MyTrak

Final Project

Yumi Lim

Agenda

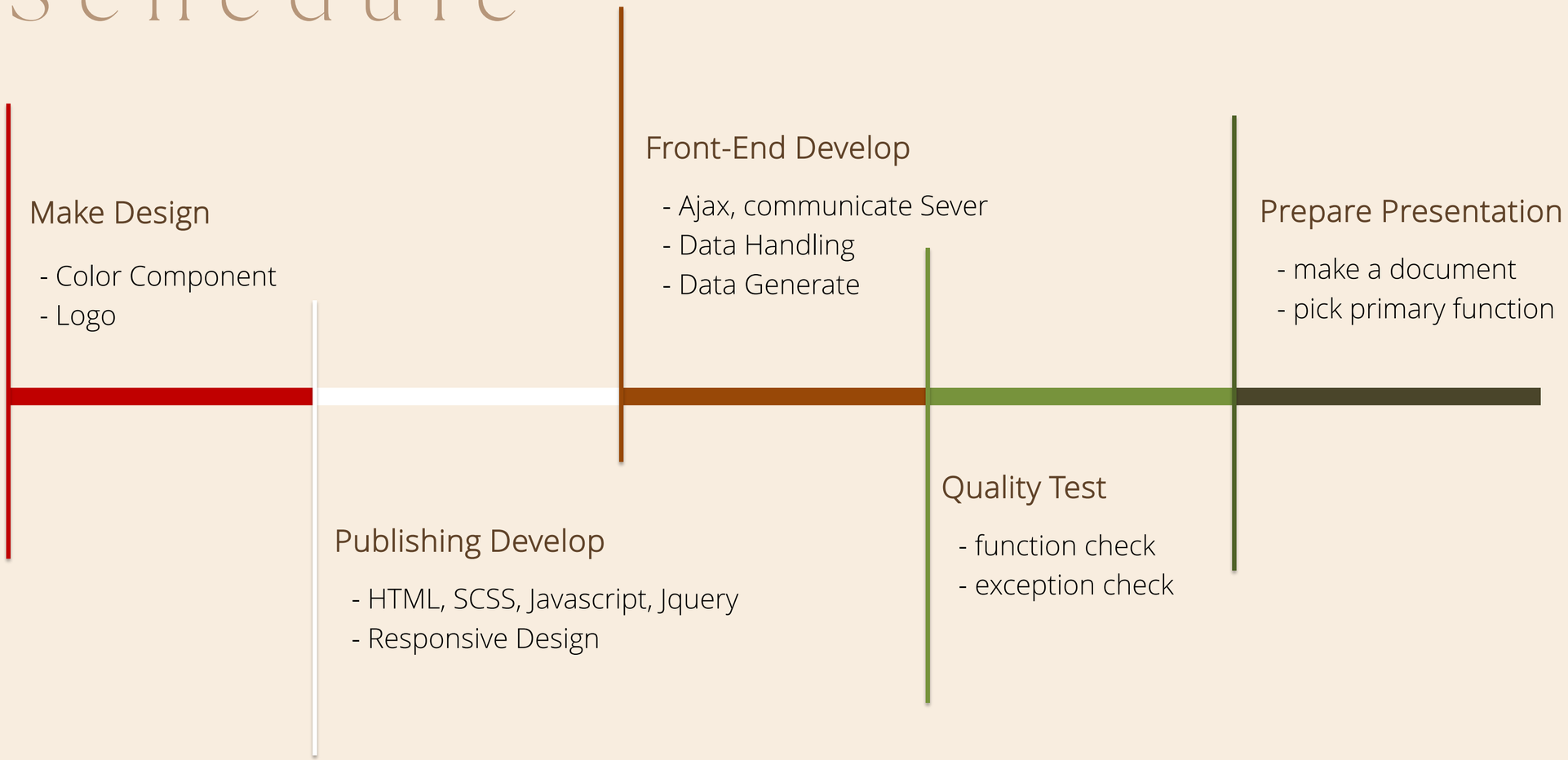
Project Schedule

Design

Demo

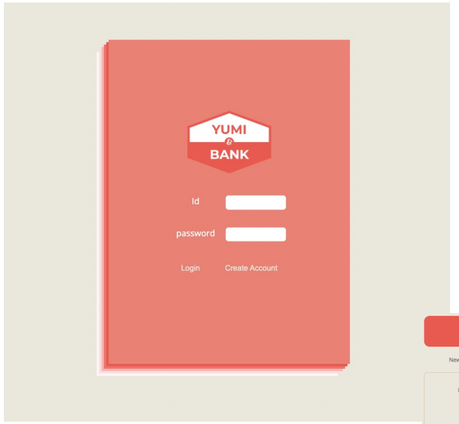
Code Review

Project Schedule

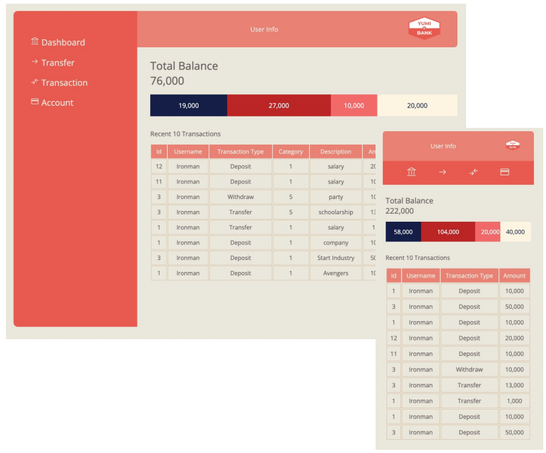


Design

Logo Design



Page Design



Color Palette

Technical Document

Presentation Website

Code Review

Index.js

```
$(() => {
  //init sequence - animation and get init data
  logoAnimation();
  //radio button toggle;
  checkRadioButton();
  //update selector
  initAccount();
  initCategory();
```

```
function initAccount() {
    $.ajax({
        url: accountUrl,
        type: 'GET',
        success: function (response) {
            console.log(response);
            initAccountSelectorUI(response);
            initAccountIdSelectorUI(response);
        },
        error: function (error) {
            console.error('Error retrieving user details:', error);
        }
    });
}
```

Account.js

initiate Function

Category.js

```
const categoryUrl = "http://localhost:3002/categories"
const categorylist = [];

// initialize Category
function initCategory() {
    $.ajax({
        url: categoryUrl,
        type: 'GET',
        success: function (response) {
            console.log(response);
            while (categorylist.length) {
                categorylist.pop();
            }
            response.forEach(element => {
                categorylist.push(element);
            });
            initCategorySelect(response);
        }
        error: function (error) {
            console.error('Error retrieving user details:', error);
        }
    });
}
```

```
function addCategory(category) {
    $.ajax({
        type: "POST",
        url: categoryUrl,
        data: JSON.stringify({ newCategory: category }),
        contentType: 'application/json',
        success: function (response) {
            showSnackbar("creat Category : " + category);
            initCategory();
        },
        error: function (error) {
            alert.log("Error Creating user" + error);
        }
    });
}
```

```
<td class="hidden_mobile">`+ categorylist[newTrans[i].categoryId - 1].name + `</td>
```

