./ i3CS 348 Project – Final Report

Toluwalogo Jibodu, Steven Shan, Hani Ly, Ethan Dobrowolski

R1. Application high-level description

We will use the dataset provided by The Washington Post that contains every fatal police shooting in the United States since 2015 (found here). This data will serve as the initial production dataset, though we will add the functionality to add additional instances via the frontend in the final application. The administrators of the database system would be a United States government organization that seeks to provide transparency and accountability into fatal police shootings in the United States. The users of our application will be members of the public. Users can query the database for information about the police shooting fatalities and have the capacity to submit a report regarding an incident.

The application supports searching and filtering incidents and agencies by attributes such as victim name, victim age, agency name, time, and location. Results are presented on a map, in a line chart, and in tables. Additionally, users are able to submit a report regarding a new incident to the database via a front-end form.

R2. System support description

The application is built with the PERN web stack, that is, PostgreSQL for the DBMS, Express for the web server, React for the frontend, and NodeJS for backend. The underlying map is represented with Mapbox. The line graph is built with Chart.js. Styling was accomplished with Bootstrap. Requests are forwarded from the front end to the back end using Axios. The application can be run on MacOS, Windows, or Linux, and will be hosted locally, and is accessed via a web browser.

To run the application, the PostgreSQL server and back-end web application must be hosted locally, and the front-end application must be run to interact with the dataset via the web UI. A script has been provided to perform this setup on Mac.

R3. Database with sample dataset:

The sample dataset is a randomly sampled subset of the production dataset. The data is stored in two CSV files. A script is used to parse the CSV files and insert the data records into the database.

R4. Database with production dataset:

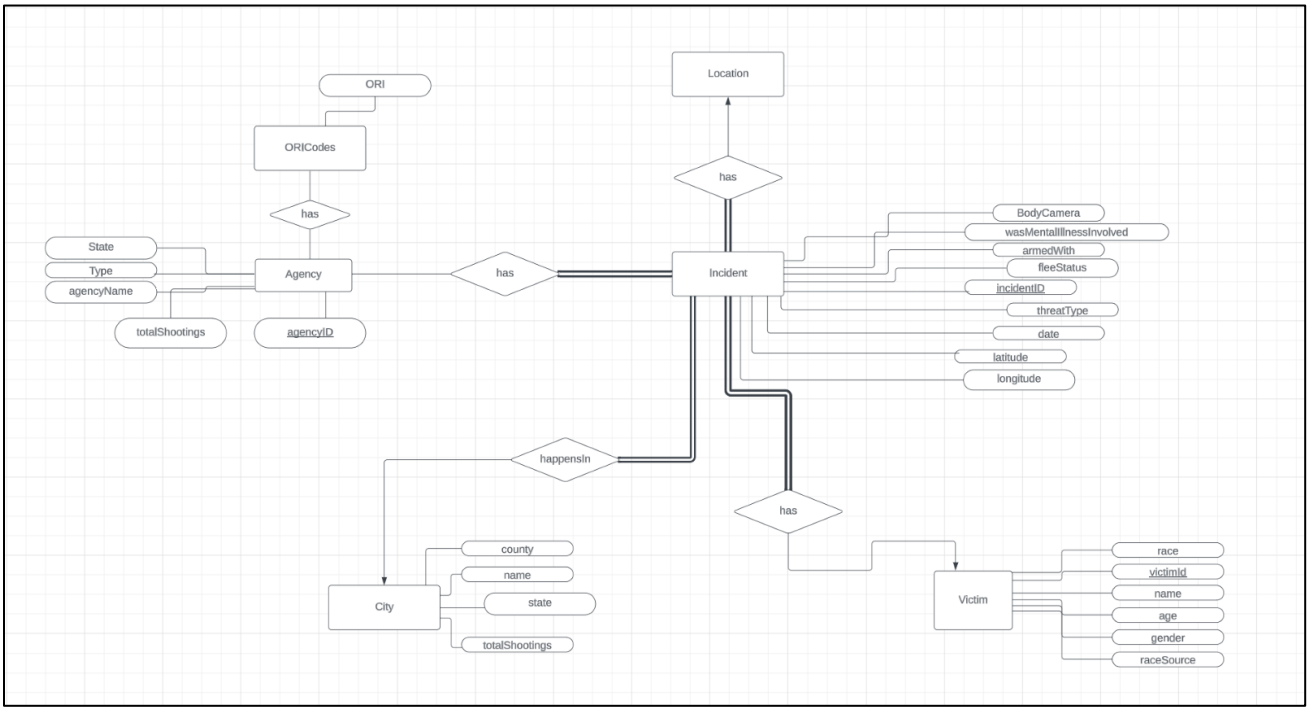
The production dataset is provided by the Washington Post. It is stored in two CSV files, one for incidents, and one for agencies. A script is used to parse the CSV files and insert the data records into the database. Additionally, there are scripts to drop and recreate all tables to enable easy resetting of the database.

