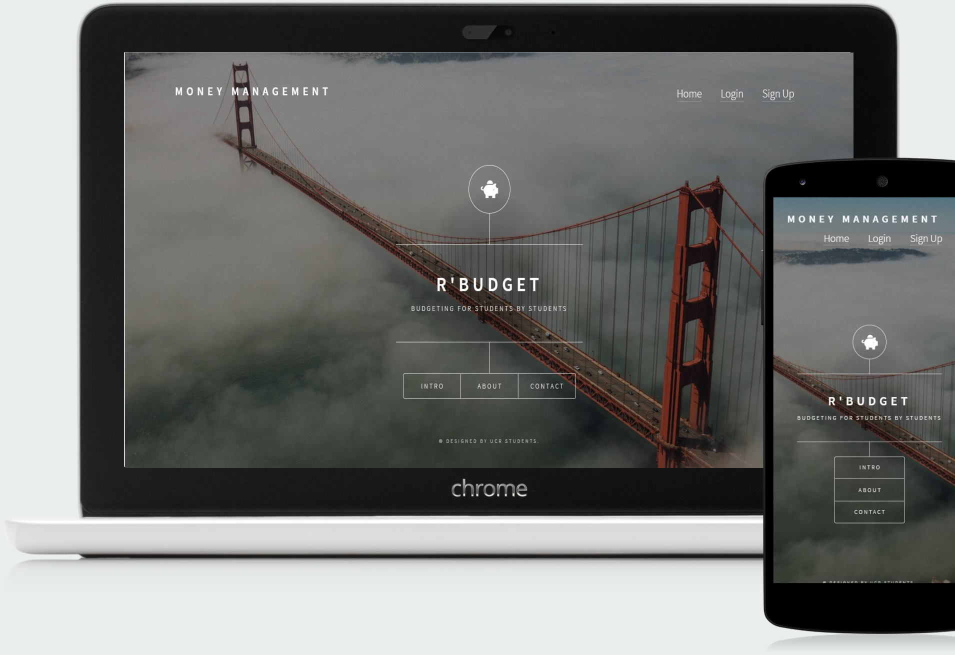
R'Budget

Money Management

Allison Nguyen, Ashley McDaniel, Eric Chaing, Jacques Fracchia



Outline

❖ Introduction

❖ Demo

❖ Major Design

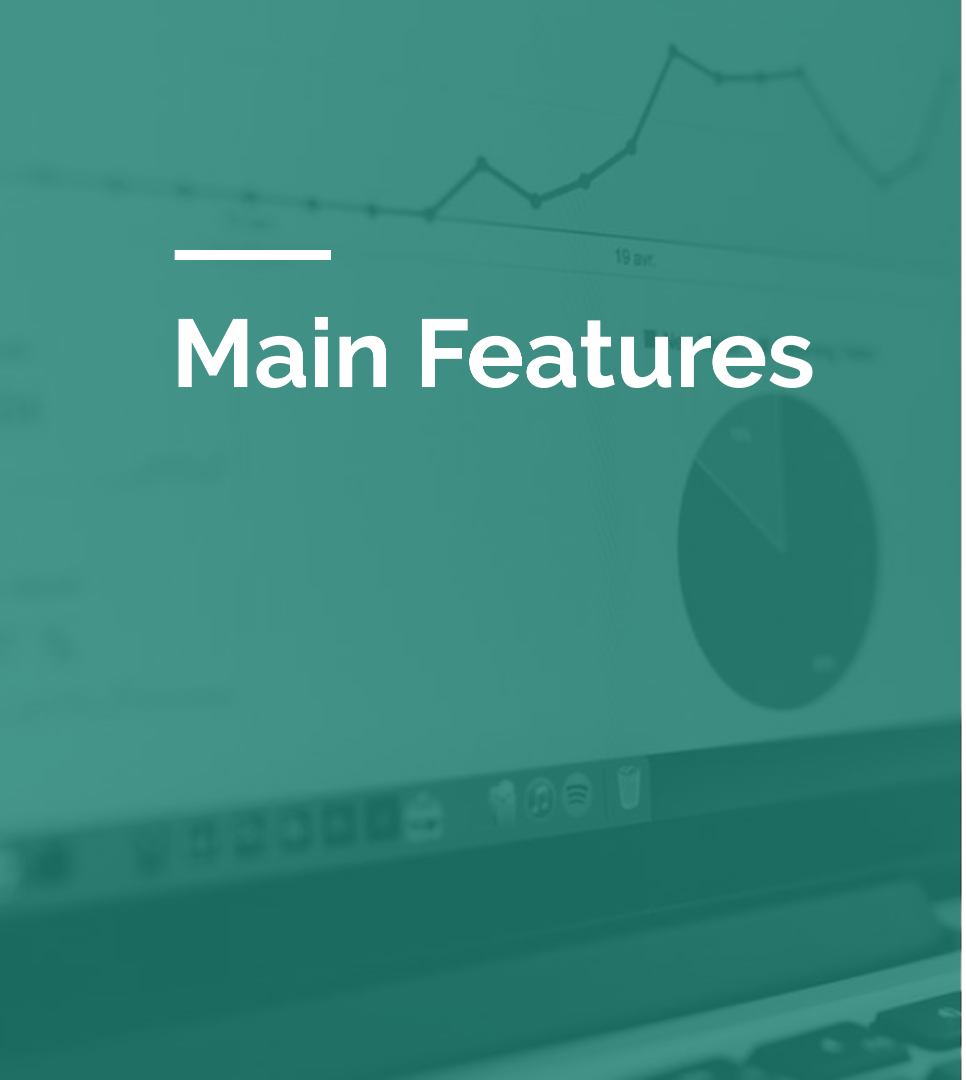
❖ Lessons Learned

❖ Q & A

INTRODUCTION

A Budgeting Solution for Students

R’Budget ensures users that it keeps track of their spending, their budget recommendation percentages, and any goals for special items they are saving up for.



