

# Recon-0: Modern Bug Bounty Platform

A feature-complete, modern clone of the HackerOne bug bounty platform built as a B.Tech IT final year project. Recon-0 provides a comprehensive solution for organizations to manage security programs and for security researchers to discover and report vulnerabilities.

## Project Overview

**Recon-0** is a simplified yet fully functional bug bounty platform that includes:

- Three distinct user roles: **Hacker**, **Organization**, and **Platform Admin**
- Complete vulnerability reporting and management workflow
- Real-time chat communication
- AI-powered report enhancement
- Gamification system with reputation points and achievements
- Modern "terminal-inspired" UI design

## Architecture

### Hybrid Architecture Approach

The project uses a **hybrid architecture** designed for development efficiency and clear backend specifications:

- **Frontend**: Completely decoupled React application
- **Mock API**: Node.js/Express server serving as the definitive backend blueprint
- **Chat Service**: Dedicated Supabase project for real-time messaging
- **AI Integration**: Local LM Studio server for AI features

### Why This Architecture?

The project initially attempted an enterprise-grade stack (Next.js + TypeScript + Supabase RLS) but encountered significant development blockers with Row-Level Security policies. The hybrid approach was adopted to:

- Eliminate authentication debugging issues
- Provide clear API contracts for the backend developer
- Enable rapid frontend development
- Maintain realistic production-ready patterns

# Tech Stack

## Frontend

- **Framework:** Vite + React 18
- **Language:** JavaScript (no TypeScript)
- **Styling:** Bootstrap 5 (CDN) + Custom CSS
- **Routing:** React Router DOM v6
- **State Management:** Zustand
- **Charts:** Chart.js + react-chartjs-2
- **API Communication:** Custom `apiService.js`

## Backend (Mock API)

- **Runtime:** Node.js
- **Framework:** Express.js
- **Authentication:** "Fake JWT" system (development blueprint)
- **File Uploads:** Multer (local storage)
- **Database:** In-memory JavaScript objects
- **AI Integration:** LM Studio API client

## Database Schema

- **Production Target:** PostgreSQL
- **Schema File:** `recon-0-lite-V2.sql`
- **Key Features:** UUID primary keys, role-based access, formal messaging system

## Real-time Chat

- **Service:** Dedicated Supabase project
- **Authentication:** Background dummy user
- **Purpose:** Single global "Safe Harbor" chat room

## Design Theme

"Modern Terminal" - A hybrid aesthetic combining:

- **Colors:** Dark charcoal background (`#111827`) with Matrix green accents (`#34d399`)
- **Typography:** Nunito Sans (headings) + JetBrains Mono (body)
- **Style:** Clean, spacious layouts with soft rounded corners

## Project Structure

```
recon-0/
├── frontend/           # React application
│   ├── src/
│   │   ├── components/  # Reusable UI components
│   │   ├── pages/       # Route-specific pages
│   │   ├── lib/         # API service and utilities
│   │   │   ├── apiService.js    # Main API client
│   │   │   ├── chatAuthService.js # Chat authentication
│   │   │   ├── gamificationUtils.js # Level/progress calculations
│   │   │   └── supabaseChatClient.js # Chat service client
│   │   └── index.css      # Custom theme styles
│   └── package.json
├── recon0-mock-api/    # Development backend
│   ├── server.js       # Express server with full API
│   ├── uploads/        # Local file storage
│   └── package.json
├── recon-0-lite-V2.sql  # Production database schema
└── docs/
    ├── recon0_api_contract.md # Complete API specification
    └── features-list.md       # Detailed feature requirements
```

## Getting Started

### Prerequisites

- **Node.js** (v18 or higher)
- **npm** or **yarn**
- **LM Studio** (for AI features)
- **Supabase Account** (for chat functionality)

### Installation

#### 1. Clone the repository

```
bash

git clone <repository-url>
cd recon-0
```

#### 2. Setup Frontend

```
bash
```

```
cd frontend
npm install
```

### 3. Setup Mock API

```
bash

cd ../recon0-mock-api
npm install
```

### 4. Configure Environment Variables

Create `.env` file in frontend directory:

```
env

VITE_SUPABASE_URL=your_supabase_project_url
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key
```

### 5. Setup LM Studio (Optional - for AI features)

- Download and install [LM Studio](#)
- Load a model (e.g., Phi-3-mini-instruct)
- Start local server on `http://localhost:1234`

