

MOFFITT Contract 23-MCC02420 -

Module 2

Table of Contents

Module 2 - Frontend	1
Lineage pane	3
Multi-Omics pane	14
Module 2, Backend 1	19
Main functions providing backend 1 functionalities	19
Module 2 - Backend 2	25
Main functions providing Backend 2 functionalities	25
Installation	31
Updating the database information	31
Package information	32
Description	32
Dependencies	32
License information	33
MIT License	33

Module 2 - Frontend

Module 2's user interface is divided into two distinct panels. On the left panel, users will find the lineage tracing module, which processes phenotypic data. The right panel is dedicated to the multi-omics module, providing genotypic data information.

- Lineage tracing module

Users have the ability to upload phenotypic information into Module 2 by dragging images into the lineage tracing module window and filling out required fields, including id, from, flask, cellCount, media, and specifying whether it's a seed or harvest event.

For both certified and uncertified users: Images are segmented using CLONEID cellpose backend code. Users can review segmentation results and exclude images with incorrect results. Resulting segmentation information is uploaded to the database. Certified users download segmentation results instead, due to scalability concerns.

- Multi-omics module

Users have the ability to upload genotypic information into Module 2 by dragging a folder containing an 'spstats' file into the multi-omics module. The folder also includes several other files as specified in Cloneid R documentation about CloneProfiles.

For both certified and uncertified users: Module 2's multi-omics pane will report back the number of profiles saved into the Perspective table and display a pie chart showing the clonal representation.

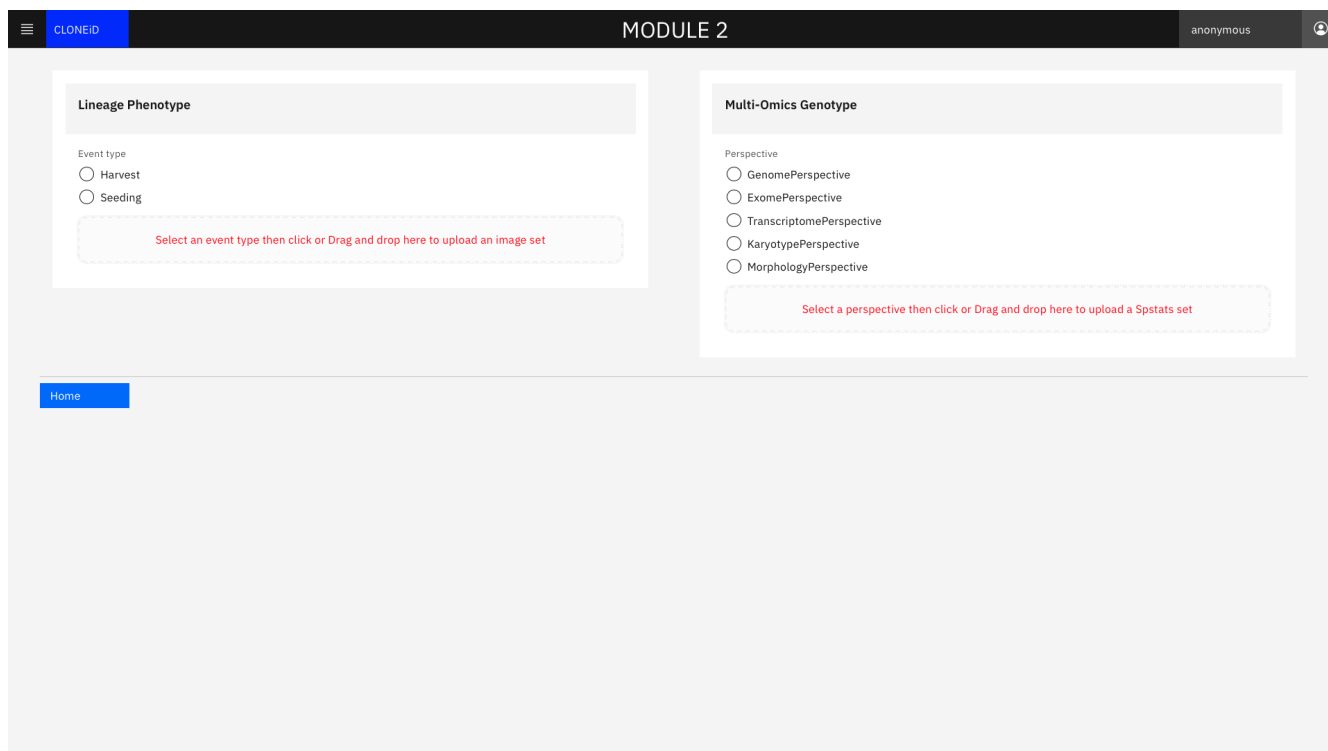


Figure 1. Home page

- Home page is handled by the following code:

webportal/frontend/src/routes/(cloneId)/module2/home/+page.svelte

```
...
<Row>
  <!-- Row for Lineage Pane and Multiomics Pane -->

  <!-- LINEAGE PANE -->
  {#if $phenotypeShowPane || $phenotypeShowOverlayImages }
  <!-- Conditional rendering of Lineage Pane based on the phenotype show pane or overlay images properties -->
    <Column>
      <LineagePaneWebWorker />
    </Column>
  {/if}

  <!-- MULTIOMICS PANE -->
  {#key $genotypeShowPane}
    {#if $genotypeShowPane}
      <!-- Conditional rendering of Multiomics Pane based on the genotype show pane property -->
      <Column>
        <MultiOmicsPane />
      </Column>
    {/if}
  {/key}
</Row>
...
```

+page.svelte file contains the main UI functions for Module 2. Sub panes are processed by

LineagePaneWebWorker.svelte and MultiOmicsPane.svelte in components folder:

Lineage pane

- webportal/frontend/src/routes/(cloneId)/module2/home/components/LineagePaneWebWorker.svelte
 - Event handling and data processing for lineage pane.

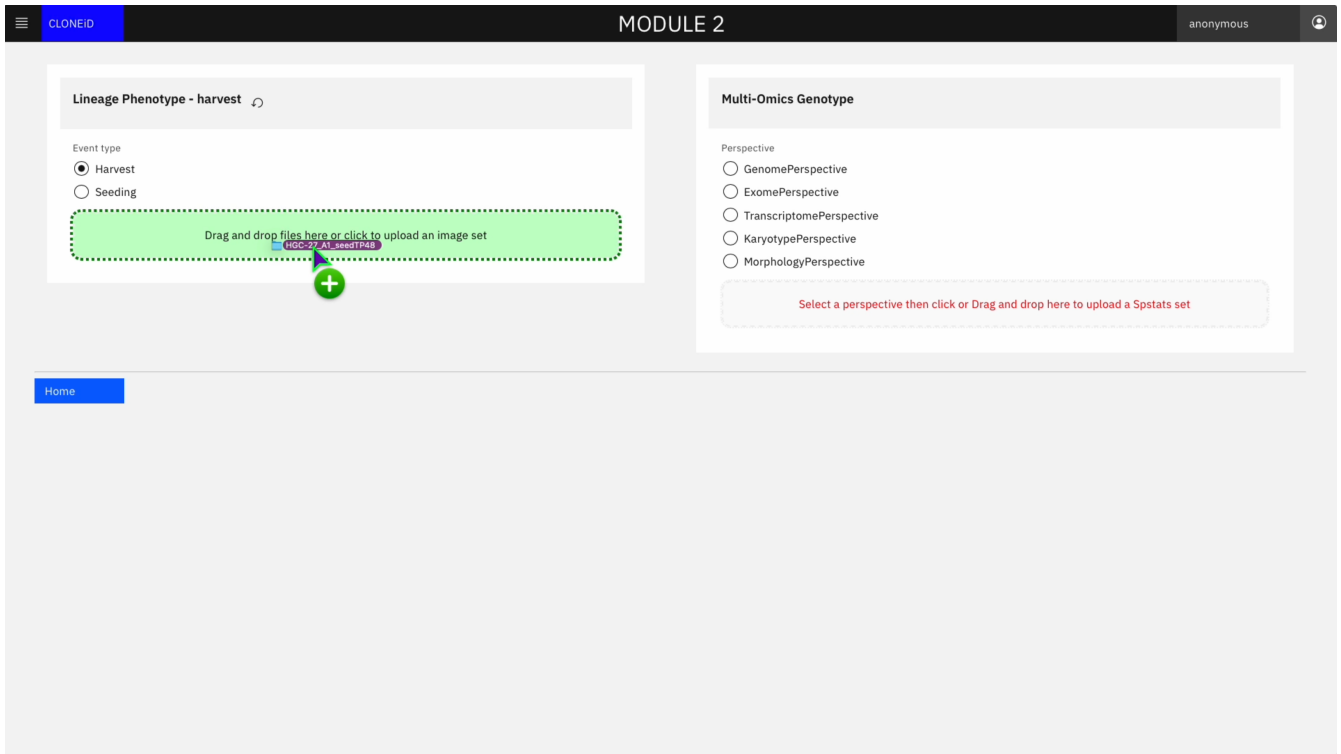


Figure 2. Lineage pane Drag and Drop

- Drag and Drop handling.

