Cybercrime Incident Management & Awareness System (CIMAS)

Complete Project Documentation

Technical Documentation and System Architecture

CIMAS Development Team

October 20, 2025

Contents

Chapter 1

Project Setup Guide

1.1 Backend Setup

1.1.1 Setup Steps

1. **Clone the repository:**

```
git clone https://github.com/SinlessRook/Cybercrime-Incident-Management-and-Awareness-System.git
cd Cybercrime-Incident-Management-and-Awareness-System/backend
```

2. **Create and activate a Python virtual environment:**

```
python -m venv venv
.\venv\Scripts\activate
```

3. **Install required dependencies:**

```
cd CIMAS
pip install -r requirements.txt
```

4. **Configure environment variables:**

   Create a `.env` file in the backend directory with the following content:

```
DEBUG=True
SECRET_KEY='django-insecure-&ge1nt%6k$ieo8l5*%a52=(@n3&!%rcnym_n7%af4al*68@=8d'

# PostgreSQL local development environment variables
DB_ENGINE=django.db.backends.postgresql
DB_NAME=CIMAS
DB_USER=postgres
DB_PASSWORD=pass
DB_HOST=localhost
DB_PORT=5432
```

5. **Apply database migrations:**

```
python manage.py migrate
```

6. **Create a superuser (for admin access):**

```
python manage.py createsuperuser
```

7. **Run the backend server:**

```
python manage.py runserver
```

1.2 Frontend Setup

1.2.1 Requirements

- Node.js (v18+) and npm
- Vite (bundled via npm scripts)

1.2.2 Install and Run

```
cd ..\frontend
npm install
npm run dev   # start development server (Vite)
npm run build # production build
npm run preview
```

1.2.3 Environment Variables (optional)

Create a `.env` file in `frontend/` as needed:

```
VITE_API_BASE_URL=http://localhost:8000
```

Chapter 2

Database Structure

2.1 Overview

The system uses PostgreSQL with a Django ORM schema. Figure 2.1 shows the high-level ER diagram.

2.2 Key Tables

oprule extbfTable	**Purpose**
extttusers_customuser	Core user table with email, role (admin, investigator, victim), names, and profile fields.
extttincidents	Incident reports submitted by victims; links to `crime_types`, `locations`, and `users_customuser` (reporter).
extttincidents_incidentassignment	Assignment of incidents to investigators; includes priority, assigned/resolved timestamps, deadlines.
extttevidence	Evidence items per incident; file path, title, tags, uploader, submitted timestamp.
extttinvestigators	One-to-one mapping to user accounts with department metadata.
extttcrime_types	Master list of crime categories.
extttlocations	Normalized addresses for incidents.
extttactivity_log	System-level audit log of actions taken by users.
extttawareness_awarenessresource	Awareness/education content authored by admins.
extttsolutions	Recommended actions per crime type for analytics/awareness.

2.3 Notable Relationships

- **User** 1–N **Incidents**
- **Incident** 1–N **Evidence**
- **Incident** 1–N **IncidentAssignments** (latest is active)

- **User** 1–1 **Investigator**
- **Incident** – **CrimeTypes**, **Locations** (FK)

