Documentation for Financial Transactions HTML Page

Jason N.

June 1, 2020

Contents

1 HTML

1.1 Preamble and head

1.2 Inputs

1.2.1 Common attributes

1.2.2 Labels

1.2.3 Date

1.2.4 Text

1.2.5 List

1.2.6 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

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "UTF-8"/>
        <link rel="stylesheet" type="text/css" href="./style.css"/>

        <script src="./script.js"></script>
        <script src="https://apis.google.com/js/api.js"></script>
        <script src="./googleApiScript.js"></script>
        <script src="./mysqlScript.js"></script>
        <script src="./localStorageScript.js"></script>
        <script src="./imageFirestore.js"></script>
    </head>
    <body>
        <article id="inputFields">
            <section class="sectionHead">
                <h1>Input:</h1>
                <button class="toggleSection" type="button" onclick="toggleSection(this)">Hide</button>
            </section>
            <form onsubmit="return false" autocomplete="off">
                <section>
                    <label for="date">Date:</label><br/>
                    <input id="date" name="date" type="date" placeholder="yyyy-mm-dd"/>
                </section>

                <section>
                    <label for="account">Account Number:</label><br/>
                    <input id="account" name="account" list="accountsList" type="text" placeholder="Account Number"/>
                    <datalist id="accountsList">
                    </datalist>
                </section>

                <section>
                    <label for="type">Transaction Type:</label><br/>
                    <select id="type" name="type">
                        <option value=""></option>
                        <option value="BUY">BUY</option>
                        <option value="SELL">SELL</option>
                        <option value="!DIVIDEND">DIVIDEND</option>
                        <option value="!INTEREST">INTEREST</option>
                        <option value="!WITHDRAW">WITHDRAW</option>
                        <option value="!DEPOSIT">DEPOSIT</option>
                    </select>
                </section>

                <section>
                    <label for="security">Security:</label><br/>
                    <input id="security" name="security" list="securitiesList" type="text" placeholder="
```

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

                    <br/>

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

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

                <section>
                    <label for="type">Transaction Type:</label><br/>
                    <select id="filterType" class="filterField" name="filterType">
                        <option value=""></option>
                        <option value="BUY">BUY</option>
                        <option value="SELL">SELL</option>
                        <option value="DIVIDEND">DIVIDEND</option>
                        <option value="INTEREST">INTEREST</option>
                        <option value="WITHDRAW">WITHDRAW</option>
                        <option value="DEPOSIT">DEPOSIT</option>
                    </select>
                </section>

                <section>
                    <label for="filterSecurity">Security:</label>
                    <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/>
                    <input id="filterSecurity" class="filterField" name="filterSecurity" type="text" placeholder="Security"/>
                </section>

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

                    <br/>

                    <label for="highAmount">Max Amount:</label><br/>
                    <input id="highAmount" class="filterField" name="highAmount" type="text" placeholder="Max Amount"/>
                </section>

                <section>
                    <label for="lowDAmount">Min $ Amount:</label><br/>
                    <input id="lowDAmount" class="filterField" name="lowDAmount" type="text" placeholder="Min $ Amount"/>

                    <br/>

                    <label for="highDAmount">Max $ Amount:</label><br/>
                    <input id="highDAmount" class="filterField" name="highDAmount" type="text" placeholder="Max $ Amount"/>
                </section>

                <section>
                    <label for="lowCostBasis">Min Cost Basis:</label><br/>
                    <input id="lowCostBasis" class="filterField" name="lowCostBasis" type="text" placeholder="Min Cost Basis"/>

                    <br/>

                    <label for="highCostBasis">Max Cost Basis:</label><br/>
                    <input id="highCostBasis" class="filterField" name="highCostBasis" type="text" placeholder="Max Cost Basis"/>

                    <br/>

                    <label for="filterNa">Filter N/A:</label>
                    <input id="filterNA" name="filterNA" type="checkbox"/>
                </section>

                <section>
                    <button type="submit" onclick="applyFilter()">Apply</button>
                    <button type="button" onclick="clearFilter()">Clear</button>
                </section>
            </form>
        </article>

        <article id="options">
            <section class="sectionHead">
                <h1>Options:</h1>
```

```html
                <button class="toggleSection" type="button" onclick="toggleSection(this)">Hide</button>
            </section>
            <form onsubmit="return false" autocomplete="off">
                <section>
                    <button id="toggleId" type="button" onclick="toggleID()">Hide Transaction ID</button>
                </section>

                <section>
                    <label for="fileInput">Import Transaction Type CSV file</label>
                    <input id="fileInput" name="fileInput" type="file" onchange="readFile(this)"/>
                </section>

                <section id="transactionTypesConfig">
                        <label for="typesArray">Transaction types (comma-separated):</label>
                        <input id="typesArray" name="typesArray" type="text" placeholder="Transaction types"/>
                        <button id="applyTypesButton" type="button" onclick="applyTypes()">Apply</button>
                        <button id="editTypesButton" type="button" onclick="editTypes()">Edit</button>
                        <button type="button" onclick="saveFile()">Save</button>
                </section>

                <section id="googleSheetButtons">
                    <button type="button" onclick="writeGoogleSheetDB()">Write to Sheets</button>
                    <button type="button" onclick="readGoogleSheetDB()">Read from Sheets</button>
                </section>

                <section>
                    <section>
                        <select id="sheet" onchange="getNewSheetData()">
                            <option value="1R0HpaAIUw-JHX8SrzvkEPCG1qgI-siJ9oucY6g5e4Co">default</option>
                        </select>
                    </section>

                    <section>
                        <select id="tab" onchange="getNewTabData()">
                        </select>
                    </section>
                </section>

                <section id="logoutSection">
                    <button type="button" onclick="loadSheetData()">Reload Sheets</button>
                    <button type="button" onclick="auth2.disconnect()">Log out</button>
                </section>

                <section>
                    <button type="button" onclick="writeToFirebase()"> Write to Firebase</button>
                    <button type="button" onclick="readFromFirebase()"> Read from Firebase</button>
                </section>

                <section>
                    <button type="button" onclick="writeToFirestore()"> Write to Firestore</button>
                    <button type="button" onclick="readFromFirestore()"> Read from Firestore</button>
                </section>

                <section>
                    <button type="button" onclick="writeToMySQL()">Write to MySQL</button>
                    <button type="button" onclick="readFromMySQL()">Read from MySQL</button>
                </section>
            </form>
        </article>

        <article>
        <h1>Table:</h1>
        <article id="table">
            <table>
                <thead>
                    <tr>
                        <th class="frozenColumn1">
                            <section>
                                Transaction ID
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable(0, true)">^</button>
                                <button type="button" onclick="sortTable(0, false)">v</button>
                            </section>
                        </th>
                        <th class="frozenColumn2">
                            <section>
                                Date
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable(1, true)">^</button>
                                <button type="button" onclick="sortTable(1, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Account Number
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (2, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (2, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Transaction Type
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (3, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (3, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Security
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (4, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (4, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Amount
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (5, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (5, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                $ Amount
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (6, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (6, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Cost Basis
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (7, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (7, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                File
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                   ↪ (8, true)">^</button>
                                <button type="button" onclick="sortTable
                                   ↪ (8, false)">v</button>
                            </section>
                        </th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody id="tableBody">
                </tbody>
            </table>
        </article>
        </article>

        <!-- The core Firebase JS SDK is always required and must be
           ↪ listed first -->
        <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
           ↪ app.js"></script>

        <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
           ↪ analytics.js"></script>
        <script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-
           ↪ database.js"></script>
        <script src="https://www.gstatic.com/firebasejs/7.14.3/firebase-
           ↪ firestore.js"></script>

        <script>
        // Your web app's Firebase configuration
        var firebaseConfig = {
            apiKey: "AIzaSyAmZLFZHDAB9evhvNunxOe5GxXRd_OizmU",
            authDomain: "financial-transactions-6f065.firebaseapp.com",
            databaseURL: "https://financial-transactions-6f065.firebaseio.
               ↪ com",
            projectId: "financial-transactions-6f065",
            storageBucket: "financial-transactions-6f065.appspot.com",
            messagingSenderId: "82206982479",
            appId: "1:82206982479:web:8937bbd1bd4fb6022b053a",
            measurementId: "G-0564DT8RNQ"
        };
        // Initialize Firebase
        firebase.initializeApp(firebaseConfig);
        firebase.analytics();

        var database = firebase.database();
        var firestore = firebase.firestore();
        </script>

        <script src="./firebaseScript.js"></script>
    </body>
</html>
```

