Visitor Log Milestones

Milestone Breakdown

Milestone 1: Project Setup & User Management

Objective: Initialize the Django project and implement user roles (Admin, Resident, Security Guard).

Steps:

- Create a Django project and apps (users, visits, reports, notifications).
- Set up a PostgreSQL database and configure Django settings.
- Create a CustomUser model with a role field (choices: Admin, Resident, Security).
- Use Django Admin for user management (CRUD operations for Admins).
- Add unit tests for user creation and role assignments.

Deliverables:

- Functional Django project with user roles.
- Admin interface to manage users.

Milestone 2: Authentication & Authorization

Objective: Implement role-based access control.

Steps:

- Build login, logout, and password reset views.
- Create templates for authentication pages.
- Use Django groups/permissions or decorators (e.g., @admin_required) for role-based access.
- Test access restrictions for unauthorized roles.

Deliverables:

- Secure login/logout functionality.
- Role-based redirection (e.g., Admins to dashboard, Residents to approval page).

Milestone 3: Visitor Request & Approval System

Objective: Allow Security Guards to register visitors and Residents to approve/reject requests.

Steps:

- Create a Visit model:
  - Fields: resident (FK to User), visitor_name, purpose, status (Pending/Approved/Rejected), requested_by (Security Guard), entry_time, exit_time.
- Build forms and views for Security Guards to create visitor requests.
- Create a Resident dashboard to view and approve/reject pending requests.
- Add email/SMS notifications for new requests (use Django signals or Celery).

Deliverables:

- Security Guards can submit visitor requests.

- Residents can approve/reject requests via a dashboard.

Milestone 4: Visit Logging (Entry/Exit)
Objective: Track visitor entry and exit times.

Steps:
- Add entry_time and exit_time fields to the Visit model.
- Create a Security Guard interface to log entry/exit (update Visit instances).
- Validate approvals before allowing entry (e.g., check status=Approved).

Deliverables:
- Security Guards can log visitor entry/exit.
- Residents see entry/exit times in their visitor logs.

Milestone 5: Reporting & Blacklisting
Objective: Enable reporting of suspicious visitors and blacklist management.

Steps:
- Create a Report model (reported_by, visit, reason).
- Create a Blacklist model (visitor_info, reason, blacklisted_by).
- Build forms for Residents/Security to report visitors.
- Add Admin interface to view reports and blacklist visitors.

Deliverables:
- Residents/Security can report suspicious visitors.
- Admins can blacklist visitors and view reports.

Milestone 6: Notification System
Objective: Notify Residents when visitors arrive.

Steps:
- Create a Notification model (user, message, timestamp, is_read).
- Use Django signals to trigger notifications when:
  - A visitor request is created.
  - A visitor's entry is logged.
- Add real-time alerts using WebSocket (Django Channels) or polling.

Deliverables:
- Residents receive in-app/email alerts for visitor arrivals.

Milestone 7: Admin Reporting & Analytics
Objective: Generate reports and analytics for Admins.

Steps:
- Build an Admin dashboard with charts (e.g., visitor trends, peak hours).

- Use libraries like Pandas/Chart.js for data visualization.
- Add CSV/PDF export functionality for reports.

Deliverables:
- Admins can view analytics and export reports.

Milestone 8: Security Guard Interface Enhancements
Objective: Streamline Security Guard workflows.

Steps:
- Add a visitor history lookup feature (search by name/date).
- Allow Security Guards to flag unauthorized visitors (auto-generate reports).
- Optimize UI for quick entry/exit logging.

Deliverables:
- Security Guards can search visitor history and flag issues.

Milestone 9: Testing & Deployment
Objective: Ensure stability and deploy the project.

Steps:
- Write unit/integration tests (e.g., approval flow, role permissions).
- Dockerize the project for deployment.
- Deploy using Gunicorn + Nginx on a cloud server (AWS/Azure).
- Create user documentation and admin guides.

Deliverables:
- Fully tested, deployed system with documentation.

Timeline

Milestone	Estimated Time
1	3-5 days
2	2-3 days
3	5-7 days
4	3-5 days
5	4-6 days
6	5-7 days
7	4-6 days
8	3-5 days
9	5-7 days

Tools & Technologies
- Backend: Django, Django REST Framework (optional for APIs).

- Database: PostgreSQL.
- Frontend: Bootstrap/jQuery or React (if SPA needed).
- Real-Time: Django Channels for WebSocket.
- Deployment: Docker, Gunicorn, Nginx.

Risk Mitigation
- Use Django's built-in security features (CSRF, XSS protection).
- Test role permissions rigorously to prevent unauthorized access.
- Implement rate-limiting for authentication endpoints.
- Backup databases regularly during development.