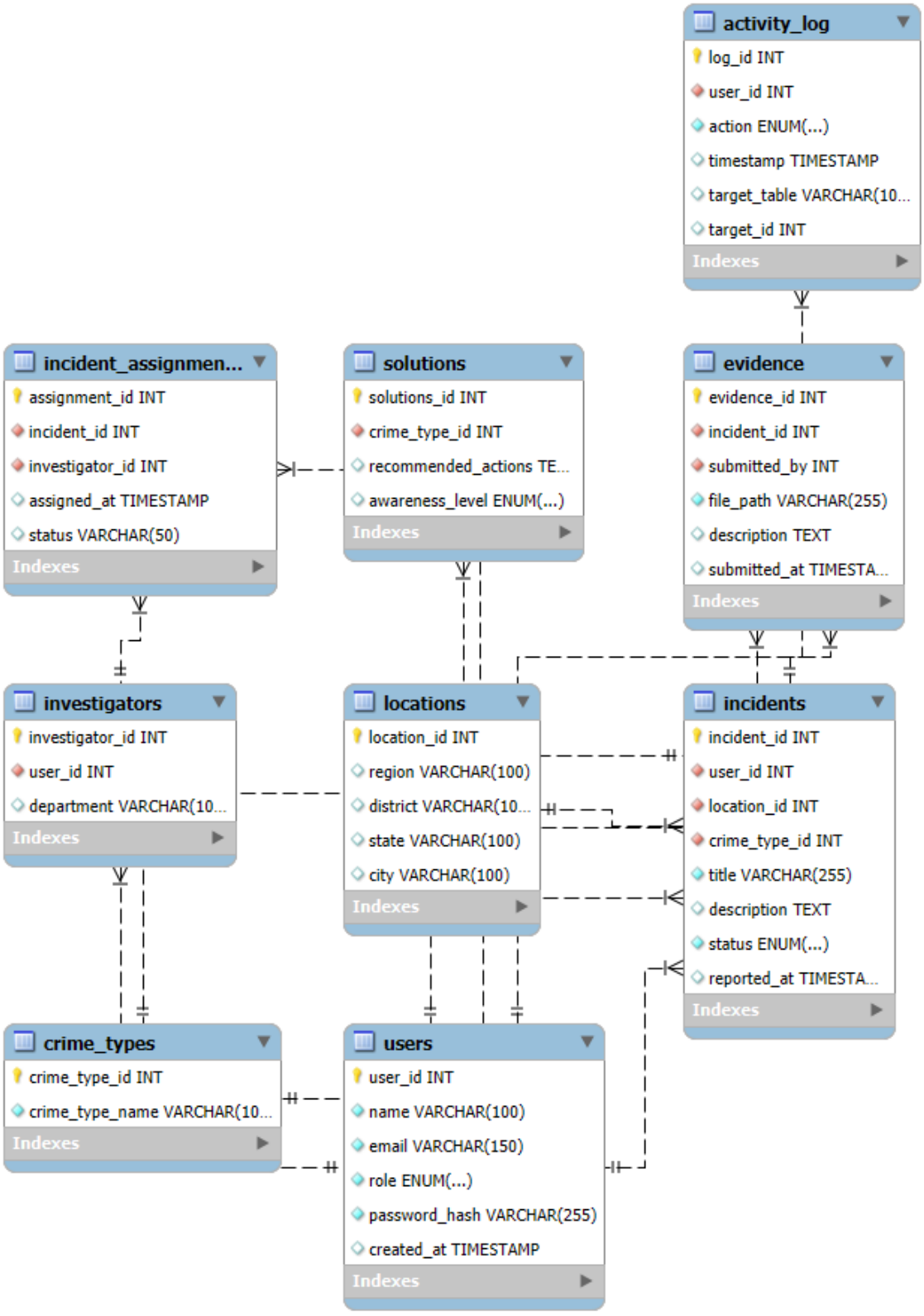

Figure 2.1: Entity-Relationship Diagram

Chapter 3

Frontend Components

3.1 Dashboard Summary

3.1.1 Completed Components

Messaging System (Standalone)

Location: `frontend/src/Messaging.jsx`

Role-Based Permissions:

- **Admin:**
  - Message all investigators
  - Message all victims
  - Broadcast to all users/investigators/victims
  - Individual messaging
- **Investigator:**
  - Message victims of assigned cases only
  - Message admin panel
- **Victim:**
  - Message assigned investigator
  - Message admin panel

Features:

- Split-screen chat interface
- Online/offline/away status indicators
- Message status (sent/delivered/read)
- Unread message badges

- Search conversations
- File attachment button
- Voice/Video call buttons
- Broadcast modal for admin
- Real-time timestamps

3.1.2 Victim Dashboard

Location: `frontend/src/components/Victim/`

VictimOverview.jsx - Dashboard Home

- 4 statistics cards (Active/In Progress/Resolved/Evidence)
- Active cases with progress bars
- Upcoming activities calendar
- Recent updates feed
- Quick action buttons

MyIncidents.jsx - Incident Management

- Stats summary (Total/In Progress/Resolved/Pending)
- Search and filter functionality
- Detailed incident cards with:
  - Status and priority badges
  - Progress tracking
  - Evidence counts
  - Investigator assignment
- Report new incident button
- Timeline modal with event tracking

CaseStatus.jsx - Case Progress Tracking

- Case selector dropdown
- Overall progress display (65%)
- Key metrics (Days Open, Evidence, Documents, Last Activity)
- 6-stage investigation timeline:
  1. Incident Reported

