Topics

- Finish Logistics and Goals of 491
- Basic Ideas in Software Engineering
- Short quiz on Thurs (not Tues)

Homework

- HW1 (out)
Grades (also see course syllabi)

**CPSC 491 (Software Engineering)**

- 30% – Three Exams
- 10% – Weekly Quizzes
- 35% – Homework Assignments
- 15% – Working Prototype (and Write-Up)
- 10% – Classwork & Participation

**CPSC 491L (Senior Design I)**

- 50% – Technical Contributions (peer evals, code reviews, advisor)
- 15% – Teamwork Contributions (peer evals, advisor)
- 15% – Non-Technical Contributions (peer evals, advisor)
- 10% – Participation (peer evals, attendance at meetings)
- 10% – CEDE Deliverables (DAB & advisor assessment)
Attendance

- required ... quizzes, in class exercises, etc.
- over 4 absences may result in a V

Academic honesty

- looking at and sharing code, working together is critical!
- using other libraries, frameworks, code also encouraged (with proper credit)
- however, adding a team member’s name to an assignment/report means you certify the person contributed
- accurate/honest evaluation of each other
- do your own work when asked in assignments, etc.

Quizzes

- Most weeks (usually Tuesday’s) we’ll have a quiz on 491 material
- Quizzes always at beginning of class
- No makeup quizzes
- Content in class, readings, homeworks

Homework Assignments

- Most will be a mix of individual and group work
- Individual work focused on SE readings and reflection
- Team work related to project plan and applying SE approaches
Technical Writing

- 491 (and 491L) have significant writing components
- 491 has a “writing enhanced” designation (for the University Core)
- Communication through writing is important in software engineering
  - Project planning (requirements, risks, milestones, ...)
  - Software design (architecture, UI, ...)
  - Project management (tasks, schedules, testing, ...)
  - Day-to-day work (communicating status, issue tracking, ...)
- We’ll discuss various aspects of technical writing for SE & CS
- Portion of homework grade on writing (more later)

Course Topics

- Basic ideas in software engineering
- Software development life-cycles (SDLCs, also know as “processes”)
- Requirements engineering
- Software estimation
- Project planning
- Project management
- Software testing
- Software development best practices
Project Plan

Overview (see Template)

1. Project Overview (summary, objectives, deliverables, related, ...)
2. Requirements
3. Design Considerations (Initial UI design, architecture)
4. Project Risks
5. Initial Product Release Plan
6. Maintenance Considerations
7. Project Management Considerations
Major Features Checklist

Minimal Viable (Useful) Product (MVP)

• the minimal set of features that make the product viable

• for us, we (mostly) are interested in the minimal useful product

The Major Features Checklist

• list of the main areas of functionality, etc., needed for your

• MVP you should view these as a contract/commitment features must be
  – approved by your sponsor and advisor team evaluated on completion
  – and quality of the features features can change, but changes must be documented and approved

Peer Evals

• Every two weeks

• I will hand out in class

• Please fill these out in class

• I will distribute to your faculty advisor
Tasks

You will be doing something like 2-week “sprints” in senior design

- Basic idea: every 2 weeks something to show to sponsor and users
- Note: We’ll discuss the idea of sprints in more detail later

You will need to keep track and manage sprint tasks

- Basic idea: meet with advisor weekly on goals / tasks / progress
- Keep track of tasks as you go (e.g., see the weekly task sheet):
  - Initial goals and what you plan to finish by end of week
  - Initial tasks, initial estimate, actual time, and completion status
  - Summary of week with ideas for improvement (reflection)

Challenge: Creating Good Task Descriptions

- Should be clear what the task is
- Should be obvious when task is “done” (testable)
  - “Research node.js” is not a good task description
  - “Do tutorial foo on node.js” is better
  - “Build a simple webpage using node.js tutorial foo” is even better
- Describe what complete means for each task
- Estimate how long it will take in hours (... more later)
Basic Ideas in Software Engineering

Product vs Process vs Project

- A **product** is the software or other deliverables being built (artifacts)
- A **process** defines steps to follow to develop the product (aka **lifecycle**)
- A **project** follows a process to develop (and deliver) the product

The four “big” ideas in software engineering

<table>
<thead>
<tr>
<th>Why?</th>
<th>... problem to solve (e.g., <strong>business objectives</strong>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who?</td>
<td>... customers and users (<strong>stakeholders</strong>)</td>
</tr>
<tr>
<td>What?</td>
<td>... product to build (<strong>requirements</strong>)</td>
</tr>
<tr>
<td>How?</td>
<td>... the plan (<strong>process</strong> + <strong>schedule</strong> + <strong>management</strong>)</td>
</tr>
</tbody>
</table>

Business Objectives (the “Why”)

- desired **outcomes** of the **client/company** creating the product

Q: What are some reasons for creating software products?

- make or increase profits
- save time or money (for company)
- attract investors (e.g., early stage company)

Requirements (the “What”)

- what the product should do or how it should work
- **features/functions** and **characteristics** of desired product
Functions vs Features

• a “function” is something a user accomplishes with product
  – e.g., make a phone call

• a “feature” is how users accomplish functions with product
  – e.g., use a keypad to dial a phone number

Exercise 1: Come up with another example of a function and supporting features

Exercise 2: Identify a potential needed function for your project

Exercise 3: Identify a potential business objective (desired outcome) for your project