ASE 285 Homework 6 (Individual Project)

NKU ASE

- 100 points in total
- Make sure you understand the HW course rules.
- Make sure you use GitHub for uploading all the project related artifacts and Canvas Individual Project page for accessing the information.

The Goal of HW6

In this assignment, students are required to design and implement applications using Node.js, WebStorm IDE, Mongoose, and Git/GitHub. They define requirements for the application; they specify modules with input, output, and interfaces; they use WebStorm IDE to write, test, and debug their design; Finally, they use Git/GitHub to upload all the artifacts, schedule, and documents.

This assignment is to assess each student's capability to use software engineering rules and tools to build an application that meets users' requirements. So, students must work alone from requirements to testing/debugging, and any cooperation among students is not allowed and is regarded as plagiarism.

Tools

- Use the SE tools as planned in HW4.
- Use GitHub for storing all the information and use Canvas for accessing GitHub information.
  - Requirements
  - Schedule and its progress
  - Code and tests
  - Design document and manual

Problem Definition (Problem Domain)

1. We are given a file password.txt with the users' email addresses and passwords. We need to make an encrypted file, password.enc.txt, and upload the information into the MongoDB database using Mongoose.
2. We need to return true or false when users give email addresses and passwords.

Example

This is an example of the password.txt.

sm.cho@hello.com:123456
john.deacon@good.com:bestpassword
alan.may@best.com:mypassword
henry.taylor@edu.com:educatorbest

Students' code should generate password.enc.txt with an email:hash format. As an example, this is the output from the password.txt.

sm.cho@hello.com:8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
john.deacon@good.com:c495634064a4baa0c6f7a5aed1f9f47488b421a4eca666a0b112baa720cee7f5
alan.may@best.com:89e01536ac207279409d4de1e5253e01f4a1769e696db0d6062ca9b8f56767c8
henry.taylor@edu.com:14f4cbccaee1fa7fe31820e2d57f1389823350a6fe23054b2a3d7dde4fa8531b

The information is stored in the MongoDB database using Mongoose.

When a user gives henry.talyor@edu.com with the password educatorbest, your software should return true. When a user gives sm.cho@hello.com with password 123, your software should return false. When a user gives noname@hello.come (not in the password file) with password 1234, your software should return false. When a user gives alan.may@best.com without a password (or empty password string), your software should return false.

Requirements

- Students translate the problem domain into requirements.

Software Architecture/Design and Diagrams

- Software design documents should include the following:
  - Software architecture diagram that shows (a) Form, (b) Modules, and (c) Rationale.
    - The input, output, and information flow should be explained clearly.
  - A module design with (a) its responsibility and (b) interfaces.

Code

Students should make Node.js (JavaScript) applications.

passwordjs template

- Use the passwordjs template.zip as a hint for the application.
  - The template is a bare skeleton without the database implementation.
- Students can ignore the template and come up with their own.

Tests

- Use the passwordjs template.zip as a hint for the tests.

Unit Tests

Students should make unit tests for their modules. 1. They should test whether their file reading and writing are working. 2. They should test whether their conversion algorithm is working. 3. They should test whether their Mongoose database is working.

Acceptance Tests

Students should make sure their program passes the acceptance test. This is an example of Mac/Unix. For Windows, run just acceptance.

```
passwordjs> sh acceptance.bat
true
true
true
true
false
false
false
false
```

Documents

- Make a user manual so your clients can use the application.

Schedule

- Use GitHub to track your plan and progress.
  - Always keep track of LoC.
  - Always keep track of the burndown rate.
  - Always keep track of test coverage.
- Use the Canvas Individual Project Page to link the corresponding information.
  - Always keep your Canvas page updated with the latest information.

Recommendations

- Be creative, you can come up with your own format to deliver the product you promised.
  - You don't have to use the template for this assignment as long as you deliver what the client requests.

- Start early to finish early.

- Don’t start coding until you finish the requirements and design.
- Be sure to make unit tests whenever you make a unit.
- Keep making notes about your ideas, implementations, and such; these will be a big part of your design documents and manual.