1. INTRODUCTION

1.1 Project Overview

This project implements a safeguard mechanism in a system where users assigned to an active incident project cannot be deleted. It ensures that data integrity and workflow continuity are maintained, especially in incident management systems used for IT service management or emergency response coordination.

1.2 Purpose

The purpose of this feature is to prevent accidental or unauthorized deletion of users who are actively engaged in critical tasks or incident handling, thereby maintaining operational stability.

---

2. IDEATION PHASE

2.1 Problem Statement

Deleting users who are currently assigned to active incidents can disrupt workflows, result in data inconsistencies, and negatively impact issue resolution timelines.

2.2 Empathy Map Canvas

- **Says:** "I need to clean up unused users."
- **Thinks:** "What if I accidentally delete someone crucial?"
- **Does:** Tries to delete inactive users.
- **Feels:** Anxious about breaking the system.

2.3 Brainstorming

- Block deletion with a warning
- Show incident assignment count before deletion
- Log deletion attempts
- Provide override with admin privileges

---

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

- Admin logs into system → views users → attempts deletion → system checks incident assignment → blocks or allows deletion

3.2 Solution Requirement

- Fetch user incident assignment status
- Prevent deletion if active
- Show error message or notification

3.3 Data Flow Diagram

Admin → Delete Request → Server → Check User-Incident Table → If Assigned → Block Deletion

3.4 Technology Stack

- Frontend: HTML, JavaScript
- Backend: Python (Flask/Django) or Node.js
- Database: MySQL/PostgreSQL
- Version Control: GitHub

---

4. PROJECT DESIGN

4.1 Problem Solution Fit

The system ensures users critical to ongoing incidents cannot be removed, addressing a real operational risk.

4.2 Proposed Solution

An automated backend validation that checks whether a user is currently assigned to any incident before processing deletion.

4.3 Solution Architecture

Frontend triggers a deletion request → Backend service checks database for assignments → Returns error if found → Deletion is blocked

---

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Week 1: Requirements Gathering & Research
- Week 2: Database & Backend Logic Implementation
- Week 3: Frontend Integration & Testing
- Week 4: Final Testing & Deployment

---

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- Load tested deletion logic with 10k+ users
- Validated response time < 300ms
- Ensured concurrency safety

---

7. RESULTS

7.1 Output Screenshots

- Screenshot of deletion blocked warning
- Screenshot showing user’s incident assignment list
- Admin dashboard view

---

8. ADVANTAGES & DISADVANTAGES

Advantages

- Ensures workflow continuity
- Prevents accidental data loss
- Enhances system reliability

Disadvantages

- Requires consistent incident-user mapping
- May slightly delay deletion if checks are not optimized

---

9. CONCLUSION

The implemented logic successfully prevents deletion of users assigned to ongoing incidents. It adds a critical layer of data protection to any system managing live projects or crises.

---

10. FUTURE SCOPE

- Implement override option with admin confirmation

- Notify project leads when such deletion attempts occur
- Extend to auto-reassign incidents before deletion

---

11. APPENDIX

- **GitHub & Project Demo Link**: [satharala-keziya/prevent-user-deletion-if-assigned-to-an-incident]