LOST AND FOUND TRACKING SYSTEM

STAKEHOLDERS

I) Registered Students (In-Campus)
II) Non-Registered Visitors (External-Campus)
III) Faculty
IV) Administrators / Management

MODULE 1: ROLE-BASED LOGIN & USER DETAILS (6 USER STORIES)

User Story 1: Student / Faculty Login

As a registered student or faculty member, I want to log in using institutional credentials so that my access is secure and aligned with my role.

Task:
Implement institutional authentication using unique user identifiers, validate credentials against the user database, identify the user's role (student or faculty), and initiate a secure session upon successful login.

Design:
Authentication layer using encrypted credential storage, role-resolution logic at login time, role-based routing to dashboards, and session/token management with expiration handling.

Testing:
Test valid and invalid credential scenarios, confirm correct role assignment, verify session creation, and ensure unauthorized role escalation is not possible.

User Story 2: Visitor Temporary Registration

As a non-registered visitor, I want to create a temporary account using OTP verification so that I can submit a lost-item request during my campus visit.

Task:
Enable temporary account creation through mobile/email OTP verification, assign visitor role permissions, and enforce automatic account expiry after a predefined duration.

Design:
OTP service integration, temporary user table with expiry timestamps, restricted access control policies, and automated cleanup of expired accounts.

Testing:
Verify OTP validation, confirm visitor access limitations, and ensure account access is revoked automatically after expiry.

User Story 3: Password Reset

As any user, I want to reset my password securely so that I can regain access without administrator support.

Task:
Allow users to initiate password reset requests, verify identity through OTP or email confirmation, and securely update credentials following password strength rules.

Design:
Password reset workflow with tokenized links or OTPs, secure password hashing, validation rules, and rate limiting to prevent abuse.

Testing:
Test reset initiation, token expiry, password update success, and rejection of weak or reused passwords.

User Story 4: Profile Management

As a user, I want to view and update my profile details so that my contact information remains current.

Task:
Provide profile retrieval and update functionality for personal details such as email, phone number, and affiliation.

Design:
Authenticated profile APIs, input validation mechanisms, change confirmation, and audit logging for profile updates.

Testing:
Verify only authenticated users can edit their profiles, confirm data persistence, and ensure changes reflect immediately in notifications.

User Story 5: Request History Access

As a user, I want to view my lost and found request history so that I can track submission outcomes.

Task:
Fetch and display all user-submitted lost and found records with current status, timestamps, and identifiers.

Design:
Paginated history view with filters by status and date, backend queries scoped to user identity.

Testing:
Ensure correct mapping of requests to users, validate accurate status updates, and test pagination performance.

User Story 6: Secure Logout & Admin Login

As a user or administrator, I want secure logout and role-verified admin login so that system access remains protected.

Task:
Terminate active sessions on logout and enforce additional role verification for administrator login.

Design:
Session invalidation mechanisms, role verification middleware, and protected admin routes.

Testing:
Confirm sessions are invalidated after logout and ensure admin features are inaccessible to non-admin users.

User Story 7: Login Activity History

As a registered student, faculty member, or administrator, I want to view my recent login activity so that I can identify any suspicious or unfamiliar access to my account.

- **Task:**
  Record and display recent login events including timestamp, device type, and approximate location.
- **Design:**
  Login activity log linked to user identity, privacy-safe device and location metadata capture, and read-only access via authenticated APIs.
- **Testing:**
  Verify accurate logging of login events, confirm visibility of recent activity only to the account owner, and ensure older records are archived appropriately.

User Story 8: Login Attempt Notification

As a registered student, faculty member, or administrator, I want to be notified after multiple failed login attempts on my account so that I am aware of possible unauthorized access.

- **Task:**
  Track consecutive failed login attempts per account and trigger a security notification to the registered email or phone number.
- **Design:**
  Failed-attempt counter linked to user identity, notification trigger thresholds, and secure alert templates without exposing sensitive details.

- **Testing:**
  Simulate repeated failed logins, verify notification delivery, and ensure alerts are not triggered for normal usage patterns.