Dropzone Component:

webportal/frontend/src/routes/(cloneId)/module2/home/components/LineagePaneWebWorker.svelte

```
...
<Dropzone
  on:drop={async (e) => {
    if (selected_phenotype_event === '') {
      alert('Select an event type, then try again.');
```

```

    }
    if (!(fileCount >= 1 && fileCount <= 4)) {
      alert('Wrong file count; Try again');
      selected_phenotype_event = '';
      e.preventDefault();
      $loadingIndicatorToggle = false;
      insideDrop = false;
      return;
    }
    for (let index = 0; index < fileCount; index++) {
      const name = acceptedFiles[index].name;
      const breakAt = /_([0-9]+)_x_ph_/;
      const splitted = name.split(breakAt);
      files.confirmed[index] = {
        fullname: name,
        name: splitted[0],
        timeStamp: timeStamp,
        shortname: splitted[0] + '...' + splitted[2]
      };
      if (index > 0) {
        if (files.confirmed[index].name !== files.confirmed[index - 1].name) {
          console.log(files);
          files = {
            accepted: [],
            rejected: [],
            confirmed: []
          };
          console.log(files);
          alert('File names are inconsistent. Try again');
          selected_phenotype_event = '';
          e.preventDefault();
          $loadingIndicatorToggle = false;
          insideDrop = false;
          return;
        }
      }
      lineageId = splitted[0];
    }
    await handleFilesForPhenotypeDropZone(lineageId).then(async (entries) => {
      if (!entries) {
        selected_phenotype_event = '';
        e.preventDefault();
        $loadingIndicatorToggle = false;
        return;
      }
      if (entries) {
        populatePhenotypeFormEntriesFrom(entries);
      } else {
        selected_phenotype_event = '';
        e.preventDefault();
        $loadingIndicatorToggle = false;
        return;
      }
      files.accepted = acceptedFiles;
      files.rejected = fileRejections;
      preview_phenotype_files = true;
    });
    insideDrop = false;
  }}
  containerStyles={insideDrop
    ? 'outline-style: dotted; outline-color: green; outline-width: 4px; display: flex; flex-direction: column;
    align-items: center; padding: 20px; border-radius: 12px; background-color: #00ff0044; color: #111111; cursor: pointer;
    transition: border 1.24s ease-in-out;'
    : selected_phenotype_event
    ? 'outline-style: dotted; outline-color: blue; outline-width: 2px; display: flex; flex-direction: column;
    align-items: center; padding: 20px; border-radius: 12px; background-color: #fbfbfb; color: #111111; cursor: copy;'
  }

```

```

transition: border 0.24s ease-in-out;'
      : 'outline-style: dotted; outline-color: #f0f0f0; outline-width: 2px; display: flex; flex-direction: column;
align-items: center; padding: 20px; border-radius: 12px; background-color: #fbfbfb; color: #111111; cursor: not-allowed;
transition: border 0.24s ease-in-out;}'
    on:dragenter={() => {
      clearPhenotypeFormEntries();
      if (
        selected_phenotype_event === 'harvest' ||
        selected_phenotype_event === 'seeding'
      ) {
        insideDrop = true;
      }
    }}
    on:dragleave={() => {
      insideDrop = false;
    }}
    multiple
    noClick={selected_phenotype_event === '' ? true : false}
    disabled={false}
  >
    <span style={selected_phenotype_event === '' ? 'color:red;' : ''}>
      {selected_phenotype_event === ''
        ? 'Select an event type then click or Drag and drop here to upload an image set'
        : 'Drag and drop files here or click to upload an image set'}
    </span>
  </Dropzone>
  ...

```

- The Dropzone component is used to create an area where users can drag and drop files. It has multiple event listeners to handle the drag-and-drop functionality and to manage the styles of the dropzone based on user interactions and the state of the component.
- DropZone event listeners.
 - on:drop: (event) - This event listener is triggered when a file or folder is dropped in the drop zone area.
 - if no files are dropped or if the file count is not between 1 and 4, alerts are shown and the process stops.
 - File Processing:
 - Extracts lineageId from filenames using regex `/[0-9]+x_ph.`
 - Ensures consistent naming; if inconsistent, clears accepted files, shows an alert, and prevents further actions.
 - Asynchronous Handling:
 - Calls backend `handleFilesForPhenotypeDropZone` with lineageId to fetch image info.
 - If no entries are returned, event is prevented from executing further actions.
 - If entries are valid, populates form fields.
 - on:dragenter
 - clears the existing form entries and checks if the selected phenotype event is 'harvest' or 'seeding'. If so, it sets insideDrop to true to highlight the dropzone.
 - on:dragleave
 - sets insideDrop to false, resetting the highlight state.

- Dropzone Styles:
 - The containerStyles attribute dynamically changes the styles of the dropzone.
 - When files are being dragged over, the dropzone is highlighted with a green border. If an event type is selected, the dropzone shows a blue border and indicates it is ready for files to be dropped. If no event type is selected, the dropzone is greyed out and displays a message instructing the user to select an event type.

The screenshot shows the CloneID Module 2 interface. The top navigation bar includes a menu icon, 'CLONEID', 'MODULE 2', 'anonymous', and a user icon. The main content area is divided into two panels. The left panel, titled 'Lineage Phenotype - harvest', contains a form with the following fields: 'Event type' (radio buttons for 'Harvest' and 'Seeding'), 'Image 1' through 'Image 4' (text input fields), 'Id' (text input field), 'From' (text input field), 'Flask' (dropdown menu), 'Cellcount' (text input field), 'Media' (dropdown menu), and 'Timestamp' (text input field). Below the form are 'Cancel' and 'Submit' buttons. The right panel, titled 'Multi-Omics Genotype', contains a 'Perspective' section with radio buttons for 'GenomePerspective', 'ExomePerspective', 'TranscriptomePerspective', 'KaryotypePerspective', and 'MorphologyPerspective'. Below this is a dashed box with the text 'Select a perspective then click or Drag and drop here to upload a Spstats set'. A 'Home' button is located at the bottom left of the interface.

Figure 3. Lineage pane Form handling

- Form handling.