## Running the Application

#### 1. Start Mock API Server

```
bash

cd recon0-mock-api
node server.js
```

#### 2. Start Frontend Development Server

```
bash

cd frontend
npm run dev
```

#### 3. Access the Application

- Frontend: `http://localhost:5173`
- Mock API: `http://localhost:3001`

## Test Accounts

The mock API includes pre-configured test accounts:

```
javascript
```

// Hacker Account

Email: asdf@g.i

Password: 12345

// Organization Account

Email: qwer@g.i

Password: 12345

// Admin Account

Email: admin@recon0.com

Password: pass123

## API Contract

The mock API serves as the definitive blueprint for the production backend. All endpoints, request/response formats, and business logic are fully documented in `recon0_api_contract.md`.

## Key API Features






- **JWT Authentication:** Bearer token system
- **Role-based Access Control:** Hacker, Organization, Admin
- **File Upload System:** Avatars, logos, and report attachments
- **Formal Messaging:** Organization-to-hacker communication
- **AI Integration:** Report enhancement and chatbot
- **Admin Panel:** User management and platform analytics

## Base URL






`http://localhost:3001/api/v1`

## Core Features




### Hacker Workflow

-  Role-based dashboard with statistics
-  Program discovery with filtering
-  Vulnerability report submission with attachments
-  "My Reports" management
-  Detailed report view with formal reply system







## Organization Workflow

-  Organization-specific dashboard
-  Program creation and management
-  Report triage and status updates
-  Program analytics with charts
-  Formal reply system with attachments

## Admin Features

-  User management (view, suspend, reactivate)
-  Platform-wide analytics dashboard
-  Complete administrative control

## Shared Features

-  Real-time "Safe Harbor" chat
-  Global leaderboard
-  Notification system
-  Achievement system
-  AI-powered report enhancement
-  AI chatbot for platform help



## AI Integration

Recon-0 includes two AI-powered features:

1. **Report Enhancement:** Improves vulnerability report quality using local LLM
2. **Platform Chatbot:** Provides help and guidance to users

## Requirements:

- LM Studio running locally on port 1234
- Compatible model loaded (e.g., Phi-3-mini-instruct)



## Database Schema

The production database schema is defined in `recon-0-lite-V2.sql` with key tables:

- `profiles` - User information and roles
- `programs` - Bug bounty programs
- `reports` - Vulnerability reports

- `report_attachments` - File attachments for reports
- `report_messages` - Formal organization replies
- `message_attachments` - File attachments for messages
- `achievements` - Gamification achievements
- `chat_messages` - Real-time chat (Supabase)

## Development Status

**Current State:** Feature-complete MVP+ ready for final backend implementation

### Completed Features

- ☒ Complete authentication system
- ☒ All three user role workflows
- ☒ File upload system
- ☒ Real-time chat
- ☒ AI integration
- ☒ Gamification system
- ☒ Admin panel
- ☒ Responsive UI design

### Production Deployment Requirements

1. **Backend Implementation:** Build Java Spring Boot backend matching the API contract
2. **Database Setup:** Deploy PostgreSQL with the provided schema
3. **File Storage:** Integrate cloud storage (Cloudinary recommended)
4. **Authentication:** Implement proper JWT system
5. **AI Services:** Deploy or configure LLM API access

## Team

- **Asim:** Frontend Developer & Project Lead
- **Sujal:** Backend Developer (Java Spring Boot)
- **Rishabh & Tanmay:** Additional team members

## Development Guidelines

### For Frontend Development

- Use Bootstrap 5 classes exclusively
- Follow established color scheme and typography

- Maintain component structure and naming conventions
- All API calls must go through `apiService.js`

## For Backend Implementation

- Follow the exact API contract in `recon0_api_contract.md`
- Use the database schema from `recon-0-lite-V2.sql`
- Implement JWT authentication matching the mock system
- Maintain all business logic and validation rules

## Future Enhancements

Post-MVP features planned for future development:

- Advanced gamification with detailed point allocation
- Enhanced chat features (presence indicators, private messaging)
- Platform analytics and monitoring
- Advanced AI features (executive summaries, quality scoring)
- Mobile application

## License

This project is developed as a B.Tech final year project. All rights reserved.

## Contributing

This is an academic project. For questions or collaboration, please contact the development team.

---

**Note:** This README represents the current state of the Recon-0 project as of September 2025. The codebase is feature-complete and ready for production backend implementation following the provided API contract and database schema.