Main Features

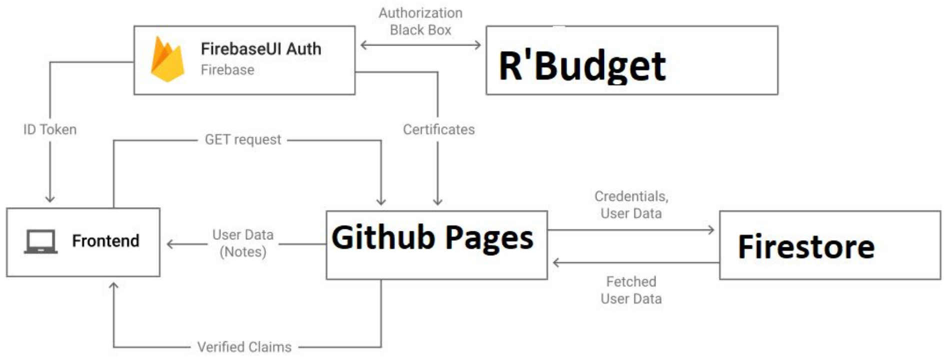
- Automatic Statements
- Analytics
- Goal Tracking
- Personalized Spending

DEMO

https://rbudget.xyz/

MAJOR DESIGN

Technology Stack



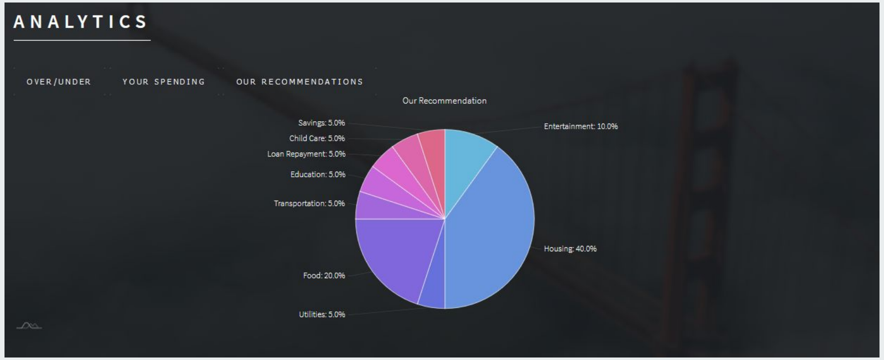
Languages

- HTML5
- CSS3
- Javascript (vanilla, jQuery, Node.js)

Services

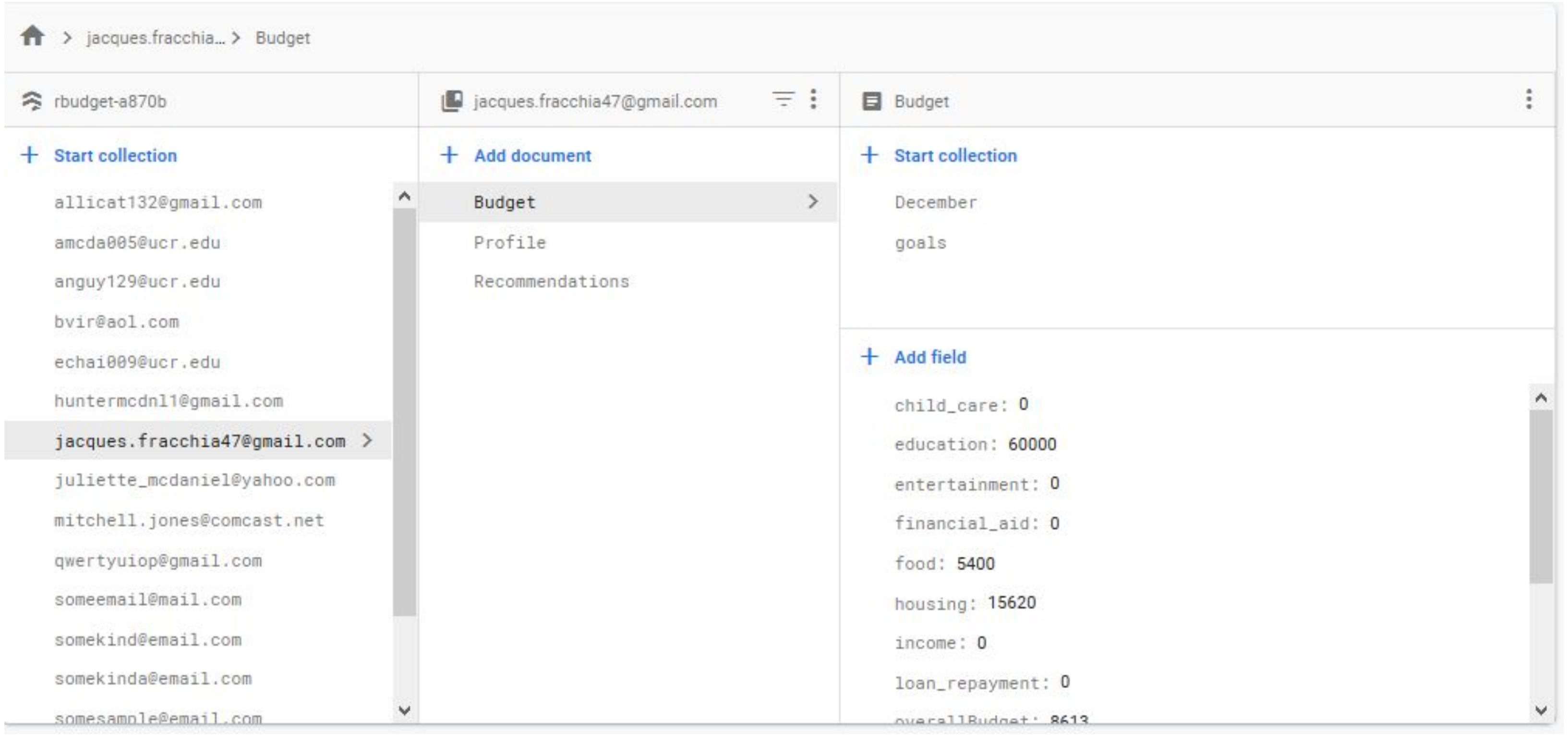
- Firebase (Firestore, Auth, Cloud Functions)
- Github Pages

