Functional Requirements (FRs)

R1: User Registration and Authentication

This section defines how users will be able to securely gain access to the platform including account creation and registration, secure credential verification, session management and logout.

R1.1:User Registration

R1.1.1: Registration of a New User
The system shall allow an admin to register a new user by providing their full name, email address, and a randomly generated password.
R1.1.2: Email Format and Uniqueness Validation
The system must validate the format of the submitted email and ensure it is not already associated with an existing user.
R1.1.3: Role Assignment
The system must allow an admin to assign a specific role from a list of predefined roles during registration.

R1.2: Login and Credential Verification

R1.2.1: User Login
The system must authenticate users by verifying their email and password, issuing a secure session token upon success.
R1.2.2: Email Verification
The system must send a time-sensitive verification token to a newly registered email, which the user must activate through verification in order to gain access to the platform.
R1.2.3: Credential Security
The system shall store user passwords using secure, one-way hashing (bcrypt) to prevent credential theft.
R1.2.4: Failed Log In
The system must implement rate limiting and account lockout mechanisms after repeated failed login attempts.

R1.3: Session and Token Management

R1.3.1: Session Token Expiry
The system shall ensure that issued tokens expire after a configurable period of inactivity.
R1.3.2: Reconnection and Token Refresh
The system should automatically refresh session tokens or redirect the user to reauthenticate upon expiration.
R1.3.3: Logout
The system must allow users to log out manually, which just revoke their current session token and mark the session inactive.

R2: Role & Permission Management (RBAC)

This section defines how the system manages roles, permissions, and access control using a Role-Based Access Control (RBAC) strategy. It outlines how permissions are enforced, roles are assigned, and security is maintained throughout the application.

R2.1: Role Definition and Storage

R2.1.1: Predefined Roles

The system shall define and store a set of predefined roles (e.g., Admin, Investigator, Responder, Analyst) using an ENUM-based schema in the database.

R2.1.3: Access control

Each role shall represent a set of responsibilities and access boundaries defined by associated permissions.

R2.2: Permission Definition and Association

R2.2.1: Permissions that map to actions

The system shall define a comprehensive list of granular permissions (e.g., `CREATE_CASE`, `VIEW_EVIDENCE`, `ASSIGN_ROLE_TO_USER`), which reflect all actionable operations in the system.

R2.2.2: Permission & Role Association

Permissions shall be stored in a dedicated table and shall be associated with one or more roles via a many-to-many relationship.

R2.2.3: Hierarchical Permission Inheritance

The system shall support hierarchical permission inheritance if required (e.g., Admin implicitly has all permissions granted to other roles).

R2.3: Permission Enforcement

R2.3.1: Access Control Authorization

The system shall enforce access control at the API level using middleware that verifies whether the authenticated user has the required permission for the requested action.

R2.3.2: Unauthorized Attempt Error Response & Audit Logging

Unauthorized access attempts shall result in appropriate HTTP error responses (e.g., 403 Forbidden) with error logging for audit purposes.

R2.3.3: Centralised Permissions Check

Permission checks shall be centralized and reusable to ensure consistent enforcement across all services and endpoints.

R2.4: Role Assignment Scope

R2.4.1: Platform-Wide Global Roles & Case-Specific Roles
The system shall support **global roles**, which apply platform-wide, and **case-specific roles**, which grant permissions limited to a particular case context.

R2.4.2: Role/User Mapping
The system shall allow mapping users to roles per case (e.g., an investigator may be assigned to `Case A` only, with restricted visibility to `Case B`).

R2.4.3: Role Enforcement
Case-level roles shall be enforced using contextual authorization checks during case-related operations.

R2.5: Administrative Role Management

R2.5.1: System & Case-Level Role Viewing
The system shall allow administrators to view a list of all users along with their assigned global and case-level roles.

R2.5.2: Updating User Roles
The system shall allow administrators to assign or update a user's role through a secure interface or API, with appropriate logging of the change.

R2.5.3: Case Assignment & Role Designation
The system shall allow only admins to assign users to a specific case and designate a role (e.g., `Investigator`, `Responder`) for each.

