Scancode workbench improvements

Google Summer of Code 2022 Project Proposal - Large size project

by Omkar Phansopkar

Idea

Refactor workbench to a React + Typescript implementation and improve various sections of the application including Table view, file uploads, data sync across sections, etc

Project

- Upgrade dependencies to the latest suitable versions

  It's ironic how workbench shows info about packages in such detail, but itself has older package versions ;)
  I'll try to update all packages to the latest working (stable) versions

  Some old packages: chai, electron-rebuild, eslint, globby, grunt, etc
  Exception:
  Electron (v 13.0.0)
  Later versions of electron break our app, due to native modules like sqlite3 not being supported in the Renderer Process.
  More info:
  https://github.com/nexB/scancode-workbench/issues/508 &
  https://github.com/electron/electron/issues/18397

- Refactor workbench (Plain Js -> Typescript + React)

  Workbench currently uses Vanilla JS for everything, Shifting to a Typescript + React implementation would improve the developer experience. Mentioned in issue #518 earlier.

  Why Typescript ?
  - Typescript implementation would make code predictable and prevent errors at compile time
  - Improves IDE auto-completion, thereby enhancing the Developer experience.
  - Robust codebase
    Moreover, Electron readily supports Typescript

Why React ?

- Separate UI and logical parts of the application effectively
- Modularize the codebase into various components to improve readability.
  Large files like renderer.js can be split into several components
- Routes can be used to represent various sections allowing access control and redirection to and from selected routes based on the app state.
- Components can automagically update themselves via a global context and local state removing the need to manually take care of dom updates and manipulation
  For example,
  setColumnFilter() is as follows:

```
setColumnFilter(columnName, value) {
  // Get the ScanData table column and make sure it's visible
  const column = this.dataTable().column(`${columnName}:name`);
  column.visible(true);

  // Get the column's filter select box
  const select = $(`select#scandata-${columnName}`);
  select.empty().append(`<option value="">All</option>`);

  // Add the chart value options and select it.
  if (value === NO_VALUE_DETECTED) {
    select.append(`<option value="${value}">No Value Detected</option>`);
  } else {
    select.append(`<option value="${value}">${value}</option>`);
  }
  select.val(value).change();
}

In html we have a select element somewhere as :
<select id=”scandata-columnID”></select>
```

Here, we are doing the following:

- Grab the column element and set its visibility to true
- Grab the select element and add an option child to it
- The previous step, but with conditional stuff

React approach will look like this: (Ignore congested formatting, did that for keeping proposal concise)

```
const TableComponent = () => {
  .......
  // To change column data, just change the state here, dom will update itself åå
 const [cols, setCols] = useState<Col[]>([ Col1 ]);
  function makeCol1Visible(){
   setCols([ { ...Col1, visible: true} ]);
 }

  // To add new option to select, just update this filterOptions state
 // and the dom will automagically take newest state values
 const [filterOptions, setFilterOptions] = useState<string[]>([]);

 function addFilterOption(newOption: string){
   setFilterOptions([...filterOptions, newOption]);
 }
 ......
  return <DataTableComponent>
     { cols.map(col => (
         <ColumnComponent visible={ col.visible }>
           { col.header }
           Col filter options:
           <select>
            { filterOptions.map(filterOption => (
                <option value={filterOption}>  { filterOption }  </option>
              )) }
           </select>
         </ColumnComponent>
      )) }
   </DataTableComponent>
}
```

Because of react, our task comes down to just updating state using the ***setState()*** function provided by react state and dom updates are taken care of by react.

- Implement a React Context to seamlessly update UI across the application

  Similar to react state, Instead of manually telling each component to update the UI, we can use a react context with global state like this:

```
{
  importStatus: STATUS.NONE,  // or STATUS.IMPORTING, STATUS.IMPORTED
  data: { ... data }          // or null
}
```

Any descendant component can consume this status and data at any depth inside the context, and automatically update itself.

