COMP4910 Senior Design Project 1, Fall 2024
Advisor: AHMET HASAN KOLTUKSUZ

GEMBOS:Global Encrypted Mobile-Based Obscured SMS Application

23.01.2025
Revision 2.0

By:
Berker Vergi, 21070001202
Giray Aksakal, 21070001030
Mert Kahraman, 20070006020
Emir Moralı, 21070006048

Revision History

Revision	Date	Explanation
1.0	10.11.2024	Initial requirements
1.1	10.12.2024	Changing AES Algorithms to Elliptic Curve
1.2	20.12.2024	Research system requirements and revisions
1.3	10.01.2025	Adding Use-Case and Activity Diagram
1.4	15.01.2025	Adding Sequence Diagram
1.5	17.01.2025	Final Revision
2.0	24.01.2025	Last time look up and discussion

Contents

1. Introduction to GEMBOS

Problem Definition:

In an increasingly interconnected world, communication security has become a critical concern. Traditional SMS systems, while widely adopted, suffer from vulnerabilities such as the SS7 protocol flaw, which can be exploited to intercept messages, manipulate data, or compromise user privacy. These vulnerabilities pose significant risks, especially in sectors where the confidentiality and integrity of communication are paramount, such as banking, legal systems, and healthcare.
The SS7 protocol, which forms the backbone of most mobile communication networks, was designed in an era with limited foresight into the modern security landscape. As a result, it lacks essential security measures to protect against eavesdropping, data manipulation, and unauthorized access. These vulnerabilities are exploited to gain unauthorized access to sensitive information, leading to data breaches, financial fraud, and compromised private communications. Additionally, users face challenges in managing their digital communication securely while maintaining convenience. Existing messaging applications often prioritize user experience over robust security measures, leaving gaps in safeguarding sensitive data. This gap is further exacerbated in high-stakes environments, where secure communication is not a luxury but a necessity.

Problem Solution:

GEMBOS (Secure Messaging for Banking, Legal, and Healthcare) is designed as a secure and user-friendly SMS application to address these pressing challenges. By integrating advanced cryptographic protocols and robust security measures, GEMBOS ensures the confidentiality, integrity, and authenticity of user communications.

Key features of GEMBOS include:

1. **Advanced Encryption Algorithms:** The application leverages the Diffie-Hellman algorithm for secure key exchange and the Elliptic Curve for encrypting message content. This ensures that messages are only accessible to intended recipients.
2. **Secure Key Management:** The use of a secure application keystore ensures that encryption keys are managed and stored securely on user devices, reducing the risk of unauthorized access.
3. **Multi-platform Accessibility:** GEMBOS allows users to manage their contacts and send messages seamlessly across mobile and web platforms while maintaining robust security protocols.
4. **Offline Message Handling:** To enhance user experience, GEMBOS provides offline messaging capabilities, enabling users to send and store messages securely even in the absence of an internet connection. These messages are later synchronized with the server when connectivity is restored.
5. **Sector-Specific Applications:** GEMBOS addresses the unique security needs of critical sectors:
    - **Banking:** Secure communication of financial transactions and account details.
    - **Legal Systems:** Protection of confidential case files and client communications.
    - **Healthcare:** Secure transmission of private medical records and patient information.

By providing these features, GEMBOS aims to mitigate the vulnerabilities of traditional SMS systems and cater to the specific security demands of sensitive communication environments.

Literature Review:

The need for secure messaging systems has been well-documented in academic and industry research. Various studies highlight the vulnerabilities of traditional SMS systems and propose cryptographic methods to address these issues. For example:

- **Zhang et al. [1]:** Proposed a secure SMS system utilizing RSA for encryption, which ensures message confidentiality but introduces computational overhead.
- **Chen et al. [2]:** Investigated the integration of secure key exchange protocols like Diffie-Hellman into messaging applications, emphasizing their role in protecting against man-in-the-middle attacks.

While these approaches provide theoretical frameworks for secure messaging, GEMBOS builds on these advancements by integrating multiple cryptographic measures into a cohesive and practical application. By focusing on user experience alongside robust security, GEMBOS bridges the gap between academic research and real-world application.

Challenges Addressed by GEMBOS:

1. **Interception and Data Manipulation:** Traditional SMS systems are vulnerable to interception due to the SS7 protocol flaw. GEMBOS overcomes this challenge by encrypting messages end-to-end, ensuring that only authorized parties can access the content.
2. **Key Management:** Secure storage and management of encryption keys are critical to preventing unauthorized access. GEMBOS employs a application keystore mechanism to manage keys locally on user devices.
3. **Cross-platform Compatibility:** Ensuring secure communication across multiple platforms without compromising user experience is a complex challenge. GEMBOS addresses this by implementing platform-agnostic protocols and ensuring seamless synchronization of messages.
4. **Offline Messaging:** In scenarios where internet connectivity is unavailable, users often face difficulties in securely storing and transmitting messages. GEMBOS resolves this by enabling offline message storage and synchronization.
5. **Sector-specific Security Requirements:** The application is tailored to meet the unique demands of critical sectors such as banking, legal, and healthcare, providing customized solutions for secure communication.

Personalized Approach:

GEMBOS adopts a user-centric approach to secure communication. Recognizing the diverse needs of users, especially in high-stakes environments, the application provides:

- **Customizable Security Settings:** Users can configure encryption settings and access controls based on their specific requirements.
- **Sector-specific Customizations:** For example, healthcare professionals can securely share patient data, while legal professionals can exchange case files without compromising confidentiality.
- **Intuitive User Interface:** Despite its robust security features, GEMBOS maintains a simple and intuitive interface, ensuring accessibility for users with varying technical expertise.

Conclusion:

In an era where communication security is paramount, GEMBOS stands out as a comprehensive solution to the vulnerabilities of traditional SMS systems. By integrating advanced cryptographic algorithms, secure key management, and sector-specific customizations, GEMBOS ensures the confidentiality, integrity, and authenticity of user communications. With its user-centric design and focus on critical sectors, GEMBOS is poised to redefine secure communication standards in an increasingly digital world.