B Javascript Source Code

B.1 Main Script

```
function formattedStringToNumber(numberAsString) {
    var number;

    if(numberAsString[0] == ’$’) {
        numberAsString = numberAsString.substr(1);
    }

    number = Number(numberAsString.replace(/,/g, ’’));

    return number;
}

function numberToFormattedString(number) {
    var numberAsString;

    numberAsString = String(number).replace(/\B(?=(\d{3})+(?!\d))/g, ",");

    return numberAsString;
}

function getData() {
    var date = document.getElementById("date");
    var account = document.getElementById("account").value;
    var type = document.getElementById("type").value;
    var security = document.getElementById("security").value;
    var amount = document.getElementById("amount").value;
    var dAmount = document.getElementById("dAmount").value;

    security = security.toUpperCase();

    amount = formattedStringToNumber(amount);

    dAmount = formattedStringToNumber(dAmount);

    if(validate(date, account, type, security, amount, dAmount)) {
        var costBasis = ’$’ + numberToFormattedString(calculateCostBasis(amount, dAmount));
        date = date.value;

        amount = numberToFormattedString(amount);
        dAmount = ’$’ + numberToFormattedString(dAmount.toFixed(2));

        return [ date, account, type, security, amount, dAmount, costBasis ];
    }
    else return false;
}

function validate(date, account, type, security, amount, dAmount) {
    if(!validateDate(date)) return false;
    if(!validateAccount(account)) return false;
    if(!validateType(type)) return false;
    if(!validateSecurity(security)) return false;
    if(!validateAmount(amount)) return false;
    if(!validateDAmount(dAmount)) return false;

    return true;
}

function validateDate(date) {
    realDate = new Date();
    inputDate = date.valueAsNumber;

    if(date.value == '') {
        alert('Error: Missing date');
        return false;
    }

    if(!date.checkValidity()) {
        alert('Error: Invalid date');
        return false;
    }

    if(realDate.valueOf() < inputDate) {
        alert('Error: Date is in the future');
        return false;
    }

    return true;
}

function validateAccount(account) {
    if(account == '') {
        alert('Error: Missing Account Number');
        return false;
    }

    return true;
}

function validateType(type) {
    if(type == '') {
        alert('Error: Missing Transaction Type');
        return false;
    }

    return true;
}

function validateSecurity(security) {
    if(security == '') {
        alert('Error: Missing Security');
        return false;
    }

    return true;
}

function validateAmount(amount) {
    if(amount == '') {
        alert('Error: Missing Amount');
        return false;
    }

    if(isNaN(amount)) {
        alert('Error: Invalid Amount');
        return false;
    }

    return true;
}

function validateDAmount(dAmount) {
    if(dAmount == '') {
        alert('Error: Missing $ Amount');
        return false;
    }

    if(isNaN(dAmount)) {
        alert('Error: Invalid $ Amount');
        return false;
    }

    return true;
}

function generateId() {
    var id = '';
    var idLength = 6;

    var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    var charactersLength = characters.length;

    var unique = false;

    while(!unique) {
        for(var i = 0; i < idLength; i++) {
            id += characters.charAt(Math.floor(Math.random() *
                ↪ charactersLength));
        }

        unique = true;
        for(var i = 0; i < document.getElementsByClassName('idCell').
            ↪ length; i++) {
            if(document.getElementsByClassName('idCell')[i].innerText ==
                ↪ id) {
                unique = false;
                break;
            }
        }
    }
    return id;
}

function calculateCostBasis(amount, dAmount) {
    costBasis = (dAmount / amount).toFixed(2);
    return costBasis;
}

function addTransaction(data) {
    var staging = data;
    var tableBody = document.getElementById('tableBody');
    var newRow = tableBody.insertRow(0);
    newRow.classList += "bodyRow";

    var actionsContent = "<button type='button' onclick='editRow(this)'>Edit</button><button type='button' onclick='deleteRow(this)'>Delete</button>";
    var fileContent = '<table><tbody>';
    if(data.length > 8) {
        for(let i = 0; i < data[8].length; i++) {
            fileContent += "<tr><td><a onclick='downloadFile(`" + data[8][i][0] + "`);' href='javascript:void(0);'>" + data[8][i][1] + "</a></td>";
            fileContent += "<td><button type='button' onclick='removeFileFromTable(`" + data[8][i][0] + "`, this);'>-</button></td></tr>";
        }
    }
    fileContent += '</tbody></table><input type="file" onchange="addFile(this, 0);" multiple/>';
    staging[8] = fileContent;
    staging[9] = actionsContent;

    var calculateCostBasis = true;
    if(data[3][0] == '!') {
        calculateCostBasis = false;
        data[3] = data[3].substr(1);
    }

    for(var i = 0; i < 10; i++) {
        var newCell = newRow.insertCell(i);
        var idShowing = (document.getElementById('toggleId').innerText == "Hide Transaction ID");

        newCell.innerHTML = data[i];

        if(i == 0) {
            if(idShowing)
                newCell.classList = "idCell frozenColumn1";
            else {
                newCell.classList = "idCell";
                newCell.setAttribute("hidden", true);
            }
        }
        else if(i == 1) {
            if(idShowing)
                newCell.classList = "frozenColumn2";
            else
                newCell.classList = "frozenColumn1";
        }
        /*
        else if(i == 2) {
            if(idShowing)
                newCell.classList = "frozenColumn3";
            else
                newCell.classList = "frozenColumn2";
        }
        */

        if(i == 7 && !calculateCostBasis) {
            newCell.innerHTML = "N/A";
        }
    }
}

function fileIdGenerator() {
    return Math.floor(Math.random() *
      ↪ 1000000000000000000000000000000000000000).toString(36);
}

function addTransactionButton() {
    var data = getData();
    if(data) {
        var id = generateId();
        data.unshift(id);

        fileList = new Array()
        uploadFile(data, addTransactionWithFileName, fileList, 0);
        clearInput(false);
    }
}

function addTransactionWithFileName(data, fileName) {
    addTransaction(data);
    console.log(data);
    loadDataLists();
}

function deleteRow(button) {
    var row = button.parentElement.parentElement;

    var fileRows = row.getElementsByTagName('td')[8].getElementsByTagName
      ↪ ('table')[0].getElementsByTagName('tr');
    for(let i = 0; i < fileRows.length; i++) {
        let fileId = fileRows[i].getElementsByTagName('a')[0].getAttribute
            ↪ ('onclick').split('`')[1];
        deleteFileFromIndexedDB(fileId);
    }

    document.getElementById("tableBody").removeChild(row);

    if(document.getElementsByClassName('editing').length == 0) {
        document.getElementById('add').removeAttribute('hidden');
        document.getElementById('save').setAttribute('hidden', true);
        document.getElementById('discard').setAttribute('hidden', true);

        document.getElementById('add').setAttribute('type','submit');
        document.getElementById('save').setAttribute('type','button');
    }
    loadDataLists();
}

function editRow(button) {
    if(document.getElementsByClassName('editing').length > 0)
        document.getElementsByClassName('editing')[0].classList = "bodyRow
            ↪ ";

    var row = button.parentElement.parentElement;
    var rowContent = row.getElementsByTagName('td');
    row.classList = "bodyRow editing";

    document.getElementById('date').value = rowContent[1].innerText;
    document.getElementById('account').value = rowContent[2].innerText;

    document.getElementById('type').value = rowContent[3].innerText;
    if(document.getElementById('type').value == '') document.
        ↪ getElementById('type').value = '!' + rowContent[3].innerText;

    document.getElementById('security').value = rowContent[4].innerText;
    document.getElementById('amount').value = rowContent[5].innerText;
    document.getElementById('dAmount').value = rowContent[6].innerText;

    document.getElementById('add').setAttribute('hidden', true);
    document.getElementById('save').removeAttribute('hidden');
    document.getElementById('discard').removeAttribute('hidden');

    document.getElementById('add').setAttribute('type','button');
    document.getElementById('save').setAttribute('type','submit');

    removeFileUpload();
    uploadLabel = document.getElementById('fileUploadLabel');
    if(rowContent[8].getElementsByTagName('tr').length > 0) {
        uploadLabel.innerHTML = String(rowContent[8].getElementsByTagName
            ↪ ('tr').length) + " file(s)";
    }
    fileEditted = false;
}

