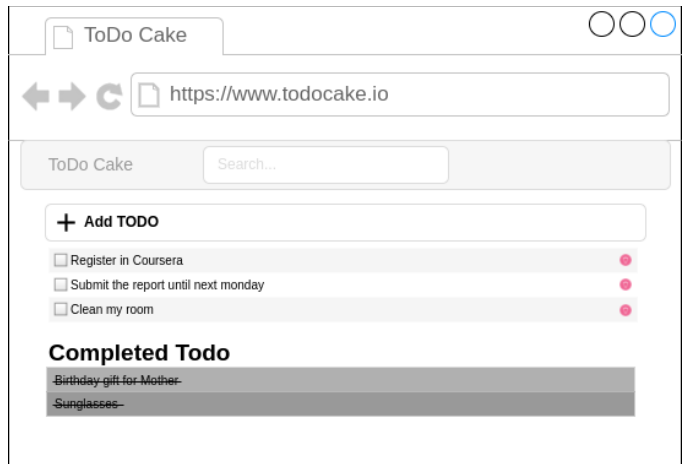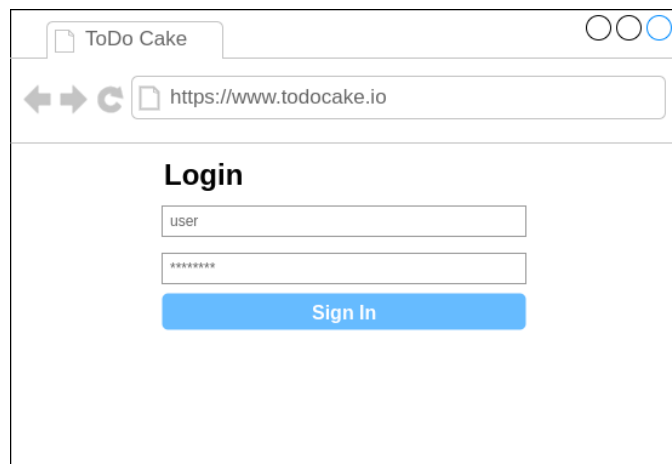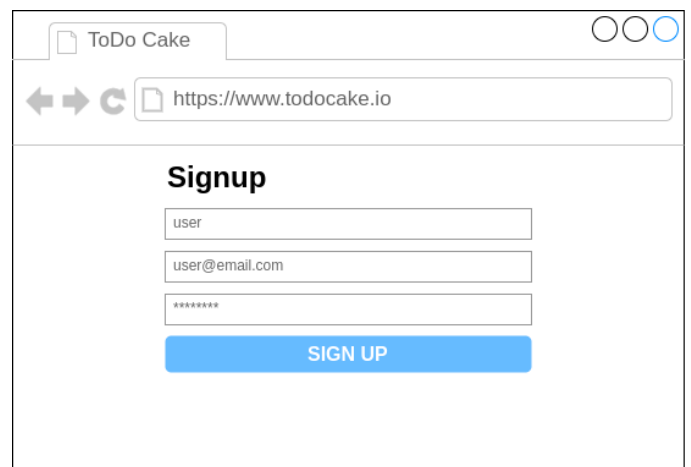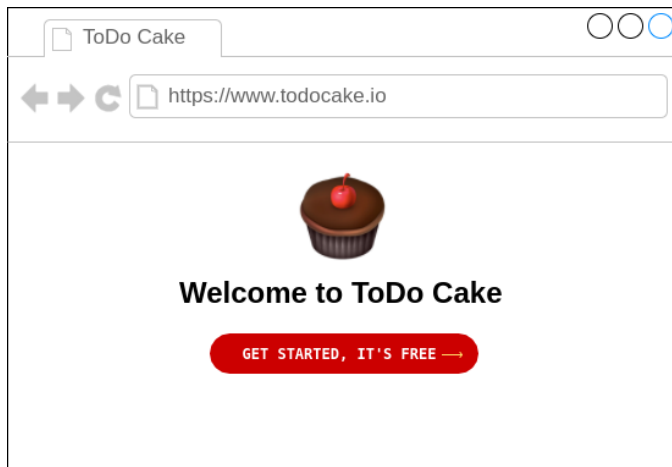# Application Design and Software Structure Report