User Story 9: Role Visibility Confirmation

As a student, faculty member, or administrator, I want to clearly see my assigned role after login so that I understand my level of access and responsibility.

- **Task:**
  Display authenticated role information within the account or profile section after login.

- **Design:**
  Role-resolution at authentication time, role metadata exposure through secured profile APIs, and read-only role display.

- **Testing:**
  Verify correct role display for each user type and ensure roles cannot be altered by the user.

User Story 10: Campus-Zone Restricted Submission

As the system, I want to restrict item submissions to valid campus zones so that external or invalid locations are rejected.

- **Task:** Validate location against campus boundary data.

- **Design:** Geo-fencing rules integrated with map input.

- **Testing:** Attempt out-of-zone submissions and verify rejection.

MODULE 2: AI-POWERED ITEM RECOVERY & REQUEST PROCESSING (6 USER STORIES)

User Story 1: Search & Browse Items

As a user, I want to search, filter, and sort reported items so that relevant matches are easily discoverable.

Task:
Implement keyword search, category filters, date range selection, and relevance-based sorting.

Design:
Indexed search engine combining textual descriptions and metadata with a responsive UI.

Testing:
Test search accuracy, filter combinations, and ranking consistency under varied datasets.

User Story 2: AI-Assisted Item Submission

As a student, faculty member, or visitor, I want AI-guided item submission so that reports are complete and accurate.

Task:
Collect textual descriptions, images, location, and contextual details using an AI-driven conversational interface.

Design:
Chat-based submission flow with follow-up prompts, image handling, and secure data storage.

Testing:
Verify all modalities are captured, ensure AI prompts trigger correctly, and validate data integrity.

User Story 3: Organization / Club Submissions

As a student or faculty member, I want to submit lost items on behalf of a club or organization so that shared assets can be recovered.

Task:
Allow delegated submissions tagged to organizations and require authorization proof.

Design:
Organization selection UI, authorization document upload, and role validation.

Testing:
Confirm unauthorized submissions are blocked and approved submissions are correctly linked.

User Story 4: Real-Time Similarity Suggestions

As a user, I want immediate AI suggestions of similar items so that potential matches can be claimed early.

Task:
Perform similarity checks during submission against existing records.

Design:
Real-time similarity engine combining text, images, location, and time features.

Testing:
Validate relevance of suggestions and ensure low-latency responses.

User Story 5: Location & Time Capture

As a user, I want to specify where and when the item was lost so that spatial-temporal matching improves accuracy.

Task:
Capture map-based location input and precise time metadata.

Design:
Interactive campus map with validation of permitted zones and time pickers.

Testing:
Confirm accurate coordinate storage and retrievability.

User Story 6: Tracking ID Generation

As a user, I want a unique tracking ID so that I can monitor my request.

Task:
Generate and associate a unique identifier with every submission.

Design:
UUID-based ID generator linked to database records.

Testing:
Verify uniqueness, persistence, and traceability across workflows.

User Story 7: Mandatory Item Attribute Validation

As a user, I want the system to validate essential item attributes during submission so that incomplete or ambiguous reports are avoided.

- **Task:**
  Enforce completion of mandatory fields (item type, color, material, approximate size) before allowing submission.

- **Design:**
  Rule-based validation layer integrated with the submission flow, adaptive prompts for missing attributes, and structured attribute schema.

- **Testing:**
  Attempt submissions with missing or vague attributes and verify that users are prompted to complete required information before proceeding.

User Story 8: Draft Saving of Submissions

As a student, faculty member, or visitor, I want to save a submission as a draft so that I can complete it later without losing partially entered information.

- **Task:**
  Enable draft persistence for incomplete submissions and allow users to retrieve, edit, and finalize drafts at a later time.

- **Design:**
  Introduce a draft state within the submission lifecycle, with autosave or manual save options, timestamped storage, and clear differentiation between draft and finalized submissions.

- **Testing:**
  Verify that drafts are correctly saved and restored across sessions, remain editable until final submission, and are not processed or visible in active workflows until explicitly submitted.

