CMPS 3390 Project 2

Fall 2024

BEFORE YOU BEGIN: LEARN YOUR TOOLS

When working in a team, there are some key tools, concepts and components you should familiarize yourself with. The more you communicate your intentions with your fellow team members, the less time you will spend code reviewing and refactoring.

Make sure you familiarize yourself with the following concepts and tools:

Team Communication

- Use tools like Slack, Discord, or Microsoft Teams to keep in touch with team members.
- Establish regular check-ins or meetings to discuss progress and challenges.
- Share updates and ask for help when needed to keep everyone on the same page.

Task Management

- Break down the project into smaller tasks and assign them to team members
- Use tools like Trello, Asana, GitHub Projects, or Linear.
- Set clear deadlines and priorities for each task.
- Track progress and adjust tasks as the project evolves.

Version Control

- Use Git and GitHub (or similar platforms) to manage changes to the codebase.
- Learn to create branches for different features, merge code, and resolve conflicts.
- Regularly commit changes and pull updates to ensure smooth collaboration.

Code Review & Refactoring

- Review each other's code to catch bugs, improve quality, and share knowledge.
- Use pull requests on platforms like GitHub to discuss and approve changes before merging.
- Foster a constructive and learning-focused environment during code reviews.

Documenting the Project

- Keep clear documentation of the project's goals, features, and technical decisions.
- Use tools like Google Docs, Notion, or Markdown files in your repository to maintain documentation.
- Make sure the documentation is accessible to all team members and updated regularly.

Part 1 - Research & Information Gathering

1. Team Communication
- Define what success looks like for your project.
- Think about how the app can make tasks easier, solve problems, or entertain.
- Make sure your goals are specific and achievable.

2. Create User Profiles
- Imagine typical users (e.g., a student, mother, bookkeeper, doctor, etc).
- Describe what they do, what they need, and how they might use the app.
- Use these profiles to guide your design and features.

3. Outline Key Scenarios
- List the main tasks users will perform with the app (e.g., making appointments, managing products, high scores, etc).
- Think about what steps they'll take to complete these tasks.
- Use these scenarios to shape the app's features.

4. List the Must-Have Features
- Identify the key things the app should do (like sending notifications, uploading files).
- Break down big features into smaller tasks.
- Prioritize what's essential versus what's nice to have.

5. Create a Simple Requirement Document
- Write down everything the app should do in clear language.
- Include diagrams or sketches that help visualize the different components of your app
- Make sure everyone involved agrees on the document.

6. Decide What's Most Important
- Rank features and requirements by importance.
- Focus on what will have the biggest impact on users.
- Be ready to adjust priorities if new ideas or issues come up.

7. Get Ready for Development
- As a team, decide what framework, database, and tools will best fulfill your requirements.
- Set up a meeting to transition from planning to building the app.
- Ensure that all members have their environments configured to develop and share project files.

Part 2 - Environment Setup

1. **Finalize Decisions**
   - Verify that all team members understand the design and scope of the project.
   - Come to a consensus on which frameworks, persistent storage, design patterns, etc. you intend to use
   - Decide which project management and communication tools will best suit your needs

2. **Configure Dev Environment**
   - Spend time as a team making sure everyone has the same environment set up
   - If you are on different operating systems, make sure any differences are accounted for
   - Make sure all of your versions match: IDEs, databases, runtime libraries, third party libraries etc.
   - Create strategies to ensure all members have the same environment. For example, if you are using SQL you should consider creating/populating tables using SQL scripts that can be pushed to your repo.

3. **Setup Version Control**
   - Have one member of your team do the following:
     - Create a simple starting project with an initial view. "Hello World" is fine
     - Create a local git repository and do an initial commit
     - Create a private remote repo on github, and push the initial commit
     - Add the other team members to the repo as collaborators, then have them do one of the following:
       - Clone the repo, make a small change,, then **commit and push directly**.
       - Fork the repo, make a small change, **commit and push to their fork**, then initiate a pull request.

4. **Setup Workflow, Discuss Timeline, Set Expectations**
   - Make sure each member of the team knows what they are responsible for completing
   - Set reasonable timelines to complete each piece of your application
   - Set a schedule with team members to discuss progress and adjust deadlines ([daily stand-up](#))
   - Consider using a project management tool like Kanban to assign and track individual tasks and goals

Part 3 - Develop Your Application

1. **Application Requirements**
   - Must contain a graphical user interface (GUI) with at least 3-5 views
   - Must contain at least two different data models (e.g. Employees/Schedules or Products/Orders)
   - Must contain a persistent data storage strategy that matches your models (api/web calls are not required)
   - Must implement all four CRUD actions for at least one of your models (Create, Read, Update, Delete)
   - Attempt to make the the user experience (UX) as cohesive and intuitive as possible
   - Anticipate user errors and malicious behavior by validating/sanitizing inputs

2. **Development Cycle**
   - Continue to meet with your team regularly
   - Discuss current progress and adjust expectations as needed
   - Focus on CORE functionality first, beware of feature creep
   - Make sure all team members are contributing to the project
   - Push incremental changes to your remote repository, **NO LEROY JENKINS!**

3. **Finalize And Present**
   - Decide what success looks like for your team. "When are we done?"
   - Prepare to present:
     - **One team member** will present and explain all documents created during **Part 1**
     - **One team member** will discuss their environment setup and challenges they faced during **Part 2**
     - **One team member** will build, explain, and demonstrate your team's application
   - Choose a timeslot to present and signup here:

PROJECT 2 PRESENTATION SIGNUP FORM