■ Lost and Found Project Code Export

Full-stack Lost and Found application with React frontend and Flask backend

■ Project File Structure:

■ Frontend Files (React):

- postcss.config.js
- public\index.html
- src\App.css
- src\App.js
- src\App.test.js
- src\components\AdminTable.js
- src\components\ClaimsTable.js
- src\components\FeedbackForm.js
- src\components\FlashMessage.js
- src\components\Header.js
- src\components\ItemTable.js
- src\components\MessageList.js
- src\index.css
- src\index.js
- src\pages\AdminPage.js
- src\pages\HomePage.js
- src\pages\ItemRegisterPage.js
- src\pages\ItemsPage.js
- src\pages\LoginPage.js
- src\pages\RegisterPage.js
- src\reportWebVitals.js
- src\setupTests.js
- tailwind.config.js

■ Backend Files (Flask/Python):

- backend\app.py
- hash.py
- script.py

■■ Configuration Files:

- .gitignore
- lost_and_found.sql

■ Other Files:

- Lost_and_Found_Code_Export.pdf

■ File: .gitignore

```
================================================================================
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

■ File: Lost_and_Found_Code_Export.pdf

```
================================================================================
[Binary file - .pdf format]
```

■ File: backend\app.py

```
================================================================================
  1: from flask import Flask, request, jsonify, session
  2: from flask_bcrypt import Bcrypt
  3: from flask_cors import CORS
  4: import mysql.connector
  5: from datetime import datetime, timedelta
  6: import os
  7: from apscheduler.schedulers.background import BackgroundScheduler
  8: import atexit
  9: from werkzeug.utils import secure_filename
 10: import uuid
 11:
 12: app = Flask(__name__)
 13:
 14: app.config["SESSION_COOKIE_SAMESITE"] = "Lax"
 15: app.config["SESSION_COOKIE_SECURE"] = False
 16:
 17: app.secret_key = "your_secret_key"
 18: bcrypt = Bcrypt(app)
 19:
 20: UPLOAD_FOLDER = 'static/uploads'
 21: ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}
 22:
 23: app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
 24:
 25: # Ensure upload folder exists
 26: os.makedirs(UPLOAD_FOLDER, exist_ok=True)
 27:
 28: def allowed_file(filename):
 29:     return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
 30:
 31: # Database connection
 32: def get_db_connection():
 33:     return mysql.connector.connect(
 34:         host="localhost",
 35:         user="root",
 36:         password="Mysql@123",
 37:         database="lost_and_founds"
```

```
    )

# Enable CORS on your Flask app
CORS(app, supports_credentials=True)

# Authentication decorator
def login_required(f):
    def decorated_function(*args, **kwargs):
        if "user_id" not in session:
            return jsonify({"error": "Unauthorized"}), 401
        return f(*args, **kwargs)
    decorated_function.__name__ = f.__name__
    return decorated_function

def admin_required(f):
    def decorated_function(*args, **kwargs):
        if "user_id" not in session or session.get("role") != "admin":
            return jsonify({"error": "Admin access required"}), 403
        return f(*args, **kwargs)
    decorated_function.__name__ = f.__name__
    return decorated_function

# API Routes
@app.route("/api/login", methods=["POST"])
def login():
    data = request.get_json()
    email = data.get("email")
    password = data.get("password")

    if not email or not password:
        return jsonify({"success": False, "error": "Email and password are required"}), 400

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    try:
        cursor.execute("SELECT * FROM users WHERE email = %s", (email,))
        user = cursor.fetchone()

        if user and bcrypt.check_password_hash(user["password"], password):
            session["user_id"] = user["id"]
            session["name"] = user["name"]
            session["role"] = user["role"]
            session["email"] = user["email"]
            return jsonify({
                "success": True,
                "user": {
                    "id": user["id"],
                    "name": user["name"],
                    "email": user["email"],
                    "role": user["role"]
                }
            })
        else:
            return jsonify({"success": False, "error": "Invalid email or password"}), 401
    except Exception as e:
        return jsonify({"success": False, "error": "Database error"}), 500
    finally:
        conn.close()

@app.route("/api/register", methods=["POST"])
def register():
    data = request.get_json()
    name = data.get("name")
    email = data.get("email")
    password = data.get("password")
    confirm_password = data.get("confirm_password")

    if not all([name, email, password, confirm_password]):
        return jsonify({"success": False, "error": "All fields are required"}), 400

    if password != confirm_password:
        return jsonify({"success": False, "error": "Passwords do not match"}), 400
```

```
111: 
112:     hashed_password = bcrypt.generate_password_hash(password).decode("utf-8")
113:     conn = get_db_connection()
114:     cursor = conn.cursor()
115: 
116:     try:
117:         cursor.execute("INSERT INTO users (name, email, password, role) VALUES (%s, %s, %s, %s)",
118:                        (name, email, hashed_password, "user"))
119:         conn.commit()
120:         return jsonify({"success": True, "message": "Registration successful"})
121:     except mysql.connector.Error as err:
122:         if err.errno == 1062:  # Duplicate entry
123:             return jsonify({"success": False, "error": "Email already registered"}), 409
124:         return jsonify({"success": False, "error": "Database error"}), 500
125:     finally:
126:         conn.close()
127: 
128: @app.route("/api/logout", methods=["POST"])
129: def logout():
130:     session.clear()
131:     return jsonify({"success": True})
132: 
133: @app.route("/api/user", methods=["GET"])
134: @login_required
135: def get_user():
136:     return jsonify({
137:         "id": session["user_id"],
138:         "name": session["name"],
139:         "email": session["email"],
140:         "role": session["role"]
141:     })
142: 
143: @app.route("/api/dashboard", methods=["GET"])
144: @login_required
145: def dashboard():
146:     conn = get_db_connection()
147:     cursor = conn.cursor(dictionary=True)
148: 
149:     try:
150:         # Get user's items
151:         cursor.execute("SELECT * FROM items WHERE created_by = %s ORDER BY created_at DESC", (session["user
152:         user_items = cursor.fetchall()
153: 
154:         # Get user's claims
155:         cursor.execute("""
156:             SELECT claims.*, items.name as item_name, items.description, items.location
157:             FROM claims
158:             JOIN items ON claims.item_id = items.id
159:             WHERE claims.claimed_by = %s
160:             ORDER BY claims.claimed_at DESC
161:         """, (session["user_id"],))
162:         user_claims = cursor.fetchall()
163: 
164:         # Get user's messages
165:         cursor.execute("""
166:             SELECT messages.*, users.name as sender_name, items.name as item_name
167:             FROM messages
168:             JOIN users ON messages.sender_id = users.id
169:             JOIN items ON messages.item_id = items.id
170:             WHERE messages.receiver_id = %s
171:             ORDER BY sent_at DESC
172:         """, (session["user_id"],))
173:         messages = cursor.fetchall()
174: 
175:         return jsonify({
176:             "user": {
177:                 "name": session["name"],
178:                 "email": session["email"],
179:                 "role": session["role"]
180:             },
181:             "items": user_items,
182:             "claims": user_claims,
183:             "messages": messages
```

```
        })
    except Exception as e:
        return jsonify({"error": "Database error"}), 500
    finally:
        conn.close()

@app.route("/api/items", methods=["GET"])
@login_required
def get_items():
    search_query = request.args.get("search", "")
    status_filter = request.args.get("status", "")

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    try:
        # Building the SQL query
        query = """
            SELECT items.*, users.name AS creator_name
            FROM items
            LEFT JOIN users ON items.created_by = users.id
            WHERE 1=1
        """
        params = []

        if search_query:
            query += " AND (items.name LIKE %s OR items.description LIKE %s OR items.location LIKE %s)"
            params.extend(['%' + search_query + '%', '%' + search_query + '%', '%' + search_query + '%'])

        if status_filter:
            query += " AND items.status = %s"
            params.append(status_filter)

        query += " ORDER BY items.created_at DESC"

        cursor.execute(query, params)
        items = cursor.fetchall()

        return jsonify({"items": items})
    except Exception as e:
        return jsonify({"error": "Database error"}), 500
    finally:
        conn.close()

@app.route("/api/items", methods=["POST"])
@login_required
def create_item():
    try:
        name = request.form.get("name")
        description = request.form.get("description")
        location = request.form.get("location")
        status = request.form.get("status")

        if not all([name, description, location, status]):
            return jsonify({"error": "All fields are required"}), 400

        image_filename = None
        if 'image' in request.files:
            image = request.files['image']
            if image and allowed_file(image.filename):
                filename = secure_filename(image.filename)
                unique_filename = str(uuid.uuid4()) + "_" + filename
                image_path = os.path.join(app.config['UPLOAD_FOLDER'], unique_filename)
                image.save(image_path)
                image_filename = f"uploads/{unique_filename}"

        conn = get_db_connection()
        cursor = conn.cursor()

        cursor.execute("""
            INSERT INTO items (name, description, location, status, image, created_by, created_at)
            VALUES (%s, %s, %s, %s, %s, %s, NOW())
        """, (name, description, location, status, image_filename, session["user_id"]))
```

```
257:
258:         conn.commit()
259:         item_id = cursor.lastrowid
260:
261:         return jsonify({"success": True, "item_id": item_id, "message": "Item registered successfully"})
262:     except Exception as e:
263:         return jsonify({"error": "Failed to register item"}), 500
264:     finally:
265:         conn.close()
266:
267: @app.route("/api/items/<int:item_id>/claim", methods=["POST"])
268: @login_required
269: def claim_item(item_id):
270:     conn = get_db_connection()
271:     cursor = conn.cursor()
272:
273:     try:
274:         # Check if item exists and is not already claimed
275:         cursor.execute("SELECT * FROM items WHERE id = %s", (item_id,))
276:         item = cursor.fetchone()
277:
278:         if not item:
279:             return jsonify({"error": "Item not found"}), 404
280:
281:         if item[5] == 'claimed':  # Assuming status is at index 5
282:             return jsonify({"error": "Item already claimed"}), 400
283:
284:         # Check if user already claimed this item
285:         cursor.execute("SELECT * FROM claims WHERE item_id = %s AND claimed_by = %s", (item_id, session["us
286:         existing_claim = cursor.fetchone()
287:
288:         if existing_claim:
289:             return jsonify({"error": "You have already claimed this item"}), 400
290:
291:         # Insert the claim
292:         cursor.execute("""
293:             INSERT INTO claims (item_id, claimed_by, claimant_name, claimant_email, claimed_at)
294:             VALUES (%s, %s, %s, %s, NOW())
295:         """, (item_id, session["user_id"], session["name"], session["email"]))
296:
297:         # Update the item's status to 'claimed'
298:         cursor.execute("UPDATE items SET status = 'claimed' WHERE id = %s", (item_id,))
299:
300:         conn.commit()
301:         return jsonify({"success": True, "message": "Item claimed successfully"})
302:     except Exception as e:
303:         return jsonify({"error": "Failed to claim item"}), 500
304:     finally:
305:         conn.close()
306:
307: @app.route("/api/messages", methods=["POST"])
308: @login_required
309: def send_message():
310:     data = request.get_json()
311:     receiver_id = data.get("receiver_id")
312:     item_id = data.get("item_id")
313:     message_text = data.get("message")
314:
315:     if not all([receiver_id, item_id, message_text]):
316:         return jsonify({"error": "All fields are required"}), 400
317:
318:     conn = get_db_connection()
319:     cursor = conn.cursor()
320:
321:     try:
322:         cursor.execute("""
323:             INSERT INTO messages (sender_id, receiver_id, item_id, message, sent_at)
324:             VALUES (%s, %s, %s, %s, NOW())
325:         """, (session["user_id"], receiver_id, item_id, message_text))
326:
327:         conn.commit()
328:         return jsonify({"success": True, "message": "Message sent successfully"})
329:     except Exception as e:
```

