need flexibility in how they interact while caring for children (hands-free voice or quiet text)
2. Different interaction modalities have distinct technical requirements and optimizations
3. The separate classes allow for specialized error handling and recovery strategies
4. External AI services provide best-in-class capabilities without rebuilding specialized ML models
5. This approach allows us to focus development efforts on parenting-specific features rather than general AI capabilities

5. Subscription and Monetization Model

Chosen Design
Tiered subscription model with free basic features and premium subscription unlocking advanced functionality, integrated with external PaymentGateway.

Alternatives Considered

1. **One-Time Purchase**: Charging a single up-front fee for the application.
   a. Pros: Simpler revenue model, no recurring billing complexity, clear value proposition
   b. Cons: Limited ongoing revenue, higher initial barrier to adoption, difficult sustainability model

2. **Feature-Based Microtransactions**: Individual payment for specific premium features.
    a. Pros: Lower entry barrier, customized user experience, potentially higher revenue
    b. Cons: User friction, complex implementation, potential for negative user perception
3. **Ad-Supported Model**: Free application with revenue from advertisements.
    a. Pros: No cost barrier, potential broad adoption, passive revenue generation
    b. Cons: Privacy concerns with child-related data, potential regulatory issues, degraded user experience

Rationale for Final Decision

We implemented the tiered subscription model because:

1. It provides sustainable ongoing revenue to support continuous development and AI service costs
2. The free tier allows users to experience core value before committing financially
3. Premium features can justify the subscription cost through enhanced AI capabilities
4. Parents are accustomed to subscription models for ongoing services
5. This model avoids advertising-related privacy concerns in a sensitive parenting application

6. UI Interaction Model

Chosen Design

A web interface with both text and speech interaction capabilities.

Alternatives Considered

1. **Native Mobile App**: Developing platform-specific mobile applications.
    - Pros: Better performance, deeper OS integration, better offline capabilities
    - Cons: Higher development cost, platform fragmentation, harder to update
2. **Text-Only Interface**: Focusing solely on text interaction.
    - Pros: Simpler implementation, wider device compatibility
    - Cons: Less accessible, more limited interaction model
3. **Voice-First Design**: Optimizing primarily for voice interaction.
    - Pros: More natural interaction, potentially more accessible
    - Cons: Privacy concerns in public settings, recognition challenges, more complex implementation

Rationale for Final Decision

We chose a web interface with multimodal interaction because:

1. It provides the widest accessibility across devices without platform-specific development

2. The combination of text and speech allows users to choose their preferred interaction mode

3. A web-based approach enables faster iterations and updates

4. It allows for progressive enhancement based on device capabilities

5. It creates a foundation that could be wrapped in a native container later if needed

This design rationale document captures the key decisions made during the design process and the reasoning behind them, providing valuable context for future development and evolution of the mood tracking application.