User Story 9: Found-Item Anonymous Reporting

As a user, I want to report found items anonymously so that I can contribute without exposing my identity or personal details to other users.

- **Task:**
  Provide an option to submit found-item reports anonymously while maintaining internal traceability for administrative purposes.

- **Design:**
  Implement an anonymized public-facing report view that hides user identity, paired with secure internal linkage accessible only to authorized administrators.

- **Testing:**
  Confirm that anonymous reports do not display identifying information in user-facing interfaces, while administrators can still access the associated user record when required.

User Story 10: Campus Location Policy Configuration

As an administrator, I want to define valid campus submission zones so that item reports are limited to authorized locations.

- **Task:**
  Configure permissible campus zones and associate them with submission validation rules.
- **Design:**
  Admin-configurable geo-boundary management interface, zone metadata storage, and integration with location input validation.
- **Testing:**
  Verify that submissions outside configured zones are rejected and that updates to zone definitions take effect immediately.

MODULE 3: CLAIM VERIFICATION & RESOLUTION (6 USER STORIES)

User Story 1: AI Match Generation

As an administrator, I want automated AI matching so that claim processing is accelerated.

Task:
Compute similarity scores across lost and found datasets.

Design:
Multi-modal AI pipeline with weighted scoring.

Testing:
Evaluate match precision and recall.

User Story 2: Match Explainability

As a user, I want explanations for matches so that trust in AI is established.

Task:
Expose contributing similarity features.

Design:
Explainability layer highlighting shared attributes.

Testing:
Ensure explanations align with AI outputs.

User Story 3: Manual Override

As an administrator, I want to override AI matches with justification so that errors are corrected.

Task:
Allow manual decision-making and logging.

Design:
Override interface with mandatory remarks.

Testing:
Confirm logs and updated statuses.

User Story 4: Learning from Rejections

As the system, I want to learn from rejected matches so that AI improves.

Task:
Ingest feedback into training data.

Design:
Feedback loop and model update workflow.

Testing:
Measure reduction in repeated mismatches.

User Story 5: Fraud Detection

As the system, I want to detect suspicious claims so that misuse is prevented.

Task:
Analyze duplicate and abnormal claim patterns.

Design:
Risk-scoring engine.

Testing:
Simulate fraud scenarios.

User Story 6: Claim Comparison

As an administrator, I want side-by-side claim comparison so that fair decisions are made.

Task:
Display claim evidence together.

Design:
Comparison dashboard.

Testing:
Verify accuracy and usability.

User Story 7: Ownership Evidence Scoring

As the system, I want to score ownership proofs so that weak or incomplete claims are flagged early.
Task: Assign confidence scores to uploaded proofs using rules and ML-assisted analysis.
Design: Scoring engine integrated with claim dashboard to highlight weak proofs.
Testing: Validate score accuracy and ensure weak proofs are flagged appropriately.

User Story 8: Multi-Claim Conflict Detection

As an administrator, I want alerts when multiple users claim the same item so that conflicts are handled carefully.
Task: Detect overlapping claims and notify administrators.
Design: Conflict detection rules with alert integration in the admin dashboard.
Testing: Simulate multiple claims and verify alerts are generated.

User Story 9: Claim Withdrawal

As a user, I want to withdraw my claim if it was submitted in error so that system integrity is maintained.
Task: Allow claim withdrawal with confirmation and update status.
Design: State rollback with audit logging and exclusion from AI matching.
Testing: Verify withdrawn claims are removed and actions logged.

User Story 10: Partial Match Handling

As an administrator, I want to review partial matches separately so that borderline cases are not auto-rejected.
Task: Categorize matches below full confidence for manual review.
Design: Tiered confidence buckets on the admin dashboard for easy review.
Testing: Ensure partial matches are routed correctly and not auto-processed.

MODULE 4: CENTRAL AUTHORITY & AI GOVERNANCE

User Story 1: Claim Approval & Official Letter Issuance

As an administrator, I want to approve or reject ownership claims and issue official confirmation letters so that only verified owners recover items.