2. Case Assigned
3. Evidence Collection
4. Evidence Analysis
5. Investigation Report
6. Case Resolution

- Investigator contact card
- Important dates (Reported, Estimated Completion)
- Evidence files list with status
- Case documents repository
- Recent updates feed

Resources.jsx - Support & Information

- Emergency contacts banner (3 helplines)
- Featured resources section
- 6 category filters (Guides/Legal/Support/FAQ/Videos/All)
- Search functionality
- 8 comprehensive resources
- Resource preview modal
- Download tracking
- External links to organizations

3.1.3 Investigator Dashboard

Location: `frontend/src/components/Investigator/`

InvestigatorOverview.jsx

- Stats Cards (Total Cases, In Progress, Resolved, Success Rate)
- Performance Metrics
- Recent Activity Feed
- Upcoming Deadlines

MyCases.jsx

- Case Display with ID, title, and description
- Crime type classification
- Priority and status badges
- Progress tracking with visual bars
- Evidence and updates count
- Search and filter capabilities
- View Details and Quick Edit buttons

ActivityLog.jsx

- Activity Types (Case Updates, Evidence Uploads, Messages, etc.)
- Filter Tabs for different activity types
- Color-coded icons
- Timeline display
- Export feature

Evidence.jsx

- Evidence Display with file type icons
- Status badges (Verified/Pending/Rejected)
- Search and filter by case ID and file type
- View, download, and delete actions
- Upload new evidence button

Reports.jsx

- Report Types (Final, Progress, Preliminary)
- Status badges (Approved/Under Review/Draft/Rejected)
- Status Timeline
- Stats Dashboard
- View, download, edit, and submit actions

Messages.jsx

- Two-Panel Layout (conversations list and chat window)
- Message history with timestamps
- Read receipts
- File attachment capability
- Search across conversations

3.1.4 Admin Components

AssignCase Component

Location: `frontend/src/components/Admin/AssignCase/`

Component Structure:

- `index.js` - Barrel export file
- `mockData.js` - Mock data for cases and investigators
- `utils.js` - Utility functions
- `ViewModeTabs.jsx` - Tabs for switching modes
- `CaseCard.jsx` - Individual case display
- `InvestigatorCard.jsx` - Individual investigator display
- `SearchAndFilter.jsx` - Search and filter controls
- `AssignmentActionBar.jsx` - Floating action bar
- `SuccessMessage.jsx` - Success notification
- `InfoCard.jsx` - Information cards

Dual Mode Operation:

- **Assign Mode (Blue Theme):** Shows unassigned cases
- **Reassign Mode (Purple Theme):** Shows assigned cases

Smart Recommendations: The recommendation algorithm scores investigators based on:

- **Specialization match** (+50 points)
- **Workload** (+10-30 points)
- **Success rate** (+10-20 points)

3.2 Design System

3.2.1 Color Scheme

- **Background:** Black to gray-900 gradients
- **Borders:** Gray-800, Gray-700
- **Accents:** Blue-600, Cyan-500
- **Status Colors:**
  - Blue: In Progress
  - Yellow: Under Review / Medium Priority
  - Green: Resolved / Verified / Low Priority
  - Red: High Priority / Rejected
  - Gray: Pending

3.2.2 Typography

- **Headings:** Bold, 2xl-3xl
- **Body:** Regular, sm-base
- **Labels:** Medium, xs-sm

3.2.3 Components

- **Cards:** Rounded-lg, border, padding-6
- **Buttons:** Rounded-lg, hover effects, scale animations
- **Modals:** Backdrop blur, scale animations
- **Progress Bars:** Rounded-full, gradient fills

3.3 Pages by Role with Screenshots

3.3.1 Admin

- Assign Cases (recommendations, workload badges)
- Analytics Dashboard (summary, trends, hotspots, categories)
- Activity Logs (filters by user/incident)
- Awareness Content Management
- Messaging (broadcasts, all users)

