Growing Young Learners
Higher Education Programming

12.20.2017

Writers:

Shreya Goel

Rachel Lockerman

Shuai Yuan

Instructional Design Document

Table of Contents

NOTE: The website is currently a work in progress.

http://esite.io/projects/D02229f6761fd1849f270a18dec7d02a5/site/index.html

Problem Description

Introduction

We will design a system that teaches programming and web design to higher education students who are new to programming. Instead of just memorization of topics, our system will teach programming and web design in a manner that encourages "playing" around with the system, making mistakes and helping peers in debugging their issues.. It uses project-based learning that will help the students develop a rudimentary portfolio and give them the skills needed to continue their development in more advanced ways or in other languages. Our system will also teach about collaborative coding resources like Github, Stackoverflow, Slack and StackExchange. This is vitally important as programming and web design is becoming more necessary in today's changing world.

Background of the Problem

It's ironic that computer science and programing, a field that prides itself in being at the cutting edge of technology and development is mired in old, ineffective learning techniques. Many systems (Lynda.com, Codecademy, W3Schools) use "drill and kill" techniques- learning a method or terminology, using it once (if that), and moving on once achieving mastery. They are phrased as individual resource sites, that consider techniques and terms as isolated. Students also were forced to learn in isolation- even though continuing to grow as programmers require them to use and understand the collaborative resources available to them on the web (Giannakos, Pappas, Jaccheri, & Sampson, 2016). Quite a number of studies have shown the largest indicator of success with self-taught mechanisms is prior knowledge with programming in general. (Agarwal, Ritu, et al, 2000). Given that, when a student is interested in learning more about programming, but lack existing experience, they have difficulty resulting in frustration and frequent dropping of the effort (Soloway & Spohrer, 1989, pp. 294-296), (Giannakos, Pappas, Jaccheri, & Sampson, 2016).

For students who are interested in exploring programming, but do not have a background in it- the learning curve is too great for self-taught instruction. Students get frustrated and give up, or they don't learn how to apply the skills they may develop, making all of their efforts for naught.

As of yet, traditional computer science courses are taught in a primarily "direct instruction" (Giannakos et al., 2016) methodology, where the lectures describes a process which is replicated by the learner. This is problematic as the field of programming demands innovative thinking and the ability to thrive in ill-defined, loosely structured environments (Kadijevich, Angeli, & Schulte, 2013), (Ben-Ari, 1998).

The GYL system trains learners in mechanisms that are used and embraced are part of a new industry paradigm. It was built on leading instructional design theories as well as the new collaborative mentality expected in the industry (Kay et al., 2000, pp. 113-115). Given that programming requires a more dynamic approach, our training process orients the learners to our new collaborative mentality.

Given the major "skills gap" for people skilled and/or familiar with programming, especially in a collaborative setting (Akhriza, Ma, & Li, 2017, pp. 675-698), we anticipate high-demand for our training.

This project is vitally important because programming and web design is becoming a necessary skill to engage and grow in the professional world (Robins, Rountree, & Rountree, 2003). Already these skills are no longer limited to just "techie" job titles- but are becoming expected and needed across the professional spectrum. For our students to succeed and thrive in today's and even tomorrow's world, we need to provide a venue and environment that will facilitate that growth. Many are saying coding is the new second language. Many recognizing this trend are learning to code even later in their careers (Marino, 2017).

Problem statement: Learning Problem

Our learners see the future in programming, where understanding of these skills and systems are vital to succeeding in the world at large. Our system is designed for non-tech background students, in both teaching the "what" and the "how" including how to grow beyond the course. The GYL system uses peer-based learning to help ease the transition, and enable these novice students to embrace and grow their skills (Muller & Kidd, 2014, pp. 175-192).

Our design is based on tried and tested instructional design theory and application. Programming, as an active dynamic field, requires learning in a situated environment that reflects that (Kadijevich, Angeli, & Schulte, 2013, p. 143) Computer science and design requires knowledge of an interconnected workflow that can be updated and changed adapted as the world changes. What works today, may not work tomorrow. Our system will both teach programming as a whole system, that has many interconnected aspects. As well as teaching skills that embrace the collaborative nature of today's programmers. Gone are the days of a guy programming alone in a dank dark basement. Today's programmers collaborate, work off existing code (forking), and help each other fix errors. Our system eschews the stereotype of the loner programmer, because that's not what today's world needs. They require one to change the model as a whole. By basing our design in project-based learning, we teach not just individual methods or solutions, but learning to integrate things as part of a larger whole.

Target Audience

The target audience includes Higher Education students who have had limited to no background in HTML and CSS development. Our learners will be those interested in the cross section and increasing prominence of knowledge of web design and programming in today's world. These are not the stereotype of computer science students, they haven't focused on programming and web design as of yet, and must be acclimated to the interconnected, dynamic nature of programming (Zilan, Jing, & Pan, 2011). Our learners want to expand their horizons and skillset, seeing a working knowledge of programming and web design vital to thriving in today's world (Muller & Kidd, 2014, pp. 176-178). As our learners will not have a significant background in programming, our design also integrates the collaborative peer-based aspect programming has.

Learner Characteristics

We anticipate **two main learner profiles** for the GYL Learning system. As non-computer science majors, neither have a strong background in computer programming.

Career Track:

The first learner is one who is interested in programming as a potential career. The GYL system will introduces the new thinking required of a programmer as well as teaches them how to engage and continue to learn and grow to become a professional programmer.

Non-Career Track:

The second learner, is one who sees that knowledge of programming will be integral to a wide variety of fields, not necessarily based programming, hence has personal interest in the domain knowledge. Our system fully immerses the learner in understanding how programming works so they can serve and grow in their role.

Learning Context

Technical Context

High speed internet, dedicated computer capable of running Sublime (a code editor), headset with microphone.

Social environment

Chances of learning and communicating with peers with the same level of education; opportunities of seeking helps from experts or experienced peers; StackExchange, StackOverflow, Github. Full usage and understanding of these resources will be provided in the initial stages of the course. Our model does not rely on traditional top-down learning, rather it encourages learners to seek out additional information, resources, knowledge on their own time. They in turn, will be able to help their peers strengthen their own skills as well as thrive as future programmers. The GYL Learning system is designed in such that the learner can learn with their peers in a self directed learning environment on their projects.

Learning Goals and Objectives

The learning goal of this project is to support the development of effective programmers who will be prepared to use HTML and CSS to showcase their work in an efficient manner and collaborate with peers in the context of being a better programmer.

Given the different motivations of the learners dependent on their track, there are some divergent learning goals for the different tracks, while also having a good deal of overlap.

- The **ultimate goal for the career track** is they have the skills and capabilities to develop into a professional program, beyond what they learn from our (or any program). The nature of being a professional programmer is that they are able to not just implement what they know, but being able to find and integrate new methods into existing infrastructure.
- Whereas the **ultimate goal for the non-programming career track** is an an understanding of how programming logic works so as to understand that said skills apply to a broad range of fields. Therefore, both requires an understanding of how programming works, both in terms of the granular mechanisms of accomplishing mechanisms, as well as understanding of how programmers will work to accomplish goals.