Analytics:



- Used framework (AMCharts) to create graphs
- Charts are defined as a list of maps
- Using a list of maps allows for easily readable code
  - Creating new categories or changing values is as simple as defining a key-value pair
- The graph generation is separate from the values

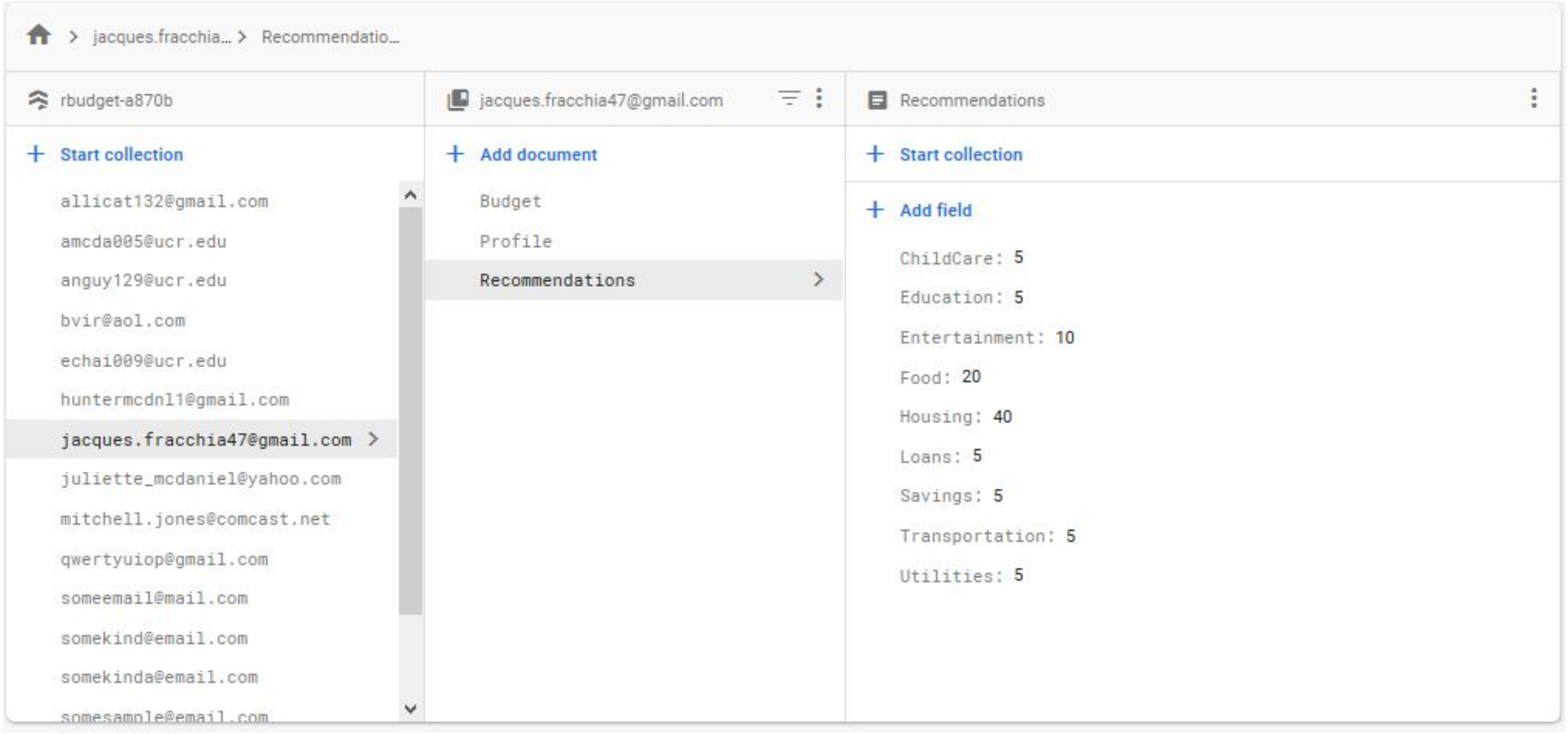
Database Categories:



jacques.fracchia... > Budget

rbudget-a870b	jacques.fracchia47@gmail.com	Budget
Start collection	Add document	Start collection
allicat132@gmail.com	Budget	December
amcda005@ucr.edu	Profile	goals
anguy129@ucr.edu	Recommendations	
bvir@aol.com		Add field
echai009@ucr.edu		child_care: 0
huntermcdnl1@gmail.com		education: 60000
jacques.fracchia47@gmail.com		entertainment: 0
juliette_mcdaniel@yahoo.com		financial_aid: 0
mitchell.jones@comcast.net		food: 5400
qwertyuiop@gmail.com		housing: 15620
someemail@mail.com		income: 0
somekind@email.com		loan_repayment: 0
somekinda@email.com		overallBudget: 8613
somesample@email.com		

Database Recommendations:

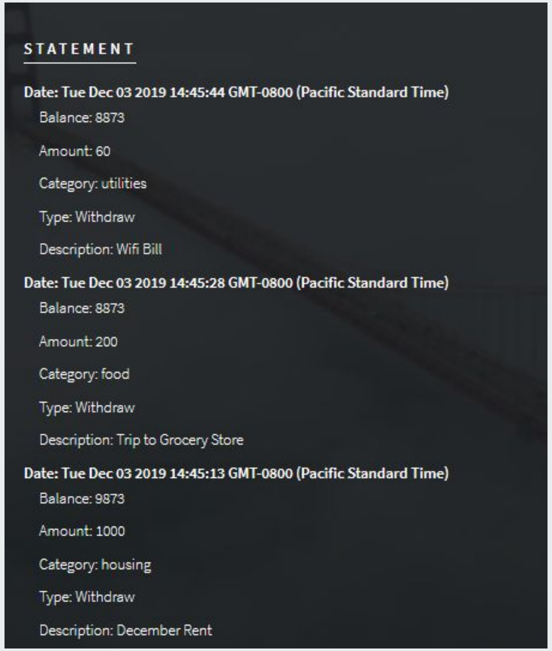


allicat132@gmail.com
amcda005@ucr.edu
anguy129@ucr.edu
bvir@aol.com
echai009@ucr.edu
huntermcdnl1@gmail.com
jacques.fracchia47@gmail.com
juliette_mcdaniel@yahoo.com
mitchell.jones@comcast.net
qwertyuiop@gmail.com
someemail@mail.com
somekind@email.com
somekinda@email.com
somesample@email.com

Budget
Profile
Recommendations

ChildCare:	5
Education:	5
Entertainment:	10
Food:	20
Housing:	40
Loans:	5
Savings:	5
Transportation:	5
Utilities:	5

Statements:



STATEMENT

Date: Tue Dec 03 2019 14:45:44 GMT-0800 (Pacific Standard Time)

Balance: 8873

Amount: 60

Category: utilities

Type: Withdraw

Description: Wifi Bill

Date: Tue Dec 03 2019 14:45:28 GMT-0800 (Pacific Standard Time)

Balance: 8873

Amount: 200

Category: food

Type: Withdraw

Description: Trip to Grocery Store

Date: Tue Dec 03 2019 14:45:13 GMT-0800 (Pacific Standard Time)