`webportal/frontend/src/routes/(cloneId)/module2/home/components/LineagePaneWebWorker.svelte`

```
<Form
  enctype="multipart/form-data"
  on:submit={async (e) => {
    $loadingIndicatorToggle = true;
    e.preventDefault();
    if (
      (selected_phenotype_event === 'harvest' &&
        !(imageId && from && cellCount && true)) ||
      (selected_phenotype_event === 'seeding' &&
        !(imageId && from && cellCount && media && flask && true))
    ) {
      $loadingIndicatorToggle = false;
      alert('Missing information, try again.');
```

```

}
submitDisabled = true;
lineageOnlyLayout();
userHash = getUserHash();
let images_are_uploaded = false;
let formData = new FormData();
timeStamp = new Date().getTime(); // time stamp set 2
formData.append('t', timeStamp.toString());
formData.append('h', userHash.toString());
for (let i = 0; i < files.accepted.length; i++) {
    formData.append(`f${i}`, files.accepted[i], files.accepted[i].name);
}
mi_notif('AUPLOADING');
// <!-- -->
// <!-- UPLOAD IMAGES -->
// <!-- -->
await fetch('?/uploadallphenotypeimages', {
    method: 'POST',
    body: formData
})
    .then(async (res) => {
        const x = await res.json();
        let d = {};
        if (x.type === 'success' && x.status === 200) {
            d = decoderesponse(x);
            images_combined_sha512 = d.s;
            images_are_uploaded = d.c == files.accepted.length ? true : false;
            await mi_notif('FUPLOADED1000');
        } else {
            await mi_notif('EUPLOADING2000');
        }
    })
    .catch(async (e) => {
        console.log('uploads', e);
        await mi_notif('EUPLOADING2000');
    });

if (images_are_uploaded && images_combined_sha512 !== '') {
    // <!-- -->
    // <!-- CELLPOSE PROCESS IMAGES -->
    // <!-- -->
    media = media === '0' ? 'NULL' : media;
    media = media ? media : 'NULL';
    flask = flask ? flask : 'NULL';
    cellCount = cellCount ? cellCount : 'NULL';

    formData = new FormData();
    for (let i = 0; i < files.accepted.length; i++) {
        formData.append(`f${i}`, files.accepted[i], files.accepted[i].name);
    }
    formData.append('timestamp', timeStamp.toString());
}

```

```

        formData.append('t', timeStamp.toString());
        formData.append('userhash', userHash.toString());
        formData.append('username', $userName.toString());
        formData.append('h', userHash.toString());
        formData.append('s', images_combined_sha512.toString());
        formData.append('imageid', imageId.toString());
        formData.append('from', from.toString());
        formData.append('media', media.toString());
        formData.append('flask', flask.toString());
        formData.append('cellcount', cellCount.toString());
        formData.append('event', selected_phenotype_event.toString());
        formData.append('dishsurfacearea', dishSurfaceArea_cm2.toString());
        formData.append('flaskitems', flaskItems.toString());
        formData.append('mediaitems', mediaItems.toString());
        formData.append('waitforresult', 'N'.toString());

        cellpose_processing = !false;

        const unused = await fetch('?/processphenotype', {
            method: 'POST',
            body: formData
        });

        const jinfo = await unused.json();
        const jinfod = decoderesponse(jinfo);
        txid = jinfod.i;
        $loadingIndicatorToggle = true;
    } else {
        await mi_notif('EPROCESSING3000');
        alert('Uploading Error');
        resetLayout();
        return false;
    }
    return true;
}}
>
...
</Form>

```

- Cancel Button: Resets form state and clears fields.
- Submit Button: Validates and submits form according to logic described below.
- Form event listeners.
 - on:submit
 - The form checks if the required fields are filled for both harvest or seeding events. If any field is missing, an alert is displayed, and the submission process is stopped.
 - Uploading Images: The form data, including the uploaded files, is sent to the server with a POST request to the 'uploadallphenotypeimages' endpoint.

- Processing Images: If the images are successfully uploaded, a POST request is made to the '/processphenotype' endpoint to start processing the images. **processphenotype** endpoint is routed from the +page.server.js file located in the home folder to its corresponding script. (see Backend process handling section)

- Notification handling.

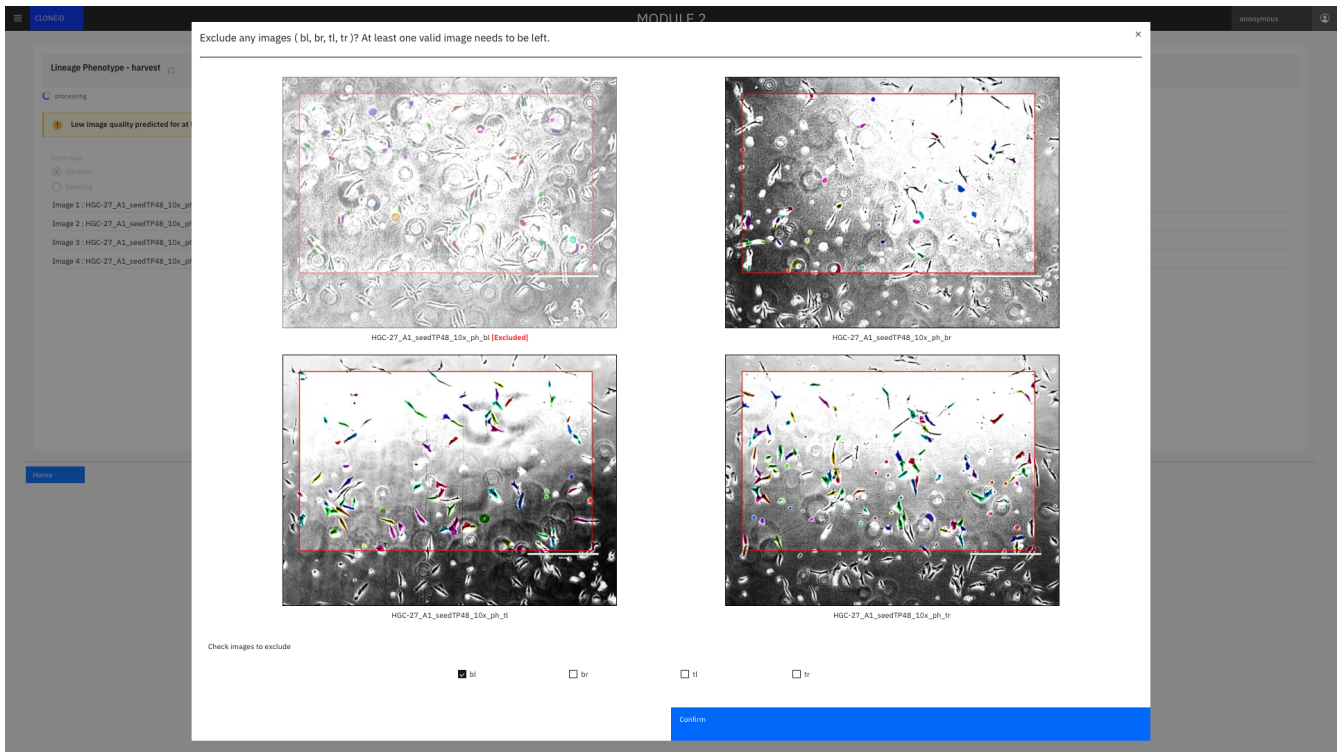


Figure 4. Lineage pane Notification handling

webportal/frontend/src/routes/(cloneId)/module2/home/components/LineagePaneWebWorker.svelte

```
...
if (!pushed) return;
if (!pushed.length) return;
for (const pevent of pushed) {
  const event = pevent.k;
  const content = pevent.v;
  notifk2 = content;
  if (!(x.evnt.AFIR && askforinputProps.show))
    rows.push({ id: rows.length, event: event, content: content });

  if (content) {
    const pipesplit = content.split('|');
    const columnsplit = content.split('::');

    switch (event) {
      case 'ABRT':
        break;

      case 'AFIQ':
```

```

const afiq = JSON.parse(pipesplit[1]);
const tmp_askforinputProps = {
  show: true,
  prompt: afiq.prompt[0],
  retry: afiq.retry_prompt[0],
  options: afiq.inputOptions,
  infomessage: afiq.infomessage.join('<br>'),
  infotype: afiq.inputType[0],
  txid: x.txid,
  s: images_combined_sha512
};
$loadingIndicatorToggle = false;
// console.log(afiq, tmp_askforinputProps);
askforinputProps = tmp_askforinputProps;
break;

case 'AFIR':
  if (!true) {
    const p1 = columnsplit;
    const p2 = p1[2].split('|');
    $loadingIndicatorToggle = false;

    const tmp_askforinputProps = {
      show: true,
      prompt: p1[1],
      retry: p1[3],
      options: p2,
      infomessage: p1[4],
      infotype: p1[5],
      txid: x.txid
    };

    askforinputProps = tmp_askforinputProps;
  }
  break;

case 'DONE':
  cellposeOnlyLayout();
  return;

case 'INFO':
  break;

case 'TIMT':
  ...
  break;

case 'JSON':
  ...
  break;