Goals for both Tracks

- Able to articulate concepts and methodologies about different HTML and CSS codes;
- Able to create and maintain a fully-functional static website using CSS and HTML;
- Able to collaborate and plan with peers to develop webpages and sites;
- Able to critique, debug, and improve peer code, and learn from others' codes;
- Able to learn and implement new methods and techniques;
- Able to understand and meaningfully use collaborative resources such as StackOverflow, StackExchange, GitHub
- Able to use collaborative programming resources to solve issues.
- Able to understand and implement the proper methodologies and protocols of a collaborative programming project.
- Able to identify the time of need of collaboration and contribution;

- Able to contribute and access information relative to their queries.

Goals for Programming Career Track:

- Learners will be ready to learn and integrate methods that go beyond what they've learned within the system.
- Learners will be ready to learn other programming languages, understanding the methodologies and skills required.
- Able to use collaborative resources to integrate additional features, collaborate on projects with other programmers, and fork off existing code to build on others work.

Goals for Non-Career Track:

- Learners will understand the role their knowledge plays in different fields.
- Learners will understand the role programming/web design plays in their professional careers.

Review of competing (existing) projects:

There are for the most part two kinds of programming learning resources that occur online:

1. Learning resources that teach programming as a repetition module. These systems follow "drill and kill" methodologies having students "spitback" what they are learning., These learning resources also often teaches concepts in Isolation, rarely showing how new concepts can be integrated into existing projects or methodologies. They are also often "static" and don't teach learners how to keep up with the new methods , and how to upgrade when older methods are deprecated and no longer supported.

   Tools like this include W3 Schools, CodeAcademy, and TeamTreehouse
2. Learning resources that contain information on newer tools or methodologies, but they fail to teach
   a. The basics of programming itself.
   b. How these newer methodologies will integrate with the existing programming structure.

   The tutorials and resources that teach to these tools, assume one already has a working knowledge of programming.

   Tools like this include: StackOverflow and Github.

Neither of the systems build an active collaboration or practice with debugging and improving peer code- an essential part of a programmers development. Thus, there is a clear lacking in a learning environment that teaches novices how to program, how to grow and find new methodologies that are not taught within the program structure, how to fix and build on other people's code, and how to utilize online collaborative resources to improve one's abilities as a programmer.

The GYL system will not only situate learners in an environment that will familiarize them with how to program, but encourage them to make mistakes, and fix them. Our system takes into account that learners need to understand how to utilize collaborative resources, understand the overall structure involved in building a website, understanding how to utilize online collaborative resources, as well as how to debug and fix peers code.

Design

Content Analysis

The content is modular and sequential. Modules and section build on each other for accomplishment of learning goals.

Set Up:

- **Select a Track** allows learners to decide which learning track they better identify with. "Learning as a Career" for the learners that want to explore programming as a career and "Non-Career" for learners that want to explore programming as it can apply to a variety of fields.
- **User Homepage** shows the learner the modules and units, and where they are in relation to the overall project. It will show them their progress as well as a "progress so far" so they see all they are able to accomplish and how it can apply to websites.
- Basic page **introduce the most elementary HTML and CSS knowledge** with texts, pictures and videos, after which there is a playground allowing learners to play around with the basics. The playground has an input field taking in learners' codes and a sidebar showing the effects of the codes synchronously. This page consists of several columns showing all essential and critical HTML and CSS concepts needed in real-world development. Those columns could be access with users click on them.
- **Bridge page shows the external resources on the Internet** that learners can utilize to boost their learning. Some platforms like Codecademy, Freecodecamp, codepen, W3School, and Stackoverflow are useful to explore. In guiding them to those websites, GYL learning does not only attach the web URL on its website, but present detailed instructions on how to use those websites efficiently and effectively.

- **Spiritual page** lists what attitudes a qualified and successful programmer needs to have. This page also contains lots of individual experience of learning codes about either technical review or emotional changes. We hope to encourage learners with optimistic attitude and build up their confidence with various successful experience of learners who became experienced programmers from novices.

Media Selection

The GYL Learning System is a ***website/webapp*** that integrates resources from around the web. The delivery platform was so chosen because it is best to situate the learner as close to the learning reality as possible. It situates the learner in a customized learning environment that guides them based on A: their track, and B: their current progress/efforts. The instructions, guidelines and context are geared towards the particular learners, while guiding them in developing their website.

Considering the learning goals for our learners who can continue to grow as web developers, the platform reflects on their needs. Both our learning system and the learner's projects which they are developing and deploying are based on a website platform.

Delivery Plan:

GYL Learning majorly has 3 units comprising of further modules and sections. A major chunk of the modules will be accomplished asynchronously. These use a combination of "code-along" videos and detailed supporting text instructions that guide the learner in adding and improving on their content knowledge, execution and project development. A combination of media for teaching and learning helps the learners grasp content knowledge better and more effectively (Mayer, 2003).

When learners are just getting started with the system, the instructions (both video and text-based), are very explicit as to what the learner should be doing. This serves to acclimate the learner to the mechanisms and methodologies or implementing code on a website using Sublime Code Editor for web development and practicing their coding skills.

As the learner becomes more adept with the system and methodologies, the "code-along" videos and supplementary text become less explicit and more ill-structured (Reiser & Dempsey, 2017, pp 63-67) this methodology helps guide the learner to being an independent programmer who can search and adapt techniques methods outside the confines of the GYL learning environment. These later modules integrate peer-based collaboration, using tool such as GitHub, Stack Overflow and Stack Exchange.

The GYL system embraces a constructivist mentality, guiding our learners to be self-sufficient in the ill-structured world of computer programming.

Requirements:

Learners will be at the minimum must be using a moderately powerful laptop/desktop (one that is capable of handling the Sublime Code Editor), Windows/Mac, that supports a high-speed Internet connection, has audio speakers or audio jack to support speakers or headphones, for the audio support needed in code along videos. This will be the hardware requirements for the learners as professionals, so it's important to situate them in the proper environment now

<u>Project Description</u>

GYL Learning is designed for higher education students who are new to programming and web design. Although there are currently many other websites and systems available for the learner to acquire the required skills in programming and web designing, but they have certain instructional design challenges as discussed in our problem statement, like isolated learning, not filling the gap between knowledge of content and the application phase.

GYL Learning's instructional design focuses on filling such gaps and giving the learner the complete knowledge of the content via self-directed learning, collaboration skills and application of knowledge and skills through peer based learning and project based learning while keeping them motivated to learn and contribute.

Currently our assumption is that there will be 2 main types of learners who will pursue this course. First, those who are interested in converting their skills from this course to a potential career. Second, those who are interested in gaining a perspective and practical knowledge of programming and web designing to advance their skill set.

Through this course, learners will:

- Gain the knowledge of programming with HTML and CSS
- Maintain a fully functional website (Course Project)
- Collaborate with peers to critique, debug and improve peer code. (Collaborative systems like StackOverflow, StackExchange, Github, Slack are introduced, cooperated in the course work for learners to gain complete practical understanding of importance of collaboration in programming industry)

More about the elaborated goals and objectives can be read in the Learning Goals and Objectives section.

- At first, learners work on projects that are only seen and used by them, and explore basic subjects, like integrating formatting and proper syntax structure.

- Once the learners have mastered that, they begin to develop their project that integrates the knowledge they've developed until this point, as well as will continue to integrate future modules. At this point in the learning structure, they will be presented with less formally structured problems.
- Once the learners have gotten familiar with working with ill-structured problems and designs, they will begin to collaborate with their peers on their projects. Later modules will have peers "fork" off each others code, find and fix errors and issues with their peer's code, and integrate additional features.

