

# Summary of Technology

## 1 Technology Overview

Name		Description
Database		
	MongDB	document-based database
	Mongoose	an Object Data Modeling (ODM) library for MongoDB and Node.js
Back-end		
	Node.JS	uses JavaScript on the web server
	Express	a Node.js web application.
Front-end		
	React	A JavaScript library for building user interfaces
	Material-UI	UI Components Framework
	Axios	Promise based HTTP client for the browser and node.js
Useful tools		
	Swagger	A professional toolset to help developers design, build, document, and test RESTful Web services
	mocha	JavaScript test framework
	chai	a BDD / TDD assertion library
	marge	Marge ( <b>mocha</b> awesome-report-generator) generates a report that helps visualize your test suites.

## 2 MongoDB & Mongoose

MongoDB works on concept of collection and document. The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key <code>_id</code> provided by mongodb itself)

The screenshot shows the Robo 3T interface. On the left, the 'Collections' list under 'datapoint' includes 'organizations'. A blue box labeled 'Database' points to the 'datapoint' folder, and another blue box labeled 'Collection' points to the 'organizations' collection. The main window displays the 'organizations' collection with a table of documents. A blue box labeled 'Document' points to the first document. A blue box labeled 'Field' points to the 'name' field. A blue box labeled 'Embedded Documents' points to the 'managers' field, which contains an array of documents. A blue box labeled 'Primary Key' points to the '\_id' field. A red line connects the 'Primary Key' box to the 'organization.js' file in the code editor below.

_id	groupNumber	name	_v	managers	types	users
1	1	King Hospital	0	[ 1 element ]	[ 1 element ]	[ 1 element ]
2	1	IT	0	[ 2 elements ]	[ 2 elements ]	[ 2 elements ]

Mongoose is used to translate between objects in code and the representation of those objects in MongoDB.

The screenshot shows a VS Code editor with a file named 'organization.js'. The code defines a Mongoose schema for an organization. A red line connects the 'Primary Key' box from the previous image to the 'mongoose.Schema.Types.ObjectId' in the code. A blue line connects the 'Embedded Documents' box from the previous image to the 'users' and 'managers' arrays in the schema definition. The code is as follows:

```
1 const mongoose = require('mongoose');
2 const ObjectId = mongoose.Schema.Types.ObjectId;
3
4 const organizationSchema = new mongoose.Schema( definition: {
5   groupNumber: {type: Number, required: true},
6   name: {type: String, required: true},
7   users: [{type: ObjectId, ref: 'User'}],
8   managers: [{type: ObjectId, ref: 'User'}],
9   types: [{type: ObjectId, ref: 'OrganizationType'}],
10 });
11
12 module.exports = mongoose.model( name: 'Organization', organizationSchema);
13
```

## 3 Node.js & Express

A simple example to show how to retrieve all organization documents on the database and return them to the API caller.

```
const express = require('express');
let router = express.Router(); // Create a new router object
const Organization = require('../models/organization/organization');

router.get( path: '/api/v2/organizations', handlers: async (req, res, next) => {
  try {
    const organizations = await Organization.find();
    if (organizations[0]) {
      return res.status( code: 200).json( body: {organizations});
    }
    return res.status( code: 204).json();
  } catch (e) {
    next(e);
  }
}
```

Request URL

`http://localhost:3000/api/v2/organizations`

Server response

Code	Details
------	---------

200

Response body

```
{
  "organizations": [
    {
      "users": [
        "5d5eafb3cc3b85c1c5cc81dc8"
      ],
      "managers": [
        "5d5eafb3cc3b85c1c5cc81dc8"
      ],
      "types": [
        "5d603977d7ac99d4a4f4d0ce"
      ],
      "_id": "5d6443d0adffa653832f057d",
      "groupNumber": 1,
      "name": "King Hospital",
      "__v": 0
    },
    {
      "users": [
        null,
        "5d5eafb3cc3b85c1c5cc81dc7"
      ],
      "managers": [
```

Download

## 4 React

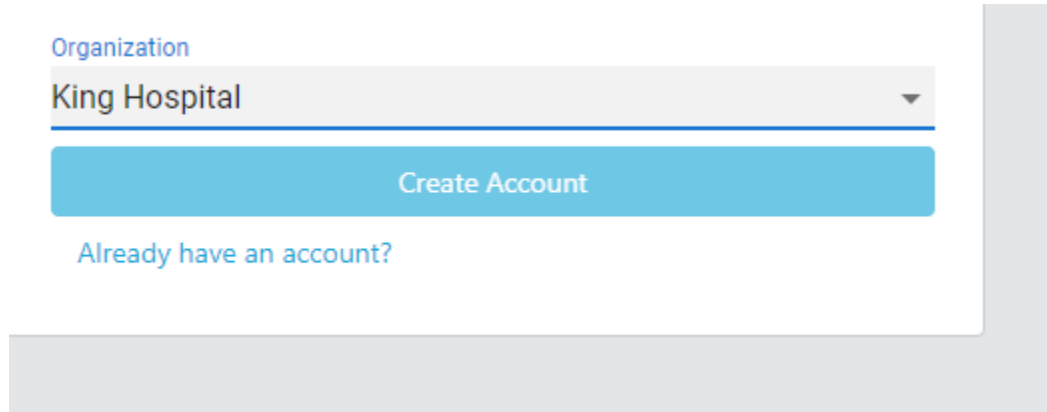
We use AXIOS to consume our GET Api to get all organizations data in the back of the front-end.

```
export async function getAllOrganizations(groupNumber) {  
  let urlStr = config.server + '/api/v2/organizations';  
  try {  
    const result = await axios.get(urlStr);  
    return result.data.organizations;  
  } catch (e) {  
    throw e;  
  }  
}
```

Then transfer them to the state of React component.

```
initialGroups = () => {  
  getAllGroups().then((dbGroups) => {  
    if (dbGroups[0]) {  
      getAllOrganizations().then(dbOrganizations => {  
        this.setState( state: {  
          dbGroups,  
          selectedGroupNumber: dbGroups[0].groupNumber,  
          dbOrganizations,  
          selectedOrganizations: dbOrganizations,  
          selectedOrganization: dbOrganizations[0].name  
        })  
      })  
    }  
  })  
}
```

In the end, the browser Displays the organization data to the end-user by using the select menu in a textfield(Material-UI component).



The screenshot shows a web form with a light gray background. At the top, there is a label "Organization" in blue. Below it is a select menu with a light gray background and a blue border. The selected option is "King Hospital". Below the select menu is a blue button with the text "Create Account". Below the button is a link "Already have an account?" in blue.

```
<TextField
  select
  InputLabelProps={{shrink: true}}
  label="Organization"
  value={this.state.selectedOrganization}
  onChange={this.handleChange('organization')}
  margin="normal"
  fullWidth
>
  {this.state.selectedOrganizations.map(option => (
    <MenuItem key={option._id} value={option.name}>
      {option.name}
    </MenuItem>
  ))}
</TextField>
```