```

```

    case 'MESG':
        break;

    case 'MSQL':
        break;

    case 'NTIF':
        const status = pipesplit[0];
        const title = pipesplit[1];
        notif = { status: status, title: title };
        break;

    case 'PRNT':
        break;

    case 'QUIT':
        console.log('JobInfoWebWorker', event);
        $loadingIndicatorToggle = false;
        resetLayout();

        break;

    case 'RSLT':
        {
            const status = pipesplit[0];
            const title = pipesplit[1];
            if (status === 'IMGS') {
                const imgs = await saveImgInfo();
            }
        }
        break;

    case 'VARS':
        const kv = pipesplit;
        $GlobPVARs[kv[0]] = JSON.parse(kv[1]);
        break;

    case 'WRNG':
        notif = { status: pipesplit[0], title: pipesplit[1] };
        break;

    default:
        break;
}
}
}

```

- The `InputRequestWebWorker` component handle messages from backend processing jobs. It updates the UI, manages notifications, requests user input, and other actions based on the event data.

- ABRT, QUIT: Resets the layout and stops the loading indicator.
- AFIQ: Requests user input. It updates askforinputProps with the required prompt details and shows the input request UI.
- DONE: Job is complete. Switches to the cellpose-only layout.
- TIMT: Toggling the visibility of askforinputProps in case of a timeout awaiting user input.
- JSON: Handles JSON result data. Updates the segmentation information (seginfo) and saves it.
- NTIF,WRNG: Handles UI notifications by updating the notif object with status and title.
- VARS: Updates global variables with event data.

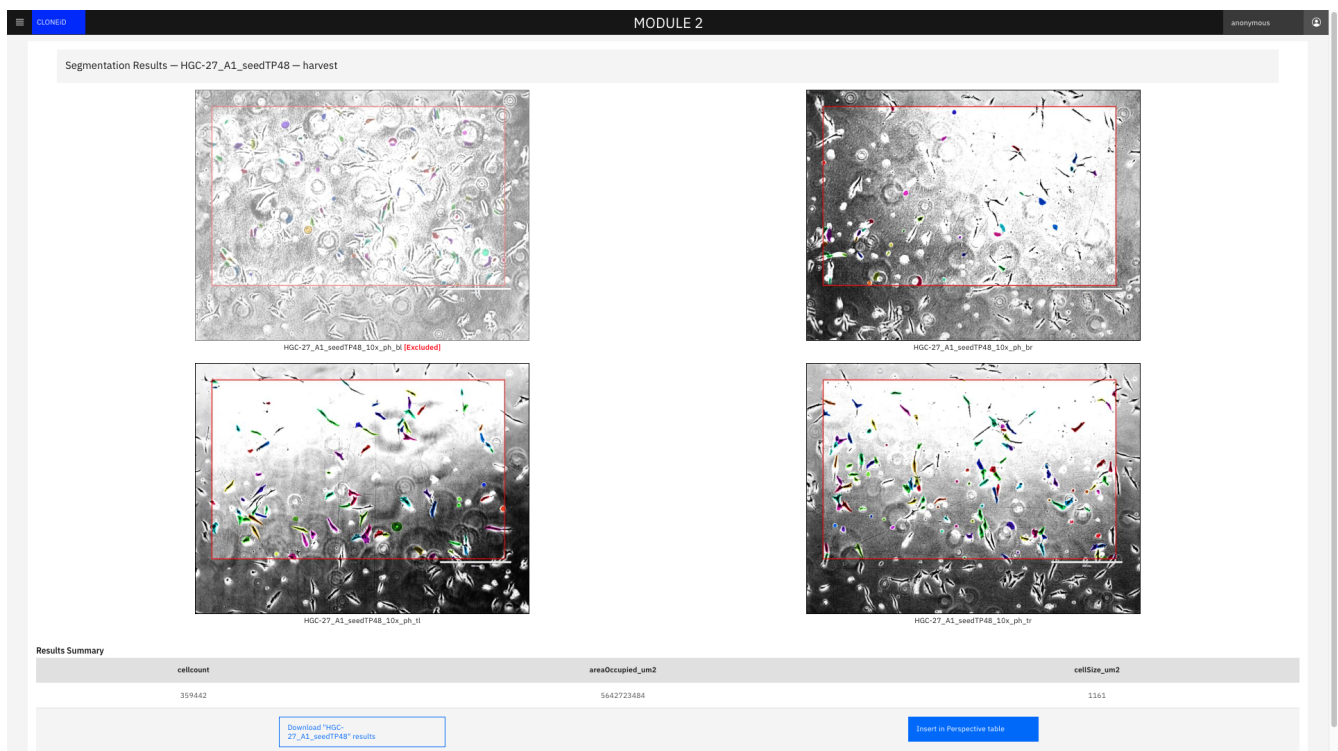


Figure 5. Lineage pane results

- Displaying Lineage pane results.

CellposeResults Component:

webportal/frontend/src/routes/(cloneId)/module2/home/components/CellposeResults.svelte

```
...
{#each [0, 1] as xy}
<Row
  style="
    margin:0px;
    padding:0px;
    background-color:rgb(200,200,200);
    outline-style:solid; outline-color:#FF0000FF; outline-width:0px;
    border-style:solid; border-color:#0000FFFF; border-width:0px;
    _display: flex; flex-direction: row; align-items: center;
```

```

    iheight:40vh;
    "
  >
  {#each [2 * xy, 2 * xy + 1] as index}
    {#if processedi[index]}
      <Column
        sm={1}
        md={4}
        lg={8}
        style="
padding:10px;
iheight:40vh;
idisplay: flex; flex-direction: column; align-items: center;
text-align:center;
"
      >
        <div style="height:100%;width:400px;text-align:center;padding:5px;">
          <img
            src={processedi[index].imgsrc}
            style="object-fit: contain;
              max-width: 80%;
              max-height: 80%;
              outline-style:solid; outline-color:#000000FF;
outline-width:2px;
              {processedi[index] && exclude &&
exclude.includes(processedi[index].suffix)
? 'opacity:0.50;'
: 'opacity:1.00;'}
"
            alt={processedi[index].short}
          />
        </div>

        {processedi[index].prefix}
        {#if processedi[index] && exclude &&
exclude.includes(processedi[index].suffix)}
          <strong><span style="color:red;">[Excluded]</span></strong>
        {/if}
      </Column>
    {/if}
  {/each}
</Row>
{/each}
...

```

- CellposeResults Component:

- After awaiting the resolution of `loadSeginfo(id)`, A grid iterates over two rows, each containing columns with images sourced from `processedi`.

Multi-Omics pane

- `webportal/frontend/src/routes/(cloneId)/module2/home/components/MultiOmicsPane.svelte`
 - Event handling and data processing for Multi-Omics pane.