Our System (GYL Learning):

- GYL Learning is offered as an asynchronous peer based learning system geared towards college-aged, non-computer science majors. Today's programmers work together to debug code, fork off new projects, and collaborate. Our design focuses on how these skills can be applied in "normal" life. Project based learning using a peer-based collaboration model focused on meaningful learning with peers.
- Fixes the issues about isolated resources by framing our design in scaffolded project based learning, and peer learning. Students will learn techniques that they apply to a growing project, and pairs them with peers to collaborate on. Instead of memorizing concepts that they won't see again, the GYL Learning System teaches methods and sequences in the scope of the larger design.
- This helps our learners, who do not have background in education, understand concepts as they are applied. Instead of learning a concept/technique that they won't use again- they immediately apply and visually see how that technique works and how it interacts with the rest of the project.
- Our system also takes into account how most online learning resources for programming present isolated concepts or methods. When trying to learn as a novice, such resources are like ingredients without a recipe- certainly the building blocks are there- but one does who does not know what to do with them, can not make food.
- The GYL System, aims to **bridge the gap** between the novice and professionals, by guiding novices in learning the not only specific methods, or resources but how to apply them.
- As of yet, learning to program and do web-design resources is either very limited in scope, or not designed around novices. A major issue is with "Drill and Kill" learning for programming,

is that programming is a dynamic ever-changing beast. One cannot learn a specific technique or methodology, and rely on that forever. Our system will not only teach programming, but how to continue to learn and integrate methods from online resources in the future.

- The GYL system encourages learners to make mistakes, play around and immerse themselves in the joy of the new skills. **Our system is geared towards adults**, who are interested in broadening their skill set to fully thrive in today's increasingly technology heavy world.

Specific Functionalities

- Targeted Audience Context
- Code Along Videos
- Need Additional Assistance
- Ask an Expert
- Peer Based Learning
- Project Based Learning

General

In sum, the GYL system has the learner build a fully functional- dynamic website. But there is a lot more involved than that.

Target Audience based Learning Context

First, the learner is asked to select a track- that track determines the context of the lessons presented.

The GYL system understands that people have different motivations for wanting to learn to program. As we discussed in the Target Audience section, we believe there are two main ones: Those that want to explore programming/web design as a potential career, and those that want to explore programming in general, as it may apply to a diverse array of fields. Based on that, the system provides different context and ways the tools and methods are presented.

For example, when learning about Github, the career learners will learn more about the granular aspects of forking code, merging code back together and building projects, while the non-career learners will learn more abstractly how Github will be used to accomplish design goals, the various kinds of projects that can be done in Github, and how it may apply to their work.

The "Code-Along" videos and instructions are the same for both tracks, but the environment situates the learner based on how they will apply that knowledge- keeping motivation high for the individual learner, and contextualize the information as to how they will apply it in their careers.

The learner first plays around with general HTML concepts, changing font colors and sizes- just acclimating to the idea of markup instead of WYSIWYG type formatting.
The learners are then guided through the process of installing the Sublime IDE, which will be used for the web development in this platform. In early modules, there is a good deal more "hand-holding" than in later modules.
This both acclimates the learner to the concepts and mentality that they need to know to succeed, while maitaing high motivation so learners don't get overly frustrated and discouraged.

There are three overarching units: "**Starter**", "**Intermediate**", and "**Advanced**".
Those units are split into modules, and modules are split into sections, which each contain instructions and context- the overall functionalities, remain fairly consistent within the module itself. There are several important overarching functionalities that are present in different modules and units:

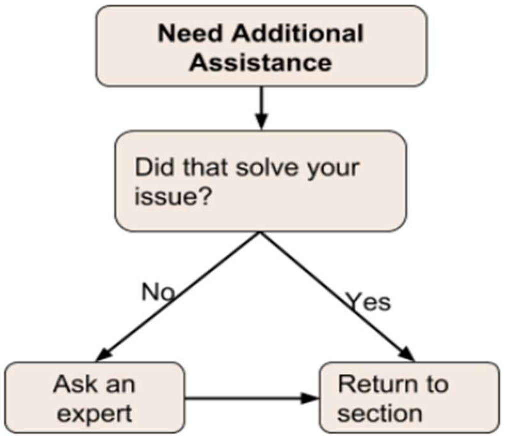
"Code-Along Videos"

Sections are all accompanied by a "code-along" video. The code-along video is built to accompany the relevant section, guiding the learner along the task. For the most part the Videos are structured as such:

1. How the technique described in the video will be used in general, and how it will fit in with the previously described methods. In essence- what is the "learning goal" of the section.
2. The proper location and phrasing of the code
   a. At the start, this will be very explicitly detailed as to both the code that needs to be used, as well as the location that the code needs to be in.
   b. As the learners progresses to the more advanced modules, the videos get more loosely structured, as learners get more acclimated to the proper methodology and mechanisms of programming, facilitating their journey to being self-directed programmers.
3. What the output of the project of the project will look like after the implementation of the code.
4. Summary of what has been accomplished so far.

Within the modules there are two important "help" functionalities that exist in each module. "**Need Additional Assistance**" and within that "**Ask an Expert**".

Figure 1. "Need Additional Assistance" Workflow



Need Additional Assistance:

If the learner is having issues understanding the instructions they can click for additional context, help, or general assistance. This can be reminders to check headers, style guides etc..

"**Ask an Expert**"

If the learner is still having issues, there is a chat button within the **"Need Additional Assistance"** module so they can ask someone for assistance if the provided assistance is insufficient. This will be staffed by professional experts who will have access to the learners code. The learner will identify specific issues they are having with their code (such as "it doesn't run and I don't know why", or "how do I incorporate these attributes") and the expert advises them. Such programming "helpdesk" services already exist. We analyzed "HackHands" and "Prestoexperts" to model our code help desk on.

Our "Ask an Expert" will be people who specialize in both in the code they are asked to focus on, as well focuses on education, tutoring programming.

At first, we will be applying for the Google.org "Code-Across America" grant to fund this project, specifically because they provide specialists in programming and programming education. Later, we hope to set up pipelines for industries who need people with the skill set we're helping develop, and accrue revenue that way. This model would follow the very successful Udacity and Coursera career pipeline models.

"Peer Based Learning"

Peer based Learning begins to come into play starting with the Intermediate Unit, and continues through the Advanced Unit.

Integrated into the sections and units, peers will be matched up with those of like interests and goals,. Based on the current lesson, peers are matched with each other to critique and improve on each others code.

- Take the peers code, create a fork of the project on GitHub.
- Fix any bugs within the code, and make suggestions so the code is more in line with the appropriate style guides.
- Integrate a more advanced version of the methods from the section into the code.
- Share the fork back with the peer.

“Project Based Learning”

Project-based learning begins to come into play starting with the Intermediate Unit and continues through the Advanced Unit.

Just like in the “real world” learners don’t apply concepts in isolation, they must be used within an existing infrastructure and learn how those concepts and methodologies can adapt and change within that infrastructure.

In the Intermediate Modules and onwards, students begin constructing a fully functional website. Each module focuses on a different aspects and methods used in a website, incorporating best practices and style guides.