```

```
function saveChanges() {
    data = getData();
    if(data) {
        rowToEdit = document.getElementsByClassName('editing')[0];
        cellsToEdit = rowToEdit.getElementsByTagName('td');

        if(data[2][0] == '!') {
            data[2] = data[2].substr(1);
            data[6] = "N/A";
        }

        for(var i = 0; i < data.length; i++) {
            cellsToEdit[i + 1].innerHTML = data[i];
        }
        rowToEdit.classList = "bodyRow";

        document.getElementById('add').removeAttribute('hidden');
        document.getElementById('save').setAttribute('hidden', true);
        document.getElementById('discard').setAttribute('hidden', true);

        document.getElementById('add').setAttribute('type','submit');
        document.getElementById('save').setAttribute('type','button');

        if(fileEditted) {
            uploadFile([cellsToEdit[8]], updateExistingFileName, new Array
              ↪ (), 0);
        }
        else {
            removeFileUpload();
        }

        resetDate();
        clearInput(true);
        loadDataLists();
    }
}

function discardChanges() {
    document.getElementsByClassName('editing')[0].classList = "bodyRow";

    document.getElementById('add').removeAttribute('hidden');
    document.getElementById('save').setAttribute('hidden', true);
    document.getElementById('discard').setAttribute('hidden', true);

    document.getElementById('add').setAttribute('type','submit');
    document.getElementById('save').setAttribute('type','button');

    resetDate();
    clearInput(true);
    removeFileUpload();
}

function sortTable(column, ascending) {
    var rows = document.getElementsByClassName('bodyRow');

    var sorting = true;
    while(sorting) {
        sorting = false;
        for(var i = 0; i < (rows.length - 1); i++) {
            rowA = rows[i].getElementsByTagName('td')[column];
            rowB = rows[i + 1].getElementsByTagName('td')[column];

            var swap = false;

            if(ascending && rowA.innerHTML.toLowerCase() > rowB.innerHTML.toLowerCase()) swap = true;
            else if(!ascending && rowA.innerHTML.toLowerCase() < rowB.innerHTML.toLowerCase()) swap = true;

            if(swap) {
                sorting = true;
                document.getElementById('tableBody').insertBefore(rows[i + 1], rows[i]);
            }
        }
    }
}

function resetDate() {
    const today = new Date();
    const year = new Intl.DateTimeFormat('en', { year: 'numeric' }).format(today);
    const month = new Intl.DateTimeFormat('en', { month: '2-digit' }).format(today);
    const day = new Intl.DateTimeFormat('en', { day: '2-digit' }).format(today);

    document.getElementById('date').value = `${year}-${month}-${day}`;
}

function clearInput(clearAccount) {
    if(clearAccount)
        document.getElementById('account').value = '';

    document.getElementById('type').value = '';
    document.getElementById('security').value = '';
    document.getElementById('amount').value = '';
    document.getElementById('dAmount').value = '';
}

function validateFilters(filterId, startDate, endDate, filterAccount, filterType, filterSecurity, minAmount, maxAmount, minDAmount, maxDAmount, minCostBasis, maxCostBasis) {
    if(!validateFilterId(filterId)) return false;
    if(!validateDateRange(startDate, endDate)) return false;
    if(!validateFilterAccount(filterAccount)) return false;
    if(!validateFilterSecurity(filterSecurity)) return false;
    if(!validateAmountRange(minAmount, maxAmount)) return false;
    if(!validateDAmountRange(minDAmount, maxDAmount)) return false;
    if(!validateCostBasisRange(minCostBasis, maxCostBasis)) return false;

    return true;
}

function validateFilterId(id) {
    return true;
}

function validateDateRange(start, end) {
    if(!start.checkValidity()) {
        alert('Error: Invalid Start Date');
        return false;
    }

    if(!end.checkValidity()) {
        alert('Error: Invalid End Date');
        return false;
    }

    if(start.valueAsNumber > end.valueAsNumber) {
        alert('Error: Invalid Date Range');
        return false;
    }

    return true;
}

function validateFilterAccount(account) {
    return true;
}

function validateFilterSecurity(security) {
    return true;
}

function validateAmountRange(min, max) {
    if(isNaN(Number(min))) {
        alert('Error: Min Amount is NaN');
        return false;
    }

    if(isNaN(Number(max))) {
        alert('Error: Max Amount is NaN');
        return false;
    }

    if(Number(min) > Number(max) && min != '' && max != '') {
        if(min != '') alert(min + '2');
        if(max != '') alert(max + '1');
        alert('Error: Invalid Amount Range');
        return false;
    }

    return true;
}

function validateDAmountRange(min, max) {
    if(isNaN(Number(min))) {
        alert('Error: Min $ Amount is NaN');
        return false;
    }

    if(isNaN(Number(max))) {
        alert('Error: Max $ Amount is NaN');
        return false;
    }

    if(Number(min) > Number(max) && min != '' && max != '') {
        alert('Error: Invalid $ Amount Range');
        return false;
    }

    return true;
}

function validateCostBasisRange(min, max) {
    if(isNaN(Number(min))) {
        alert('Error: Min Cost Basis is NaN');
        return false;
    }

    if(isNaN(Number(max))) {
        alert('Error: Max Cost Basis is NaN');
        return false;
    }

    if(Number(min) > Number(max) && min != '' && max != '') {
        alert('Error: Invalid Cost Basis Range');
        return false;
    }

    return true;
}

function stringFilter(filtertext, tableitem) {
    filters = filtertext.split(" && ");

    for(var i = 0; i < filters.length; i++) {
        filterORs = filters[i].split(" || ");
        var meetsCriteria = false;

        for(var ii = 0; ii < filterORs.length; ii++) {
            if(filterORs[ii][0] == "!" && !tableitem.toUpperCase().
                ↪ includes(filterORs[ii].toUpperCase().substr(1)))
                ↪ meetsCriteria = true;
            if(filterORs[ii][0] != "!" && tableitem.toUpperCase().includes(filterORs[ii].toUpperCase())) meetsCriteria = true;
        }

        if(!meetsCriteria) return false;
    }

    return true;
}

function applyFilter() {
    unfilterAll();

    rows = document.getElementsByClassName('bodyRow');

    filterId = document.getElementById('filterId').value;
    startDate = document.getElementById('startDate');
    endDate = document.getElementById('endDate');
    filterAccount = document.getElementById('filterAccount').value;
    filterType = document.getElementById('filterType').value;
    filterSecurity = document.getElementById('filterSecurity').value;
    lowAmount = document.getElementById('lowAmount').value;
    highAmount = document.getElementById('highAmount').value;
    lowDAmount = document.getElementById('lowDAmount').value;
    highDAmount = document.getElementById('highDAmount').value;
    lowCostBasis = document.getElementById('lowCostBasis').value;
