Team manager : Tamir Baruch

Team members:

Tamir Baruch, with ID number 316441716, can be reached at the email address tamirb10@gmail.com, phone number : 054-2191377.

Omri Asher, with ID number 316224997, can be contacted via email at omriash4@gmail.com , phone .number : 055-6655474.

Link to the video:

https://youtu.be/bEx6GSoXw_w

models/about.js

```
const data = [
  {
    firstname: "Omri",
    lastname: "Asher",
    id: 316224997,
    email: "omriash4@gmail.com"
  },
  {
    firstname: "Tamir",
    lastname: "Baruch",
    id: 316441716,
    email: "tamirb10@gmail.com"
  }
];
module.exports = data;
```

models/cost.js

```
const mongoose = require('mongoose');

const costSchema = new mongoose.Schema({

  user_id:{
    type:String
  },
  year: {
    type:Number,
    required:true,
    min:0
  },
  month:{
    type:Number,
    required:true,
    min:1,
    max:12
  },
  day:{
    type:Number,
    required:true,
    min:1,
    max:31
  },
  description:{
    type:String,
    required:true

  },
  category:{
    type: String,
    enum:['food','health','housing','sport',
      'education','transportation','other'],
    lowercase:true
  },
  sum:{
    type:Number,
    required:true
  }

})

const Cost = mongoose.model('Cost',costSchema);
module.exports = Cost;
```

module/report.js

```
const mongoose = require('mongoose');

const costSchema = new mongoose.Schema({
   year: {
      type:Number,
      required:true,
      min:0
   },
   month:{
      type:Number,
      required:true,
      min:1,
      max:12
   },
   user_id:{
      type:String,
      required:true
   },
   food: { type: Array, default: [] },
   health: { type: Array, default: [] },
   housing: { type: Array, default: [] },
   sport: { type: Array, default: [] },
   education: { type: Array, default: [] },
   transportation: { type: Array, default: [] },
   other: { type: Array, default: [] },
});

const Report = mongoose.model('Report', costSchema);
module.exports = Report;
```

module/user.js

```
const mongoose = require('mongoose');

const costSchema = new mongoose.Schema({

   id:{
      type:String
   },
   first_name: {
      type:String,
      required:true,

   },
   last_name:{
      type:String,
      required:true,

   },
   birthday: {
      day: {
         type: Number,
         required: true
      },
      month: {
         type: Number,
         required: true
      },
      year: {
         type: Number,
         required: true
      }
   }



})

const User = mongoose.model('Users',costSchema);
module.exports = User;
```

Index.js

```
const data=require('./models/about');
const express = require('express')
const app = express();
const path = require('path');
const mongoose = require('mongoose');
const Cost=require('./models/cost');
const Report=require('./models/report');
const User=require('./models/user');

const methodOverride = require('method-override');

mongoose.connect("mongodb+srv://tamirb10:b9649263@cluster0.3tt7yh6." +
   "mongodb.net/?retryWrites=true&w=majority",
   {useNewUrlParser:true,useUnifiedTopology:true})
   .then(()=> {
      console.log("MONGO CONNECTION WORK")
   }).catch(err => {
   console.log("MONGO Not working")
   console.log(err)
})

const categories = ['food','health','housing','sport'
   ,'education','transportation','other']



app.set('views',path.join(__dirname,'views'));
app.set('view engine','ejs');
app.use(express.urlencoded({extended:true}));
app.use(methodOverride('_method'));



app.get('/costs',async (req,res)=>{
   const {month} = req.query;

   if(month)
   {
      const costs = await Cost.find({month})
      res.render('costs/index',{costs})
   }
   else
   {
      const costs = await Cost.find({})
      res.render('costs/index',{costs})
   }


})

app.get('/about', (req,res)=> {

   res.send(data);
})

app.get('/costs/report', (req,res)=>{

  res.render('costs/report');

})
```

```
app.post('/costs/report', async (req, res) => {

  try {


    const {user_id, year, month} = req.body;
    const findReport = await Report.find(req.body)
    if (findReport.length > 0) {


      const catego = {};

      for (let report of findReport) {
        for (let cate of categories) {
          catego[cate] = report[cate];
        }
      }


      res.send(catego);
    } else {


      const cost2 = []
      const users = await User.find({id:user_id});
      if (users.length<1)
      {
        throw new Error("user not found")
      }
      const costs = await Cost.find({user_id});
      if (costs.length > 0 ) {
        for (const cost of costs) {

          if (cost.year == year && cost.month == month) {

            cost2.push(cost)
          }
        }

      }


      const reported = make_it_report_right(cost2);

      const parsedReported = JSON.parse(reported);

      const report = new Report(parsedReported);
      report.user_id = user_id;
      report.year = year;
      report.month = month;
      await report.save();
      res.send(reported);

    }
  }
```

```
    catch (err)
    {
        res.status(500).json({ error: err.message });

    }


});


function make_it_report_right(costs) {
    const catego = {};

    for (let cate of categories) {
        catego[cate] = [];
        for (let cost of costs) {
            if (cate === cost.category) {
                catego[cate].push({
                    day: cost.day,
                    description: cost.description,
                    sum: cost.sum
                });
            }
        }
    }

    return JSON.stringify(catego);
}

app.get('/costs/addcost',(req,res)=>
{

    res.render('costs/new',{categories})
})
```

```
app.post('/costs/addcost',async (req,res)=>
{
   try {
      const newcost = new Cost(req.body);
      const users = await User.find({id:newcost.user_id});


         if (users.length>0) {
            await newcost.save()
            const report = await Report.find({user_id:newcost.user_id,
               year: newcost.year, month: newcost.month});

            if (report.length > 0) {
               const {user_id, year, month} = req.body;
               await Report.deleteMany({user_id, year, month});
            }


            res.send(newcost)
         } else {
            res.send({
                  "error": {
                     "code": 404,
                     "message": "User not found",
                     "details": "The requested user does
                      not exist in the system."
                  }
               }
            )

         }


   } catch (err) {
      res.status(500).json({ error: err.message });

   }
})


app.get('/costs/:id',async (req,res)=>{
 const {id}=req.params;
 const cost =await Cost.findById(id);
 res.render('costs/show',{cost});
})


app.get('/costs/:id/edit',async(req,res)=>
{
   const {id}=req.params;
   const cost =await Cost.findById(id);
   res.render('costs/edit',{cost,categories})
})

app.put('/costs/:id',async (req,res)=>
{
   const {id}=req.params;
   await Cost.findByIdAndUpdate(id,req.body, {runValidators:true,
   new :true});
   res.redirect("/costs")
})
```

```
app.delete('/costs/:id',async (req,res)=>{
    const {id}=req.params;
    await Cost.findByIdAndDelete(id);
    res.redirect("/costs")

})


app.listen(8080,()=>{
    console.log("listening on port 8080!");
})
```