Documentation for Financial Transactions HTML Page
Jason N.
June 1, 2020

Contents

1 HTML

1.1 Preamble and head

1.1.1 Preamble

Declares the document as HTML5.

```
1  <!DOCTYPE html >
```

1.1.2 meta charset

Specifies that characters in the file are encoded in UTF-8.

```
4  <meta charset = "UTF-8"/>
```

1.1.3 link rel="stylesheet"

Imports the CSS file.

```
5  <link rel="stylesheet" type="text/css" href="./style.css"/>
```

1.1.4 Scripts

Imports the main Javascript file, responsible for the table and UI.

```
7  <script src="./script.js"></script>
```

Imports the Google API library.

```
8  <script src="https://apis.google.com/js/api.js"></script>
```

Imports other Javascript files, responsible for database management.

```
9   <script src="./googleApiScript.js"></script>
10  <script src="./mysqlScript.js"></script>
11  <script src="./localStorageScript.js"></script>
12  <script src="./imageFirestore.js"></script>
```

1.2 Inputs

Disables autocomplete which remembers past user input by default. `return false` specifies that no POST request should be made to the server.

```
20  <form onsubmit="return false" autocomplete="off">
```

1.2.1 Labels

Identifies the purpose of the field to the user, allows the user to select the field by clicking the label. This element is also used by accessibility tools to identify the field.

```
22  <label for="date">Date:</label><br/>
```

1.2.2 Date

1.2.3 Text

1.2.4 List

1.2.5 Buttons

1.3 Table

1.3.1 thead

1.3.2 tbody

2 Main Javascript

2.1 getData()

2.2 validate()

2.2.1 Check empty

2.2.2 Check NaN

2.2.3 Check date

2.3 generateId()

2.4 calculateCostBasis()

2.5 addTransaction()

2.6 deleteRow()

2.7 editRow()

2.8 saveChanges()

2.9 discardChanges()

2.10 sortTable()

3 CSS

3.1 Vertical Scrolling Table

3.2 Horizontal Scrolling on Overflow

3.3 Miscellaneous

3.3.1 Sort buttons

3.3.2 Editing highlight

3.3.3 Table borders

A   HTML Source Code

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset = "UTF-8"/>
5          <link rel="stylesheet" type="text/css" href="./style.css"/>
6
7          <script src="./script.js"></script>
8          <script src="https://apis.google.com/js/api.js"></script>
9          <script src="./googleApiScript.js"></script>
10         <script src="./mysqlScript.js"></script>
11         <script src="./localStorageScript.js"></script>
12         <script src="./imageFirestore.js"></script>
13     </head>
14     <body>
15         <article id="inputFields">
16             <section class="sectionHead">
17                 <h1>Input:</h1>
18                 <button class="toggleSection" type="button" onclick="
                       ↪ toggleSection(this)">Hide</button>
19             </section>
20             <form onsubmit="return false" autocomplete="off">
21                 <section>
22                     <label for="date">Date:</label><br/>
23                     <input id="date" name="date" type="date" placeholder="
                           ↪ yyyy-mm-dd"/>
24                 </section>
25
26                 <section>
27                     <label for="account">Account Number:</label><br/>
28                     <input id="account" name="account" list="accountsList"
                           ↪ type="text" placeholder="Account Number"/>
29                     <datalist id="accountsList">
30                     </datalist>
31                 </section>
32
33                 <section>
34                     <label for="type">Transaction Type:</label><br/>
35                     <select id="type" name="type">
36                         <option value=""></option>
37                         <option value="BUY">BUY</option>
38                         <option value="SELL">SELL</option>
39                         <option value="!DIVIDEND">DIVIDEND</option>
40                         <option value="!INTEREST">INTEREST</option>
41                         <option value="!WITHDRAW">WITHDRAW</option>
42                         <option value="!DEPOSIT">DEPOSIT</option>
43                     </select>
44                 </section>
45
46                 <section>
47                     <label for="security">Security:</label><br/>
48                     <input id="security" name="security" list="
                           ↪ securitiesList" type="text" placeholder="
```

```html
                                   ↪ Security"/>
                <datalist id="securitiesList">
                </datalist>
            </section>

            <section>
                <label for="amount">Amount:</label><br/>
                <input id="amount" name="amount" type="text"
                    ↪ placeholder="Unit Amount"/>
            </section>

            <section>
                <label for="dAmount">$ Amount:</label><br/>
                <input id="dAmount" name="dAmount" type="text"
                    ↪ placeholder="$ Amount"/>
            </section>

            <section>
                <label id="fileUploadLabel" for="fileUpload">Upload
                    ↪ file</label>
                <input id="fileUpload" name="fileUpload" type="file"
                    ↪ onchange="fileUploadChanged();" multiple/>
                <button id="removeFile" type="button" onclick="
                    ↪ removeFileUpload();">X</button>
            </section>

            <section>
                <button id="add" type="submit" onclick="
                    ↪ addTransactionButton();">Add Transaction</button
                    ↪ >
                <button id="save" type="button" hidden="true" onclick
                    ↪ ="saveChanges();">Save</button>
                <button id="discard" type="button" hidden="true"
                    ↪ onclick="discardChanges();">Discard</button>
            </section>
        </form>
    </article>

    <article id="filter">
        <section class="sectionHead">
            <h1>Filters:</h1>
            <button class="toggleSection" type="button" onclick="
                ↪ toggleSection(this)">Hide</button>
        </section>
        <form onsubmit="return false" autocomplete="off">
            <section>
                <label for="filterId">Transaction ID:</label><br/>
                <input id="filterId" class="filterField" name="
                    ↪ filterId" type="text" placeholder="Transaction
                    ↪ ID"/>
            </section>

            <section>
                <label for="startDate">From:</label><br/>
```

```
90              <input id="startDate" class="filterField" name="
                    ↪ startDate" type="date" placeholder="yyyy-mm-dd
                    ↪ "/>

92              <br/>

94              <label for="endDate">To:</label><br/>
95              <input id="endDate" class="filterField" name="endDate"
                    ↪  type="date" placeholder="yyyy-mm-dd"/>
96          </section>

98          <section>
99              <label for="filterAccount">Account Number:</label><br
                    ↪ />
100             <input id="filterAccount" class="filterField" name="
                    ↪ filterAccount" type="text" placeholder="Account
                    ↪ Number"/>
101         </section>

103         <section>
104             <label for="type">Transaction Type:</label><br/>
105             <select id="filterType" class="filterField" name="
                    ↪ filterType">
106                 <option value=""></option>
107                 <option value="BUY">BUY</option>
108                 <option value="SELL">SELL</option>
109                 <option value="DIVIDEND">DIVIDEND</option>
110                 <option value="INTEREST">INTEREST</option>
111                 <option value="WITHDRAW">WITHDRAW</option>
112                 <option value="DEPOSIT">DEPOSIT</option>
113             </select>
114         </section>

116         <section>
117             <label for="filterSecurity">Security:</label>
118             <span title="Enter search terms here. Separate search
                    ↪ terms with && or || for AND and OR statements,
                    ↪ respectively. Exclusive filters are marked by a
                    ↪ leading !. Use || to filter by multiple
                    ↪ securities (e.g. SPY || TLT) and && to exclude
                    ↪ multiple securities (e.g. !SPY && !TLT).">?</
                    ↪ span><br/>
119             <input id="filterSecurity" class="filterField" name="
                    ↪ filterSecurity" type="text" placeholder="
                    ↪ Security"/>
120         </section>

122         <section>
123             <label for="lowAmount">Min Amount:</label><br/>
124             <input id="lowAmount" class="filterField" name="
                    ↪ lowAmount" type="text" placeholder="Min Amount
                    ↪ "/>

126             <br/>
```

```
127
128                    <label for="highAmount">Max Amount:</label><br/>
129                    <input id="highAmount" class="filterField" name="
                           ↪ highAmount" type="text" placeholder="Max Amount
                           ↪ "/>
130                </section>
131
132            <section>
133                    <label for="lowDAmount">Min $ Amount:</label><br/>
134                    <input id="lowDAmount" class="filterField" name="
                           ↪ lowDAmount" type="text" placeholder="Min $
                           ↪ Amount"/>
135
136                    <br/>
137
138                    <label for="highDAmount">Max $ Amount:</label><br/>
139                    <input id="highDAmount" class="filterField" name="
                           ↪ highDAmount" type="text" placeholder="Max $
                           ↪ Amount"/>
140                </section>
141
142            <section>
143                    <label for="lowCostBasis">Min Cost Basis:</label><br/>
144                    <input id="lowCostBasis" class="filterField" name="
                           ↪ lowCostBasis" type="text" placeholder="Min Cost
                           ↪ Basis"/>
145
146                    <br/>
147
148                    <label for="highCostBasis">Max Cost Basis:</label><br
                           ↪ />
149                    <input id="highCostBasis" class="filterField" name="
                           ↪ highCostBasis" type="text" placeholder="Max Cost
                           ↪  Basis"/>
150
151                    <br/>
152
153                    <label for="filterNa">Filter N/A:</label>
154                    <input id="filterNA" name="filterNA" type="checkbox"/>
155                </section>
156
157            <section>
158                    <button type="submit" onclick="applyFilter()">Apply</
                           ↪ button>
159                    <button type="button" onclick="clearFilter()">Clear</
                           ↪ button>
160                </section>
161            </form>
162        </article>
163
164        <article id="options">
165            <section class="sectionHead">
166                <h1>Options:</h1>
```