    highCostBasis = document.getElementById('highCostBasis').value;

    if(lowDAmount[0] == '$') lowDAmount = lowDAmount.substr(1);
    if(highDAmount[0] == '$') highAmount = highDAmount.substr(1);
    if(lowCostBasis[0] == '$') lowCostBasis = lowCostBasis.substr(1);
    if(highCostBasis[0] == '$') highCostBasis = highCostBasis.substr(1);

    if(validateFilters(filterId, startDate, endDate, filterAccount, filterType, filterSecurity, lowAmount, highAmount, lowDAmount, highDAmount, lowCostBasis, highCostBasis)) {
        for(var i = 0; i < rows.length; i++) {
            cells = rows[i].getElementsByTagName('td');
            var hide = false;

            if(filterId != '' && !stringFilter(filterId,cells[0].innerText))
                hide = true;

            if(startDate.value != '' && startDate.value > cells[1].innerText)
                hide = true;

            if(endDate.value != '' && endDate.value < cells[1].innerText)
                hide = true;

            if(filterAccount != '' && !stringFilter(filterAccount, cells[2].innerText))
                hide = true;

            if(filterType != '' && filterType != cells[3].innerText)
                hide = true;

            if(filterSecurity != '' && !stringFilter(filterSecurity, cells[4].innerText))
                hide = true;

            if(lowAmount != '' && Number(lowAmount) > formattedStringToNumber(cells[5].innerText))
                hide = true;

            if(highAmount != '' && Number(highAmount) < formattedStringToNumber(cells[5].innerText))
                hide = true;

            if(lowDAmount != '' && Number(lowDAmount) > formattedStringToNumber(cells[6].innerText.substr(1)))
                hide = true;

            if(highDAmount != '' && Number(highDAmount) < formattedStringToNumber(cells[6].innerText.substr(1)))
                hide = true;

            if(lowCostBasis != '' && Number(lowCostBasis) > formattedStringToNumber(cells[7].innerText.substr(1)))
                hide = true;

            if(highCostBasis != '' && Number(highCostBasis) < formattedStringToNumber(cells[7].innerText.substr(1)))
                hide = true;

            if(filterNA.checked && cells[7].innerText == "N/A")
                hide = true;

            if(hide)
                rows[i].setAttribute('hidden', true);
        }
    }
}

function clearFilter() {
    unfilterAll();

    var fields = document.getElementsByClassName('filterField');

    for(var i = 0; i < fields.length; i++) {
        fields[i].value = '';
    }

    document.getElementById('filterNA').checked = false;
}

function unfilterAll() {
    rows = document.getElementsByClassName('bodyRow');

    for(var i = 0; i < rows.length; i++) {
        rows[i].removeAttribute('hidden');
    }
}

function toggleID() {
    var button = document.getElementById('toggleId');
    var rows = document.getElementsByTagName('tr');
    var cells = rows[0].getElementsByTagName('th');

    if(button.innerText == "Hide Transaction ID") {
        button.innerText = "Show Transaction ID";

        cells[0].setAttribute('hidden', true);
        cells[0].classList = "";
        cells[1].classList = "frozenColumn1";
        //cells[2].classList = "frozenColumn2";
        for(var i = 1; i < rows.length; i++) {
            cells = rows[i].getElementsByTagName('td');

            cells[0].setAttribute('hidden', true);
            cells[0].classList = "idCell";
            cells[1].classList = "frozenColumn1";
            //cells[2].classList = "frozenColumn2";
        }
    }
    else {
        button.innerText = "Hide Transaction ID";

        cells[0].removeAttribute('hidden');
        cells[0].classList = "frozenColumn1";
        cells[1].classList = "frozenColumn2";
        //cells[2].classList = "frozenColumn3";
        for(var i = 1; i < rows.length; i++) {
            cells = rows[i].getElementsByTagName('td');

            cells[0].removeAttribute('hidden');
            cells[0].classList = "idCell frozenColumn1";
            cells[1].classList = "frozenColumn2";
            //cells[2].classList = "frozenColumn3";
        }
    }
}

function readFile(fileIn){
    if(fileIn.files && fileIn.files[0]) {
        var reader = new FileReader();
        reader.onload = function (e) {
            var output = e.target.result;
            document.getElementById('typesArray').value = output;
        };
        reader.readAsText(fileIn.files[0]);
    }
}

function saveFile() {
  var element = document.createElement('a');
  element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(document.getElementById('typesArray').value));
  element.setAttribute('download', 'transaction-types.csv');

  element.style.display = 'none';
  document.body.appendChild(element);

  element.click();

  document.body.removeChild(element);
}

function applyTypes() {
    var typesArray = document.getElementById('typesArray').value.split(',');
    setTransactionTypesList(typesArray);
}

function editTypes() {
    document.getElementById('typesArray').value = readCurrentTypes().join(',');
}

function setTransactionTypesList(typesArray) {
    var type = document.getElementById('type');
    var filterType = document.getElementById('filterType');

    type.innerHTML = '<option value=""></option>';
    filterType.innerHTML = '<option value=""></option>';

    for(var i = 0; i < typesArray.length; i++) {
        var typeAsText = typesArray[i];
        if(typesArray[i][0] == '!') typeAsText = typesArray[i].substr(1);

        type.innerHTML += '<option value="' + typesArray[i] + '">' + typeAsText + '</option>';
        filterType.innerHTML += '<option value="' + typeAsText + '">' + typeAsText + '</option>';
    }
}

function toggleSection(button) {
    var form = button.parentElement.parentElement.getElementsByTagName('form')[0];

    if(button.innerText == "Hide") {
        form.setAttribute("hidden",true);
        button.innerText = "Show";
    }
    else {
        form.removeAttribute("hidden");
        button.innerText = "Hide";
    }
}

function loadDataLists() {
    var accountsList = document.getElementById("accountsList");
    var securitiesList = document.getElementById("securitiesList");
    var rows = document.getElementsByClassName("bodyRow");

    var accounts = [];
    var securities = [];

    for(var i = 0; i < rows.length; i++) {
        var tableAccount = rows[i].getElementsByTagName("td")[2].innerText;
        var tableSecurity = rows[i].getElementsByTagName("td")[4].innerText;

        if(!accounts.includes(tableAccount)) accounts.push(tableAccount);
        if(!securities.includes(tableSecurity)) securities.push(tableSecurity);
    }

    accountsList.innerHTML = '';
    securitiesList.innerHTML = '';

    for(var i = 0; i < accounts.length; i++) {
        accountsList.innerHTML += '<option value="' + accounts[i] + '"/>';
    }

    for(var i = 0; i < securities.length; i++) {
        securitiesList.innerHTML += '<option value="' + securities[i] + '"/>';
    }
}

window.onload = function() {
    resetDate();
    initDb();
}
```