Common.js

Primary Function

Index.js

```
// button click and select change
// current Account summary
$("#currentAccount").on("change", function () {
  var username = "";
  $("#currentAccount option:selected").each(function () {
    username = $(this).text();
  });
  if (username != "") {
    getAccountSummary(username);
  }
})

// selected Account transation
$("#currentDetailAccount").on("change", function () {
  var username = "";
  $("#currentDetailAccount option:selected").each(func
    username = $(this).text();
  });
  if (username != "") {
    getTransationDetail(username);
  }
})
```

```
// button New Account
$("#btnNewAccount").on("click", function () {
  let username = $("input[name=txtNewAccount]").val();
  if (username == "") {
    showSnackbar("please enter the text");
  } else {
    checkMaximumAccount(username);
  }
});
  // button New Category
$("#btnNewCategory").on("click", function () {
  let category = $("input[name=txtNewCategory]").val();
  if (category == "") {
    showSnackbar("please enter the text");
  } else {
    checkDuplicateCategory(category);
  }
});
```

```
  // button New Transaction
$("#btnAddTrans").on("click", function(){
  let formData = {
      accountId: $("select[name=accountSelect").val(),
      accountIdFrom: $("select[name=fromSelect]").val(),
      accountIdTo: $("select[name=toSelect]").val(),
      type: $('input[type=radio][name=transType]:checked').val(),
      amount: $('input[type=number][name=amount]').val(),
      categoryId: $(".categorySelect").val(),
      description: $('input[type=text][name=description]').val()
  }
  if (formData.type == "Withdraw" || formData.type == "Deposit") {…
  } else if (formData.type == "Transfer") {…
  }
  if (formData.categoryId == null) {…
  }
  if (formData.amount == "") {…
  }
  if(flag) {
    checkTransaction(formData);
    $("#transForm")[0].reset();
  }
})
$("#btnAdminLogout").on("click", function () {
  sessionStorage.clear();
  window.location = "./login.html";
})
);
```

Primary Function

Account.js

```
$("#btnNewAccount").on("click", function () {
  let username = $("input[name=txtNewAccount]").val();
  if (username == "") {
    showSnackbar("please enter the text");
  } else {
    checkMaximumAccount(username);
  }
});
```

```
function checkMaximumAccount(username) {
    $.ajax({
        url: accountUrl,
        type: 'GET',
        success: function (response) {
            let newData = response.filter(function (obj) {
                return obj.username == username;
            });
            if(newData.length < 5){
                addNewAccount(username);
            }else{
                showSnackbar(username + " : maximum account is 5");
            }
        },
        error: function (error) {
            console.error('Error retrieving user details:', error);
        }
    });
}
```

```
function addNewAccount(username) {
    $.ajax({
        type: "POST",
        url: accountUrl,
        data: JSON.stringify({newAccount: username}),
        contentType: 'application/json',
        success: function (response) {
            showSnackbar("creat account : " + username);
            initAccount();
        },
        error: function (error) {
            alert.log("Error Creating user" + error);

        }
    });
}
```

Primary Function

Category.js

```
$("#btnNewCategory").on("click", function () {
  let category = $("input[name=txtNewCategory]").val();
  if (category == "") {
    showSnackbar("please enter the text");
  } else {
    checkDuplicateCategory(category);
  }
});
```

```
function checkDuplicateCategory(category) {
    $.ajax({
        url: categoryUrl,
        type: 'GET',
        success: function (response) {
            var index = response.map(i => i.name).indexOf(category);
            if(index == -1){
                addCategory(category);
            }else{
                showSnackbar(category + " alrady exsist");
            }
        },
        error: function (error) {
            console.error('Error retrieving user details:', error);
        }
    });
}
```

```
function addCategory(category) {
    $.ajax({
        type: "POST",
        url: categoryUrl,
        data: JSON.stringify({ newCategory: category }),
        contentType: 'application/json',
        success: function (response) {
            showSnackbar("creat Category : " + category);
            initCategory();
        },
        error: function (error) {
            alert.log("Error Creating user" + error);
        }
    });
}
```

