JS Training Assignment

The task is split into three part, generic steps/requirements in each part is mentioned below.

Part 1: Setting up the Client-Side

1. Draw the dependency diagram (paper/ ms paint) required for the application.
2. Setup an Angular JS application with proper folder structure.
3. Create Services for the Data Models required.
4. Build Controllers and bind UI with Scope.
5. Integrate Services & Controllers

Important Points

- Setup Routes & Templates (using $routeProvider)
- Use Provider/Factory for Services
- Integrate YUIDoc & include inline comments where necessary.
- Bootstrap UI can be added as a dependency for easy interface designing.
- Using Bower for dependency is mandatory also maintain the dependencies list in bower.json

Part 1: Setting up the Server-Side

1. Setup a SailsJS (Express, KOA can also be used) Application
2. Generate Models & Controllers for all the data models required.
3. Design each Model with their attributes and their constraints, including Associations between models.
4. Install MongoDB on your system
5. Install MonoDB Adapter for SailsJS and add your local MongoDB Installation as a connection.
6. Setup your REST APIs, and test them via Post Man (Chrome App)

Important Points

- All dependencies need to be injected via NPM and package.json has to be updated with your application dependencies.
- Read about using associations on Sails JS Docs/API
- Installing MongoDB is a DIY. (Read online tutorials)

- Create custom actions/controllers and add their routes via config/routes.js (Sails). You can also override the blueprint actions.
- YUIDoc is necessary for the non-blueprint actions & data models.

Part 3: Integrating Server-Client

1. Build REST Services in your Angular JS application using $resource.
2. Add your REST API URL as a constant in the application.
3. Integrate your Model Services with REST Services, implementing the callback or promises pattern.
4. Your application should now save data in MongoDB via the rest api.

Important Points

- You can integrate additional endpoints for your custom actions using $resource.
- Implementing Promises for the REST calls using $q is a bonus but it is encouraged.