```
        return jsonify({"error": "Failed to send message"}), 500
    finally:
        conn.close()

@app.route("/api/reply", methods=["POST"])
def api_reply_message():
    if "user_id" not in session:
        return jsonify({"error": "Please log in to send messages."}), 401

    data = request.get_json()
    receiver_id = data.get("receiver_id")
    item_id = data.get("item_id")
    reply_text = data.get("message")

    if not all([receiver_id, item_id, reply_text]):
        return jsonify({"error": "All fields are required."}), 400

    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        cursor.execute(
            """
            INSERT INTO messages (sender_id, receiver_id, item_id, message, sent_at)
            VALUES (%s, %s, %s, %s, NOW())
            """,
            (session["user_id"], receiver_id, item_id, reply_text),
        )
        conn.commit()
        return jsonify({"success": True, "message": "Reply sent successfully!"})
    except mysql.connector.Error:
        return jsonify({"error": "Error sending reply."}), 500
    finally:
        conn.close()



@app.route("/api/feedback", methods=["POST"])
@login_required
def submit_feedback():
    data = request.get_json()
    feedback_text = data.get("feedback")

    if not feedback_text:
        return jsonify({"error": "Feedback text is required"}), 400

    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        cursor.execute("""
            INSERT INTO feedback (user_id, feedback_text, submitted_at)
            VALUES (%s, %s, NOW())
        """, (session["user_id"], feedback_text))

        conn.commit()
        return jsonify({"success": True, "message": "Thank you for your feedback!"})
    except Exception as e:
        return jsonify({"error": "Failed to submit feedback"}), 500
    finally:
        conn.close()

@app.route("/api/admin/feedback/<int:feedback_id>", methods=["DELETE"])
@admin_required
def delete_feedback(feedback_id):
    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute("DELETE FROM feedback WHERE id = %s", (feedback_id,))
        conn.commit()
        return jsonify({"success": True, "message": "Feedback deleted successfully"})
    except Exception as e:
        print("Error deleting feedback:", e)
```

```
        return jsonify({"error": "Failed to delete feedback"}), 500
    finally:
        conn.close()

# Admin routes
@app.route("/api/admin/dashboard", methods=["GET"])
@admin_required
def admin_dashboard():
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    try:
        # Get all items
        cursor.execute("""
            SELECT items.*, users.name as creator_name
            FROM items
            JOIN users ON items.created_by = users.id
            ORDER BY items.created_at DESC
        """)
        items = cursor.fetchall()

        # Get all claims
        cursor.execute("""
            SELECT claims.*, items.name as item_name, users.name as claimer_name
            FROM claims
            JOIN items ON claims.item_id = items.id
            JOIN users ON claims.claimed_by = users.id
            ORDER BY claims.claimed_at DESC
        """)
        claims = cursor.fetchall()

        # Get all users
        cursor.execute("SELECT id, name, email, role FROM users WHERE role != 'admin'")
        users = cursor.fetchall()

        # Get all feedback (FIXED)
        cursor.execute("""
            SELECT feedback.*, users.name as user_name
            FROM feedback
            JOIN users ON feedback.user_id = users.id
            ORDER BY feedback.submitted_at DESC
        """)
        feedback = cursor.fetchall()

        return jsonify({
            "items": items,
            "claims": claims,
            "users": users,
            "feedback": feedback
        })
    except Exception as e:
        print("Error in admin_dashboard:", e)
        return jsonify({"error": "Database error"}), 500
    finally:
        conn.close()

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    try:
        # Get all items
        cursor.execute("""
            SELECT items.*, users.name as creator_name
            FROM items
            JOIN users ON items.created_by = users.id
            ORDER BY items.created_at DESC
        """)
        items = cursor.fetchall()

        # Get all claims
        cursor.execute("""
            SELECT claims.*, items.name as item_name, users.name as claimer_name
            FROM claims
```

```
            JOIN items ON claims.item_id = items.id
            JOIN users ON claims.claimed_by = users.id
            ORDER BY claims.claimed_at DESC
        """)
        claims = cursor.fetchall()

        # Get all users
        cursor.execute("SELECT id, name, email, role, created_at FROM users WHERE role != 'admin'")
        users = cursor.fetchall()

        # Get all feedback
        cursor.execute("""
            SELECT feedback.*, users.name as user_name
            FROM feedback
            JOIN users ON feedback.user_id = users.id
            ORDER BY feedback.created_at DESC
        """)
        feedback = cursor.fetchall()

        return jsonify({
            "items": items,
            "claims": claims,
            "users": users,
            "feedback": feedback
        })
    except Exception as e:
        return jsonify({"error": "Database error"}), 500
    finally:
        conn.close()

@app.route("/api/admin/items/<int:item_id>", methods=["DELETE"])
@admin_required
def delete_item(item_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # First delete any claims associated with the item
        cursor.execute("DELETE FROM claims WHERE item_id = %s", (item_id,))

        # Delete any messages associated with the item
        cursor.execute("DELETE FROM messages WHERE item_id = %s", (item_id,))

        # Then delete the item from the items table
        cursor.execute("DELETE FROM items WHERE id = %s", (item_id,))

        conn.commit()
        return jsonify({"success": True, "message": "Item deleted successfully"})
    except Exception as e:
        return jsonify({"error": "Failed to delete item"}), 500
    finally:
        conn.close()

@app.route("/api/admin/users/<int:user_id>", methods=["DELETE"])
@admin_required
def delete_user(user_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Delete claims associated with the user
        cursor.execute("DELETE FROM claims WHERE claimed_by = %s", (user_id,))

        # Delete messages sent by the user
        cursor.execute("DELETE FROM messages WHERE sender_id = %s OR receiver_id = %s", (user_id, user_id))

        # Delete feedback by the user
        cursor.execute("DELETE FROM feedback WHERE user_id = %s", (user_id,))

        # Delete items created by the user
        cursor.execute("DELETE FROM items WHERE created_by = %s", (user_id,))

        # Delete the user
```

```
        cursor.execute("DELETE FROM users WHERE id = %s", (user_id,))

        conn.commit()
        return jsonify({"success": True, "message": "User deleted successfully"})
    except Exception as e:
        return jsonify({"error": "Failed to delete user"}), 500
    finally:
        conn.close()

@app.route("/api/admin/claims/<int:claim_id>", methods=["DELETE"])
@admin_required
def delete_claim(claim_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Step 1: Get item_id for the claim
        cursor.execute("SELECT item_id FROM claims WHERE id = %s", (claim_id,))
        result = cursor.fetchone()

        if not result:
            return jsonify({"error": "Claim not found"}), 404

        item_id = result[0]

        print(f"?? Deleting claim_id={claim_id}, linked item_id={item_id}")

        # Step 2: Delete the claim
        cursor.execute("DELETE FROM claims WHERE id = %s", (claim_id,))

        # Step 3: Only update item status if it was 'claimed'
        cursor.execute(
            "UPDATE items SET status = 'found' WHERE id = %s AND status = 'claimed'",
            (item_id,)
        )

        conn.commit()
        return jsonify({"success": True, "message": "Claim deleted and item status updated"})

    except Exception as e:
        import traceback
        traceback.print_exc()  # Log full error to terminal
        return jsonify({"error": str(e)}), 500
    finally:
        conn.close()


# Background job for cleaning old claims
def delete_old_claims():
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Delete claims older than 7 days
        cursor.execute("""
            DELETE FROM claims WHERE claimed_at < %s
        """, (datetime.now() - timedelta(days=7),))
        claims_deleted = cursor.rowcount

        # Update items status back to available if no active claims
        cursor.execute("""
            UPDATE items
            SET status = 'available'
            WHERE status = 'claimed' AND id NOT IN (SELECT item_id FROM claims)
        """)
        items_updated = cursor.rowcount

        conn.commit()
        print(f"Deleted {claims_deleted} old claims and updated {items_updated} items.")
    except mysql.connector.Error as err:
        print(f"Error during cleanup: {err}")
    finally:
        conn.close()
```

```
622:
623: # Initialize the scheduler
624: scheduler = BackgroundScheduler()
625: scheduler.add_job(func=delete_old_claims, trigger="interval", hours=24)  # Run daily
626: scheduler.start()
627:
628: # Ensure the scheduler shuts down properly when the app exits
629: atexit.register(lambda: scheduler.shutdown())
630:
631: # Error handlers
632: @app.errorhandler(404)
633: def not_found(error):
634:     return jsonify({"error": "Not found"}), 404
635:
636: @app.errorhandler(500)
637: def internal_error(error):
638:     return jsonify({"error": "Internal server error"}), 500
639:
640: if __name__ == "__main__":
641:     app.run(debug=True, port=5000)
```

■ File: hash.py

```
  1: import bcrypt
  2:
  3: # Normal password
  4: password = input("Admin@123").encode('utf-8')
  5:
  6: # Generate a salt
  7: salt = bcrypt.gensalt()
  8:
  9: # Hash the password
 10: hashed_password = bcrypt.hashpw(password, salt)
 11:
 12: print("Bcrypt Hash:", hashed_password.decode('utf-8'))
```

■■ File: lost_and_found.sql

```
  1: -- Drop and create the database
  2: DROP DATABASE IF EXISTS lost_and_founds;
  3: CREATE DATABASE lost_and_founds;
  4: USE lost_and_founds;
  5:
  6: -- Users Table
  7: CREATE TABLE users (
  8:     id INT AUTO_INCREMENT PRIMARY KEY,
  9:     name VARCHAR(255) NOT NULL,
 10:     email VARCHAR(255) NOT NULL UNIQUE,
 11:     password VARCHAR(255) NOT NULL,
 12:     role ENUM('user', 'admin') DEFAULT 'user'
 13: );
 14:
 15: -- Items Table
 16: CREATE TABLE items (
 17:     id INT AUTO_INCREMENT PRIMARY KEY,
 18:     name VARCHAR(255) NOT NULL,
 19:     description TEXT NOT NULL,
 20:     location VARCHAR(255) NOT NULL,
 21:     status ENUM('lost', 'found', 'claimed') NOT NULL DEFAULT 'lost',
 22:     image VARCHAR(255) NOT NULL,
 23:     created_by INT NOT NULL,
 24:     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 25:     FOREIGN KEY (created_by) REFERENCES users(id) ON DELETE CASCADE
 26: );
 27:
 28: -- Claims Table
 29: CREATE TABLE claims (
 30:     id INT AUTO_INCREMENT PRIMARY KEY,
 31:     item_id INT NOT NULL,
```