R3: Case Management
This section outlines the functional requirements for creating, managing, and updating investigation cases within the system. It includes functionality related to case lifecycle, access, team assignment, and operational metadata such as status, priority, and investigation stage.

R3.1: Case Creation

R3.1.1: Role Case Creation
The system shall allow only admins to create a new case by providing a title, description, their name, and a team name.

R3.1.2: Case ID
The system shall generate a unique identifier (e.g., UUID) for each case upon creation and store a timestamp of creation.

R3.1.3: Audit Log Case Creation
The system shall log every case creation event, including:

- User ID of the creator
- Case ID and title
- Timestamp of creation
- Initial values (description, team name, owner)

R3.2: Case Viewing

R3.2.1: Assigned Case Viewing
The system shall allow users to view their assigned cases.
R3.2.2: Filtering and Sorting
The case listing shall support filtering and sorting by fields such as status, priority, creation date, or investigation stage.
R3.2.3: Case Details & Assigned Users Viewing
The system shall allow viewing of individual case details, including assigned users, status, priority, timeline history, and metadata.
R3.2.4: Audit Logging of Case Viewing
The system shall log each case view action, each log will include the following mandatory fields, and any other attributes deemed necessary for specific actions:

- User ID of the viewer
- Case ID viewed
- Timestamp of access

R3.3: Case Status Updates

R3.3.1: Authorized Case Update Access
The system shall allow authorized users with the `UPDATE_CASE_STATUS` permission to update the status of a case to one of the predefined states (e.g., `open`, `under_review`, `closed`).
R3.3.2: Audit Logging of Case Status Updates
Case status transitions shall be logged with:

- Case ID
- Acting user ID
- Previous and new status values
- Timestamp of change

R3.3.3: Restrictive Case Status Update
The system shall enforce any defined status transition rules (e.g., disallow reopening a closed case without elevated permissions).

R3.4: Case Role Assignment

R3.4.1: Permission
Assigned roles shall define the user's permissions within that specific case context, as enforced by the RBAC system.
R3.4.2: Updating and Revoking Case Level Roles
The system shall support updating and revoking case-level role assignments.
R3.4.3: Audit Logging of Case-Level Role Assignments and Updates
The system shall log all case-level role assignments or updates, including:

- Acting user ID
- Target user ID
- Case ID
- Assigned or revoked role
- Timestamp of the action

R3.5: Case Metadata Updates

R3.5.1: Authorized Users Can Update Case Metadata

The system shall allow authorized users with the `UPDATE_CASE_METADATA` permission to update the **priority** of a case (e.g., `low`, `medium`, `high`, `critical`).

R3.5.2: Investigation Stage Updates By Authorized Users

The system shall allow authorized users to update the **investigation stage** of a case (e.g., `analysis`, `research`, `evaluation`, `finalization`), used for tracking case progress.

R3.5.3: Audit Logging for Investigation Stage Updates

The system shall log all changes to case metadata (priority or stage), including:

- Case ID
- Acting user ID
- Field modified (e.g., priority or stage)
- Old and new values
- Timestamp of change

R4: Evidence Management

This section outlines the requirements the system shall fulfill for the secure and careful management of digital evidence, including file upload to IPFS, metadata storage, retrieval, deletion, and integrity verification.

The system shall ensure that all critical actions are logged for accountability and an immutable chain-of-custody.

R4.1: Uploading Evidence

R4.1.1: Evidence Upload By the Admin

The system shall allow only the system admin to upload evidence. This is so as to ensure fine-grained control over the handling of evidence given the sensitivity of evidence uploaded to the platform.

R4.1.2: Uploaded Evidence Stored on IPFS

Upon successful upload, evidence files shall be stored on IPFS, and their CIDs (Content Identifier) shall be returned and stored in PostgreSQL to allow the retrieval of evidence.