2. Functional Requirements

2.1. Main User Interface and Functions

The GEMBOS application provides a streamlined user interface to facilitate secure communication. The main menu includes the following core functionalities:

GEMBOS Chatting Operations:

1. **Log in or Register an Account:** Users can access the application by registering a new account or logging into an existing one.
2. **Phone Number Verification with Password:** Ensures that users verify their identity through a secure process combining phone number validation and password entry.
3. **Connect Contacts:** Enables users to sync their phone contacts with the GEMBOS application, making it easier to select recipients for messaging.
4. **Send SMS to Selected Contacts:** Allows users to send SMS messages securely to their chosen contacts through the GEMBOS interface.

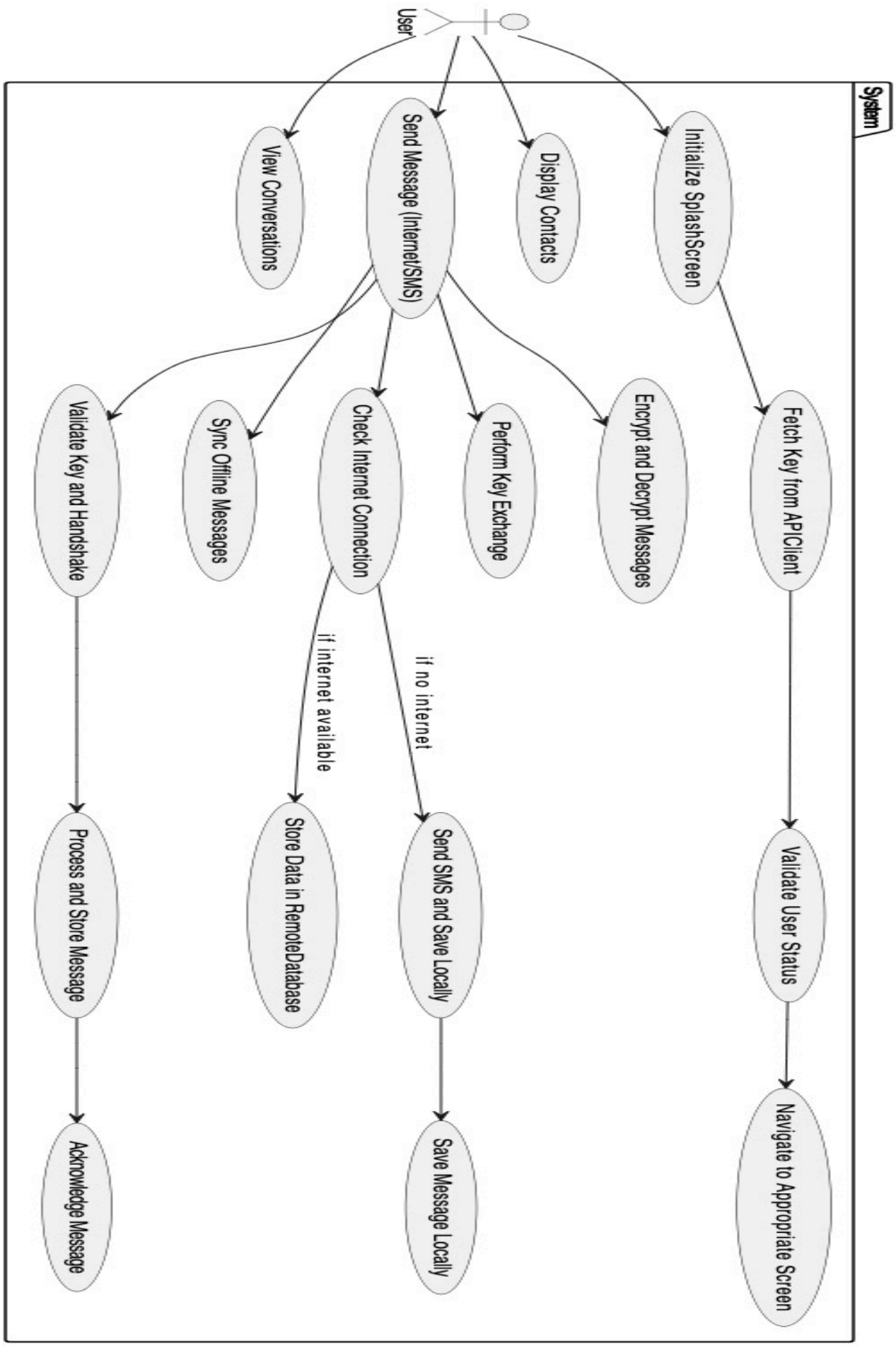
User
System
Initialize SplashScreen
Display Contacts
Send Message (Internet/SMS)
View Conversations
Fetch Key from APIClient
Encrypt and Decrypt Messages
Perform Key Exchange
Check Internet Connection
Sync Offline Messages
Validate Key and Handshake
Validate User Status
Navigate to Appropriate Screen
Send SMS and Save Locally
Store Data in RemoteDatabase
Save Message Locally
Process and Store Message
Acknowledge Message
if internet available
if no internet

figure 1

Key Components and System Flow

The Use Case Diagram identifies the key components and processes involved in the GEMBOS application, emphasizing the user's role as the primary actor interacting with the system. The following subsections outline the operations that the user can initiate and the corresponding system responses.

Actor (User)

The user serves as the central actor, engaging with various system functionalities to achieve secure communication. The main operations initiated by the user include:

- Launching the application via the initialization of the **SplashScreen**.
- Sending messages to contacts using the **Send Message** functionality.
- Viewing previous conversations through the **View Conversations** feature.
- Performing message encryption and decryption for secure communication.
- Verifying internet connectivity before proceeding with message transmission.

Use Cases

Initialize SplashScreen

Upon launching the application, the SplashScreen is initialized. This process acts as the entry point for verifying the user's registration status and directing them to either the main interface or the login/registration screen.

Fetch Key from APIClient

The system retrieves encryption keys from the APIClient to facilitate secure communication. If no encryption key exists, the system generates and securely stores a new key, ensuring robust encryption mechanisms for all subsequent operations.