```
    claimed_by INT NOT NULL,
    claimant_name VARCHAR(255) NOT NULL,
    claimant_email VARCHAR(255) NOT NULL,
    claimed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE,
    FOREIGN KEY (claimed_by) REFERENCES users(id) ON DELETE CASCADE
);

-- Messages Table
CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sender_id INT NOT NULL,
    receiver_id INT NOT NULL,
    item_id INT NOT NULL,
    message TEXT NOT NULL,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (receiver_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE
);

-- Feedback Table
CREATE TABLE feedback (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    feedback_text TEXT NOT NULL,
    submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Insert an admin user
INSERT INTO users (name, email, password, role) VALUES
('Admin', 'admin@example.com', '$2b$12$/mfZLFCm9krbLWeMYZfVD.neFNZ7y4VvK0ysOMPt1HT1Q0y5Li/HC', 'admin');

-- Trigger to update item status when claimed
DELIMITER //
CREATE TRIGGER update_item_status_on_claim
AFTER INSERT ON claims
FOR EACH ROW
BEGIN
    UPDATE items
    SET status = 'claimed'
    WHERE id = NEW.item_id;
END;
//
DELIMITER ;

-- Stored Procedure to search for items
DELIMITER //
CREATE PROCEDURE search_items(IN search_term VARCHAR(255))
BEGIN
    SELECT *
    FROM items
    WHERE name LIKE CONCAT('%', search_term, '%')
       OR description LIKE CONCAT('%', search_term, '%');
END;
//
DELIMITER ;

-- Example Data (optional)
INSERT INTO users (name, email, password, role) VALUES
('John Doe', 'john@example.com', 'password123', 'user'),
('Jane Smith', 'jane@example.com', 'password456', 'user');

INSERT INTO items (name, description, location, status, image, created_by) VALUES
('Lost Wallet', 'A brown leather wallet lost near the park.', 'City Park', 'lost', 'wallet.jpg', 1),
('Found Keys', 'A set of keys with a red keychain found in the library.', 'Library', 'found', 'keys.jpg', 2

INSERT INTO claims (item_id, claimed_by, claimant_name, claimant_email) VALUES
(1, 2, 'Jane Smith', 'jane@example.com');

INSERT INTO messages (sender_id, receiver_id, item_id, message) VALUES
(1, 2, 1, 'Hi, I think this is my wallet. Can we meet to discuss?'),
```

```
105: (2, 1, 1, 'Sure, let?s meet at the library tomorrow.');
106:
107: INSERT INTO feedback (user_id, feedback_text) VALUES
108: (1, 'Great platform, very helpful!'),
109: (2, 'It was easy to find the owner of the keys.');
```

■ File: postcss.config.js

```
  1: module.exports = {
  2:   plugins: {
  3:     tailwindcss: {},
  4:     autoprefixer: {},
  5:   },
  6: }
```

■ File: public\index.html

```
  1: <!DOCTYPE html>
  2: <html lang="en">
  3:   <head>
  4:     <meta charset="utf-8" />
  5:     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  6:     <meta name="viewport" content="width=device-width, initial-scale=1" />
  7:     <meta name="theme-color" content="#000000" />
  8:     <meta
  9:       name="description"
 10:       content="Web site created using create-react-app"
 11:     />
 12:     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
 13:     <!--
 14:       manifest.json provides metadata used when your web app is installed on a
 15:       user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
 16:     -->
 17:     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
 18:     <!--
 19:       Notice the use of %PUBLIC_URL% in the tags above.
 20:       It will be replaced with the URL of the `public` folder during the build.
 21:       Only files inside the `public` folder can be referenced from the HTML.
 22:
 23:       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
 24:       work correctly both with client-side routing and a non-root public URL.
 25:       Learn how to configure a non-root public URL by running `npm run build`.
 26:     -->
 27:     <title>React App</title>
 28:   </head>
 29:   <body>
 30:     <noscript>You need to enable JavaScript to run this app.</noscript>
 31:     <div id="root"></div>
 32:     <!--
 33:       This HTML file is a template.
 34:       If you open it directly in the browser, you will see an empty page.
 35:
 36:       You can add webfonts, meta tags, or analytics to this file.
 37:       The build step will place the bundled scripts into the <body> tag.
 38:
 39:       To begin the development, run `npm start` or `yarn start`.
 40:       To create a production bundle, use `npm run build` or `yarn build`.
 41:     -->
 42:   </body>
 43: </html>
```

■ File: script.py

```
  1: import os
  2: from reportlab.lib.pagesizes import A4
  3: from reportlab.lib.units import mm
  4: from reportlab.pdfgen import canvas
```

```
 5:
 6: def get_code_files(directory, excluded_files=None, excluded_dirs=None):
 7:     """Fetch all project files except specified exclusions."""
 8:     if excluded_files is None:
 9:         excluded_files = {
10:             'package.json',
11:             'package-lock.json',
12:             'yarn.lock',
13:             'README.md',
14:             '.DS_Store',
15:             'Thumbs.db',
16:             'Desktop.ini',
17:             'favicon.ico',
18:             'logo192.png',
19:             'logo512.png',
20:             'manifest.json',
21:             'robots.txt'
22:         }
23:
24:     if excluded_dirs is None:
25:         excluded_dirs = {
26:             'node_modules',
27:             '.git',
28:             '__pycache__',
29:             'build',
30:             '.next',
31:             'dist',
32:             'coverage',
33:             '.nyc_output',
34:             'logs',
35:             'uploads',  # Exclude the uploads directory with images
36:             'venv',     # Exclude Python virtual environment
37:             'env',      # Alternative venv name
38:             '.venv'     # Alternative venv name
39:         }
40:
41:     code_files = {}
42:
43:     for root, dirs, files in os.walk(directory):
44:         # Skip excluded directories
45:         dirs[:] = [d for d in dirs if d not in excluded_dirs]
46:
47:         # Skip if current directory is an excluded directory
48:         if any(excluded_dir in root.split(os.sep) for excluded_dir in excluded_dirs):
49:             continue
50:
51:         for file in files:
52:             # Skip excluded files
53:             if file in excluded_files:
54:                 continue
55:
56:             file_path = os.path.join(root, file)
57:
58:             # Get file extension
59:             _, ext = os.path.splitext(file)
60:
61:             try:
62:                 # Try to read as text file first
63:                 if ext.lower() in {'.js', '.jsx', '.ts', '.tsx', '.css', '.scss', '.sass', '.less',
64:                                    '.html', '.htm', '.json', '.md', '.txt', '.xml', '.yaml', '.yml',
65:                                    '.config', '.gitignore', '.env', '.py', '.sh', '.bat', '.cmd',
66:                                    '.svg', '.dockerfile', '.editorconfig', '.eslintrc', '.prettierrc',
67:                                    '.sql', '.toml', '.ini', '.conf'} or file.startswith('.'):
68:                     with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
69:                         code_files[file_path] = f.readlines()
70:                 else:
71:                     # For binary files, just note them as binary
72:                     code_files[file_path] = [f"[Binary file - {ext} format]"]
73:
74:             except Exception as e:
75:                 print(f"? Error reading {file_path}: {e}")
76:                 code_files[file_path] = [f"[Error reading file: {str(e)}]"]
77:
```

```
    return code_files


def create_pdf(code_data, output_pdf="Lost_and_Found_Code_Export.pdf"):
    c = canvas.Canvas(output_pdf, pagesize=A4)
    width, height = A4
    margin = 20 * mm
    line_height = 10
    y = height - margin

    # Title
    c.setFont("Helvetica-Bold", 16)
    c.drawString(margin, y, "? Lost and Found Project Code Export")
    y -= 2 * line_height

    # Project description
    c.setFont("Helvetica", 10)
    c.drawString(margin, y, "Full-stack Lost and Found application with React frontend and Flask backend")
    y -= line_height

    c.setFont("Helvetica-Bold", 12)
    c.drawString(margin, y, "? Project File Structure:")
    y -= 2 * line_height

    file_paths = sorted(list(code_data.keys()))

    # 1. File list organized by category
    c.setFont("Courier", 8)

    # Group files by type/location
    frontend_files = [f for f in file_paths if '/src/' in f or f.endswith(('.js', '.jsx', '.css', '.html'))
    backend_files = [f for f in file_paths if '/backend/' in f or f.endswith('.py')]
    config_files = [f for f in file_paths if any(f.endswith(ext) for ext in ['.json', '.js', '.config', '.s
    other_files = [f for f in file_paths if f not in frontend_files and f not in backend_files and f not in

    # Frontend files
    if frontend_files:
        c.setFont("Helvetica-Bold", 10)
        c.drawString(margin, y, "? Frontend Files (React):")
        y -= line_height
        c.setFont("Courier", 8)
        for path in frontend_files:
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin
            display_path = os.path.relpath(path)
            c.drawString(margin + 10, y, f"- {display_path}")
            y -= line_height
        y -= line_height

    # Backend files
    if backend_files:
        c.setFont("Helvetica-Bold", 10)
        c.drawString(margin, y, "? Backend Files (Flask/Python):")
        y -= line_height
        c.setFont("Courier", 8)
        for path in backend_files:
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin
            display_path = os.path.relpath(path)
            c.drawString(margin + 10, y, f"- {display_path}")
            y -= line_height
        y -= line_height

    # Configuration files
    if config_files:
        c.setFont("Helvetica-Bold", 10)
        c.drawString(margin, y, "?? Configuration Files:")
        y -= line_height
        c.setFont("Courier", 8)
```

```
        for path in config_files:
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin
            display_path = os.path.relpath(path)
            c.drawString(margin + 10, y, f"- {display_path}")
            y -= line_height
        y -= line_height

    # Other files
    if other_files:
        c.setFont("Helvetica-Bold", 10)
        c.drawString(margin, y, "? Other Files:")
        y -= line_height
        c.setFont("Courier", 8)
        for path in other_files:
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin
            display_path = os.path.relpath(path)
            c.drawString(margin + 10, y, f"- {display_path}")
            y -= line_height

    # Add page break before code content
    c.showPage()
    y = height - margin

    # 2. File contents
    for file_path in file_paths:
        lines = code_data[file_path]
        print(f"? Adding: {file_path}")

        if y < margin + 3 * line_height:
            c.showPage()
            y = height - margin

