```
1
2   var express = require("express");
3   var morgan = require("morgan");
4   var bodyParser = require("body-parser");
5   var mongoose = require("mongoose");
6   var port = 3000;
7
8   var app = express();
9   var Agenda = require("./models/model.js");
10
11  mongoose.connect("mongodb://woutdt:Wout2002@ds135290.mlab.com:35290/agenda",
    function(err) {
12    if(err) throw err;
13    console.log("database connected");
14  });
15
16  app.use(bodyParser.json());
17  app.use(bodyParser.urlencoded({ extended: true }));
18
19  var errorHandler = {
20    "message": "Oops, something went wrogn..."
21  };
22
23  //check array values are equal
24  Array.prototype.AllValuesSame = function(){
25
26      if(this.length > 0) {
27          for(var i = 1; i < this.length; i++)
28          {
29              if(this[i] !== this[0])
30                  return false;
31          }
32      }
33      return true;
34  }
35
36  //find all
37  function findAll(res) {
38       Agenda.find({}, function(err, data) {
39        if(err) res.json({"message": "oops something went wrogn..."});
40        return res.json(data);
41      });
42  };
43
44  //find find one by task
45  function findOneByTask(req, res) {
46    Agenda.findOne({'taken._id': new mongoose.Types.ObjectId(req.query.id) },
      function(err, data) {
47      return res.json(data);
48    });
49  };
50
51  //find one
52  function findOne(req, res) {
53    Agenda.findOne({'_id': req.query.id}, function(err, data) {
54      if(err) return res.json(errorHandler);
55      return res.json(data);
56    });
57  };
58
59  function timeConversion(millisec) {
60
61          var seconds = (millisec / 1000).toFixed(1);
62
63          var minutes = (millisec / (1000 * 60)).toFixed(1);
64
65          var hours = (millisec / (1000 * 60 * 60)).toFixed(1);
66
67          var days = (millisec / (1000 * 60 * 60 * 24)).toFixed(1);
```

```
68
69          if (seconds < 60) {
70              return seconds + " Sec";
71          } else if (minutes < 60) {
72              return minutes + " Min";
73          } else if (hours < 24) {
74              return hours + " Hours";
75          } else {
76              return days + " Days"
77          }
78      }
79
80  function returnTime(data) {
81    var date1 = data.created_At;
82    var date2 = Date.now();
83    var date3 = date2 - date1;
84    var final = timeConversion(date3) + ' Ago';
85    data.time_created = final;
86    if(data )
87    var l;
88    for(l = 0; l < data.taken.length; l++) {
89      taak = data.taken[l];
90      var datum1 = taak.deadline;
91      var datum2 = Date.now();
92      var datum3 = datum1 - datum2;
93      var finaldate = 'Deadline: '+timeConversion(datum3);
94      taak.time_until_deadline = finaldate;
95      };
96  };
97
98  //get all vakken
99  app.get("/find", function(req, res) {
100   if(req.query.id) {
101     Agenda.findOne({ '_id': req.query.id }, function(err, data) {
102       if(err) return res.json(errorHandler);
103       var date1 = data.created_At;
104       var date2 = Date.now();
105       var date3 = date2 - date1;
106       var final = timeConversion(date3) + ' Ago';
107       data.time_created = final;
108       var l;
109       for(l = 0; l < data.taken.length; l++) {
110         taak = data.taken[l];
111         var datum1 = taak.deadline;
112         var datum2 = Date.now();
113         var datum3 = datum1 - datum2;
114         var finaldate = 'Deadline: '+ timeConversion(datum3);
115         taak.time_until_deadline = finaldate;
116       };
117       return res.json(data);
118     });
119   } else {
120     Agenda.find({}, function(err, data) {
121       if(err) return res.json(errorHandler);
122       var i;
123       for(i = 0; i < data.length; i++) {
124         vak = data[i];
125         var date1 = vak.created_At;
126         var date2 = Date.now();
127         var date3 = date2 - date1;
128         var final = timeConversion(date3) + ' Ago';
129         vak.time_created = final;
130         var l;
131         for(l = 0; l < data[i].taken.length; l++) {
132           taak = data[i].taken[l];
133           var datum1 = taak.deadline;
134           var datum2 = Date.now();
135           var datum3 = datum1 - datum2;
136           var finaldate = 'Deadline: '+ timeConversion(datum3);