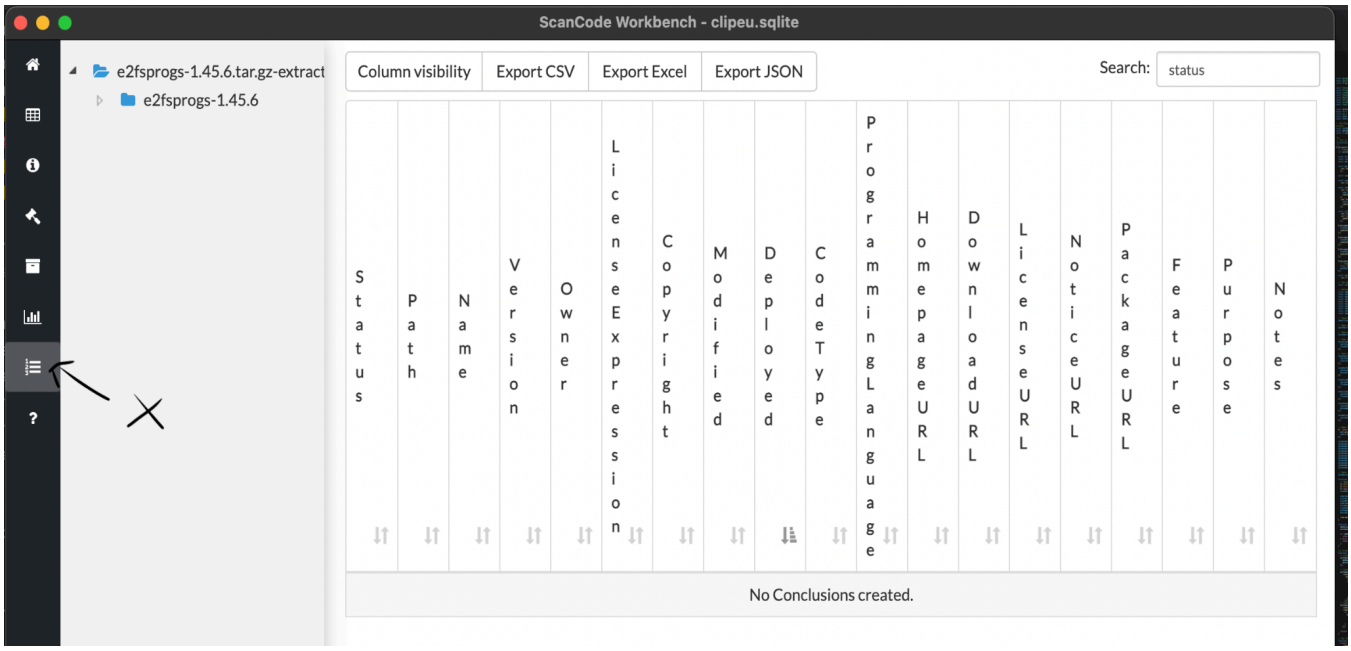
For example,
In this hierarchical component structure, even though we create context in the app component,
The context provider will make its latest value available to all the descendants component 1, component 2, 3, child1, child2, etc. So, we'll have to just update once and consume everywhere!

```
export const App = () => {
  const ImportDataContext = createContext({ importStatus: "IMPORTING", data: { something } })

  return (
    <ImportDataContext.Provider>
      <component1></component1>
      <component2></component2>
      <component3>
        <child1>
          <child2>
          </child2>
        </child1>
      </component3>
      <component4></component4>
    </importDataContext.Provider>
  )
}
```

- Remove the conclusions module

  Remove conclusions screen and nav item

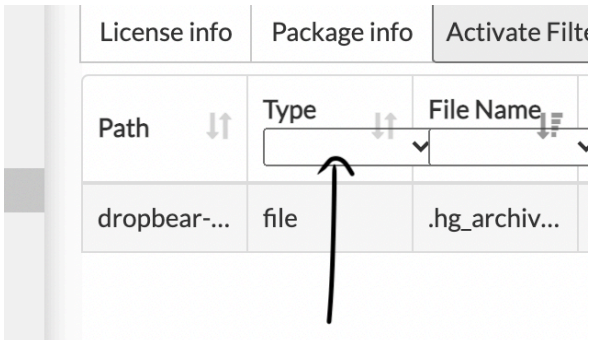


- Improve table view sort and filter

  If refactoring to React is approved, we can either use vanilla datatables with some more config or use its React implementation - [react-data-table-component]

  Propose and discuss alternative libraries for tables to improve UX.
  Suggested libraries:
  - AG-grid
    JS / TS implementation - https://www.npmjs.com/package/@ag-grid-community/core
    React implementation - https://www.npmjs.com/package/ag-grid-react