R5. Schema Design

R5a. Assumptions

- The `id` field for each incident in the data provided is unique and not `null`.
- The `id` field for each agency in the data provided is unique and not `null`.
- The `id` field for each victim is unique and not `null`.
- The `id` field for each city is unique and not `null`.
- A victim cannot be involved in more than one incident. No two incidents in the provided `fatal-police-shootings-data.csv` file will have the exact same values for all the victim information fields.
- There are no incidents where the agencies involved do not exist.
- An ORI code is uniquely identified by an `AgencyID` and `ORI` code pair.
- An entry in the AgenciesInvolved table is uniquely identified by an `IncidentID` and an `AgencyID`.
- The State field must be one of the official state codes or `null`.

R5b. E/R Diagram



R5c. Relational Data Model

Incident(IncidentID, VictimID, CityID, Date, ThreatenType, FleeStatus, ArmedWith, WasMentalIllnessRelated, BodyCamera, Latitude, Longitude)

Victim(VictimID, Name, Age, Gender, Race, RaceSource)

City(CityID, CityName, County, State)

Agency(AgencyID, AgencyName, Type, State, TotalShootings)

ORICode(AgencyID, ORI)

AgenciesInvolved(IncidentID, AgencyID)

FK: IncidentID references Incident

FK: AgencyID references Agency

R6. Qualifying Feature 1 – Agencies Page

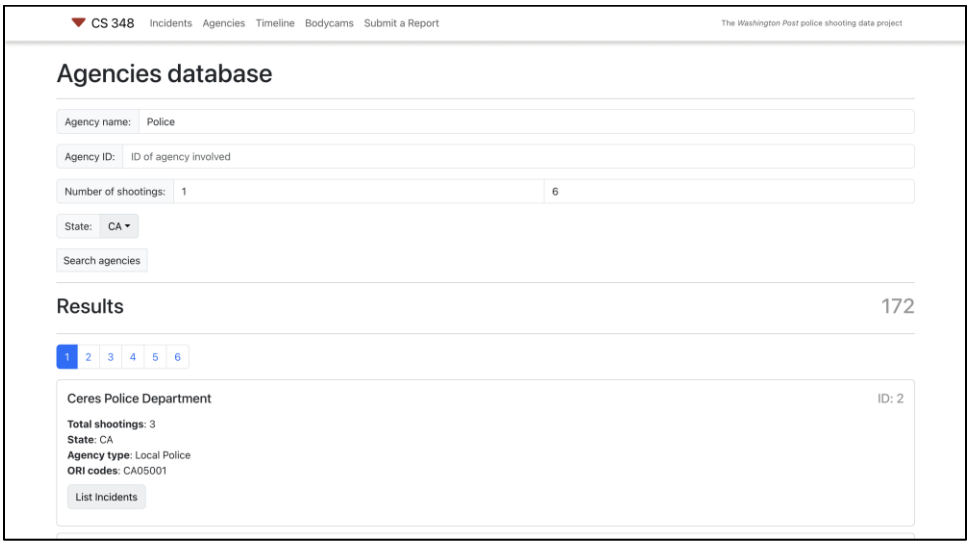
R6a.

There is an Agencies webpage which lists police agencies stored in the database. The user enters a form to query the database, which will then return information about all agencies that satisfy this query.