R4.1.3: Audit Logging For Uploaded Evidence

The system shall log each evidence upload, and include the following attributes:

- User ID of the uploader
- Case ID (where applicable)
- Filename and file type
- IPFS CID and file size
- Timestamp of upload

R4.2: Checksum Generation

R4.2.1: Computed Checksum for Each Uploaded File
The system shall compute a cryptographic checksum (e.g., SHA-256) of each uploaded file at the time of upload.

R4.2.2: Checksum Stored as Part of Metadata
The checksum shall be stored in the evidence metadata for future validation.

R4.2.3: Audit Log Computed Checksum
The system shall log the computed checksum along with the evidence upload event (covered in R4.1.3).

R4.3: Metadata Storage

R4.3.1: Metadata Stored Fields
The system shall store the following metadata for each evidence file:

- IPFS CID
- File size (in bytes)
- File type (MIME)
- Original filename
- Upload timestamp
- Uploader's user ID
- Associated case ID
- Checksum value

R4.3.2: Metadata Stored In PostgresQL
Metadata shall be stored in a structured relational database (e.g., PostgreSQL) and linked to the case and user entities.

R4.4: Evidence Retrieval

R4.4.1: Case Level, Evidence Metadata Retrieval
The system shall allow users which are assigned to a case to retrieve evidence metadata and details of all the evidence uploaded for that case by:

- Evidence ID
- Case ID

R4.4.2: Audit Logging for Evidence Access
The system shall log every evidence access event. Each log entry shall include the following mandatory fields:

- User ID of the requester
- Evidence ID or Case ID
- Timestamp of access
- IP address (optional for forensics)

R4.5: Evidence Deletion

R4.5.1: Evidence Deletion Authorization
The system shall allow only the system admin to delete evidence.
R4.5.2: Mark Evidence for Deletion
Deleted evidence shall be soft-deleted (flagged) to preserve auditability unless a hard delete is explicitly authorized by an admin.
R4.5.3: Audit Logging for Evidence Deletion
The system shall log each deletion action. Each log entry shall include the following mandatory fields:

- User ID performing deletion
- Evidence ID
- Case ID
- Timestamp of deletion
- Reason for deletion (optional comment field)

R4.6: Checksum Validation

R4.6.1: Authorization for Evidence Integrity Check
The system shall allow system admins to validate the integrity of evidence files by re-computing and comparing its checksum.
R4.6.2: Compromised Evidence
The system shall determine and display the outcome of the checksum validation by comparing the current hash of the evidence file with its originally stored checksum. If the values differ, the evidence shall be flagged as compromised; if they match, it shall be marked as intact. This result must be visibly presented to the validating user and persistently recorded for auditing and chain-of-custody integrity.
R4.6.3: Audit Logging for Evidence Validation
The system shall log all integrity validation attempts, including:

- User ID performing the validation
- Evidence ID
- Timestamp
- Validation result (`match` / `mismatch`)

R5: Evidence Metadata & Tagging

This section defines how the system handles custom metadata and tags associated with evidence files. Metadata is stored in PostgreSQL during upload, with support for updating, retrieving, and querying by tags for efficient organization and filtering.

R5.1: Metadata Storage (Embedded JSON)

R5.1.1: Embedded Metadata Format
The system shall store metadata associated with an evidence file as a structured JSON object during upload.
R5.1.2: Metata Field Key Management
The system shall enforce validation to ensure metadata keys are non-empty, unique within the object, and do not overlap with reserved system fields.
R5.1.3: Metadata Updates

The system shall allow only system admins to update JSON evidence metadata objects for a given evidence record. This may involve adding new keys(fields), updating values, or removing existing fields.

R5.1.4: Audit Logging for metadata storage and updates

All metadata storage and update actions shall be logged, including:

- User ID
- Evidence ID
- Fields added/updated
- Timestamp

R5.2: Metadata Retrieval (JSONB)

R5.2.1: Metadata Viewing Permissions

The system shall allow users with the `VIEW_EVIDENCE_METADATA` permission to retrieve the `metadata` field of a given evidence record, returned as a JSON object.

R5.2.2: Storage Format

The JSON metadata shall be fully structured and consistent in key-value format.