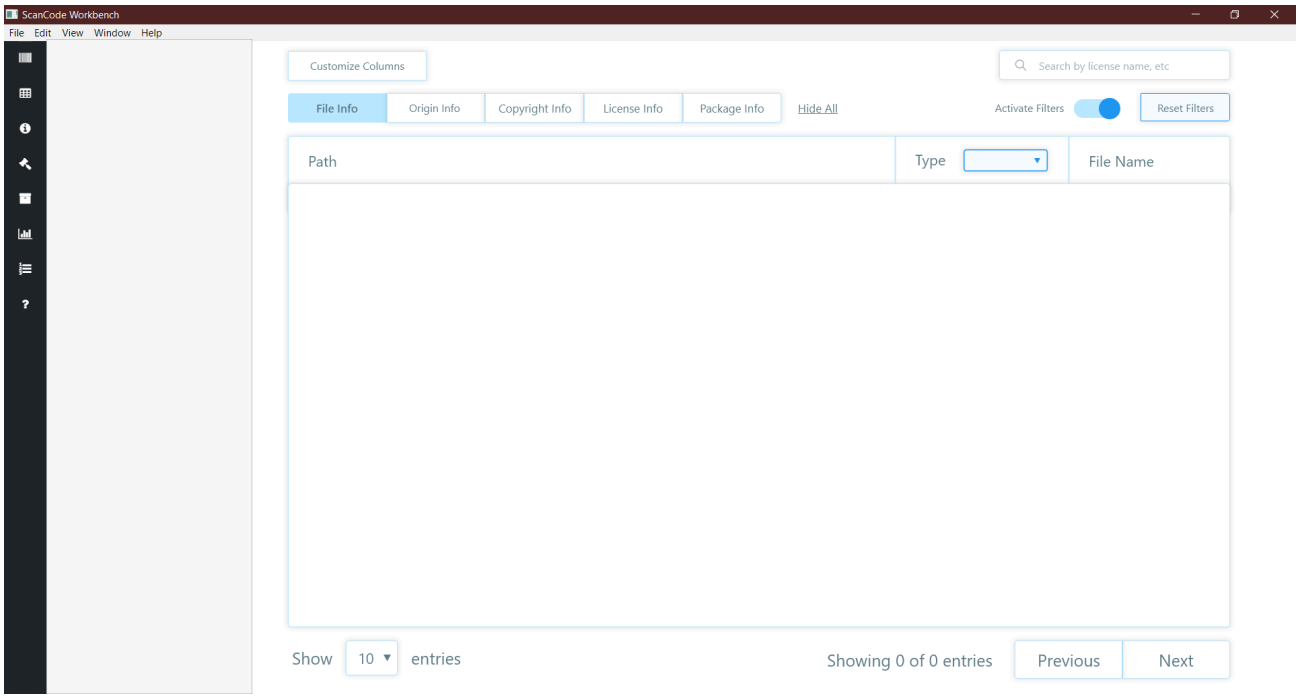
  Work on bugs while sorting and filtering table view columns.
  Eg. Filter options not populating here, Clicking on this doesn't do anything except show processing for a few milliseconds