R6bc. Query

```
SELECT A.AgencyID, A.AgencyName, A.Type, A.State, A.TotalShootings,
       JSONB_AGG(DISTINCT AI.IncidentID) AS IncidentIDs, JSONB_AGG(DISTINCT O.ori)
        AS OriCodes

FROM Agency A

LEFT OUTER JOIN AgenciesInvolved AI ON A.AgencyID = AI.AgencyID

LEFT OUTER JOIN ORICode O ON A.AgencyID = O.AgencyID

       LEFT OUTER JOIN AgenciesInvolved AI ON A.AgencyID = AI.AgencyID

       LEFT OUTER JOIN HasORICodes O ON A.AgencyID = O.AgencyID

       WHERE LOWER(A.AgencyName) LIKE "{SEARCH_TERM}"

       GROUP BY A.AgencyID

       ORDER BY A.AgencyID
```

R6d.



R7. Qualifying Feature 2 – Map Visualizer

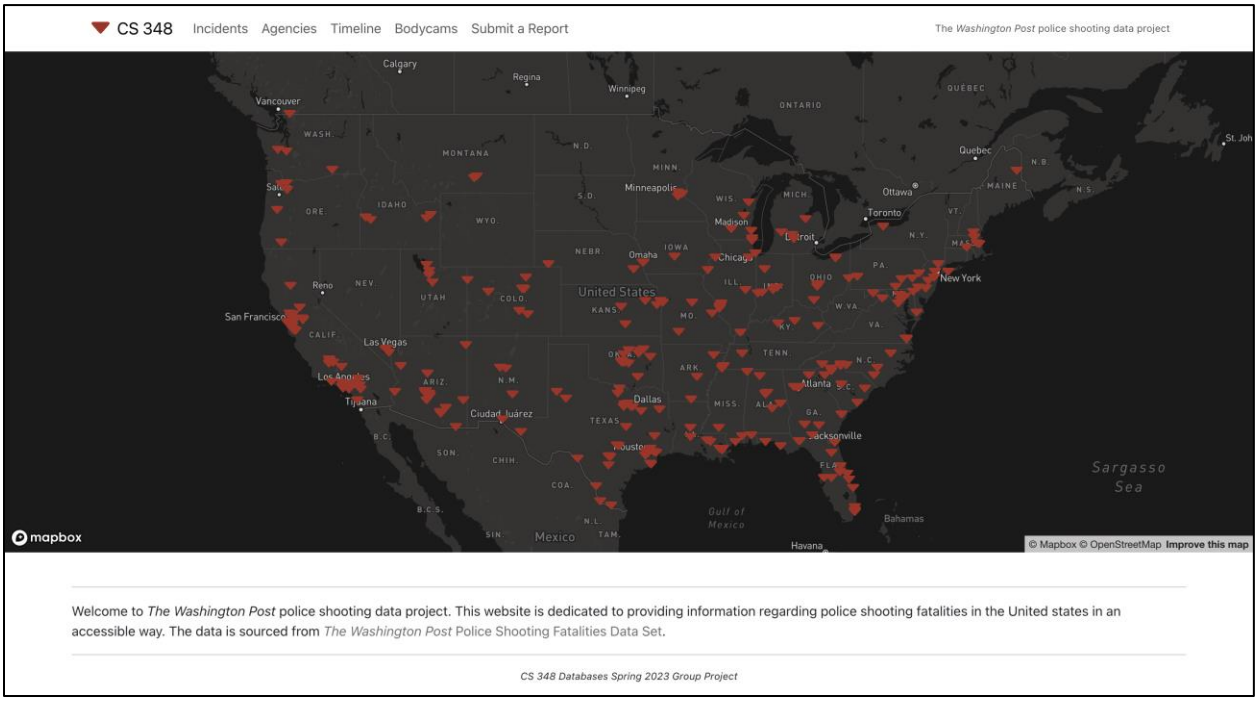
R7a.

The website will contain map visualization capabilities such as the ability for users to visually see where incidents occurred on a map.

R7bc. Query

```
SELECT IncidentID, V.Name, date, longitude, latitude
FROM Incident I
LEFT OUTER JOIN Victim V ON I.VictimID = V.VictimID
WHERE longitude > 'low_longitude' AND longitude < 'high_longitude' AND
        latitude > 'low_latitude AND latitude < 'high_latitude'
LIMIT 300;
```

R7d.



