https://drive.google.com/drive/folders/1UCjESyFQWf1fwnIcOcqTNq8IHS2Oc4_P?usp=sharing

1. Overall

In this diagram is a high level overview of the working structure of the application. The rationale for the following to be the MVC pattern is the plan to later build the application using **Laravel** which is a PHP framework that lends itself by default to the MVC model. Additionally, event sourcing will be used which integrates well with the MVC model. The project is also a web application so MVC is a perfect choice for this case.

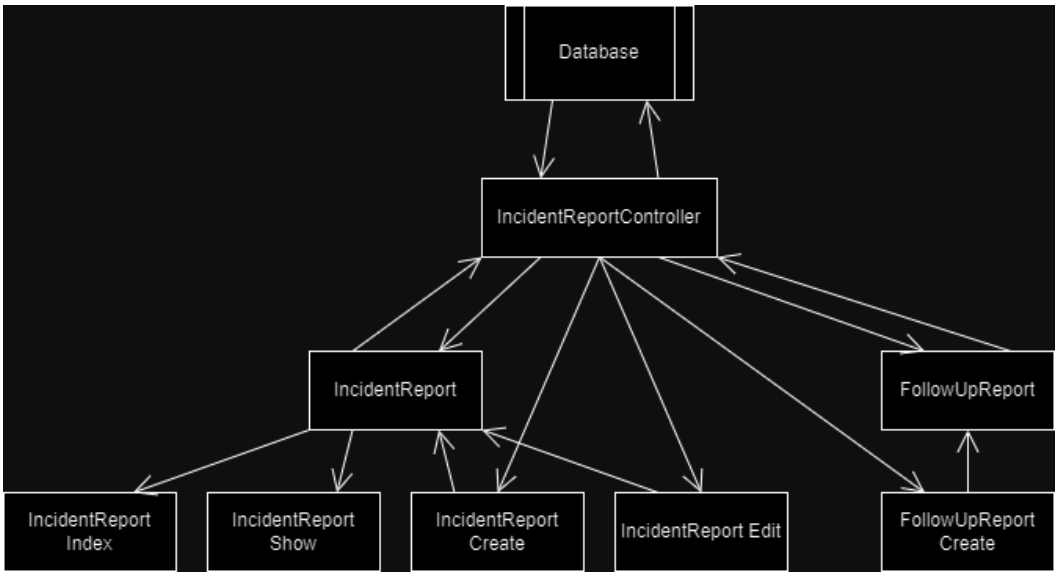


Figure 1. Overall Architecture

2. Sequence Diagram

The following sequence diagram is for the submission of an incident report. Notifications will be sent asynchronously as a background job.