Balance: 9873

Amount: 1000

Category: housing

Type: Withdraw

Description: December Rent

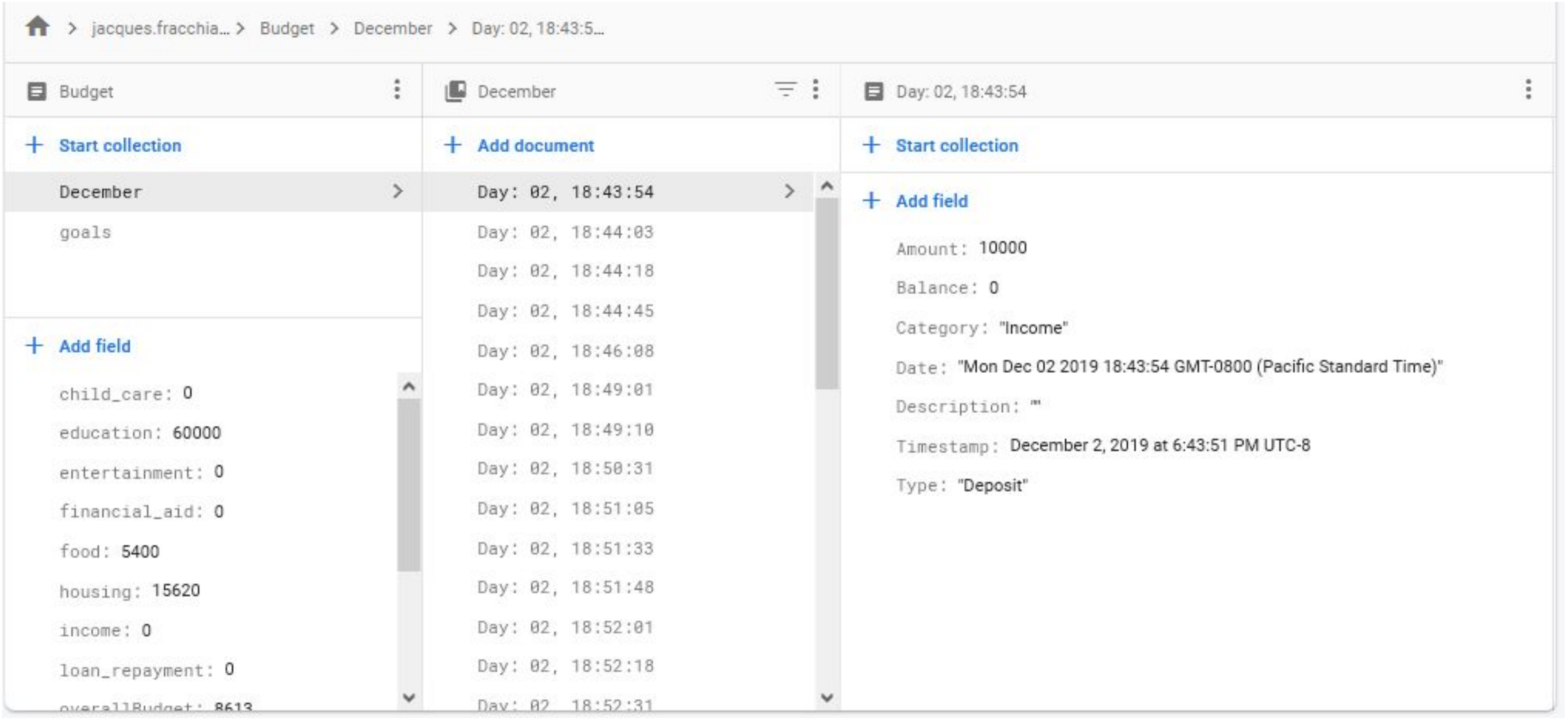
- Get Timestamp in Budget.js in withdraw and deposit: `Timestamp:firebase.firestore.FieldValue.serverTimestamp()`
- Used a function renderStatement(doc)to create elements (Balance/Category/etc.)

```
let lis0 = document.createElement('dd');
lis0.setAttribute('data-id', doc.id);
lis0.setAttribute('id', 'statementlist');
lis0.textContent="Balance:"+doc.data().Balance
;
statementlist.appendChild(lis0);
```

- Call renderStatement() in Statement()
- Get Data from Collection in: `db.collection(user_email).doc("Budget").collection...`
- Snapshot allows us to cycle through each document

```
snapshot.docs.forEach(doc => {
                    renderStatement(doc);
```

Database Statements:



jacques.fracchia... > Budget > December > Day: 02, 18:43:5...

Budget

Start collection

December

goals

Add field

child_care: 0

education: 60000

entertainment: 0

financial_aid: 0

food: 5400

housing: 15620

income: 0

loan_repayment: 0

overallBudget: 8613

December

Add document

Day: 02, 18:43:54

Day: 02, 18:44:03

Day: 02, 18:44:18

Day: 02, 18:44:45

Day: 02, 18:46:08

Day: 02, 18:49:01

Day: 02, 18:49:10

Day: 02, 18:50:31

Day: 02, 18:51:05

Day: 02, 18:51:33

Day: 02, 18:51:48

Day: 02, 18:52:01

Day: 02, 18:52:18

Day: 02, 18:52:31

Day: 02, 18:43:54

Start collection

Add field

Amount: 10000

Balance: 0

Category: "Income"

Date: "Mon Dec 02 2019 18:43:54 GMT-0800 (Pacific Standard Time)"