R8. Qualifying Feature 3 – Incident Search

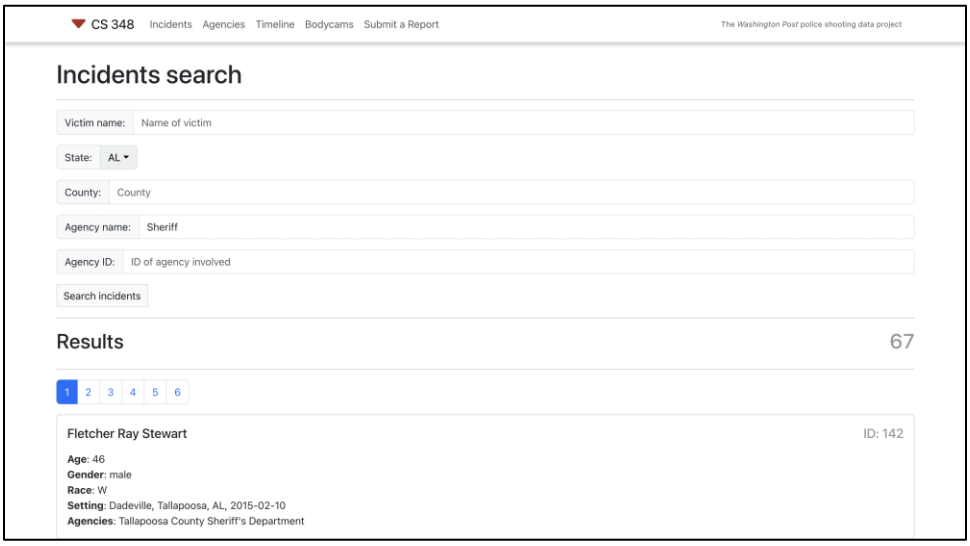
R8a.

The website will have a general search feature where users can search and filter for incidents by criteria such as name, location, agencies involved, victim race and victim gender. For example, given a full name, a query will be performed that will look for incidents where the victim's name matches the given full name.

R8bc. Query

```
SELECT I.IncidentID, I.Date, I.ThreatenType, I.FleeStatus,
       I.ArmedWith, I.WasMentalIllnessRelated, I.BodyCamera, I.Latitude,
       I.Longitude,
       V.VictimID, V.Name, V.Age, V.Gender, V.Race, V.RaceSource,
       JSONB_AGG(DISTINCT AI.AgencyID) AS AgencyIDs,
       JSONB_AGG(A.AgencyName) AS AgencyNames,
       C.CityID, C.CityName, C.County, C.State

FROM Incident I

LEFT OUTER JOIN Victim V ON I.VictimID = V.VictimID

LEFT OUTER JOIN AgenciesInvolved AI ON I.IncidentID = AI.IncidentID

LEFT OUTER JOIN Agency A ON AI.AgencyID = A.AgencyID

LEFT OUTER JOIN City C ON I.CityID = C.CityID

ORDER BY I.IncidentID
```

R8d.



R9. Qualifying Feature 4 – Report Submission

R9a.

The website will contain a form which users can fill out to submit a report of a fatal police shooting incident. A query will then be performed which inserts this entry into the database.

R9bc. Query

```
INSERT INTO Incident (IncidentID, Date, ThreatenType, FleeStatus, ArmedWith,
      WasMentalIllnessRelated, BodyCamera, Latitude, Longitude)

      VALUES (1, '2023-05-30', 'point', 'not', 'Gun', TRUE, TRUE, 123.456,
      789.012);

INSERT INTO AgenciesInvolved (IncidentID, AgencyID)

      VALUES ('agencyIDs', UNNEST(ARRAY[agencyIDListString]));
```

R9d.