        # File header
        rel_path = os.path.relpath(file_path)
        c.setFont("Helvetica-Bold", 12)

        # Add file type indicator
        if '/src/' in rel_path:
            file_icon = "??"  # React
        elif rel_path.endswith('.py'):
            file_icon = "?"  # Python
        elif rel_path.endswith('.sql'):
            file_icon = "??"  # Database
        elif rel_path.endswith(('.css', '.scss')):
            file_icon = "?"  # Styles
        elif rel_path.endswith(('.js', '.jsx', '.ts', '.tsx')):
            file_icon = "?"  # JavaScript
        else:
            file_icon = "?"  # General

        c.drawString(margin, y, f"{file_icon} File: {rel_path}")
        y -= line_height

        # Add separator line
        c.setFont("Courier", 8)
        c.drawString(margin, y, "=" * 80)
        y -= line_height

        # File content
        for line_num, line in enumerate(lines, 1):
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin

            # Clean and truncate line
            line = line.strip("\n").encode("latin-1", "replace").decode("latin-1")
```

```
224:
225:             # Add line numbers for code files
226:             if any(rel_path.endswith(ext) for ext in ['.js', '.jsx', '.ts', '.tsx', '.css', '.py', '.html',
227:                 display_line = f"{line_num:3d}: {line[:275]}"
228:             else:
229:                 display_line = line[:300]
230:
231:             c.drawString(margin, y, display_line)
232:             y -= line_height
233:
234:         # Add spacing between files
235:         y -= line_height
236:         if y > margin:
237:             c.setFont("Courier", 8)
238:             c.drawString(margin, y, "-" * 80)
239:             y -= 2 * line_height
240:
241:     c.save()
242:     print(f"? PDF successfully created: {output_pdf}")
243:     print(f"? Total files processed: {len(code_data)}")
244:
245:
246: def main():
247:     root_dir = os.path.dirname(os.path.abspath(__file__))
248:
249:     # Files to exclude (updated for Lost and Found project)
250:     excluded_files = {
251:         'package.json',
252:         'package-lock.json',
253:         'yarn.lock',
254:         'README.md',
255:         '.DS_Store',
256:         'Thumbs.db',
257:         'Desktop.ini',
258:         'favicon.ico',
259:         'logo192.png',
260:         'logo512.png',
261:         'manifest.json',
262:         'robots.txt',
263:         'logo.svg'  # React default logo
264:     }
265:
266:     # Directories to exclude (updated for Lost and Found project)
267:     excluded_dirs = {
268:         'node_modules',
269:         '.git',
270:         '__pycache__',
271:         'build',
272:         'dist',
273:         '.next',
274:         'coverage',
275:         '.nyc_output',
276:         'logs',
277:         'uploads',  # Contains uploaded images
278:         'static/uploads',  # Flask static uploads directory
279:         'venv',     # Python virtual environment
280:         'env',      # Alternative venv name
281:         '.venv'     # Alternative venv name (hidden)
282:     }
283:
284:     print("? Scanning Lost and Found project files...")
285:     print("? Including React frontend and Flask backend code...")
286:     code_files = get_code_files(root_dir, excluded_files, excluded_dirs)
287:
288:     if not code_files:
289:         print("? No files found to process!")
290:         return
291:
292:     print(f"? Found {len(code_files)} files to include in PDF")
293:
294:     # Show file breakdown
295:     frontend_count = len([f for f in code_files.keys() if '/src/' in f])
296:     backend_count = len([f for f in code_files.keys() if f.endswith('.py')])
```

```
297:     config_count = len([f for f in code_files.keys() if any(f.endswith(ext) for ext in ['.json', '.js', '.s
298:
299:     print(f"  ??  Frontend files: {frontend_count}")
300:     print(f"  ? Backend files: {backend_count}")
301:     print(f"  ??  Config files: {config_count}")
302:     print(f"  ? Other files: {len(code_files) - frontend_count - backend_count - config_count}")
303:
304:     create_pdf(code_files)
305:
306:
307: if __name__ == "__main__":
308:     main()
```

■ File: src\App.css

```
================================================================================

--------------------------------------------------------------------------------
```

■ File: src\App.js

```
================================================================================
  1: import {
  2:   BrowserRouter as Router,
  3:   Routes,
  4:   Route,
  5:   useLocation,
  6: } from "react-router-dom";
  7: import { useEffect } from "react";
  8: import { Toaster } from "react-hot-toast";
  9:
 10: import LoginPage from "./pages/LoginPage";
 11: import RegisterPage from "./pages/RegisterPage";
 12: import HomePage from "./pages/HomePage";
 13: import AdminPage from "./pages/AdminPage";
 14: import ItemRegisterPage from "./pages/ItemRegisterPage";
 15: import ItemsPage from "./pages/ItemsPage";
 16: import Header from "./components/Header";
 17:
 18: // Wrapper component to conditionally show header
 19: function LayoutWrapper() {
 20:   const location = useLocation();
 21:
 22:   // Define paths where header should be hidden
 23:   const hideHeaderPaths = ["/", "/register"];
 24:   const shouldHideHeader = hideHeaderPaths.includes(location.pathname);
 25:
 26:   // Optional: Scroll to top on route change
 27:   useEffect(() => {
 28:     window.scrollTo(0, 0);
 29:   }, [location.pathname]);
 30:
 31:   return (
 32:     <>
 33:       <Toaster position="top-center" reverseOrder={false} />
 34:       {!shouldHideHeader && <Header />}
 35:       <Routes>
 36:         <Route path="/" element={<LoginPage />} />
 37:         <Route path="/register" element={<RegisterPage />} />
 38:         <Route path="/home" element={<HomePage />} />
 39:         <Route path="/admin" element={<AdminPage />} />
 40:         <Route path="/register-item" element={<ItemRegisterPage />} />
 41:         <Route path="/lost-found-items" element={<ItemsPage />} />
 42:       </Routes>
 43:     </>
 44:   );
 45: }
 46:
 47: export default function App() {
 48:   return (
 49:     <Router>
 50:       <LayoutWrapper />
 51:     </Router>
```

```
 52:   );
 53: }
```

■ File: src\App.test.js

```
  1: import { render, screen } from '@testing-library/react';
  2: import App from './App';
  3:
  4: test('renders learn react link', () => {
  5:   render(<App />);
  6:   const linkElement = screen.getByText(/learn react/i);
  7:   expect(linkElement).toBeInTheDocument();
  8: });
```

■ File: src\components\AdminTable.js

```
  1: export default function AdminTable({
  2:   title,
  3:   type,
  4:   data,
  5:   headers,
  6:   rowRenderer,
  7: }) {
  8:   return (
  9:     <section className="mb-8">
 10:       <h2 className="text-xl font-bold text-gray-800 mb-3">{title}</h2>
 11:
 12:       {data.length === 0 ? (
 13:         <p className="text-gray-500 text-sm italic">No {type}s found.</p>
 14:       ) : (
 15:         <div className="overflow-x-auto shadow border border-gray-200 rounded-lg">
 16:           <table className="min-w-full text-sm text-left text-gray-700">
 17:             <thead className="bg-gray-100 text-xs uppercase text-gray-600 sticky top-0 z-10">
 18:               <tr>
 19:                 {headers.map((h, idx) => (
 20:                   <th key={idx} className="px-5 py-3 border-b font-bold">
 21:                     {h}
 22:                   </th>
 23:                 ))}
 24:               </tr>
 25:             </thead>
 26:
 27:             <tbody>
 28:               {data.map((item, i) => (
 29:                 <tr
 30:                   key={i}
 31:                   className={`border-b ${
 32:                     i % 2 === 0 ? "bg-white" : "bg-gray-50"
 33:                   } hover:bg-blue-50 transition duration-150`}
 34:                 >
 35:                   {rowRenderer(item).map((cell, j) => (
 36:                     <td key={j} className="px-5 py-3 whitespace-nowrap">
 37:                       {cell}
 38:                     </td>
 39:                   ))}
 40:                 </tr>
 41:               ))}
 42:             </tbody>
 43:           </table>
 44:         </div>
 45:       )}
 46:     </section>
 47:   );
 48: }
```

■ File: src\components\ClaimsTable.js

```
================================================================================
```

```
  1: import React from "react";
  2:
  3: export default function ClaimsTable({ claims }) {
  4:   return (
  5:     <div className="overflow-x-auto rounded-xl border border-gray-200">
  6:       <table className="min-w-full table-fixed">
  7:         <thead>
  8:           <tr className="bg-gray-100 text-left text-sm font-semibold text-gray-700">
  9:             <th className="w-1/3 px-4 py-3">ITEM</th>
 10:             <th className="w-1/3 px-4 py-3">LOCATION</th>
 11:             <th className="w-1/3 px-4 py-3">CLAIMED AT</th>
 12:           </tr>
 13:         </thead>
 14:         <tbody>
 15:           {claims.length === 0 ? (
 16:             <tr>
 17:               <td colSpan={3} className="text-center py-4 text-gray-500">
 18:                 No claims yet.
 19:               </td>
 20:             </tr>
 21:           ) : (
 22:             claims.map((claim, idx) => (
 23:               <tr
 24:                 key={idx}
 25:                 className="border-t text-sm text-gray-800 hover:bg-gray-50 transition"
 26:               >
 27:                 <td className="px-4 py-3 font-medium break-words">
 28:                   {claim.item_name}
 29:                 </td>
 30:                 <td className="px-4 py-3 break-words">{claim.location}</td>
 31:                 <td className="px-4 py-3 text-sm text-gray-600">
 32:                   {new Date(claim.claimed_at).toUTCString()}
 33:                 </td>
 34:               </tr>
 35:             ))
 36:           )}
 37:         </tbody>
 38:       </table>
 39:     </div>
 40:   );
 41: }
```

```
--------------------------------------------------------------------------------
```

■ File: src\components\FeedbackForm.js

```
================================================================================
```

```
  1: import { useState } from "react";
  2: import toast from "react-hot-toast";
  3:
  4: export default function FeedbackForm() {
  5:   const [text, setText] = useState("");
  6:   const [loading, setLoading] = useState(false);
  7:
  8:   const handleSubmit = async (e) => {
  9:     e.preventDefault();
 10:     if (!text.trim()) return;
 11:
 12:     setLoading(true);
 13:
 14:     try {
 15:       const res = await fetch("/submit_feedback", {
 16:         method: "POST",
 17:         headers: { "Content-Type": "application/json" },
 18:         credentials: "include",
 19:         body: JSON.stringify({ feedback: text.trim() }),
 20:       });
 21:
 22:       if (res.ok) {
 23:         setText("");
 24:         toast.success("Thank you for your feedback!");
 25:       } else {
```