(ToDo Cake) *To Do app*

1. Introduction

App that allows users to sign up for an account and store To Do items to their account, mark to do items as complete and show or hide completed items.
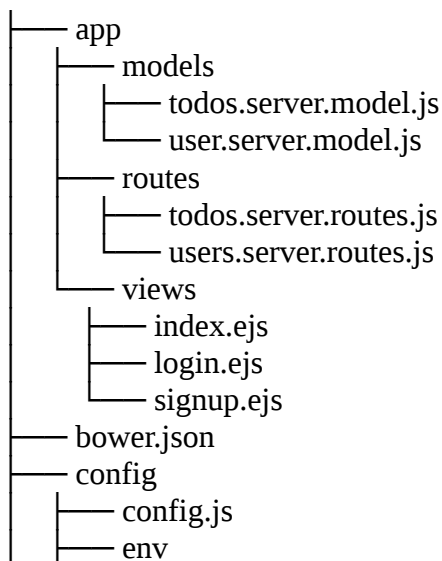
2. Design and Implementation

2.1 The REST API Specification

Method	End Point	Explanation
POST	/signup	Create new user
GET	/login	If user is not logged in then show Login
GET	/users/<userId>	Get user by userid
GET	/api/todos	If user is logged in then pull todos from Mongo
GET	/api/todos/<todoId>	Get Todo by todoById
PUT	/api/todos/<todoId>	Update Todo fetched by todoById
DELETE	/api/todos/<todoId>	Delete Todo fetched by todoById
POST	/api/todos	Create a new todo item

2.2 Front-end Architecture Design

Package Structure

```
├── app
│   ├── models
│   │   ├── todos.server.model.js
│   │   └── user.server.model.js
│   ├── routes
│   │   ├── todos.server.routes.js
│   │   └── users.server.routes.js
│   └── views
│       ├── index.ejs
│       ├── login.ejs
│       └── signup.ejs
├── bower.json
├── config
│   ├── config.js
│   ├── env
```

```
│           ├── development.js
│           ├── production.js
│       └── passport.js
├── gulpfile.js
├── manifest.yml
├── npm-debug.log
├── package.json
├── Procfile
├── public
│       ├── application.js
│       ├── main-module
│       │       ├── main.client.module.js
│       │       └── views
│       │           └── main.client.view.html
│       ├── resources
│       │       ├── css
│       │       │       ├── styles.css
│       │       │       └── styles.scss
│       │       └── img
│       │               ├── background.jpg
│       │               ├── todo-100.png
│       │               ├── todolist-header.jpg
│       │               └── todo.png
│       ├── todos-module
│       │       ├── controllers
│       │       │       └── todos.client.controller.js
│       │       ├── todos.client.module.js
│       │       └── views
│       │               ├── edit-todo.client.view.html
│       │               ├── list-todos.client.view.html
│       │               └── view-todo.client.view.html
│       └── users-module
│           └── users.client.module.js
└── server.js
```

2.3 Database Schemas, Design and Structure

- **Model**

Very simple data model

Based on two entities: user and todos

A user have 1 todo list

A todo has some items:

      ○ An unique ID

      ○ A creation date

The data is stored in a MongoJS database.

- **MongoDB Schema**

```
// Set up User Schema
// =============================
var UserSchema = new Schema({
 name: String,
 email: {
  type: String,
  index: true
 },
 username: {
  type: String,
  trim: true,
  unique: true
 },
 password: String,
 provider: String,
 providerId: String,
 providerData: {}
});


// Set up Todo Schema
// =============================
var TodoSchema = new Schema({
 created: {
  type: Date,
  default: Date.now
 },
 title: {
  type: String,
  default: '',
  trim: true,
  required: "Please enter a To Do item"

 },
```

```
comment: {
  type: String,
  default: '',
  trim: true
},
creator: {
  type: Schema.ObjectId,
  ref: 'User'
},
completed: {
  type: Boolean,
  default: false
}
});
```