B.2 Google API Script

```
var auth2;

var spreadsheetId = "1R0HpaAIUw-JHX8SrzvkEPCG1qgI-siJ9oucY6g5e4Co";
var sheetId = "Sheet1";
var sheetIdNum = 0;

function loadSheetData() {
    if(auth2.isSignedIn.get())
    {
        getAllUserSheets();
    }
    else {
        authenticate()
            .then(function() {
                if(auth2.isSignedIn.get()) getAllUserSheets();
            });
    }
}

function getNewSheetData() {
    spreadsheetId = document.getElementById('sheet').value;
    if(auth2.isSignedIn.get())
    {
        getTabsOfSheet();
    }
    else {
        authenticate()
            .then(function() {
                if(auth2.isSignedIn.get()) getTabsOfSheet();
            });
    }
}

function populateSheetSelector(arrayOfSheets) {
    document.getElementById('sheet').innerHTML = '<option value="1R0HpaAIUw-JHX8SrzvkEPCG1qgI-siJ9oucY6g5e4Co">default</option>';

    for(var i = 0; i < arrayOfSheets.length; i++) {
        document.getElementById('sheet').innerHTML += '<option value="' + arrayOfSheets[i].id + '">' + arrayOfSheets[i].name + '</option>';
    }
}

function getNewTabData() {
    data = document.getElementById('tab').value.split(/,(.+)/);
    sheetIdNum = data[0];
    sheetId = data[1];
}

function populateTabSelector(arrayOfTabs) {
    document.getElementById('tab').innerHTML = '';

    for(var i = 0; i < arrayOfTabs.length; i++) {
        document.getElementById('tab').innerHTML += '<option value="' + arrayOfTabs[i].properties.sheetId + ',' + arrayOfTabs[i].properties.title + '">' + arrayOfTabs[i].properties.title + '</option>';
    }
}

function getAllUserSheets() {
    return gapi.client.drive.files.list({
        "pageSize": 1000,
        "orderBy": "name",
        "q": "mimeType = 'application/vnd.google-apps.spreadsheet'",
    })
        .then(function(response) {
            populateSheetSelector(JSON.parse(response.body).files);
            getNewSheetData();
            console.log("Response", response);
        },
        function(err) { console.error("Execute error", err); });
}

function getTabsOfSheet() {
    return gapi.client.sheets.spreadsheets.get({
      "spreadsheetId": spreadsheetId,
      "includeGridData": false
    })
        .then(function(response) {
            populateTabSelector(JSON.parse(response.body).sheets);
            getNewTabData();
            console.log("Response", response);
        },
        function(err) { console.error("Execute error", err); });
}

function tableToArrays() {
    var rows = document.getElementsByClassName('bodyRow');
    var data = new Array();
    data.push(["Transaction Id", "Date", "Account Number", "Transaction Type", "Security", "Amount", "$ Amount", "Cost Basis", "Files"]);

    for(var i = 0; i < rows.length; i++) {
        var cells = rows[i].getElementsByTagName('td');
        var cellData = new Array();

        for(var j = 0; j < 8; j++) {
            cellData.push(cells[j].innerText);
        }
        cellData.push(getFileNamesIds(cells[8]));
        data.push(cellData);
    }

    console.log(data);
    return data;
}

function arraysToTable(dataArr) {
    while(document.getElementsByClassName('bodyRow').length > 0) {
        document.getElementById("tableBody").removeChild(document.getElementsByClassName('bodyRow')[0]);
    }

    document.getElementById('add').removeAttribute('hidden');
    document.getElementById('save').setAttribute('hidden', true);
    document.getElementById('discard').setAttribute('hidden', true);

    document.getElementById('add').setAttribute('type', 'submit');
    document.getElementById('save').setAttribute('type', 'button');

    while(dataArr.length > 0) {
        let data = dataArr[dataArr.length - 1];
        let files = parseFileNamesIds(data[8]);
        data.pop();
        if(files.length > 0) {
            data.push(files);
        }
        addTransaction(data);
        dataArr.pop();
    }
    loadDataLists();
}

function authenticate() {
return gapi.auth2.getAuthInstance()
    .signIn({scope: "https://www.googleapis.com/auth/drive"})
    .then(function() { console.log("Sign-in successful"); },
        function(err) { console.error("Error signing in", err); });
}

function loadClientSheets() {
gapi.client.setApiKey("AIzaSyDC6JNuMW78Q-gWsp0PFEaTsICYjHWymAo");
return gapi.client.load("https://content.googleapis.com/discovery/v1/apis/sheets/v4/rest")
    .then(function() { console.log("GAPI client loaded for API"); loadSheetData(); },
        function(err) { console.error("Error loading GAPI client for API", err); });
}

function loadClient() {
gapi.client.setApiKey("AIzaSyDC6JNuMW78Q-gWsp0PFEaTsICYjHWymAo");
return gapi.client.load("https://content.googleapis.com/discovery/v1/apis/drive/v3/rest")
    .then(function() { console.log("GAPI client loaded for API"); loadClientSheets(); },
        function(err) { console.error("Error loading GAPI client for API",
          ↪  err); });
}

function readGoogleSheetDB() {
    clearIndexedDb("sheets");
    return gapi.client.sheets.spreadsheets.values.get({
        "spreadsheetId": spreadsheetId,
        "range": sheetId + "!A2:I214748354"
    })
        .then(function(response) {
            console.log("Response", response);

            dataArr = [];
            if(JSON.parse(response.body).values != undefined) {
                dataArr = JSON.parse(response.body).values;
            }
            arraysToTable(dataArr);

            readGoogleTypes();
        },
        function(err) { console.error("Execute error", err); });
}

function readGoogleTypes() {
    return gapi.client.sheets.spreadsheets.values.get({
        "spreadsheetId": spreadsheetId,
        "range": sheetId + "!J1:J214748354",
        "majorDimension": "COLUMNS"

    })
        .then(function(response) {
            console.log("Response", response);
            var typesArr = JSON.parse(response.body).values[0];
            setTransactionTypesList(typesArr);
        },
        function(err) { console.error("Execute error", err); });
}

function readCurrentTypes() {
    var types = document.getElementById('type').getElementsByTagName('
      ↪ option');
    var currentTypes = [];

    for(var i = 1; i < types.length; i++) {
        currentTypes.push(types[i].value);
    }

    return currentTypes;
}

function writeGoogleSheetDB() {
    if(auth2.isSignedIn.get())
    {
        setGoogleRows()
    }
    else {
        authenticate()
            .then(function() {
                if(auth2.isSignedIn.get()) setGoogleRows();
            });
    }
}

function setGoogleRows() {
    return gapi.client.sheets.spreadsheets.batchUpdate({
        "spreadsheetId": spreadsheetId,
        "resource": {
        "requests": [
            {
            "updateSheetProperties": {
                "properties": {
                    "gridProperties": {
                        "columnCount": 10,
                        "rowCount": 1
                },
                    "sheetId": sheetIdNum
                },
                "fields": "gridProperties"
            }
            }
        ]
        }
    })
        .then(function(response) {
            console.log("Response", response);

            clearGoogleRow();
        },
        function(err) { console.error("Execute error", err); });
}

function clearGoogleRow() {
    return gapi.client.sheets.spreadsheets.values.clear({
    "spreadsheetId": spreadsheetId,
    "range": sheetId + "!A1:J1",
    "resource": {}
    })
        .then(function(response) {
            console.log("Response", response);

            writeGoogleDB();
        },
        function(err) { console.error("Execute error", err); });
}

function writeGoogleDB() {
    writeImagesToFirestore("sheets");
    return gapi.client.sheets.spreadsheets.values.batchUpdate({
        "spreadsheetId": spreadsheetId,
        "resource": {
            "data": [
            {
                "range": sheetId + "!A1",
                "values": tableToArrays(),
                "majorDimension": "ROWS"
            },
            {
                "range": sheetId + "!J1",
                "values": [readCurrentTypes()],
                "majorDimension": "COLUMNS"
            }
            ],
            "valueInputOption": "RAW"
        }
        })
        .then(function(response) {
            console.log("Response", response);
        },
        function(err) { console.error("Execute error", err); });
}

gapi.load("client:auth2", function() {
    auth2 = gapi.auth2.init({client_id: "217251662395-9
       ↪ pu2qa1hubgrblav1nhvnfaascd6povv.apps.googleusercontent.com"});
    loadClient();
});
```

