Security Management System for Strathmore University

(6-Week Project Plan)

1. Visitor Management System Module

Team Members: Lance Munyao, Cynthia Chemutai, Celestine Kariuki

---

📈 **Task Breakdown & Timelines (Updated)**

🗓️ Week 1: Backend Development (Laravel API & Database Setup)

Lead: Backend Developer

Tasks:

- Set up Laravel Project & PostgreSQL Database:
  - Install Laravel and configure the database connection with PostgreSQL.
  - Set up database migrations for the Visitor Table (fields like name, id_number, id_type, destination, appointment, vehicle_info, etc.).
- Develop Visitor Registration API:
  - Create the API endpoint (POST /register-visitor) to handle visitor registration with necessary validations (name, ID type, ID number, destination, etc.).
- Generate Visitor Tag (PDF + QR Code):
  - Use Laravel DomPDF to generate PDF tags.
  - Implement QR code generation (using Laravel QR code package) that encodes visitor data for easy scanning.
- Check-In / Check-Out Logic:
  - Develop API to mark visitors as "checked-in" and "checked-out."
  - Store timestamps for check-in/check-out actions in the database.

Expected Deliverables:

- Visitor Registration API.
- Visitor Tag PDF generation (with QR code).
- Check-in / Check-out logic working in the backend.

🗓️ Week 2: Backend Enhancements & Initial Testing

Lead: Backend Developer

Tasks:

- Optimize API endpoints for better performance.
- Implement initial unit testing for backend components.
- Ensure secure API communication.
- Fix bugs identified during development.
- Documentation for backend logic and API usage.

Expected Deliverables:

- Optimized backend API.
- Basic backend documentation.
- Initial test cases.

🗓️ Week 3: Frontend Development (ShadCN UI with React)

Lead: Frontend Developer

Tasks:

- Set up React Project & ShadCN UI Library:
    - Install ShadCN UI components to quickly build out the UI.
    - Configure project with necessary dependencies (e.g., Axios for API calls, React Router for navigation).
- Visitor Registration Form:
    - Create a form UI using ShadCN that includes fields for:
        - Full Name
        - ID Type & ID Number
        - Destination
        - Appointment Details
        - Vehicle Details (if applicable)
    - Integrate API Calls:
        - Implement the API call to the backend (POST /register-visitor) on form submission.
        - Handle form validation, errors, and success messages.
- Display Visitor List:
    - Create a Visitor List Page that displays all registered visitors.
    - Include a search feature (by name or ID number).
    - Add a PDF download button for visitor tags linking to the backend API endpoint.

Expected Deliverables:

- Visitor Registration Form.
- Visitor List Page with search and download functionality.
- Full API integration between frontend and backend.

🗓️ Week 4: Frontend Enhancements & Testing

Lead: Frontend Developer

Tasks:

- Conduct UI/UX testing for form validation and responsiveness.
- Enhance visitor list pagination and sorting features.
- Refine form error handling and feedback messages.
- Debug frontend-backend communication issues.
- Improve overall user interface design.

Expected Deliverables:

- Enhanced visitor registration UI.
- Improved visitor list with additional features.
- Bug-free frontend system.

🗓️ Week 5: Security, Deployment, and Role-Based Authentication

Lead: Full-Stack Developer

Tasks:

- Implement Role-Based Authentication:
  - Secure pages like Visitor Registration and Visitor List to authorized staff only.
  - Use Laravel Passport or Sanctum for authentication.
- Performance Optimization:
  - Ensure fast API responses for visitor registration and check-in/out.
  - Optimize database queries and pagination for the visitor list.
- Deployment Setup:
  - Set up the self-hosted server (e.g., DigitalOcean VPS or AWS EC2).
  - Deploy the Laravel backend and React frontend.
  - Configure environment variables for production (database credentials, API keys, etc.).
  - Set up Nginx/Apache to handle Laravel requests and React static files.

Expected Deliverables:

- Secure and authenticated system.
- Fully deployed backend and frontend.