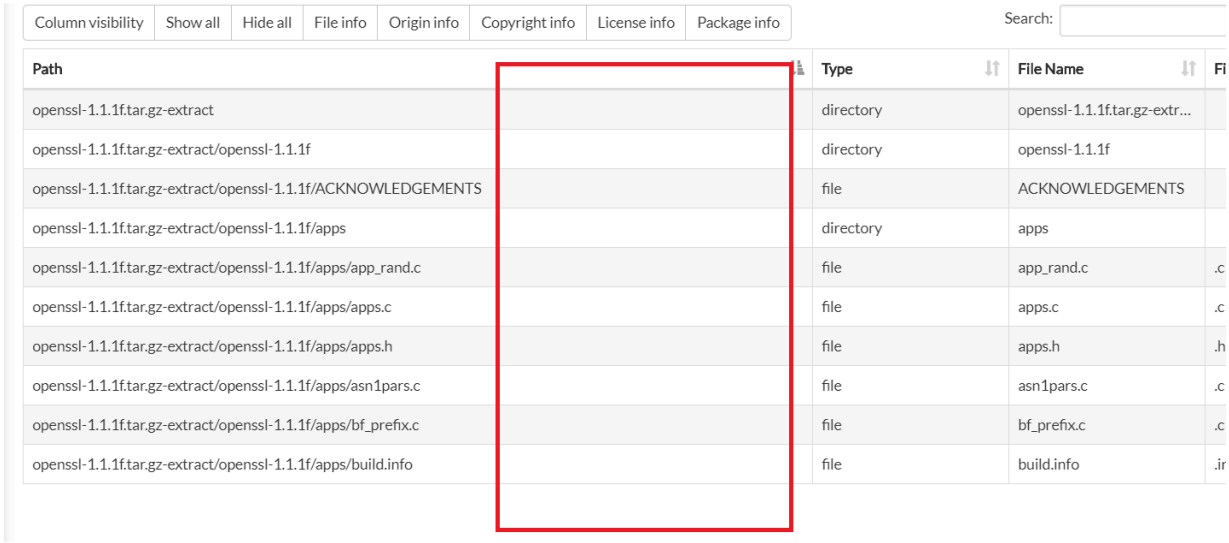
- UI improvements

1. **Table View**

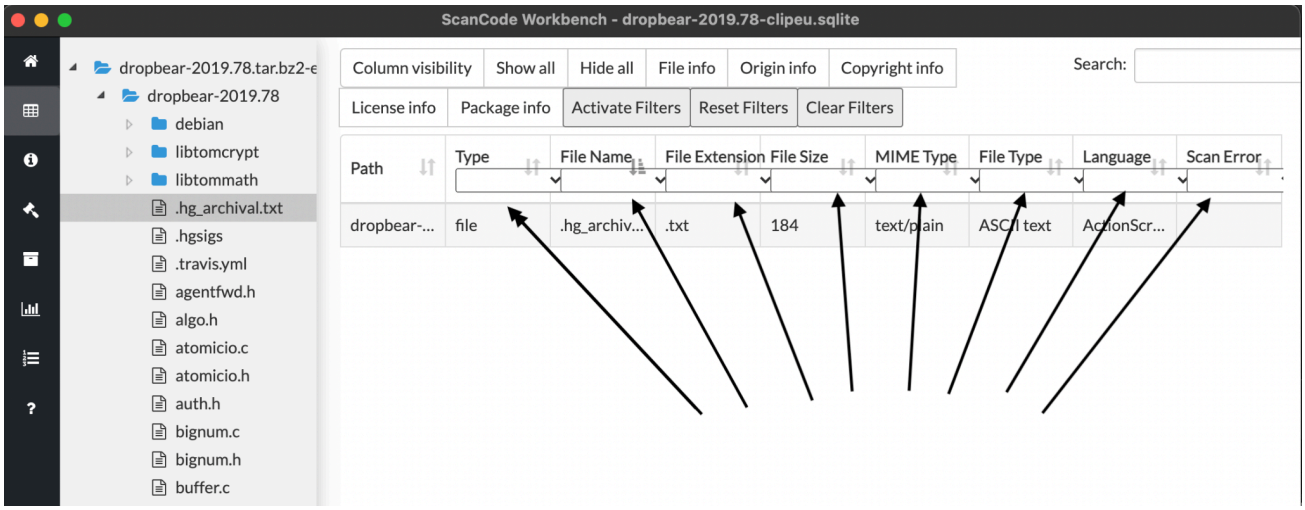
   Implement new designs for the Table view
   Example:
   Design credits: https://github.com/nitin10s
   Mentioned in Issue #451