R5.2.3: Audit Logging for Metadata Retrieval

Each metadata retrieval action shall be logged, including:

- User ID
- Evidence ID
- Timestamp

R5.3: Evidence Tagging

R5.3.1: Users with Tagging Permissions

The system shall allow users who are assigned to a case to tag evidence of that particular case.

R5.3.2: Tag Storage format

Tags shall be normalized (e.g., case-insensitive, trimmed) and stored in a PostgreSQL table with a many-to-many relationship to evidence records.

R5.3.3: Removing Tags

The system shall allow authorized users to remove tags from evidence.

R5.3.4: Audit Logging for Tag operations

The system shall log all tag operations, including:

- User ID
- Evidence ID
- Tag(s) added or removed
- Timestamp

R5.4: Searching and Listing by Tag

R5.4.1: Case Level Evidence Tag Viewing and Filtering
The system shall allow users assigned to a case to list and search for evidence files by tag.

R5.4.2: Searching Feature
Tag-based search shall support exact match, partial match, and compound tag filters.

R5.4.3: Audit Logging for Tag Searching
All search actions involving tags shall be logged with:

- User ID
- Search tags or keywords used
- Timestamp

R6: Audit Logging

This section defines how the system captures and manages logs of critical user actions for accountability, traceability, and compliance. Logs are securely stored and queryable by authorized personnel.

R6.1: Logging of Critical Actions

R6.1.1: Audit Log major system actions
The system shall automatically log all critical user actions, including but not limited to:

- User login and logout
- Evidence upload or deletion
- Case creation and status changes
- Role assignments and revocations
- Metadata updates
- Permission violations or unauthorized attempts

R6.1.2: Audit Log Fields
Each audit entry shall include:

- The action performed (e.g., `LOGIN_SUCCESS`, `UPLOAD_EVIDENCE`, `ASSIGN_ROLE`)
- The user ID performing the action
- The target entity (e.g., case ID, evidence ID, user ID)
- A contextual description (optional, e.g., IP address, browser agent)
- A timestamp (`TIMESTAMPTZ`)

R6.1.3: Audit Log Trigger
Audit logging must be triggered by backend services using a consistent and centralized mechanism (e.g., middleware, service-level logging function).

R6.2: Retrieval and Filtering of Logs

R6.2.1: Recent Activity Viewing
The system shall allow all users to view audit logs of their recent activities.
R6.2.2: Audit Log Retrieval Filtering
The retrieval endpoint shall support filtering by:

- User ID
- Case ID
- Action type
- Time range

R6.2.3: Sorting and Pagination
Results shall be paginated, sorted by newest-first, and returned in a structured format (e.g., JSON array of log entries).

R6.3: Timestamping and Context Preservation

6.3.1: Entry Log Timestamping
Each audit entry shall be stored with a server-generated timestamp reflecting when the action occurred.
R6.3.2: Entry Log Metadata
The system shall capture contextual information if available, including:

- Source IP address
- Request origin (user agent)
- Authenticated session/token ID

R6.3.3: Immutable Log Entries
Audit entries shall be immutable once stored and protected from tampering through access control and restricted write permissions.

R6.4: Exporting Audit Logs

R6.4.1: Exporting Logs
The system shall allow users with the `EXPORT_AUDIT_LOGS` permission to export filtered audit log data in a downloadable format (e.g., CSV or JSON).

R6.4.2: Operations on Logs
The export operation shall apply all currently active filters (user, case, action type, time range) and return only authorized records.
R6.4.3: Log Entries for Export Operations
Each export event shall itself be logged in the audit log, including:

- User ID performing the export
- Export format
- Time of export
- Scope or filters used

R6.4.4: Exported File Details
Exported files shall contain only audit data; no sensitive credentials, secrets, or private keys shall be included.

R7: Dashboard and Analytics

This section outlines requirements for providing users and administrators with summarized, visualized, and actionable insights into system activity, including case trends, evidence flow, and user behavior.