B.3 Firebase Script

```
function clearFirebase() {
    firebase.database().ref('Data').remove();
}

function writeToFirebase() {
    writeImagesToFirestore("firebase");
    clearFirebase();

    var data = tableToArrays();
    var typesArr = readCurrentTypes();

    for(var i = 1; i < data.length; i++) {
        firebase.database().ref('Data/' + String(i - 1)).set({
            id: data[i][0],
            date: data[i][1],
            account: data[i][2],
            type: data[i][3],
            security: data[i][4],
            amount: data[i][5],
            dAmount: data[i][6],
            costBasis: data[i][7],
            files: data[i][8]
        });
    }
    for(var i = 0; i < typesArr.length; i++) {
        firebase.database().ref('Types/' + String(i)).set({
            value: typesArr[i]
        });
    }
}

function readFromFirebase() {
    clearIndexedDb("firebase");
    return firebase.database().ref('/').once('value').then(function(snapshot) {

        while(document.getElementsByClassName('bodyRow').length > 0) {
            document.getElementById("tableBody").removeChild(document.getElementsByClassName('bodyRow')[0]);
        }

        data = snapshot.val().Data;
        console.log(data);
        for(var i = data.length - 1; i >= 0; i--) {
            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];
            if(parseFileNamesIds(data[i].files).length > 0) {
                staged.push(parseFileNamesIds(data[i].files));
            }
            console.log(staged);
            addTransaction(staged);
        }

        types = snapshot.val().Types;
        var typesArr = [];
        for(var i = 0; i < types.length; i++) {
            typesArr.push(types[i].value);
        }
        setTransactionTypesList(typesArr);
        loadDataLists();
    });
}

function clearFirestore() {
    firestore.collection("Data").get().then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            firestore.collection("Data").doc(doc.id).delete();
        });
    }).then(function() {
        firestore.collection("Types").get().then((querySnapshot) => {
            querySnapshot.forEach((doc) => {
                firestore.collection("Types").doc(doc.id).delete();
            });
        })
    }).then(function() { return 0 });
}

function writeToFirestore() {
    writeImagesToFirestore("firestore");
    firestore.collection("Data").get().then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            firestore.collection("Data").doc(doc.id).delete();
        });
    }).then(function() {
        firestore.collection("Types").get().then((querySnapshot) => {
            querySnapshot.forEach((doc) => {
                firestore.collection("Types").doc(doc.id).delete();
            });
        }).then(function() {
            var data = tableToArrays();
            var typesArr = readCurrentTypes();

            for(var i = 1; i < data.length; i++) {
                firestore.collection("Data").add({
                    id: data[i][0],
                    date: data[i][1],
                    account: data[i][2],
                    type: data[i][3],
                    security: data[i][4],
                    amount: data[i][5],
                    dAmount: data[i][6],
                    costBasis: data[i][7],
                    files: data[i][8],
                    index: i - 1
                })
                .then(function(docRef) {
                    console.log("Document written with ID: ", docRef.id);
                })
                .catch(function(error) {
                    console.error("Error adding document: ", error);
                });
            }
            for(var i = 0; i < typesArr.length; i++) {
                firestore.collection("Types").add({
                    value: typesArr[i],
                    index: i
                })
                .then(function(docRef) {
                    console.log("Document written with ID: ", docRef.id);
                })
                .catch(function(error) {
                    console.error("Error adding document: ", error);
                });
            }
        })
    });
}

function readFromFirestore() {
    clearIndexedDb("firestore");
    firestore.collection("Data").get().then((querySnapshot) => {
        var data = new Array();

        querySnapshot.forEach((doc) => {
            data[doc.data().index] = doc.data();
            console.log(data);
        });

        while(document.getElementsByClassName('bodyRow').length > 0) {
            document.getElementById("tableBody").removeChild(document.
              ↪ getElementsByClassName('bodyRow')[0]);
        }

        for(let i = data.length - 1; i >= 0; i--) {
            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];
            if(parseFileNamesIds(data[i].files).length > 0) {
                staged.push(parseFileNamesIds(data[i].files));
            }
            console.log(staged);
            addTransaction(staged);
        }
        loadDataLists();
    });

    firestore.collection("Types").get().then((querySnapshot) => {
        var typesArr = [];

        querySnapshot.forEach((doc) => {
            typesArr[doc.data().index] = doc.data().value;
        });

        setTransactionTypesList(typesArr);
    });
}
```