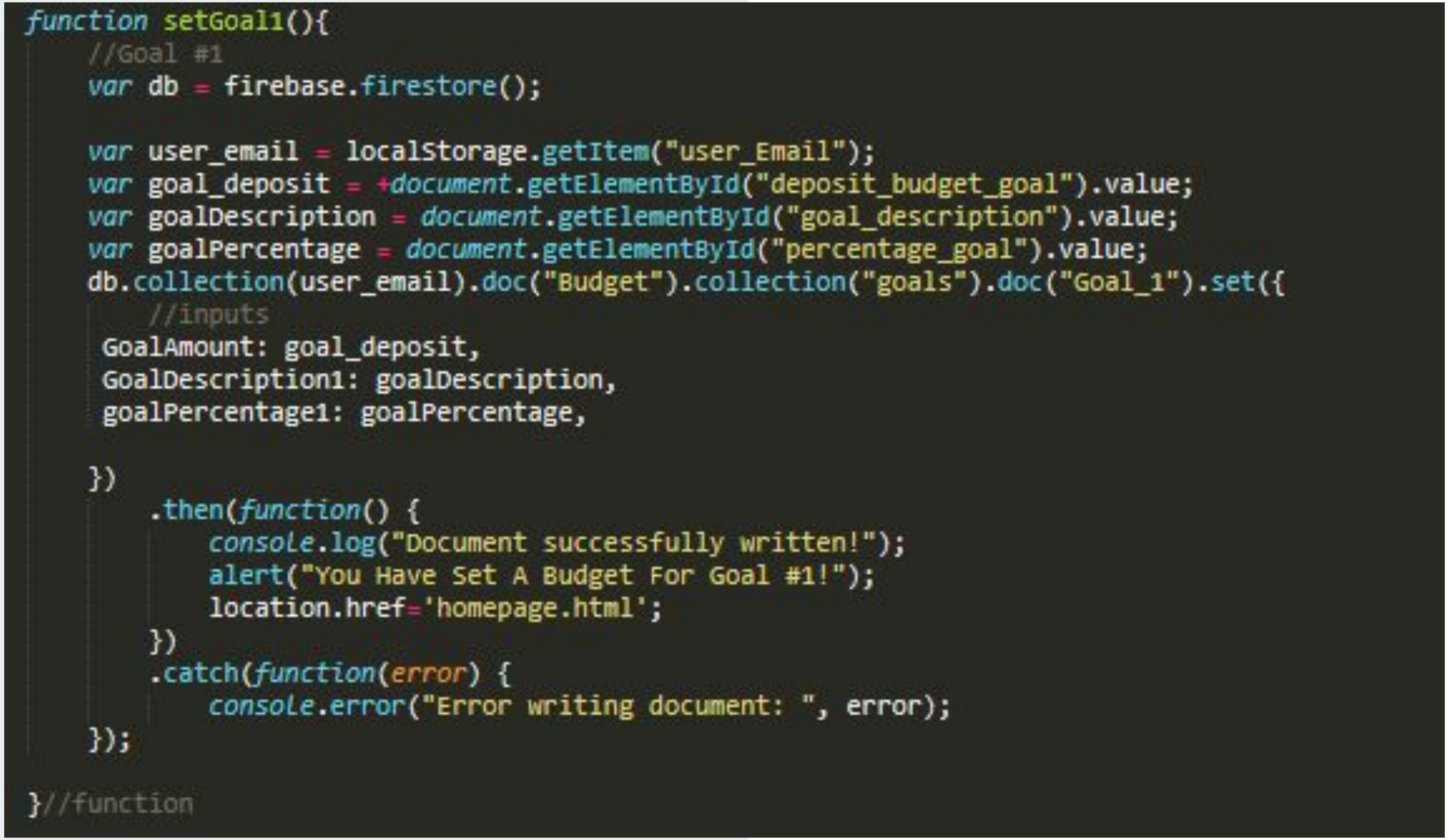
Description: ""

Timestamp: December 2, 2019 at 6:43:51 PM UTC-8

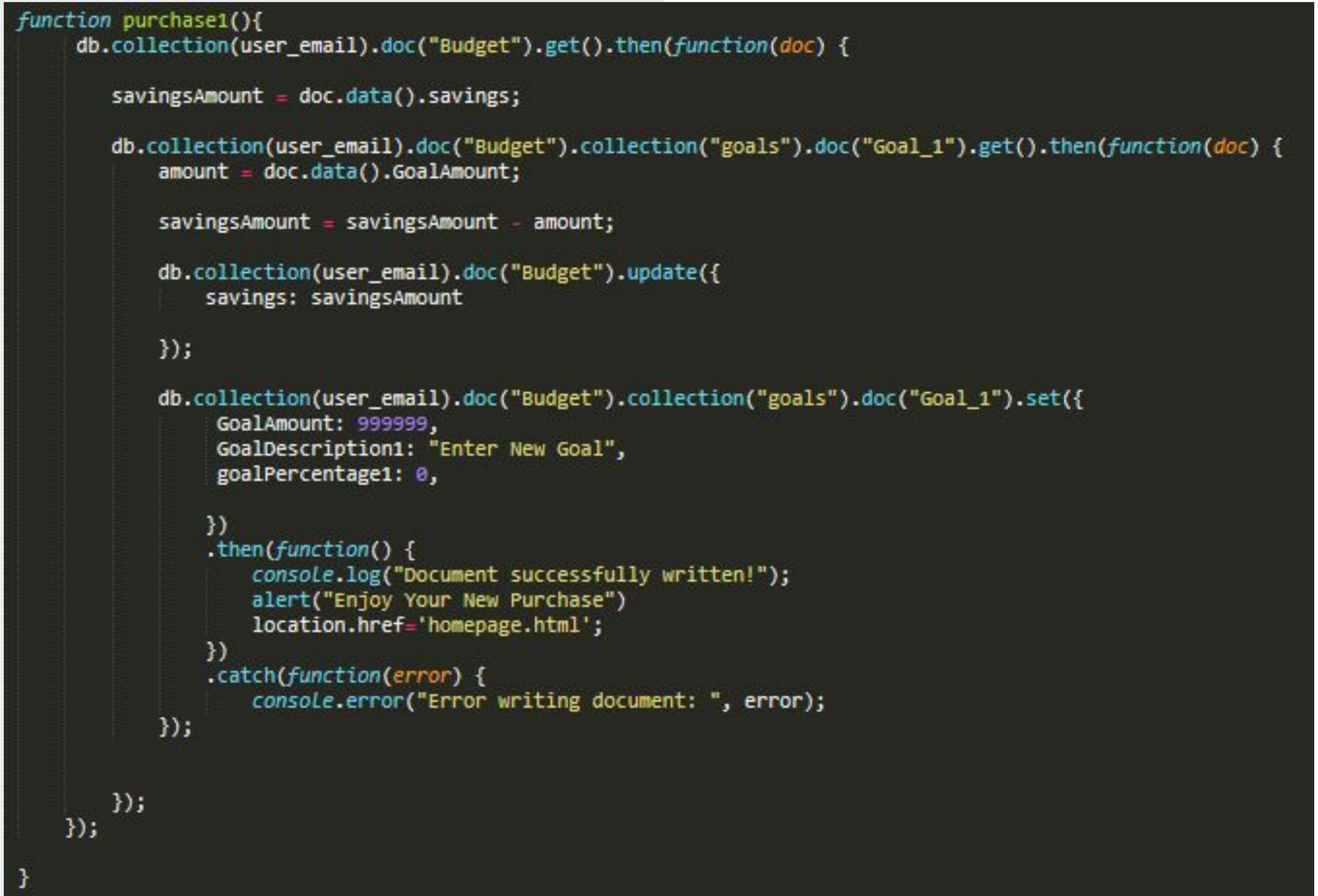
Type: "Deposit"