```
167            <button class="toggleSection" type="button" onclick="
                ↪ toggleSection(this)">Hide</button>
168        </section>
169        <form onsubmit="return false" autocomplete="off">
170            <section>
171                <button id="toggleId" type="button" onclick="toggleID
                    ↪ ()">Hide Transaction ID</button>
172            </section>
173
174            <section>
175                <label for="fileInput">Import Transaction Type CSV
                    ↪ file</label>
176                <input id="fileInput" name="fileInput" type="file"
                    ↪ onchange="readFile(this)"/>
177            </section>
178
179            <section id="transactionTypesConfig">
180                    <label for="typesArray">Transaction types (comma-
                        ↪ separated):</label>
181                    <input id="typesArray" name="typesArray" type="
                        ↪ text" placeholder="Transaction types"/>
182                    <button id="applyTypesButton" type="button"
                        ↪ onclick="applyTypes()">Apply</button>
183                    <button id="editTypesButton" type="button" onclick
                        ↪ ="editTypes()">Edit</button>
184                    <button type="button" onclick="saveFile()">Save</
                        ↪ button>
185            </section>
186
187            <section id="googleSheetButtons">
188                <button type="button" onclick="writeGoogleSheetDB()">
                    ↪ Write to Sheets</button>
189                <button type="button" onclick="readGoogleSheetDB()">
                    ↪ Read from Sheets</button>
190            </section>
191
192            <section>
193                <section>
194                    <select id="sheet" onchange="getNewSheetData()">
195                        <option value="1R0HpaAIUw-JHX8SrzvkEPCG1qgI-
                            ↪ siJ9oucY6g5e4Co">default</option>
196                    </select>
197                </section>
198
199                <section>
200                    <select id="tab" onchange="getNewTabData()">
201                    </select>
202                </section>
203            </section>
204
205            <section id="logoutSection">
206                <button type="button" onclick="loadSheetData()">Reload
                    ↪   Sheets</button>
```

```
207              <button type="button" onclick="auth2.disconnect()">Log
                 ↪  out</button>
208           </section>
209
210           <section>
211              <button type="button" onclick="writeToFirebase()">
                 ↪ Write to Firebase</button>
212              <button type="button" onclick="readFromFirebase()">
                 ↪ Read from Firebase</button>
213           </section>
214
215           <section>
216              <button type="button" onclick="writeToFirestore()">
                 ↪ Write to Firestore</button>
217              <button type="button" onclick="readFromFirestore()">
                 ↪ Read from Firestore</button>
218           </section>
219
220           <section>
221              <button type="button" onclick="writeToMySQL()">Write
                 ↪ to MySQL</button>
222              <button type="button" onclick="readFromMySQL()">Read
                 ↪ from MySQL</button>
223           </section>
224        </form>
225     </article>
226
227     <article>
228     <h1>Table:</h1>
229     <article id="table">
230        <table>
231           <thead>
232              <tr>
233                 <th class="frozenColumn1">
234                    <section>
235                       Transaction ID
236                    </section>
237                    <section class="sort">
238                       <button type="button" onclick="sortTable
                          ↪ (0, true)">^</button>
239                       <button type="button" onclick="sortTable
                          ↪ (0, false)">v</button>
240                    </section>
241                 </th>
242                 <th class="frozenColumn2">
243                    <section>
244                       Date
245                    </section>
246                    <section class="sort">
247                       <button type="button" onclick="sortTable
                          ↪ (1, true)">^</button>
248                       <button type="button" onclick="sortTable
                          ↪ (1, false)">v</button>
249                    </section>
```

```
250                    </th>
251                    <th>
252                        <section>
253                            Account Number
254                        </section>
255                        <section class="sort">
256                            <button type="button" onclick="sortTable
                                 ↪ (2, true)">^</button>
257                            <button type="button" onclick="sortTable
                                 ↪ (2, false)">v</button>
258                        </section>
259                    </th>
260                    <th>
261                        <section>
262                            Transaction Type
263                        </section>
264                        <section class="sort">
265                            <button type="button" onclick="sortTable
                                 ↪ (3, true)">^</button>
266                            <button type="button" onclick="sortTable
                                 ↪ (3, false)">v</button>
267                        </section>
268                    </th>
269                    <th>
270                        <section>
271                            Security
272                        </section>
273                        <section class="sort">
274                            <button type="button" onclick="sortTable
                                 ↪ (4, true)">^</button>
275                            <button type="button" onclick="sortTable
                                 ↪ (4, false)">v</button>
276                        </section>
277                    </th>
278                    <th>
279                        <section>
280                            Amount
281                        </section>
282                        <section class="sort">
283                            <button type="button" onclick="sortTable
                                 ↪ (5, true)">^</button>
284                            <button type="button" onclick="sortTable
                                 ↪ (5, false)">v</button>
285                        </section>
286                    </th>
287                    <th>
288                        <section>
289                            $ Amount
290                        </section>
291                        <section class="sort">
292                            <button type="button" onclick="sortTable
                                 ↪ (6, true)">^</button>
293                            <button type="button" onclick="sortTable
                                 ↪ (6, false)">v</button>
```

```
294                                              </section>
295                                          </th>
296                                          <th>
297                                              <section>
298                                                  Cost Basis
299                                              </section>
300                                              <section class="sort">
301                                                  <button type="button" onclick="sortTable
                                                     ↪ (7, true)">^</button>
302                                                  <button type="button" onclick="sortTable
                                                     ↪ (7, false)">v</button>
303                                              </section>
304                                          </th>
305                                          <th>
306                                              <section>
307                                                  File
308                                              </section>
309                                              <section class="sort">
310                                                  <button type="button" onclick="sortTable
                                                     ↪ (8, true)">^</button>
311                                                  <button type="button" onclick="sortTable
                                                     ↪ (8, false)">v</button>
312                                              </section>
313                                          </th>
314                                          <th>Actions</th>
315                                      </tr>
316                                  </thead>
317                                  <tbody id="tableBody">
318                                  </tbody>
319                          </table>
320                  </article>
321                  </article>
322
323                  <!-- The core Firebase JS SDK is always required and must be
                    ↪ listed first -->
324                  <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
                    ↪ app.js"></script>
325
326                  <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
                    ↪ analytics.js"></script>
327                  <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
                    ↪ database.js"></script>
328                  <script src="https://www.gstatic.com/firebasejs/7.14.3/firebase-
                    ↪ firestore.js"></script>
329
330                  <script>
331                  // Your web app's Firebase configuration
332                  var firebaseConfig = {
333                      apiKey: "AIzaSyAmZLFZHDAB9evhvNunxOe5GxXRd_OizmU",
334                      authDomain: "financial-transactions-6f065.firebaseapp.com",
335                      databaseURL: "https://financial-transactions-6f065.firebaseio.
                        ↪ com",
336                      projectId: "financial-transactions-6f065",
337                      storageBucket: "financial-transactions-6f065.appspot.com",
```

```
338          messagingSenderId: "82206982479",
339          appId: "1:82206982479:web:8937bbd1bd4fb6022b053a",
340          measurementId: "G-0564DT8RNQ"
341        };
342        // Initialize Firebase
343        firebase.initializeApp(firebaseConfig);
344        firebase.analytics();
345
346        var database = firebase.database();
347        var firestore = firebase.firestore();
348        </script>
349
350        <script src="./firebaseScript.js"></script>
351    </body>
352 </html>
```

B Javascript Source Code

B.1 Main Script

```
1  function formattedStringToNumber(numberAsString) {
2      var number;
3
4      if(numberAsString[0] == '$') {
5          numberAsString = numberAsString.substr(1);
6      }
7
8      number = Number(numberAsString.replace(/,/g, ''));
9
10     return number;
11 }
12
13 function numberToFormattedString(number) {
14     var numberAsString;
15
16     numberAsString = String(number).replace(/\B(?=(\d{3})+(?!\d))/g, ",");
17
18     return numberAsString;
19 }
20
21 function getData() {
22     var date = document.getElementById("date");
23     var account = document.getElementById("account").value;
24     var type = document.getElementById("type").value;
25     var security = document.getElementById("security").value;
26     var amount = document.getElementById("amount").value;
27     var dAmount = document.getElementById("dAmount").value;
28
29     security = security.toUpperCase();
30
31     amount = formattedStringToNumber(amount);
32
33     dAmount = formattedStringToNumber(dAmount);
34
35     if(validate(date, account, type, security, amount, dAmount)) {
36         var costBasis = '$' + numberToFormattedString(calculateCostBasis(
             ↪ amount, dAmount));
37         date = date.value;
38
39         amount = numberToFormattedString(amount);
40         dAmount = '$' + numberToFormattedString(dAmount.toFixed(2));
41
42         return [ date, account, type, security, amount, dAmount, costBasis
             ↪ ];
43     }
44     else return false;
45 }
46
47 function validate(date, account, type, security, amount, dAmount) {
```