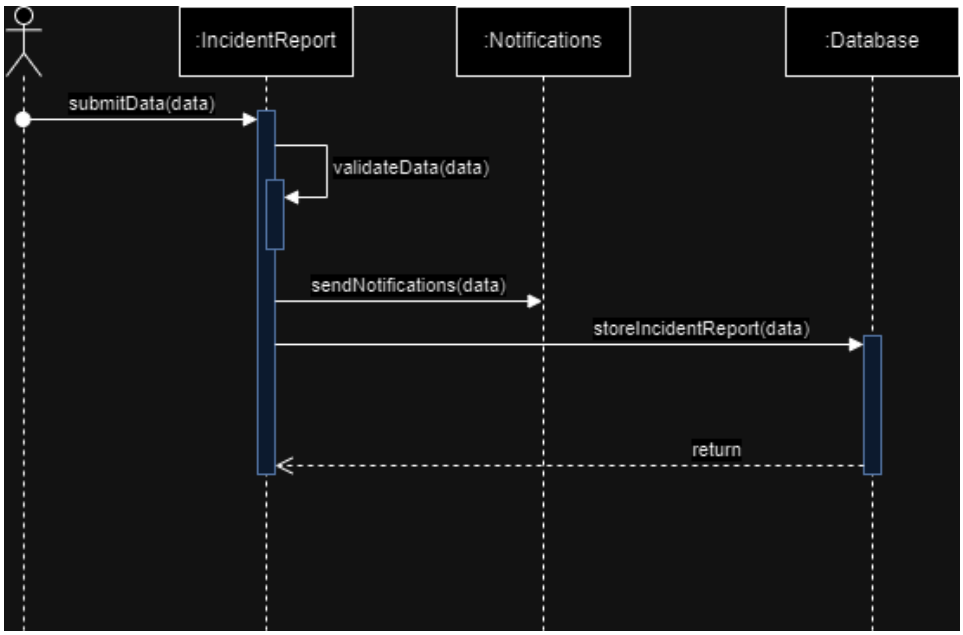
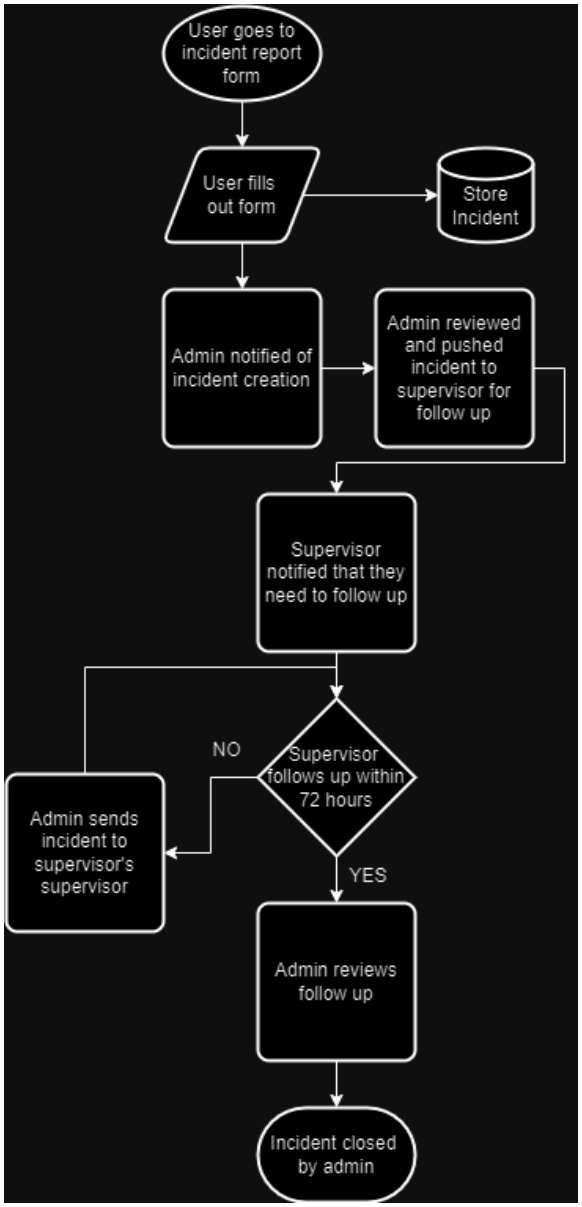