Primary Function

```
$("#btnAddTrans").on("click", function(){
  let flag = true;
  let formData = {
  }
  if (formData.type == "Withdraw" || formData.type == "Deposit") {
    if (formData.accountId == null) {
      showSnackbar("please select account ID");
      flag = false;
    }
    formData.accountIdFrom = null;
    formData.accountIdTo = null;
  } else if (formData.type == "Transfer") {
    if (formData.accountIdFrom == null || formData.accountIdTo == null) {
      showSnackbar("please select account ID");
      flag = false;
    }
    if(formData.accountIdFrom ==formData.accountIdTo){
      showSnackbar("select different ID");
      flag = false;
    }
  }
  if (formData.categoryId == null) {
    showSnackbar("please select category");
    flag = false;
  }
  if (formData.amount == "") {
    showSnackbar("please write amount");
    flag = false;
  }
  if(flag) {
    checkTransaction(formData);
    $("#transForm")[0].reset();
  }
})
```

```
function checkTransaction(formData) {
  if(formData.type == "Withdraw" || formData.type == "Transfer"){
    checkBalance(formData);
  }else{
    addTransaction(formData);
  }
}
```

Index.js

```
function checkBalance(transData){
    $.ajax({
        url: accountUrl,
        type: 'GET',
        success: function (response) {
            var index = -1;
            if(transData.type == "Withdraw"){
                index = transData.accountId - 1;
            }else if( transData.type == "Transfer"){
                index = transData.accountIdFrom - 1;
            }
            if(getBalance(response[index]) < parseFloat(transData.amount)){
                alert("Fail : current Balance _ " + getBalance(response[index]))
            }else{
                addTransaction(transData);
            }
        },
        error: function (error) {
            console.error('Error retrieving user details:', error);
        }
    });
}
```

Account.js

Primary Function

Transaction.js

```
function addTransaction(data) {
    var tmpData = {
        newTransaction: {
            accountId: parseFloat(data.accountId),
            accountIdFrom: parseFloat(data.accountIdFrom),
            accountIdTo: parseFloat(data.accountIdTo),
            type: data.type,
            amount: parseFloat(data.amount),
            categoryId: parseFloat(data.categoryId),
            description: data.description
        }
    }
    $.ajax({
        type: "POST",
        url: transactionUrl,
        data: JSON.stringify(tmpData),
        contentType: 'application/json',
        success: function (response) {
            console.log(response)
            showSnackbar("success transacion : " + tmpData.newTransaction.type)
            getAccountAfterTransaction(tmpData.newTransaction);
        },
        error: function (error) {
            alert.log("Error Creating user" + error);

        }
    });
}
```

Primary Function

Common.js

```
// update UI .. select, table

function initAccountSelectorUI(data) {…
}

function initAccountIdSelectorUI(data) {…
}

// null check confirm
function initCategorySelect(data) {…
}

// null check confirm
function updataAccountSummary(data) {…
}

//null check confirm
function updataTransationDetail(data, username) {…
}

function updateTableAfterTransaction(data, id){…
}

function getBalance(data){…
}

function showSnackbar(str) {…
}
```

```
function initAccountSelectorUI(data) {
    if (data == null) {
        return;
    }
    //remove duplicate account name
    let newData = [];
    for(let index = 0 ; index < data.length ; index++){
        if (data.map(i => i.username).indexOf(data[index].username) == index) {
            newData.push(data[index]);
        }
    }
    $(".accountSelect").empty();
    const defaultOption = new Option('Choose a Account', '0');
    $(".accountSelect").append(defaultOption);
    newData.forEach((element, index) => {
        const option = new Option(element.username, element.id);
        $(".accountSelect").append(option);
    });

}
```

Primary Function

Common.js

```
// update UI .. select, table

function initAccountSelectorUI(data) {…
}

function initAccountIdSelectorUI(data) {…
}

// null check confirm
function initCategorySelect(data) {…
}

// null check confirm
function updataAccountSummary(data) {…
}

//null check confirm
function updataTransationDetail(data, username) {…
}

function updateTableAfterTransaction(data, id){…
}

function getBalance(data){…
}

function showSnackbar(str) {…
}
```

```
function updataTransationDetail(data, username) {
    // currentID filter
    let newTrans = [];
    data.forEach(element => {
        if(element.username == username){
            element.transactions.forEach(tran => {
                newTrans.push(tran);
            })
        }
    })
    newTrans.sort((a, b) => b.id - a.id);
```

```
for (let i = 0; i < newTrans.length; i++) {
    var userNameFrom = "";
    var userNameTo = ""
    if( newTrans[i].accountIdFrom != null &&  newTrans[i].accountIdTo != null){
        let tmpFrom = data.filter(function (obj) {
            return obj.id == newTrans[i].accountIdFrom ;
        });
        let tmpTo = data.filter(function (obj) {
            return obj.id == newTrans[i].accountIdTo ;
        });
        console.log(tmpFrom);
        userNameFrom = tmpFrom[0].username;
        userNameTo = tmpTo[0].username;
    }
```

