Weekly tasks

- Creating meeting notes for 2/20/2026 and 2/21/2026 meeting
- Aggregate team progress into week 1 progress report
- Create data set for AI
- Look into using PyTorch
    - Can transition to a different AI tool or pretrained AI
- Research data needed to train an AI and how to get desired output

1.5 hour creating meeting notes and aggregating weekly reports for week 1.

30 minute meeting on 2/21/2026

Looking through data in C.R.I.S.(1 hour)

Column Data:

Using the create attributes list, it is likely that whatever data that will be aggregated will need to be gathered in chunks for specific counties. For example, Collin county has over 4,000 incidents in 2025 alone. To create any sort of history (for the sake of argument 2020-2025), this may result in somewhere around 20,000 incidents per county. C.R.I.S only allows for 50,000 incidents at a time.

Through the attribute list, a CSV file can be created. This allows for specific attributes to be determined from columns. The following I believe are of the most interest:

- Crash ID (Mandatory Incorporation)

This is a unique identifier and thus while the C.R.I.S. id may not need to be used, some unique ID should be associated with each individual crash

- At intersection (Possible Incorporation)

This might be difficult to pull from reverse geocoding and I am not sure any map API would easily pull this information for the user, but it does exist.

- City (Mandatory Incorporation)

This should be used and should be easy to pull from API

- Year / Date / Day of Week / Time (Mandatory Incorporation)

This is useful to build the history of the road/city/county. The year may not be important but the day/date/time should be important to determine if one day over the other involves specific risks or if a particular time of day has more risks than another.

This may also allow us to create ‘negative data’ or data that shows when there weren’t crashes. It may be ideal to create some sort of code that goes through the csv file and creates incidents where there are no crashes. For example, we can assume that in every country or in every city there was at least one car on the road every hour for example. If we have a crash during an hour but not in another hour, we create an example based on the general conditions where there was not a crash. This may result in a column called crashed that would need to be populated either by hand or by program. **This requires further research on my part**

- Roadway Type (Possible Incorporation)

I was assuming that this would be like ‘highway’ or ‘service road’ but seems to be data like ‘4 OR MORE LANES’ or ‘2 LANE’ with most of the incidents being ‘No Data’ which will probably result in less than useful information. However this can be considered.

- Road Class (Mandatory Incorporation)

This is road types like ‘TOLLWAY’, ‘US & STATE HIGHWAYS’. ‘CITY STREET’, ‘COUNTY ROAD’, ‘FARM TO MARKET’, ‘NON TRAFFICWAY’. There may be others, but these are the most important. The road class is going to be important information to the AI. The data should be accessible by any type of road/map/geocode API and thus should be incorporated into the project

- Street Name/Street Number (Mandatory/Possible Incorporation)

Important for when or if we can get to determining specific streets. I don’t know how feasible picking how individual streets are at this phase of the project, it may be that the first step of the project is to determine how safe each county is and then maybe iterate to how safe each county and general location (urban vs rural) and then each street is. Thus street names may need to be added later or omitted in the first run through of data.

- Road Urban Type (Possible Incorporation)

Contains ‘RURAL’, ‘SMALL URBAN’, and ‘LARGE URBAN’. I think this would be a useful tag to have inside of the AI in order to determine if such a place poses a larger threat to a driver than another place. However, it may be difficult to call an API that would be able to reflect if the area the user is in is rural or urban. Hard coded data may be possible through making a city generally rural, lightly urban, or high urban. However, generalization might pollute the data

- Surface Type (Not incorporated)

This could have provided good data about possibly the type of road, the material it would be made out of. This might have done good for scalability or non map querying about if a particular material would be better for road construction than another. However more often than not it returns ‘No Data’ and for the entirety of Collin County for 2025 it returned ‘No data’ and thus should not be incorporated

- Surface Condition (Possible Incorporation)

This could be useful and contains tags like ‘1 - DRY’, ‘2 - WET’, ‘3 - STANDING WATER’, ‘4 - SNOW’, ‘5 - SLUSH’, ‘6 - ICE’ and possibly more though these are going to be the most common ones. This would be amazing information for the AI to have, but it is likely that there are no good APIs that will exist to return real time information about the state of such roads. Thus, the app would have to infer off of weather which could be an issue.