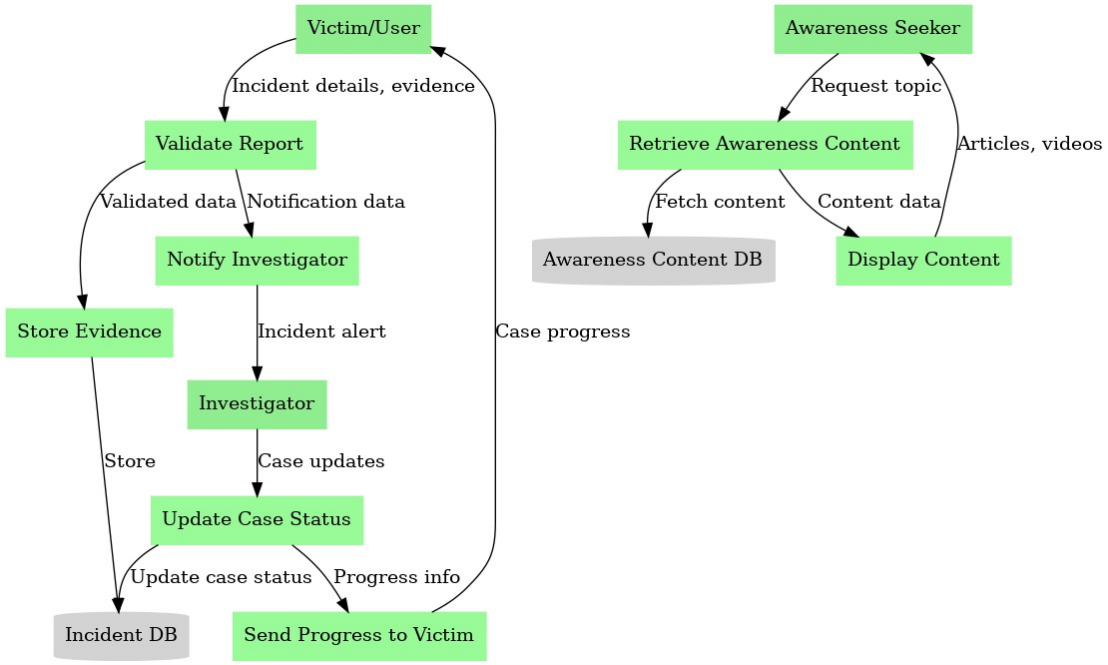

Figure 3.1: Admin - Case Assignment Overview

3.3.2 Investigator

- Overview (stats, deadlines)
- My Cases (search, filters, progress)
- Activity Log (timeline)
- Evidence (case evidence list)
- Reports (draft/submit/download)
- Messages (with victims and admins who contacted first)

Figure 3.2: Investigator - Cases Grid

3.3.3 Victim

- Overview (active cases, quick actions)
- My Incidents (search, filter, details)
- Case Status (timeline, evidence, documents)
- Resources (guides, legal info, support)
- Messages (with assigned investigator, Admin Panel broadcasts)

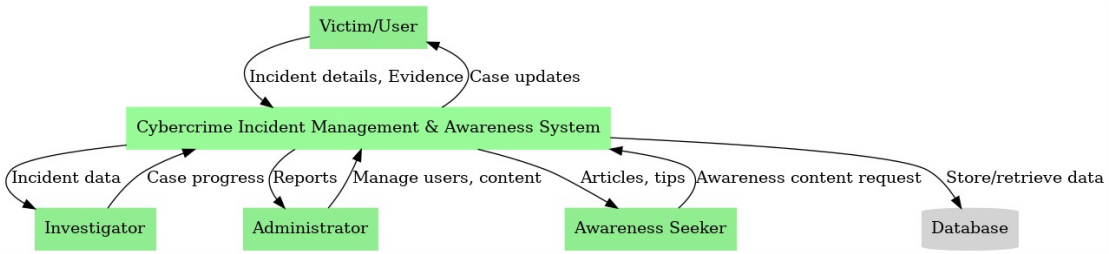

Figure 3.3: Victim - Dashboard Overview

Chapter 4

Messaging System Architecture

4.1 System Overview

4.1.1 Purpose

The messaging system enables **secure, role-based communication** within CIMAS, facilitating:

- Support Communication
- Case Collaboration
- Victim Assistance
- Team Coordination
- System Announcements

4.1.2 Key Principles

1. **Role-Based Access Control (RBAC):** Different roles have different messaging capabilities
2. **Real-Time Communication:** Messages sync in near real-time (5-10 second polling)
3. **Message Tracking:** Delivery status, read receipts, timestamp tracking
4. **Security First:** JWT authentication, permission validation