These functionalities are incorporated into the different units, specifically focuses on building
The Unit module set-up structure is as follows:

<u>Functionalities of Starter Unit Modules</u>

- At the beginning of the Starter Unit, Learners introduce themselves to each other using FlipGrid- making short, minute long videos about themselves and what they hope to accomplish in the system.
- Learner is shown the context as to how this lesson fits in to the previous lessons, as well as how it will be used in their future career (such help text depends on the track of the learner).
- Learner is given a step by step guide as to how to accomplish the goal of the section. This guide is in a “code-along” video that appears on the side of the screen. The video has the instructor model the task, so the learner can follow
- The instructions are also shown in text on the website itself.
- Learner integrates the code as described in the video into their project.
- Learner is shown errors with their code, and an explanation of how it’s incorrect. Studies have shown this kind of instant feedback and correction is essential to maintaining high

motivation and success rates with novice learners in computer science (Erhel, S., & Jamet, E. (2013).

- Ie: a missing semicolon will show an explanation on the role the semicolon serves in the program, and shows how failing to end it results in the program not understanding the command/instruction was finished.

- Learner is shown a summary of what they've learned, and do a "mini-project" that encompassess everything they've learned.

<u>Functionalities of Intermediate Unit Modules</u>

- At the start of the Intermediate Unit, Learners make another Flipgrid video about themselves and what they've learned so far- At this point, learners are paired up to begin collaborating on code based on mutual interests.
    - One Section in this module focuses on "forking" and collaborating on Github.
- The **Project-Based learning component begins here.** Learners begin constructing a website that encompasses all aspects taught until this point.
- In the Intermediate Unit, learners continue to integrate lessons into their ongoing project. Learner is shown the context as to how this lesson fits in to the previous lessons, as well as how it will be used in their future career (such help text depends on the track of the learner).
- Learner is given a step by step guide as to how to accomplish the goal of the section. This guide is in a <u>"code-along" video</u> that appears on the side of the screen. The video has the instructor model the task, so the learner can follow.
- These guides are more loosely structured, encouraging learners to find resources that serve the need to the assignments.
    - In the early modules in the Intermediate Modules, learners are given exact links and shown where in the webpage to find the material, as learners advance through the Intermediate modules, they are given scaffolded less explicit instructions, with the final modules at the section directing learners to the website that will contain the information they need. If learners are struggling to locate the resource, there will be a "Need some more guidance" help button that will more specifically direct the learner to the resource they need.
- Learner integrates the methods into their project, they are encouraged to adapt those methods to serve their own interests and tastes, while conforming to recommended style guidelines.

- Learner is shown basic errors with their code, and an explanation of how it's incorrect
    - Ie: a missing semicolon will show an explanation on the role the semicolon serves in the program, and shows how failing to end it results in the program not understanding the command/instruction was finished.
    - It is expected by this stage in the unit, the learner is not making such simple mistakes, if mistakes are frequent and recurrent (ie: the same type of mistake is made repeatedly, they have a "Refresher" module shown that helps reinforce the proper methodology that the learner is having issues with.
- **Peer-based learning begins here:**
    - Learners for each week are "paired" with someone whom they have mutual interests with
        - It should be noted, that everyone will have different mutual differences pending on the individual learner. Therefore, while every effort will be made to pair people based on mutual interest, it can't be guaranteed.
- Learners look at their assigned peers code:
    - Identify and comment out errors made
    - Suggest Style Changes to be better in line with the Style Guide
    - "Fork" it off, and add an additional feature/type/etc.. That the module suggested.
    - Share the code back with the original author, including comments, and the additional feature.
        - For example, if the module focused on embedding pictures, the peer additional code could include a gallery of pictures, or a video.
- Summary of module, and how it will play in the future.

<u>Functionalities of Advanced Unit Modules</u>

- **Project and Peer-based learning** continues in the Advanced Unit Modules.
- At the start of the Advanced Unit, Learners have the option to "merge" like projects together and collaborate on their final projects. While some learners will not take advantage, we foresee many learners doing so.
    - Note: There will continue to be mini project "SandBoxes" so that Learners can "play" with code before integrating it into their final projects.

- In the Advanced Unit, learners continue to integrate lessons into their ongoing project. Learner is shown the context as to how this lesson fits in to the previous lessons, as well as how it will be used in their future career (such help text depends on the track of the learner).
- Learner is given a step by step guide as to how to accomplish the goal of the section. This guide is in a "code-along" video that appears on the side of the screen. The video has the instructor model the task, so the learner can follow.
- These guides are loosely structured, encouraging learners to find resources that serve the need to the assignments.
    - In early modules in the Advanced Learning Unit, learners are directed to the exact website (but not webpage) that will contain the information they need. As the lerner advances through the unit, this help turns into the type of website the learner needs (IE: a help website, or a code repository website). Towards the end of the advanced unit, the learner is expected to know how to find the resource they need on their own. However, there will be an "Need some more guidance" help button that will more specifically direct the learner to the resource they need.
    - Learners for each week are "paired" with someone whom they have mutual interests with
        - It should be noted, that everyone will have different mutual differences pending on the individual learner. Therefore, while every effort will be made to pair people based on mutual interest, it can't be guaranteed.
- Learners look at their assigned peers code:
    - Identify and comment out errors made
    - Suggest Style Changes to be better in line with the Style Guide
    - "Fork" it off, and add an additional feature/type/etc.. That the module suggested.
    - Share the code back with the original author, including comments, and the additional feature.
        - For example, if the module focused on embedding pictures, the peer additional code could include a gallery of pictures, or a video.
- The Advanced Unit focuses on finishing up the cumulative project and bringing all the aspects learned until this point together.
- The Advanced Unit finishes with an industry expert evaluating and providing feedback

Learning Theories: Theoretical Framework

The GYL system embraces the constructivist learning theory for our model. Constructivism emphasises situated learning and facilitated learning. Our system does not have students mindlessly mimic a instructor. Rather, we embrace the constructivist "guide on the side" by encouraging students to play in the system, make their projects their own, and make mistakes. (Reiser & Dempsey, 2017, pp 63).

Project based learning

Project based learning was selected so as to help students build projects that also build their skills. This can simplify the outwardly confusing and inscrutable nature of web design in order to teach students the process (Giannakos, Pappas, Jaccheri, & Sampson, 2016). A key component to successful learning of programing and web design is not just learning isolated concepts, but understanding how those concepts are applied as part of a larger whole. Project based learning helps us achieve our learning goals of learners becoming fully-capable web developers, even after they leave our "classroom."

- The learners will design a fully functional website. Each week, they will learn a new methodology/project and implement it into their site-in-progress. As the weeks progress, they will build out their site in line with their new skills.
- By the end of the course, their site will be a polished one that showcases the learners newly developed skills, and can be featured on a professional portfolio.

Peer based learning

Peer based learning was selected due to the novice skill level of the entering student. By utilizing peer based learning, students can help each other with difficult problems and help their cohort succeed. There is a distinct issue with novice programmers who have difficulty keeping up with the material struggling to understand concepts, and often, dropping the class entirely. Our methodology of peer-based learning is based on the theory and data that shows novice programmers that work collaboratively understand concepts more quickly, with deeper understanding, and significantly higher completion rates (Williams, et al., 2002, pp. 197-212).

We also want to ingrain the mentality of collaboration while programming in our learners from the onset. Programmers in today's world need to know how to navigate and utilize **"collaborative" programming** resources such as StackExchange, Github, and StackOverflow. These resources will guide our students in their development as programmers after they complete the course.. Programming is not a stagnant skill. What is learned today, will not necessarily work in six months, a year, two years, etc.... More important than the specific programming methods taught, is learning how to find information on how to accomplish a goal and how to integrate that information in an existing structure.