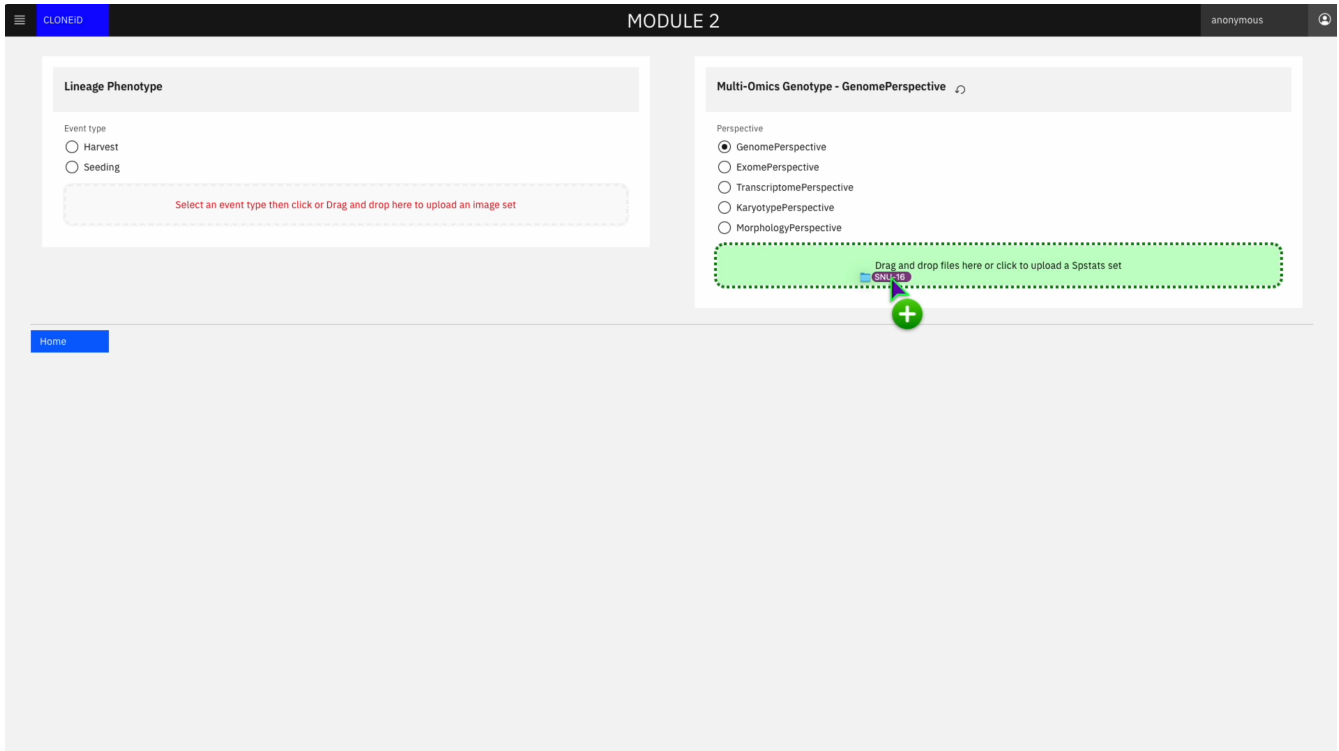


Figure 6. Multi-Omics pane Drag and Drop

- Drag and Drop handling.

Dropzone Component:

`webportal/frontend/src/routes/(cloneId)/module2/home/components/MultiOmicsPane.svelte`

```
...
<Dropzone
  on:drop={async (e) => {
    if (selected_genotype_perspective === '') {
      alert('No perspective selected');
      e.preventDefault();
      $loadingIndicatorToggle = false;
      return;
    }
    $loadingIndicatorToggle = true;
    const uploaded = await uploadFiles(e).catch((e) => {
      alert('CATCH Error uploading, please try again');
    });
    if (!uploaded) {
      alert('Error uploading, please try again');
      e.preventDefault();
      selected_genotype_perspective = '';
      $loadingIndicatorToggle = false;
    }
  }
}
```

```

        return;
    }
    if (uploaded && !uploaded.uploaded && uploaded.error) {
        alert(uploaded.error);
        e.preventDefault();
        selected_genotype_perspective = '';
        $loadingIndicatorToggle = false;
        return;
    }
    if (uploaded && !uploaded.uploaded) {
        alert('Error uploading, please try again');
        e.preventDefault();
        selected_genotype_perspective = '';
        $loadingIndicatorToggle = false;
        return;
    }
    start_spstats_upload = true;
    const dir = await SPSTATS(
        uploaded.timestamp,
        uploaded.uniqName,
        selected_genotype_perspective
    ).then(async (res) => {
        const result = await res.json();

        const data = decoderesponse(result);
        const r = JSON.parse(data.result);
        return r;
        const keys = data[0];
        return JSON.parse(data[1]);
    });
    if (Object.keys(dir.resultArray).length === 0) {
        alert(
            dir.file.toString() +
            ' does not exist in table Passaging. Sample needs to be recorded
in DB before adding omics-results to table Perspective.'
        );
        resetLayout();
        return;
    }
    ChartContainerSimplePieData = Object.keys(dir.resultArray)
        .sort()
        .map((x) => {
            return { name: `${x}`, value: dir.resultArray[x] };
        });
    let rowi = 1;
    tableRows = Object.keys(dir.resultArray)
        .sort()
        .map((x) => {
            return { id: `${rowi++}`, name: `${x}`, saved: dir.resultArray[x] };
        });
    preview_genotype = true;

```

```

        insideDrop_genotype = false;

        multiOmicsOnlyLayout();

        $loadingIndicatorToggle = false;
    }}
    on:click={async (e) => {
        if (selected_genotype_perspective === '') {
            e.preventDefault();
            e.stopImmediatePropagation();
            return;
        }
    }}
    containerStyles={insideDrop_genotype
        ? 'outline-style: dotted; outline-color: green;    outline-width: 4px;
display: flex; flex-direction: column; align-items: center; padding: 20px; border-
radius: 12px; background-color: #00ff0044;color: #111111; cursor:pointer; transition:
border 1.24s ease-in-out;'
        : selected_genotype_perspective
        ? 'outline-style: dotted; outline-color: blue;    outline-width: 2px;
display: flex; flex-direction: column; align-items: center; padding: 20px; border-
radius: 12px; background-color: #fbfbfb; color: #111111; cursor:copy; transition:
border 0.24s ease-in-out;'
        : 'outline-style: dotted; outline-color: #f0f0f0; outline-width: 2px;
display: flex; flex-direction: column; align-items: center; padding: 20px; border-
radius: 12px; background-color: #fbfbfb; color: #111111; cursor:not-allowed;
transition: border 0.24s ease-in-out;'}
    on:dragenter={() => {
        insideDrop_genotype = selected_genotype_perspective === '' ? false : true;
    }}
    on:dragleave={() => {
        insideDrop_genotype = false;
    }}
    multiple
    noClick={selected_genotype_perspective === '' ? true : false}
    disabled={false}
>
    <span style={selected_genotype_perspective == '' ? 'color:red;' : ''}>
        {selected_genotype_perspective === ''
            ? 'Select a perspective then click or Drag and drop here to upload a
Spstats set'
            : 'Drag and drop files here or click to upload a Spstats set'}
    </span>
</Dropzone>
...

```

- The Dropzone component is used to create an area where users can drag and drop files. It has multiple event listeners to handle the drag-and-drop functionality and to manage the styles of the dropzone based on user interactions and the state of the component.
- DropZone event listeners.

- on:drop: (event) - This event listener is triggered when a file or folder is dropped in the drop zone area.
 - Perspective Check: It first checks whether a genotype perspective has been selected. If not, it alerts the user and prevents the default behavior, disabling further actions.
 - File Upload and Error Handling:
 - The `uploadFiles` function is called to upload the dropped files. Multiple checks ensure that the uploaded files are valid and correctly uploaded. If any condition fails an alert message is displayed, and further operations are halted.
 - SPSTATS Function is triggered once files are successfully uploaded. Response is decoded `decoderesponse`.
 - Result Handling and Display: If data is valid, `ChartContainerSimplePieData` and `tableRows` are updated to display the processed Multi-Omics data.
- on:dragleave,dragenter
 - Updates `insideDrop_genotype` based on the drag state to change the visual appearance.
- Dropzone Styles:
 - Dynamic Style: The CSS styles of the Dropzone are dynamically adjusted based on `insideDrop_genotype` and `selected_genotype_perspectives` to provide visual feedback to the user.