4.2 Database Schema

4.2.1 Message Model

```
class Message(models.Model):
    # Who sent the message
    sender = ForeignKey(User, related_name='sent_messages')

    # Who receives it (null for broadcasts)
```

```
    receiver = ForeignKey(User, related_name='received_messages',
                          null=True, blank=True)

    # The actual message content
    content = TextField()

    # Is this a broadcast message?
    is_broadcast = BooleanField(default=False)

    # Who should receive the broadcast?
    broadcast_type = CharField(max_length=20,
                               choices=[('all', 'All Users'),
                                        ('investigators', 'All
                                           Investigators'),
                                        ('victims', 'All Victims')],
                               null=True, blank=True)

    # When was it sent?
    timestamp = DateTimeField(auto_now_add=True)

    # Status tracking
    delivered = BooleanField(default=False)
    read = BooleanField(default=False)

    class Meta:
        ordering = ['timestamp']
```

4.3 Backend Implementation

4.3.1 View Classes

MessageListCreateView

Handles GET (list messages) and POST (send message) requests.

GET Request:

```
GET /api/chat/messages/?chat_with=5
```

Returns all messages between current user and user #5, ordered by timestamp.

POST Request:

```
POST /api/chat/messages/
{
    "receiver": 1,
    "content": "Can you help me?",
    "is_broadcast": false
}
```

MessageDetailView

Handles updating individual messages (marking as read).

```
PATCH /api/chat/messages/10/
{
    "read": true
}
```

AvailableUsersView

Returns list of users current user can message based on their role.

```
GET /api/chat/available-users/
```

4.3.2 Permission Logic

Action	Admin	Investigator	Victim
View own messages	✓	✓	✓
View others' messages	✓	×	×
Message any user	✓	×	×
Message admin	✓	✓	✓
Message investigator	✓	×	✓*
Message victim	✓	✓*	×
Send broadcast	✓	×	×

Table 4.1: Role-Based Messaging Permissions (*if assigned)

4.4 Admin Panel Broadcast Feature

4.4.1 Overview

The Admin Panel is a special system user serving as a centralized location for viewing all broadcast messages sent by administrators.

4.4.2 Key Features

- **System User:** `admin.panel@system.internal`
- **Auto-created:** Generated automatically or via management command
- **Security:** Cannot login (unusable password, is_active=False)
- **Purpose:** Stores copies of all admin broadcasts

4.4.3 Setup Command

```
python manage.py setup_admin_panel
```

4.4.4 API Endpoint

```
GET /api/chat/admin-panel-broadcasts/
```

Returns all broadcast messages filtered by user role:

- **Admins:** All broadcasts
- **Investigators:** Broadcasts with type 'all' or 'investigators'
- **Victims:** Broadcasts with type 'all' or 'victims'

4.4.5 Broadcast Filtering

User Role	Can See Broadcasts Of Type
Admin	all, investigators, victims (everything)
Investigator	all, investigators
Victim	all, victims

Table 4.2: Broadcast Visibility Rules

4.5 Frontend Implementation

4.5.1 Component Structure

The messaging component manages:

- Authentication state
- Conversation state
- UI state
- Data loading effects
- Event handlers

4.5.2 Data Flow Patterns

Optimistic Updates

1. User types message and clicks Send
2. Message appears instantly in chat (optimistic)
3. Loading spinner on message
4. Backend confirms message sent
5. Spinner becomes checkmark

Polling System

Dual Polling:

- **Selected Conversation:** 5 seconds - Fast updates for active chat
- **All Conversations:** 10 seconds - Background sync

4.5.3 Message Lifecycle

1. **COMPOSING:** User is typing
2. **SENDING:** HTTP request in progress, optimistic UI update
3. **SENT:** Message in database (single gray checkmark)
4. **DELIVERED:** Receiver saw message in list (double gray checkmark)
5. **READ:** Receiver viewed message (double blue checkmark)

4.6 Security Considerations

4.6.1 Authentication Layer

- JWT token required for all requests
- Token stored in localStorage
- Included in Authorization header
- Backend validates signature and extracts user_id

4.6.2 Security Features

1. **Read-Only Access:** Users can only read broadcasts, not send messages to Admin Panel
2. **No Login:** Admin Panel account cannot be used for authentication
3. **System-Only:** Account managed by system commands only
4. **Audit Trail:** All broadcasts permanently stored for compliance

4.6.3 SQL Injection Prevention

Django ORM automatically parameterizes queries, preventing SQL injection attacks.

4.6.4 XSS Prevention

React automatically escapes user content, preventing cross-site scripting attacks.