Goals:

- Checks the database every time the user withdraws money for their savings.
- If the savings amount is equal to the amount the user wishes to save for, the goals page will notify them.
- Once the user has purchased that item they can click ‘purchased’ which will withdraw that amount from their savings.

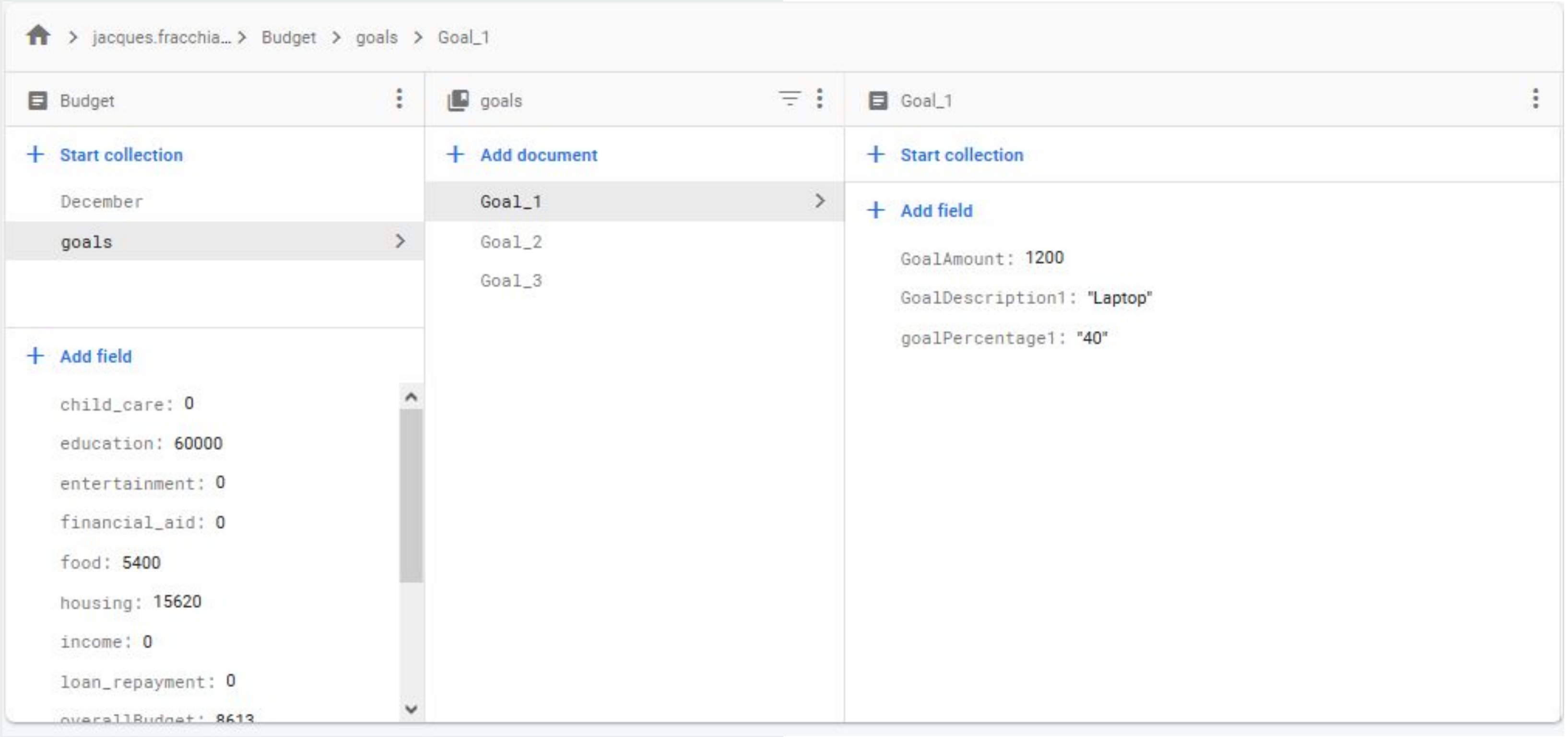


```javascript
function setGoal1(){
    //Goal #1
    var db = firebase.firestore();

    var user_email = localStorage.getItem("user_Email");
    var goal_deposit = +document.getElementById("deposit_budget_goal").value;
    var goalDescription = document.getElementById("goal_description").value;
    var goalPercentage = document.getElementById("percentage_goal").value;
    db.collection(user_email).doc("Budget").collection("goals").doc("Goal_1").set({
        //inputs
     GoalAmount: goal_deposit,
     GoalDescription1: goalDescription,
     goalPercentage1: goalPercentage,

    })
        .then(function() {
            console.log("Document successfully written!");
            alert("You Have Set A Budget For Goal #1!");
            location.href='homepage.html';
        })
        .catch(function(error) {
            console.error("Error writing document: ", error);
    });

}//function
```



```
function purchase1(){
    db.collection(user_email).doc("Budget").get().then(function(doc) {

        savingsAmount = doc.data().savings;

        db.collection(user_email).doc("Budget").collection("goals").doc("Goal_1").get().then(function(doc) {
            amount = doc.data().GoalAmount;

            savingsAmount = savingsAmount - amount;

            db.collection(user_email).doc("Budget").update({
                savings: savingsAmount

            });

            db.collection(user_email).doc("Budget").collection("goals").doc("Goal_1").set({
                 GoalAmount: 999999,
                 GoalDescription1: "Enter New Goal",
                 goalPercentage1: 0,

                })
                .then(function() {
                    console.log("Document successfully written!");
                    alert("Enjoy Your New Purchase")
                    location.href='homepage.html';
                })
                .catch(function(error) {
                    console.error("Error writing document: ", error);
            });


        });
    });

}
```