Task:
Review submitted claims, validate ownership proofs, record approval or rejection decisions, and generate an official confirmation letter for approved claims.

Design:
Claim review workflow with approval states, digital signature support, and automated PDF letter generation linked to the claim record.

Testing:
Verify only authorized administrators can approve claims, confirm correct status updates, and validate accuracy and format of generated letters.

User Story 2: AI Parameter & Confidence Threshold Configuration

As an administrator, I want to configure AI confidence thresholds so that automated matching aligns with institutional risk tolerance.

Task:
Allow administrators to define similarity thresholds, adjust weightage of text, image, location, and time features, and enable or disable AI-assisted decisions.

Design:
AI governance interface with configurable parameters, validation rules, and versioned configuration storage.

Testing:
Confirm AI behavior changes after configuration updates and ensure invalid parameter values are rejected.

User Story 3: AI Performance Monitoring

As an administrator, I want to monitor AI performance metrics so that the system's reliability and effectiveness can be evaluated.

Task:
Collect metrics such as match accuracy, override frequency, claim resolution time, and fraud detection rates.

Design:
Analytics dashboard with charts, trend analysis, and time-based filters.

Testing:
Verify metrics accuracy, real-time updates, and consistency with underlying data.

User Story 4: Role & Permission Management

As an administrator, I want to manage user roles and permissions so that system access remains secure and controlled.

Task:
Assign, update, or revoke roles for students, faculty, visitors, and administrators.

Design:
Role-Based Access Control (RBAC) system with permission matrices and change audit logs.

Testing:
Ensure permissions are enforced correctly and role changes take effect immediately.

User Story 5: Audit Logs & Governance Reports

As an administrator, I want access to audit logs and governance reports so that all system actions are traceable.

Task:
Record all critical actions including logins, claim approvals, overrides, and role changes, and generate summarized reports.

Design:
Centralized logging framework with report generation and export capabilities.

Testing:
Validate completeness of logs, accuracy of reports, and restricted access to sensitive data.

User Story 6: Archival of Resolved Claims

As the system, I want to archive resolved claims so that historical data is preserved without impacting active system performance.

Task:
Identify resolved claims, migrate them to archive storage, and maintain retrievability for audits.

Design:
Archival data store with indexing and access control.

Testing:
Verify archived claims remain accessible and do not appear in active workflows.

User Story 7: Scoped Administrative Delegation

As a senior administrator,
I want to delegate *limited and time-bound administrative privileges* to designated staff so that operational workload can be shared without compromising system security or authority.

Task:
Enable assignment of scoped administrative permissions (e.g., review-only, approval-only, monitoring-only) with defined validity periods.

Design:
Hierarchical RBAC with delegation scopes, expiry timestamps, and mandatory audit logging for all delegated actions.

Testing:
Verify delegated admins cannot exceed assigned scopes, confirm automatic revocation after expiry, and ensure all delegated actions are traceable.

User Story 8: AI Decision Versioning & Rollback

As an administrator,
I want to revert AI-assisted decisions to prior validated states so that incorrect or disputed automated outcomes can be corrected transparently.

Task:
Maintain version history of AI-driven decisions and allow controlled rollback to a selected previous state.

Design:
Decision versioning framework with immutable logs, rollback authorization checks, and linkage to justification records.

Testing:
Verify rollback restores correct system state, confirm no data corruption occurs, and ensure rollbacks are logged and auditable.

User Story 9: Data Retention Policy Enforcement

As the system, I want to enforce data retention limits automatically so that compliance requirements and institutional policies are consistently met.

Task: Auto-delete or anonymize records that have exceeded their retention period, based on configured policies.
Design: Retention scheduler with policy rules, automated execution, and audit logging for all deleted or anonymized data.
Testing: Verify that expired records are correctly processed, retention policies are respected, and audit logs accurately record all actions.

User Story 10: Critical Action Multi-Level Approval

As institutional management,
I want high-impact actions to require multi-level approval so that accountability and oversight are enforced for sensitive operations.