Primary Function

Common.js

```
// update UI .. select, table

function initAccountSelectorUI(data) {…
}

function initAccountIdSelectorUI(data) {…
}

// null check confirm
function initCategorySelect(data) {…
}

// null check confirm
function updataAccountSummary(data) {…
}

//null check confirm
function updataTransationDetail(data, username) {…
}

function updateTableAfterTransaction(data, id){…
}

function getBalance(data){…
}

function showSnackbar(str) {…
}
```

```
function updateTableAfterTransaction(data, id){
    let tmpData = data.filter(function (obj) {
        return obj.id == id;
    });
    var username = tmpData[0].username;

    let userData = data.filter(function (obj) {
        return obj.username == username;
    });

    updataTransationDetail(data, username);
    updataAccountSummary(userData);
}
```

```
function getBalance(data){
    var balance = 0;
    if(data.transactions.length != 0){
        for(let i = 0; i < data.transactions.length ; i++){
            if(data.transactions[i].type == "Deposit"){
                balance += data.transactions[i].amount;
            }else if(data.transactions[i].type == "Transfer" && data.transactions[i].accountIdTo == data.id){
                balance += data.transactions[i].amount
            }else if(data.transactions[i].type == "Transfer" && data.transactions[i].accountIdFrom == data.id){
                balance -= data.transactions[i].amount
            }else if(data.transactions[i].type == "Withdraw"){
                balance -= data.transactions[i].amount
            }
        }
    }
    return balance;
}
```

Primary Function

login.js

```
$(() => {
    $("#btnLogin").on("click", function () {…
    });

    $("#btnCreateAccount").on("click", function () {…
    });

})

function addUser(user) {
    var users = getUsers();
    console.log(users);
    if (users.length != 0) {
        let checkUser = users.filter(function (obj) {
            return obj.id == user.id;
        });
        console.log(checkUser);

        if (checkUser.length != 0) {
            showSnackbar("already exsist user")
            return;
        }
        console.log(checkUser);
    }
    // Modifying
    var user = {
        id: user.id,
        pw: user.pw
    };
    users.push(user);
    // Saving
    localStorage.setItem("users", JSON.stringify(users));
}


function getUsers() {
    return JSON.parse(localStorage.getItem('users')) || [];
}
```

```
$("#btnLogin").on("click", function () {
    let loginId = $("#loginId").val();
    let loginPW = $("#loginPW").val();
    let users = getUsers();
    console.log(users)
    if (users.length != 0) {
        let checkUser = users.filter(function (obj) {
            return obj.id == loginId;
        });

        if (checkUser != null) {
            if (checkUser[0].pw != loginPW) {
                showSnackbar("wrong password");
            } else {
                sessionStorage.setItem("loginUser", JSON.stringify(checkUser));
                if (checkUser[0].id == "admin") {
                    console.log("aaa")
                    window.location = "./index.html"
                } else {
                    window.location = "./user.html"
                }
            }
        }
    }
});
```