R7.1: Dashboard Overview and Statistics

R7.1.2: User Dashboard Statistics
The dashboard shall display high-level statistics relevant to the user's role and scope, such as:

- Total number of cases accessible to the user
- Number of open vs. closed
- Total number of evidence files uploaded
- Recent user activity (logins, uploads, case status updates)

R7.1.3: Real Time Recent Accesses
Data presented shall reflect real-time or near real-time values from the database, cached appropriately for performance if needed.

R7.2: Charts and Summary Visualizations

R7.2.1: Visual Representation Formats
The system shall provide visual indicators such as:

- Time-series graphs for case creation trends
- Line or area graphs for user activity over time

R7.2.2: Metrics
Summary tiles (KPIs) shall include key metrics such as:

- Number of cases assigned to the current user
- Number of uploads in the last 7 days
- Number of role assignments or changes made

R7.2.3: Visualization Filtering
All visualizations shall support user-level filtering where applicable (e.g., show only my cases, or by role/team).

R7.2.4: Services Provided
The system shall use secure backend APIs to serve aggregated data to frontend dashboards, and ensure no raw sensitive data is exposed unnecessarily.

R8: Notification Service

This section defines requirements for system-generated notifications, primarily focused on account-related email flows such as verification and password reset. Logging is enforced for audit and debugging purposes.

R8.1: Email-Based Notifications

R8.1.1: Email Trigger
The system shall send email notifications for the following events:

- **Email verification** during user registration
- **Password reset requests**, including secure token delivery

R8.1.2: Email Details
Emails shall include:

- A personalized greeting (e.g., user's full name)
- A secure, time-limited verification or reset link/token
- System branding and support contact information

R8.2: Logging of Email Delivery Attempts

R8.2.1: Audit Logging for Notification
The system shall log every email delivery attempt, including:

- Email type (e.g., `VERIFICATION_EMAIL`, `RESET_PASSWORD_EMAIL`)
- Recipient email address
- Status (`sent`, `failed`, `mocked`)
- Timestamp
- Error message if applicable

R8.2.2: Email Notification Environment
Email logs shall be recorded even in development environments where email delivery is mocked or simulated.

R8.2.3: Notification Logs Viewing
Logs shall be accessible to administrators or developers with the `VIEW_NOTIFICATION_LOGS` permission for debugging and auditing.

R9: Secure File Access (E2EE-Ready)

This section defines how the system provides encrypted file access using IPFS and end-to-end encryption mechanisms. It ensures that only authorized recipients, through a secure key exchange protocol, can decrypt and access evidence.

R9.1: Downloading Encrypted Files from IPFS

R9.1.1: IPFS File Download Using CID
The system shall allow users who are assigned to a case to initiate a file download by referencing the files IPFS CID.

R9.1.2: End to End Encryption
All files stored on IPFS shall be encrypted client-side before upload, and decrypted client-side after download using the appropriate decryption key.

R9.1.3: Storage of Encrypted Artefacts
The backend shall **never store or access plaintext file contents** — it shall only relay encrypted blobs and metadata.

R9.2: Key Exchange Protocol Integration (X3DH)

R9.2.1: Key Exchange Protocol
The system shall integrate with a secure key exchange protocol such as **X3DH (Extended Triple Diffie-Hellman)** to enable encrypted key sharing between sender and recipient(s).

R9.2.2: Encryption Keys
The sender shall encrypt a file encryption key using the recipient's X3DH public keys and attach the encrypted symmetric key as part of the evidence metadata or envelope.

R9.2.3: Decryption Keys
The recipient shall use their identity and ephemeral keys to decrypt the symmetric file key and then decrypt the actual file locally.

R9.2.4: Retrieval Processes
All key exchange and retrieval processes shall occur over authenticated, secure channels.

R9.3: Authorization and Decryption Access Control

R9.3.1: RBAC Enforced on File Sharing
The system shall ensure that only users authorized via RBAC and case-specific roles can retrieve the encrypted file or associated key material.