Task:
Enforce dual or multi-step approvals for critical actions such as claim approvals, AI overrides, data deletion, or archival triggers.

Design:
Configurable approval chains with role-based approvers, escalation rules, and execution blocking until all approvals are completed.

Testing:
Ensure restricted actions cannot proceed without full approval, validate escalation behavior, and confirm approval trails are recorded.

MODULE 5: NOTIFICATION & COMMUNICATION

User Story 1: Match Notifications

As a user, I want to receive notifications when matching items are identified so that I can act promptly.

Task:
Trigger notifications based on AI match events and user preferences.

Design:
Notification service supporting email and push alerts with message templates.

Testing:
Confirm notifications are delivered accurately and on time.

User Story 2: Claim Status Updates

As a user, I want to receive updates when my claim status changes so that I remain informed throughout the process.

Task:
Monitor claim state transitions and send corresponding notifications.

Design:
Event-driven notification triggers integrated with claim workflow.

Testing:
Validate notifications reflect correct status and timestamps.

User Story 3: Duplicate Notification Prevention

As the system, I want to prevent duplicate notifications so that users are not confused by repeated messages.

Task:
Track notification history and suppress duplicates.

Design:
Notification log with de-duplication checks before dispatch.

Testing:
Simulate repeated events and verify only one notification is sent.

User Story 4: Proof Request Communication

As a user, I want clear instructions when additional proof is required so that I can respond correctly and on time.

Task:
Generate structured proof requests specifying required documents and deadlines.

Design:
Template-based messaging with dynamic content insertion.

Testing:
Confirm clarity, correctness, and actionability of proof requests.

User Story 5: Secure Administrator Messaging

As an administrator, I want to communicate securely with users so that sensitive information is protected and traceable.

Task:
Enable one-way or controlled messaging between administrators and users.

Design:
Secure messaging interface with encryption and audit logging.

Testing:
Verify message confidentiality and completeness of communication logs.

User Story 6: Escalation of Unresolved Notifications

As the system, I want to escalate unresolved notifications and proof requests so that claim resolution delays are minimized.

Task:
Track deadlines and trigger escalation alerts to higher authorities when deadlines lapse.

Design:
Escalation rules engine integrated with notification workflows.

Testing:
Simulate overdue cases and confirm escalation notifications are triggered correctly.

User Story 7: Notification Preference Management

As a user, I want to control how I receive notifications and which channels are used so that communication aligns with my personal preferences and I am not overwhelmed.

Task: Allow users to opt in or out of specific notification channels (email, push, SMS) and set priority preferences.
Design: Preference store linked to the notification service, with dynamic application of preferences during message dispatch.
Testing: Confirm that notifications respect user choices across all channels and updates to preferences take effect immediately.

User Story 8: Read Receipt Tracking

As an administrator, I want to track whether users have read critical messages so that follow-ups can be appropriately planned and communication gaps minimized.

Task: Maintain read/unread status for all messages, including timestamps and user identifiers.
Design: Read-receipt metadata stored in a centralized system, integrated with notification logs for traceability.
Testing: Validate read/unread state updates, ensure reports reflect accurate read status, and confirm administrators can access this information securely.

User Story 9: Scheduled Reminder Notifications

As the system, I want to send reminder notifications ahead of deadlines so that users are prompted to complete actions on time and claim processing remains efficient.

Task: Schedule reminders for pending actions with configurable lead times and recurrence options.
Design: Time-based notification scheduler integrated with the claim workflow, supporting multiple channels and priority settings.
Testing: Confirm reminders trigger at correct times, are sent via preferred channels, and repeated reminders are handled correctly.

User Story 10: System-Wide Announcements

As administrators, we want to broadcast announcements to all relevant users so that campus-wide lost-and-found updates or emergency alerts are communicated effectively.

Task: Send targeted broadcast notifications by user role or location while ensuring proper formatting and delivery tracking.
Design: Announcement service with role-based and location-based targeting, message templates, and delivery logging.
Testing: Verify correct audience targeting, message formatting consistency, and that delivery reports reflect accurate distribution.