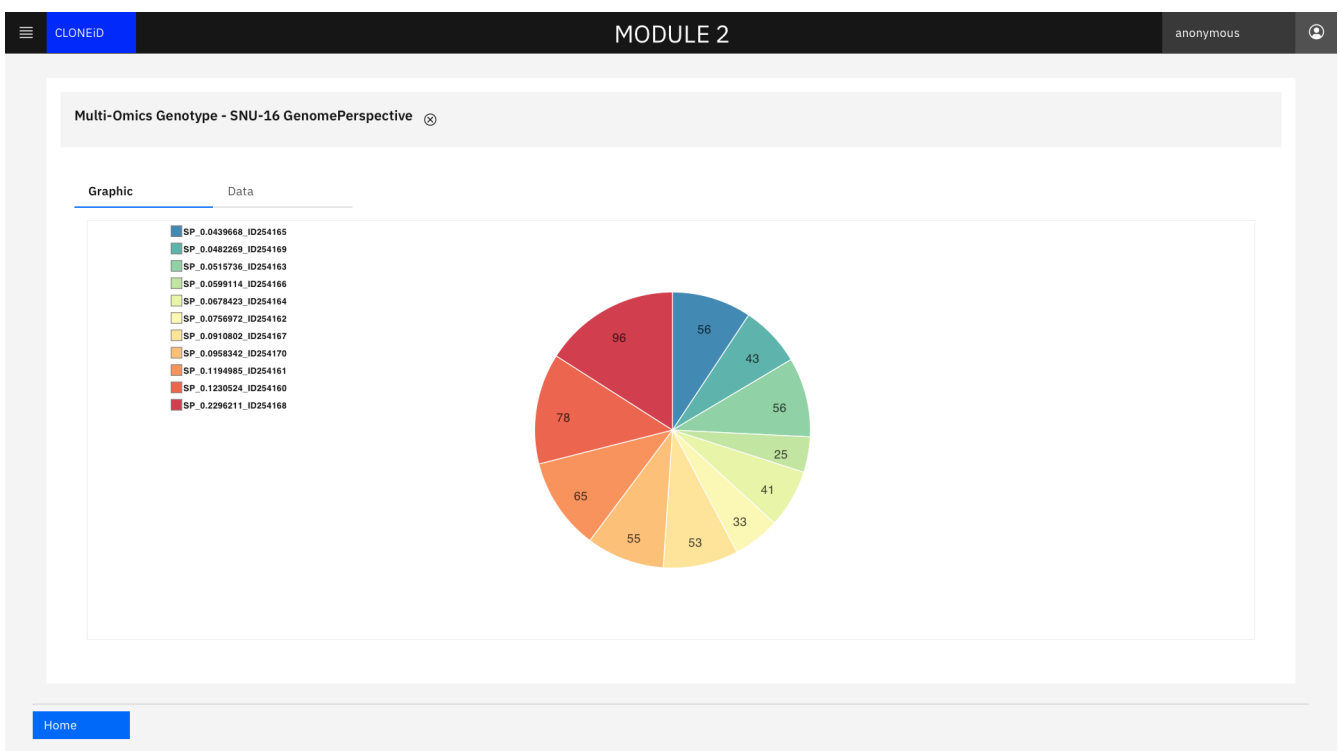


Figure 7. Multi-Omics results

- Displaying Multi-Omics results.

Pie Component:

```
...
<TabContent>
<!-- PIE -->
<ChartContainerSimplePie
  title={'clonal representation'}
  data={ChartContainerSimplePieData}
  completion={() => {}}
  size="md"
  HASH={` ${1} - ${22} - P`}
  _data={[]}
/>
</TabContent>
...
```

- Pie Component: A `<TabContent>` container holds a pie chart, using `ChartContainerSimplePieData`.
 - Component properties:
 - title: Sets the title of the pie chart to "clonal representation."
 - data: uses `ChartContainerSimplePieData`, which contains the data processed.
 - completion: Callback that executes when the chart has finished rendering or processing.
 - size: Specifies the size of the chart, set to "md" (medium) in this case.
 - HASH: A string used as a unique identifier for the chart.

Module 2, Backend 1

The Backend 1 code implements the necessary functions to handle the interactions with the Multi-Omics frontend pane.

Main functions providing backend 1 functionalities

Code includes file `webportal/bakend1/SpStats.java` and a patch to `Clone.java` in `webportal/data/gitdiff/cloneid.patch`

Class *SpStats* manages the processing of the SPSTATS folder passed by the frontend Multi-Omics pane using an endpoint (See frontend section).

webportal/data/gitdiff/cloneid.patch

```
public void backend_message(long txid, String Level, String k, String v) {
    DateFormat dbTimeFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String tx2 = dbTimeFormat.format(new Date(0L));

    String[] levels = { "ABRT", "AFIQ", "AFIR", "DONE", "INFO", "JSON", "MESG",
"MSQL", "NTIF", "PRNT", "QUIT", "RSLT", "VARS", "WRNG" };
    boolean levelMatched = Arrays.asList(levels).contains(Level);

    if (levelMatched) {
        System.out.println "[" + Level + "]: " + txid + " : " + tx2 + " : " + k + " | " +
v + " : " + txid);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        if ("ABRT".equals(Level)) {
            try {
                Thread.sleep(10000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.exit(0);
        }
    }
}

public void save2DBX(long transactionId) throws Exception{

    String tableName=CLONEID.getTableNameForClass(this.getClass().getSimpleName());
    informUserX(transactionId);
    for(Clone c: children){
        c.save2DBX(transactionId);
    }
}
```

```

CLONEID cloneid= new CLONEID();
cloneid.connect();

Integer existingid=isinDB(tableName, cloneid);
Map<String,String> map = this.getDBformattedAttributesAsMap();
if(existingid==null){
    String addstmt = "INSERT INTO " + tableName+ "(";
    String valstmt=" VALUES(";
    for(Entry<String, String> kv : map.entrySet()){
        addstmt+=""+kv.getKey()+", ";
        valstmt+=""+kv.getValue()+", ";
    }
    addstmt+="transactionId )";
    valstmt+=""+transactionId+" )";
    addstmt+=valstmt;
    PreparedStatement prest= cloneid.getConnection().prepareStatement(addstmt,
Statement.RETURN_GENERATED_KEYS);
    prest.executeUpdate();
    ResultSet rs = prest.getGeneratedKeys();
    rs.next();
    this.cloneID = rs.getInt(1);

    SerializeProfile_MySQL serp=new
SerializeProfile_MySQL(this.cloneID,this.getClass().getSimpleName());
    serp.writeProfile2DB(cloneid.getConnection(), profile);
}else{
    this.cloneID=existingid;
    this.redundant=true;
}

if(children.size()>0){
    for(Clone c: children){
        String updateSTmt =
        "UPDATE " +
        tableName +
        " SET parent=" + this.cloneID +
        ", rootID="+this.getRoot().cloneID +
        " WHERE cloneID=" + c.cloneID +
        ";";
        cloneid.getStatement().executeUpdate(updateSTmt);
    }
    String updateSTmt = "UPDATE " + tableName + " SET hasChildren=true WHERE
cloneID=" + cloneID;
    cloneid.getStatement().executeUpdate(updateSTmt);
}

cloneid.close();

if(this.parent==null && countRedundant()>0){
    backend_message(transactionId, "NTIF", "RDD", countRedundant()+"" );
    backend_message(transactionId, "JSON", "RDD",

```

```

        "[ \\"" + countRedundant() + "\" ]"
    );
}

}

private void informUserX(long transactionId) {
    Map<Double,Integer> spfreq=Helper.count(getChildrensSizes(),0.001);
    if(spfreq.size(>0){
        backend_message(transactionId, "NTIF", "CSS", "");
        backend_message(transactionId, "JSON", "CSS", "[\"NULL\"]");
        for(Entry<Double, Integer> e : spfreq.entrySet()){
            backend_message(transactionId, "NTIF", "SFS", e.getValue() + "|" +
String.format("%.7f", e.getKey()));
            backend_message(transactionId, "JSON", "SFS",
                "[ \\"" + e.getValue() + "\", \\"" + String.format("%.7f", e.getKey()) +
"\\" ]"
                );
        }
    }
}
}