```
26:         toast.error("Error submitting feedback.");
27:       }
28:     } catch (err) {
29:       toast.error("Network error. Please try again.");
30:     } finally {
31:       setLoading(false);
32:     }
33:   };
34:
35:   return (
36:     <form onSubmit={handleSubmit} className="space-y-4">
37:       <textarea
38:         className="w-full px-4 py-2 border border-gray-300 rounded-xl focus:outline-none focus:ring-2 focus
39:         rows="4"
40:         required
41:         value={text}
42:         onChange={(e) => setText(e.target.value)}
43:         placeholder="Write your feedback..."
44:       />
45:       <div className="flex justify-end">
46:         <button
47:           type="submit"
48:           disabled={loading}
49:           className="bg-green-600 hover:bg-green-700 text-white px-6 py-2 rounded-xl transition disabled:opa
50:         >
51:           {loading ? "Submitting..." : "Submit"}
52:         </button>
53:       </div>
54:     </form>
55:   );
56: }
```

■ File: src\components\FlashMessage.js

==========

■ File: src\components\Header.js

==========

```
 1: import { Link, useNavigate } from "react-router-dom";
 2: import { useEffect, useState } from "react";
 3:
 4: export default function Header() {
 5:   const [user, setUser] = useState(null);
 6:   const [loading, setLoading] = useState(true);
 7:   const [sidebarOpen, setSidebarOpen] = useState(false);
 8:   const navigate = useNavigate();
 9:
10:   useEffect(() => {
11:     const fetchUser = async () => {
12:       try {
13:         const res = await fetch("http://localhost:5000/api/user", {
14:           credentials: "include",
15:         });
16:
17:         if (res.ok) {
18:           const data = await res.json();
19:           setUser(data);
20:         } else {
21:           setUser(null);
22:         }
23:       } catch (err) {
24:         console.error("Error fetching user:", err);
25:         setUser(null);
26:       } finally {
27:         setLoading(false);
28:       }
29:     };
30:
31:     fetchUser();
32:   }, []);
```

```

  const handleLogout = async () => {
    try {
      await fetch("http://localhost:5000/api/logout", {
        method: "POST",
        credentials: "include",
      });
      setUser(null);
      navigate("/");
    } catch (err) {
      console.error("Logout error:", err);
    }
  };

  if (loading) return null;

  return (
    <>
      <header className=" text-gray-800 shadow-md">
        <div className="max-w-7xl mx-auto px-4 py-3 flex items-center justify-between">
          {/* Logo */}
          <Link to={user?.role === "admin" ? "/admin" : "/home"}>
            <h1 className="text-2xl font-bold">Lost & Found</h1>
          </Link>

          {/* Desktop nav */}
          <nav className="hidden md:flex items-center gap-6 text-sm">
            {/* Home link (always visible) */}
            <Link to="/home" className="hover:underline">
              Home
            </Link>

            {user ? (
              <>
                <Link to="/register-item" className="hover:underline">
                  Register Item
                </Link>
                <Link to="/lost-found-items" className="hover:underline">
                  Browse Items
                </Link>
                {user.role === "admin" && (
                  <Link to="/admin" className="hover:underline">
                    Admin Dashboard
                  </Link>
                )}
                <button
                  onClick={handleLogout}
                  className="bg-white text-blue-600 px-4 py-2 rounded hover:bg-gray-100 transition"
                >
                  Logout
                </button>
              </>
            ) : (
              <>
                <Link to="/" className="hover:underline">
                  Login
                </Link>
                <Link to="/register" className="hover:underline">
                  Register
                </Link>
              </>
            )}
          </nav>

          {/* Mobile Hamburger */}
          <button
            className="md:hidden text-white focus:outline-none"
            onClick={() => setSidebarOpen(true)}
          >
            <svg className="w-6 h-6" fill="none" stroke="currentColor">
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
```

```
                strokeWidth={2}
                d="M4 6h16M4 12h16M4 18h16"
              />
            </svg>
          </button>
        </div>
      </header>

      {/* Sidebar overlay */}
      {sidebarOpen && (
        <div
          className="fixed inset-0 bg-opacity-40 z-40"
          onClick={() => setSidebarOpen(false)}
        ></div>
      )}

      {/* Sidebar */}
      <div
        className={`fixed top-0 left-0 h-full w-64 bg-white shadow-lg z-50 transform transition-transform du
          sidebarOpen ? "translate-x-0" : "-translate-x-full"
        } md:hidden`}
      >
        <div className="p-4 flex items-center justify-between border-b">
          <h2 className="text-lg font-bold text-blue-600">Menu</h2>
          <button
            className="text-gray-600"
            onClick={() => setSidebarOpen(false)}
          >
            close
          </button>
        </div>
        <div className="flex flex-col gap-4 p-4 text-sm ">
          {user ? (
            <>
              <Link to="/register-item" onClick={() => setSidebarOpen(false)}>
                Register Item
              </Link>
              <Link
                to="/lost-found-items"
                onClick={() => setSidebarOpen(false)}
              >
                Browse Items
              </Link>
              {user.role === "admin" && (
                <Link to="/admin" onClick={() => setSidebarOpen(false)}>
                  Admin Dashboard
                </Link>
              )}
              <button
                onClick={() => {
                  handleLogout();
                  setSidebarOpen(false);
                }}
                className="text-left text-red-600 hover:underline"
              >
                Logout
              </button>
            </>
          ) : (
            <>
              <Link to="/" onClick={() => setSidebarOpen(false)}>
                Login
              </Link>
              <Link to="/register" onClick={() => setSidebarOpen(false)}>
                Register
              </Link>
            </>
          )}
        </div>
      </div>
    </>
  );
}
```

■ File: src\components\ItemTable.js

```
================================================================================
```

```
  1: import React from "react";
  2:
  3: export default function ItemTable({ items }) {
  4:   return (
  5:     <div className="overflow-x-auto rounded-xl border border-gray-200">
  6:       <table className="min-w-full table-fixed">
  7:         <thead>
  8:           <tr className="bg-gray-100 text-left text-sm font-semibold text-gray-700">
  9:             <th className="w-1/4 px-4 py-3">ITEM</th>
 10:             <th className="w-1/4 px-4 py-3">STATUS</th>
 11:             <th className="w-1/4 px-4 py-3">LOCATION</th>
 12:             <th className="w-1/4 px-4 py-3">CREATED AT</th>
 13:           </tr>
 14:         </thead>
 15:         <tbody>
 16:           {items.length === 0 ? (
 17:             <tr>
 18:               <td colSpan={4} className="text-center py-4 text-gray-500">
 19:                 No items registered.
 20:               </td>
 21:             </tr>
 22:           ) : (
 23:             items.map((item, idx) => (
 24:               <tr
 25:                 key={idx}
 26:                 className="border-t text-sm text-gray-800 hover:bg-gray-50 transition"
 27:               >
 28:                 <td className="px-4 py-3 font-medium break-words">
 29:                   {item.name}
 30:                 </td>
 31:                 <td className="px-4 py-3">
 32:                   <span
 33:                     className={`inline-block px-2 py-1 rounded-full text-xs font-semibold ${
 34:                       item.status === "found"
 35:                         ? "bg-green-100 text-green-800"
 36:                         : "bg-yellow-100 text-yellow-800"
 37:                     }`}
 38:                   >
 39:                     {item.status}
 40:                   </span>
 41:                 </td>
 42:                 <td className="px-4 py-3 break-words">{item.location}</td>
 43:                 <td className="px-4 py-3 text-sm text-gray-600">
 44:                   {new Date(item.created_at).toUTCString()}
 45:                 </td>
 46:               </tr>
 47:             ))
 48:           )}
 49:         </tbody>
 50:       </table>
 51:     </div>
 52:   );
 53: }
```

```
--------------------------------------------------------------------------------
```

■ File: src\components\MessageList.js

```
================================================================================
```

```
  1: import { useState } from "react";
  2: import toast from "react-hot-toast";
  3:
  4: export default function MessageList({ messages }) {
  5:   const [replies, setReplies] = useState({});
  6:   const [sentReplies, setSentReplies] = useState({});
  7:
  8:   const handleReply = async (msgId, senderId, itemId) => {
  9:     const replyText = replies[msgId]?.trim();
 10:     if (!replyText) return;
 11:
 12:     try {
 13:       const res = await fetch("http://localhost:5000/api/reply", {
```

```
        method: "POST",
        credentials: "include",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          receiver_id: senderId,
          item_id: itemId,
          message: replyText,
        }),
      });

      if (res.ok) {
        toast.success("Reply sent!");

        // Store the sent reply
        setSentReplies((prev) => ({
          ...prev,
          [msgId]: [
            ...(prev[msgId] || []),
            {
              text: replyText,
              timestamp: new Date().toLocaleString(),
            },
          ],
        }));

        // Clear the input
        setReplies((prev) => ({ ...prev, [msgId]: "" }));
      } else {
        const err = await res.json();
        toast.error(err?.error || "Failed to send reply.");
      }
    } catch (error) {
      console.error("Reply error:", error);
      toast.error("Server error while sending reply.");
    }
  };

  if (!messages.length)
    return <p className="text-gray-500 text-sm italic">No messages yet.</p>;

  return (
    <div className="space-y-6">
      {messages.map((msg) => (
        <div
          key={msg.id}
          className="bg-white border border-gray-200 rounded-2xl p-5 shadow-sm"
        >
          <div className="mb-2">
            <p className="text-sm text-gray-500">
              <span className="font-medium text-gray-700">From:</span>{" "}
              {msg.sender_name}
            </p>
            <p className="text-sm text-gray-500">
              <span className="font-medium text-gray-700">Regarding Item:</span>{" "}
              {msg.item_name}
            </p>
          </div>

          <div className="mb-4">
            <div className="bg-gray-50 border border-gray-100 rounded-xl p-3 text-gray-800 text-sm leading-
              {msg.message}
            </div>
            <div className="text-xs text-gray-400 mt-1">{msg.sent_at}</div>
          </div>

          {sentReplies[msg.id]?.length > 0 && (
            <div className="mb-4 space-y-3">
              <div className="text-xs text-gray-500 font-medium uppercase tracking-wide">
                Your Replies
              </div>
              {sentReplies[msg.id].map((reply, index) => (
                <div key={index} className="flex justify-end">
                  <div className="max-w-xs lg:max-w-md">
```

```
 87:                     <div className="bg-blue-600 text-white rounded-xl px-4 py-2 text-sm">
 88:                       {reply.text}
 89:                     </div>
 90:                     <div className="text-xs text-gray-400 mt-1 text-right">
 91:                       {reply.timestamp}
 92:                     </div>
 93:                   </div>
 94:                 </div>
 95:               ))}
 96:             </div>
 97:           )}
 98:
 99:           <textarea
100:             placeholder="Write your reply..."
101:             className="w-full border border-gray-300 rounded-xl px-4 py-2 text-sm focus:outline-none focus:
102:             rows="3"
103:             value={replies[msg.id] || ""}
104:             onChange={(e) =>
105:               setReplies((prev) => ({ ...prev, [msg.id]: e.target.value }))
106:             }
107:           />
108:
109:           <div className="text-right mt-3">
110:             <button
111:               className="bg-blue-600 hover:bg-blue-700 text-white text-sm px-4 py-2 rounded-xl transition d
112:               onClick={() => handleReply(msg.id, msg.sender_id, msg.item_id)}
113:               disabled={!replies[msg.id]?.trim()}
114:             >
115:               Send Reply
116:             </button>
117:           </div>
118:         </div>
119:       ))}
120:     </div>
121:   );
122: }
```

■ File: src\index.css

