Project Sprints

September 29, 2022

Byung-Gon Chun

Project Progress

- Project team formation
- Project proposal
- Project sprints
  - Sprint 1 (Features + Setup) – requirements and specification
  - Sprint 2 - design and planning document, starting development
  - Sprint 3
  - Sprint 4
  - Sprint 5
- Final demo poster session – 6pm-10pm, Dec. 14, 2022 (Wed)
- Final report – 6pm, Dec. 19, 2022 (Mon)

You have to update your project requirements and specification document and your design and planning document every sprint.

Detailed info on the course github

- **Sprint 1**
  - **Requirements and Specification**
- **Sprint 2**
  - Design and Plan, starting from Sprint 2
    - We expect you to first update the Requirements and the Design documents to reflect your current accomplishments and your plans for Sprint 1.
    - We will ask that you submit updated copies of these documents with the code and tests for Sprint 2.
    - You will have to submit a short progress report.
    - **Start development**
- **Sprint 3**
  - Repeat the same procedure as for Sprint 2.
- **Sprint 4**
  - Repeat the same procedure as for Sprint 3.
- **Sprint 5**
  - Repeat the same procedure as for Sprint 4.
  - Your service should be up and available at the end of Sprint 5.
  - Potential clients will use your services, grade them, and give you feedback.
- **Demo Poster & Final Report**

Project Timeline

	Start	End	TA meeting
Sprint 1	Oc1. 3 (Mon)	Oct. 15 (Sat), 6pm(report due)	TH/F
Sprint 2	Oct. 17 (Mon)	Oct. 29 (Sat), 6pm(report due)	TH/F
Sprint 3	Oct. 31 (Mon)	Nov. 12 (Sat), 6pm(report due)	TH/F
Project progress presentation	Nov. 9		
Sprint 4	Nov. 14 (Mon)	Nov. 26 (Sat), 6pm(report due)	TH/F
Sprint 5	Nov. 28 (Mon)	Dec. 10 (Sat), 6pm(report due)	TH/F
Final poster	Dec. 14 (Wed)		
Final report		Dec. 19 (Mon) 6pm	

Project Requirements and Specification (From Sprint 1)

Project Requirements and Specification

- Inevitably in preparing this document you will discuss design and planning, but limit this document to requirements, a description of how the system should interact with the outside world.
- This will be **a living document**. In subsequent iterations you will expand the applicable sections.
- You should use Github wiki, which has its own versioning system.
- Be concise!

What to Include: Project Abstract, Customer, Competitive Landscape

- Project Abstract
  - A one paragraph summary (~200 words) of what the software will do.
- Customer
  - A brief description of the customer for this software, both in general (the population who might eventually use such a system) and specifically for this document (the customer(s) who informed this document).
- Competitive Landscape
  - Briefly identify the competitors in this market, and list the main ways in which your project is going to be different.
- **User stories**
- **User interface requirements**

What to Include: User Stories

- Start with a **short description of the actors** involved (e.g., regular user, administrator, ...) and then follow with a list of the **user stories**.
- Each user story should also have a field called ”Sprint" where you specify in which sprint you implemented or plan to implement this feature.
- At the end, you should maintain a bullet list of user stories that you plan to get to in future iterations, with only minimal detail for each such story. We expect that in future iterations some of these items will be promoted to full fledged user stories.

(Must be expanded for future iterations)

User Story Example (Connextra Format) (Style #3)

Feature: [name]

As a [kind of actor (who)]

I want to [do some task (what)]

So that [I can achieve some goal (why)]