- Learners will be paired with others of similar interests on week-to-week projects and collaborate on assignments. They will be expected to not only help develop their own work, but build on their peers work as well.
- Learners will be required to "debug" each others code, add additional features and help their peers succeed. Since our learners are more or less coming into this course with the same skill level, we expect them to proceed together at a similar pace.

Self-Directed Learning

Self-Directed Learning was chosen because the issue isn't lack of access to resources about HTML and CSS, but how to convert knowledge of those resources into a fully functional project. As the nature of programming and development is constantly changing, **Self-directed learning is essential for any programmer or web-developer**, and we integrate that as well into our Project-Based Learning methodology.

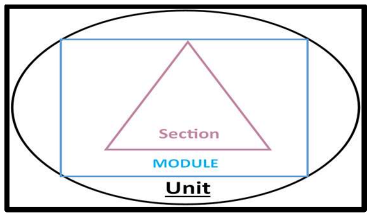
- The website the learners will develop won't be some abstract generic "hello world" type of website. Rather, it will be developed in line with the learner's interests and goals.
- The skills required to build the website are not merely taught in the tutorials, but are slightly above their current level, which requires them to learn on their own from provided external resources.

The majority of resources for programmers, are geared towards individual concepts, not about learning how to integrate those concepts and methodologies into a cohesive project. Further, many of these resources have a barrier that is to high for a novice to grasp. GYL Learning system, will teach students not only how to integrate a myriad of different resources into their projects, but how to go forward and do so with other more advanced resources and projects, including learning on their own.

This is why self-directed learning is so essential to the process of becoming a programmer. Learners need to be able to not only replicate what they've seen done already, but learn how to adapt and grow those techniques and tools into fully functional projects (Latchem 2016). Knowing how to utilize independently found resources into an existing project, is how programmers thrive when they want to apply those skills in the so-called "real-world" (Conradie, 2014).

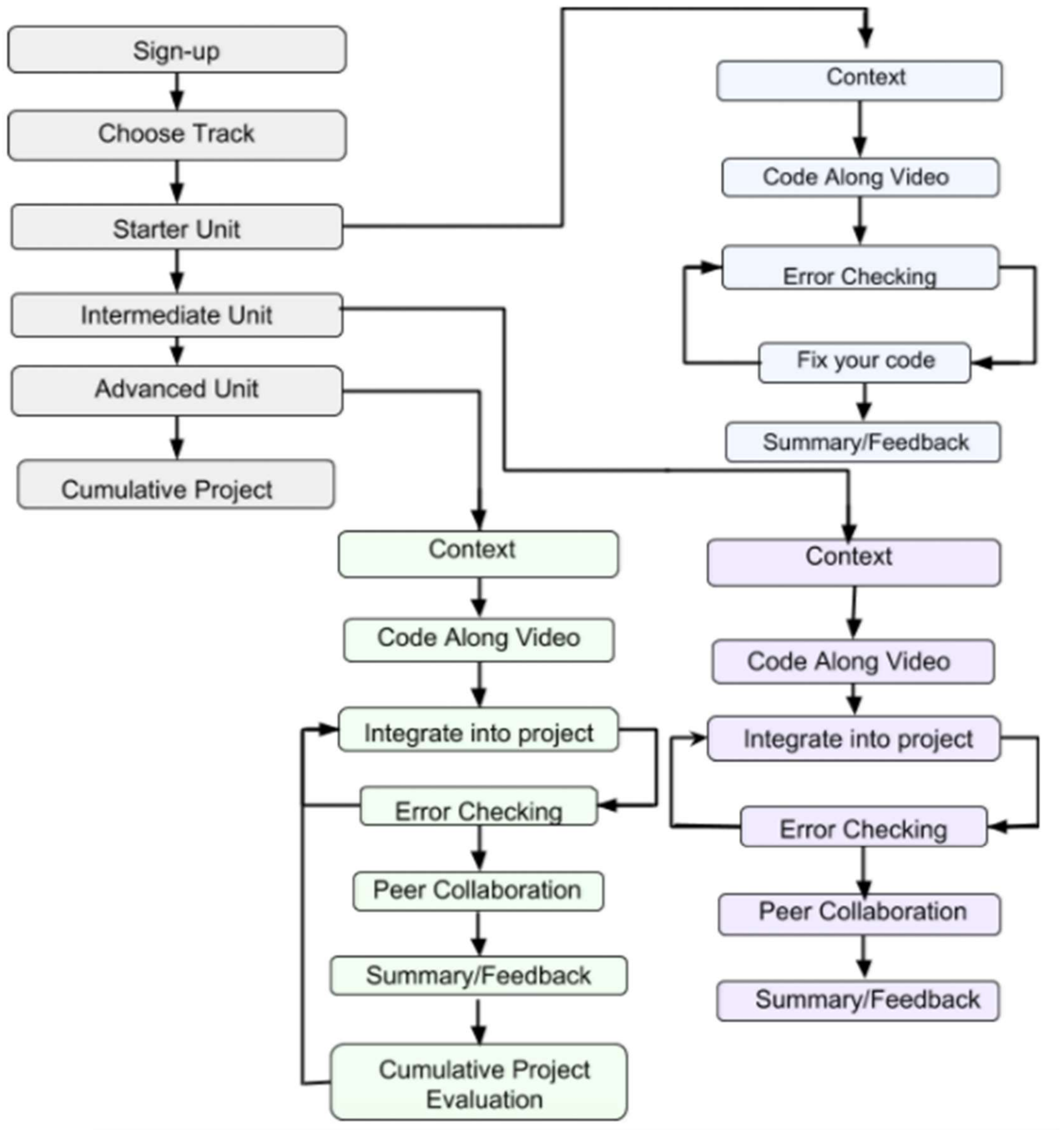
Information Architecture

Figure 2: Unit Architecture



The GYL system is split into 3 units. Each unit contains modules, which are dedicated to an overarching concept. Those modules are divided into sections each which focus on accomplishing a specific skill or technique that serves their overall module aim, and ultimate unit aim.

Figure 3: Overall Work Flow of GYL Learning



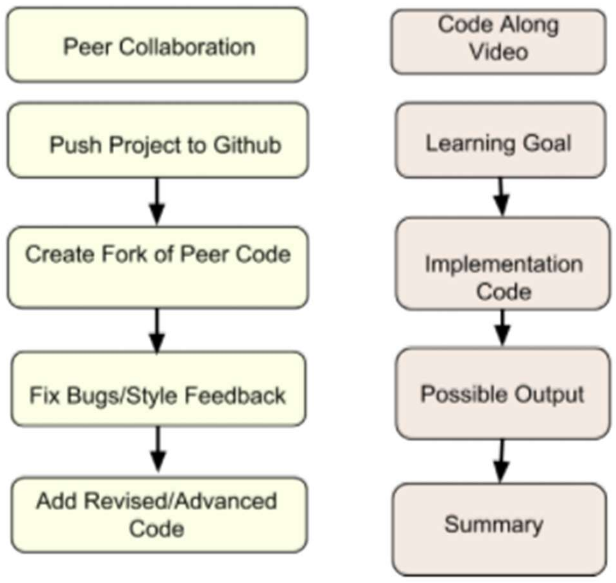
This is the overall workflow of the GYL Learning System. Each track leads into another., and each module contains sections which lead into another as well.

