Testing

General part

- Describe testing in general and the different kind of testing used in software development.
- Explain test principles like mocks, stubs, suites, etc.
- Describe the difference between *test first* and *test last* and their pros and cons.

Practical part

Testing is an important part of software development. The backend of web applications need testing in order to ensure that, for instance, the Restful APIs is working as expected.

1) In a simple web application, we need the backend to provide a CRUD Restful API that can handle user login information. In the file testing1.zip you find a simple Restful web service API written in node.js/express.js.
   Write tests with Mocha/Should that will test this API. Organize you test such that each HTTP verb is tested in its own suite or file.

2) To simplify the use of the REST API from other servers (Server-B on the figure), you must write a data layer that should use (call) the REST-API given in testing1.zip with the following interface:
   Do this in the same project, but as a reusable control, since this is meant to be used by other servers that want's to access the REST-API.

   - `getUsers(callback)`
   - `getUser(email, callback)`
   - `addUser(user, callback)`
   - `updateUser(user, callback)`
   - `deleteUser(user, callback)`

   The callbacks will take the usual two parameters **err** and **data**.

   Write tests in Mocha/Should + a sufficient HTTP mocking library (**nock**[1] will do the job) that should verify that the data layer call the correct URLs in the Rest API. The tests should not invoke the underlying Rest API.

3) Write tests that verify that the callback in your data layer gets the right parameters, i.e. that **err** is defined and hold the error message when an error occur and the **err** is undefined and **data** hold the data when no error occur. The tests should not invoke the underlying Rest API.

[1] https://www.npmjs.com/package/nock

Note: this part is NOT a part of the exercise, it's meant as FYI.

Test:

You have been introduced to tests several times during your three semesters with us.

As pre-preparation for test questions, you could find some of your old test examples. You will not have time to do this during your 80 minutes preparation, so do this during your general preparation before the exam.

You should have examples from second semester, including your second semester project, but also first semester introduced unit testing.

This semester, we have worked a lot with testing so do not forget to revisit these examples too.

General advices:

Spend you 80 minutes well.

In this question you get some code to start with. Get the code up and running. You don't need to run `npm install` in this example so it ready to roll.

Read the question carefully identify the parts you know how to get started with and begin there. If you get stock continue to the next part. You can always return later if you have the time.

Links and references:

- http://en.wikipedia.org/wiki/Software_testing
- http://en.wikipedia.org/wiki/Unit_testing
- https://docs.angularjs.org/guide/unit-testing
- http://mochajs.org/
- http://en.wikipedia.org/wiki/JUnit
- http://en.wikipedia.org/wiki/Immediately-invoked_function_expression