R10. Qualifying Feature 5 – Timeline Graph

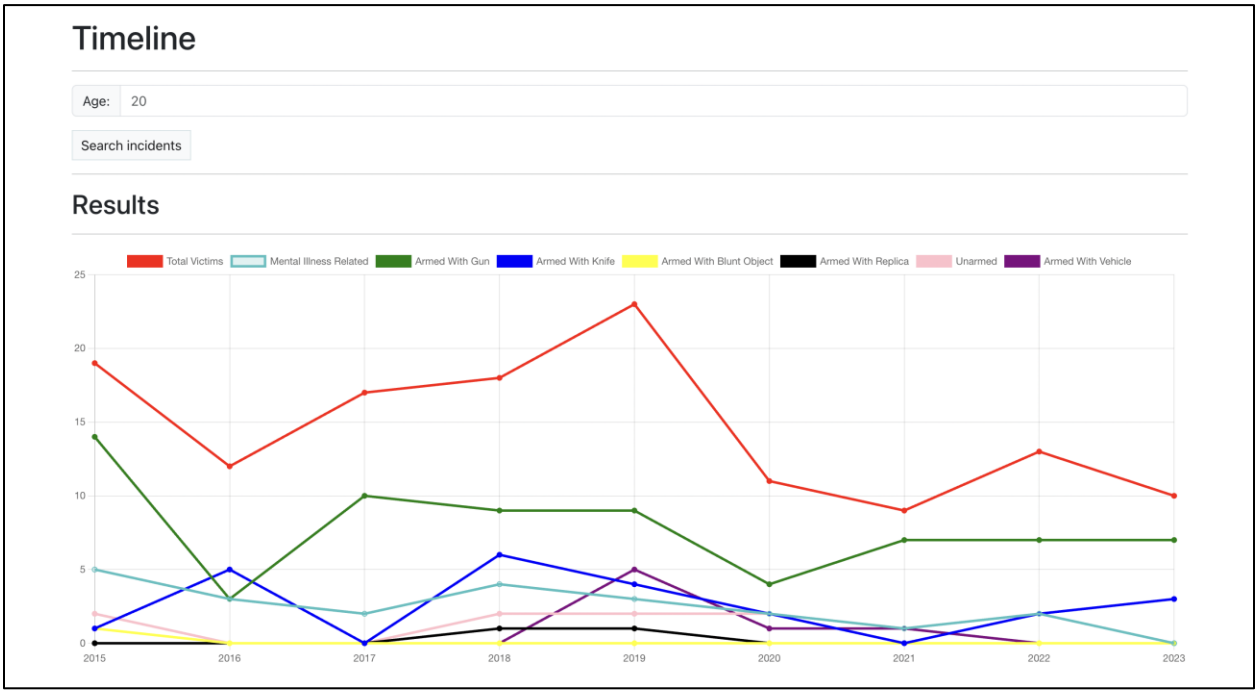
R10a.

The website will have a search feature where users can input an age number. A timeline graph will be shown which contains the number of incidents, percentage of those incidents which are related to mental illness and percentage of the victims who are armed with.

R10bc. Query

```
SELECT

    DATE_PART('YEAR', I.date),

    COUNT(*) AS total_number,

    COUNT(CASE WHEN I.WasMentalIllnessRelated = true THEN 1 END) AS
    mental_number,

    COUNT(CASE WHEN I.ArmedWith = 'gun' THEN 1 END) AS gun,

    COUNT(CASE WHEN I.ArmedWith = 'knife' THEN 1 END) AS knife,

    COUNT(CASE WHEN I.ArmedWith = 'blunt_object' THEN 1 END) AS bo,

    COUNT(CASE WHEN I.ArmedWith = 'replica' THEN 1 END) AS rep ,

    COUNT(CASE WHEN I.ArmedWith = 'unarmed' THEN 1 END) AS una,

    COUNT(CASE WHEN I.ArmedWith = 'vehicle' THEN 1 END) AS veh

    FROM Incident I NATURAL JOIN Victim V

    WHERE V.age = $1

    GROUP BY DATE_PART('YEAR', I.date)

    ORDER BY DATE_PART('YEAR', I.date)
```

R10d.


Timeline
Age:
20
Search incidents
Results
Total Victims
Mental Illness Related
Armed With Gun
Armed With Knife
Armed With Blunt Object
Armed With Replica
Unarmed
Armed With Vehicle
25
20
15
10
5
0
2015
2016
2017
2018
2019
2020
2021
2022
2023