🗓️ Week 6: Final Testing, Documentation, and Training

Lead: Full-Stack Developer

Tasks:

- Comprehensive Testing:
  - Final round of unit and integration testing for all system components.
  - Performance and stress testing.
- Bug Fixes:
  - Resolve all identified issues.
- User Documentation:
  - Provide detailed user documentation for the security team.
  - Include troubleshooting tips and best practices.
- Training:
  - Conduct an onboarding session for security guards on system usage (visitor registration, tag verification, etc.).

Expected Deliverables:

- Fully functional and tested Visitor Management System.
- Comprehensive user guide.
- Successful user training session.

---

2. Incident Detection & Response Module

Team Members: David Ngahu, Makenna Wahu, RickCharles Muchira

📌 Task Breakdown & Timelines (Updated)

🗓️ Week 1 – Backend Development (Laravel API & DB Setup)

Lead: Backend Developer

📌 Tasks:

- Set up Laravel Project & PostgreSQL Database
- Configure database connection with PostgreSQL
- Set up database migrations for the **Incident Table** (fields: incident_ID, id_number, id_type, incident_type, incident_description_short, incident_description_detailed, evidence_attachment, incident_location, escalate_to, settled_by, closed, etc.)
- Develop **Incident Reporting API** logic with autogenerated timestamps
- Implement error handling for incorrect data input

📌 Additional Tasks:

- Implement API authentication using **Laravel Sanctum or Passport**
- Develop API endpoints for:
- **Registering an Incident** (POST /registerincident)
- **Fetching Incident List** (GET /incidents)
- **Fetching Incident by ID** (GET /incident/{id})
- Write API documentation using **Swagger (OpenAPI 3.0) or Postman Collection**

📌 **Expected Deliverables:**

- Cloned required repository
- Laravel & PostgreSQL setup with migrations
- Incident API functional with authentication and validation
- API documentation available

🗓️ Week 2 – Frontend Development (React with ShadCN UI)

Lead: Frontend Developer

📌 **Tasks:**

- Set up **React Project & ShadCN UI Library**
- Configure **Axios** for API calls and **React Router** for navigation
- Develop **Incident Registration Form** with the following fields: Incident_ID, Persons Involved, id_number, id_type, incident_type, incident_description_short, incident_description_detailed, evidence_attachment, incident_location, escalate_to, settled_by, closed, etc.
- Implement **API call integration** with Laravel backend (POST /registerincident)
- Validate form input and handle errors

📌 **Additional Tasks:**

- Implement **userfriendly notifications** for successful or failed form submissions
- Develop a **loading indicator and error messages** for API calls
- Create a **userfriendly sidebar navigation menu**

📌 **Expected Deliverables:**

- Fully functional **Incident Registration Form**
- API successfully integrated with **frontend**
- Improved **UI/UX for incident reporting**

🗓️ Week 3 – Incident List & Search Functionality

📌 **Tasks:**

- Develop the **Incident List Page** with:
    - List of **all registered incidents**
    - **Filter and search functionality** (by ID Number, Incident Location, Incident Type, Time Reported, Incident_ID, Person Name)

- Add a **PDF download button** for incident reports
- Implement **pagination for incident records**

📌 **Additional Tasks:**

- Improve **frontend UI components** to enhance visibility of active vs. resolved incidents
- Implement **date range filter** for incident history

📌 **Expected Deliverables:**

- Working **Incident List Page** with filtering and searching
- **PDF download feature** for reports

🗓️ Week 4 – Testing & Security Enhancements

📌 **Tasks:**

- Conduct **unit testing** on API endpoints and frontend form submissions
- Test **rolebased authentication** (restrict securityrelated pages to authorized personnel)
- Implement **CSRF protection, input validation, and security best practices**

📌 **Additional Tasks:**

- Optimize **database queries and API response time**
- Simulate **attack scenarios (SQL injection, XSS, CSRF)** and harden security
- Add **logging functionality** to track API requests

📌 **Expected Deliverables:**

- Fully tested API and UI components
- Security features implemented and tested

🗓️ Week 5 – Deployment & System Optimization

📌 **Tasks:**

- Set up **selfhosted server environment**
- Deploy the **Laravel API & PostgreSQL database**
- Deploy **React frontend on a production server**
- Configure **Nginx/Apache** for handling backend and frontend requests
- Set up **environment variables (database credentials, API keys, CORS settings, etc.)**