Chapter 5

Chat System Updates

5.1 Updated Messaging Permissions

5.1.1 Victim Permissions (Restricted)

- **Can:** Message investigators assigned to their cases
- **Cannot:** Message admins directly, message other users
- **Validation:** System checks `IncidentAssignments` table

5.1.2 Investigator Permissions (Conditional)

- **Can:** Message victims assigned to their cases, message admins who contacted them first
- **Cannot:** Initiate contact with admins
- **Validation:** Checks victim assignment and prior admin contact

5.1.3 Admin Permissions (Unchanged)

- Can message anyone
- Can send broadcast messages
- Can view all conversations

5.2 File Changes

5.2.1 New Files Created

1. `/backend/CIMAS/chat/management/` (directory)
2. `/backend/CIMAS/chat/management/commands/` (directory)
3. `/backend/CIMAS/chat/management/__init__.py`
4. `/backend/CIMAS/chat/management/commands/__init__.py`

5. `/backend/CIMAS/chat/management/commands/setup_admin_panel.py`

5.2.2 Modified Files

1. `/backend/CIMAS/chat/views.py`
    - Added `ADMIN_PANEL_EMAIL` constant
    - Added `get_admin_panel_user()` helper function
    - Updated permission checks
    - Added `AdminPanelBroadcastsView` class
2. `/backend/CIMAS/chat/urls.py`
    - Added import for `AdminPanelBroadcastsView`
    - Added route: `api/chat/admin-panel-broadcasts/`

5.3 Testing Checklist

5.3.1 Functionality Tests

☐ All conversations load on mount

☐ Messages sorted correctly

☐ Unread counts accurate

☐ Last message previews correct

☐ Timestamps formatted properly

☐ Selected chat updates every 5s

☐ All chats update every 10s

☐ Read receipts work

☐ Sending messages updates immediately

5.3.2 Permission Tests

☐ Victim can message assigned investigator

☐ Victim cannot message admin

☐ Investigator can message assigned victim

☐ Investigator can message admin who contacted them

☐ Admin broadcast sent successfully

☐ Broadcasts visible in Admin Panel

5.3.3 Performance Tests

- ☐ No lag with 10+ conversations
- ☐ No lag with 50+ conversations
- ☐ Smooth scrolling
- ☐ No memory leaks
- ☐ Fast initial load

Chapter 6

API Integration

6.1 Backend API Endpoints

6.1.1 Overview APIs

- `GET /api/victim/stats` - Dashboard statistics
- `GET /api/victim/cases/active` - Active cases
- `GET /api/victim/activities` - Upcoming activities
- `GET /api/victim/updates` - Recent updates

6.1.2 Incidents APIs

- `GET /api/victim/incidents` - All incidents
- `POST /api/victim/incidents` - Report new incident
- `GET /api/victim/incidents/:id` - Incident details

6.1.3 Case Status APIs

- `GET /api/victim/cases` - All cases
- `GET /api/victim/cases/:id` - Case details
- `GET /api/victim/cases/:id/timeline` - Case timeline
- `GET /api/victim/cases/:id/evidence` - Case evidence
- `GET /api/victim/cases/:id/documents` - Case documents

6.1.4 Resources APIs

- `GET /api/resources` - All resources
- `GET /api/resources/featured` - Featured resources
- `GET /api/resources/:id/download` - Download resource
- `GET /api/emergency-contacts` - Emergency contacts

6.1.5 Messaging APIs

- `GET /api/chat/available-users/` - Get contactable users
- `GET /api/chat/messages/?chat_with=<user_id>` - Get messages
- `POST /api/chat/messages/` - Send message
- `PATCH /api/chat/messages/<id>/` - Update message (mark as read)
- `GET /api/chat/admin-panel-broadcasts/` - Get admin broadcasts

Chapter 7

Backend Modules and APIs

7.1 Django Apps

- **users**: auth (register, login, logout), user profile CRUD
- **incidents**: incidents list/detail, user incidents
- **evidence**: evidence per user/incident, evidence detail
- **cases**: assign, reassign, unassigned, my assigned
- **activity_logs**: logs listing and filters
- **analytics**: summary, detailed, trends, hotspots, categories
- **awareness**: awareness resources CRUD
- **chat**: messages, available users, admin panel broadcasts