```

```
137             taak.time_until_deadline = finaldate;
138           };
139         };
140         return res.json(data);
141       });
142     };
143   });
144
145   //create new vak
146   app.post("/new", function(req, res) {
147     var date = Date.now();
148     var agenda = new Agenda({
149       vak: req.body.vak,
150       important: req.body.important,
151       created_At: date
152     });
153
154     agenda.save(function(err) {
155       if(err) res.json({"message": "sorry something went wrogn..."});
156       Agenda.find({}, function(err, data) {
157         if(err) res.json({"message": "sorry something went wrogn..."});
158         res.json(data);
159       });
160     });
161   });
162
163   //create new task
164   app.post("/newTask", function(req, res) {
165     var millisec = Number(req.body.millisec) + Number(Date.now());
166     var takenObj = { naam: req.body.taakname, deadline: millisec };
167     Agenda.findOneAndUpdate({ _id: req.query.id}, { $push: { taken: takenObj }},
      function(err, data) {
168       Agenda.findOne({_id: req.query.id}, function(err, data) {
169         if(err) {
170           console.log(err);
171           res.json({"message": "sorry something went wrogn..."});
172         } else {
173           res.json(data);
174         };
175       });
176     });
177   });
178
179   //update task => true/false
180   app.put("/taskSwitch", function(req, res) {
181     Agenda.aggregate([
182       { "$unwind" : '$taken' },
183       { "$match" :
184           { "taken._id" : new mongoose.Types.ObjectId(req.query.id) }
185       },
186       { "$limit": 1 }
187     ], function(err, data) {
188       data = data[0];
189       if(data.taken.completed == true) {
190         Agenda.update({ 'taken._id': req.query.id }, { '$set': { 'taken.$.completed':
          false, 'taken.$.show': true}}, function(err, data) {
191           if(err) return res.json(errorHandler);
192           findOneByTask(req, res);
193         });
194       } else if(data.taken.completed == false) {
195         Agenda.aggregate([
196           { '$match':
197               { 'taken.completed': true }
198           }
199         ], function(err, data) {
200           if(err) res.json(errorHandler);
201           var i;
202           var analyse = [];
203           for(i = 0; i < data.length; i++) {
```

```
204             taken = data[i].taken;
205             var l;
206             for(l = 0; l < taken.length; l++) {
207               if(taken[l].completed == true) {
208                 analyse.push(taken[l].show);
209               };
210             };
211           };
212           var analysys = analyse.AllValuesSame();
213           if(analysys == true && analyse[0]==true) {
214             var setShow = true;
215           } else if(analysys == true && analyse[0]==false) {
216             var setShow = false;
217           } else if(analysys == false) {
218             res.json(errorHandler);
219           } else {
220             var setShow = true;
221           };
222           if(setShow == true) {
223             Agenda.update({ 'taken._id': req.query.id }, { '$set': { 'taken.$.completed':
              true, 'taken.$.show': true }}, function(err, data) {
224                if(err) return res.json(errorHandler);
225                findOneByTask(req, res);
226              });
227           } else {
228             Agenda.update({ 'taken._id': req.query.id }, { '$set': { 'taken.$.completed':
              true, 'taken.$.show': false }}, function(err, data) {
229                if(err) return res.json(errorHandler);
230                findOneByTask(req, res);
231              });
232           }
233         });
234       } else {
235         return res.json(errorHandler);
236       };
237     });
238   });
239
240   //hide/show completed tasks
241   app.put("/toggleshow", function(req, res) {
242     Agenda.aggregate([
243       { '$match':
244         { 'taken.completed': true }
245       }
246     ], function(err, data) {
247       if(err) return res.json(errorHandler);
248       var i;
249       for(i = 0; i < data.length; i++) {
250         var taken = data[i].taken;
251         var l;
252         for(l = 0; l < taken.length; l++) {
253           if(taken[l].completed == true) {
