Use-case:	GUI-2: Display the current district plan by default
Primary actor:	General User
Goal in context: *one sentence description*	After selecting a state either from the map or the dropdown, by default, the user should be shown the current (2022) State Assembly district plan displayed on the centered state map at a zoom level appropriate to the size and location of the state.
Preconditions:	The map component was loaded.
Trigger:	The state was selected by the user.
Scenario: *step by step actions taken by the system to achieve goal in context*	1. The client displays a loading screen overlaying the map component while the data is being fetched. 2. The client makes a GET request to the server for the selected state's current district plan (see Interface Document for specific API). 3. The server returns the current state's district plan as a geojson object to the client. 4. The client takes the returned geojson and passes it to the map component for rendering. 5. The map component clears the loading screen and renders the provided geojson in the map and outlines the district boundaries for the provided state.
Exceptions: *Special cases that involve deviations from scenario above*	1. The server returns HTTP status code 500, signifying geojson will not be returned. The client's loading screen overlay will display a message to the user, signifying they should click the reset button on the web app to try again. 2. The client's get request is met with HTTP status code 404, signifying that the request is unable to be sent to the server. The client's loading screen overlay will display a message to the user, signifying they should click the reset button on the web app to try again.
Priority: *Build number (with reasons)*	Build 1 because this is the default state for when a state is selected.
Open issues: *List of open interface questions or unresolved issues*	1. Should district numbers be displayed inside the outlined boundaries of the state?

Use-case:	GUI-5: Display photo of district representatives
Primary actor:	General User
Goal in context: *one sentence description*	A recent photo of each district representative will be displayed, either directly in the table or by clicking on a component in the table or by hovering over a component in the table.
Preconditions:	A state has been selected.
Trigger:	The user clicks on the photo icon component in the representative's row in the table.
Scenario: *step by step actions taken by the system to achieve goal in context*	1. The client makes a GET request to the server for the representative's photo (see Interface Document for specific API). 2. The server returns a URL to a recent head shot of the representative. 3. The client displays a popup dialog with the representative's name and an image component, whose source is the URL returned by the server.
Exceptions: *Special cases that involve deviations from scenario above*	1. The server returns HTTP status code 500, signifying the URL for the representative could not be fetched. The client's popup dialog will display the message "Unable to load representative's picture". 2. The client's get request is met with HTTP 404, signifying that the request is unable to be sent to the server. The client's popup dialog will display the message "Unable to load representative's picture". 3. The client is unable to load the image component with the returned URL from the server. The client's popup dialog will display the message "Unable to load representative's picture".
Priority: *Build number (with reasons)*	Build 3 because this is a feature that requires specific user interaction in the interface to activate and isn't user facing by default.
Open issues: *List of open interface questions or unresolved issues*	1. Is a URL the best way to store representatives' photo in the database? Or should we instead store the actual bytes of the image? Storing image bytes will require extra space in the database and extra bandwidth for the request, however will avoid the issue of a URL being malformed or unavailable when loaded by the client. 2. Would a better UI design be to automatically show a tooltip displaying the image after the user hovers over the row for a certain period of time?

Use-case:	GUI-8: Reset page
Primary actor:	General User
Goal in context: *one sentence description*	When the user clicks a reset button, the GUI will reset to the condition before the user selected a state.
Preconditions:	The user has modified the user interface into a state that differs from the default state.
Trigger:	The user clicks the reset button in the top right corner of the web app.
Scenario: *step by step actions taken by the system to achieve goal in context*	1. The client changes the current page back to the state map overview page, with New York selected as the default state. 2. The client and server perform the actions listed in GUI-2 to load the data for New York and display it appropriately
Exceptions: *Special cases that involve deviations from scenario above*	See those listed in GUI-2
Priority: *Build number (with reasons)*	Build 2 because the reset action is not user-facing by default, but is important for restoring the default state after extensive testing or to attempt to reload data after failures mentioned in previous exceptions sections.
Open issues: *List of open interface questions or unresolved issues*	1. Should there be a confirmation dialog for the user to confirm they want to reset the web app after clicking the initial reset button in the header bar? 2. Should there be an alert confirming that the page was reset successfully after the scenario is executed?

Use-case:	Prepro-1: Integrate multiple data sources
Primary actor:	System programmer
Goal in context: *one sentence description*	Integrate and merge US Census data (population, both for total and for the any opportunity groups), precinct data (boundary, name, demographics, etc.), and existing district data (boundary, name, district#, etc.). Geographic boundary data should be converted (if necessary) to a consistent format (e.g., GeoJSON).
Preconditions:	None
Trigger:	Data collection phase
Scenario: *step by step actions taken by the system to achieve goal in context*	1. US Census data provides shape files based off of block groups for a state. It also provides demographic data, such as racial population, for each block group. 2. Precinct data, (Redistricting Data Hub) provides geographic boundaries of the precincts in a state. 3. Utilizing the GQIS software, we can overlay the block group coordinates with the precinct boundaries and determine which precincts belong to which block groups. We can then use the demographic data per block group to associate demographic data with a precinct. 4. The provided precinct data from RDH associates each precinct with their state assembly district. We can then aggregate said precinct data to provide accurate district data to store in our database. 5. The associated precinct boundaries and state assembly district boundaries will all be converted from shape files to the GeoJSON format for use with the Leaflet mapping library.
Exceptions: *Special cases that involve deviations from scenario above*	1. Not all precinct boundaries provided from RDH are accurate, meaning when overlaid with the block groups, no association can be made. This then requires manual intervention to find a precinct's coordinates and associate it with a block group for data mapping.
Priority: *Build number (with reasons)*	Build 1 as collected information is used in determining racial groups of focus in a state, required to build appropriate GUI components (i.e. which races to display a heat map for) and appropriate server endpoints (i.e. races allowed as parameters to server requests)
Open issues: *List of open interface questions or unresolved issues*	1. If two or more precinct boundaries overlap in a block group boundary, demographic data may not be truly accurate for a precinct because the area of a precinct in said block group boundary needs to be estimated to divide said data amongst a precinct.

Use-case:	GUI-3: Display demographic heat map
Primary actor:	General User
Goal in context: *one sentence description*	When the user selects a minority group from a drop-down menu, the heat map for the demographic group in the state will be displayed.
Preconditions:	A state is selected in the user interface and the map with the state's district boundaries is loaded.
Trigger:	The user selects a race from a drop-down menu.
Scenario: *step by step actions taken by the system to achieve goal in context*	1. The client displays a loading screen overlaying the map component while the data is being fetched. 2. The client makes a GET request to the server for the selected race's population information per district (See Interface Document for specific API). 3. The server queries the database for population data per race per precinct. 4. The server aggregates connected precincts by district and forms population data per race per district. 5. The server returns a JSON dictionary object with each district as a key and the value being the percentage of said race in said district. 6. The loading screen overlaying the map is removed. 7. The client calculates a monochromatic hue scale for the selected race and displays a legend underneath the map with the calculated scale. 8. The client colors in each district with their corresponding hue based on the provided percentage from the server's response.
Exceptions: *Special cases that involve deviations from scenario above*	1. The server returns HTTP status code 500, signifying json will not be returned. The client's loading screen overlay will display a message to the user, signifying they should click the reset button on the web app to try again. 2. The client's get request is met with HTTP status code 404, signifying that the request is unable to be sent to the server. The client's loading screen overlay will display a message to the user, signifying they should click the reset button on the web app to try again.
Priority: *Build number (with reasons)*	Build 3 because this is a feature that requires specific user interaction in the interface to activate and isn't user facing by default.
Open issues: *List of open interface questions or unresolved issues*	None