7.2 Endpoints Summary

7.2.1 Authentication and Users

```
POST   /api/auth/register
POST   /api/auth/login
POST   /api/auth/logout
POST   /api/auth/token/
POST   /api/auth/token/refresh/
GET    /api/users
GET    /api/users/me
PATCH  /api/users/me
GET    /api/users/<id>
PATCH  /api/users/<id>
DELETE /api/users/<id>
```

7.2.2 Incidents

```
GET    /api/incidents
POST   /api/incidents
GET    /api/incidents/<id>
PUT    /api/incidents/<id>
DELETE /api/incidents/<id>
GET    /api/incidents/user/<id>
```

7.2.3 Evidence

```
GET    /api/evidence
GET    /api/incidents/<id>/evidence
GET    /api/evidence/<eid>
```

7.2.4 Cases

```
POST   /api/cases/<id>/assign/<userId>
POST   /api/cases/<id>/reassign/<userId>
GET    /api/cases/unassigned
GET    /api/cases/assigned/
```

7.2.5 Activity Logs

```
GET    /api/logs
GET    /api/logs/<id>
GET    /api/logs/user/<userId>
GET    /api/logs/incidents/<incidentId>
```

7.2.6 Analytics

```
GET    /api/analytics/summary
GET    /api/analytics/detailed
GET    /api/analytics/trends
GET    /api/analytics/hotspots
GET    /api/analytics/categories
```

7.2.7 Awareness

```
GET    /api/awareness/resources/
POST   /api/awareness/resources/
GET    /api/awareness/resources/<pk>/
PUT    /api/awareness/resources/<pk>/
PATCH  /api/awareness/resources/<pk>/
DELETE /api/awareness/resources/<pk>/
```

7.2.8 Chat

```
GET    /api/chat/available-users/
GET    /api/chat/messages/?chat_with=<user_id>
POST   /api/chat/messages/
PATCH  /api/chat/messages/<id>/
GET    /api/chat/admin-panel-broadcasts/
```

Chapter 8

Security Features

8.1 Authentication

- JWT-based authentication (access/refresh tokens)
- Password hashing via Django auth

8.2 Authorization

- Role-based permissions (admin, investigator, victim)
- Messaging path restrictions (victim<->assigned investigator, investigator<->admin who contacted first)

8.3 Data Protection

- ORM parameterization to prevent SQL injection
- Automatic XSS escaping in React views
- File upload paths isolated under MEDIA_ROOT

8.4 Audit and Monitoring

- Activity logs for key actions
- Broadcast messages retained for compliance

8.5 Operational Hardening

- Admin Panel system user: inactive, unusable password
- Sensible defaults for missing broadcast_type

Chapter 9

Future Plans

- Real-time WebSocket messaging (replace polling)
- File attachments with virus scanning and type validation
- Message search and advanced filters
- Infinite scroll and pagination for message history
- Offline caching (IndexedDB) and sync on reconnect
- Analytics dashboards for case trends and workloads
- Notification system (email/push)

Chapter 10

Quick Reference Guide

10.1 Chat System Setup

10.1.1 Setup Admin Panel

```
cd "F:\Btech\DBMS Project\backend\CIMAS"
python manage.py setup_admin_panel
```

10.1.2 Test the System

```
python manage.py shell < chat/test_chat_system.py
```

10.2 Messaging Rules Summary

Role	Can Message
Victims	Investigators assigned to their cases
Investigators	Assigned victims, Admins who contacted them first
Admins	Anyone, Can send broadcasts

Table 10.1: Messaging Rules by Role

10.3 Database Queries

10.3.1 Check Admin Panel

```
from django.contrib.auth import get_user_model
User = get_user_model()
admin_panel = User.objects.get(email='admin.panel@system.internal'
    )
print(f"ID: {admin_panel.id}, Active: {admin_panel.is_active}")
```

10.3.2 View All Broadcasts

```
from chat.models import Message

broadcasts = Message.objects.filter(is_broadcast=True)
for msg in broadcasts:
    print(f"{msg.sender.email} -> {msg.broadcast_type}: {msg.
       content[:50]}")
```

10.3.3 Check User Permissions

```
from incidents.models import IncidentAssignments

# Check victim's investigators
victim_id = 123
investigators = IncidentAssignments.objects.filter(
    incident__user_id=victim_id
).values_list('assigned_to__email', flat=True).distinct()
print(f"Victim can message: {list(investigators)}")
```

Chapter 11

Performance Optimization

11.1 Parallel Loading

The system uses `Promise.all()` to load all conversations simultaneously rather than sequentially, significantly reducing initial load time.