Validate User Status

The system performs a check to determine the user’s registration status:

- If the user is registered, they are redirected to the main screen.
- If not, they are prompted to log in or register for an account.

Display Contacts

The system synchronizes the user’s phone contacts, displaying them within the application interface. This feature allows users to select a specific contact for secure communication.

Send Message (Internet/SMS)

The system supports two methods of message transmission:

- **When Internet Is Available:** Messages are encrypted using the Diffie-Hellman algorithm, securely stored in the RemoteDatabase, and transmitted to the recipient.
- **When Internet Is Unavailable:** Messages are sent as SMS and stored locally on the device for later synchronization.

Perform Key Exchange

To ensure the security of messages, the system executes a key exchange using the Diffie-Hellman algorithm. This process establishes secure communication channels by facilitating encryption and decryption.

Check Internet Connection

Before transmitting messages, the system verifies internet connectivity:

- If a connection is available, messages are encrypted and stored in the RemoteDatabase.
- If no connection is available, messages are transmitted as SMS and saved locally.

Sync Offline Messages

When internet connectivity is restored, messages stored locally on the device are synchronized with the RemoteDatabase, ensuring no loss of communication data.

Validate Key and Handshake

The system employs a handshake protocol to authenticate encryption keys. This step guarantees data integrity and ensures secure message exchange between users.

Process and Store Message

All transmitted messages are processed and stored in the RemoteDatabase. The system provides feedback to the user upon successful message storage and delivery.

Acknowledge Message

Once the recipient receives a message, the system sends an acknowledgment to the sender, confirming successful delivery.

View Conversations

Users can access and review their message history within the application. During this process, the system decrypts stored messages to make them readable, while maintaining their confidentiality.

Technical Features

Encryption

The system employs a combination of the Diffie-Hellman key exchange protocol and the ,Elliptic Curve algorithm to encrypt and decrypt messages. These techniques ensure the confidentiality and security of user data and messages during transmission and storage.

Database Integration

The RemoteDatabase serves as the central repository for securely storing all user messages and data. Additionally, the system synchronizes locally stored data with the RemoteDatabase whenever internet connectivity is re-established, ensuring seamless integration and data consistency.

Offline Support

The application provides offline messaging functionality, allowing users to send messages even in the absence of an internet connection. These messages are stored locally and synchronized with the server when a connection becomes available.

Message Processing and Feedback

The system delivers real-time feedback to users regarding the status of sent messages. For instance, users are notified when their messages are successfully delivered to the recipient.

Advantages of the System

The GEMBOS system offers several notable advantages:

1. **Security:** The implementation of advanced encryption techniques and a robust handshake protocol ensures the privacy and security of all communications.
2. **Flexibility:** Offline messaging reduces the dependency on constant internet connectivity, enhancing the system's usability in varying conditions.
3. **User-Friendly Design:** The intuitive interface, coupled with seamless contact synchronization, makes the application accessible and convenient for users.

2.2. Enter Registration Information

The registration process is the first interaction users have with the GEMBOS application. This process ensures data integrity, secure account creation, and a seamless user experience.

User Registration Form:

The registration form requires users to provide the following mandatory information:

- **Phone Number***: Ensures that each user account is linked to a unique identifier.
- **Password***: Protects the account from unauthorized access.
- **Submit & Cancel Buttons:**

- **Submit:** Validates the entered data, securely registers the user, and redirects them to the main menu.
- **Cancel:** Clears all fields and navigates the user back to the registration screen.

Login Form:

The login form allows existing users to access the application by providing:

- **Phone Number***
- **Password***

The registration and login mechanisms are secured using cryptographic algorithms, ensuring that sensitive user data is protected from unauthorized access.