Feature: Add a movie to MovieRank

As a movie fan

I want to add a movie to MovieRank service

So that I can share a movie with other movie fans

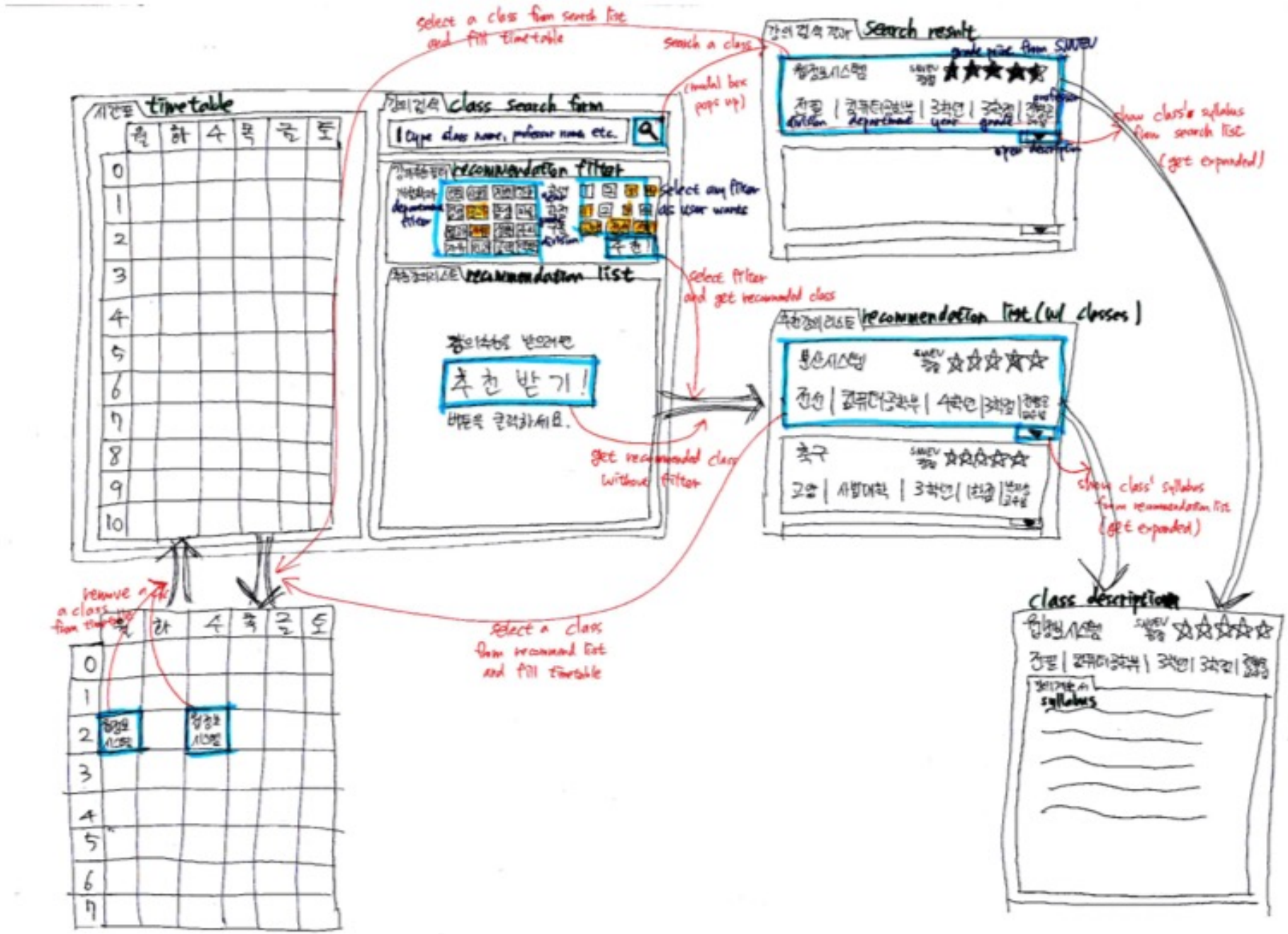
What to Include: User Interface Requirements

- Describes any customer user interface requirements including graphical user interface requirements. Here you should have **sketches or mockups** for the main parts of the interface. To save time you may want to use scanned drawings of interface mockups here, instead of Photoshop drawings.

- Just like for the User Stories section, you need to list here only the parts of the user interface that are applicable to the previous iterations and the current one.

  (Must be expanded for future iterations)

User Interface Sketch Example



Requirements and Specification Evaluation Criteria

Max Points	Content
5	Do the requirements state the customers needs?
5	Competitive analysis
5	Do the requirements avoid specifying a design (customer-specified design elements are allowed) ?
5	Do you believe all parts are possible to implement?
5	Is the project scope big enough?
	Completeness
20	Are the user stories written in sufficient detail to allow for design and planning?
5	Do the user stories have acceptance tests ?
5	Do the user stories mention error conditions and required behavior ?
5	Are there sufficient user stories for the first iteration?
5	Is there a discussion of the stories for future iterations ?
20	Are the User Interface Requirements given with some detail? Are there some sketches, mockups?
	Clarity
5	Is the document carefully written, without typos and grammatical errors?
5	Is each part of the document in agreement with all other parts?
5	Are all items clear and not ambiguous? (Minor document readability issues should be handled off-line, not in the review, e.g. spelling, grammar, and organization).

Design and Planning Document
(From Sprint 2)

Design and Planning Document

Design and Planning Document

Project Name
Design and Planning Document
??/??/??,
Version major.minor

Instructions

This is a design and planning document template for SWPP. Please fill out this template carefully.

This will be a living document. For the first iteration you will fill in the document with the design details as you can see them before the first iteration. In subsequent iterations you will expand this document.

You have to use Github wiki for this document.

Design and Planning Document: System architecture

- The **high-level architecture** of your system: the major pieces and how they fit together.
- Use graphical notations as much as possible
- Try to be concise!
- Try to use standard architectural elements (e.g., pipe-and-filter, client-server, event-based, model-view-controller)
- Describe the **major interfaces between components**, which you will describe in more detail in the "Design details"

Design Details

- Important facets that are not at the level of “architecture,” such as descriptions of critical algorithms, protocols, and key invariants
- Wherever possible items should be linked back to your specification
- Specify at least the **API** among the major components
  - If there are messages sent between clients and servers, you should identify **what messages and what data they contain, and in what format, and in what order they should be sent**.
- We expect to see a more refined design for the features to be included in the current sprint, and perhaps a more rough design for the features to be implemented in future sprints.
- If you have considered alternative designs, please describe briefly your reasons for choosing the final design.