```
48        if(!validateDate(date)) return false;
49        if(!validateAccount(account)) return false;
50        if(!validateType(type)) return false;
51        if(!validateSecurity(security)) return false;
52        if(!validateAmount(amount)) return false;
53        if(!validateDAmount(dAmount)) return false;
54
55        return true;
56   }
57
58   function validateDate(date) {
59        realDate = new Date();
60        inputDate = date.valueAsNumber;
61
62        if(date.value == '') {
63            alert('Error: Missing date');
64            return false;
65        }
66
67        if(!date.checkValidity()) {
68            alert('Error: Invalid date');
69            return false;
70        }
71
72        if(realDate.valueOf() < inputDate) {
73            alert('Error: Date is in the future');
74            return false;
75        }
76
77        return true;
78   }
79
80   function validateAccount(account) {
81        if(account == '') {
82            alert('Error: Missing Account Number');
83            return false;
84        }
85
86        return true;
87   }
88
89   function validateType(type) {
90        if(type == '') {
91            alert('Error: Missing Transaction Type');
92            return false;
93        }
94
95        return true;
96   }
97
98   function validateSecurity(security) {
99        if(security == '') {
100           alert('Error: Missing Security');
101           return false;
```

```
102 |      }
103 |
104 |      return true;
105 | }
106 |
107 | function validateAmount(amount) {
108 |      if(amount == '') {
109 |          alert('Error: Missing Amount');
110 |          return false;
111 |      }
112 |
113 |      if(isNaN(amount)) {
114 |          alert('Error: Invalid Amount');
115 |          return false;
116 |      }
117 |
118 |      return true;
119 | }
120 |
121 | function validateDAmount(dAmount) {
122 |      if(dAmount == '') {
123 |          alert('Error: Missing $ Amount');
124 |          return false;
125 |      }
126 |
127 |      if(isNaN(dAmount)) {
128 |          alert('Error: Invalid $ Amount');
129 |          return false;
130 |      }
131 |
132 |      return true;
133 | }
134 |
135 | function generateId() {
136 |      var id = '';
137 |      var idLength = 6;
138 |
139 |      var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
140 |      var charactersLength = characters.length;
141 |
142 |      var unique = false;
143 |
144 |      while(!unique) {
145 |          for(var i = 0; i < idLength; i++) {
146 |              id += characters.charAt(Math.floor(Math.random() *
     |               ↪ charactersLength));
147 |          }
148 |
149 |          unique = true;
150 |          for(var i = 0; i < document.getElementsByClassName('idCell').
     |               ↪ length; i++) {
151 |              if(document.getElementsByClassName('idCell')[i].innerText ==
     |                   ↪ id) {
152 |                  unique = false;
```

```
153                      break;
154                 }
155             }
156         }
157         return id;
158 }
159
160 function calculateCostBasis(amount, dAmount) {
161     costBasis = (dAmount / amount).toFixed(2);
162     return costBasis;
163 }
164
165 function addTransaction(data) {
166     var staging = data;
167     var tableBody = document.getElementById('tableBody');
168     var newRow = tableBody.insertRow(0);
169     newRow.classList += "bodyRow";
170
171     var actionsContent = "<button type='button' onclick='editRow(this)'>
        ↪ Edit</button><button type='button' onclick='deleteRow(this)'>
        ↪ Delete</button>";
172     var fileContent = '<table><tbody>';
173     if(data.length > 8) {
174         for(let i = 0; i < data[8].length; i++) {
175             fileContent += "<tr><td><a onclick='downloadFile('" + data[8][
                ↪ i][0] + "');' href='javascript:void(0);'>" + data[8][i
                ↪ ][1] + "</a></td>";
176             fileContent += "<td><button type='button' onclick='
                ↪ removeFileFromTable('" + data[8][i][0] + "', this);'>-</
                ↪ button></td></tr>";
177         }
178     }
179     fileContent += '</tbody></table><input type="file" onchange="addFile(
        ↪ this, 0);" multiple/>';
180     staging[8] = fileContent;
181     staging[9] = actionsContent;
182
183     var calculateCostBasis = true;
184     if(data[3][0] == '!') {
185         calculateCostBasis = false;
186         data[3] = data[3].substr(1);
187     }
188
189     for(var i = 0; i < 10; i++) {
190         var newCell = newRow.insertCell(i);
191         var idShowing = (document.getElementById('toggleId').innerText ==
            ↪ "Hide Transaction ID");
192
193         newCell.innerHTML = data[i];
194
195         if(i == 0) {
196             if(idShowing)
197                 newCell.classList = "idCell frozenColumn1";
198             else {
```

```
199              newCell.classList = "idCell";
200              newCell.setAttribute("hidden", true);
201          }
202      }
203      else if(i == 1) {
204          if(idShowing)
205              newCell.classList = "frozenColumn2";
206          else
207              newCell.classList = "frozenColumn1";
208      }
209      /*
210      else if(i == 2) {
211          if(idShowing)
212              newCell.classList = "frozenColumn3";
213          else
214              newCell.classList = "frozenColumn2";
215      }
216      */
217
218      if(i == 7 && !calculateCostBasis) {
219          newCell.innerHTML = "N/A";
220      }
221      }
222 }
223
224 function fileIdGenerator() {
225      return Math.floor(Math.random() *
          ↪ 1000000000000000000000000000000000000000).toString(36);
226 }
227
228 function addTransactionButton() {
229      var data = getData();
230      if(data) {
231          var id = generateId();
232          data.unshift(id);
233
234          fileList = new Array()
235          uploadFile(data, addTransactionWithFileName, fileList, 0);
236          clearInput(false);
237      }
238 }
239
240 function addTransactionWithFileName(data, fileName) {
241      addTransaction(data);
242      console.log(data);
243      loadDataLists();
244 }
245
246 function deleteRow(button) {
247      var row = button.parentElement.parentElement;
248
249      var fileRows = row.getElementsByTagName('td')[8].getElementsByTagName
          ↪ ('table')[0].getElementsByTagName('tr');
250      for(let i = 0; i < fileRows.length; i++) {
```

```
251        let fileId = fileRows [i]. getElementsByTagName ('a')[0]. getAttribute
            ↪ ('onclick'). split ('`')[1];
252        deleteFileFromIndexedDB ( fileId );
253    }
254
255    document.getElementById ("tableBody"). removeChild (row);
256
257    if ( document.getElementsByClassName ('editing'). length == 0) {
258        document.getElementById ('add'). removeAttribute ('hidden');
259        document.getElementById ('save'). setAttribute ('hidden', true);
260        document.getElementById ('discard'). setAttribute ('hidden', true);
261
262        document.getElementById ('add'). setAttribute ('type','submit');
263        document.getElementById ('save'). setAttribute ('type','button');
264    }
265    loadDataLists ();
266 }
267
268 function editRow ( button ) {
269    if ( document.getElementsByClassName ('editing'). length > 0)
270        document.getElementsByClassName ('editing')[0]. classList = "bodyRow
            ↪ ";
271
272    var row = button.parentElement.parentElement;
273    var rowContent = row.getElementsByTagName ('td');
274    row.classList = "bodyRow editing";
275
276    document.getElementById ('date'). value = rowContent [1]. innerText;
277    document.getElementById ('account'). value = rowContent [2]. innerText;
278
279    document.getElementById ('type'). value = rowContent [3]. innerText;
280    if ( document.getElementById ('type'). value == '') document.
            ↪ getElementById ('type'). value = '!' + rowContent [3]. innerText;
281
282    document.getElementById ('security'). value = rowContent [4]. innerText;
283    document.getElementById ('amount'). value = rowContent [5]. innerText;
284    document.getElementById ('dAmount'). value = rowContent [6]. innerText;
285
286    document.getElementById ('add'). setAttribute ('hidden', true);
287    document.getElementById ('save'). removeAttribute ('hidden');
288    document.getElementById ('discard'). removeAttribute ('hidden');
289
290    document.getElementById ('add'). setAttribute ('type','button');
291    document.getElementById ('save'). setAttribute ('type','submit');
292
293    removeFileUpload ();
294    uploadLabel = document.getElementById ('fileUploadLabel');
295    if ( rowContent [8]. getElementsByTagName ('tr'). length > 0) {
296        uploadLabel.innerHTML = String ( rowContent [8]. getElementsByTagName
            ↪ ('tr'). length ) + " file(s)";
297    }
298    fileEditted = false;
299 }
300
```

```
301  function saveChanges() {
302      data = getData();
303      if(data) {
304          rowToEdit = document.getElementsByClassName('editing')[0];
305          cellsToEdit = rowToEdit.getElementsByTagName('td');
306
307          if(data[2][0] == '!') {
308              data[2] = data[2].substr(1);
309              data[6] = "N/A";
310          }
311
312          for(var i = 0; i < data.length; i++) {
313              cellsToEdit[i + 1].innerHTML = data[i];
314          }
315          rowToEdit.classList = "bodyRow";
316
317          document.getElementById('add').removeAttribute('hidden');
318          document.getElementById('save').setAttribute('hidden', true);
319          document.getElementById('discard').setAttribute('hidden', true);
320
321          document.getElementById('add').setAttribute('type','submit');
322          document.getElementById('save').setAttribute('type','button');
323
324          if(fileEditted) {
325              uploadFile([cellsToEdit[8]], updateExistingFileName, new Array
                     ↪ (), 0);
326          }
327          else {
328              removeFileUpload();
329          }
330
331          resetDate();
332          clearInput(true);
333          loadDataLists();
334      }
335  }
336
337  function discardChanges() {
338      document.getElementsByClassName('editing')[0].classList = "bodyRow";
339
340      document.getElementById('add').removeAttribute('hidden');
341      document.getElementById('save').setAttribute('hidden', true);
342      document.getElementById('discard').setAttribute('hidden', true);
343
344      document.getElementById('add').setAttribute('type','submit');
345      document.getElementById('save').setAttribute('type','button');
346
347      resetDate();
348      clearInput(true);
349      removeFileUpload();
350  }
351
352  function sortTable(column, ascending) {
353      var rows = document.getElementsByClassName('bodyRow');
```