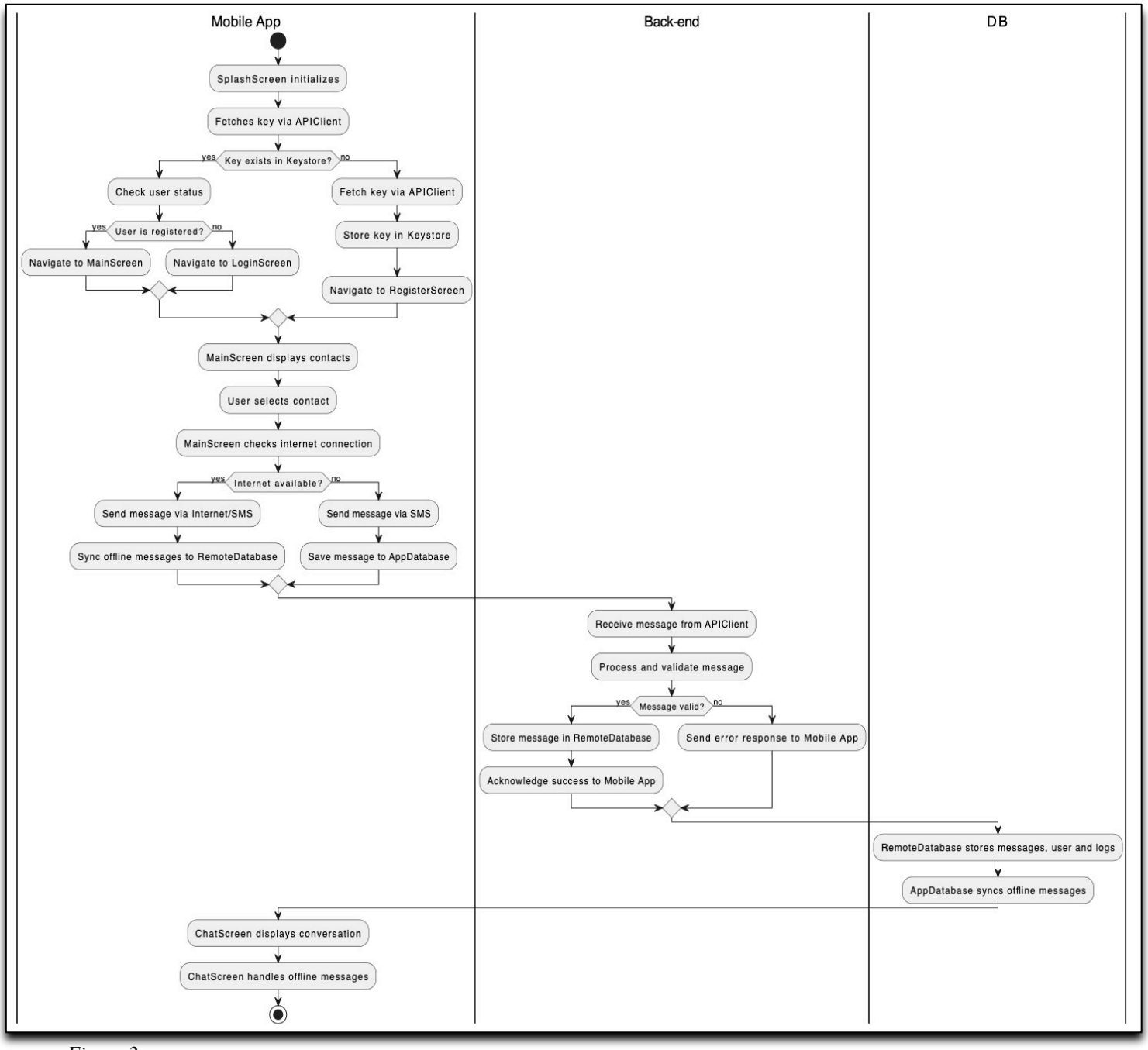

Figure 2

This Activity Diagram illustrates the sequential processes within the GEMBOS system, highlighting the interactions between the **Mobile App**, **Back-end**, and **Database (DB)** layers. It showcases the system's ability to handle secure communication, both online and offline, through coordinated workflows. Each component in the diagram plays a critical role in ensuring the functionality and security of the system.

1. Mobile App Layer

The **Mobile App** acts as the interface between the user and the GEMBOS system. It initiates critical operations such as user authentication, contact management, message transmission, and offline data handling. Below are the key actions:

1.1 SplashScreen Initialization

- The application begins with the initialization of the **SplashScreen**. This serves as the entry point for system processes, preparing the app to fetch required resources such as encryption keys and user data.
- The initialization ensures that all prerequisites for secure communication are in place before proceeding.

1.2 Key Retrieval via APIClient

- The application attempts to retrieve encryption keys stored in the **keystore**:
    - **If a key exists:** It is used for encryption during communication.
    - **If no key exists:** The app fetches a new key from the **APIClient**, which is securely stored for future use.
- This step ensures robust encryption protocols are in place before data transmission begins.

1.3 User Status Validation

- The system verifies the user's registration status:
    - **Registered users** are directed to the **MainScreen**.
    - **New users** are navigated to either the **LoginScreen** or **RegisterScreen** to complete the onboarding process.

1.4 MainScreen Display

- After successful authentication, the system displays the **MainScreen**, which includes a synchronized list of the user’s contacts. This enables the user to seamlessly select a recipient for communication.

1.5 Internet Connectivity Check

- Before transmitting messages, the system checks the availability of an internet connection:
    - **If connected:** Messages are encrypted and sent via the internet. They are also stored in the **RemoteDatabase**for centralized logging.
    - **If not connected:** Messages are sent via SMS and stored locally in the **AppDatabase** for later synchronization.

1.6 Sync Offline Messages

- When an internet connection is restored, messages stored in the local **AppDatabase** are synchronized with the **RemoteDatabase**, ensuring that the communication history remains consistent and accessible.

1.7 ChatScreen Operations

- The **ChatScreen** manages the display of conversations, including messages sent both online and offline. Users can view past interactions while the system ensures the secure handling of all data.

2. Back-end Layer

The **Back-end** layer processes messages transmitted by the Mobile App and ensures their validity and security. Key actions include:

2.1 Receiving Messages

- Messages sent from the Mobile App are received by the back-end system via the **APIClient**. This step ensures the secure handoff of user data for processing.

2.2 Message Validation

- The back-end validates each message for integrity and compliance with security protocols:
    - **Valid messages** proceed to storage in the **RemoteDatabase**.
    - **Invalid messages** trigger an error response, which is sent back to the Mobile App to inform the user.

2.3 Message Storage

- All valid messages are securely stored in the **RemoteDatabase**. This ensures centralized data management, enabling consistency across devices.

2.4 Acknowledgment

- After successfully storing a message, the back-end sends a confirmation to the Mobile App. This provides real-time feedback to the user, confirming that their message has been delivered and logged.

3. Database Layer

The **Database (DB)** layer serves as the central repository for managing user and communication data. It comprises the **RemoteDatabase** and **AppDatabase**, each with specific responsibilities:

3.1 RemoteDatabase

- The **RemoteDatabase** securely stores all messages, user data, and logs. This centralized storage ensures that communication history is preserved and accessible across devices.

3.2 AppDatabase Synchronization

- The **AppDatabase**, located on the user's device, temporarily stores messages and user data when offline. Upon regaining internet connectivity, the system synchronizes the locally stored data with the **RemoteDatabase**, ensuring consistency and preventing data loss.

Key Technical Processes

1. **Encryption and Key Management:**
    - Encryption keys are securely managed via the **APIClient** and stored in the keystore. Messages are encrypted using Diffie-Hellman and Elliptic Curve algorithms, ensuring end-to-end security.
2. **Offline Messaging Support:**
    - The system seamlessly transitions between online and offline modes. Offline messages are stored locally and synchronized with the RemoteDatabase when connectivity is restored.
3. **Real-Time Feedback:**
    - The system provides users with real-time status updates for sent messages, ensuring transparency and enhancing the user experience.

Advantages of the Workflow

1. **Security:** The use of robust encryption protocols and secure key management ensures that all communication remains private and protected.
2. **Reliability:** Offline messaging and synchronization prevent data loss, ensuring that communication remains uninterrupted.
3. **User Convenience:** The intuitive interface and seamless integration of contacts make the application user-friendly and accessible.
4. **Scalability:** The centralized database architecture supports consistent data management across devices, allowing for future scalability.