2.4 Communication
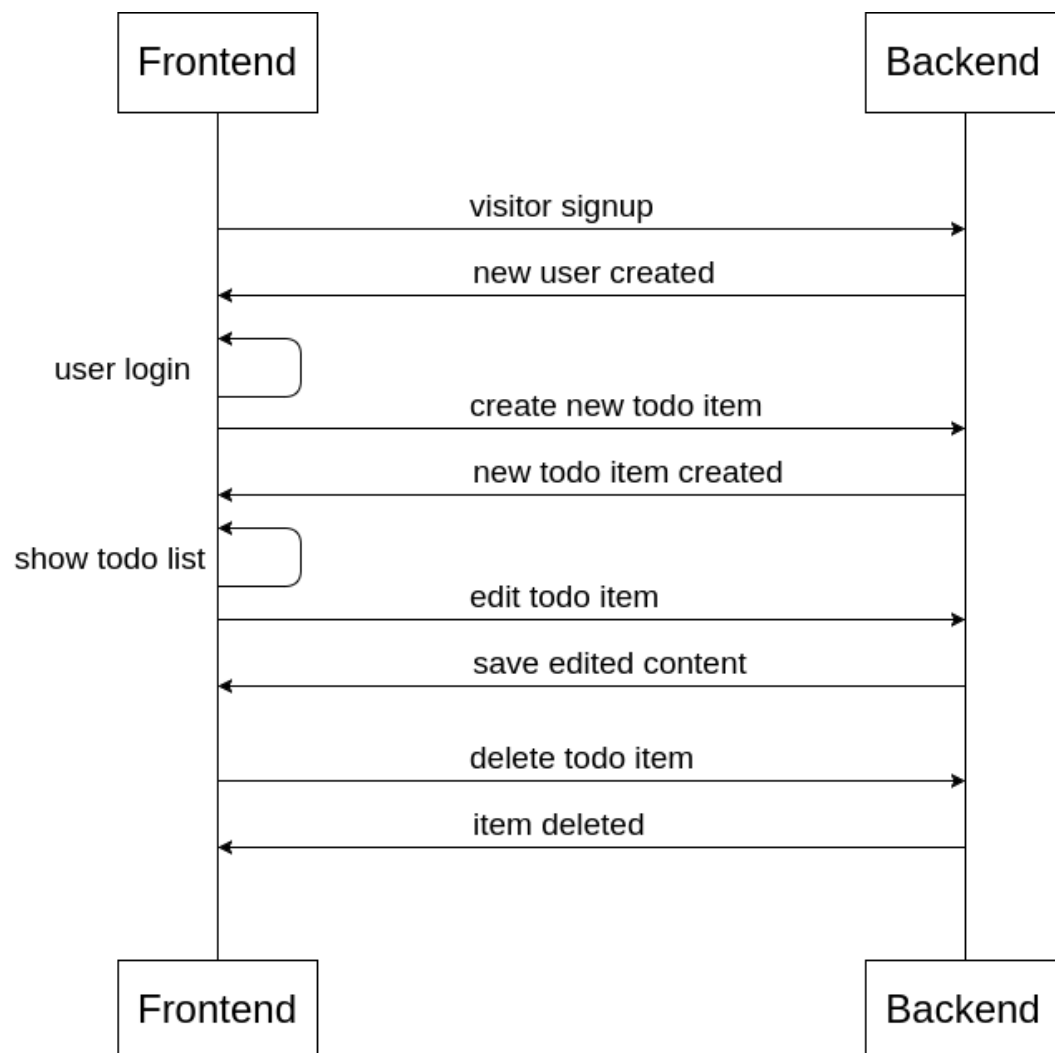
3. Conclusions

- **Users can create register and login**
- **Users can write to do items**
- **Every user can edit, delete or mark items as complete item**
- **To-Do items can be searched and ordered**

4. Reference

Tool used to draw the UI mock-ups:

- https://www.draw.io

Tool used to draw Package Structure

- http://www.computerhope.com/unix/tree.htm