Software Engineering, Winter 2022

1 COURSE SCOPE

This course deals with how to design and develop a reasonably big software – big enough that it warrants multiple phases and multiple developers – such as for example a distributed system - including architecture, design, implementation, and testing. Good software development mandates that we do a set of tasks before committing to code. As part of this course you will learn about:

- Software development lifecycle.
- Spec the software requirements.
- Model the software using UML diagrams.
- Architecting big software. This involves analyzing the following:
    - The various components that are required.
    - The dependencies between the components, and how they would communicate with each other.
    - Make well thought-out design choices (cost-benefit ratio, risks).
- Design and design patterns.
- Predict the performance.
- Implement.
- Validate.
- Track the progress of the project throughout the development cycle.

2 DESIRED LEARNING OUTCOMES

1. Software development lifecycle.
2. Software design patterns.
3. Writing design specs.
4. Coding standards and guidelines.
5. Source depot and version control systems.
6. Debugging.

7. Working in groups.
8. Working on shared code.
9. Code reviewing.
10. Integrating and testing your code.
11. Demonstrating your project to the customer (your class and your faculty in this case).
12. Plus familiarity with the specific project that you are assigned.

3 RECOMMENDED BOOKS AND RESOURCES

3.1 BOOKS:

- Design Patterns: Elements of Reusable Object-Oriented Software by Ralph Johnson · Erich Gamma · Richard Helm · John Vlissides
- Software Project Survival Guide by Steve McConnell
- The Mythical Man-Month by Fred Brooks
- Joel on Software by Joel Spolsky

3.2 WEB RESOURCES:

- UML: UML (embarcadero.com), UML (uml-diagrams.org)
- XML: https://www.w3schools.com/

4 COURSE STRUCTURE

In this course, you'll be engineering specs, design and code for a software component which will work together with other components that your friends work on, creating the overall software product. Your marks will largely be the result of your individual component although the small remaining fraction is decided on how your component integrates with the rest of the components, and how well you work as a team.

5 PREREQUISITES

- **Required**:

Students are expected to know basic programming – strings, arrays and pointer basics, dynamic memory allocation, data structures.

- **NOT required**:

Students are not required to know advanced topics like GUI programming, threading, IPC or networking. We'll learn some of those as part of this course.

6 REFERENCES AND CITATIONS

- You'll be working on a shared project, so collaboration is encouraged. But make sure that you do your part while helping each other out. Give credit and citations where due.
- The Internet is a great source of information. You are encouraged to use it as a resource. But please keep in mind that not everything that you see in there is authentic. You may see design and code that is seemingly correct but is actually flawed. So read and make sure you understand it before using it.
- If you are referring to code from the Internet (or anywhere else), give proper citations. Also keep in mind that code can have licenses. If you are planning to use any, make sure that the appropriate license permits it.

7 TIMELINE

Course will be completed in 13 weeks (~2 hours/week). In between we might have one additional week which we shall use for the quizzes, and for any ad-hoc topics/tasks.

8 PROJECT:

The course consists of a group project. The class is presented with a large distributed system to develop. Students are given roles: Project Manager, Technical Architect, Team Leader, and Team Member. There are five or six development teams. The class then manages the development of this system, using a standard development process model. The instructor acts as the customer, but does not participate in managing the project except, perhaps, for occasional advice. The project has a development process schedule, with formal reviews, and a final qualification test at the end of the semester.

There are two parts to the project. In the first part, you are required to choose a role for a module, and design and critically analyze it. Analysis deals with your module's specification, architecture, design, implementation,

test, and documentation. You are required to develop a Design Document, and Testing methodology. In the second part, you are required to implement your module and integrate it with the other relevant modules.

9 SCHEDULE

Lecture	Student work in Lab
Week 1	

• Introduction
  ○ management of large software systems (functional programming, OOD, COM, AOP)
  ○ what makes good software
• Project specifications
	Understanding project requirements

Week 2	
• OOD • UML • Writing design spec	**Group Project: Choose Teams**
Week 3	
• Software Development Life Cycle • Design Patterns	Group Project: Spec / prototyping
Week 4	
• UI programming	Spec / prototyping

• XAML, MVVM • Unit Testing the UX	
Week 5	
• Testing methodologies • Code coverage	Work on individual components/tests
Week 6	
• Processes, Threads • Synchronization	**Due: Spec for group project**
Week 7	
• Networking • Distributed systems • Peer-to-peer versus client/server	Work on individual components/tests
Week 8	
• XML, JSON, Schema • Databases	Work on individual components/tests Integration/Testing
Week 9	
• Performance, • Security, Threat modeling • Profiling	Integration/Testing