B.4 MySQL Script

```
function writeToMySQL() {
    writeImagesToFirestore("mysql");
    var data = tableToArrays();
    var types = readCurrentTypes();

    fetch('http://localhost:5000/api', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify([data, types]),
    });
}

function readFromMySQL() {
    clearIndexedDb("mysql");
    fetch('http://localhost:5000/api')
        .then(response => {
            return response.json()
        })
        .then(fullresponse => {
            console.log(fullresponse);

            while(document.getElementsByClassName('bodyRow').length > 0) {
                document.getElementById("tableBody").removeChild(document.getElementsByClassName('bodyRow')[0]);
            }

            var data = fullresponse[0];
            for(var i = data.length - 1; i >= 0; i--) {
                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];
                if(parseFileNamesIds(data[i].files).length > 0) {
                    staged.push(parseFileNamesIds(data[i].files));
                }
                addTransaction(staged);
            }
            loadDataLists();

            var types = fullresponse[1];
            var typesArr = [];
            for(var i = 0; i < types.length; i++) {
                typesArr.push(types[i].typename);
            }
            setTransactionTypesList(typesArr);
        })
}
```

B.5 Local Storage Script

```
let db;
let dbVersion = 1;
let dbReady = false;

var fileEditted = false;

function initDb() {
    let reset = indexedDB.deleteDatabase('FileStorage');
    reset.onsuccess = function(a) {
        let request = indexedDB.open('FileStorage', dbVersion);

        request.onerror = function(e) {
            console.error('Unable to open database.');
        }

        request.onsuccess = function(e) {
            db = e.target.result;
            console.log('db opened');
        }

        request.onupgradeneeded = function(e) {
            let db = e.target.result;
            db.createObjectStore('files', {keyPath:'id', autoIncrement:
                ↪ false});
            dbReady = true;
        }
    }
}

function fileUploadChanged() {
    fileIn = document.getElementById('fileUpload');
    if(fileIn.files && fileIn.files[0]) {
        document.getElementById('fileUploadLabel').innerHTML = String(
            ↪ fileIn.files.length) + " file(s)";
    }
    fileEditted = true;
    console.log("updated file upload");
}

function uploadFile(data, cb, fileList, index) {
    fileId = fileIdGenerator();
    fileIn = document.getElementById('fileUpload');
    if(fileIn.files && fileIn.files[index]) {
        var reader = new FileReader();
        reader.onload = function (e) {
            console.log(e.target.result);

            let bits = btoa(e.target.result);
            let ob = {
                id: fileId,
                type: fileIn.files[index].type,
                name: fileIn.files[index].name,
                data: bits
            };

            let trans = db.transaction(['files'], 'readwrite');
            let addReq = trans.objectStore('files').put(ob);

            addReq.onerror = function(e) {
                console.log('error storing data');
                console.error(e);
            }

            trans.oncomplete = function(e) {
                console.log('data stored');
                fileList.push([fileId, fileIn.files[index].name]);
                index++;
                uploadFile(data, cb, fileList, index);
            }
        };
        reader.readAsBinaryString(fileIn.files[index])
    }
    else {
        removeFileUpload();
        data.push(fileList);
        cb(data);
    }
}

function addFile(fileIn, index) {
    fileId = fileIdGenerator();
    if(fileIn.files && fileIn.files[index]) {
        var reader = new FileReader();
        reader.onload = function (e) {
            console.log(e.target.result);

            let bits = btoa(e.target.result);
            let ob = {
                id: fileId,
                type: fileIn.files[index].type,
                name: fileIn.files[index].name,
                data: bits
            };

            let trans = db.transaction(['files'], 'readwrite');
            let addReq = trans.objectStore('files').put(ob);

            addReq.onerror = function(e) {
                console.log('error storing data');
                console.error(e);
            }

            trans.oncomplete = function(e) {
                console.log('data stored');
                let table = fileIn.parentElement.getElementsByTagName('
                    ↪ tbody')[0];
                let newRowContent = "<tr><td><a onclick='downloadFile(`" + fileId + "`);' href='javascript:void(0);'>" + fileIn.files[index].name + "</a></td>";
                newRowContent += "<td><button type='button' onclick='removeFileFromTable(`" + fileId + "`, this);'>-</button></td></tr>";
                table.innerHTML += newRowContent;
                index++;
                addFile(fileIn, index);
            }
        };
        reader.readAsBinaryString(fileIn.files[index])
    }
    else {
        fileIn.value = null;
    }
}

function updateExistingFileName(data) {
    for(let i = 0; i < data[0].getElementsByTagName('tr').length; i++) {
        let fileId = data[0].getElementsByTagName('tr')[i].getElementsByTagName('a')[0].getAttribute('onclick').split('`')[1];
        deleteFileFromIndexedDB(fileId);
    }

    var fileContent = '<table><tbody>';
    if(data.length > 1) {
        for(let i = 0; i < data[1].length; i++) {
            fileContent += "<tr><td><a onclick='downloadFile(`" + data[1][i][0] + "`);' href='javascript:void(0);'>" + data[1][i][1] + "</a></td>";
            fileContent += "<td><button type='button' onclick='removeFileFromTable(`" + data[1][i][0] + "`, this);'>-</button></td></tr>";
        }
    }
    fileContent += '</tbody></table><input type="file" onchange="addFile(this, 0);" multiple/>';
    data[0].innerHTML = fileContent;
}

function removeFileFromTable(fileId, cell) {
    var row = cell.parentElement.parentElement;

    if(confirm("Delete " + row.getElementsByTagName('a')[0].innerText + "?")) {
        row.parentElement.removeChild(row);
        deleteFileFromIndexedDB(fileId);
    }
}

function deleteFileFromIndexedDB(fileId) {
    let trans = db.transaction(['files'], 'readwrite');
    let addReq = trans.objectStore('files').delete(fileId);
}

function removeFileUpload() {
    document.getElementById('fileUpload').value = null;
    document.getElementById('fileUploadLabel').innerHTML = "Upload file";
    fileEditted = true;
}

function downloadFile(fileId) {
    console.log('downloading');
    var trans = db.transaction(['files'], 'readonly');
    var dlReq = trans.objectStore('files').get(fileId);

    dlReq.onerror = function(e) {
        console.log('error reading data');
        console.error(e);
    };

    dlReq.onsuccess = function(e) {
        console.log('data read');
        console.log(dlReq.result);
        var element = document.createElement('a');
        element.setAttribute('href', 'data:' + dlReq.result.type + ';
           ↪ base64,' + dlReq.result.data);
        element.setAttribute('download', dlReq.result.name);

        element.style.display = 'none';
        document.body.appendChild(element);

        element.click();

        document.body.removeChild(element);
    };
}

window.onbeforeunload = function(){
    indexedDB.deleteDatabase('FileStorage');
}
```