```
  1: @tailwind base;
  2: @tailwind components;
  3: @tailwind utilities;
  4:
  5:
  6:
```

■ File: src\index.js

```
  1: import React from 'react';
  2: import ReactDOM from 'react-dom/client';
  3: import './index.css';
  4: import App from './App';
  5: import reportWebVitals from './reportWebVitals';
  6:
  7: const root = ReactDOM.createRoot(document.getElementById('root'));
  8: root.render(
  9:   <React.StrictMode>
 10:     <App />
 11:   </React.StrictMode>
 12: );
 13:
 14: // If you want to start measuring performance in your app, pass a function
 15: // to log results (for example: reportWebVitals(console.log))
 16: // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
 17: reportWebVitals();
```

■ File: src\pages\AdminPage.js

```
================================================================================
```

```
  1: import { useEffect, useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3: import AdminTable from "../components/AdminTable";
  4: import toast, { Toaster } from "react-hot-toast";
  5: import Swal from "sweetalert2";
  6: import withReactContent from "sweetalert2-react-content";
  7:
  8: const MySwal = withReactContent(Swal);
  9:
 10: export default function AdminPage() {
 11:   const navigate = useNavigate();
 12:   const [loading, setLoading] = useState(true);
 13:   const [items, setItems] = useState([]);
 14:   const [claims, setClaims] = useState([]);
 15:   const [users, setUsers] = useState([]);
 16:   const [feedback, setFeedback] = useState([]);
 17:
 18:   // ? Step 1: Validate Admin Access
 19:   const validateAdminAccess = async () => {
 20:     try {
 21:       const res = await fetch("http://localhost:5000/api/user", {
 22:         credentials: "include",
 23:       });
 24:
 25:       if (!res.ok) throw new Error("Not authenticated");
 26:
 27:       const user = await res.json();
 28:       if (!user || user.role !== "admin") {
 29:         toast.error("Access denied. Redirecting...");
 30:         navigate("/");
 31:         return false;
 32:       }
 33:       return true;
 34:     } catch (err) {
 35:       toast.error("Error validating admin access.");
 36:       navigate("/");
 37:       return false;
 38:     }
 39:   };
 40:
 41:   // ? Step 2: Fetch Admin Dashboard Data
 42:   const fetchAdminData = async () => {
 43:     try {
 44:       const res = await fetch("http://localhost:5000/api/admin/dashboard", {
 45:         credentials: "include",
 46:       });
 47:
 48:       if (!res.ok) throw new Error("Failed to load dashboard");
 49:
 50:       const data = await res.json();
 51:       setItems(data.items || []);
 52:       setClaims(data.claims || []);
 53:       setUsers(data.users || []);
 54:       setFeedback(data.feedback || []);
 55:     } catch (err) {
 56:       toast.error("Failed to load admin data.");
 57:     } finally {
 58:       setLoading(false);
 59:     }
 60:   };
 61:
 62:   useEffect(() => {
 63:     const init = async () => {
 64:       const isAdmin = await validateAdminAccess();
 65:       if (isAdmin) await fetchAdminData();
 66:     };
 67:     init();
 68:   }, []);
 69:
 70:   // ?? Step 3: Delete Handler with Confirmation
 71:   const handleDelete = async (type, id) => {
```

```
    const confirmed = await MySwal.fire({
      title: `Delete this ${type}?`,
      text: `This action cannot be undone.`,
      icon: "warning",
      showCancelButton: true,
      confirmButtonColor: "#d33",
      cancelButtonColor: "#3085d6",
      confirmButtonText: "Yes, delete it!",
    });

    if (!confirmed.isConfirmed) return;

    let url;
    if (type === "feedback") {
      url = `http://localhost:5000/api/admin/feedback/${id}`;
    } else {
      url = `http://localhost:5000/api/admin/${type}s/${id}`;
    }

    try {
      const res = await fetch(url, {
        method: "DELETE",
        credentials: "include",
      });

      if (res.ok) {
        toast.success(
          `${type.charAt(0).toUpperCase() + type.slice(1)} deleted.`
        );
        await fetchAdminData();
      } else {
        const data = await res.json();
        toast.error(data?.error || ` Failed to delete ${type}.`);
      }
    } catch (err) {
      toast.error(`Error deleting ${type}.`);
    }
  };

  // ? Step 4: Show loader while validating
  if (loading) {
    return (
      <div className="min-h-screen flex items-center justify-center text-gray-500">
        Verifying admin access...
      </div>
    );
  }

  // ? Step 5: Render Admin Dashboard
  return (
    <div className="max-w-7xl mx-auto px-6 py-8 space-y-8">
      <Toaster position="top-center w-half" />

      <h1 className="text-2xl font-bold text-center text-blue-600 mb-4">
        Admin Dashboard
      </h1>

      {/* Items Table */}
      <AdminTable
        title="Registered Items"
        type="item"
        data={items}
        headers={["Name", "Location", "Status", "Actions"]}
        rowRenderer={(item) => [
          item.name,
          item.location,
          item.status,
          <button
            onClick={() => handleDelete("item", item.id)}
            className="text-red-600 hover:underline text-sm"
          >
            Delete
          </button>,
```

```
145:         ]}
146:       />
147:
148:       {/* Claims Table */}
149:       <AdminTable
150:         title="Claims"
151:         type="claim"
152:         data={claims}
153:         headers={["Item", "Claimer", "Location", "Actions"]}
154:         rowRenderer={(claim) => [
155:           claim.item_name,
156:           claim.claimer_name || claim.claimed_by,
157:           claim.location,
158:           <button
159:             onClick={() => handleDelete("claim", claim.id)}
160:             className="text-red-600 hover:underline text-sm"
161:           >
162:             Delete
163:           </button>,
164:         ]}
165:       />
166:
167:       {/* Users Table */}
168:       <AdminTable
169:         title="Users"
170:         type="user"
171:         data={users}
172:         headers={["Name", "Email", "Role", "Actions"]}
173:         rowRenderer={(user) => [
174:           user.name,
175:           user.email,
176:           user.role || "user",
177:           <button
178:             onClick={() => handleDelete("user", user.id)}
179:             className="text-red-600 hover:underline text-sm"
180:           >
181:             Delete
182:           </button>,
183:         ]}
184:       />
185:
186:       {/* Feedback Table */}
187:       <AdminTable
188:         title="Feedback"
189:         type="feedback"
190:         data={feedback}
191:         headers={["User", "Message", "Actions"]}
192:         rowRenderer={(fb) => [
193:           fb.user_name || "Anonymous",
194:           fb.feedback_text,
195:           <button
196:             onClick={() => handleDelete("feedback", fb.id)}
197:             className="text-red-600 hover:underline text-sm"
198:           >
199:             Delete
200:           </button>,
201:         ]}
202:       />
203:     </div>
204:   );
205: }
```

■ File: src\pages\HomePage.js

```
  1: import { useEffect, useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3: import ItemTable from "../components/ItemTable";
  4: import MessageList from "../components/MessageList";
  5: import ClaimsTable from "../components/ClaimsTable";
  6:
  7: export default function HomePage() {
```

```
  const navigate = useNavigate();
  const [name, setName] = useState("");
  const [role, setRole] = useState("");
  const [items, setItems] = useState([]);
  const [claims, setClaims] = useState([]);
  const [messages, setMessages] = useState([]);
  const [loading, setLoading] = useState(true);

  const [feedbackText, setFeedbackText] = useState("");
  const [feedbackMessage, setFeedbackMessage] = useState("");

  useEffect(() => {
    const fetchHomeData = async () => {
      try {
        const res = await fetch("http://localhost:5000/api/dashboard", {
          method: "GET",
          credentials: "include",
        });

        if (!res.ok) {
          navigate("/");
          return;
        }

        const data = await res.json();

        setName(data.user.name);
        setRole(data.user.role);
        setItems(data.items || []);
        setClaims(data.claims || []);
        setMessages(data.messages || []);
      } catch (err) {
        console.error("Failed to fetch dashboard:", err);
        navigate("/");
      } finally {
        setLoading(false);
      }
    };

    fetchHomeData();
  }, [navigate]);

  const handleFeedbackSubmit = async (e) => {
    e.preventDefault();
    if (!feedbackText.trim()) return;

    try {
      const res = await fetch("http://localhost:5000/api/feedback", {
        method: "POST",
        credentials: "include",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ feedback: feedbackText.trim() }),
      });

      if (res.ok) {
        const data = await res.json();
        setFeedbackMessage(data.message || "Thank you for your feedback!");
        setFeedbackText("");
      } else {
        const err = await res.json();
        setFeedbackMessage(err.error || "Failed to submit feedback.");
      }
    } catch (err) {
      console.error(err);
      setFeedbackMessage("Something went wrong while submitting feedback.");
    }
  };

  if (loading) {
    return (
      <div className="min-h-screen flex items-center justify-center text-gray-600">
```

```
 81:         Loading your dashboard...
 82:       </div>
 83:     );
 84:   }
 85:
 86:   return (
 87:     <div className="p-6 max-w-6xl mx-auto space-y-10">
 88:       <div className="text-center">
 89:         <h1 className="text-3xl font-bold text-gray-800">Welcome, {name}</h1>
 90:         <p className="text-sm text-gray-500 mt-1">
 91:           Here?s an overview of your account and activity.
 92:         </p>
 93:       </div>
 94:
 95:       <section className="bg-white p-6 rounded-2xl shadow-sm border">
 96:         <h2 className="text-xl font-semibold text-gray-700 mb-4 border-b pb-2">My Registered Items</h2>
 97:         <ItemTable items={items} />
 98:       </section>
 99:
100:       <section className="bg-white p-6 rounded-2xl shadow-sm border">
101:         <h2 className="text-xl font-semibold text-gray-700 mb-4 border-b pb-2">My Claims</h2>
102:         <ClaimsTable claims={claims} />
103:       </section>
104:
105:       <section className="bg-white p-6 rounded-2xl shadow-sm border">
106:         <h2 className="text-xl font-semibold text-gray-700 mb-4 border-b pb-2">Messages</h2>
107:         <MessageList messages={messages} />
108:       </section>
109:
110:       {role !== "admin" && (
111:         <section className="bg-white p-6 rounded-2xl shadow-sm border">
112:           <h2 className="text-xl font-semibold text-gray-700 mb-4 border-b pb-2">Platform Feedback</h2>
113:           <form onSubmit={handleFeedbackSubmit} className="space-y-4">
114:             <textarea
115:               rows={4}
116:               className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:outline-none focus:ring-2
117:               placeholder="Write your feedback here..."
118:               value={feedbackText}
119:               onChange={(e) => setFeedbackText(e.target.value)}
120:               required
121:             />
122:             <div className="flex justify-end">
123:               <button
124:                 type="submit"
125:                 className="bg-blue-600 text-white px-6 py-2 rounded-xl hover:bg-blue-700 transition"
126:               >
127:                 Submit Feedback
128:               </button>
129:             </div>
130:           </form>
131:           {feedbackMessage && (
132:             <div className="mt-3 text-sm text-center text-green-600">
133:               {feedbackMessage}
134:             </div>
135:           )}
136:         </section>
137:       )}
138:     </div>
139:   );
140: }
```

■ File: src\pages\ItemRegisterPage.js