R9.3.2: Authorization and Authentication of Shared Files
Upon download request, the system shall verify:

- The requesting user's identity and role
- That the user is assigned to the case associated with the evidence
- That the user is the intended recipient of the encrypted key (via public key match)

R9.3.3: Audit Logging Upon Unauthorized Access

Access attempts by unauthorized users shall be denied and logged with an audit entry indicating attempted access to secured content.

R10: Case Collaboration

This section defines functionality for users to collaborate within a case using notes, replies, and tagging. Notes are role-protected, timestamped, and audit-logged to ensure accountability and traceability.

R10.1: Posting Case Notes

R10.1.1: Case-Level Note Sharing

The system shall allow users who are assigned to a case to post written notes to a specific case.

R10.1.2: Format of Notes

Notes may contain plain text, markdown, or basic formatting for clarity and context.

R10.2: Replying to Case Notes

R10.2.1: Discussion Threads

Users shall be able to reply to existing notes, forming threaded discussions within the same case.

R10.2.2: Note Ordering

Replies shall be visually grouped with their parent notes and ordered chronologically.

R10.3: Timestamping and Author Tracking

R10.3.1: Audit Logging for Case Notes

Every case note and reply shall be stored with:

- Author's user ID
- Timestamp of creation
- Case ID
- Parent note ID (for replies)

R10.4: Viewing Case Discussions

R10.4.1: Case-Level Note Viewing

The system shall allow users assigned to a case to view all case notes they are authorized to access, presented in a chronological or threaded format.

R10.4.2: Case-Level Note Grouping

Notes shall be grouped per case and loaded with associated metadata (e.g., author name, created_at).

R10.5: User Tagging for Directed Collaboration

R10.5.1: Tagging Users Within a Case Note

The system shall support tagging users within case notes using a standard

`@username` format.

R10.5.2: Tagging for System Notification
When tagged, the mentioned user shall receive a system notification or email (if notification service is active).

R10.5.3: Tagging Range
The system shall validate that only users assigned to the same case may be tagged.

R10.6: Access Control

R10.6.1: Case-Level Note Access Control
Only users who are assigned to the case shall be allowed to create, view, or reply to notes within that case.

R10.7: Auditing Note Activity

R10.7.1: Audit Log All Note Replies
The system shall log all note and reply actions in the audit log, including:

- Action type (`add_note`, `reply_note`)
- Author's user ID
- Case ID
- Note or reply ID
- Timestamp

R11: Secure Chat (Real-Time Messaging)

This section defines how the system supports secure, encrypted real-time messaging using authenticated WebSockets. It enables one-on-one and group conversations within cases while enforcing strict encryption and access control.

R11.1: WebSocket Authentication

R11.1.1: JWT/Session Token Authentication
The system shall establish WebSocket connections only after authenticating the user using a valid **JWT** or **session token**.

R11.1.2: Authorization Upon Connection
The WebSocket server shall verify the token on connection and reject unauthorized or expired sessions.

R11.1.3: Websocket Linking
Each connected socket shall be linked to the authenticated user ID and limited to their assigned case channels or private message threads.

R11.2: Encrypted Messaging

R11.2.1: End-to-End Messaging
All messages shall be end-to-end encrypted **client-side before transmission**, using pre-shared symmetric keys or session-derived keys.

R11.2.2: Single And Grouped Encryption
The system shall support both:

- **One-on-one encrypted messages** between two users
- **Group encrypted messages** within a case-specific chat

R11.2.3: Messaging Keys and Encryption Protocol
Message encryption keys shall be managed using a secure key exchange protocol (e.g., X3DH or Double Ratchet).

R11.3: Real-Time Delivery

R11.3.1: Real Time Messaging
The system shall deliver messages in real-time using a WebSocket-based gateway service.
R11.3.2: Access Control for Delivered Messages
The gateway shall route messages only to authenticated clients who are valid recipients.
R11.3.3: Undelivered Messages
Undelivered messages shall be queued or temporarily stored for users who are offline.