```
354
355         var sorting = true;
356         while(sorting) {
357             sorting = false;
358             for(var i = 0; i < (rows.length - 1); i++) {
359                 rowA = rows[i].getElementsByTagName('td')[column];
360                 rowB = rows[i + 1].getElementsByTagName('td')[column];
361
362                 var swap = false;
363
364                 if(ascending && rowA.innerHTML.toLowerCase() > rowB.innerHTML.
                        ↪ toLowerCase()) swap = true;
365                 else if(!ascending && rowA.innerHTML.toLowerCase() < rowB.
                        ↪ innerHTML.toLowerCase()) swap = true;
366
367                 if(swap) {
368                     sorting = true;
369                     document.getElementById('tableBody').insertBefore(rows[i +
                            ↪ 1], rows[i]);
370                 }
371             }
372         }
373 }
374
375 function resetDate() {
376     const today = new Date();
377     const year = new Intl.DateTimeFormat('en', { year: 'numeric' }).format
            ↪ (today);
378     const month = new Intl.DateTimeFormat('en', { month: '2-digit' }).
            ↪ format(today);
379     const day = new Intl.DateTimeFormat('en', { day: '2-digit' }).format(
            ↪ today);
380
381     document.getElementById('date').value = `${year}-${month}-${day}`;
382 }
383
384 function clearInput(clearAccount) {
385     if(clearAccount)
386         document.getElementById('account').value = '';
387
388     document.getElementById('type').value = '';
389     document.getElementById('security').value = '';
390     document.getElementById('amount').value = '';
391     document.getElementById('dAmount').value = '';
392 }
393
394 function validateFilters(filterId, startDate, endDate, filterAccount,
        ↪ filterType, filterSecurity, minAmount, maxAmount, minDAmount,
        ↪ maxDAmount, minCostBasis, maxCostBasis) {
395     if(!validateFilterId(filterId)) return false;
396     if(!validateDateRange(startDate, endDate)) return false;
397     if(!validateFilterAccount(filterAccount)) return false;
398     if(!validateFilterSecurity(filterSecurity)) return false;
399     if(!validateAmountRange(minAmount, maxAmount)) return false;
```

```
400         if(!validateDAmountRange(minDAmount, maxDAmount)) return false;
401         if(!validateCostBasisRange(minCostBasis, maxCostBasis)) return false;
402
403         return true;
404  }
405
406  function validateFilterId(id) {
407         return true;
408  }
409
410  function validateDateRange(start, end) {
411         if(!start.checkValidity()) {
412              alert('Error: Invalid Start Date');
413              return false;
414         }
415
416         if(!end.checkValidity()) {
417              alert('Error: Invalid End Date');
418              return false;
419         }
420
421         if(start.valueAsNumber > end.valueAsNumber) {
422              alert('Error: Invalid Date Range');
423              return false;
424         }
425
426         return true;
427  }
428
429  function validateFilterAccount(account) {
430         return true;
431  }
432
433  function validateFilterSecurity(security) {
434         return true;
435  }
436
437  function validateAmountRange(min, max) {
438         if(isNaN(Number(min))) {
439              alert('Error: Min Amount is NaN');
440              return false;
441         }
442
443         if(isNaN(Number(max))) {
444              alert('Error: Max Amount is NaN');
445              return false;
446         }
447
448         if(Number(min) > Number(max) && min != '' && max != '') {
449              if(min != '') alert(min + '2');
450              if(max != '') alert(max + '1');
451              alert('Error: Invalid Amount Range');
452              return false;
453         }
```

```
454
455        return true;
456  }
457
458  function validateDAmountRange(min, max) {
459        if(isNaN(Number(min))) {
460            alert('Error: Min $ Amount is NaN');
461            return false;
462        }
463
464        if(isNaN(Number(max))) {
465            alert('Error: Max $ Amount is NaN');
466            return false;
467        }
468
469        if(Number(min) > Number(max) && min != '' && max != '') {
470            alert('Error: Invalid $ Amount Range');
471            return false;
472        }
473
474        return true;
475  }
476
477  function validateCostBasisRange(min, max) {
478        if(isNaN(Number(min))) {
479            alert('Error: Min Cost Basis is NaN');
480            return false;
481        }
482
483        if(isNaN(Number(max))) {
484            alert('Error: Max Cost Basis is NaN');
485            return false;
486        }
487
488        if(Number(min) > Number(max) && min != '' && max != '') {
489            alert('Error: Invalid Cost Basis Range');
490            return false;
491        }
492
493        return true;
494  }
495
496  function stringFilter(filtertext, tableitem) {
497        filters = filtertext.split(" && ");
498
499        for(var i = 0; i < filters.length; i++) {
500            filterORs = filters[i].split(" || ");
501            var meetsCriteria = false;
502
503            for(var ii = 0; ii < filterORs.length; ii++) {
504                if(filterORs[ii][0] == "!" && !tableitem.toUpperCase().
                    ↪ includes(filterORs[ii].toUpperCase().substr(1)))
                    ↪ meetsCriteria = true;
```

```
505              if(filterORs[ii][0] != "!" && tableitem.toUpperCase().includes
                    ↪ (filterORs[ii].toUpperCase())) meetsCriteria = true;
506          }
507
508          if(!meetsCriteria) return false;
509      }
510
511      return true;
512 }
513
514 function applyFilter() {
515      unfilterAll();
516
517      rows = document.getElementsByClassName('bodyRow');
518
519      filterId = document.getElementById('filterId').value;
520      startDate = document.getElementById('startDate');
521      endDate = document.getElementById('endDate');
522      filterAccount = document.getElementById('filterAccount').value;
523      filterType = document.getElementById('filterType').value;
524      filterSecurity = document.getElementById('filterSecurity').value;
525      lowAmount = document.getElementById('lowAmount').value;
526      highAmount = document.getElementById('highAmount').value;
527      lowDAmount = document.getElementById('lowDAmount').value;
528      highDAmount = document.getElementById('highDAmount').value;
529      lowCostBasis = document.getElementById('lowCostBasis').value;
530      highCostBasis = document.getElementById('highCostBasis').value;
531
532      if(lowDAmount[0] == '$') lowDAmount = lowDAmount.substr(1);
533      if(highDAmount[0] == '$') highAmount = highDAmount.substr(1);
534      if(lowCostBasis[0] == '$') lowCostBasis = lowCostBasis.substr(1);
535      if(highCostBasis[0] == '$') highCostBasis = highCostBasis.substr(1);
536
537      if(validateFilters(filterId, startDate, endDate, filterAccount,
             ↪ filterType, filterSecurity, lowAmount, highAmount, lowDAmount,
             ↪ highDAmount, lowCostBasis, highCostBasis)) {
538        for(var i = 0; i < rows.length; i++) {
539              cells = rows[i].getElementsByTagName('td');
540              var hide = false;
541
542              if(filterId != '' && !stringFilter(filterId,cells[0].innerText
                    ↪ ))
543                  hide = true;
544
545              if(startDate.value != '' && startDate.value > cells[1].
                    ↪ innerText)
546                  hide = true;
547
548              if(endDate.value != '' && endDate.value < cells[1].innerText)
549                  hide = true;
550
551              if(filterAccount != '' && !stringFilter(filterAccount, cells
                    ↪ [2].innerText))
552                  hide = true;
```

24

```
553
554                 if( filterType != '' && filterType != cells [3]. innerText )
555                     hide = true ;
556
557                 if( filterSecurity != '' && ! stringFilter ( filterSecurity , cells
                      ↪ [4]. innerText ))
558                     hide = true ;
559
560                 if( lowAmount != '' && Number ( lowAmount ) >
                      ↪ formattedStringToNumber ( cells [5]. innerText ))
561                     hide = true ;
562
563                 if( highAmount != '' && Number ( highAmount ) <
                      ↪ formattedStringToNumber ( cells [5]. innerText ))
564                     hide = true ;
565
566                 if( lowDAmount != '' && Number ( lowDAmount ) >
                      ↪ formattedStringToNumber ( cells [6]. innerText . substr (1)))
567                     hide = true ;
568
569                 if( highDAmount != '' && Number ( highDAmount ) <
                      ↪ formattedStringToNumber ( cells [6]. innerText . substr (1)))
570                     hide = true ;
571
572                 if( lowCostBasis != '' && Number ( lowCostBasis ) >
                      ↪ formattedStringToNumber ( cells [7]. innerText . substr (1)))
573                     hide = true ;
574
575                 if( highCostBasis != '' && Number ( highCostBasis ) <
                      ↪ formattedStringToNumber ( cells [7]. innerText . substr (1)))
576                     hide = true ;
577
578                 if( filterNA . checked && cells [7]. innerText == "N/A")
579                     hide = true ;
580
581                 if( hide )
582                     rows [i]. setAttribute ('hidden ', true );
583         }
584     }
585 }
586
587 function clearFilter () {
588     unfilterAll ();
589
590     var fields = document . getElementsByClassName ('filterField ');
591
592     for( var i = 0; i < fields . length ; i++) {
593         fields [i]. value = '';
594     }
595
596     document . getElementById ('filterNA '). checked = false ;
597 }
598
599 function unfilterAll () {
```