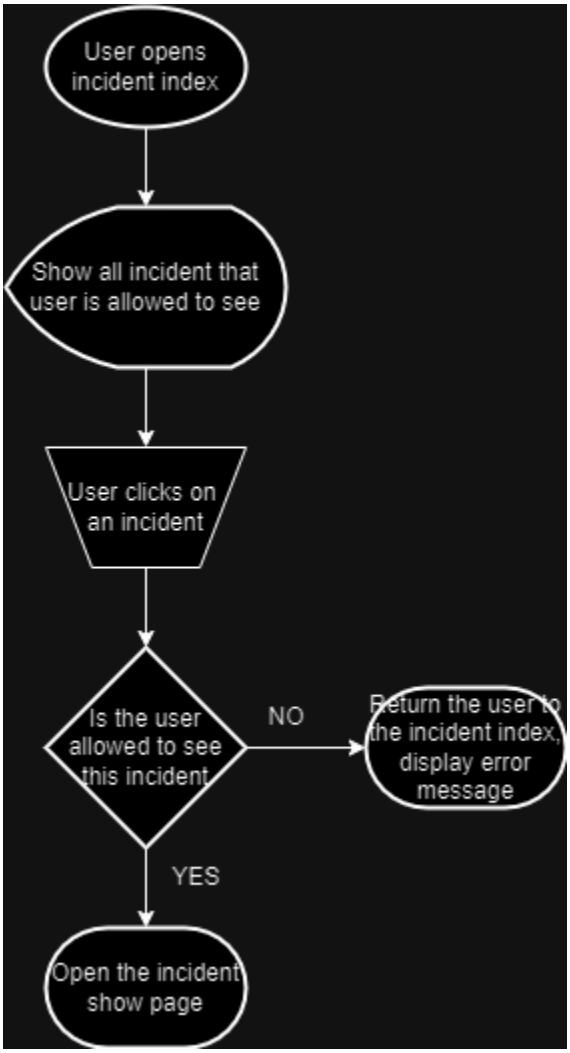
Figure 2. Incident Submission Sequence Diagram

3. Activity Diagrams

Left Diagram: Describes the activity diagram for submitting an Incident Report and the supervisor following up on it. **Figure 3. Incident Workflow Activity Diagram**

Right Diagram: Describes the activity diagram for viewing a specific incident. **Figure 4. Viewing incident Activity Diagram**





4. Domain Class Diagram

Models and Controllers have been defined here as per **Figure 1. Overall Architecture**. Additionally the projectors and some events that will be present have been laid out here. These are Laravel Event Sourcing Style.