```
  1: import { useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3: import { CameraIcon } from "lucide-react";
  4: import toast, { Toaster } from "react-hot-toast";
  5:
  6: export default function RegisterItemPage() {
  7:   const navigate = useNavigate();
  8:   const [formData, setFormData] = useState({
```

```
    name: "",
    description: "",
    location: "",
    status: "lost",
  });
  const [image, setImage] = useState(null);
  const [previewUrl, setPreviewUrl] = useState(null);

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleFileChange = (e) => {
    const file = e.target.files[0];
    if (file) {
      setImage(file);
      setPreviewUrl(URL.createObjectURL(file));
    }
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!image) {
      toast.error("Please select an image file.");
      return;
    }

    const data = new FormData();
    Object.entries(formData).forEach(([key, val]) => data.append(key, val));
    data.append("image", image);

    try {
      const res = await fetch("http://localhost:5000/api/items", {
        method: "POST",
        body: data,
        credentials: "include",
      });

      if (res.ok) {
        toast.success("Item registered successfully!");
        setTimeout(() => navigate("/lost-found-items"), 2000);
      } else {
        const err = await res.json();
        toast.error(err.error || "Failed to register item.");
      }
    } catch (err) {
      console.error(err);
      toast.error("Something went wrong. Please try again.");
    }
  };

  return (
    <div className="max-w-3xl mx-auto px-4 py-10">
      {/* Toast container */}
      <Toaster position="top-center" reverseOrder={false} />

      <div className="bg-white shadow-xl rounded-xl p-8 border">
        <div className="text-center mb-6">
          <h2 className="text-3xl font-bold text-blue-600 flex justify-center items-center gap-2">
            <CameraIcon className="w-6 h-6" />
            Register Lost / Found Item
          </h2>
          <p className="text-sm text-gray-500 mt-1">
            Submit the details to help match items with their rightful owners.
          </p>
        </div>

        <form onSubmit={handleSubmit} className="space-y-5">
          {/* Item Name */}
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Item Name
```

```
            </label>
            <input
              type="text"
              name="name"
              value={formData.name}
              onChange={handleChange}
              required
              className="w-full border px-4 py-2 rounded focus:ring-2 focus:ring-blue-500 focus:outline-none
              placeholder="e.g., Black Wallet"
            />
          </div>

          {/* Description */}
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Description
            </label>
            <textarea
              name="description"
              value={formData.description}
              onChange={handleChange}
              required
              rows={4}
              className="w-full border px-4 py-2 rounded focus:ring-2 focus:ring-blue-500 focus:outline-none
              placeholder="Provide details like color, size, features..."
            />
          </div>

          {/* Location */}
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Location
            </label>
            <input
              type="text"
              name="location"
              value={formData.location}
              onChange={handleChange}
              required
              className="w-full border px-4 py-2 rounded focus:ring-2 focus:ring-blue-500 focus:outline-none
              placeholder="Where was it found/lost?"
            />
          </div>

          {/* Status */}
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Status
            </label>
            <select
              name="status"
              value={formData.status}
              onChange={handleChange}
              className="w-full border px-4 py-2 rounded focus:ring-2 focus:ring-blue-500 focus:outline-none
            >
              <option value="lost">Lost</option>
              <option value="found">Found</option>
            </select>
          </div>

          {/* Image Upload */}
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Upload Image
            </label>
            <input
              type="file"
              accept="image/*"
              onChange={handleFileChange}
              required
              className="w-full"
            />
          </div>
```

```
155:
156:           {/* Preview */}
157:           {previewUrl && (
158:             <div className="mt-4">
159:               <label className="block text-sm font-medium text-gray-700 mb-1">
160:                 Preview
161:               </label>
162:               <img
163:                 src={previewUrl}
164:                 alt="Preview"
165:                 className="w-full max-h-[400px] object-contain border rounded shadow"
166:               />
167:             </div>
168:           )}
169:
170:           {/* Submit Button */}
171:           <div>
172:             <button
173:               type="submit"
174:               className="w-full bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2 px-4 rounded tra
175:             >
176:               Register Item
177:             </button>
178:           </div>
179:         </form>
180:       </div>
181:     </div>
182:   );
183: }
```

■ File: src\pages\ItemsPage.js

```
  1: import { useEffect, useState } from "react";
  2: import toast, { Toaster } from "react-hot-toast";
  3: import dayjs from "dayjs";
  4: import customParseFormat from "dayjs/plugin/customParseFormat";
  5: import utc from "dayjs/plugin/utc";
  6: import timezone from "dayjs/plugin/timezone";
  7:
  8: dayjs.extend(customParseFormat);
  9: dayjs.extend(utc);
 10: dayjs.extend(timezone);
 11:
 12: // ? Fixed date formatter for MySQL datetime format
 13: function formatDateIST(mysqlDateStr) {
 14:   if (!mysqlDateStr) return "Invalid date";
 15:   return dayjs(mysqlDateStr, "YYYY-MM-DD HH:mm:ss").format(
 16:     "DD MMM YYYY, hh:mm"
 17:   );
 18: }
 19:
 20: // Alternative: Simple local time version (if you want user's timezone)
 21: function formatDateLocal(mysqlDateStr) {
 22:   if (!mysqlDateStr) return "Invalid date";
 23:
 24:   let date;
 25:   if (typeof mysqlDateStr === "string") {
 26:     date = dayjs(mysqlDateStr, "YYYY-MM-DD HH:mm:ss");
 27:   } else {
 28:     date = dayjs(mysqlDateStr);
 29:   }
 30:
 31:   if (date.isValid()) {
 32:     return date.format("DD MMM YYYY, hh:mm A");
 33:   }
 34:
 35:   return `Invalid date: ${mysqlDateStr}`;
 36: }
 37:
 38: export default function ItemsPage() {
 39:   const [items, setItems] = useState([]);
```

```
  const [search, setSearch] = useState("");
  const [filter, setFilter] = useState("all");
  const [currentUser, setCurrentUser] = useState(null);

  const [showMessageBox, setShowMessageBox] = useState(false);
  const [showClaimModal, setShowClaimModal] = useState(false);
  const [currentItem, setCurrentItem] = useState(null);
  const [messageText, setMessageText] = useState("");

  useEffect(() => {
    fetchUser();
    fetchItems();
  }, []);

  useEffect(() => {
    const interval = setInterval(() => {
      fetchItems();
    }, 60000);
    return () => clearInterval(interval);
  }, []);

  const fetchUser = async () => {
    try {
      const res = await fetch("http://localhost:5000/api/user", {
        credentials: "include",
      });
      if (res.ok) {
        const data = await res.json();
        setCurrentUser(data);
      }
    } catch (err) {
      console.error("Failed to fetch user");
    }
  };

  const fetchItems = async () => {
    try {
      const res = await fetch("http://localhost:5000/api/items", {
        credentials: "include",
      });
      if (!res.ok) throw new Error("Failed to fetch items");
      const data = await res.json();
      setItems(data.items || []);
    } catch (err) {
      console.error(err);
    }
  };

  const handleClaim = async (itemId) => {
    try {
      const res = await fetch(
        `http://localhost:5000/api/items/${itemId}/claim`,
        {
          method: "POST",
          credentials: "include",
        }
      );
      if (res.ok) {
        toast.success("Item claimed successfully!");
        fetchItems();
      } else {
        const data = await res.json();
        toast.error(`${data.error || "Failed to claim item."}`);
      }
    } catch (err) {
      console.error("Claim failed", err);
      toast.error("Something went wrong.");
    }
  };

  const handleSendMessage = async () => {
    if (!currentItem || !messageText.trim()) return;

```

```
113:     try {
114:       const res = await fetch("http://localhost:5000/api/messages", {
115:         method: "POST",
116:         headers: { "Content-Type": "application/json" },
117:         credentials: "include",
118:         body: JSON.stringify({
119:           receiver_id: currentItem.created_by,
120:           item_id: currentItem.id,
121:           message: messageText.trim(),
122:         }),
123:       });
124:
125:       if (res.ok) {
126:         toast.success("Message sent!");
127:       } else {
128:         const err = await res.json();
129:         toast.error(`${err.error || "Failed to send message."}`);
130:       }
131:     } catch (err) {
132:       console.error("Message failed", err);
133:       toast.error("Something went wrong.");
134:     } finally {
135:       setShowMessageBox(false);
136:       setMessageText("");
137:       setCurrentItem(null);
138:     }
139:   };
140:
141:   const filteredItems = items.filter((item) => {
142:     const searchText = search.toLowerCase();
143:     const matchesSearch =
144:       item.name.toLowerCase().includes(searchText) ||
145:       item.location.toLowerCase().includes(searchText) ||
146:       item.description.toLowerCase().includes(searchText);
147:     const matchesFilter =
148:       filter === "all" ? true : item.status.toLowerCase() === filter;
149:     return matchesSearch && matchesFilter;
150:   });
151:
152:   return (
153:     <div className="max-w-6xl mx-auto p-6 space-y-6">
154:       <Toaster position="top-center" reverseOrder={false} />
155:       <h1 className="text-2xl font-bold text-blue-600 text-center">
156:         Lost and Found Items
157:       </h1>
158:
159:       <div className="flex flex-col sm:flex-row gap-4 justify-between items-center mt-4">
160:         <input
161:           type="text"
162:           placeholder="Search by name, location, or description..."
163:           className="w-full px-4 py-2 border rounded"
164:           value={search}
165:           onChange={(e) => setSearch(e.target.value)}
166:         />
167:         <select
168:           value={filter}
169:           onChange={(e) => setFilter(e.target.value)}
170:           className="px-4 py-2 border rounded"
171:         >
172:           <option value="all">All</option>
173:           <option value="lost">Lost</option>
174:           <option value="found">Found</option>
175:           <option value="claimed">Claimed</option>
176:         </select>
177:       </div>
178:
179:       <div className="grid gap-6 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 mt-4">
180:         {filteredItems.map((item) => (
181:           <div
182:             key={item.id}
183:             className="border rounded-lg overflow-hidden shadow bg-white flex flex-col"
184:           >
185:             {item.image && (
```