11.2 Efficient State Updates

Messages are stored in an object structure: `{userId: [messages]}` for O(1) lookup time and efficient updates.

11.3 Polling Configuration

Action	Interval	Purpose
Initial Load	On mount	Load all conversations
Selected Chat Polling	5 seconds	Fast updates for active chat
All Chats Polling	10 seconds	Background sync
Read Receipt	Immediate	Mark as read on view

Table 11.1: Polling Timing Configuration

11.4 Performance Metrics

Scenario	Users	Network	Load Time	Memory
Small	5	Fast 4G	500ms	2MB
Medium	20	Fast 4G	1.5s	5MB
Large	50	Fast 4G	3s	10MB
Small	5	Slow 3G	2s	2MB

Table 11.2: Estimated Performance Metrics

Chapter 12

Future Enhancements

12.1 Phase 1: Current Implementation

- HTTP polling (5-10 seconds)
- Role-based permissions
- Admin broadcasts
- Read receipts

12.2 Phase 2: WebSocket Integration

- True real-time updates (0 delay)
- Typing indicators
- Online presence
- Lower server load

12.3 Phase 3: Enhanced Features

- File attachments
- Message reactions
- Message search
- Infinite scroll for history

12.4 Phase 4: Offline Support

- IndexedDB caching
- Offline message queue
- Sync on reconnect

Chapter 13

Troubleshooting

13.1 Common Issues

13.1.1 Admin Panel user not found

Solution:

```
python manage.py setup_admin_panel
```

13.1.2 Permission denied when messaging

Check:

1. Is user assigned to the case? (for victim/investigator pairs)
2. Has admin messaged the investigator? (for investigator→admin)
3. Is the victim trying to message admin? (not allowed)

13.1.3 Broadcasts not appearing

Check:

1. Admin Panel user exists
2. Broadcast was sent with `is_broadcast=true`
3. User is authenticated when calling endpoint

13.1.4 Old permissions still applying

Solution: Clear browser cache and restart Django server

Chapter 14

Appendices

14.1 Appendix A: File Structure

14.1.1 Frontend Structure

```
frontend/
+- src/
|   +- components/
|   |   +- Admin/
|   |   |   +- AssignCase/
|   |   +- Investigator/
|   |   |   +- Dashboard.jsx
|   |   |   +- InvestigatorOverview.jsx
|   |   |   +- MyCases.jsx
|   |   |   +- ActivityLog.jsx
|   |   |   +- Evidence.jsx
|   |   |   +- Reports.jsx
|   |   |   +- Messages.jsx
|   |   +- Victim/
|   |       +- Dashboard.jsx
|   |       +- VictimOverview.jsx
|   |       +- MyIncidents.jsx
|   |       +- CaseStatus.jsx
|   |       +- Resources.jsx
|   +- Messaging.jsx
|   +- api/
|       +- messaging.js
```

14.1.2 Backend Structure

```
backend/CIMAS/
+- chat/
|   +- models.py
|   +- views.py
|   +- serializers.py
|   +- urls.py
```

```
|   +- management/
|       +- commands/
|           +- setup_admin_panel.py
+- incidents/
+- cases/
+- evidence/
+- users/
```

14.2 Appendix B: Dependencies

14.2.1 Frontend Dependencies

- React 19.1.1
- Framer Motion 12.23.22
- Lucide React 0.544.0
- Tailwind CSS 4.1.14

14.2.2 Backend Dependencies

- Django 4.x
- Django REST Framework
- PostgreSQL
- djangorestframework-simplejwt

14.3 Appendix C: Glossary

CIMAS Cybercrime Incident Management & Awareness System

JWT JSON Web Token - Authentication standard

RBAC Role-Based Access Control

WebSocket Protocol for bidirectional real-time communication

ORM Object-Relational Mapping

XSS Cross-Site Scripting attack

SQL Injection Database attack technique

API Application Programming Interface

14.4 Appendix D: References

- Django REST Framework: https://www.django-rest-framework.org/
- React Documentation: https://react.dev/reference/react
- WebSocket Protocol: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket
- JWT Authentication: https://jwt.io/introduction
- PostgreSQL Documentation: https://www.postgresql.org/docs/