2.3. Connect to Contacts

This feature allows users to integrate their phone contacts with the GEMBOS application. Synced contacts are displayed within the application interface, enabling users to:

- View a list of all synced contacts.
- Select specific contacts for secure messaging.
- Synchronize contacts seamlessly across multiple devices.

The contact synchronization process employs secure communication protocols to prevent unauthorized access or data leaks during transfer.

2.4. Two-Factor Authentication and Messaging

Two-Factor Authentication (2FA):

GEMBOS implements a robust two-factor authentication system to verify user identity:

1. Users provide their phone number during login or registration.
2. The system sends a one-time password (OTP) via SMS to the provided phone number.
3. Users must enter the OTP within a given time frame to complete the verification process.

Messaging Feature:

The messaging functionality allows users to securely send SMS messages to their selected contacts. Key features include:

- **End-to-End Encryption:** Messages are encrypted using the Diffie-Hellman key exchange protocol, ensuring secure transmission. The Elliptic Curve algorithm is employed for message encryption and decryption, guaranteeing data confidentiality.
- **Cross-Platform Messaging:** Users can send messages through the mobile or web interface.
- **Key Management:** Encryption keys are securely stored in Android and iOS keystores, enhancing security for both platforms.
- **Offline Support:** In cases where internet connectivity is unavailable, messages are stored locally and synced to the server once the connection is restored.

The messaging process includes an initial handshake between users to establish a secure communication channel. This handshake ensures that encryption keys are exchanged securely before any data transmission occurs.

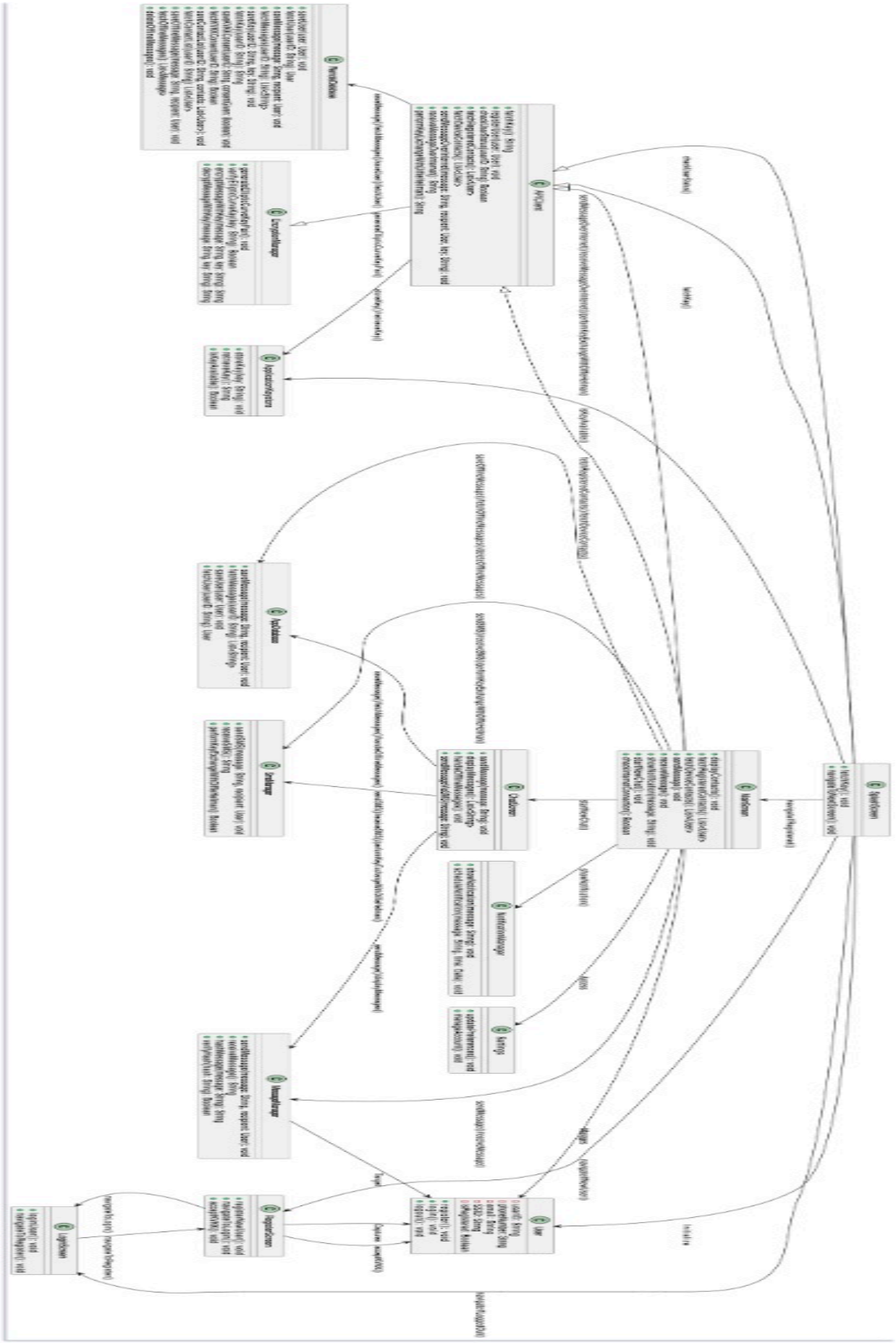

figure 3

The provided class diagram illustrates the structural design of the GEMBOS system, focusing on its key components, their relationships, and the methods that define their interactions. This diagram offers a comprehensive view of how the system manages secure communication, user interactions, and backend processes.

Key Classes and Their Responsibilities

1. SplashScreen

- **Purpose:** Serves as the entry point for the application.
- **Key Methods:**
  - initialize(): Prepares the application by loading necessary resources and initializing components.
  - navigateIfRegistered() and navigateIfNewUser(): Determine whether to direct the user to the main interface or the registration screen, based on their status.