Implementation Plan

- Break down **each user story** described in your requirements document into **programming tasks**.
- Try to estimate the number of developer-days that the tasks should take.
- Try to also determine **dependencies** among tasks.
- You should list all of the tasks to be done in the current sprint, a preliminary assignment of tasks to people in the group, estimates of the time for each task, dependencies between tasks, and a preliminary division into sprints
- **The plan should be designed to get some prototype system running as quickly as possible and then growing towards to the full project over a sequence of sprints.**
- Try to identify the major risks for the project in general and the plan in particular. Explain what the risks are and discuss how you plan to mitigate them.

Testing Plan

- Describe how you plan to test the system.
- Thought should be given to how mostly automatic testing can be carried out
  - **Unit testing**: explain briefly for what modules you plan to write unit tests, and what framework you plan to use
  - **Functional testing**: What APIs you plan to test? How will you test them? What tools you will use? Will you write mocks?
  - **Acceptance & integration testing**: how do you plan to test the user interface and scenarios?

Registering Github Issues

- You have to register Github issues regarding tasks for design, implementation, and testing and mark them with sprints (for example use milestones)

- Close issue(s) when a pull request addresses them. Usually one pull request addresses one issue.

Design Document and Grading Guidelines

Max	**Design**
8	Are all parts of the document in agreement with the product requirements?
10	Is the architecture of the system described, with the major components and their interfaces?
10	Are all the external interfaces to the system specified in detail?
10	Are the major internal interfaces (e.g., client-server) specified in detail?
8	Is there sufficient detail in the design to start Iteration 1?
4	Is there reasonable detail in the design for features in future iterations?
	Planning
8	Is the plan for Iteration 1 sufficiently complete to start the implementation ?
4	Are the subteams identified and has enough thought been given to parallelization of effort and team dependencies?
4	Is there a discussion of the risks associated with this plan?
4	Are the testing activities scheduled at the appropriate times?
	Testing
5	Does the design take into account testability of the various units, components, and subsystems ?
4	Is there a discussion of how unit testing will be done?
6	Is there a discussion of how functional (API) testing will be done automatically?
4	Is there a discussion of how acceptance/integration testing will be done?
	Clarity
4	Is the solution at a fairly consistent and appropriate level of detail?
4	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
5	Is the document making good use of semi-formal notation (UML, diagrams, etc)
4	Is the document identifying common architectural or design patterns, where appropriate?
4	Is the document carefully written, without typos and grammatical errors?

From Sprint 2, Revised requirements/spec and design documents

- Update the implementation and testing plans (in the design document) to reflect what you have already accomplished.
- Identify the features you are going to implement in the current sprint. Add user stories (to the requirements document), task breakdowns (design doc), and assignments (design doc). You may need to update tasks that weren't completed in the last sprint. Reassess the task difficulties if necessary.
- Flesh out your list of possible features for future sprints (requirements document). Ideally, you should have a rough outline in place for all sprints. You will change this as we do each sprint, but having a general plan in place will be helpful to everyone involved.

From Sprint 2, Revised requirements/spec and design documents

- Identify any changes in the requirements, system architecture and design details.
- Ensure the design document is current for the set of features you've implemented to date and will implement in the next sprint.
- Discuss design decisions that affect testing and describe any test interfaces built into the system (in the testing plan section).

You **must mark clearly the parts of the document that were changed** (use the Changes section at the start of the document).

Project Sprint Deliverable

- Sprint report (1-2 pages)
  - Including the contribution of each team member
  - Including your code coverage **(the coverage must be over 80%. If not, we will deduct 20% of the overall score of that sprint)**
- Updated project requirements and specification document

  - The first version is part of Sprint 1
- Updated design and planning document
  - The first version is part of Sprint 2

Sprint Instructions

- A (short) progress report
  - Contribution of team members
- An update of your specification and design documents
- Code for features and tests (tag)

Sprint Progress Report

Write a short (1~2 pages) summary for sprint and submit it along with the revised documents.

- What were the main difficulties so far? You should consider both technical and organization issues.
- Were there any features you did not implement as planned, and why? Are you pushing some features to later iterations, and if so, why?
- What tests did you prepare for this sprint, and what are they covering? Did the tests you wrote deviate from your plan? What features are you not testing yet?

Sprint Progress Report

Write a short (1~2 pages) summary for sprint and submit it along with the revised documents.

- You must include in your progress report a test coverage report. You must briefly specify what tool you use, and you must include one or more screenshots showing:
  - The overall coverage metric
  - The list of classes with lowest coverage. Explain why the coverage is low, and what (if anything) you plan to do about it
- Tag a master branch in your repository to identify the code you want to submit for the current sprint. The date and time of this revision should be before the deadline. The branch should include a README file with instructions on how to run the application and tests.

Submission

- Each team must send an email to **swpp.22.staff@spl.snu.ac.kr** by the deadline. The subject line of the email message must start with the word "[SWPP22] teamX sprintN" (replace X with your team number and replace N with 1, 2, ...) followed by your team number and sprint number. The email should contain:
  - Pointers to the (previously submitted) requirements document and design document.
  - The sprint summary progress report, as described above.