Database Goals:



jacques.fracchia... > Budget > goals > Goal_1

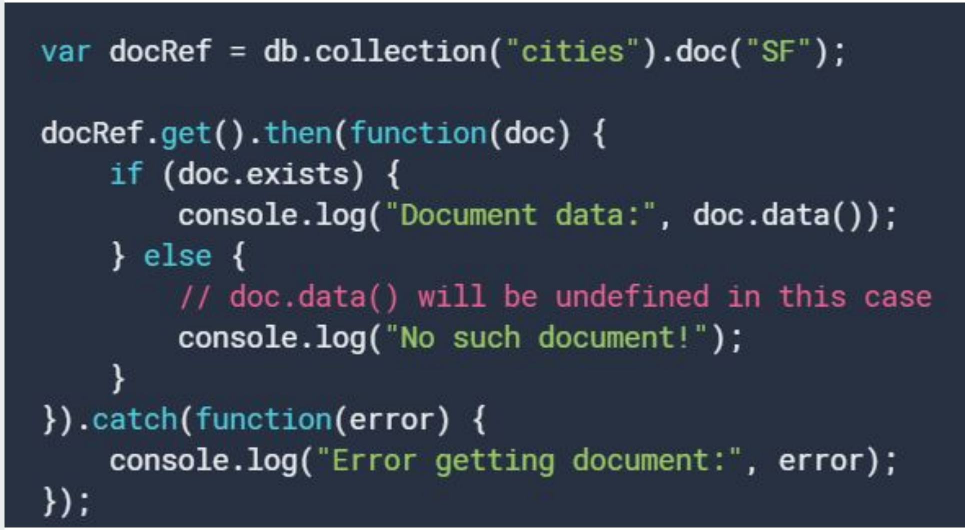
Budget	goals	Goal_1
+ Start collection	+ Add document	+ Start collection
December	Goal_1	+ Add field
goals	Goal_2	GoalAmount: 1200
	Goal_3	GoalDescription1: "Laptop"
+ Add field		goalPercentage1: "40"
child_care: 0		
education: 60000		
entertainment: 0		
financial_aid: 0		
food: 5400		
housing: 15620		
income: 0		
loan_repayment: 0		
overallBudget: 8613		

A Quick Note About Asynchronicity

The nature of the modern internet is asynchronous; in order to maintain a quick and responsive experience.

Asynchronous function

- A function that resolves at a different time than other pieces of code it lives with.



```
var docRef = db.collection("cities").doc("SF");

docRef.get().then(function(doc) {
    if (doc.exists) {
        console.log("Document data:", doc.data());
    } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
    }
}).catch(function(error) {
    console.log("Error getting document:", error);
});
```

Lessons Learned

- Making clear user stories prevents any ambiguity for each feature.
- Working with numbers assures everything must function 100%
- Determining how the Database is set up made creating the front end much easier.
- Always leave time for fixing bugs!

Thank you for listening.

Questions?