• Debugging	
Week 10	
• Unmanaged versus managed code • Interop	Integration/Testing
Week 11	
• Project readiness review	Integration/Testing Project should be ***almost*** complete by now
Week 12	
• Project demo	**Due: Demo for group project**
Week 13	
• Project evaluation	**Due: Final group project**

10 GROUP PROJECT: LAB SESSION MONITOR

10.1 GOAL

You shall design and develop a **Lab Session Monitor**: **An instructor and students can interact, collaborate and share contents (messages, files), share screen, and share a digital whiteboard across this application. There is a dashboard that will summarize the contents of the discussion and persist it, for later retrieval.**

Requirements are intentionally kept brief. You are expected to spec it in more detail, and cover it in more detail before you design, code and test the product.

10.2 TRACKING PROGRESS:

For every class session, we shall use the last 30 minutes for project tracking and open questions. Team leads and project leadership shall present the status.

10.3 PARTICIPANT TEAMS:

- Networking:
    - Serialization, Queueing (priority queueing), Sockets
    - Strength: 3
- Cloud:
    - Backend, REST APIs, Authentication, UX
    - Strength: 3
- Dashboard:
    - Session management, User Profile, Authentication, Persistence, Telemetry, Summary Logic, UX
    - Strength: 4
- Content:
    - File, Chat (emoticons, reply to messages, deleting messages, editing messages), UX
    - Strength: 3
- Screenshare:
    - Image capture, processing, filtering, management, UX
    - Strength: 6
- Whiteboard:
    - State management, objects, UX
    - Strength: 6
- UX:
    - Client/Server UX Layouts, Theme, Customization, Localization
    - Strength: 3
- Leadership:
    - Scheduling, checkpoints, communication, product quality, analytics, E2E automation

- Overall design, interfaces between various components, code quality, diagnostics
- Strength: 2

11 GRADING POLICY

TEAM MEMBER:

- Exams:
  - Quiz: 30 %
- Group project:
  - Your part of the spec: 15 %
  - Code:
    - Functionality: 20 %
    - Code cleanliness: 5 %
    - Unit Testing and code coverage: 15 %
  - Overall software product:
    - Integration with other components: 10 %
    - Feedback from leads and teammates: 5%

TEAM LEAD:

- Exams:
  - Quiz: 30 %
- Group project:
  - Spec: 15 %
  - Module functionality, overall process, Code reviewing and Depot management: 10 %
  - Code:
    - Functionality: 10 %
    - Code cleanliness: 5 %
    - Unit Testing and code coverage: 15 %
  - Overall software product:
    - Integration with other components: 10 %
    - Feedback from other leads and teammates: 5%

PROJECT MANAGER:

- Exams:
    - Quiz: 30 %
- Group project:
    - Project Management - Tracking, weekly updates, scrum: 15 %
    - Source depot - Setup, management, project wiki, process: 15 %
    - Code:
        - Functionality: 10 %
        - Code cleanliness: 5 %
        - Unit Testing: 5 %
    - Conflict resolution:
        - Bringing clarity and sorting communication issues: 5 %
    - Overall software product:
        - Feedback from leads and teams: 5%
    - Final demo:
        - On-time delivery: 5%
        - Downloadable product: 5 %

TECHNICAL ARCHITECT:

- Exams:
    - Quiz: 30 %
- Group project:
    - Spec: design and analysis: 15 %
    - Code reviewing and overall code quality: 10 %
    - Code:
        - Functionality: 10 %
        - Code cleanliness: 5 %
        - Unit Testing: 5 %
    - Conflict resolution:
        - Bringing clarity and addressing unexpected technical roadblocks: 5 %
    - Overall software product:

        - Meets requirements and works without crashing: 10 %
        - Feedback from leads and teams: 5%
    - Final demo:
        - Demo and demo video: 5 %

12 HELP

- Course instructor: Ram, chittur@iitpkd.ac.in, (+91)9496294392
- Teaching Assistant: Karri Nirnaeshitha, 112102002@smail.iitpkd.ac.in.
- Note:
    - “Help will always be given at Hogwarts to those who ask for it.”
        - The Harry Potter Series.
    - “If you've got a problem, If you're feelin' low
      Lookin' for some answers, Things you need to know,
      All you've got to do is ask, All you've got to do is ask.”
        - Frasier, Season 7 Episode 13.