Individual study work submission

This is a description of the simple individual study work that has to be done by every student.

Manually program a simple CRUD REST based API. Frameworks like Express for Node.js or Sinatra for Ruby are recommended.

You can create a simple API for example Flight booking, Food order, ...

Requirements:

- Use GIT and submit your code the SAD classroom on GitHub. Commit often and regularly. Just committing all in one is not permitted. Don't commit anonymously and use commit messages.
- Use routing, separate controllers and models in your code.
- Separate code in individual files.
- Minimum create all CRUD operations.
- Data must be stored in a database using a pattern like Active Record or a Data Mapper.
- Implement at least an API that implements Richardson's Maturity Model at level 2 (means HATEOAS is optional).

Final project submission (SAD)

This document should help you to structure your final project submission for the SAD course. All of the following elements have to be included in your final project documentation.

In SAD you will have to submit a documentation file and a link to your GIT repository.

Requirements for your GIT repository:

- ***No anonymous commits.***
- ***User commit messages***
- ***Commit often and regularly***
- ***Every group member should contribute code***
- ***Optional: You can include your project documentation in your repository***

After the 5 week SAD course your GIT repository should contain at least minimum prototype of your application.

Your documentation contains must contain the following items. In general: Please submit all your iterations. Describe how you came to the decision to change your previous iteration. Your final documentation should document the development of your thinking!

Formal aspects

- Group member names and matriculation numbers and GIT user names
- Source of your GIT repository

Vision/mission statement

- A short and simple to understand statement of your vision and objectives.

Personas (or equivalent method)

- 3 full persona description
- Make notes when/why you had to refine your personas
- Alternatives to personas are possible, if you decide that an other method will better suit for user analysis in your project.

→ Evaluation: Completeness and quality of task

Requirement analysis: E.g. Interviews (or equivalent method)

- Your documentation describes all iterations of interviews you do in the project. (Iterations means re-check of assumptions, re-check of refines)
- Detailed interview documentation with justification of your selected methods. E.g. justification of a standardized questionnaire construction.
- Audio/video and text transcription of the recorded audio.
- Minimum 3-5 interviews per persona or user role.
- Include name and contact details of your interview partners
- Try to interview real users or experts.

→ Evaluation: Completeness and quality of task

UX User Stories

- Structured collection of epics and all user stories collected.
- If - for any reason - there are “internal user stories”, means assumptions by your team that are not evaluated you have to justify the reason you keep them in your user story list.

→ Evaluation: Quality of task; Use of testing methods, documentation and justification of your iterations.

Experience map (or equivalent method)

- Create an “experience map” to show the entire customer experience and the value of your product idea.
- Chart the course - a customer's journey through your product (use a persona which takes this journey), tell a story with beginning, middle and end

→ Evaluation: Completeness and quality of task

Low-fi and High-fi prototypes

- Create low-fi (paper prototype) and high-fi prototypes.
- Your prototypes describe the layout of each screen and the possible interaction between the screens.
- The high-fi prototype will look quite exactly like the final product. E.g. for HTML based applications you could already write HTML code that can later be easily reused.
- Do user tests with the methods as discussed: Of course you cannot test “everything” with the high-fidelity prototype as there is no real functionality. But test, what is testable.
- Document your iterations. Why did you change things? What did you change
- The High-fi prototype can be created using your final technology if you like.

→ Evaluation: Quality of task; Use of testing methods, documentation and justification of your iterations.

Technology/Framework decisions and Software Architecture

- Which Technology/Framework did you choose for your project and why?
- Please also document your general architectural software design. This can be quite abstract at this moment (details needed in ACS course later).
    - Layer architecture
    - Front-/Backend architecture
    - UML to describe roles, workflows etc.

→ Evaluation: Quality of task; Use of testing methods, documentation and justification of your iterations.

User-Story to task mapping (for prototype only)

- Map your story to tasks. Note priority and implementation date.
- Plan which features will be implemented to your proof of concept/first prototype after 5 weeks.

Document workload

- Document the individual tasks and workload (hours per task) of every group member.