R11.4: Encrypted Message Storage

R11.4.1: Encrypted Messages
The system shall persist encrypted message payloads in the database, along with:

- Sender ID
- Recipient ID(s) or case ID
- Message timestamp
- Delivery status (`sent`, `delivered`, etc.)

R11.4.2: Storage of Encrypted Messages
Plaintext content shall never be stored server-side.

R11.5: Delivery Acknowledgment

R11.5.1: Message Acknowledgement Status Update
The system shall support acknowledgment of message delivery and update the message status to:

- `sent` (message was dispatched)
- `delivered` (message reached the recipient's client)

R11.6: Message Visibility & Access Control

R11.6.1: Viewing Case-Level Messages
Only users assigned to the same case (or valid peer in one-on-one chat) shall be

able to send or view messages related to that conversation.

R11.6.2: Authorization of Case-Level Messages

Attempts to access messages outside of one's assigned cases or private conversations shall be blocked and logged.

R11.7: Message Retrieval (Encrypted History)

R11.7.1: Retrieval of Encrypted Messages

The system shall allow authenticated users to retrieve their encrypted message history from the backend.

R11.7.2: Client Side Decryption

The backend shall return only encrypted payloads — decryption shall occur on the client using previously negotiated or stored keys.

R11.7.3: History Retrieval

History retrieval shall support pagination and filtering by time or conversation.

R11.8: Encryption Key Management

R11.8.1: Key Rotation

The system shall support secure key rotation or refresh per session, per case, or per user — based on defined security policies.

R11.8.2: Invalidation of Keys

Old keys may be invalidated or archived securely. Key refresh shall be securely negotiated and authenticated using the key exchange protocol.

R11.8.3: Key Storage

Key material shall **never be stored unencrypted** on the backend. All key storage (if any) must be encrypted using a derived key from the user's credentials or device-based secrets.

R12: End-to-End Encryption (E2EE) for Chat

This section outlines how the system uses strong cryptographic protocols to ensure chat messages remain confidential, authentic, and tamper-proof, even from the server. The design follows modern E2EE standards like **X3DH** and optionally **Double Ratchet**.

R12.1: Key Agreement with X3DH

R12.1.1: Encryption Protocol

The system shall use the **X3DH (Extended Triple Diffie-Hellman)** protocol to establish secure shared secrets between sender and recipient.

R12.1.2: Encryption Keys

The following are published on behalf of a user by the system:

- An **Identity Key (IK)**
- One or more **Signed PreKeys (SPK)**
- A set of **One-Time PreKeys (OTPKs)**

R12.1.3: Retrieval of Keys Upon Connection

Upon initiating a new session, the sender shall fetch the recipient's pre-key bundle and derive a shared encryption key using X3DH.

R12.2: Secure Key Storage

R12.2.1: Secure Key Storage

The system shall store users' Identity Keys and PreKeys in a secure **Key Service** or backend module with:

- Encrypted storage (e.g., libsodium's `crypto_secretbox`)
- Access control restricted to the authenticated owner
- Optional client-side derived encryption for Identity Keys using password-derived keys

R12.2.2: Sharing of Only Public Keys

Only the public parts of the keys (IK, SPK, OTPKs) shall be accessible to other users for key exchange. Private keys shall never leave the user's device unencrypted.

R12.3: Client-Side Message Encryption

R12.3.1: Client Side Encryption

All chat messages shall be encrypted **on the sender's device** using a session key derived from X3DH (and optionally advanced to support the Double Ratchet protocol).

R12.3.2: Metadata Storage

Encrypted payloads shall be accompanied by a metadata envelope that includes:

- Sender public key
- Ephemeral public key (if using Double Ratchet)
- Message counter (if using forward secrecy)
- Message authentication code (MAC) or signature

R12.4: Client-Side Message Decryption

R12.4.1: Decryption Keys

Only the **intended recipient** shall be able to decrypt the message using their private identity key and the sender's ephemeral/public keys.

R12.4.2: Client-side Decryption

Message decryption shall occur entirely on the recipient's device — no server-side decryption is allowed.

R12.4.3: Chat Payload

In one-to-one chats, each recipient gets a uniquely encrypted payload. In group chats, per-user encryption or sender keys shall be applied depending on the model.