B.6 Firestore Images Script

```
function writeImagesToFirestore(database) {
    firestore.collection("Images:" + database).get().then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            firestore.collection("Images:" + database).doc(doc.id).delete();
        });
    }).then(function() {
        var trans = db.transaction(['files'], 'readonly');
        var dlReq = trans.objectStore('files').getAll();

        dlReq.onerror = function(e) {
            console.log('error reading data');
            console.error(e);
        };

        dlReq.onsuccess = function(e) {
            console.log(dlReq.result);

            for(let i = 0; i < dlReq.result.length; i++) {
                firestore.collection("Images:" + database).add({
                    id: dlReq.result[i].id,
                    type: dlReq.result[i].type,
                    name: dlReq.result[i].name,
                    data: dlReq.result[i].data,
                })
                .then(function(docRef) {
                    console.log("Image written with ID: ", docRef.id);
                })
                .catch(function(error) {
                    console.log("Error adding image: ", error);
                });
            }
        };
    });
}

function readImagesFromFirestore(database) {
    firestore.collection("Images:" + database).get().then((querySnapshot) => {
        let trans = db.transaction(['files'], 'readwrite');

        trans.oncomplete = function(e) {
            console.log('data stored');
        }

        querySnapshot.forEach((doc) => {
            console.log("Writing ", doc.data().name);

            let ob = {
                id: doc.data().id,
                type: doc.data().type,
                name: doc.data().name,
                data: doc.data().data
            };

            let addReq = trans.objectStore('files').put(ob);
        });
    });
}

function getFileNamesIds(cell) {
    var links = cell.getElementsByTagName('a');
    var result = '';

    for(let i = 0; i < links.length; i++) {
        result += links[i].innerText + '/' + links[i].getAttribute('
           ↪ onclick').split('`')[1] + '/';
    }

    return result;
}

function parseFileNamesIds(string) {
    var result = new Array();

    if(string != '') {
        let strarr = string.split('/');
        for(let i = 0; i < strarr.length - 1; i += 2) {
            result.push([strarr[i + 1], strarr[i + 0]]);
        }
    }

    return result;
}

function clearIndexedDb(database) {
    console.log("db reset");
    let trans = db.transaction(['files'], 'readwrite');
    var clearReq = trans.objectStore('files').clear();

    trans.oncomplete = function(e) {
        readImagesFromFirestore(database);
    }
}
```

C CSS Source Code

```css
body {
    font-size: 14px;
}

input,
select {
    min-width: 100px;
    width: 80%;
}

button {
    width: 40%;
}

span {
    border: solid;
    border-width: 1px;
}

#fileUpload {
    display: none;
}

#removeFile {
    width: 16px;
    padding: 0;
    border: 0;
}

.sectionHead {
}

    .toggleSection {
        width: 100px;
    }

    .sectionHead > * {
        display: inline-block;
    }

#table {
    max-height: 50vh;
    overflow: auto;
}

#inputFields,
#filter {
    padding: 10px 0;
    overflow-x: auto;
}

#inputFields > form,
#filter > form {
    min-width: 1800px;
}

    #inputFields > form > section {
        width: 14%;
        display: inline-block;
    }

    #filter > form > section {
        width: 10%;
        display: inline-block;
    }

        #filter > form > section > button {
            width: 100%;
        }

#filterNA {
    display: inline-block;
    width: 50%;
    min-width: 16px;
}

#options > form > section {
    display: inline-block;
    width: 300px;
    margin: 15px;
}

    #toggleId,
    #fileInput {
        width: 80%;
    }

    #transactionTypesConfig {
        width: 500px !important;
    }

    #transactionTypesConfig > button {
        display: inline-block;
        width: 30%;
        padding: 0;
    }

    #typesArray {
        width: 40%;
    }

    #options > form > #googleSheetButtons {
        width: 200px;
    }

    #logoutSection#logoutSection {
        width: 100px;
    }

    #googleSheetButtons > button,
    #logoutSection > button {
        width: 100%;
        padding: 0;
        margin: 0;
    }

table {
    width: 100%;
    margin: auto;
    border-collapse: collapse;
}
    th {
        min-width: 200px;
        width: 10%;
        position: sticky;
        top: 0;
    }

    th > section {
        width: 80%;
        display: inline-block;
        padding: 0;
        margin: 0;
    }

    th,
    td {
        z-index: 0;
        background: white;
    }

    .frozenColumn1,
    .frozenColumn2,
    .frozenColumn3 {
        position: sticky;
        padding: 0;
        min-width: 200px;
    }

    td.frozenColumn1,
    td.frozenColumn2,
    td.frozenColumn3 {
        z-index: 1;
    }

    th.frozenColumn1,
    th.frozenColumn2,
    th.frozenColumn3 {
        z-index: 2;
    }

    .frozenColumn1 {
        left: 0;
    }

    .frozenColumn2 {
        left: 200px;
    }

    .frozenColumn3 {
        left: 400px;
    }

    .sort {
        width: 10%;
    }

    .sort > button {
        padding: 0;
        border: 0;
        display: block;
        width: 100%;
    }

    .editing > td {
        background: yellow;
    }

    #table,
    table,
    td,
    th {
        box-shadow: 1px 1px black, inset 1px 1px black;
    }
```

D Server-side NodeJs Code

```
const functions = require('firebase-functions');
const express = require('express');
const mysql = require('mysql');
const cors = require('cors');
const api = express();
api.use(cors({ origin: true }));

var con = mysql.createConnection({
    host: "localhost",
    user: "user",
    password: "pass",
    database: "mydb"
});

con.connect(function(err) {
    if (err) throw err;
    console.log("Connected to MySQL Database!");
});

function readDataFromMySQL(cb) {
    con.query("SELECT * FROM data;", function (err, data, fields) {
        if (err) throw err;
        cb(data);
    });
}

function readTypesFromMySQL(cb) {
    con.query("SELECT * FROM types;", function (err, types, fields) {
        if (err) throw err;
        cb(types);
    });
}

function writeToMySQL(jsonData) {
    var data = jsonData[0];
    var types = jsonData[1];

    con.query("TRUNCATE TABLE data;");

    for(var i = 1; i < data.length; i++) {
        con.query("INSERT INTO data (id, date, account, type, security, amount, dAmount, costBasis, files) VALUES ('" + data[i].join("','") + "');");
    }

    con.query("TRUNCATE TABLE types;");

    for(var i = 0; i < types.length; i++) {
        con.query("INSERT INTO types (typename) VALUES ('" + types[i] + "');");
    }
}
```

```

api.post('/api', (req, res) => {
    console.log(req.body);
    writeToMySQL(req.body);
    return res.status(200).send(req.body);
});

api.get('/api', (req, res) => {
    readDataFromMySQL(function(data) {
        console.log(data);
        readTypesFromMySQL(function(types) {
            console.log(types);
            return res.status(200).send([ data, types ]);
        });
    });
});

exports.api = functions.https.onRequest(api);
```