Software Engineering Capstone Project - Phase 3
Implementation, Testing & Delivery

Due Date: Dec 7th

Overview

Welcome to the final and most exciting phase of your capstone project! In Phase 1, you defined your problem and gathered requirements. In Phase 2, you created a comprehensive design blueprint. Now, in Phase 3, it's time to bring your vision to life by building, testing, and delivering a **working software application**.

Remember: *Done is better than perfect.*

Learning Objectives

By completing Phase 3, you will:

- Transform design specifications into working code
- Apply software development best practices (version control, code documentation, testing)
- Debug and troubleshoot real implementation challenges
- Conduct software testing and quality assurance
- Create professional technical and user documentation
- Present and demonstrate your work effectively
- Reflect on the complete software development lifecycle experience

Deliverables

1. Source Code Repository

Requirements:

- Complete, well-organized source code
- Version control using Git or GitHub
- Repository must include:
    - **README.md** file with:
        - Project title and description
        - Features implemented
        - Technology stack used
        - Installation/setup instructions (step-by-step)
        - How to run the application
        - Screenshots or demo link
        - Known issues/limitations
        - Credits and acknowledgments
    - **.gitignore** file (exclude unnecessary files)
    - Meaningful commit history (minimum 15 commits showing development progress)
    - Clear folder/file structure

Code Quality Standards:

- **Comments:** Include comments for complex logic, not obvious statements
- **Naming conventions:** Use clear, descriptive variable/function names
- **Formatting:** Consistent indentation and style
- **Modularity:** Break code into logical functions/classes
- **Error handling:** Include basic error handling for user inputs and edge cases

Submission: Provide the repository link (ensure it's accessible - public repo or add instructor as collaborator)

2. Working Application

Your software must be **functional and demonstrable**. This means:

- Core features from your requirements are implemented and working
- The application runs without critical errors
- Users can interact with the main functionality
- It may have bugs or limitations, but it must demonstrate your concept

Acceptable Delivery Formats:

- **Option A:** Deployed application with live URL (web apps)
- **Option B:** Downloadable application with clear installation guide (desktop/mobile apps)
- **Option C:** Demonstration video (5-8 minutes) showing all features + runnable code
- **Option D:** Live demonstration during presentation + runnable code

3. Final Project Report
Submit a comprehensive Technical Report with the following sections:

a) Executive Summary

- Project overview
- Key accomplishments
- Technologies used

b) Project Overview

- Problem statement (from Phase 1)
- Solution summary
- Target users

c) Design & Implementation

- Brief recap of your architecture (reference Phase 2)
- Key design decisions and rationale

- Deviations from original design (What changed from Phase 2 and why?)
- Technical challenges and how you solved them
- Code structure overview

d) Features Implemented

- List of implemented features with brief descriptions
- Features NOT implemented (and why)

e) Challenges & Learning - *THIS IS CRITICAL*

- Technical challenges: What was harder than expected?
- Solutions: How did you overcome obstacles?
- What you learned: Technical skills, debugging strategies, time management
- What went well: What are you proud of?
- What would you do differently: If you started over, what would you change?

f) Future Enhancements

- Features you'd like to add
- Improvements to existing functionality
- Technical debt to address

g) Conclusion

- Project summary
- Personal reflection on the capstone experience
- How this project connects to your understanding of SDLC

h) References

- Any tutorials, libraries, or resources you used (proper citations and links)

Submission Requirements

What to Submit: Create a single submission folder containing:

1. **README.txt** - Index of all files and repository link
2. **Source Code Repository Link** (in README.txt or separate document)
3. **Final_Project_Report.pdf** - Technical report
4. **Demo_Video.mp4** (optional but recommended as backup)

Important Guidelines & Tips

DO:

- Focus on core functionality first
- Implement must-have features completely
- Test as you go, not just at the end
- Document as you code (don't leave it for the last minute)
- Commit code frequently to Git
- Ask for help when stuck (forums, google, office hours, other students, etc.)

DON'T:

- Try to implement every possible feature
- Ignore testing until the end
- Wait until the last week to start coding
- Skip error handling entirely
- Forget to save/commit your work

Technical Challenges Are Expected

- You will encounter bugs and obstacles—that's part of learning
- Document your challenges and solutions—this becomes great content for your report
- If you get stuck for more than 2 hours, seek help
- It's okay if your final product differs from Phase 2 design—just explain why

Code Backups

- Commit to Git regularly (at least every time you complete a feature)
- Keep local backups
- Don't rely solely on your laptop—computers fail!

What If Things Go Wrong?

- Plan B: If major features fail, ensure you can demonstrate what DOES work
- Partial Credit: A partially working project with good documentation earns more points than no submission
- Communication: If you encounter serious issues, contact me early—we can discuss options

Key Takeaways to Remember:

- Software development is iterative—problems are opportunities to learn
- Good documentation is as important as good code
- Testing isn't optional—it's how you ensure quality
- The process matters as much as the product
- Every professional developer faces the same challenges you're facing now

Final Thoughts

- This capstone project represents the culmination of everything you've learned about software engineering. You've taken an idea from concept to requirements to design to working code—that's the complete SDLC!

I'm excited to see what you build and learn from your experiences. Good luck!

Mohsen