R12.5: Server Transparency & Content Protection

R12.5.1: No Server Side Decryption

The server shall act solely as a **message relay** and shall be **unable to decrypt,**

alter, or inspect message contents.

R12.5.2: Messaging Format

All stored chat messages (in transit and at rest) shall remain in their encrypted form. The system shall not log or index plaintext chat content.

R12.6: Forward Secrecy and Session Expiry

R12.6.1: Forward Secrecy

The system shall support **forward secrecy** by rotating session keys periodically or after each message, using the **Double Ratchet** protocol or equivalent.

R12.6.2: Session State

Session state shall expire after:

- A period of inactivity
- Session key compromise
- A user logout or explicit key reset

R12.6.3: Re-establishing a Session

Upon session expiration, a new shared secret must be re-established via X3DH before communication can continue.

R12.6.3: Terminating a Session

The system shall terminate a session after an extensive period of inactivity is detected.

R13: Case Timeline Management

R13.1: Event Entry and Linking

The system shall support the creation, linking and reordering of events

R13.1.1: Event Creation

The system shall allow DFIR team members to create events in a chronological timeline. Each event shall include:

- Date and time
- Event description
- Linked evidence items

R13.1.2: Linking Evidence to Events

The system shall allow each timeline event to be linked to one or more evidence items. Linked evidence shall reference the evidence ID and include a short description of relevance.

R13.2: Timeline Editing and Review

R13.2.1: Event Editing

Lead DFIR Investigators shall be able to edit, remove, or reorder timeline events for accuracy and completeness.

R13.2.2: Event Approval

Investigators shall be able to approve events, confirming that linked evidence and descriptions are accurate.

R13.3: Timeline Navigation and Search

The system shall provide filtering and search capabilities to view events by:

- Date or date range
- Event type
- Linked evidence

R13.4: Session and Security Considerations

Timeline modifications shall be logged automatically, including actor ID, timestamp, and action type. Only authorized users with access to the case shall modify or view timeline events.

R14: Graphical Evidence Mapping

R14.1: Evidence Node Management

R14.1.1: Node Creation

The system shall allow DFIR team members to create nodes representing evidence items. Each node shall include:

- Evidence ID
- Evidence type
- Optional short description

R14.1.2: Node Linking

The system shall allow users to visually link nodes to represent relationships, dependencies, or causal connections between evidence items.

R14.2: Map Editing and Highlighting

R14.2.1: Map Editing

Lead DFIR Investigators shall be able to add, edit, or remove nodes and links to ensure accuracy and highlight critical relationships.

R14.2.2: Node/Link Annotations

Users shall be able to annotate nodes or links with notes, comments, or references to timeline events. External Collaborators shall have annotations clearly marked as “External Collaborator.”

R14.3: Map Viewing and Export

The system shall allow authorized users to:

- View the full graphical evidence map
- Export the map in visual formats (PDF, PNG) for reporting or external review

R14.4: Security and Access Control

All map modifications shall be logged automatically. Only authorized users with case access shall view or edit maps. Read-only access shall be enforced for external collaborators unless temporary edit access is explicitly granted.

R15: Case Report Generation

R15.1: Report Composition

R15.1.1: Report Content

The system shall generate structured case reports that may include:

- Evidence items and metadata
- Annotation threads
- Timeline events
- Graphical evidence maps
- Executive summary or custom notes

R15.1.2: Section Selection

Lead DFIR Investigators shall be able to include or exclude specific sections when generating reports.

R15.2: Report Export and Formats

R15.2.1: Export Options

Reports shall be exportable in multiple formats, including PDF and DOCX.

R15.2.2: Preview

The system shall allow users to preview reports prior to export to verify content and layout.

R15.3: Security and Audit

R15.3.1: Access Control

Only authorized users shall generate or export reports. External collaborators may view reports only if explicitly granted access.

R15.3.2: Logging

All report generation actions shall be logged automatically, including user ID, timestamp, and report version.