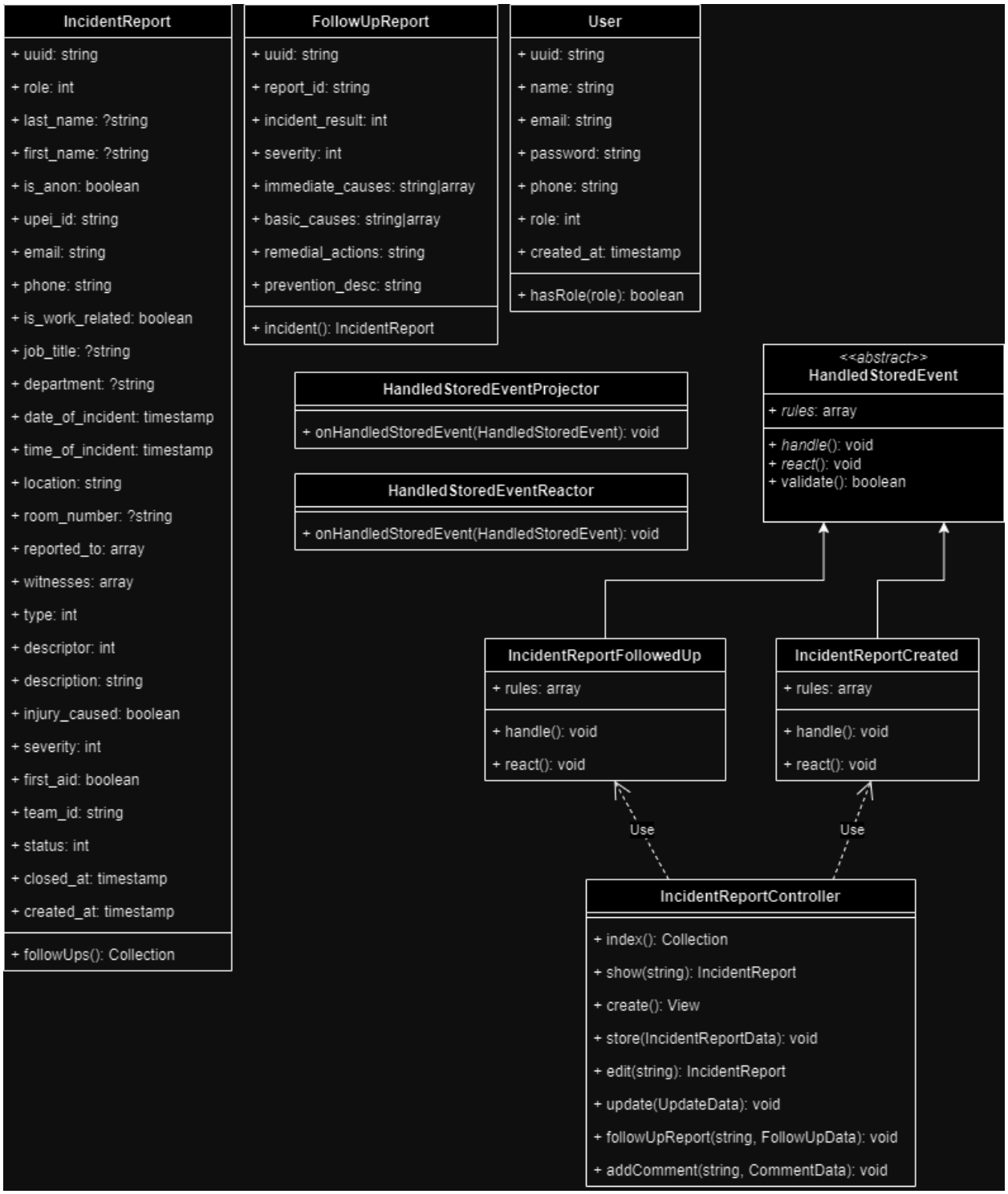


Figure 5. Domain Class Diagram

5. CRC Models

Class: IncidentReportController (Figure 6. IncidentReportController CRC Model)	
Responsible for performing and delegating all CRUD operations related to IncidentReport's and FollowUpReport's.	
Responsibility	**Collaborators**
Display an index listing of incidents for the user	IncidentReport User
Show a specific incident	IncidentReport User
Show the create page for an incident	User
Create a new IncidentReport model	IncidentReport
Show the edit page for a specific incident	User IncidentReport
Update the given incident with the given data	IncidentReport
Add comment to incident report	User IncidentReport

Class: HandledStoredEventProjector (Figure 7. HandledStoredEventProjector CRC Model)	
Responsible for handling all events that inherit the base class of HandledStoredEvent. Can be then used to reproject events in the future.	
Responsibility	**Collaborators**
Fires handle method on subclass event that was fired	HandledStoredEvent Subclasses

Class: HandledStoredEventReactor (Figure 8. HandledStoredEventReactor CRC Model)	
Responsible for reacting to all events that inherit the base class of HandledStoredEvent. Reacted to events involve operations that are external to the data saving and shouldn't ever run again like how a projector can be reprojected	
Responsibility	**Collaborators**
Fires react method on subclass event that was fired	HandledStoredEvent Subclasses