   Fix inappropriate spacing at several parts
   Eg. Remove Extra space here

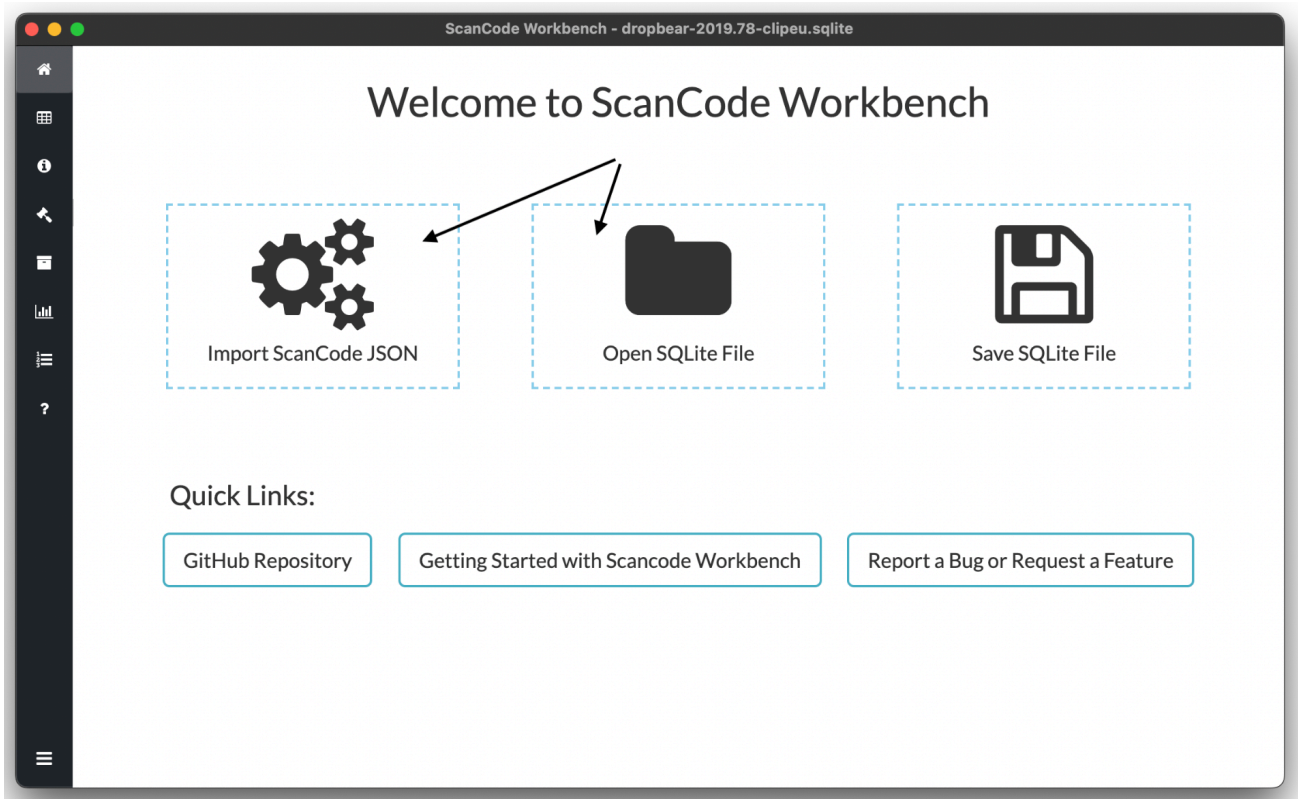


Fix Congested filters here



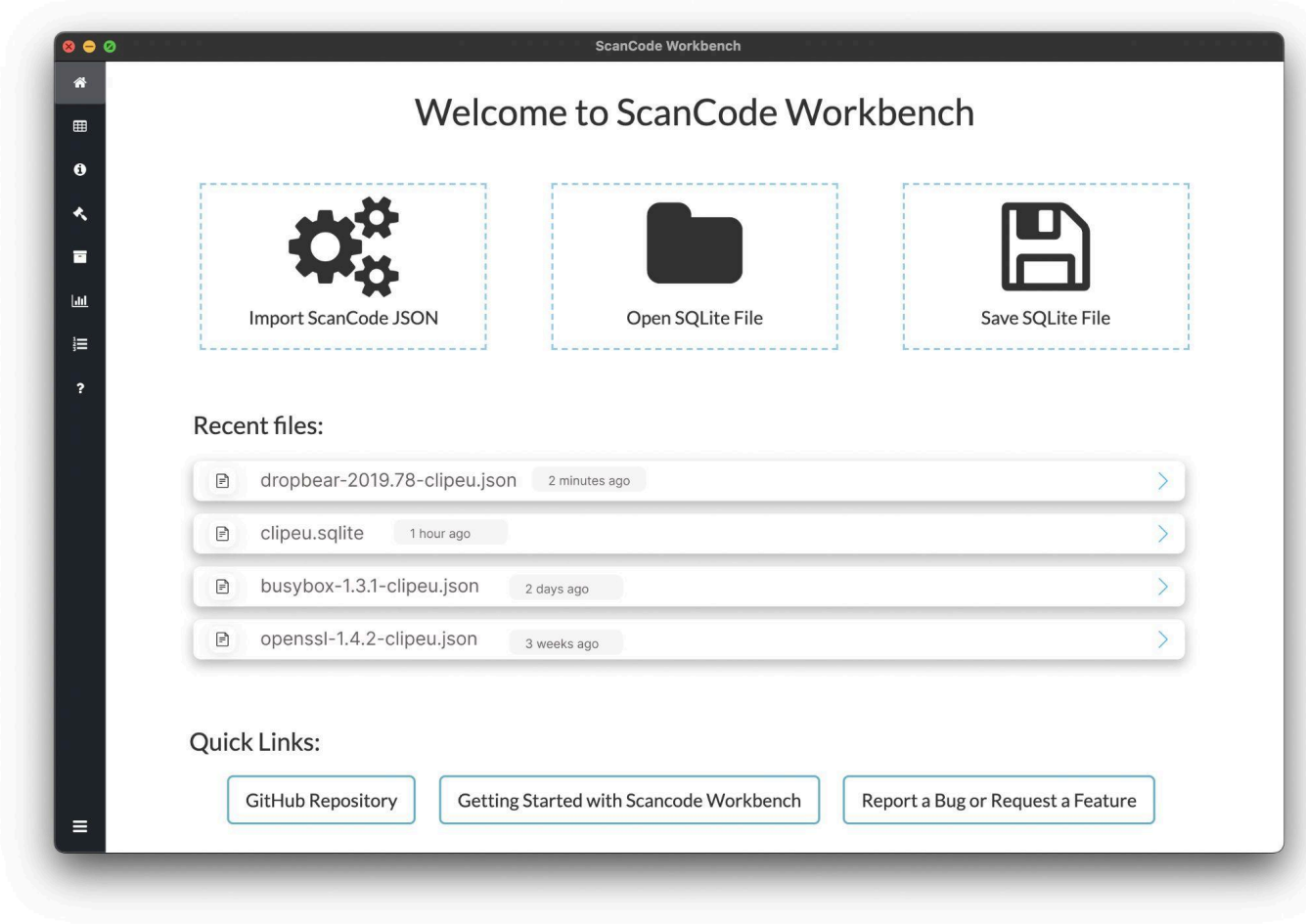
2. **Drag and drop file uploads**

Enable drag and drop support for sqlite and JSON files on the home screen.



