

Team Project and Software Development Life Cycle

Textbook

Team Project and Software Development Life Cycle

Requirements for the Turtles Team Project

- The team members should collaborate effectively and distribute tasks evenly among themselves.
- Each team member needs to create the code for a unique turtle.
- When the code is combined together, the turtles need to create a cohesive drawing. The drawing needs to be of something identifiable, not just abstract shapes.
- Each team member turtle needs to have the following:
 - Unique turtle shape
 - At least 2 colors other than black
 - A shape filled in with color
 - A loop
 - An if statement
 - Comments
 - An input statement
 - An example of pen up / pen down
 - A function

Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a structured process used to plan, create, test, and deploy software effectively. Its primary purpose is to produce high-quality software that meets user needs efficiently, on time, and within budget.

Walk through Software Development Life Cycle (SDLC) with your students. The Software Development Life Cycle is a method of working on software projects in a way that helps to make quality programs. It consists of the following steps.

1. Planning – Decide what your project will do and how to get there

2. Define Requirements – Get the requirements very clear
3. Design the Program – brainstorm, plan, storyboard, diagram, and layout the program
4. Software Development – Build the program
5. Testing – Test to see if your program works with a variety of circumstances
6. Operations and Maintenance – Keep your program updated and clear out bugs that come up

Following these steps will help guide you as you approach your group project.

Note: You can assign the team project as a group project on the platform if you wish. But you do not need to—this project can be completed without organizing your students into official groups on the platform.

Maintenance in the SDLC is crucial for ensuring software continues to function correctly and optimally after its initial deployment. It addresses bugs, adapts the software to new environments, and implements enhancements or new features to meet evolving user needs over time.

Everyone Has a Role: Teamwork in Action

When you work on a team project, it's not just about doing everything together—it's about each person having a **clear role** that helps the group succeed. In a collaborative team, everyone brings their strengths to the table.

Some common roles include:

- **Leader** – helps organize the team and keeps the project on track.
- **Researcher** – gathers important facts, images, or data for the project.
- **Designer** – creates visuals, slide layouts, or any graphic elements.
- **Writer or Editor** – puts ideas into words and checks for clarity and correctness.
- **Presenter** – speaks for the team or explains the final project to others.

When everyone knows their job, teamwork becomes smoother, more fun, and more productive. Plus, switching roles in future projects helps you build new skills and understand how every part of a team matters.

Actively Participate in Learning

Learning new things, especially in technology, can sometimes be challenging, but that's where the most exciting growth happens! To truly master a new skill, it's important to **actively participate** in what we call "effortful learning." This means you'll work hard to solve tough problems, both on your own and with classmates.

When faced with a challenging task, a key part of this is building **perseverance**: if one method doesn't work, don't give up! Instead, try to think of new ways to approach the problem, modifying your methods until you find a solution. Throughout this process, it's essential to **stay engaged and maintain a positive mindset**, even when things get tricky. Remember, errors aren't failures; they're valuable opportunities to learn.

When you're working with others, actively supporting your peers, sharing your ideas, and asking questions (of both your teacher and classmates) helps everyone understand better and reach their goals together.

Give and Receive Feedback

A very important part of active and collaborative learning is learning how to **give and receive feedback** from your peers. When you give feedback, aim to be helpful and specific, pointing out what's working well and suggesting ways to improve, always with a positive and respectful tone. And when you receive feedback, try to listen with an open mind, even if it feels a little uncomfortable at first. Consider different

viewpoints, ask clarifying questions, and use the advice to strengthen your work. This exchange of ideas helps everyone see their projects from new angles, catch mistakes they might have missed, and ultimately grow smarter together as a team.

Groups will present their work to classmates and receive valuable insights. This is your chance to improve your project and learn from others! When giving feedback, be kind, specific, and helpful, focusing on both strengths and areas for improvement. When receiving feedback, listen actively and with an open mind. Every suggestion helps make your project better and supports everyone's learning.

Cite Evidence

In math, coding, and indeed in all parts of life, it's not enough to just say you have the right answer or a great idea. You also need to be able to prove it! This means you need to cite evidence to explain and justify your reasoning.

Your Invitation: Whenever you present a solution, an answer, or even an opinion in our class, get ready to back it up. We'll be asking you to show us *how* you know your answer is correct. This might mean pointing to specific lines in your code, referencing a mathematical rule, using data from a problem, or explaining the steps you took to get there. Just like a detective uses clues to solve a mystery, you'll use evidence to explain and justify your thinking. This skill helps you build strong, logical arguments and makes your understanding much clearer to everyone else!