Introduction

The summer student project included building of the web user interface (UI) for Post Mortem data from scratch using modern frontend technologies including Angular 6, TypeScript and SCSS. Post Mortem Framework orchestrates the analysis flow and provides all necessary infrastructure to the analysis modules. Next to implementation, data visualization and design principles were considered before and during the implementation of the application. The work included design and development of three main pages of the web user interface: raw data page, event data page and analysis data page. Additionally advanced filtering connected to URL and possibility to explore and visualize selected data further were implemented. In this summarizing report, different phases of the work from design, visualization, and implementation are shortly introduced.

Design

Usability of a user interface (UI) is invisible for users if it is done well, but users notice bad usability very fast. Creating a good usability takes time, as each input and state change has to be validated to prevent errors and dynamically guide user instead of just adding UI components to the UI and let user solve problems when the state is incorrect. Below are some of the main design principles that were applied during the design process.

- **Dynamic UI and updates**
  - Although one page applications are asynchronous, it is important that UI adjusts to the status of the UI in real time. This can be seen for example in filtering, where filter options change based on selected filtering type. All fields that are not related to the current state of the UI are hidden to reduce cognitive load of the users.

- **Error prevention**
  - One of the main principles of usability guidelines is to prevent errors before they can happen, and this is used in the UI for example before API calls are sent. In other words, because call parameters are validated before sending, all sent API calls should give valid response if the API is working as expected.

- **Contextual help**
  - Good UI does not need manuals but sometimes additional help is needed especially if novice users are using the UI. Contextual help means that the help is provided in context, just in the place where the help might be needed, instead of searching the information from manuals. In modern UI design, it is even recommended to avoid external manuals.

- **Style guidelines**

---

1. [https://mpe.web.cern.ch/content/structure/mpe-ms/postmortem](https://mpe.web.cern.ch/content/structure/mpe-ms/postmortem)
○ CERN has own design guidelines for websites that define for example the font and usage of toolbar and logo. Guidelines are available here [https://design-guidelines.web.cern.ch/guidelines-cern-websites](https://design-guidelines.web.cern.ch/guidelines-cern-websites)

- Accessibility
  ○ The UI utilizes color palette that is suitable for different variations of color blindness and it is made sure that the contrast between background and foreground is high. For screen readers, UI uses aria-labels. The selected color scheme is presented below.

```
conservative 7-color palette adapted for color blindness

<table>
<thead>
<tr>
<th>Color</th>
<th>Color name</th>
<th>RGB (1-255)</th>
<th>CMYK (%)</th>
<th>P</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td></td>
<td>0, 0, 0</td>
<td>0, 0, 0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>Orange</td>
<td></td>
<td>232, 168, 0</td>
<td>0, 90, 100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sky blue</td>
<td></td>
<td>66, 160, 233</td>
<td>80, 0, 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Beige</td>
<td></td>
<td>156, 115</td>
<td>70, 75, 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Yellow</td>
<td></td>
<td>240, 228, 66</td>
<td>10, 90, 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Blue</td>
<td></td>
<td>6, 114, 176</td>
<td>100, 50, 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Reddish purple</td>
<td>204, 121, 167</td>
<td>10, 70, 0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

[http://mkweb.bcgsc.ca/colorblind/](http://mkweb.bcgsc.ca/colorblind/)

The actual design process was started with drawing mockups for each planned page. Example of mockup that was selected in the early design process is presented below:

![Initial mockup of the layout of the page utilizing details on demand navigation strategy and filtering on top.](https://example.com/mockup.png)

**Visualization**

The goal of the information visualization is to present as much data as possible as clearly as possible. Below are some visualization principles that were used in the UI design.

- Visual Seeking Mantra
  ○ The core of the website is based on Shneidermann’s visual information seeking mantra: overview first, zoom and filter, and details on demand. In the visualization of large datasets, this mantra is widely used and validated to be one of the most effective ways to visualize large amounts of data efficiently.

2 [https://infovis-wiki.net/wiki/Visual_Information-Seeking_Mantra](https://infovis-wiki.net/wiki/Visual_Information-Seeking_Mantra)
**Preattentive attributes**

- Preattentive processing attributes are attributes that human eye processes automatically without need of cognitive effort. Examples of preattentive attributes are color, size, length, shape, and position. These attributes make data interpretation significantly faster compared to using just text for example. However, it is not recommended to use more than 9 types of visual glyphs and less than five is most effective for limited perception capabilities of a human. In pm-browser UI, the maximum amount of colors (datasets) in graphs is 9.

**Visualization pipeline**

- Main idea of visualization pipeline\(^3\) is to find what kind of visualizations are suitable for the underlying data structures. Post mortem data consists of high density temporal data and this data can also be divided into systems that have a hierarchical structure of classes and sources for example. Scatter plots are well suitable for temporal data, and for this reason scatter plot was utilized in the web UI. For hierarchical structures visual seeking mantra is always a good choice to avoid visual clutter of large data sets. If too many values are tried to visualize simultaneously, human eye cannot visualize any parts of the data efficiently.

**Implementation**

The angular application consisted of three main modules (one for each page) and a shared module which contained elements that were used multiple times. In Angular, services work as a bridge between API and UI and all data should always be transferred via services. In pm-browser, all api calls go through dedicated services for each page that follow also current API structure: PmData, PmEvent, and AnalysisResult. Additionally, the global service stores data about the global state of the application and global functions such as filter states and url functions.

Angular applications consist of components that contain other components. Pm-browser contains three similar pages that follow the similar structure presented below. Details panel includes also other components including generic responsive info tables and chart builder components. Each page extends page layout and connects this layout to the API data by using dedicated services. Additionally generic functions used in multiple components are added to helper services where they can be easily maintained and injected into components.

Summary

The finished project allows users to search raw, event, and analysis data by using advanced time and target filtering. A user can see the results in a table or on a timeline graph and by clicking a row in the table or data point in the graph, a details panel will open. The details panel lets the user explore data in a table view, zoomable generic chart view (one or multiple data sets) or as a beautified raw JSON data. The main challenge faced during the project was
Javascript's inability to support 'long' data type values, which meant that the data from the API could not be queried directly as a JSON but as a string to avoid rounding of the numbers. This was solved by receiving data as string from the API and manually converting long values to strings with regex by adding quotation marks to long values. Future work includes improvement of automatic tests and possibly adding more pages and filters to the website. The project was wide for a two month summer student project, but all goals set in the beginning were met with high usability, abstraction, and maintainability of code.