- Weather Conditions (Mandatory Incorporation)

Tags including ‘ 1 - Clear’, ‘2 - Cloudy‘, ‘3 - Rain, ‘4 - Sleet/Hail’, ‘5 - Snow’, ‘7 - Blowing Sand/Snow’, ‘98 - OTHER’, ‘99 - Unknown’ As well as others. It is likely that 1-5 are going to be the most likely conditions on the road.

The issue with this tag is that the weather API that we intend to use and this system of tagging weather may not be formatted the same way. Thus, our app may have to do some sort of conversion of weather type in order to return data that the AI can process.

The other issue may be that if there are truly 99 possible weather type, there may be some that are so unlikely to happen that it may be worth it to remove them from the data set. ‘7 - Blowing Sand/Snow’ will likely not occur often and so may be able to be removed from data. However, this could reduce how effective the final product is.

AI Research (2 hour)

I do not have any experience with AI and so while I know that large amounts of data are needed to create an AI, how this data should be formatted, if simply having the crash reports is enough data, and what pretrained models

exist are not things that are clear to me. Thus, this week I want to spend an hour or two doing general AI research. Below will be links that I explore be that youtube or general documentation.

- Training Your Own AI Model Is Not As Hard As You (Probably) Think (https://www.youtube.com/watch?v=fCUkvL0mbxI)

About 10 minutes in length. General idea about training an AI. Basically break a problem down and then see where AI can be applied and where AI shouldn't be applied. The general idea was to use code as opposed to LLMs or Pretrained models whenever possible. For this project, AI is the ideal use case, but large amounts of math can be done to create an algorithm which can work similarly, however an AI likely will be able to scale better than our own code.

Talk about pretrained models was not done in depth, the example used was image recognition. A pretrained model that does not take hours to run would be ideal however the following obstacles were brought up: Costs and large datasets. Given that only a few months are available for us to work part time on this project, waiting hours to train an AI is not ideal. I assume that a .csv file does not count as a large dataset but there will be thousands of rows of data and potentially hundreds of thousands of rows. The biggest overall concerns are training time and monetary costs.

- What is PyTorch

At this point in the project, PyTorch seems to be the AI tool that we are going to use, I want to understand what it is and how it works and maybe use the information to begin working on making sure the data is set up in a way that works.

I started with this video: What is PyTorch? (Machine/Deep Learning) (https://www.youtube.com/watch?v=fJ40w_2h8kk)

The ability for PyTorch aids in:
- Preparing data

This has continued off the idea of large data sets, I do not think that a .csv file even thousands of rows large will have the terabytes or petabytes that are required for a larger model. Thus, the earlier fears I had about size in regards to training time and monetary cost might be relieved.

It allows for iterating batches of testing data and ensuring that data is not fed in a static order. This may be effective, I do wonder how much of an effect that will have on AI that needs to be aware of time in order to make it's predictions. It might be handy to know that if rain is imminent crashes are more likely, however if it does hallucinate rain based on a particular day of the year, that may not be good but could be stopped by including multiple years of data.

- Building Model

Works well at building layers and removing linearity.

- Training the model
- Testing the model

List of sources to help me understand the What Is PyTorch Video:
- What is Deep Learning (https://aws.amazon.com/what-is/deep-learning/)
- What is Shuffling the Data? (https://medium.com/@sanjay_dutta/what-is-shuffling-the-data-a-guide-for-students-0f874572baf6)
- A Guild to Deep Learning Layers (https://adgefficiency.com/blog/guide-deep-learning/)

After some researching about what specific type of AI we would be using in order to solve this problem, I found this article.
- An introduction to Tabular AI (https://www.neuralk-ai.com/post/an-introduction-to-tabular-ai)

This seems to be the type of AI that will be used in order to pull out our results. This seems to be a general type of AI, though I imagine simpler than the Deep Learning AI provided by PyTorch. As opposed to an AI that needs to iterate several times over, this seems to be able to be trained once and then no more.