3. **Recent file uploads on the home page**

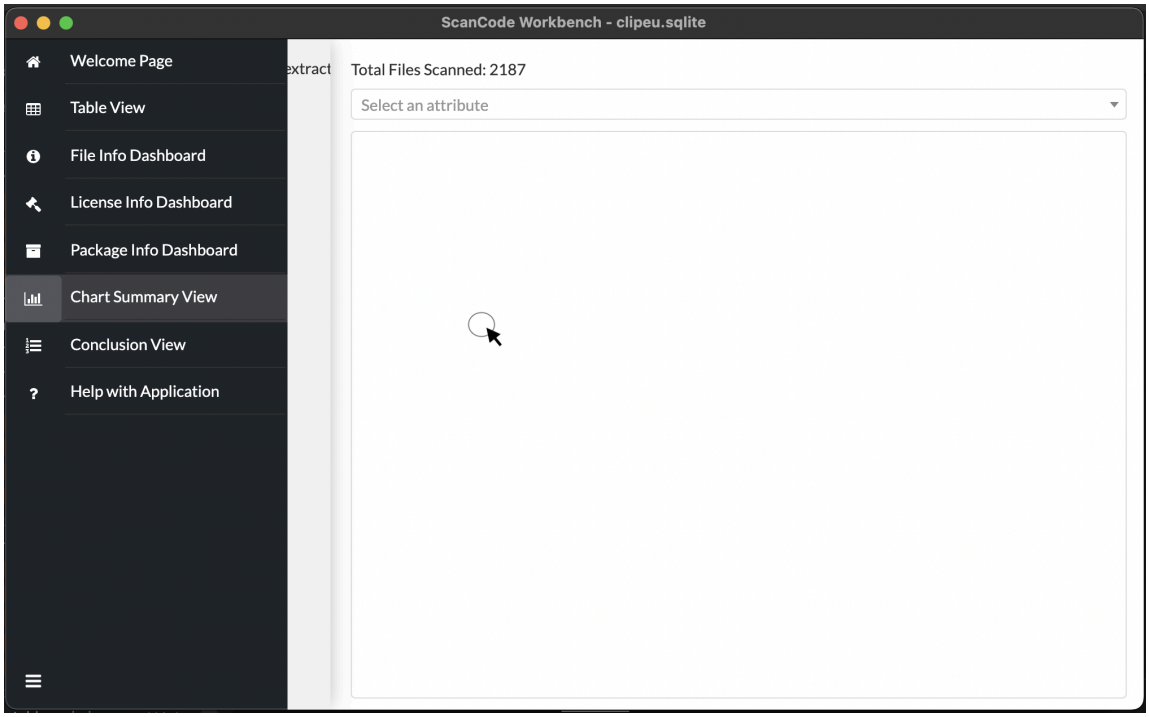
   Store the last 4 / 5 opened files and show a list on the home page, through which any of these can be opened with a click.
   Design - https://www.figma.com/file/Ifn6b54LIb135DORW8NgcR/Untitled?node-id=2%3A73