Primary Function

user.js

```
var username = "";

$(() => {
    const loginUser = JSON.parse(sessionStorage.getItem('loginUser')) || [];
    username = loginUser[0].id;
    console.log(username);
    getUserDashInfo(username);
    $(".transferToBox").hide();


    $("#btnLogout").on("click", function(){
        sessionStorage.clear();
        window.location = "./login.html";
    })
      // menu controll
    $(".userDashboard").on("click", function(){…
    })
    $(".userTransfer").on("click", function(){…
    })
    $(".userTransaction").on("click", function(){…
    })
    $(".userAccount").on("click", function(){…
    })

    //input change
    $('input[type=radio][name=usertransType]').change(function () {…
      });

    $("#btnUserTransfer").on("click", function(){…
      })

      $(".userTransAccount").on("change", function () {…
      })

      $("#btnUserAddAccount").on("click", function(){…
      })
})
```

```
  // menu controll
$(".userDashboard").on("click", function(){
    $(".infoDashboard").show();
    $(".infoTransfer").hide();
    $(".infoTransaction").hide();
    $(".infoAccount").hide();
    getUserDashInfo();

})
$(".userTransfer").on("click", function(){
    $(".infoDashboard").hide();
    $(".infoTransfer").show();
    $(".infoTransaction").hide();
    $(".infoAccount").hide();
    getUserTransfer();

})
$(".userTransaction").on("click", function(){
    $(".infoDashboard").hide();
    $(".infoTransfer").hide();
    $(".infoTransaction").show();
    $(".infoAccount").hide();
    getUserTransaction();


})
$(".userAccount").on("click", function(){
    $(".infoDashboard").hide();
    $(".infoTransfer").hide();
    $(".infoTransaction").hide();
    $(".infoAccount").show();
    getUserAccount();

})
```

Primary Function

user.js

```
function updateUserDashboard(data) {
    var totalBalance = 0;
    var barChart = "";

    $(".userInfo").html(username);
    let userData = data.filter(function (obj) {
        return obj.username == username;
    });
    userData.forEach(function (element, index) {
        totalBalance += getBalance(element);
        barChart += `<div class="accountBarItem` + index + `">` + getBalance(ele
    });

    let newTrans = [];
    userData.forEach(element => {
        if (element.username == username) {
            element.transactions.forEach(tran => {
                newTrans.push(tran);
            })
        }
    })
    newTrans.sort((a, b) => b.id - a.id);
    $(".dash_total").empty();
    $(".dash_accountBar").empty();
    $(".tb_recentTrans").empty();

    //total value
    $(".dash_total").append(`Total Balance <div class="numbers">` + totalBalance.toLocaleString("en-US") + `</div>`)
    gsap.fromTo($(".dash_total"), { opacity : 0 } , { opacity : 1, duration : 1.5})

    //balance chart
    if (totalBalance != 0) {…
    }
    //table
    var txt = `<tr><th>Id</th>…
     </tr>`;
    // generate no record
    if (newTrans.length == 0) {…
    } else {…
    }
    $(".tb_recentTrans").append(txt);
    gsap.fromTo($(".tb_recentTrans"), { opacity : 0 } , { opacity : 1, duration : 1.5});
```

```
//balance chart
if (totalBalance != 0) {
    $(".dash_accountBar").append(barChart);
    for (let i = 0; i < userData.length; i++) {
        let tempWidth = parseFloat($(".dash_accountBar div:eq(" + i + ")").text()) / totalBalance * 100;
        $(".dash_accountBar div:eq(" + i + ")").html(parseFloat($(".dash_accountBar div:eq(" + i + ")").text()).toLoc
        $(".dash_accountBar div:eq(" + i + ")").css({
            "width": tempWidth + "%",
            padding: "1em"
        })
        if (i == 0) {
            $(".dash_accountBar div:eq(" + i + ")").css({
                "background-color": "#141E46",
                color: "white"
            })
        } else if (i == 1) {…
        } else if (i == 2) {…
        } else if (i == 3) {…
        } else if (i == 4) {…
        }
        gsap.fromTo($(".dash_accountBar div:eq(" + i + ")"), { width: 0 }, { width: tempWidth + "%", duration: 1.5 })
    }
}
```

Primary Function

Struggle and learning

Struggle Point

- Ajax type
- Button Type
  ”submit / button”
- data type error from .val()

Learning Point

- localStorage and Session Storage
- efficient function such as sort(), filter(), indexof()

thank you