

Program Reports: A brief how-to

CPSC 2150

Program Reports

- This semester we'll be creating reports to go with our programs
 - The idea behind them is to help you start viewing assignments in the terms of the software development life-cycle
 - Stage 1: Requirements Analysis
 - Stage 2: Design
 - Stage 3: Implementation
 - Stage 4: Testing
 - Stage 5: Deployment

Stage 1: Requirements

- The first stage of our Software Development lifecycle and our report is Requirements Analysis
- We need to identify functional and non-functional requirements
 - Functional Requirements describe an *action*
 - Something the user can do in or with the system
 - Something the system can do for the user
 - Non-functional requirements are everything else
 - They may affect our functional requirements by adding restrictions on them
 - The system must <action> in less than 5 seconds
 - They may be hardware or software restrictions on the system
 - The system must run on Windows 10
 - The system must run on a machine with less than 2 gigs of RAM
 - They may be requirements on the way the developer must implement the system
 - The system must be written in Java

Requirements

- Requirements are written *before* any coding or design work has been completed
- Requirements generally come from our customers and users
 - May not be the same people
 - We don't have a real customer in our class
 - No one really wants a Connect 4 Game
 - We just have a pretend customer
- Requirements describe what the user or the customer wants from the system
- This can get confusing, because as an assignment there may be requirements that are not from a real customer or user
 - You can think of myself or the TAs as the customer if that helps.

User Stories

- User Stories are one way express our functional requirements
- They are a part of User-Centric design
 - Stay focused on the wants and needs of the User
- User stories are written from the User's point of view
- User stories contain 3 parts
 - The role of the user
 - The action they can take
 - The benefit or reason for the user.

Non-functional requirements

- Any requirements *for the program* that are **not an action**
 - Could describe speed or efficiency requirements
 - Could be about the OS that it must run on
 - Could be language or technology restrictions
- **Note:** some requirements on the prompt are just requirements for the assignment
 - Don't use magic numbers
 - Use good comments
 - Write contracts
 - Make a program report

Design

- For the design section of our report, we need to create some UML diagrams
- Usually:
 - Class diagrams for each class/interface
 - UML activity diagrams for methods
- It is very helpful to make these diagrams before you write your code
 - Class diagrams help you work out what data and methods you will need
 - You can ask yourself “What does each class know? What does each class do?”
 - Work out your information hiding and Separation of Concerns
 - Activity diagrams help work out your logic
 - Identify decisions that need to be made
 - Identify loops

Testing

- We will talk about testing in detail later this semester
 - How to format test cases
 - Techniques for identifying the best test cases
- Because of this, we aren't going to include testing in our reports until later projects

Deployment

- You must include instructions on how to compile and run your program on Unix
- Now that makefiles are required, these steps are simple
 - Navigate to the project directory on your command line terminal
 - Enter the command `“make”`
 - Enter the command `“make run”`
 - Later projects may have more commands in our make files
- However, you do need to provide a make file in order for these commands to work.

File Structure

- In order for your program to run correctly, you must follow a strict file structure
- Create a directory for your project. You can call it whatever you want, such as HW1
 - Inside this directory, you need to have 3 things
 - Your Package Directory
 - This must match your package name, i.e., `cpsc2150/connectX`
 - Inside of this directory is where your code files will be located
 - Your `makefile`
 - Your program report (as a pdf)
- Then you can zip up your project directory and submit that.