Project Sprints

October 10, 2017

Byung-Gon Chun

Reminder: Cheating

- What is cheating?
  - Sharing code: by copying, retyping, looking at, or supplying a file
  - Coaching: helping your friend to write a programming assignment, line by line
  - Copying code from pervious course or from elsewhere in the Internet
  - Cheating attendance
- Penalty for cheating: your grade

Announcement

- HW2 due Oct. 11, 8:59PM
- HW3 out today – due Oct. 25 (Wed)
- Project 1 sprint **starting today**
- **In-class 1-hour mid-term exam** – Oct. 31
  - Write code on the **paper**
  - Open only **lecture/practice session notes**
- Project roadmap + guest lecture today

Homework 3 - Django

Due: 10/25 (Wed) 20:59 (This is a hard deadline)

In this assignment you will implement a backend service for the blog frontend that you have created in homework 2. This is an **individual** assignment.

Through this assignment, you are expected to:

- Build a RESTful API server with Django
- Understand how communication between the client and the server occurs
- Test your Django application

Features

As you have seen in homework 2, our blog has three models: *user*, *article*, and *comment*.

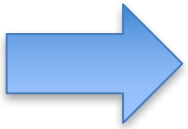
- Each user should be able to sign up, sign in and sign out.
- Only those users who are wigned in are allowed to read or write articles and comments.
- Users should be able to update or delete articles and comments only which they have created.

Upcoming Topics

- Requirements and Specification
- Software Development Process
- Wrapping Up Taming Complexity
- Defensive Programming
- Testing
- …

Project Progress

- Project team formation
- Project proposal
- Project sprints
  - Sprint 1 – requirements and specification (mandatory), the first design and planning document (optional)
  - Sprint 2 - the first design and planning document (mandatory)
  - Sprint 3
  - Sprint 4
  - Sprint 5
- Final demo poster session – Dec. 19, 2017



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

- **Sprint 1**
  - **Requirements and Specification (Required)**
  - **Design and Plan (Optional)**: Based on your specifications, you will have to start making a design and a plan for the second sprint, and to implement the features for this sprint.
- **Sprint 2**
  - Design and Plan (Required), starting from Sprint 2
    When you start 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.
- **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 Iteration 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**

Course Web Site Updated

Project

The project will follow this process:

- Project team formation
- Project proposal
- Project sprint 1
  - Requirements and specification (required)
  - Design and planning (optional)
- Project sprint 2
  - Design and planning (required)
- Project sprint 3
- Project sprint 4
- Project sprint 5
- Final demo poster session & final report

Note : Every document for each milestones must be written in **English**!

Sprint Instructions

Project Timeline

Project Timeline

	Start	End	TA meeting
Sprint 1	Oct. 10	Oct. 23, 6pm(report due) (Mon)	Oct. 20 (Fri)
Sprint 2	Oct. 24 (Tue)	Nov. 6, 6pm (report due)	Nov. 3 (Fri)
Project progress presentation (demo)	Nov. 4 practice session (graded by me)		
Sprint 3	Nov. 7 (Tue)	Nov. 20, 6pm (report due)	Nov. 17 (Fri)
Sprint 4	Nov. 21 (Tue)	Dec. 4, 6pm (report due)	Dec. 1 (Fri)
Project progress presentation (demo)	Nov. 22 practice session (graded by me)		
Sprint 5	Dec. 5 (Tue)	Dec. 18 (Mon) - no report	Dec. 15 (Fri)
Final demo poster	Dec. 19 (11am-1pm)		
Final report	Dec. 21 6pm		

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.
  - Send us your team number and Github repo name ASAP
- Be concise!

Project Requirements and Specification

- You **must send an email to swpp-staff@spl.snu.ac.kr with a PDF version of your document.**
- You must send the email by the deadline in the class calendar. This is a **HARD** deadline.

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

- This section will include the specification for your project in the form of user stories. The section should 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 "Iteration" where you specify in which iteration you implemented or plan to implement this feature.
- You should list only the user stories for the previous iterations and those for the current iteration.
- At the end of this section 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 Stories

- Name, Actors, Triggers (what initiates the story)
- Preconditions (in what system state is this user story applicable)
- Actions (what actions will the code take to implement the user story)
- Postconditions (what is the system state after the story is done)
- Acceptance tests (list **one or more acceptance tests with concrete values for the parameters, and concrete assertions that you will make to verify the postconditions**).

User Story Example

- Feature: Create Tour
- Actors: Traveler or Guide User
- Precondition: The traveler or the guide has to be a registered user and logged in
- Trigger: User clicks on the “Create Tour” button
- Scenario:
  1. The page displays a map with all the map markers that the route that the user has created
  2. On the left you can find buttons to access all the information of the tour such as: Name of the tour, begin and end times of the tour, description of the tour, description of the places of route, dates in which the tour is available, transportation means.
- Exceptions: The user does not fill out all fields to create the Tour
- Acceptance Test:

  Given that the user is registered and logged in

  And the user has fill out all the required fields to create the tour

  When the user clicks on the “Create Tour” Button

  Then the user should see “Your tour has been created”

User Story Example

- **Feature**: Search a course and add it to time table
- **Scenario**: Search a course
- **Acceptance Test:**

  Given I am on the snusugg homepage
  When I type 'Calculus' in search box
  And I click the search button
  Then I see a list of courses that have the name 'Calculus'

User Stories

Feature name

As a [kind of actor]

So that [I can achieve some goal]

I want to [do some task]