```
600        rows = document.getElementsByClassName('bodyRow');
601
602        for(var i = 0; i < rows.length; i++) {
603            rows[i].removeAttribute('hidden');
604        }
605  }
606
607  function toggleID() {
608        var button = document.getElementById('toggleId');
609        var rows = document.getElementsByTagName('tr');
610        var cells = rows[0].getElementsByTagName('th');
611
612        if(button.innerText == "Hide Transaction ID") {
613            button.innerText = "Show Transaction ID";
614
615            cells[0].setAttribute('hidden', true);
616            cells[0].classList = "";
617            cells[1].classList = "frozenColumn1";
618            //cells[2].classList = "frozenColumn2";
619            for(var i = 1; i < rows.length; i++) {
620                cells = rows[i].getElementsByTagName('td');
621
622                cells[0].setAttribute('hidden', true);
623                cells[0].classList = "idCell";
624                cells[1].classList = "frozenColumn1";
625                //cells[2].classList = "frozenColumn2";
626            }
627        }
628        else {
629            button.innerText = "Hide Transaction ID";
630
631            cells[0].removeAttribute('hidden');
632            cells[0].classList = "frozenColumn1";
633            cells[1].classList = "frozenColumn2";
634            //cells[2].classList = "frozenColumn3";
635            for(var i = 1; i < rows.length; i++) {
636                cells = rows[i].getElementsByTagName('td');
637
638                cells[0].removeAttribute('hidden');
639                cells[0].classList = "idCell frozenColumn1";
640                cells[1].classList = "frozenColumn2";
641                //cells[2].classList = "frozenColumn3";
642            }
643        }
644  }
645
646  function readFile(fileIn){
647        if(fileIn.files && fileIn.files[0]) {
648            var reader = new FileReader();
649            reader.onload = function (e) {
650                var output = e.target.result;
651                document.getElementById('typesArray').value = output;
652            };
653            reader.readAsText(fileIn.files[0]);
```

```
654          }
655  }
656
657  function saveFile() {
658      var element = document.createElement('a');
659      element.setAttribute('href', 'data:text/plain;charset=utf-8,' +
              ↪ encodeURIComponent(document.getElementById('typesArray').value));
660      element.setAttribute('download', 'transaction-types.csv');
661
662      element.style.display = 'none';
663      document.body.appendChild(element);
664
665      element.click();
666
667      document.body.removeChild(element);
668  }
669
670  function applyTypes() {
671      var typesArray = document.getElementById('typesArray').value.split
              ↪ (',');
672      setTransactionTypesList(typesArray);
673  }
674
675  function editTypes() {
676      document.getElementById('typesArray').value = readCurrentTypes().join
              ↪ (',');
677  }
678
679  function setTransactionTypesList(typesArray) {
680      var type = document.getElementById('type');
681      var filterType = document.getElementById('filterType');
682
683      type.innerHTML = '<option value=""></option>';
684      filterType.innerHTML = '<option value=""></option>';
685
686      for(var i = 0; i < typesArray.length; i++) {
687          var typeAsText = typesArray[i];
688          if(typesArray[i][0] == '!') typeAsText = typesArray[i].substr(1);
689
690          type.innerHTML += '<option value="' + typesArray[i] + '">' +
                  ↪ typeAsText + '</option>';
691          filterType.innerHTML += '<option value="' + typeAsText + '">' +
                  ↪ typeAsText + '</option>';
692      }
693  }
694
695  function toggleSection(button) {
696      var form = button.parentElement.parentElement.getElementsByTagName('
              ↪ form')[0];
697
698      if(button.innerText == "Hide") {
699          form.setAttribute("hidden",true);
700          button.innerText = "Show";
701      }
```

```
702       else {
703           form.removeAttribute("hidden");
704           button.innerText = "Hide";
705       }
706  }
707
708  function loadDataLists() {
709      var accountsList = document.getElementById("accountsList");
710      var securitiesList = document.getElementById("securitiesList");
711      var rows = document.getElementsByClassName("bodyRow");
712
713      var accounts = [];
714      var securities = [];
715
716      for(var i = 0; i < rows.length; i++) {
717          var tableAccount = rows[i].getElementsByTagName("td")[2].innerText
                 ↪ ;
718          var tableSecurity = rows[i].getElementsByTagName("td")[4].
                 ↪ innerText;
719
720          if(!accounts.includes(tableAccount)) accounts.push(tableAccount);
721          if(!securities.includes(tableSecurity)) securities.push(
                 ↪ tableSecurity);
722      }
723
724      accountsList.innerHTML = '';
725      securitiesList.innerHTML = '';
726
727      for(var i = 0; i < accounts.length; i++) {
728          accountsList.innerHTML += '<option value="' + accounts[i] + '"/>';
729      }
730
731      for(var i = 0; i < securities.length; i++) {
732          securitiesList.innerHTML += '<option value="' + securities[i] +
                 ↪ '"/>';
733      }
734  }
735
736  window.onload = function() {
737      resetDate();
738      initDb();
739  }
```

28

B.2 Google API Script

```
 1 var auth2;
 2
 3 var spreadsheetId = "1ROHpaAIUw-JHX8SrzvkEPCG1qgI-siJ9oucY6g5e4Co";
 4 var sheetId = "Sheet1";
 5 var sheetIdNum = 0;
 6
 7 function loadSheetData() {
 8     if(auth2.isSignedIn.get())
 9     {
10         getAllUserSheets();
11     }
12     else {
13         authenticate()
14             .then(function() {
15                 if(auth2.isSignedIn.get()) getAllUserSheets();
16             });
17     }
18 }
19
20 function getNewSheetData() {
21     spreadsheetId = document.getElementById('sheet').value;
22     if(auth2.isSignedIn.get())
23     {
24         getTabsOfSheet();
25     }
26     else {
27         authenticate()
28             .then(function() {
29                 if(auth2.isSignedIn.get()) getTabsOfSheet();
30             });
31     }
32 }
33
34 function populateSheetSelector(arrayOfSheets) {
35     document.getElementById('sheet').innerHTML = '<option value="1
        ↪ ROHpaAIUw-JHX8SrzvkEPCG1qgI-siJ9oucY6g5e4Co">default</option>';
36
37     for(var i = 0; i < arrayOfSheets.length; i++) {
38         document.getElementById('sheet').innerHTML += '<option value="' +
            ↪ arrayOfSheets[i].id + '">' + arrayOfSheets[i].name + '</
            ↪ option>';
39     }
40 }
41
42 function getNewTabData() {
43     data = document.getElementById('tab').value.split(/,(.+)/);
44     sheetIdNum = data[0];
45     sheetId = data[1];
46 }
47
48 function populateTabSelector(arrayOfTabs) {
49     document.getElementById('tab').innerHTML = '';
```

```
50
51      for(var i = 0; i < arrayOfTabs.length; i++) {
52          document.getElementById('tab').innerHTML += '<option value="' +
            ↪ arrayOfTabs[i].properties.sheetId + ',' + arrayOfTabs[i].
            ↪ properties.title + '">' + arrayOfTabs[i].properties.title +
            ↪ '</option>';
53      }
54  }
55
56  function getAllUserSheets() {
57      return gapi.client.drive.files.list({
58          "pageSize": 1000,
59          "orderBy": "name",
60          "q": "mimeType = 'application/vnd.google-apps.spreadsheet'",
61      })
62          .then(function(response) {
63              populateSheetSelector(JSON.parse(response.body).files);
64              getNewSheetData();
65              console.log("Response", response);
66          },
67          function(err) { console.error("Execute error", err); });
68  }
69
70  function getTabsOfSheet() {
71      return gapi.client.sheets.spreadsheets.get({
72        "spreadsheetId": spreadsheetId,
73        "includeGridData": false
74      })
75          .then(function(response) {
76              populateTabSelector(JSON.parse(response.body).sheets);
77              getNewTabData();
78              console.log("Response", response);
79          },
80          function(err) { console.error("Execute error", err); });
81  }
82
83  function tableToArrays() {
84      var rows = document.getElementsByClassName('bodyRow');
85      var data = new Array();
86      data.push(["Transaction Id", "Date", "Account Number", "Transaction
            ↪ Type", "Security", "Amount", "$ Amount", "Cost Basis", "Files"])
            ↪ ;
87
88      for(var i = 0; i < rows.length; i++) {
89          var cells = rows[i].getElementsByTagName('td');
90          var cellData = new Array();
91
92          for(var j = 0; j < 8; j++) {
93              cellData.push(cells[j].innerText);
94          }
95          cellData.push(getFileNamesIds(cells[8]));
96          data.push(cellData);
97      }
98
```