Unit Workflow:

The overall setup is- **Login**→ **Choose Track** (between Career and Non-Career)→**Starter Track** (which introduces the learner to basic concepts and methods)→**Intermediate Track** (which learners start to incorporate the methods they learn into a cumulative project and collaborate with their peers) →**Advanced Unit** (in which learners fully flesh out the cumulative project and continue to collaborate with peers). The advanced unit ends, with an **expert giving feedback** on the website, and the learner integrating said feedback into the project.

Within the different aspects of the GYL system, there are workflows as to how those aspects work. Below is the workflow for the "**Code Along Video**" and the "**Peer Collaboration**"

Figure 4: Workflow of different aspects of the GYL System

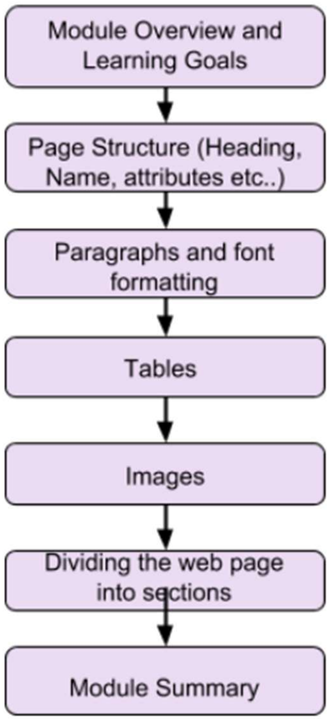


The "**Peer Collaboration**" part of the learning section is in the "Intermediate" and Advanced" units of the learning system. It occurs after the initial error checking and feedback part of of the learning section. It comes with it's own error checking, integrating into project aspects (just like the primary learning section does). Once the learners share the forks back with the original learner, the learner can make the suggested changes (which they may not have spotted on their own). We also anticipate the learner will use the peers alternate perspective in their project to improve on their own code.

The "**Code-along Video"** aspect of the learning section is present in all three units in the system. After the learner has read the context about the lesson, the video walks the learner through the process of integrating the code into the program. In the starter unit and early intermediate unit, these videos are explicit; showing the learner exactly what and how to implement the code to achieve the desired result, In the later intermediate and advanced unit, the videos get more loosely structured, shifting to a more constructivist ideal.

Module Workflow

Figure 5: Sample Module:


Module Overview and Learning Goals
Page Structure (Heading, Name, attributes etc..)
Paragraphs and font formatting
Tables
Images
Dividing the web page into sections
Module Summary

Modules are set up to accomplish an overall aim (ie: style guides, page attributes etc..). The module begins by setting up context and learning goals for the overall module, and directs the learners to the relevant section. For example, in the above module, learners are focusing on integrating various attributes to their code (structure, paragraphs, fonts, etc…), each section in that module follows the below workflow (also shown above in the overall workflow)

Section Workflow:

The sections is where the learners will really get into the "nitty gritty" of learning how their code influences and applies to various sections. Each section is dedicated to accomplishing a different aspect of the module aims.

The Starter Section workflow is as follows:

- **Learning Context** is presented as to how this method will be used, how it relates to prior content, and the role it plays in the "Big Picture." This context is different depending on the "track" the learner is a part of.
- Then, the learner is shown a "**code-along video**" that describes the steps needed for them to accomplish the goal of the section. The steps are also detailed in text as well.
- The learner then **runs their code** and the system automatically gives feedback on their code and corrects their errors.
- The learner takes the feedback, and corrects their code.

The Intermediate workflow continues:

- Sections from here on out involve building out a cumulative project.
- The learner pushes their code to GitHub, and is matched up with a peer and shares their repository with them.
- The peer "forks" off the code and adds suggestions and feedback though //comments
- The peer adds a more advanced version of the code that was learned in that section and shares the fork back with the original learner.
- The learner looks at the feedback, and integrates relevant aspects into their own code.

The Advanced workflow continues:

- All of the work until now had been focused on building a cumulative project. In the advanced unit, the instructions focused on that is dedicated to more advanced topics as well as becomes more loosely structured.
- The final project is shared with an expert in the field, who provides feedback and critique on the final design. The learner incorporates that feedback into their code.
- The unit (and GYL system) ends with a summary of what's been learned, and where to go from there (dependent on track).

Further modules will delve into indirect programming skills, such as use of collaborative programming resources like Github and StackExchange/StackOverflow.

Purpose of Github: To maintain the files of code in place so that all the team members can help improve wherever needed.

Purpose of Stack Overflow: To look or solutions to debug issues in their code and finally learn how to help their peers on this platform.

Examples of skills learned in later modules:

➔ Embedding videos in web page.

- ➔ Menus: Side, Top, Hamburger etc.
- ➔ Adding Graphics and Animation.

Information and Interaction Design

GYL Learning is an online web based application to aid learners who are interested in exploring HTML and CSS Programming and web designing course for higher education students. It will be a resource guide for learner to keep track of their progress in terms of modules completed, what is coming up, what is due, and collaborate.

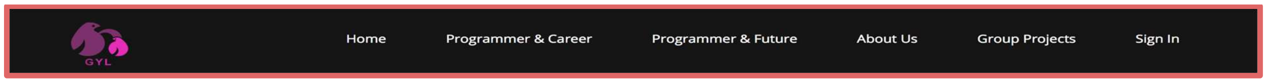
Implementation

The design will be implemented in the form of a website. Information presented on the landing/front page will include some initial learning functions and instructions. Learner would be able to view the basic introduction about our program and will also be able to understand the learning goals. This page will also include the section where the learner will be able to reflect on their motivation and select their preferred learning track and the corresponding curriculum as we provide with the tracks respectively. Instructions and curriculums will be implemented in the inner system which would be shown only after learners log into their accounts.

Prototype

Our prototype will be a functional website, which is under construction. We have achieved an example of almost every design in our document. Below are the featured functions of our prototype:

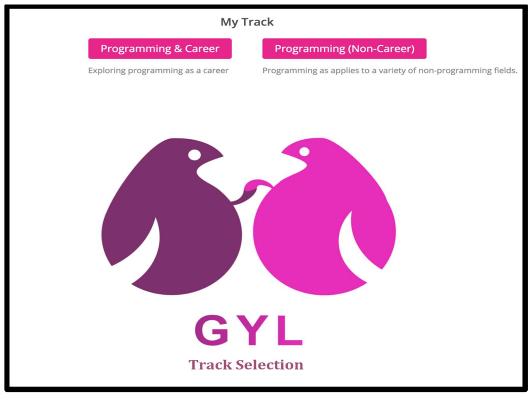
Figure 6: Taskbar (Main Menu)



"Home" which in general describes the project | "**Programming and Career**" + **"Programming and Future"** which shows the two tracks for the two types of learners | "**About Us"** which describes our learning methodologies and functionalities | "**Group Projects**" which describes projects of our learners | "**Sign-in"** which is how learners will access the system.

This taskbar is designed keeping in mind easy navigation of tolls and content in the website for the learner. If it's the learners first time accessing the system, after they sign up, they'll be prompted to select a track.