2. MainScreen

- **Purpose:** Acts as the primary user interface after successful login or registration. Displays synchronized contacts and allows users to initiate communication.
- **Key Methods:**
  - displayContacts(contactList: List<User>): Retrieves and displays the user's synchronized contacts.
  - startNewChat(contact: User): Initiates a chat session with a selected contact.
  - showNotification(notification: Notification): Manages in-app notifications.

3. LoginScreen & RegisterScreen

- **LoginScreen:**
  - Handles user authentication by validating their credentials.
  - **Key Methods:**
    - loginUser(userId: String, password: String): Authenticates the user and directs them to the main screen if successful.
  - **RegisterScreen:**
    - Facilitates new user registration by collecting necessary details.
    - **Key Methods:**
      - registerUser(userDetails: User): Validates and stores user information.

4. ChatScreen

- **Purpose:** Manages individual chat sessions, handling message display, input, and synchronization.
- **Key Methods:**
  - sendMessage(message: String): Sends messages to the selected contact.
  - saveOfflineMessage(message: Message): Stores messages locally when offline.
  - fetchMessages(chatHistory: List<Message>): Retrieves previous chat logs for display.

5. MessageManager

- **Purpose:** Orchestrates the sending and receiving of messages, ensuring secure transmission.
- **Key Methods:**
  - sendMessage(content: String, recipient: User): Prepares and encrypts messages for transmission.
  - fetchMessages(chatId: String): Retrieves message history from the database.
  - verifyMessage(content: String): Validates message integrity before processing.

6. EncryptionManager

- **Purpose:** Ensures all communications are encrypted for security.
- **Key Methods:**
  - generateKeyPair(): Creates a secure key pair for encryption.
  - encryptMessage(message: String, key: String): Encrypts messages before transmission.
  - decryptMessage(encryptedMessage: String, key: String): Decrypts incoming messages.

7. Keystore

- **Purpose:** Manages encryption keys securely within the application.
- **Key Methods:**
  - storeKey(key: String): Saves encryption keys for future use.
  - retrieveKey(): Fetches saved keys for message decryption.

8. RemoteDatabase & AppDatabase

- **RemoteDatabase:**
  - Acts as the central repository for all messages and user data, accessible across devices.
  - **Key Methods:**
    - saveMessage(message: Message): Stores messages in the database.
    - syncOfflineMessages(messages: List<Message>): Synchronizes locally stored messages with the server.
- **AppDatabase:**
  - Handles local data storage for offline usage.
  - **Key Methods:**
    - saveOfflineMessage(message: Message): Temporarily saves messages until internet connectivity is restored.

9. SmsManager

- **Purpose:** Facilitates SMS-based communication when internet connectivity is unavailable.
- **Key Methods:**
  - sendSMS(message: String, recipient: User): Sends messages via SMS.
  - receiveSMS(): Handles incoming SMS messages.

10. NotificationManager

- **Purpose:** Manages system notifications for the user.
- **Key Methods:**
  - throwNotification(notification: Notification): Sends notifications for events such as new messages or updates.
  - manageNotifications(notificationQueue: List<Notification>): Queues and handles multiple notifications.

Relationships Between Classes

1. **Direct Interactions:**
   - The MainScreen directly interacts with the ChatScreen to handle communication-related tasks.
   - The MessageManager communicates with the EncryptionManager for encrypting and decrypting messages before they are sent or displayed.

2. **Centralized Data Management:**
   - The RemoteDatabase serves as the central hub for storing messages, ensuring data consistency across devices.
   - The AppDatabase bridges the gap during offline usage, synchronizing with the RemoteDatabase when connectivity is restored.

3. **Security-Focused Interactions:**
    - The Keystore and EncryptionManager collaborate to ensure that encryption keys are securely generated, stored, and retrieved for secure messaging.

Key Features of the Class Diagram

1. **Security and Encryption:**
    - The integration of EncryptionManager and Keystore highlights the system's focus on secure communication. The use of encryption protocols ensures that messages remain private and protected.

2. **Offline and Online Support:**
    - The interaction between AppDatabase and RemoteDatabase ensures seamless communication even when the user is offline. Messages are stored locally and synchronized later, preventing data loss.

3. **User-Centric Design:**
    - Classes like MainScreen, LoginScreen, and ChatScreen ensure a user-friendly interface, prioritizing ease of access and smooth navigation.

4. **Extensibility:**
    - The modular design of the system enables easy scalability. New features or enhancements can be integrated into specific classes without affecting the overall architecture.