R11. Qualifying Feature 6 – Bodycam Usage Percentage

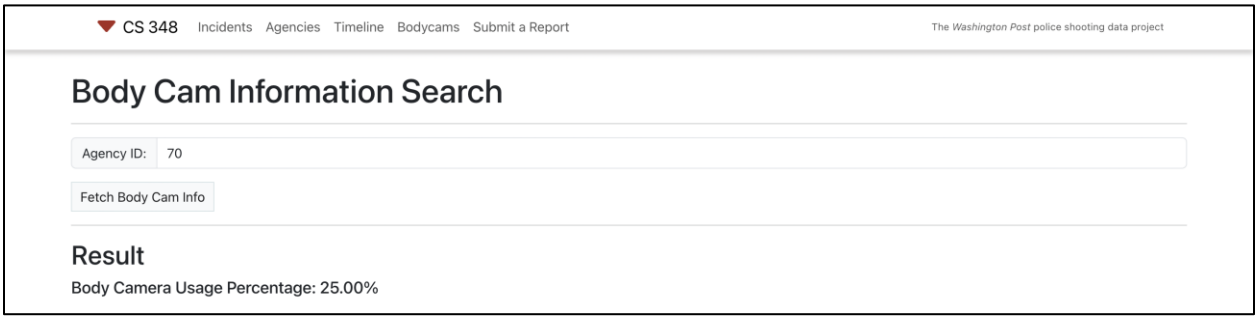
R11a.

The website will have a search feature where users can input a police department ID. A query will output the percentage of police officers from the given police department who wear a body camera when they are involved with an incident.

R11bc. Query

```
SELECT COUNT(*) as count FROM Incident I

JOIN AgenciesInvolved AI ON I.IncidentID = AI.IncidentID

WHERE AI.AgencyID = “input_ID” AND I.BodyCamera = TRUE;
```

R11d.



R12. Fancy Feature 1 – Optimization 1

Search pages support pagination to improve the performance of the website. Without pagination, all tuples would be returned and displayed, which would seriously degrade the performance of the web page. That is, the data that is returned is partitioned into a set of even-sized pages of tuples. Only one page is displayed at a time in the UI. Pagination enables the web UI to properly display all results.

R13. Fancy Feature 2 – UX 2

The UI has been improved to facilitate a better user experience. Namely, a navbar has been added to enable simple navigation throughout the website. Forms that take user queries use responsive dropdown menus, text inputs, and numeric inputs. Data is displayed in a descriptive and compact way using the map, tables, and the line chart.

R14. Fancy Feature 3 – Optimization 2

The map has been greatly optimized to allow any number of markers to be supported. If markers for all incidents were drawn simultaneously, it would greatly diminish the performance and readability of the map web page. When the web page is initially loaded, memory is allocated for a fixed number (I.e., 300) of markers and their associated `div` elements. A random sample of 300 records from the production data is taken, and the markers are drawn with this data. As the map bounds change (I.e., when the user moves, zooms, or rotates), a new random sample is taken using the new map bounds. Thus the 'resolution' of the map display is increased so that more markers are displayed in a region.

R15. Fancy Feature 4 – Validation 1

Input validation is performed on all form inputs. The 'Submit a Report' form Agency name field queries the database to ensure that the provided agency exists. Empty inputs are coerced to `null`. Numeric inputs are facilitated by numeric input forms that ensure the input is numeric. Several input fields use a drop-down menu to ensure the query is one of a set of entries.

R16. Fancy Feature 5 – UX 2

The map includes additional features beyond displaying where each incident occurs. Incident markers become highlighted when the user hovers the cursor over them, indicating that they can be clicked. When clicked, an incident marker will display a popup that provides additional information about that incident. If the victim's name was not recorded, it is displayed as 'Unknown.'

R17. Member Contributions

Toluwalogo Jibodu

- Created ER diagram
- Wrote data layer code to create database tables
- Wrote data layer code to populate database from CSV files
- Created a script to start database server, create the database, and create the user
- Authored several report sections
- Aided in front end implementation for Feature #2
- Implemented 6th feature (frontend and backend)

Steven Shan

- Implemented front end routing
- Wrote 5th and 6th feature descriptions
- Created the base for the frontend application
- Wrote data layer code to populate database from CSV files
- Assisted in ER diagram design
- Implemented feature #5 (timeline graph)

Hani Ly

- Implemented front end for Feature #1 and Feature #3
- Created scripts to parse CSV files
- Assisted in feature design
- Assisted in ER diagram design
- Authored several report sections
- Contributed the first feature
- Implemented frontend for features #1 (Agencies page), #3 (Incidents page), and #4 (Submit a Report page)

Ethan Dobrowolski

- Implemented Fancy Features #2, #3, and #5
- Co-implemented backend for all features
- Implemented Feature #2
- Co-implemented Fancy Features #1 and #4
- Authored R12-R16, co-authored R1-R11
- Added unit tests
- Wrote README
- Wrote setup script

GitLab repository: https://git.uwaterloo.ca/m2waqar/cs-348-group-project