Furthermore, pretrained models seem to exist that would allow for quick implementation. There is a possibility through this that the general technologies being used could be switched.

https://www.youtube.com/watch?v=3jqdk0MkCLU This goes over the TabPFN tool which seems to do exactly what we need in providing if an incident will happen or not (1 or 0) as well as a general level of confidence in that output. Thus, TabPFN may be a good tool to switch to and will be an issue I bring up in the next meeting I have with my team.

Work done on the SSoT (1 hour)

Data Manipulation (1.5 hour)

To get an AI that predicts how safe or unsafe a road is, negative sampling will be needed to ensure that it views roads as a range of risks as opposed to just places where crashes happen. Before creating negative sampling, I wanted a general overview of the data that would exist in the .csv file.

To begin with I pulled the 2025 results, this is a small enough area that there would not be hundreds of cities to look over. However, it is big enough that the data and cities represented show off a diverse enough representation of cities in terms of populations. I used Copilot to assist in writing the Python file (csv_overview inside the backend repository) to query the general data inside the file.

The 2025 result for Collin County had 16870 crashes reported over about 30 cities. The crashes per city were:

```
Crashes by City (2025):
ALLEN                :  1049 crashes
ANNA                 :   356 crashes
BLUE RIDGE           :    65 crashes
CARROLLTON           :    11 crashes
CELINA               :   372 crashes
DALLAS               :   804 crashes
FAIRVIEW (COLLIN)    :   163 crashes
FARMERSVILLE         :   157 crashes
FRISCO               :  2292 crashes
GARLAND              :     1 crashes
JOSEPHINE            :    38 crashes
LAVON                :   147 crashes
LOWRY CROSSING       :     8 crashes
LUCAS                :   146 crashes
MCKINNEY             :  1650 crashes
MELISSA              :   340 crashes
MURPHY               :   243 crashes
NEVADA               :    37 crashes
NEW HOPE (COLLIN)    :    14 crashes
OUTSIDE CITY LIMITS  :   889 crashes
PARKER (COLLIN)      :    86 crashes
PLANO                :  5928 crashes
PRINCETON            :   388 crashes
PROSPER              :   299 crashes
RICHARDSON           :   494 crashes
ROYSE CITY           :    21 crashes
SACHSE               :   106 crashes
ST. PAUL (COLLIN)    :    21 crashes
VAN ALSTYNE          :    28 crashes
WESTON               :     9 crashes
WYLIE                :   708 crashes
Total Crashes: 16870 crashes
```

My immediate thoughts are around how negative sampling is going to need to use this. Plano has the highest rate of crashes and while it should probably be flagged generally as more dangerous than Garland that has a single crash

score, the reason why should be looked into. A weakness of this data is that it does not fully explain the reason why Plano is magnitudes higher than Garland. The population is comparable and some basic research indicates that both likely have highways inside of their borders.

My initial guess would be that Garland likely has lower speed limits than Plano, but this is conjecture.

Sources:
https://www.plano.gov/1187/Plano-Demographics
https://www.texas-demographics.com/garland-demographics

The problems are twofold: Negative Sampling at a constant rate may result in reinforcing behaviors such as Weston and Garland are completely safe places while Plano is incredibly dangerous. The second issue is data growth. General research that I have done suggests that the best way to do negative sampling is to have 'time slices' or every x amount of time we check if a crash happened. If it did, we create no data, if not we create non-crash data. We can reasonably assume that at least one car passed by a point every x amount of time. The issue with this becomes time scaling. Say we want our time slice to be every hour. For a year there are 8760 hours. We then decide how small to check:

- Counties

If we check by county and each county has at least one crash per hour of the year we get 8760 * 254 or 2,225,040 row of data

- Cities

We can check cities and if we were to slice every city in Texas we get even more data. With around 1,200 cities: 8760 * 1200 = 10,512,000 for a single year. This grows close to a gigabyte and given we have no funding, this is to be avoided.