```

- **backend_message** method sends messages back to the frontend through a webworker.
- **save2DBX** and **informUserX** are updates to their counterpart that send messages back to the frontend through **backend_message**.

webportal/bakend1/SpStats.java

```

public class SpStats {
    public static Map<String, Number> spstats(
        String index,
        String timestamp,
        String userhash,
        String file,
        String destination,
        String perspective
    ) throws Exception {
        getYmlConf();
        String spstatsPath = destination + userhash + "/" + timestamp + "/";
        CLogger.log().info(String.format("%s", spstatsPath));

        String spfName0 = file + ".sps.cbs";
        String spfPath0 = spstatsPath + spfName0;
        File spfFile0 = new File(spfPath0);
        final Map<String, Number> resultMap = new LinkedHashMap<>();
        System.out.println("==== file ==== " + spfFile0);
        CLogger.log().info(String.format("%s", perspective));
        long transactionId = (new Date()).getTime();
        try {
            spstatsForPerspective(userhash, file, spfFile0, transactionId,

```

```

spstatsPath, resultMap, perspective);
    } catch (java.sql.SQLException e) {
        System.out.println("==== Exce ==== " + e);
    }
    return resultMap;
}

private static void spstatsForPerspective(String userhash, String file, File
spfFile0, long transactionId, String spstatsPath, Map<String, Number> resultMap,
String perspective) throws Exception {

    Perspective p0 = getPerspective(spfFile0, "CN_Estimate", perspective);
    Map<Double, Integer> results;
    results = p0.save2DBX(transactionId);
    System.out.println("==== p ==== " + p0);

    for (var sp : p0.getChildrenSizes()) {
        String s1 = String.valueOf(sp);
        if (s1.length() > 9) {
            s1 = s1.substring(0, 8);
        }
        String n1 = String.format("%s.%s.sps.cbs", file, s1);
        File f1 = new File(spstatsPath + n1);

        Perspective p1 = getPerspective(f1, "SP_" + sp, perspective);
        results = p1.save2DBX(transactionId);

        int sptotal = 0;
        for (Map.Entry<Double, Integer> kv : results.entrySet()) {
            sptotal += kv.getValue();
        }
        resultMap.put(p1.toString(), sptotal);

        System.out.println("==== transactionid ++++" + transactionId + "==== file
++++" + n1 + ":: perspective ++++" + p1 + ":: sptotal ++++" + sptotal);
    }
    removeLaterIfUncertifiedUser(2 * 60 * 1000, userhash, transactionId);
}

private static Perspective getPerspective(File file, String rootName, String
perspective) throws Exception {
    Perspective p0 = switch (perspective) {
        case "GenomePerspective" -> new GenomePerspective(file, rootName);
        case "TranscriptomePerspective" -> new TranscriptomePerspective(file,
rootName);
        case "KaryotypePerspective" -> new KaryotypePerspective(file, rootName);
        case "ExomePerspective" -> new ExomePerspective(file, rootName);
        case "MorphologyPerspective" -> new MorphologyPerspective(file, rootName);
        default -> null;
    };
}

```

```

    };
    return p0;
}

private static boolean IsCertifiedUser(String userhash) {
    String uncertified = "anonymous";
    boolean u1 = userhash.equals(uncertified);
    String usha512 = SHA512.sha512(uncertified);
    boolean u2 = userhash.equals(usha512);
    return !(u1 || u2);
}

private static void removeLaterIfUncertifiedUser(long timelapse, String userhash,
long transactionId) {
    if (!IsCertifiedUser(userhash)) dbRowsRemoval(timelapse, transactionId);
}

private static void dbRowsRemoval(long delay, long transactionId) {
    TimerTask task = new TimerTask() {
        public void run() {
            try {
                removeTransaction(transactionId);
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
            System.out.println(Thread.currentThread().getName());
            System.out.println("Removal Task performed on: " + new Date());
        }
    };
    Timer timer = new Timer("Perspective Removal " + transactionId, false);
    Date later = new Date(System.currentTimeMillis() + delay);
    System.out.println("Removal of transaction " + transactionId + " Task will
occur on: " + later.toString());
    timer.schedule(task, delay);
}

public static void removeTransaction(long transactionId) throws Exception {
    String updateSTmt = "DELETE FROM Perspective where transactionId = " +
transactionId + " ";
    dbIO.update(updateSTmt);
}

public static Map<String, Number> cleanspstats(
    long timestamp,
    long delta,
    String transaction
) throws Exception {
    getYmlConf();
    final Map<String, Number> resultMap = new LinkedHashMap<>();
    System.out.println("==== timestamp ==== " + timestamp);
}

```

```
        resultMap.put("timestamp", timestamp);  
        return resultMap;  
    }  
  
}  
...
```

- The `spstats` method orchestrates the processing of the spstats folder.
- The `spstatsForPerspective` method initializes a perspective, saves data to the database, processes child perspectives, calculates totals, and stores them in a result map. It schedules cleanup tasks for uncertified users.

Module 2 - Backend 2

The Backend 2 code implements the necessary functions to handle the interactions with the Lineage frontend pane.

Main functions providing Backend 2 functionalities

Code includes file `webportal/bakend2/node/dw.mjs` wich handles queue processing, `webportal/bakend2/node/server` folder contains helpers for queue processing and R scripts located in `webportal/bakend2/R`

- `processPayloadDir` orchestrates the processing of queued jobs.
- `webportal/bakend2/node/server` contains helpers for processing the payload directory.

webportal/bakend2/node/dw.mjs

```
async function processPayloadDir(dir, app, queue) {

  const qdir = normalisePathForQueue(app, queue);
  if (!existsSync(qdir)) {
    throw qdir;
  }

  let joblist = [];
  let totaljobs = 0;
  let donejobs = 0;

  joblist = enqueuewaitingdir(qdir);
  totaljobs = joblist.length;
  logmessage(["ADDING WATCH DIR:WAITING", qdir, "A =", totaljobs, joblist]);

  watch(qdir, async (e, f) => {
    if (f.endsWith('.json') && existsSync(qdir + '/' + f)) {
      if (!joblist.includes(f)) {
        joblist.push(f);
        logmessage(["QUEUING", app, queue, f, e, totaljobs++,
joblist.length]);
      }
    }
  });
  while (true) {
    donejobs += await processQueue(qdir, joblist, app, queue);
    logmessage(["DONE/TOTAL:", donejobs, totaljobs]);
    await new Promise((res) => setTimeout(res, 5000));
  }
};
```

- R scripts `webportal/bakend2/R/IVU9.R` and `webportal/bakend2/R/T2.R` are spawn from `server.js` code, respectively `webportal/frontend/src/routes/(cloneId)/module2/home/server/processphenotype.js` and `webportal/frontend/src/routes/api/seginfo/+server.js`. Both scripts use functions `handleUserInput` and `backend_message` to interact with the queue manager.