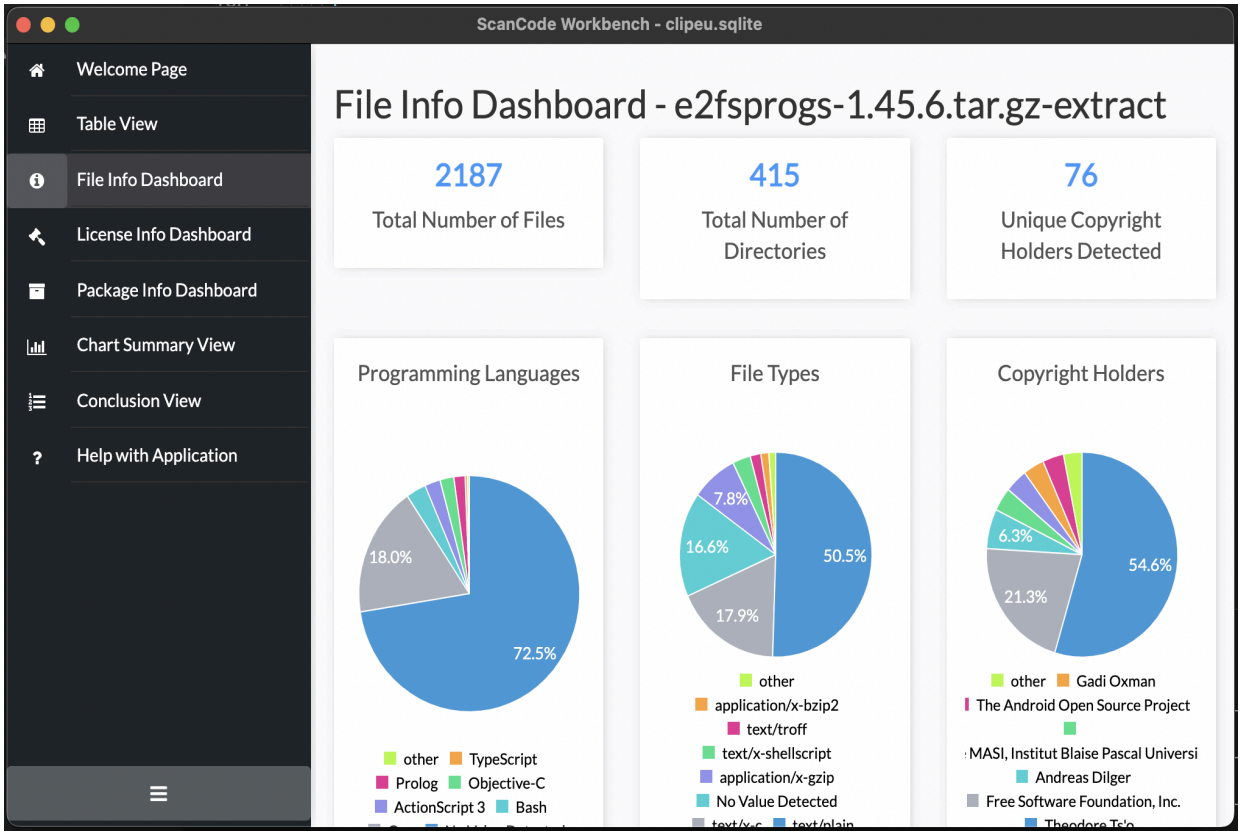
   To persist these file paths, and the last opened files, we can simply use window.localStorage or use electron-store or similar utility libraries

4. ## Improve navigation controls

Auto-shrink the expanded navbar if clicked outside the navigation drawer.



Expand the menu bar button to fit navbar width (in the expanded state)
Eg.



Roadmap

Upgrade packages

Test out latest version of all the packages, propose a list of suitable package versions

Remove conclusions module

Improve Table view functionality (search and filter)

Improve the logic for search and filter options on Table View

Prototype a TS + React version of workbench

Create basic routes, navigation and fundamental functionalities in React + TS implementation to check the feasability

Refactor the workbench to TS + React version

Bring all the features of js workbench to Refactored version

Improve the UI of workbench

Improve styling and layout structure. Eg. Decide which library to use for Table view section
Eg .

Detailed timeline

Period	Tasks
Pre-GSoC	● Discuss the project roadmap in detail ● Settling smaller pending issues in workbench
Phase 1	
Community Bonding Period (May 20 - June 12)	● Research about the libraries to be used ● Discuss in detail about Table view, conclusions ● Resolve overall queries about project and roadmap
June 13 - June 19 (Week 1)	● Research about all the packages and choose optimal versions ● Test and finalizing all the package versions.
June 20 - June 26 (Week 2)	● Remove conclusions module and associated helpers ● Discuss and explore all the issues regarding table view
June 27 - July 10 (Week 3 & 4)	● Search about the reported filter issues and check for functional bugs in table view ● Fix the filter related issues ● Start preparing a React + TS prototype of workbench to finalize the approach and get an overview of the refactoring
July 11 - July 24 (Week 5 & 6)	● Work on feedbacks on the prototype ● Demonstrate context API for global state sync ● Research other charts libraries and decide whether to stay with original one or port to another one.
July 25	● Submit phase 1 evaluation report
Phase 2	
July 26 - August 8 (Week 7 & 8)	● Port all features from vanilla JS implementation to React + TS ● Implement best suitable charts library
August 9 - August 15 (Week 9)	● Implement a production-ready refactored version of workbench
Aug 16 - Aug 22 (Week 10)	● Buffer for pending tasks
Aug 23 - Sept 5 (Week 11 & Week 12)	● Discuss datatable design and library(whether to use react version of the same lib or try other libs) ● Prototype different datatable libs and designs to decide which one suits the workbench
Sept 6 - Sept 12 (Week 13)	● Leave for Ganesh festival
Sept 13 - Sept 19 (Week 14)	● Decide final design for datatable ● Implement final iteration of datatable