- Roads

For all the roads in Texas, we could end up with billions of rows per which is not sustainable.

This, I think the best move is to abstract our data by either city or county and then become more specific as we move on

2/26/2026 Meeting (1 hour Long)

While the notes of this meeting are inside the github, this did provide some new insights. The team is in agreement that the smartest thing to do at present is to focus on one county for now, Dallas County and slice based on roads. However, it would be impossible to get the information for every single road so roads that do not meet a certain level of criteria or X number of crashes in a given year will be removed.

Data Manipulation Pt 2 (3 hours)

To start we will pull data from C.R.I.S. This will be made up as so:

Crash ID
City
County
Crash Date
Crash Time
Crash Year
Day of Week
Hour of Day
Road Class
Rural Urban Flag
Street name
Surface Condition
Weather Condition

Crash

Crash will need to be added on, 1 being a positive (crash happened) and 0 being negative, crash did not happen

Crash ID and Date are not going to be necessary for the larger data set most likely but will be included for now in order to make the process of negative sampling easier at the moment. Hour of day will also be used to gauge the time slice as well.

Switching to Dallas County from Collin County brings the total up from around 16,000 to around 40,000 results. This will result in a much larger final data set than Collin County. It is also likely that it will not represent an as diverse population as the Collin County area but will result in AI that represents a larger and more populated area. The .csv file is around 7.5 megabytes large.

This is the initial result per city

```
Crashes by City (2025):
ADDISON              :   352 crashes
BALCH SPRINGS        :   931 crashes
CARROLLTON           :  1728 crashes
CEDAR HILL           :   540 crashes
COCKRELL HILL        :    70 crashes
COMBINE              :     1 crashes
COPPELL              :   327 crashes
DALLAS               : 25276 crashes
DESOTO               :   547 crashes
DUNCANVILLE          :   839 crashes
FARMERS BRANCH       :   992 crashes
GARLAND              :  3225 crashes
GLENN HEIGHTS        :   141 crashes
GRAND PRAIRIE        :  2211 crashes
GRAPEVINE            :    33 crashes
HIGHLAND PARK        :   152 crashes
HUTCHINS             :   358 crashes
IRVING               :  4695 crashes
LANCASTER            :   846 crashes
LEWISVILLE           :    17 crashes
MESQUITE             :  2367 crashes
OUTSIDE CITY LIMITS  :   114 crashes
RICHARDSON           :  1496 crashes
ROWLETT              :   583 crashes
SACHSE               :   345 crashes
SEAGOVILLE           :    84 crashes
SUNNYVALE            :   257 crashes
UNIVERSITY PARK      :   123 crashes
WILMER               :   144 crashes
WYLIE                :     9 crashes
Total Crashes: 48803 crashes
------------------------------
```

During the meeting it was discovered that Garland which had only a single crash was located primarily inside of Dallas County and here it is reflected that Garland has many more crashes than one. However, there are also many counties that have very few crashes. It is likely that there is some county overlap. One potential solution to this would be to blank out cities that overlap other counties. For now, this will not be done so as to preserve the data. However, the process of refining the AI model later may be to blank out the data

The next step is to get the number of crashes per street, regardless of city. We will take the median and then reclassify the roads from there.

The general analysis of the new road data is this:

```
Total unique roads: 4277
Total crashes: 48802
Median crashes per road: 1
Average crashes per road: 11.41
```

Now we need to decide what our ‘problem roads’ are. Basically, this project we designed was to create a way to view the roads and ideally keep in mind the history so the AI could make informed decisions about truly how risk the road was. However, with 4277 roads represented in the Dallas area alone, this will become hard with time slicing. If we did every hour of the year, we would get about 37,466,520. This is a large amount of data for a single city in a single year. Thus, during the meeting, the idea of ‘danger roads’ were proposed where we cut roads off at the median. This was before the median was discovered though, so I will also consider using the average to cut off the roads.