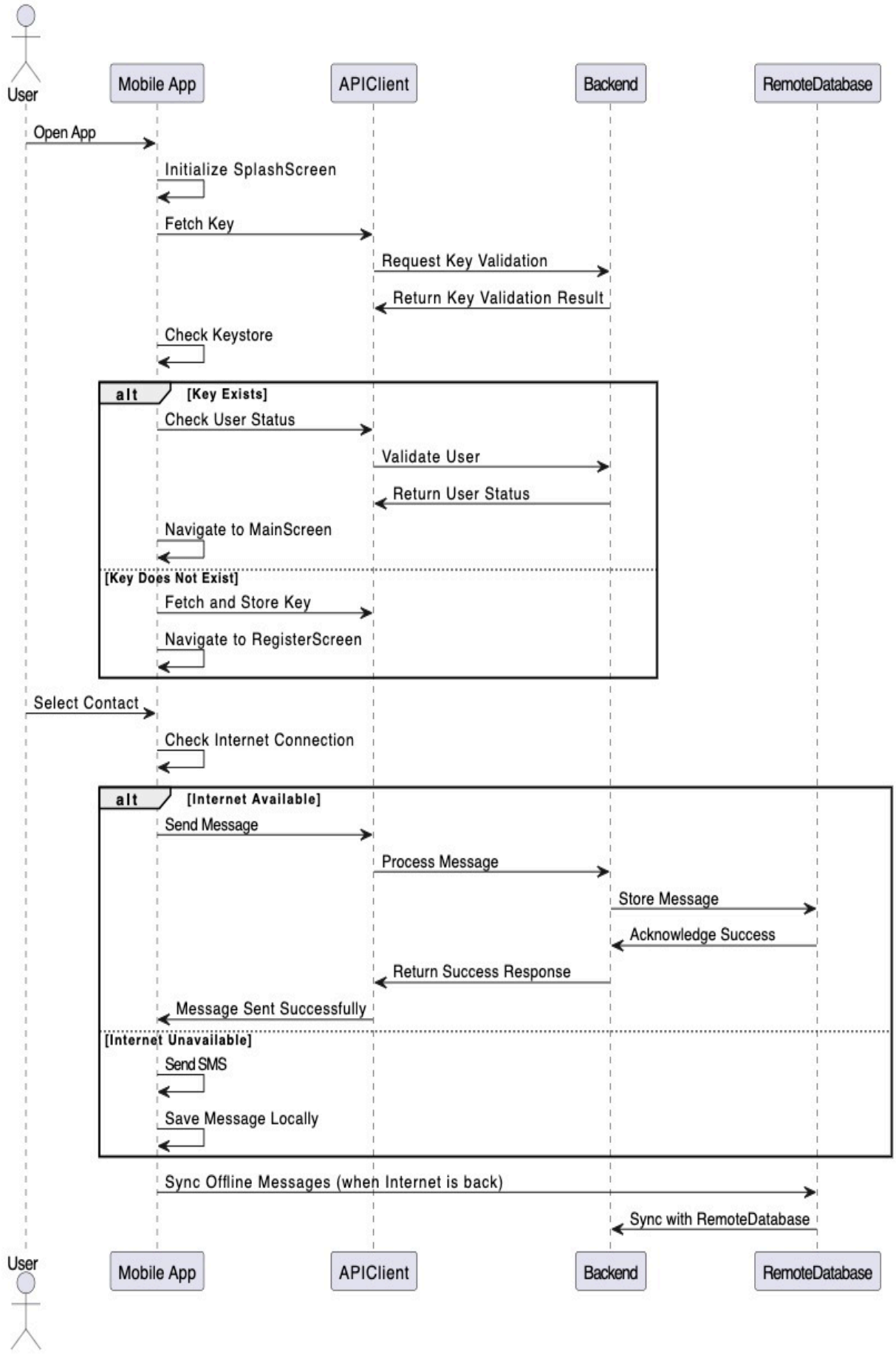
User
Mobile App
APIClient
Backend
RemoteDatabase
Open App
Initialize SplashScreen
Fetch Key
Request Key Validation
Return Key Validation Result
Check Keystore
alt
[Key Exists]
Check User Status
Validate User
Return User Status
Navigate to MainScreen
[Key Does Not Exist]
Fetch and Store Key
Navigate to RegisterScreen
Select Contact
Check Internet Connection
alt
[Internet Available]
Send Message
Process Message
Store Message
Acknowledge Success
Return Success Response
Message Sent Successfully
[Internet Unavailable]
Send SMS
Save Message Locally
Sync Offline Messages (when Internet is back)
Sync with RemoteDatabase
User
Mobile App
APIClient
Backend
RemoteDatabase

Figure 4

This sequence diagram illustrates the flow of interactions between the **User**, **Mobile App**, **APIClient**, **Backend**, and **RemoteDatabase** in the GEMBOS system. It highlights the process of application initialization, user authentication, message sending, and offline synchronization, ensuring a secure and seamless communication experience.

Key Components and Their Roles

1. **User**:
   The primary actor initiating interactions with the system, such as opening the application, sending messages, and selecting contacts.
2. **Mobile App**:
   The client-side interface that facilitates user actions, including authentication, contact management, and message transmission. It serves as the gateway to the backend services via the **APIClient**.
3. **APIClient**:
   Acts as the intermediary between the mobile app and backend, handling API requests, key validation, and secure communication.
4. **Backend**:
   Processes user requests, validates keys and user data, and ensures secure storage and retrieval of messages.
5. **RemoteDatabase**:
   Provides centralized storage for user data and messages, enabling synchronization across devices and ensuring data consistency.

Detailed Flow Description

1. Application Initialization

- The process begins with the **User** opening the application on the **Mobile App**.
- The **Mobile App** initializes the **SplashScreen**, preparing the application for operation.
- The app then fetches the encryption key from the **Keystore**:
    - **Alternative Flow: Key Validation**
        - If the key exists, the **APIClient** requests validation from the **Backend**, which verifies the key and returns the result.
        - If the key does not exist, the **Mobile App** fetches a new key from the **APIClient**, stores it securely in the keystore, and navigates the user to the **RegisterScreen** for account creation.

2. User Authentication

- If a valid key is present, the system checks the user's status:
    - **Registered User**: The user is directed to the **MainScreen**, where synchronized contacts are displayed.
    - **Unregistered User**: The user is navigated to the **RegisterScreen** for registration.

3. Message Sending

- The **User** selects a contact from the **MainScreen** and initiates the message sending process.
- The **Mobile App** checks the internet connection:
    - **Alternative Flow: Internet Available**
        - The **Mobile App** encrypts the message and sends it via the **APIClient** to the **Backend**.
        - The **Backend** processes the message, stores it in the **RemoteDatabase**, and returns a success acknowledgment.
        - The **Mobile App** notifies the user that the message was successfully sent.
    - **Alternative Flow: Internet Unavailable**
        - If no internet connection is available, the **Mobile App** sends the message via SMS and stores it locally.

- The stored message will be synchronized with the **RemoteDatabase** once the internet connection is restored.

4. Offline Synchronization

- When the internet connection is re-established, the **Mobile App** automatically syncs locally stored messages with the **RemoteDatabase** via the **APIClient** and **Backend**.
- This ensures that all messages are logged in the centralized database for consistency and future access.

Key Features Highlighted in the Diagram

1. **Secure Key Management**:
   - The system validates and securely stores encryption keys using the **Keystore** and **APIClient**, ensuring robust data protection.
2. **Seamless User Experience**:
   - The application dynamically adapts to the user's registration status, guiding them to the appropriate screen (MainScreen or RegisterScreen).
3. **Dual Message Transmission Modes**:
   - The system supports both online (via the internet) and offline (via SMS) communication, ensuring uninterrupted messaging capabilities.
4. **Offline Support and Synchronization**:
   - Locally stored messages are automatically synchronized with the **RemoteDatabase** when the internet becomes available, maintaining data consistency.
5. **Backend Processing and Acknowledgment**:
   - The backend ensures messages are securely processed, stored, and acknowledged, providing feedback to the user.