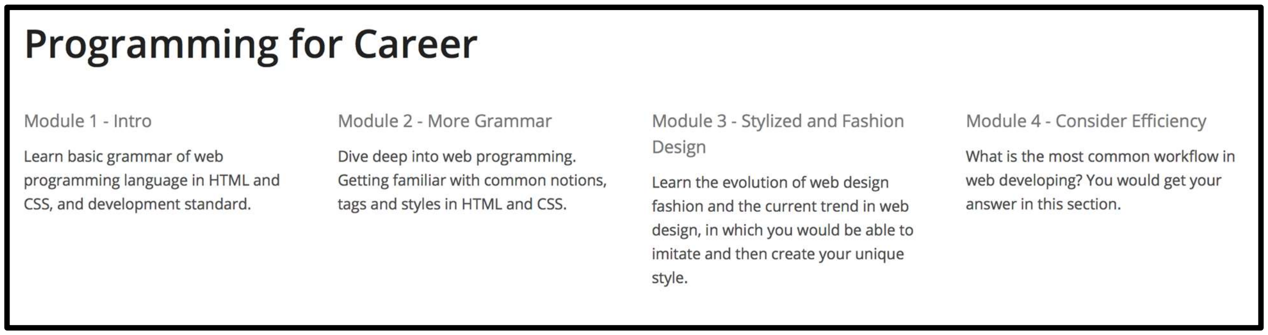
Figure 7: Track Selection


My Track
Programming & Career
Exploring programming as a career
Programming (Non-Career)
Programming as applies to a variety of non-programming fields.
GYL
Track Selection

After signing up, learners will be asked to select a track. The options will be "**Programming and Career**" and "**Programming (Non-Career**)." This selection contextualizes the learning in the learning modules. So methods are taught as they would be relevant to the context, learner motivation, aspiration and choice.

Functionality: This is interface will help the learners formally make the decision and direct them into different learning tracks which, in our system, means different instructional pages being shown afterwards.

Figure 8: Inside a Unit- Career Track (Contains Modules):



Programming for Career

Module 1 - Intro
Learn basic grammar of web programming language in HTML and CSS, and development standard.

Module 2 - More Grammar
Dive deep into web programming. Getting familiar with common notions, tags and styles in HTML and CSS.

Module 3 - Stylized and Fashion Design
Learn the evolution of web design fashion and the current trend in web design, in which you would be able to imitate and then create your unique style.

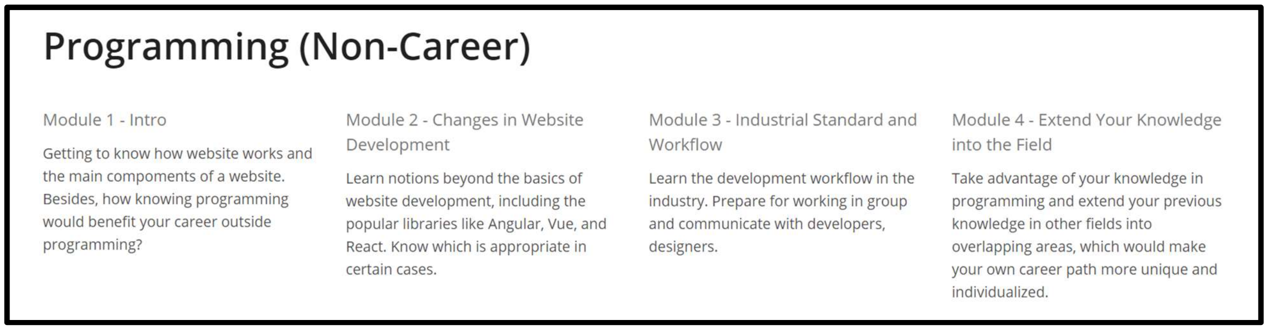
Module 4 - Consider Efficiency
What is the most common workflow in web developing? You would get your answer in this section.

Once the learner will have selected their track, they will be shown the "Module Homepage" for their track and the current unit. As learners complete modules in the unit, the next module will be shown and on completion of the unit, the next unit will be shown/unlocked.

An important note is, as the progress, previous modules remain visible. This way, if learners want to go back and refresh knowledge on the particular methodologies, they can. Learners will also be shown brief "refresher" modules if they make repeated mistakes with the same concepts in already passed modules.

We chose a minimalist design for the navigation so the learner can clearly and easily find the information they want, reducing unnecessary cognitive load.

Figure 9: Inside a Unit- Non Career track (Contains Modules):



Programming (Non-Career)

Module 1 - Intro

Getting to know how website works and the main compoments of a website. Besides, how knowing programming would benefit your career outside programming?

Module 2 - Changes in Website Development

Learn notions beyond the basics of website development, including the popular libraries like Angular, Vue, and React. Know which is appropriate in certain cases.

Module 3 - Industrial Standard and Workflow

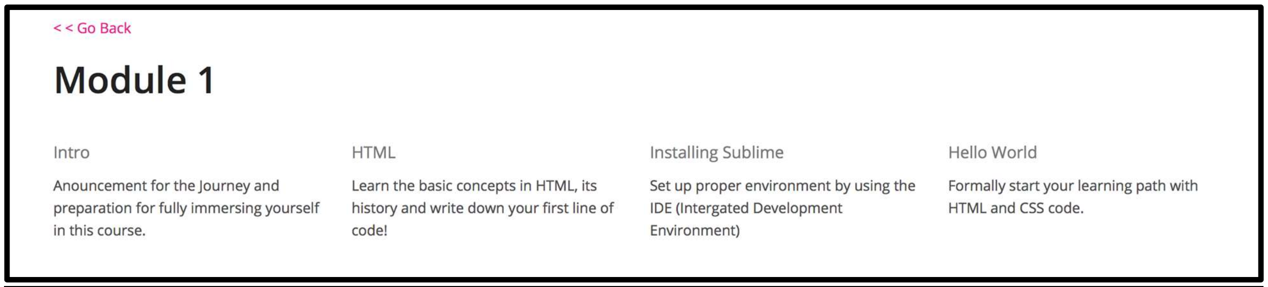
Learn the development workflow in the industry. Prepare for working in group and communicate with developers, designers.

Module 4 - Extend Your Knowledge into the Field

Take advantage of your knowledge in programming and extend your previous knowledge in other fields into overlapping areas, which would make your own career path more unique and individualized.

While the actual “nitty-gritty” aspects of the skills presented are similar in nature, the context on the **Career and non-Career track are different.** The career track focuses on how the syntax will be implemented and applied in later code, while the non career track focuses on how these skills will apply to a non-programming field.

Figure 10. Inside a Module (Contains Sections)



< < Go Back

Module 1

Intro

Anouncement for the Journey and preparation for fully immersing yourself in this course.

HTML

Learn the basic concepts in HTML, its history and write down your first line of code!