254             if(taken[l].show == true) {
255               Agenda.update({ 'taken._id': taken[l]._id }, { '$set': { 'taken.$.show':
                false }}, function(err) {
256                 if(err) return res.json(errorHandler);
257               });
258             } else if(taken[l].show == false) {
259               Agenda.update({ 'taken._id': taken[l]._id }, { '$set': { 'taken.$.show':
                true }}, function(err) {
260                 if(err) return res.json(errorHandler);
261               });
262             } else {
263               Agenda.update({ 'taken._id': taken[l]._id }, { '$set': { 'taken.$.show':
                true}}, function(err) {
264                 if(err) res.json(errorHandler);
265               });
266             };
267           } else if(taken[l].completed == false) {
```

```
            console.log("false");
          };
        };
      };
      Agenda.find({ 'taken.completed': true }, function(err, data) {
        if(err) res.json(errorHandler);
        res.json(data);
      });
    });
  });

  //permanent delete a vak
  app.delete("/delVak", function(req, res) {
    Agenda.deleteOne({'_id': req.query.id}, function(err) {
      if(err) res.json({"message": "oops something went wrogn"});
      findAll(res);
    });
  });

  //soft delete vak
  app.put("/softDelVak", function(req, res) {
    Agenda.findOne({ '_id': req.query.id }, function(err, data) {
      if(err) return res.json(errorHandler);
      if(data.deleted == false) {
        data.deleted = true;
        data.save(function(err) {
          if(err) return res.json(errorHandler);
          findOne(req, res);
        });
      } else if(data.deleted == true) {
        data.deleted = false;
        data.save(function(err) {
          if(err) return res.json(errorHandler);
          findOne(req, res);
        });
      } else {
        return res.json(errorHandler);
      };
    });
  });

  //recover all softdeleted objects
  app.put("/recover", function(req, res) {
    Agenda.aggregate([
      { '$match':
            { 'deleted': true }
      }
    ], function(err, data) {
      var length = data.length;
      var i;
      for (i = 0; i < length; i++) {
        vak = data[i];
        Agenda.update({ '_id': vak._id }, { '$set': { 'deleted': false }}, function(err) {
          if(err) return res.json(errorHandler);
        });
      };
      findAll(res);
    });
  });

  //get all softdeleted objects
  app.get("/litterbin", function(req, res) {
    Agenda.aggregate([
      { '$match':
        { 'deleted' : true }
      }
    ], function(err, data) {
      if(err) return res.json(errorHandler);
      if(data.length == 0) {
```

```
        res.json({ "message": "no objects in  litterbin" });
      } else {
        res.json(data);
      };
    });
  });

  //delete task
  app.delete("/delTask", function(req, res) {
    Agenda.findOne({'taken._id': req.query.id}, function(err, data) {
      var dataid = data._id;
      if(err) {
        res.json(errorHandler);
      } else {
        Agenda.update({'taken._id': new mongoose.Types.ObjectId(req.query.id)}, { $pull :
        { taken :  { _id: new mongoose.Types.ObjectId(req.query.id) }}}, function(err) {
          if(err) res.json({"message": "sorry, something went wrogn"});
        });
        Agenda.findOne({'_id': new mongoose.Types.ObjectId(dataid)}, function(err, docs) {
          if(err) res.json(errorHandler);
          res.json(docs);
        });
      }
    });
  });

  //update vak to important
  app.put("/vakSwitch", function(req, res) {
    Agenda.findOne({'_id': req.query.id}, function(err, data){
      if(err) res.json(errorHandler);
      if(data.important == true) {
        data.important = false;
        data.save(function(err) {
          if(err) res.json(errorHandler);
          findOne(req, res);
        });
      } else if(data.important == false) {
        data.important = true;
        data.save(function(err) {
          if(err) res.json(errorHandler);
          findOne(req, res);
        });
      } else {
        res.json(errorHandler);
      };
    });
  });

  app.listen(port, function(err) {
    if(err) {
      console.log(err);
    } else {
      console.log("server running on port: "+port);
    }
  });

```