```
 99        console.log(data);
100        return data;
101    }
102
103    function arraysToTable(dataArr) {
104        while(document.getElementsByClassName('bodyRow').length > 0) {
105            document.getElementById("tableBody").removeChild(document.
                   ↪ getElementsByClassName('bodyRow')[0]);
106        }
107
108        document.getElementById('add').removeAttribute('hidden');
109        document.getElementById('save').setAttribute('hidden', true);
110        document.getElementById('discard').setAttribute('hidden', true);
111
112        document.getElementById('add').setAttribute('type', 'submit');
113        document.getElementById('save').setAttribute('type', 'button');
114
115        while(dataArr.length > 0) {
116            let data = dataArr[dataArr.length - 1];
117            let files = parseFileNamesIds(data[8]);
118            data.pop();
119            if(files.length > 0) {
120                data.push(files);
121            }
122            addTransaction(data);
123            dataArr.pop();
124        }
125        loadDataLists();
126    }
127
128    function authenticate() {
129    return gapi.auth2.getAuthInstance()
130        .signIn({scope: "https://www.googleapis.com/auth/drive"})
131        .then(function() { console.log("Sign-in successful"); },
132            function(err) { console.error("Error signing in", err); });
133    }
134
135    function loadClientSheets() {
136    gapi.client.setApiKey("AIzaSyDC6JNuMW78Q-gWsp0PFEaTsICYjHWymAo");
137    return gapi.client.load("https://content.googleapis.com/discovery/v1/apis/
           ↪ sheets/v4/rest")
138        .then(function() { console.log("GAPI client loaded for API");
               ↪ loadSheetData(); },
139            function(err) { console.error("Error loading GAPI client for API",
                   ↪  err); });
140    }
141
142    function loadClient() {
143    gapi.client.setApiKey("AIzaSyDC6JNuMW78Q-gWsp0PFEaTsICYjHWymAo");
144    return gapi.client.load("https://content.googleapis.com/discovery/v1/apis/
           ↪ drive/v3/rest")
145        .then(function() { console.log("GAPI client loaded for API");
               ↪ loadClientSheets(); },
```

```
146          function(err) { console.error("Error loading GAPI client for API",
               ↪   err); });
147  }
148
149  function readGoogleSheetDB() {
150      clearIndexedDb("sheets");
151      return gapi.client.sheets.spreadsheets.values.get({
152          "spreadsheetId": spreadsheetId,
153          "range": sheetId + "!A2:I214748354"
154      })
155          .then(function(response) {
156              console.log("Response", response);
157
158              dataArr = [];
159              if(JSON.parse(response.body).values != undefined) {
160                  dataArr = JSON.parse(response.body).values;
161              }
162              arraysToTable(dataArr);
163
164              readGoogleTypes();
165          },
166          function(err) { console.error("Execute error", err); });
167  }
168
169  function readGoogleTypes() {
170      return gapi.client.sheets.spreadsheets.values.get({
171          "spreadsheetId": spreadsheetId,
172          "range": sheetId + "!J1:J214748354",
173          "majorDimension": "COLUMNS"
174
175      })
176          .then(function(response) {
177              console.log("Response", response);
178              var typesArr = JSON.parse(response.body).values[0];
179              setTransactionTypesList(typesArr);
180          },
181          function(err) { console.error("Execute error", err); });
182  }
183
184  function readCurrentTypes() {
185      var types = document.getElementById('type').getElementsByTagName('
               ↪ option');
186      var currentTypes = [];
187
188      for(var i = 1; i < types.length; i++) {
189          currentTypes.push(types[i].value);
190      }
191
192      return currentTypes;
193  }
194
195  function writeGoogleSheetDB() {
196      if(auth2.isSignedIn.get())
197      {
```

```
198              setGoogleRows ()
199          }
200          else {
201              authenticate ()
202                  . then ( function () {
203                      if ( auth2 . isSignedIn . get ()) setGoogleRows ();
204                  });
205          }
206  }
207
208  function setGoogleRows () {
209      return gapi . client . sheets . spreadsheets . batchUpdate ({
210          " spreadsheetId ": spreadsheetId ,
211          " resource ": {
212          " requests ": [
213              {
214              " updateSheetProperties ": {
215                  " properties ": {
216                      " gridProperties ": {
217                          " columnCount ": 10 ,
218                          " rowCount ": 1
219                  },
220                      " sheetId ": sheetIdNum
221                  },
222                      " fields ": " gridProperties "
223              }
224              }
225          ]
226          }
227      })
228          . then ( function ( response ) {
229              console . log (" Response ", response );
230
231              clearGoogleRow ();
232          },
233          function ( err ) { console . error (" Execute error ", err ); });
234  }
235
236  function clearGoogleRow () {
237      return gapi . client . sheets . spreadsheets . values . clear ({
238      " spreadsheetId ": spreadsheetId ,
239      " range ": sheetId + "! A1 :J1 ",
240      " resource ": {}
241      })
242          . then ( function ( response ) {
243              console . log (" Response ", response );
244
245              writeGoogleDB ();
246          },
247          function ( err ) { console . error (" Execute error ", err ); });
248  }
249
250  function writeGoogleDB () {
251      writeImagesToFirestore (" sheets ");
```

```
252    return gapi.client.sheets.spreadsheets.values.batchUpdate({
253        "spreadsheetId": spreadsheetId,
254        "resource": {
255            "data": [
256            {
257                "range": sheetId + "!A1",
258                "values": tableToArrays(),
259                "majorDimension": "ROWS"
260            },
261            {
262                "range": sheetId + "!J1",
263                "values": [readCurrentTypes()],
264                "majorDimension": "COLUMNS"
265            }
266            ],
267            "valueInputOption": "RAW"
268        }
269        })
270        .then(function(response) {
271            console.log("Response", response);
272        },
273        function(err) { console.error("Execute error", err); });
274 }
275
276 gapi.load("client:auth2", function() {
277     auth2 = gapi.auth2.init({client_id: "217251662395-9
          ↪ pu2qa1hubgrblav1nhvnfaascd6povv.apps.googleusercontent.com"});
278     loadClient();
279 });
```

B.3 Firebase Script

```
1  function clearFirebase() {
2      firebase.database().ref('Data').remove();
3  }
4
5  function writeToFirebase() {
6      writeImagesToFirestore("firebase");
7      clearFirebase();
8
9      var data = tableToArrays();
10     var typesArr = readCurrentTypes();
11
12     for(var i = 1; i < data.length; i++) {
13         firebase.database().ref('Data/' + String(i - 1)).set({
14             id: data[i][0],
15             date: data[i][1],
16             account: data[i][2],
17             type: data[i][3],
18             security: data[i][4],
19             amount: data[i][5],
20             dAmount: data[i][6],
21             costBasis: data[i][7],
22             files: data[i][8]
23         });
24     }
25     for(var i = 0; i < typesArr.length; i++) {
26         firebase.database().ref('Types/' + String(i)).set({
27             value: typesArr[i]
28         });
29     }
30 }
31
32 function readFromFirebase() {
33     clearIndexedDb("firebase");
34     return firebase.database().ref('/').once('value').then(function(
         ↪ snapshot) {
35
36         while(document.getElementsByClassName('bodyRow').length > 0) {
37             document.getElementById("tableBody").removeChild(document.
                 ↪ getElementsByClassName('bodyRow')[0]);
38         }
39
40         data = snapshot.val().Data;
41         console.log(data);
42         for(var i = data.length - 1; i >= 0; i--) {
43             let staged = [data[i].id, data[i].date, data[i].account, data[
                 ↪ i].type, data[i].security, data[i].amount, data[i].
                 ↪ dAmount, data[i].costBasis];
44             if(parseFileNamesIds(data[i].files).length > 0) {
45                 staged.push(parseFileNamesIds(data[i].files));
46             }
47             console.log(staged);
48             addTransaction(staged);
```

```
 49            }
 50
 51            types = snapshot.val().Types;
 52            var typesArr = [];
 53            for(var i = 0; i < types.length; i++) {
 54                typesArr.push(types[i].value);
 55            }
 56            setTransactionTypesList(typesArr);
 57            loadDataLists();
 58        });
 59    }
 60
 61    function clearFirestore() {
 62        firestore.collection("Data").get().then((querySnapshot) => {
 63            querySnapshot.forEach((doc) => {
 64                firestore.collection("Data").doc(doc.id).delete();
 65            });
 66        }).then(function() {
 67            firestore.collection("Types").get().then((querySnapshot) => {
 68                querySnapshot.forEach((doc) => {
 69                    firestore.collection("Types").doc(doc.id).delete();
 70                });
 71            })
 72        }).then(function() { return 0 });
 73    }
 74
 75    function writeToFirestore() {
 76        writeImagesToFirestore("firestore");
 77        firestore.collection("Data").get().then((querySnapshot) => {
 78            querySnapshot.forEach((doc) => {
 79                firestore.collection("Data").doc(doc.id).delete();
 80            });
 81        }).then(function() {
 82            firestore.collection("Types").get().then((querySnapshot) => {
 83                querySnapshot.forEach((doc) => {
 84                    firestore.collection("Types").doc(doc.id).delete();
 85                });
 86        }).then(function() {
 87                var data = tableToArrays();
 88                var typesArr = readCurrentTypes();
 89
 90                for(var i = 1; i < data.length; i++) {
 91                    firestore.collection("Data").add({
 92                        id: data[i][0],
 93                        date: data[i][1],
 94                        account: data[i][2],
 95                        type: data[i][3],
 96                        security: data[i][4],
 97                        amount: data[i][5],
 98                        dAmount: data[i][6],
 99                        costBasis: data[i][7],
100                        files: data[i][8],
101                        index: i - 1
102                    })
```