Feature: Add a movie to MovieRank

As a movie fan

So that I can share a movie with other movie fans

I want to add a movie to MovieRank service

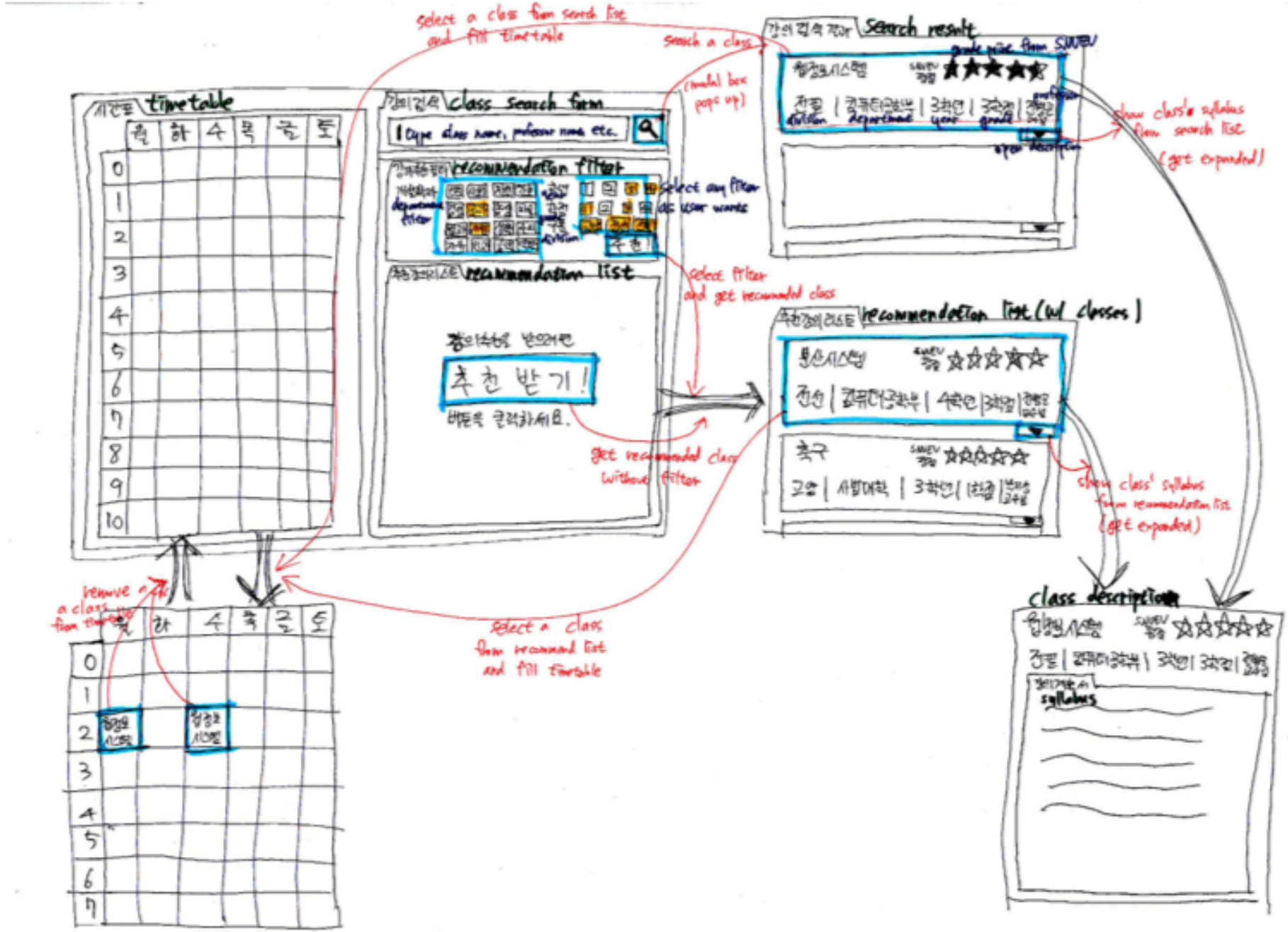
User Interface Requirements

- 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



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

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

- 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 must send an email to swpp-staff@spl.snu.ac.kr with a **PDF** version of your document by the deadline.

- You must send the email by the deadline in the class calendar. This is a **HARD** deadline.

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 iteration, and perhaps a more rough design for the features to be implemented in future iterations.
- 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 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 iterations
- 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 iteration. 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.
- 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 70%. If not, we will deduct 10% of the overall score of that sprint)
- Updated project requirements and specification document
- Updated design and planning document
  - The first version is part of Sprint 1 or Sprint 2
- Presentation: description and demo (if available)
- Use your notebook for presentation

Sprint Instructions

- An update of your specification and design documents (Requirements and Design)
- A completion of a running version of your code with a partial set of features
- Testing support and tests for the implemented features
- A (short) progress report
  - Contribution of team members

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? Did you use any test frameworks?
- You must include in your progress report a test coverage report. You must specify what tool did you use, and you must include one or more screenshots showing:
  - The overall coverage metric
  - The list of classes with lowest coverage. Explain why is the coverage low, and what (if anything) you plan to do about it

- Tag a branch or revision 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. This 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-staff@spl.snu.ac.kr by the deadline. The subject line of the email message must start with the word “sprintN” (replace N with 1, 2, ...) followed by your project's name. The email should contain:
  - Pointers to the (previously submitted) requirements document and design document.
  - The sprint summary progress report, as described above.
  - Instructions for your TA to be able to check out a tagged version of the code that constitutes your sprint release. You will figure out how to add your TA to the list of people allowed to check out code. Do not leave this for the last minute.
  - A url and instructions for using the app in the current stage.