```
              <img
                src={`http://localhost:5000/static/${item.image}`}
                alt={item.name}
                className="w-full h-48 object-cover"
              />
            )}
            <div className="p-4 space-y-2 flex-1 flex flex-col justify-between">
              <div>
                <h2 className="text-lg font-semibold">{item.name}</h2>
                <p className="text-sm text-gray-500">{item.location}</p>
                <p className="text-sm">{item.description}</p>
              </div>

              <div className="text-xs text-gray-500 mt-1">
                <p>
                  Posted by: <strong>{item.creator_name}</strong>
                </p>
                {/* <p>
                  On: <span>{formatDateIST(item.created_at)}</span>
                </p> */}
                {/* Debug info - remove this after fixing */}
                <p className="text-black-500 text-xs">
                  Posted on: {JSON.stringify(item.created_at)}
                </p>
              </div>

              <div className="mt-2">
                <span
                  className={`inline-block text-xs px-2 py-1 rounded ${
                    item.status === "found"
                      ? "bg-green-100 text-green-700"
                      : item.status === "lost"
                      ? "bg-yellow-100 text-yellow-700"
                      : "bg-blue-100 text-blue-700"
                  }`}
                >
                  {item.status.toUpperCase()}
                </span>
              </div>

              <div className="mt-3 flex gap-2">
                {item.status === "claimed" ? (
                  <p className="text-sm text-red-700 mt-1">
                    This item has been claimed
                  </p>
                ) : currentUser && item.created_by === currentUser.id ? (
                  <p className="text-sm text-gray-700 italic">
                    You posted this item
                  </p>
                ) : item.status === "found" ? (
                  <>
                    <button
                      onClick={() => {
                        setCurrentItem(item);
                        setShowClaimModal(true);
                      }}
                      className="w-full text-sm bg-blue-600 text-white py-1.5 rounded hover:bg-blue-700"
                    >
                      Claim Item
                    </button>
                    <button
                      onClick={() => {
                        setCurrentItem(item);
                        setShowMessageBox(true);
                      }}
                      className="w-full text-sm border border-gray-700 text-gray-800 py-1.5 rounded hover:b
                    >
                      <i className="fas fa-comment mr-1"></i> Message
                    </button>
                  </>
                ) : item.status === "lost" ? (
                  <button
                    onClick={() => {
```

```
                      setCurrentItem(item);
                      setShowMessageBox(true);
                    }}
                    className="w-full text-sm border border-gray-700 text-gray-800 py-1.5 rounded hover:bg-
                  >
                    <i className="fas fa-comment mr-1"></i> Message
                  </button>
                ) : null}
              </div>
            </div>
          </div>
        ))}
      </div>

      {filteredItems.length === 0 && (
        <p className="text-center text-gray-500 mt-10">No items found.</p>
      )}

      {showMessageBox && currentItem && (
        <div className="fixed inset-0 bg-black bg-opacity-40 px-1 flex items-center justify-center z-50">
          <div className="bg-white p-6 rounded shadow w-full max-w-md space-y-4">
            <h3 className="text-lg font-bold text-blue-600">Send Message</h3>
            <p className="text-sm text-gray-600">
              To: <strong>{currentItem.creator_name}</strong> <br />
              Item: <strong>{currentItem.name}</strong>
            </p>
            <textarea
              rows={4}
              className="w-full border rounded px-3 py-2"
              placeholder="Type your message..."
              value={messageText}
              onChange={(e) => setMessageText(e.target.value)}
            />
            <div className="flex justify-end gap-2">
              <button
                onClick={() => setShowMessageBox(false)}
                className="text-sm px-4 py-1 bg-gray-300 rounded hover:bg-gray-400"
              >
                Cancel
              </button>
              <button
                onClick={handleSendMessage}
                className="text-sm px-4 py-1 bg-blue-600 text-white rounded hover:bg-blue-700"
              >
                Send
              </button>
            </div>
          </div>
        </div>
      )}

      {showClaimModal && currentItem && (
        <div className="fixed inset-0 bg-black bg-opacity-40 flex items-center justify-center z-50">
          <div className="bg-white p-6 rounded shadow w-full max-w-md space-y-4">
            <div className="flex justify-between items-center">
              <h3 className="text-lg font-bold text-blue-600">
                Claim Item: {currentItem.name}
              </h3>
            </div>
            <p className="text-sm text-gray-600">
              Your registered name and email will be used for this claim.
            </p>
            <div className="flex justify-end gap-2">
              <button
                onClick={() => setShowClaimModal(false)}
                className="text-sm px-4 py-1 border rounded"
              >
                Cancel
              </button>
              <button
                onClick={() => {
                  handleClaim(currentItem.id);
                  setShowClaimModal(false);
```

```
                }}
                className="text-sm px-4 py-1 bg-blue-600 text-white rounded hover:bg-blue-700"
              >
                Confirm Claim
              </button>
            </div>
          </div>
        </div>
      )}
    </div>
  );
}
```

■ File: src\pages\LoginPage.js

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function LoginPage() {
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");

  const handleLogin = async (e) => {
    e.preventDefault();
    setError("");

    try {
      const res = await fetch("http://localhost:5000/api/login", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        credentials: "include",
        body: JSON.stringify({ email, password }),
      });

      const data = await res.json();

      if (!res.ok || !data.success) {
        setError(data.error || "Invalid email or password.");
        return;
      }

      // ? Redirect based on user role
      if (data.user?.role === "admin") {
        navigate("/admin");
      } else {
        navigate("/home");
      }
    } catch (err) {
      console.error("Login error:", err);
      setError("Login failed. Please try again.");
    }
  };

  return (
    <div className="flex items-center justify-center min-h-screen bg-black-100 text-black-100 px-4">
      <div className="bg-white p-8 rounded-lg shadow-md w-full max-w-md">
        <h2 className="text-2xl font-bold text-center text-blue-600 mb-6">
          Login
        </h2>

        {error && (
          <div className="bg-red-100 text-red-700 px-3 py-2 mb-4 rounded text-sm text-center">
            {error}
          </div>
        )}

        <form onSubmit={handleLogin} className="space-y-4">
```

```
 57:           <input
 58:             type="email"
 59:             placeholder="Email"
 60:             className="w-full px-4 py-2 border rounded-md"
 61:             value={email}
 62:             onChange={(e) => setEmail(e.target.value)}
 63:             required
 64:           />
 65:           <input
 66:             type="password"
 67:             placeholder="Password"
 68:             className="w-full px-4 py-2 border rounded-md"
 69:             value={password}
 70:             onChange={(e) => setPassword(e.target.value)}
 71:             required
 72:           />
 73:           <button
 74:             type="submit"
 75:             className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700 transition"
 76:           >
 77:             Login
 78:           </button>
 79:         </form>
 80:       </div>
 81:     </div>
 82:   );
 83: }
```

■ File: src\pages\RegisterPage.js

```
  1: import { useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3:
  4: export default function RegisterPage() {
  5:   const navigate = useNavigate();
  6:   const [formData, setFormData] = useState({
  7:     name: "",
  8:     email: "",
  9:     password: "",
 10:     confirm_password: "",
 11:   });
 12:   const [error, setError] = useState("");
 13:   const [success, setSuccess] = useState("");
 14:
 15:   const handleChange = (e) => {
 16:     setFormData((prev) => ({
 17:       ...prev,
 18:       [e.target.name]: e.target.value,
 19:     }));
 20:   };
 21:
 22:   const handleSubmit = async (e) => {
 23:     e.preventDefault();
 24:     setError("");
 25:     setSuccess("");
 26:
 27:     if (formData.password !== formData.confirm_password) {
 28:       setError("Passwords do not match.");
 29:       return;
 30:     }
 31:
 32:     try {
 33:       const res = await fetch("http://localhost:5000/api/register", {
 34:         method: "POST",
 35:         headers: {
 36:           "Content-Type": "application/json",
 37:         },
 38:         credentials: "include",
 39:         body: JSON.stringify(formData),
 40:       });
 41:
```

```
      const data = await res.json();

      if (!res.ok || !data.success) {
        setError(data.error || "Registration failed.");
        return;
      }

      setSuccess("Registration successful! Redirecting to login...");
      setTimeout(() => navigate("/"), 2000);
    } catch (err) {
      console.error("Registration error:", err);
      setError("Something went wrong. Please try again.");
    }
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 px-4">
      <div className="bg-white p-8 rounded-lg shadow-md w-full max-w-md">
        <h2 className="text-2xl font-bold text-center text-blue-600 mb-6">
          Create Account
        </h2>

        {error && (
          <div className="bg-red-100 text-red-700 p-2 rounded text-sm mb-4">
            {error}
          </div>
        )}

        {success && (
          <div className="bg-green-100 text-green-700 p-2 rounded text-sm mb-4">
            {success}
          </div>
        )}

        <form onSubmit={handleSubmit} className="space-y-4">
          <input
            type="text"
            name="name"
            placeholder="Full Name"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.name}
            required
          />
          <input
            type="email"
            name="email"
            placeholder="Email"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.email}
            required
          />
          <input
            type="password"
            name="password"
            placeholder="Password"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.password}
            required
          />
          <input
            type="password"
            name="confirm_password"
            placeholder="Confirm Password"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.confirm_password}
            required
          />
          <button
            type="submit"
```

```
115:             className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700 transition"
116:           >
117:             Register
118:           </button>
119:         </form>
120:       </div>
121:     </div>
122:   );
123: }
```

■ File: src\reportWebVitals.js

```
  1: const reportWebVitals = onPerfEntry => {
  2:   if (onPerfEntry && onPerfEntry instanceof Function) {
  3:     import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
  4:       getCLS(onPerfEntry);
  5:       getFID(onPerfEntry);
  6:       getFCP(onPerfEntry);
  7:       getLCP(onPerfEntry);
  8:       getTTFB(onPerfEntry);
  9:     });
 10:   }
 11: };
 12:
 13: export default reportWebVitals;
```

■ File: src\setupTests.js

```
  1: // jest-dom adds custom jest matchers for asserting on DOM nodes.
  2: // allows you to do things like:
  3: // expect(element).toHaveTextContent(/react/i)
  4: // learn more: https://github.com/testing-library/jest-dom
  5: import '@testing-library/jest-dom';
```

■ File: tailwind.config.js

```
  1: /** @type {import('tailwindcss').Config} */
  2: module.exports = {
  3:   content: [
  4:     "./src/**/*.{js,jsx,ts,tsx}", // very important for React
  5:   ],
  6:   theme: {
  7:     extend: {},
  8:   },
  9:   plugins: [],
 10: };
```
