Sprint 3 - Artifact

Features:

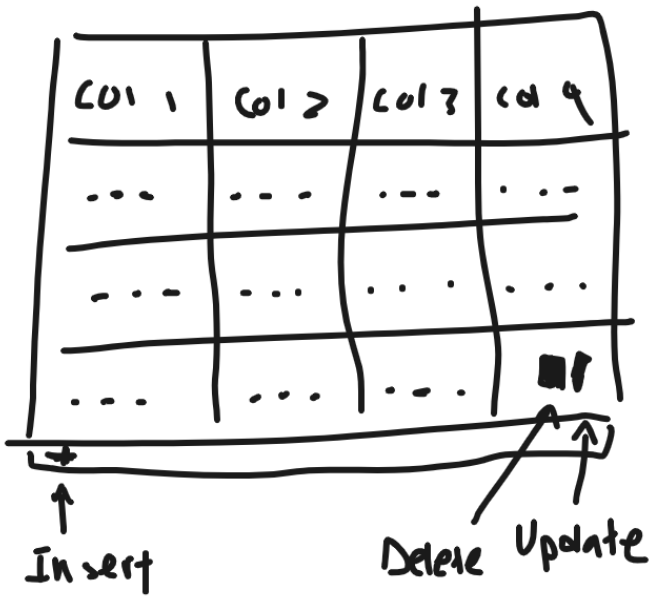
- Insert/Update/Delete
  - Summons number will be used as an ID
  - API calls that need to be made/handled by the frontend and backend:
  - Insert: PUT /endpoint with JSON data:
    {
    "summons number": <id>,
    … every field below is optional and should be handled by the backend (for example…)
    "issue date": <new date>,
    "plate number": <new plate number>
    }
  - Update: Same JSON body as INSERT endpoint, should only update the entry that matches the `summons number`. PUT /endpoint
  - Delete: DELETE /endpoint, body is JSON data:
    { "summons number": <id> }
  - Can be update/insert column:
    - // summons number //user doesn't input, length = 10
    - // Plate ID //must 0 < length <= 7 string
    - // Registration State //string length = 2
    - // Issue Date //must //string length <= 23
    - // Violation Time //string
    - // Violation Code //must 1-99 int
    - // Vehicle Make //string
    - // Vehicle Body Type //string
    - // Vehicle Year //string
    - // Street Name //string
    - // County //string

    [
    {
    "summons number" : …..;
    "Plate ID" : … ;
    ….
    },
    {
    "summons number" : ,...;
    "Plate Id" : ….;
    …
    }

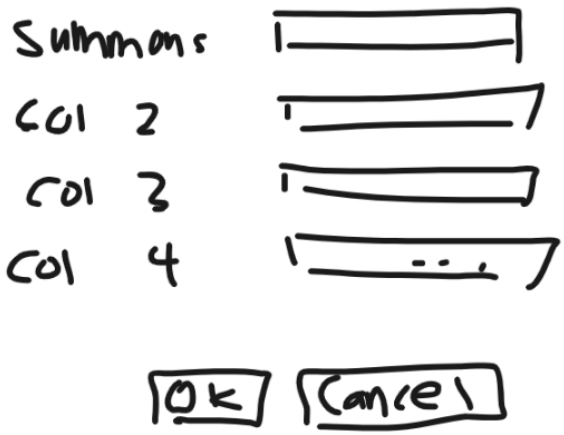
]

- User Stories
    - Insert: As a user, I want to input another violation which occurred in NY.
    - Delete: As a user, I want to remove a violation which occurred in NY.
    - Update: As a user, I want to update incorrect data for a given violation.
    - Backup: As a user, I want to save my data before I edit it so data is not lost.
    - Import: As a user, I want to import a dataset with various edits and changes to our data set

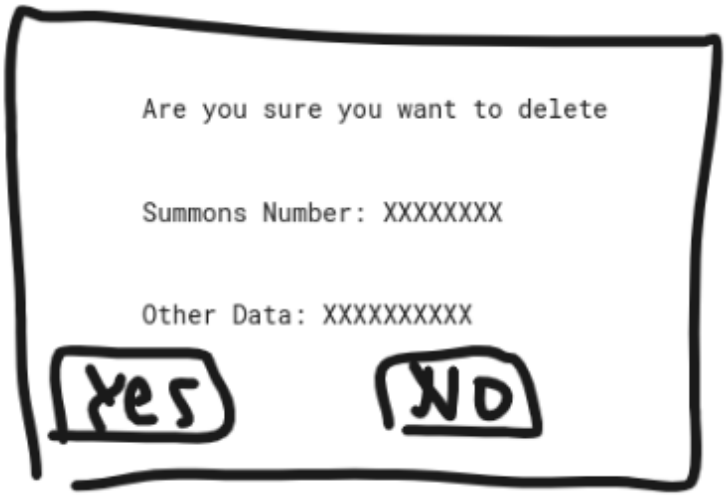
GUI Design and Test Cases:



For insert, there will be a plus icon at the bottom of the data table that opens a form (same with update). There will be a trashcan and pencil icon to the right of every row for delete and update.
Insert/update form:



Delete form:



TEST CASES:

- Insert:
    - Test case 1: as a user, I first press the “Insert” button
        - Correct Output: the website displays a form with 10 input text fields for Plate ID, Registration State, Issue Date, Violation Time, Violation Code, Vehicle Make, Vehicle Body Type, Vehicle Year, Street Name, County along with an “X” button at the upper right corner and a submit button at the bottom
    - Test case 2: as a user, I press the “X” button
        - Correct Output: Form is closed, no create request sent to backend
    - Test case 3: as a user, I enter in the information requested by the form and click the submit button

        - Correct Output: website sends a create request to backend. The backend creates a new record with the information from the form
    - Test case 4: as a user, I fail to enter in all the boxes needed and press submit
        - Correct Output: form does not submit and prompts user to enter in input into missing fields
- Delete:
    - Test case 1: as a user, I first search for records by entering violation code, violation time, street name, and plate ID
        - Correct Output: the website displays a table with existing records for the inputs provided, next to each row there is an update and a delete button
    - Test case 2: as a user, I press the delete button next to the record I want to delete
        - Correct Output: website displays a delete form with a confirmation of delete button and an exit button in upper-right corner
    - Test case 3: as a user, I click the delete confirmation button
        - Correct Output: website sends a delete request to the backend. The backend deletes the record and the frontend removes the record from the table
    - Test case 4: as a user, I click the exit button in upper-right corner
        - Correct Output: The delete form is closed and no delete request sent to backend.
    - Test case 5: as a user, I search for records by only entering violation code
        - Correct Output: Form prompts you to input more data
- Update:
    - Test case 1: as a user, I first search for records by inputting violation code, violation time, street name, and plate ID
        - Correct Output: the website displays a table with existing records for the inputs provided, next to each row there is an update and a delete button
    - Test case 2: as a user, I press the edit button next to the record I want to update
        - Correct Output: the website displays an update form for the selected record. There is an “X” button in the upper right corner and an update button at the bottom.
    - Test case 3: as a user, I press the “X” button
        - Correct output: Form exits, and no update request is sent to backend

        - Test case 4: as a user, I enter in the fields I want to update and pres the update button
            - Correct output: website sends an update request to the backend. The record is updated by the backend, and the website displays the record with the updated information
        - Test case 5: as a user, I delete a field and don't enter in updated information and press update
            - Correct output: Form prompts the user to input data into empty field
- Backup/Import: Checking to make sure that Backup correctly saves the dataset and that import correctly brings in the correct data.

Done list of last sprint

- Implemented search box on frontend
    - [finished by Heng Tan and Diane Shan and verified by everyone]
- Added hamburger menu for different search areas
    - [finished by Diane Shan and verified by everyone]
- Added data table on frontend
    - [finished by Heng Tan and verified by everyone]
- Implemented search function
    - [finished by Justin Pham and Ethan Bayer and verified by everyone]
- Implemented parse function
    - [finished by Justin Pham and Ethan Bayer and verified by everyone]

ToDo Tasklist for this sprint

- Optimize search and parse functions
    - Acceptance Criteria: Search return correct value and runs in a reasonable amount of time
- Backend program to overwrite existing CSV
    - Acceptance Criteria: Method successfully writes existing csv with new values
- Add edit button next to each searched record
    - Acceptance Criteria: Button appears next to each searched record. Clicking on button opens update form.
- Add delete button next to each searched record
    - Acceptance Criteria: Button appears next to each searched record. Clicking on button opens delete form.
- Backend method to delete one record
    - Acceptance Criteria: Method deletes one record specified by the user
- Backend method to add one record
    - Acceptance Criteria: Method adds one record specified by the user

- Backend method to update one record
    - Acceptance Criteria: Method updates one record specified by the user
- Create frontend form for updating records
    - Acceptance Criteria: form receives data input from user and send information to backend when user submits form
- Create frontend form for adding records
    - Acceptance Criteria: form receives data input from user and send information to backend when user submits form
- Create frontend form for deleting records
    - Acceptance Criteria: form receives yes/no from user and confirms deletion in backend