```
103                    .then(function(docRef) {
104                        console.log("Document written with ID: ", docRef.id);
105                    })
106                    .catch(function(error) {
107                        console.error("Error adding document: ", error);
108                    });
109            }
110            for(var i = 0; i < typesArr.length; i++) {
111                firestore.collection("Types").add({
112                    value: typesArr[i],
113                    index: i
114                })
115                    .then(function(docRef) {
116                        console.log("Document written with ID: ", docRef.id);
117                    })
118                    .catch(function(error) {
119                        console.error("Error adding document: ", error);
120                    });
121            }
122        })
123    });
124 }
125
126 function readFromFirestore() {
127     clearIndexedDb("firestore");
128     firestore.collection("Data").get().then((querySnapshot) => {
129         var data = new Array();
130
131         querySnapshot.forEach((doc) => {
132             data[doc.data().index] = doc.data();
133             console.log(data);
134         });
135
136         while(document.getElementsByClassName('bodyRow').length > 0) {
137             document.getElementById("tableBody").removeChild(document.
                ↪ getElementsByClassName('bodyRow')[0]);
138         }
139
140         for(let i = data.length - 1; i >= 0; i--) {
141             let staged = [data[i].id, data[i].date, data[i].account, data[
                ↪ i].type, data[i].security, data[i].amount, data[i].
                ↪ dAmount, data[i].costBasis];
142             if(parseFileNamesIds(data[i].files).length > 0) {
143                 staged.push(parseFileNamesIds(data[i].files));
144             }
145             console.log(staged);
146             addTransaction(staged);
147         }
148         loadDataLists();
149     });
150
151     firestore.collection("Types").get().then((querySnapshot) => {
152         var typesArr = [];
153
```

```
154        querySnapshot.forEach((doc) => {
155            typesArr[doc.data().index] = doc.data().value;
156        });
157
158        setTransactionTypesList(typesArr);
159    });
160 }
```

B.4 MySQL Script

```
 1  function writeToMySQL () {
 2      writeImagesToFirestore ("mysql");
 3      var data = tableToArrays ();
 4      var types = readCurrentTypes ();
 5
 6      fetch('http://localhost:5000/api', {
 7          method: 'POST',
 8          headers: {
 9              'Content-Type': 'application/json',
10          },
11          body: JSON.stringify([data, types]),
12      });
13  }
14
15  function readFromMySQL () {
16      clearIndexedDb ("mysql");
17      fetch('http://localhost:5000/api')
18          .then(response => {
19              return response.json()
20          })
21          .then(fullresponse => {
22              console.log(fullresponse);
23
24              while(document.getElementsByClassName('bodyRow').length > 0) {
25                  document.getElementById("tableBody").removeChild(document.
                        ↪ getElementsByClassName('bodyRow')[0]);
26              }
27
28              var data = fullresponse [0];
29              for(var i = data.length - 1; i >= 0; i--) {
30                  let staged = [data[i].id, data[i].date, data[i].account,
                        ↪ data[i].type, data[i].security, data[i].amount, data
                        ↪ [i].dAmount, data[i].costBasis];
31                  if(parseFileNamesIds(data[i].files).length > 0) {
32                      staged.push(parseFileNamesIds(data[i].files));
33                  }
34                  addTransaction(staged);
35              }
36              loadDataLists ();
37
38              var types = fullresponse [1];
39              var typesArr = [];
40              for(var i = 0; i < types.length; i++) {
41                  typesArr.push(types[i].typename);
42              }
43              setTransactionTypesList(typesArr);
44          })
45  }
```

B.5 Local Storage Script

```
1   let db;
2   let dbVersion = 1;
3   let dbReady = false;
4
5   var fileEditted = false;
6
7   function initDb() {
8       let reset = indexedDB.deleteDatabase('FileStorage');
9       reset.onsuccess = function(a) {
10          let request = indexedDB.open('FileStorage', dbVersion);
11
12          request.onerror = function(e) {
13              console.error('Unable to open database.');
14          }
15
16          request.onsuccess = function(e) {
17              db = e.target.result;
18              console.log('db opened');
19          }
20
21          request.onupgradeneeded = function(e) {
22              let db = e.target.result;
23              db.createObjectStore('files', {keyPath:'id', autoIncrement:
                  ↪ false});
24              dbReady = true;
25          }
26      }
27  }
28
29  function fileUploadChanged() {
30      fileIn = document.getElementById('fileUpload');
31      if(fileIn.files && fileIn.files[0]) {
32          document.getElementById('fileUploadLabel').innerHTML = String(
              ↪ fileIn.files.length) + " file(s)";
33      }
34      fileEditted = true;
35      console.log("updated file upload");
36  }
37
38  function uploadFile(data, cb, fileList, index) {
39      fileId = fileIdGenerator();
40      fileIn = document.getElementById('fileUpload');
41      if(fileIn.files && fileIn.files[index]) {
42          var reader = new FileReader();
43          reader.onload = function (e) {
44              console.log(e.target.result);
45
46              let bits = btoa(e.target.result);
47              let ob = {
48                  id: fileId,
49                  type: fileIn.files[index].type,
50                  name: fileIn.files[index].name,
```

```
51                    data: bits
52                };
53
54            let trans = db.transaction(['files'], 'readwrite');
55            let addReq = trans.objectStore('files').put(ob);
56
57            addReq.onerror = function(e) {
58                console.log('error storing data');
59                console.error(e);
60            }
61
62            trans.oncomplete = function(e) {
63                console.log('data stored');
64                fileList.push([fileId, fileIn.files[index].name]);
65                index++;
66                uploadFile(data, cb, fileList, index);
67            }
68        };
69        reader.readAsBinaryString(fileIn.files[index])
70    }
71    else {
72        removeFileUpload();
73        data.push(fileList);
74        cb(data);
75    }
76 }
77
78 function addFile(fileIn, index) {
79    fileId = fileIdGenerator();
80    if(fileIn.files && fileIn.files[index]) {
81        var reader = new FileReader();
82        reader.onload = function (e) {
83            console.log(e.target.result);
84
85            let bits = btoa(e.target.result);
86            let ob = {
87                id: fileId,
88                type: fileIn.files[index].type,
89                name: fileIn.files[index].name,
90                data: bits
91            };
92
93            let trans = db.transaction(['files'], 'readwrite');
94            let addReq = trans.objectStore('files').put(ob);
95
96            addReq.onerror = function(e) {
97                console.log('error storing data');
98                console.error(e);
99            }
100
101            trans.oncomplete = function(e) {
102                console.log('data stored');
103                let table = fileIn.parentElement.getElementsByTagName('
                   ↪ tbody')[0];
```

```
104                     let newRowContent = "<tr><td><a onclick='downloadFile('" +
                        ↪   fileId + "');' href='javascript:void(0);'>" +
                        ↪   fileIn.files[index].name + "</a></td>";
105                     newRowContent += "<td><button type='button' onclick='
                        ↪   removeFileFromTable('" + fileId + "', this);'>-</
                        ↪   button></td></tr>";
106                     table.innerHTML += newRowContent;
107                     index++;
108                     addFile(fileIn, index);
109                 }
110             };
111         reader.readAsBinaryString(fileIn.files[index])
112     }
113     else {
114         fileIn.value = null;
115     }
116 }
117
118 function updateExistingFileName(data) {
119     for(let i = 0; i < data[0].getElementsByTagName('tr').length; i++) {
120         let fileId = data[0].getElementsByTagName('tr')[i].
              ↪   getElementsByTagName('a')[0].getAttribute('onclick').split
              ↪   ('''')[1];
121         deleteFileFromIndexedDB(fileId);
122     }
123
124     var fileContent = '<table><tbody>';
125     if(data.length > 1) {
126         for(let i = 0; i < data[1].length; i++) {
127             fileContent += "<tr><td><a onclick='downloadFile('" + data[1][
                  ↪   i][0] + "');' href='javascript:void(0);'>" + data[1][i
                  ↪   ][1] + "</a></td>";
128             fileContent += "<td><button type='button' onclick='
                  ↪   removeFileFromTable('" + data[1][i][0] + "', this);'>-</
                  ↪   button></td></tr>";
129         }
130     }
131     fileContent += '</tbody></table><input type="file" onchange="addFile(
          ↪   this, 0);" multiple/>';
132     data[0].innerHTML = fileContent;
133 }
134
135 function removeFileFromTable(fileId, cell) {
136     var row = cell.parentElement.parentElement;
137
138     if(confirm("Delete " + row.getElementsByTagName('a')[0].innerText +
          ↪   "?")) {
139         row.parentElement.removeChild(row);
140         deleteFileFromIndexedDB(fileId);
141     }
142 }
143
144 function deleteFileFromIndexedDB(fileId) {
145     let trans = db.transaction(['files'], 'readwrite');
```