Amount of roads that have incidents above the average: 415

Amount of roads that have incidents above the median: 1873

Both of these numbers were taken from a .csv saved in the backend: craches_by_road.csv

If we use the time slice of one hour per year, we get 3,635,400 ((415 + 1) * 8760) entities if we cut off at the average and 16,407,480 ((1873 + 1) * 8760) entities if we cut off at the median. These are pretty far off and for now we are going to cut off at the average for a reduced data set size. We add one to both of these numbers as any number below will have the road turn ‘Non-Critical Road’ which will be a type of road alongside all the other types. It is possible that when the actual expanded data set is computed, the category for what a ‘critical road’ is is refined.

This new file is DallasCounty2025_Modified.csv.

From here the crash column was added and a new file was added
DallasCounty2025_Modified_with_crash_label.csv

This allowed the general data structure to go from

```
Crash ID,City,County,Crash Date,Crash Month,Crash Time,Crash Year,Day of Week,Hour of Day,Road Class,Rural Urban Type,Street Name,Surface Condition,Weather Condition
```

To

```
Crash ID,City,County,Crash Date,Crash Month,Crash Time,Crash Year,Day of Week,Hour of Day,Road Class,Rural Urban Type,Street Name,Surface Condition,Weather Condition,Crash
```

At this point in time, I am running out of time before the meeting and so I intend to simplify the process slightly. From here, I have created a function that simplifies some amount of data to generalize the road:

```
Road Name,Rural Urban Type,County,City
```

From here, after the simplification, we have 415 rows of data. The last step for this week is to create dummy data. A new file will be created full of the dummy data. For every row here, 8,760 rows of data will be generated that will demonstrate no crashes.

At this point, I do not have enough time before the meeting to take into account the weather and both implement and test this. For the negative sampling I am randomizing the weather and road conditions, but everything else from Urban Rural Type to Road Name and Road Class has been taken from the .csv provided. This is not the full data that can be used to accurately train the model, but it is a starting point.

The file called negative_sample.csv has 3,626,641 rows inside of it in comparison to the 48,804 that we started with. This might be a giant ballooning of data that may not be the most necessary but it is a large dataset and the AI model should not be under the assumption that every row should lead to a crash at this point. I am worried that this means that the scope of the AI is going to be limited. Given that there are almost a million incidents (as per my memory not necessarily per C.R.I.S.) in the year 2025 alone, a potential issue may be that a single year could expan into 20-30 million rows of data. This is small by model training data, but given that our apparent budget seems to be none, it would be preferable to avoid gathering monstrous amounts of data.

However, back to the data manipulation, the last thing to create this rough draft of model training data is to combine the negative samples and the positive samples provided by C.R.I.S. However, if a road has a crash in an hour, the corresponding negative sample should not be concatenated onto the csv file. These will be placed into a a csv file called TrainingData.csv

```
Starting training data combination...

======================================================================
COMBINING CRASH AND NEGATIVE SAMPLES
======================================================================
Reading crash data...
Reading negative samples...
Building crash event index...
Total crash events indexed: 42444
----------------------------------------------------------------------
Filtering negative samples...
Negative samples kept: 3584196
Negative samples skipped (conflicts): 42444
----------------------------------------------------------------------
Combining datasets...
----------------------------------------------------------------------

[OK] Training data created successfully!
  Crash samples: 48,803
  Negative samples (filtered): 3,584,196
  Total training records: 3,632,999
  Crash/Negative ratio: 1:73
  Output file: c:\<>\Ai Road Report\Road-Report-AI-Backend\csv\TrainingData.csv
======================================================================

[OK] Training data ready!
```

This process took around 5-7 minutes to complete, but a training record is complete. A README will be in the Backend Repository that goes over the general steps to create the TrainingData.csv as well as what each of the several .csv created along the way do.

The next week will be about streamlining this, about not hardcoding any of the file names and making the weather and road conditions reasonable

General Documentation (.5 hours)

Personally documenting the functions inside of the files csv_editing.py and csv_overview.py as well as creating the general README file to explain the process