Project Review Presentation — Lost & Found Application (Team 15)

1. Project Overview

The Lost & Found Application is a web-based system designed for students and staff at a university. Staff members can upload lost items into the system, while students can browse available items and submit claims.
The system includes a staff dashboard for managing items and claims, and enforces role-based access control.

---

2. Target Users & Roles

- **Students**
  - Register using a university email address
  - Browse lost items
  - Submit claims for items
- **Staff**
  - Upload and manage lost items
  - Review and process claims via a staff dashboard
- **Access Control**
  - Only authenticated users can view item listings
  - Staff-only permissions for item creation and claim management

---

3. Core Features Implemented

- Account creation and login using Flask session-based authentication
- Role-based access control (student vs staff)
- Lost item creation, update, and deletion (staff)
- Item browsing with search and filtering

- Claim submission and claim status tracking
- Email notifications for claim-related events
- Administrative staff dashboard

---

4. Technology Stack

- **Frontend:** React
- **Backend:** Flask (Python)
- **Database:** SQLite
- **Authentication:** Flask sessions
- **Testing:** pytest
- **Email:** smtplib (SMTP)
- **Deployment:** Self-hosted servers

---

5. High-Level Architecture

- React single-page application communicates with Flask backend via HTTP.
- Flask handles routing, business logic, authentication, and authorization.
- SQLite stores persistent data for users, items, and claims.
- Session cookies are used to maintain authenticated state.
- SMTP email service sends notifications on claim events.

---

6. Database Schema (Implemented)

Users Table

Stores all registered users (students and staff).

```
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    email TEXT UNIQUE NOT NULL,
    name TEXT NOT NULL,
    password_hash TEXT NOT NULL,
    role TEXT NOT NULL CHECK(role IN ('student', 'staff')),
    watcard_number TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    last_login TIMESTAMP
);
```

Items Table

Stores all reported lost items.

```
CREATE TABLE items (
    item_id INTEGER PRIMARY KEY AUTOINCREMENT,
    description TEXT,
    category TEXT NOT NULL,
    location_found TEXT NOT NULL,
    pickup_at TEXT NOT NULL CHECK(pickup_at IN ('SLC', 'PAC', 'CIF')),
    date_found TIMESTAMP NOT NULL,
    status TEXT NOT NULL DEFAULT 'unclaimed'
        CHECK(status IN ('unclaimed', 'claimed', 'deleted')),
    image_url TEXT,
    found_by_desk TEXT NOT NULL,
    created_by_user_id INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    claimed_at TIMESTAMP,
    FOREIGN KEY (created_by_user_id) REFERENCES users(user_id)
);
```

---

7. Sprint-by-Sprint Review

Sprint 1 — Foundations

- Selected technology stack (Flask + React)
- Set up repository, project structure, and authentication design
- Created initial documentation
- **Challenge:** Tasks were too large and documentation lagged

Sprint 2 — Authentication & Integration

- Implemented login and signup functionality
- Connected frontend and backend
- Improved code organization

- **Improvement:** Better sprint planning and issue tracking

Sprint 3 — Core Features

- Implemented Lost Items page
- Added staff portal functionality
- Stabilized frontend–backend synchronization
- **Challenge:** Some tasks still required mid-sprint splitting

Sprint 4 — Refinement & Documentation

- UI improvements and authentication fixes
- Added user manual and technical documentation
- Improved sprint tracking consistency
- **Challenge:** Minor merge conflicts and underestimated tasks

Sprint 5 — Finalization

- Completed remaining features and bug fixes
- Conducted QA testing
- Created demo video and presentation materials
- **Challenge:** Limited time for performance optimization

---

8. Testing & Quality Assurance

- pytest used for backend unit and integration tests
- Authentication flows and core endpoints tested
- Testing started later than ideal
- **Recommendation:** Integrate automated testing into CI pipeline

---

9. Deployment & Operations

- Application deployed on self-managed servers
- SQLite database used with file-level persistence
- Manual monitoring of server health and logs
- **Future Improvement:** Add automated backups and monitoring

---

10. Security & Privacy Considerations

- University email requirement reduces misuse
- Session cookies should be configured with Secure and HttpOnly flags
- Input validation to prevent injection attacks
- Minimal storage of personally identifiable information

---

11. Outcomes & Lessons Learned

- Delivered a feature-complete lost-and-found system
- Improved sprint planning and communication over time
- Learned importance of early testing and smaller user stories

---

12. Future Work

- Add CI pipeline for testing and deployment
- Migrate to PostgreSQL for scalability
- Improve search with full-text indexing
- Collect user feedback and usage metrics