```
146        let addReq = trans.objectStore('files').delete(fileId);
147  }
148
149  function removeFileUpload() {
150        document.getElementById('fileUpload').value = null;
151        document.getElementById('fileUploadLabel').innerHTML = "Upload file";
152        fileEditted = true;
153  }
154
155  function downloadFile(fileId) {
156        console.log('downloading');
157        var trans = db.transaction(['files'], 'readonly');
158        var dlReq = trans.objectStore('files').get(fileId);
159
160        dlReq.onerror = function(e) {
161            console.log('error reading data');
162            console.error(e);
163        };
164
165        dlReq.onsuccess = function(e) {
166            console.log('data read');
167            console.log(dlReq.result);
168            var element = document.createElement('a');
169            element.setAttribute('href', 'data:' + dlReq.result.type + ';
                ↪ base64,' + dlReq.result.data);
170            element.setAttribute('download', dlReq.result.name);
171
172            element.style.display = 'none';
173            document.body.appendChild(element);
174
175            element.click();
176
177            document.body.removeChild(element);
178        };
179  }
180
181  window.onbeforeunload = function(){
182        indexedDB.deleteDatabase('FileStorage');
183  }
```

B.6 Firestore Images Script

```
 1  function writeImagesToFirestore(database) {
 2      firestore.collection("Images:" + database).get().then((querySnapshot)
            ↪ => {
 3          querySnapshot.forEach((doc) => {
 4              firestore.collection("Images:" + database).doc(doc.id).delete
                  ↪ ();
 5          });
 6      }).then(function() {
 7          var trans = db.transaction(['files'], 'readonly');
 8          var dlReq = trans.objectStore('files').getAll();
 9
10          dlReq.onerror = function(e) {
11              console.log('error reading data');
12              console.error(e);
13          };
14
15          dlReq.onsuccess = function(e) {
16              console.log(dlReq.result);
17
18              for(let i = 0; i < dlReq.result.length; i++) {
19                  firestore.collection("Images:" + database).add({
20                      id: dlReq.result[i].id,
21                      type: dlReq.result[i].type,
22                      name: dlReq.result[i].name,
23                      data: dlReq.result[i].data,
24                  })
25                  .then(function(docRef) {
26                      console.log("Image written with ID: ", docRef.id);
27                  })
28                  .catch(function(error) {
29                      console.log("Error adding image: ", error);
30                  });
31              }
32          };
33      });
34  }
35
36  function readImagesFromFirestore(database) {
37      firestore.collection("Images:" + database).get().then((querySnapshot)
            ↪ => {
38          let trans = db.transaction(['files'], 'readwrite');
39
40          trans.oncomplete = function(e) {
41              console.log('data stored');
42          }
43
44          querySnapshot.forEach((doc) => {
45              console.log("Writing ", doc.data().name);
46
47              let ob = {
48                  id: doc.data().id,
49                  type: doc.data().type,
```

44

```
50              name: doc.data().name,
51              data: doc.data().data
52          };
53
54          let addReq = trans.objectStore('files').put(ob);
55      });
56  });
57  }
58
59  function getFileNamesIds(cell) {
60      var links = cell.getElementsByTagName('a');
61      var result = '';
62
63      for(let i = 0; i < links.length; i++) {
64          result += links[i].innerText + '/' + links[i].getAttribute('
             ↪ onclick').split('''')[1] + '/';
65      }
66
67      return result;
68  }
69
70  function parseFileNamesIds(string) {
71      var result = new Array();
72
73      if(string != '') {
74          let strarr = string.split('/');
75          for(let i = 0; i < strarr.length - 1; i += 2) {
76              result.push([strarr[i + 1], strarr[i + 0]]);
77          }
78      }
79
80      return result;
81  }
82
83  function clearIndexedDb(database) {
84      console.log("db reset");
85      let trans = db.transaction(['files'], 'readwrite');
86      var clearReq = trans.objectStore('files').clear();
87
88      trans.oncomplete = function(e) {
89          readImagesFromFirestore(database);
90      }
91  }
```

C CSS Source Code

```
 1  body {
 2      font-size: 14px;
 3  }
 4
 5  input,
 6  select {
 7      min-width: 100px;
 8      width: 80%;
 9  }
10
11  button {
12      width: 40%;
13  }
14
15  span {
16      border: solid;
17      border-width: 1px;
18  }
19
20  #fileUpload {
21      display: none;
22  }
23
24  #removeFile {
25      width: 16px;
26      padding: 0;
27      border: 0;
28  }
29
30  .sectionHead {
31  }
32
33      .toggleSection {
34          width: 100px;
35      }
36
37      .sectionHead > * {
38          display: inline-block;
39      }
40
41  #table {
42      max-height: 50vh;
43      overflow: auto;
44  }
45
46  #inputFields,
47  #filter {
48      padding: 10px 0;
49      overflow-x: auto;
50  }
51
52  #inputFields > form,
```

```css
53  #filter > form {
54      min-width: 1800px;
55  }
56
57      #inputFields > form > section {
58          width: 14%;
59          display: inline-block;
60      }
61
62      #filter > form > section {
63          width: 10%;
64          display: inline-block;
65      }
66
67          #filter > form > section > button {
68              width: 100%;
69          }
70
71  #filterNA {
72      display: inline-block;
73      width: 50%;
74      min-width: 16px;
75  }
76
77  #options > form > section {
78      display: inline-block;
79      width: 300px;
80      margin: 15px;
81  }
82
83      #toggleId,
84      #fileInput {
85          width: 80%;
86      }
87
88      #transactionTypesConfig {
89          width: 500px !important;
90      }
91
92      #transactionTypesConfig > button {
93          display: inline-block;
94          width: 30%;
95          padding: 0;
96      }
97
98      #typesArray {
99          width: 40%;
100     }
101
102     #options > form > #googleSheetButtons {
103         width: 200px;
104     }
105
106     #logoutSection#logoutSection {
```

```
107         width: 100px;
108     }
109
110     #googleSheetButtons > button,
111     #logoutSection > button {
112         width: 100%;
113         padding: 0;
114         margin: 0;
115     }
116
117 table {
118     width: 100%;
119     margin: auto;
120     border-collapse: collapse;
121 }
122     th {
123         min-width: 200px;
124         width: 10%;
125         position: sticky;
126         top: 0;
127     }
128
129     th > section {
130         width: 80%;
131         display: inline-block;
132         padding: 0;
133         margin: 0;
134     }
135
136     th,
137     td {
138         z-index: 0;
139         background: white;
140     }
141
142     .frozenColumn1,
143     .frozenColumn2,
144     .frozenColumn3 {
145         position: sticky;
146         padding: 0;
147         min-width: 200px;
148     }
149
150     td.frozenColumn1,
151     td.frozenColumn2,
152     td.frozenColumn3 {
153         z-index: 1;
154     }
155
156     th.frozenColumn1,
157     th.frozenColumn2,
158     th.frozenColumn3 {
159         z-index: 2;
160     }
```

```
161
162        .frozenColumn1 {
163            left: 0;
164        }
165
166        .frozenColumn2 {
167            left: 200px;
168        }
169
170        .frozenColumn3 {
171            left: 400px;
172        }
173
174        .sort {
175            width: 10%;
176        }
177
178        .sort > button {
179            padding: 0;
180            border: 0;
181            display: block;
182            width: 100%;
183        }
184
185        .editing > td {
186            background: yellow;
187        }
188
189        #table,
190        table,
191        td,
192        th {
193            box-shadow: 1px 1px black, inset 1px 1px black;
194        }
```

D   Server-side NodeJs Code

```
1  const functions = require('firebase-functions');
2  const express = require('express');
3  const mysql = require('mysql');
4  const cors = require('cors');
5  const api = express();
6  api.use(cors({ origin: true }));
7
8  var con = mysql.createConnection({
9      host: "localhost",
10     user: "user",
11     password: "pass",
12     database: "mydb"
13 });
14
15 con.connect(function(err) {
16     if (err) throw err;
17     console.log("Connected to MySQL Database!");
18 });
19
20 function readDataFromMySQL(cb) {
21     con.query("SELECT * FROM data;", function (err, data, fields) {
22         if (err) throw err;
23         cb(data);
24     });
25 }
26
27 function readTypesFromMySQL(cb) {
28     con.query("SELECT * FROM types;", function (err, types, fields) {
29         if (err) throw err;
30         cb(types);
31     });
32 }
33
34 function writeToMySQL(jsonData) {
35     var data = jsonData[0];
36     var types = jsonData[1];
37
38     con.query("TRUNCATE TABLE data;");
39
40     for(var i = 1; i < data.length; i++) {
41         con.query("INSERT INTO data (id, date, account, type, security,
                ↪ amount, dAmount, costBasis, files) VALUES ('" + data[i].join
                ↪ ("','") + "');");
42     }
43
44     con.query("TRUNCATE TABLE types;");
45
46     for(var i = 0; i < types.length; i++) {
47         con.query("INSERT INTO types (typename) VALUES ('" + types[i] +
                ↪ "');");
48     }
49 }
```

```
50
51 api.post('/api', (req, res) => {
52     console.log(req.body);
53     writeToMySQL(req.body);
54     return res.status(200).send(req.body);
55 });
56
57 api.get('/api', (req, res) => {
58     readDataFromMySQL(function(data) {
59         console.log(data);
60         readTypesFromMySQL(function(types) {
61             console.log(types);
62             return res.status(200).send([ data, types ]);
63         });
64     });
65 });
66
67 exports.api = functions.https.onRequest(api);
```