Summary of Technical Features:

1. **Secure Registration and Login:** The system implements robust security measures for user authentication, combining OTP (One-Time Password) verification and password-based access control. This dual-layered approach ensures that only verified users can access the application, thereby safeguarding user identity.

2. **Contact Integration:** The application facilitates seamless synchronization of user contacts from their device. These contacts are securely stored within the system, allowing users to easily select recipients for communication while maintaining data integrity and privacy.

3. **End-to-End Encryption:** Secure communication is achieved through the use of advanced encryption techniques. The system employs the Diffie-Hellman algorithm for secure key exchange and the Elliptic Curve algorithm for message encryption and decryption, ensuring that all communications remain confidential and tamper-proof.

4. **Cross-Platform Functionality:** GEMBOS offers a consistent and cohesive user experience across multiple platforms, including mobile and web interfaces. This ensures that users can seamlessly switch between devices without compromising functionality or security.

5. **Offline Message Handling:** The system supports the storage of messages locally when internet connectivity is unavailable. These messages are automatically synchronized with the central database once connectivity is restored, ensuring uninterrupted communication and preventing data loss.

These functionalities collectively ensure that GEMBOS provides a secure, efficient, and user-centric communication platform, meeting the diverse needs of its users.

3. Non-Functional Requirements

3.1. Development Environment

Frontend - MOBILE

The GEMBOS application utilizes modern technologies for mobile frontend development to ensure seamless user interaction and a robust interface:

- **Languages:** Java, Swift.
  These languages are used to develop Android and iOS applications, providing native support and optimal performance on their respective platforms.

- **Frameworks/Libraries:**
  - **Android SDK:** Enables efficient development of Android applications with rich features and system-level integration.
  - **SwiftUI:** Simplifies UI design for iOS apps with declarative syntax and seamless integration into the Apple ecosystem.
  - **Android Keystore:** Manages cryptographic keys in a secure environment for Android devices, ensuring data protection.
  - **iOS Keychain:** Provides a secure mechanism for storing sensitive data like encryption keys on Apple devices.

Backend

The backend infrastructure is designed to handle secure communication and robust data processing:

- **Languages:** Java.
  Chosen for its reliability, scalability, and compatibility with enterprise-grade backend development.
- **Frameworks:**
  - **Spring Boot:** Provides a lightweight and modular architecture, enabling rapid development of RESTful APIs and handling complex backend operations.

Databases

SQL: MySQL is the primary database management system used to handle structured data, ensuring reliability, scalability, and efficiency for GEMBOS's backend operations.

Database Management Strategy:

- Relational database management to handle user profiles, contact information, and message histories.
- Periodic backup strategies, including daily incremental backups and weekly full backups, to ensure data recovery in case of failures.

DevOps and Hosting

To streamline development and ensure reliable hosting, the following tools and platforms are employed:

- **GitHub:** Used for version control, collaboration, and code repository management.
- **AWS (Amazon Web Services):** Provides scalable and reliable cloud hosting for backend services, ensuring high availability and performance.

3.2. Security

The GEMBOS system places a strong emphasis on security to safeguard user communication and data:

- **SSL (Secure Sockets Layer):** Encrypts data during transmission to protect against interception.
- **Password Hashing:** All user passwords are hashed using secure algorithms (e.g., SHA-256 or bcrypt) to prevent unauthorized access.
- **API Key Management:** API keys are securely stored in environment variables, reducing the risk of exposure.
- **Encryption Algorithms:**
  - **Diffie-Hellman Algorithm:** Ensures secure key exchange between users for encrypted communication.
  - **Elliptic Curve Cryptography (ECC):** Provides strong encryption for securing sensitive data, ensuring confidentiality, integrity, and efficiency with smaller key sizes compared to traditional methods.

- **Handshake Protocol:** Establishes a secure connection between users before initiating any messaging, ensuring mutual authentication.

3.3. Scalability

The system is designed with scalability to handle increasing user demands and data growth:

- **Scalable Database:** MySQL's robust architecture supports horizontal scaling, ensuring it can manage growing user data, including message logs and contact lists.
- **API Architecture:** RESTful APIs are designed to efficiently handle increased traffic and concurrent requests, maintaining high performance.

3.4. Testing

The GEMBOS system employs rigorous testing methodologies to ensure the reliability and robustness of its features:

- **Unit Testing:** Verifies individual components such as message encryption, user authentication, and contact synchronization to ensure proper functionality.
- **Integration Testing:** Ensures seamless interaction between various system components, such as backend APIs, database operations, and frontend interfaces.
- **User Acceptance Testing (UAT):** Involves end-users in testing to validate that the application meets their expectations and requirements.

3.5. Data Privacy

The system prioritizes the privacy of user data, ensuring compliance with industry standards and regulations:

- **Data Encryption:** All user data, including messages and contact information, is encrypted at rest and during transmission using Elliptic Curve encryption.
- **Secure Storage:** Sensitive information, such as user credentials and encryption keys, is securely stored in the Android Keystore and iOS Keychain.
- **Privacy Policies:** The system ensures compliance with data protection regulations, such as GDPR (General Data Protection Regulation), to safeguard user rights.

4. References

1. General Mobile. (2024, November 8). Meeting with representatives from General Mobile regarding potential improvements on our project.
2. Repel Cyber Security. (2024, November 8). Meeting with representatives from Repel Cyber Security regarding potential improvements and future collaboration opportunities on our project.
3. Diffie, W., & Hellman, M. (1976). **New Directions in Cryptography**. IEEE Transactions on Information Theory.
4. Miller, V. S. (1986). **Use of Elliptic Curves in Cryptography**. Advances in Cryptology – CRYPTO '85 Proceedings, 417, 417–426. DOI: 10.1007/3-540-39799-X_31