📌 **Additional Tasks:**

- Implement **API rate limiting** for protection against excessive requests
- Optimize **incident report generation performance**

📌 **Expected Deliverables:**

- Fully deployed and accessible **Incident Detection & Response System**

🗓️ Week 6 – Training, User Documentation & Final Testing

📌 **Tasks:**

- Create user documentation for security guards
- Conduct training sessions on:
- Incident registration workflow
- Incident searching & filtering
- Report generation & PDF export
- Perform final system testing & debugging

📌 **Expected Deliverables:**

- User guide/manual for security personnel
- Trained security team ready to use the system
- Fully functional Incident Detection & Response System 🎯

Summary of Improvements in the 6 Week Plan

- More **structured development phases** for smooth progress
- Security, testing, and performance optimization **spread over multiple weeks**
- Addition of **training and documentation** to ensure successful adoption
- Focus on **UI/UX improvements** for better user experience

3.Car Sticker Issuance Module

Week 1: Initial Setup & Database Design
Goal: Set up the foundational structure of the application and the database.

1. **Database Design:**
    - Define tables for user creation and management with constraints (userNumber, userEmail, userPhone).

    - Set up the car registration table with attributes (e.g., car model, year, owner), ensuring unique constraints on No_plate.
    - Create the sticker table with fields such as shape, color, and expirationDate, and enforce constraints linking stickers to users.
2. **Backend Setup:**
    - Set up the Laravel backend structure.
    - Define routes, controllers, and migrations for users, cars, and stickers.
3. **Application Framework:**
    - Set up React and Inertia.js for smooth data handling.

End of Week 1 Outcome:

- User, car, and sticker tables created.
- Basic authentication and registration ready.

Week 2: User Registration & Car Registration
Goal: Develop user authentication and car registration features.

1. **User Registration & Authentication:**
    - Implement signup, login, and password reset functionality.
    - Set up role-based access control (regular users vs. security users).
    - Validate user inputs such as email, phone, and unique identifiers.
2. **Car Registration:**
    - Build a user-friendly car registration form.
    - Validate No_plate values and ensure mandatory fields are filled.
    - Establish relationships between users and their cars.

End of Week 2 Outcome:

- Functional user registration and login.
- Car registration linked to users.

Week 3: Sticker Application & Issuance Workflow
Goal: Build the workflow for sticker application and management.

1. **Sticker Application:**
    - Create forms for sticker application (attributes like shape, color, expirationDate).
    - Enforce constraints linking stickers to user numbers.
2. **Sticker Workflow:**
    - Implement statuses: Pending, Approved, and Collected.
    - Business logic for approving/rejecting stickers.

End of Week 3 Outcome:

- Sticker application and status tracking functional.
- Backend logic for sticker management implemented.

Week 4: Admin Panel & Security User Functions
Goal: Build the admin panel and enable security functions.

1. **Admin Panel:**
    - Develop an interface where security users can approve/reject sticker applications.
    - Allow search and tracking of sticker progress.
2. **Security User Functions:**
    - Implement role-based features for security users.
3. **Database Logic:**
    - Implement queries for updating sticker statuses.

End of Week 4 Outcome:

- Security users can manage sticker applications via the Admin Panel.

Week 5: Expired Stickers, Notifications & User Dashboard
Goal: Handle expired stickers and build user dashboards with notification systems.

1. **Expired Sticker Management:**
    - Automatically mark expired stickers.
2. **User Dashboard:**
    - Display registered cars and sticker statuses.
3. **Notification System:**
    - Set up email or in-app notifications for status updates.

End of Week 5 Outcome:

- Expired stickers marked automatically.
- Functional user dashboard and notifications.

Week 6: Testing, Refinements & Deployment
Goal: Conduct testing and prepare for deployment.

1. **Database Testing:**
    - Verify CRUD operations for users, cars, and stickers.
2. **Workflow Testing:**
    - Test sticker issuance workflows.

3. **User Interface Testing:**
    - Ensure forms and dashboards function properly.
4. **Final Refinements:**
    - Address bugs and improve performance.
5. **Deployment:**
    - Prepare the system for deployment on the server.

End of Week 6 Outcome:

- Comprehensive testing complete.
- All features are functional.
- Ready for deployment.

SECURITY SYSTEM GANTT CHART