6. DFD Context, Level 0, and Level 1 Diagrams

Figure 9. Context Diagram

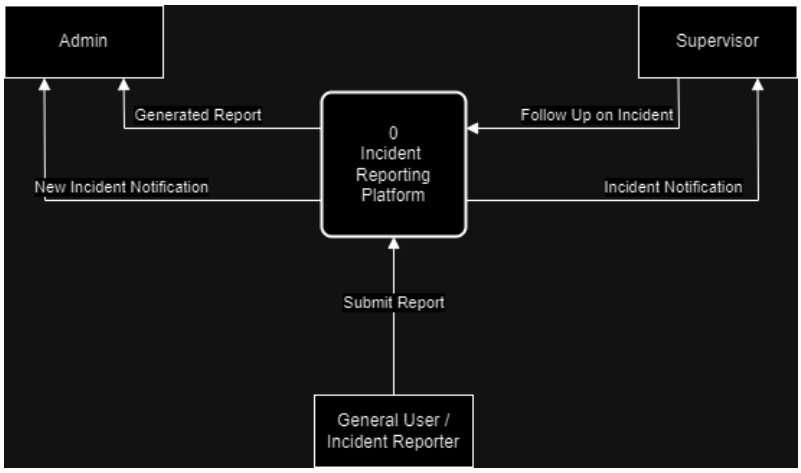


Figure 10. Level 0 Diagram

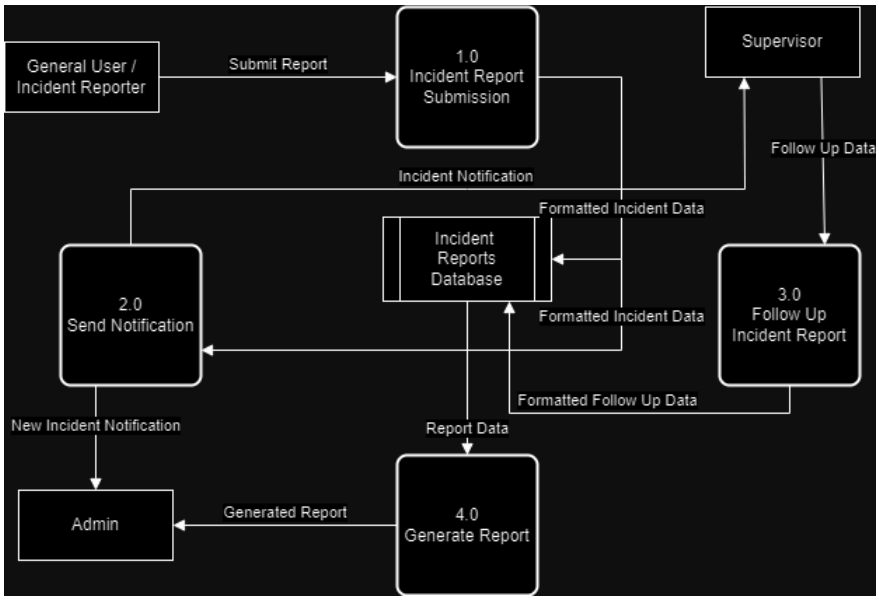
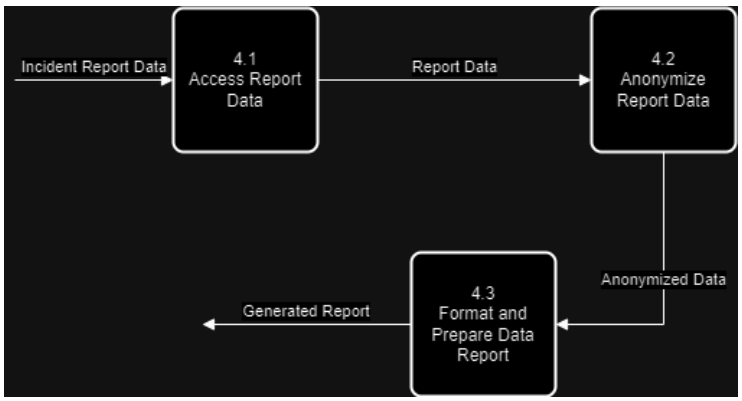


Figure 11. Level 1 Diagram



7. Tracing table

SEE CS4810 - HSE Incident Reporting Platform.pdf in project folder linked at top of document.

Figure	**Reference (as per phase 1 document)**
1	1. Introduction
2	2.1 Incident Report Submission
3	2.1 Incident Report Submission 2.3 Incident Report Follow Up
4	2.2 Viewing Incidents
5	2.1 Incident Report Submission 2.2 Viewing Incidents 2.3 Incident Report Follow Up 2.4 Notifications
6	2.1 Incident Report Submission 2.2 Viewing Incidents 2.3 Incident Report Follow Up 2.4 Notifications 2.6 Incident Report Audit Log
7	2.1 Incident Report Submission 2.3 Incident Report Follow Up 2.6 Incident Report Audit Log
8	2.4 Notifications
9	1. Introduction
10	2.1 Incident Report Submission 2.2 Viewing Incidents 2.3 Incident Report Follow Up 2.4 Notifications
11	2.7 Report Generation 3.2 Anonymity