Sept 20 - Sept 26 (Week 15)	• Implement drag and drop file uploads (JSON and SQLite uploads) • Create Recently imported files section
Sept 27 - Oct 3 (Week 16)	• Improve navbar (Navigation bugs and design)
Oct 4 - Oct 10 (Week 17)	• Resolve any pending feedbacks • Final polishes
Oct 11 - Oct 17 (Week 18)	• Prepare report for GSoC • Final Project Submission

About me

Name	Omkar Phansopkar
Email	omkarphansopkar@gmail.com
Github	https://github.com/OmkarPh
LinkedIn	https://www.linkedin.com/in/omkarphansopkar/
Gitter	@OmkarPh https://gitter.im/OmkarPh
Phone no.	+91 70452 70840
Location	Mumbai, India
Timezone	IST (EST +9:30)
Education	Qualification: Diploma Year: 3rd Year Major: Computer Engineering Institute: Government Polytechnic, Mumbai
Resume	https://drive.google.com/file/d/1dOQjE_NJDJzn4Vx68qE52pLKrfNYznyf/view?usp=sharing

My Skills:

Languages : Javascript, Typescript, Solidity, Java, Go, Python
Frameworks : React, Node.js, Electron
Utilities : Git, Github, VS Code, Unix
Interests : Web development, Blockchain & web3

Pre - GSoC contributions for AboutCode

nexB/scancode-workbench

ISSUE #508 - PR #509 - Updated old dependencies

ISSUE #502 - PR #511 - Add support for python3

ISSUE #446 - PR #513 - Changed issues link

Some of my best projects

- **Medblock**
  A decentralized platform to store medical records of citizens securely.
  Built as a part of SPIT hackathon and we won **1st prize** for the same.

  Smart contracts are written in solidity and ready to be deployed on any EVM compatible blockchain (Currently deployed on ethereum testnet).
  The user interface is in Typescript
  Learn more here: https://github.com/code-squads/medblock

- **Althsis**
  Althsis is a financial analysis solution to analyze the financial data provided and approved through the Setu Aggregator API. Right now it analyses only the bank data because of the limitations of the API.
  Built as a part of PhonePe fintech open month hackathon and we won **3rd prize** for the same.

  The project was built using React, NodeJS, and Setu's API
  Learn more here: https://github.com/code-squads/althsis

- **OpCoin**
  Blockchain with native crypto OP coin built from scratch using javascript. It serves as an understandable version of actual blockchain implementation which is hard to understand for newbies. For me, it helped to learn the nitty-gritty of the blockchain.

  The project is built using NodeJS & React
  Learn more here: https://github.com/OmkarPh/opcoin

Time Commitments

Our university's current semester is reserved for internships, hence I don't have exams or other academic commitments this summer. Moreover, I don't have any other commitments or vacations planned for the GSoC duration except for the 1 week's leave for the Ganesh festival. Hence, GSoC will be my primary focus and am willing to devote more than 40 hrs per week. I'll keep project status posted to mentors regularly.

Contact: I will be available anytime by Email, Gitter, or any Video Conference if required.

What will be my working hours?

I will be working anytime from 10:00 AM to 11:00 PM IST
(12:30 AM to 1:30 PM EST / 4:30 am to 5:30 pm UTC)

Post-GSoC

Open source is a great way to gain some professional work experience, hence I look forward to work with Aboutcode in the future as well (Hopefully, in other projects like scancode.io as well).
Also, I feel that the Scancode workbench is an underrated part of the AboutCode ecosystem. We can expand it to work out other utilities right from the workbench making the scancode workflow seamless.
Apart from my contributions, I'll also help the newcomers to the best of my ability and help the AboutCode community grow bigger 💪

- Signing off 🫡 ✌️