`handleUserInput` and `backend_message`

```
handleUserInput <- function(inputOptions, prompt, retry_prompt, infomessage='NONE',
inputType="R") {
  functionName<-as.character(match.call()[[1]])
  while (TRUE)
  {
    user_input <- NULL
    be_verbose <- FALSE
    timeout <- 900
    quit_on_timeout <- TRUE
    quit_on_noenv <- TRUE
    afia_filename <- "afinputa.txt"
    afiq_filename <- "afir.txt"

    # Check if session is interactive
    if (!inbackend_session) {
      # Print inputOptions
      if (be_verbose) {
        backend_message("INFO", functionName,"Interactive session")
      }
      if(infomessage!='NONE'){
        backend_message("INFO", functionName, infomessage)
      }
      for (i in seq_along(inputOptions)) {
        backend_message("INFO", functionName, paste(i, ":", inputOptions[i]))
      }
      # Get user input
      user_input <- readline(prompt)
      if (be_verbose) {
        backend_message("INFO", functionName, paste("Terminal user_input: ",
user_input))
      }
    } else {
      # Read input from file
      if (be_verbose) {
        backend_message("INFO", functionName, "Non Interactive session")
      }
      txid <- inbackend_session_TXID
      if (is.na(txid) || txid == '') {
        # txid = "TXID_1708137304985"
        if (be_verbose) {
          backend_message("INFO", functionName, "NO TXID IN ENV:")
        }
      }
    }
  }
}
```

```

    }
    if (quit_on_noenv) {
      q(save = "no",
        status = -1,
        runLast = FALSE)
    }
  }
  txid_dir <- getEnvAFIDIR()
  if (is.na(txid_dir) || txid_dir == '') {
    # txid_dir = "/opt/lake/data/cloneid/module02/data/txdir"
    if (be_verbose) {
      backend_message("INFO", functionName, "NO TXDIR IN ENV:")
    }
    if (quit_on_noenv) {
      q(save = "no",
        status = -1,
        runLast = FALSE)
    }
  }
}

Sys.sleep(5);
cat(paste(
  c(
    "[AFIR]",
    txid,
    gsub(":", "", prompt),
    paste(gsub(":", "", inputOptions), collapse = "|"),
    gsub(":", "", retry_prompt),
    gsub(":", "", infomessage),
    inputType,
    txid
  ),
  collapse = "::"
), "\n")

afiq=c()
afiq$inputType = inputType
afiq$prompt = prompt
afiq$inputOptions = inputOptions
afiq$retry_prompt = retry_prompt
afiq$infomessage = infomessage

backend_message("AFIQ", 'afiq', jsonlite::toJSON(afiq))

if (be_verbose) {
  backend_message("INFO", functionName, paste(c(txid_dir, afia_filename),
collapse = "::"))
}
afia_filepath <- paste(c(txid_dir, afia_filename), collapse = "/")
afiq_filepath <- paste(c(txid_dir, afiq_filename), collapse = "/")
if (be_verbose) {

```

```

        backend_message("INFO", functionName, paste(c("FILE_WITH_INPUT",
afia_filepath), collapse = "::"))
        backend_message("INFO", functionName, paste(c("FILE_WITH_INPUT",
afia_filepath), collapse = "::"))
    }
    start_wait <- Sys.time()
    # until file avail or timeout
    while (TRUE) {
        if (file.exists(afia_filepath)) {
            user_input <- readLines(afia_filepath, n = 1)
            if (length(user_input) > 0) {
                if (be_verbose) {
                    backend_message("INFO", functionName, paste(c( "File User Input",
user_input), collapse = "="))
                }

                tstp = nowSeconds()

                tryCatch(file.rename(afia_filepath,paste0(afia_filepath, '_', tstp,
'.txt')),
                        warning = function(e){ backend_message("WRNG", functionName,
e$message) },
                        error = function(e) { backend_message("ERRR", functionName,
e$message) },
                        finally = {}
                )

                if (file.exists(afiq_filepath))
                    tryCatch(file.rename(afiq_filepath, paste0(afiq_filepath, '_', tstp,
'.txt')),
                            warning = function(e){ backend_message("WRNG", functionName,
e$message) },
                            error = function(e) { backend_message("ERRR", functionName,
e$message) },
                            finally = {}
                    )

                break
            }
        }
        # quit on timeout
        if (difftime(Sys.time(), start_wait, units = "secs") > timeout) {
            if (be_verbose) {
                backend_message("INFO", functionName, "Timeout error: File not found")
            }
            if (quit_on_timeout) {
                q(save = "no", status = -1, runLast = FALSE)
            }
        }
        if (be_verbose) {
            backend_message("INFO", functionName, paste(c("Waiting for", afia_filepath),

```

```

collapse = " ")
    }
    Sys.sleep(1)
  }
  if (be_verbose) {
    backend_message("INFO", functionName, paste(c("File user_input: ",
user_input), collapse = " "))
  }
}

# Bypass when starts with '#'
if (startsWith(user_input, '#')) {
  final_input = gsub("#", "", user_input)
  backend_message("INFO", functionName, paste(c("Final File User Input",
final_input), collapse = "="))
  return(final_input)
}

suppressWarnings(choice <- as.numeric(user_input))

if (be_verbose) {
  backend_message("INFO", functionName, paste(c("choice: ", choice), collapse = "
"))
}

# If the input is a number and within the range of inputOptions
if (!is.na(choice) && choice >= 1 && choice <= length(inputOptions)) {
  return(choice)
} else {
  # Check if the input matches any option
  matched_option <- which(tolower(inputOptions) == tolower(user_input))
  if (length(matched_option) > 0) {
    return(matched_option)
  } else {
    backend_message("INFO", functionName, retry_prompt)
  }
}
}
}

backend_message <- function(Level, k, v){
  if ((is.null(Level) || length(Level)==0 )){
    cat("NO Level", "\n")
    q();
  }
  functionName<-as.character(match.call()[[1]])
  if(inbackend_session){
    txid <- inbackend_session_TXID
    if ((is.null(txid) || length(txid)==0 )){
      cat("NO TXID", "\n")
      q();
    }
  }
}

```

```

}
# tx1 = Sys.time()
tx2 <- dbTimeFormat(Sys.time())

if (Level=="ABRT" || Level=="AFIQ" || Level=="AFIR" || Level=="DONE" ||
Level=="INFO" || Level=="JSON" || Level=="MSG" || Level=="MSQL" || Level=="NTIF" ||
Level=="PRNT" || Level=="QUIT" || Level=="RSLT" || Level=="VARS" || Level=="WRNG"){
  cat(paste(
    c(
      paste0('[',Level,']'),
      as.character(txid),
      as.character(tx2),
      paste(c(k, v), collapse = "|"),
      as.character(txid)
    ),
    collapse = "::"
  ), "\n")
  if (Level=="ABRT"){
    Sys.sleep(10)
    q()
  }
  Sys.sleep(0.2)
}else{
  print(Level)
  Sys.sleep(100)
}
}else{
  if (!(is.null(v) || length(v)==0)){
    if(Level=="INFO"){
      cat(paste(c(v,"\n")))
      return
    }
    if(Level=="PRNT"){
      cat(paste(c(v,"\n")))
      return
    }
    if(Level=="NTIF"){
      cat(paste(c(v,"\n")))
      return
    }
    if(Level=="WRNG"){
      warning(v, immediate=T)
      return
    }
  }
}
}
}

```

Installation

- Module 2 is build using docker compose.yml file located at webportal/bakend1/docker-compose.yml
 - frontend will be accessible through port 4172 in your browser

Updating the database information

```
# replace following values in .env if needed
SQLHOST = database.XXXXXXXXXXX.us-east-1.rds.amazonaws.com
SQLPORT = 3306
SQLSCHM = CLONEID
```

Package information

Description

- Project: cloneid-module2
- Title: Model-Based Systems for tracing and steering clonal dynamics / Home page to view database content
- Version: 0.1
- Copyright: (c) 2023 MOFFITT
- Date: 2024-07-01
- Author: Daniel Hannaby
- License: MIT

Dependencies

```
@babel/core @carbon/colors @carbon/type @ibm/plex @observablehq/plot
@observablehq/runtime @sveltejs/adaptor-auto @sveltejs/adaptor-node @sveltejs/kit
DessimozLab/phylo-io FileReader autoprefixer carbon-components-svelte carbon-icons-
svelte carbon-pictograms-svelte carbon-preprocess-svelte classnames constructs d3 d3-
array d3-cloud d3-delaunay d3-fetch d3-interpolate d3-sankey d3-scale d3-scale-
chromatic d3-time-format eslint eslint-config-prettier eslint-plugin-svelte fs-extra
glyphicons-halflings js-sha512 just-compare just-throttle mdsvex mysql2 os pnpm
postcss postcss-load-config prettier prettier-plugin-svelte promises sass seedrandom
sha-1 svelte svelte-canvas svelte-check svelte-heros-v2 svelte-meta-tags svelte-
preprocess typescript umap-js unipept-visualizations vite vitest
```


License information

MIT License

Copyright (c) 2023 MOFFITT

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.