Installing Sublime

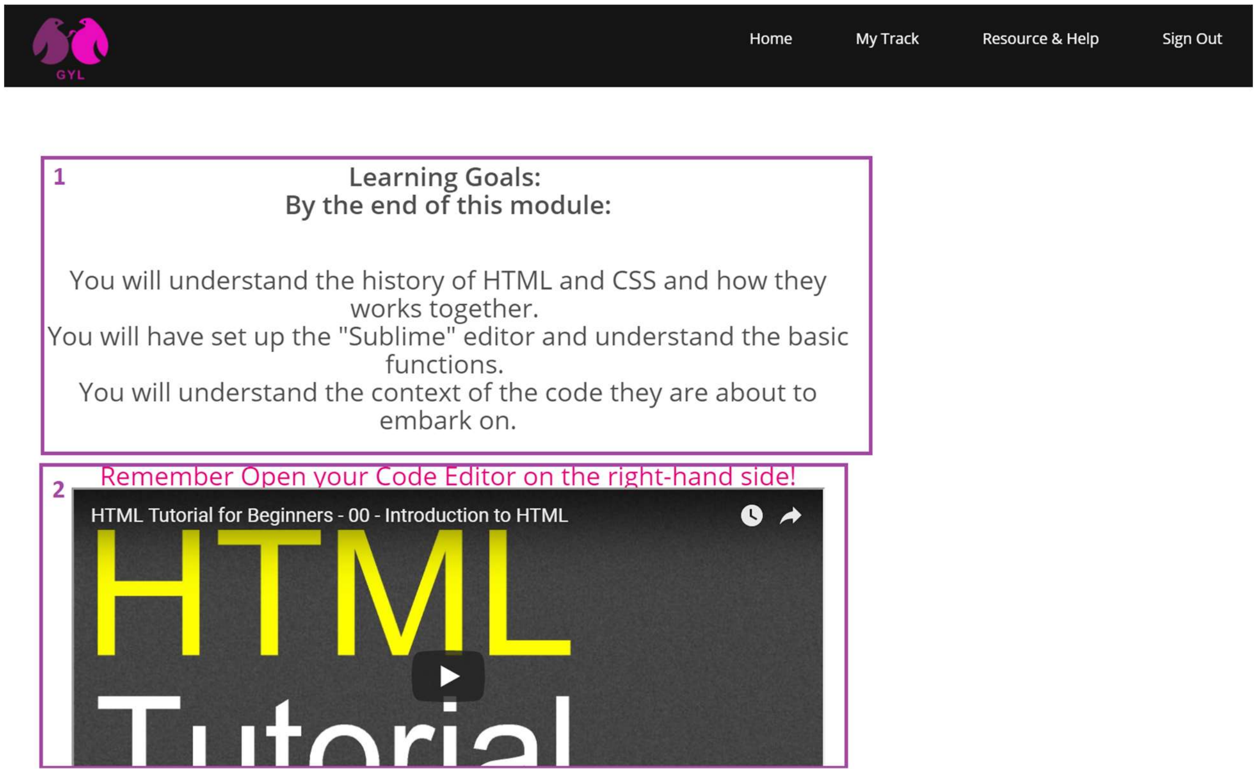
Set up proper environment by using the IDE (Intergated Development Environment)

Hello World

Formally start your learning path with HTML and CSS code.

Each module has further modularized sections as shown above. These learner selects each one section at a time in sequence to complete the current new module. At first, sections of the module are “unlocked” as the leaner proceeds through them. The modules remain open after “unlocking.” If learners want to revisit a previously completed module, there is no restriction of sequence as the learner is expected to have a clear idea of what they may want to revise.

Figure 11. Learning Interface (Inside the Section: Part 1)



Learning Goals:
By the end of this module:

You will understand the history of HTML and CSS and how they works together.
You will have set up the "Sublime" editor and understand the basic functions.
You will understand the context of the code they are about to embark on.

Remember Open your Code Editor on the right-hand side!

The Section Learning interface contains five main components.

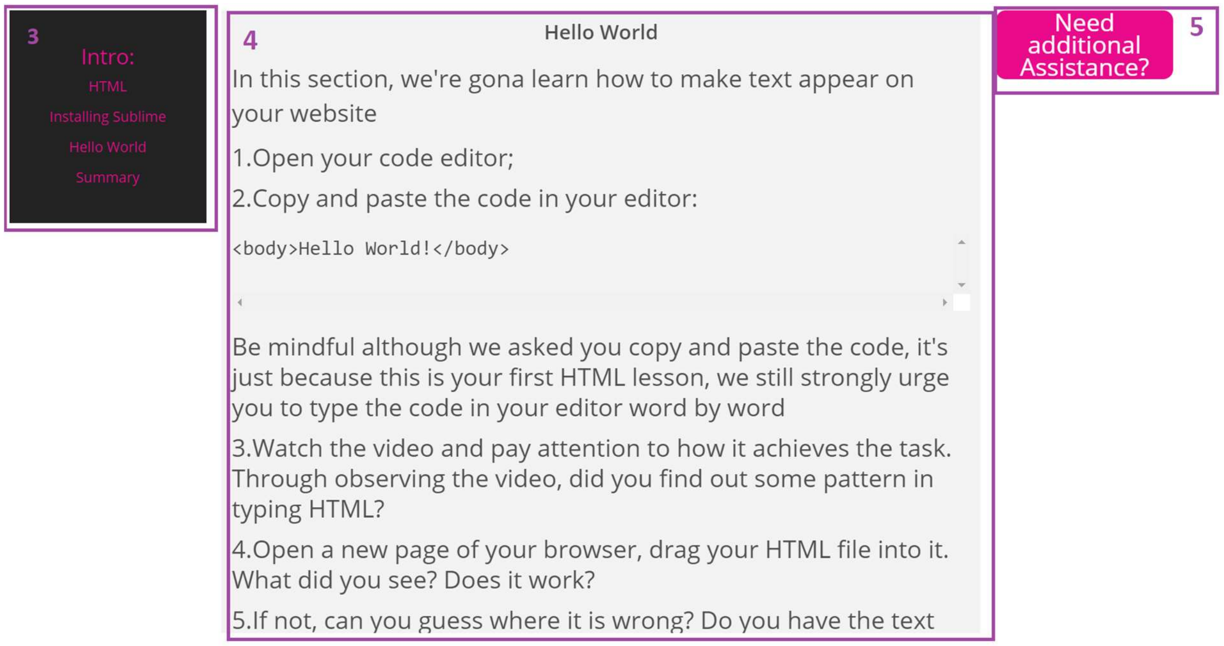
1: **Learning goal bar** where the learning goal of the current module is pinned to remind the learners what they're learning in this module. The words in pink remind the learners opening up their code editor and juxtapose it with the browser.

Note: The learning goals are majorly defined similarly for both tracks, in the Starter Unit. In the Intermediate and Advanced Unit, these goals diverge. The **career track goals** focus more on how to continue to develop and grow as a programmer, while the **non-career track goals** focuses on how understanding these skills and techniques will apply to a variety of fields.

2: **Code-Along Video** It is the component related to the topic that can be viewed on the right side of the web page.

The module learning goal is focused on the "big picture" while the section learning goal is more focused on the "nitty-gritty" of the skills the section focuses on.

Figure 12. Learning Interface (Inside the Section: Part 2).



Hello World

In this section, we're gona learn how to make text appear on your website

1.Open your code editor;

2.Copy and paste the code in your editor:

```
<body>Hello World!</body>
```

Be mindful although we asked you copy and paste the code, it's just because this is your first HTML lesson, we still strongly urge you to type the code in your editor word by word

3.Watch the video and pay attention to how it achieves the task. Through observing the video, did you find out some pattern in typing HTML?

4.Open a new page of your browser, drag your HTML file into it. What did you see? Does it work?

5.If not, can you guess where it is wrong? Do you have the text

3: **Section Table of Contents** It is the left sidebar which shows the table of contents included in the complete current module so that learners can jump from one section to another.

4: **Code Instructions** It is the main part in the middle where learners can view text instruction related to the current topic.

By using the text instruction with the Code along video, the GYL system helps the learner easily refer to the text and video about the content in the same place. These learning materials will supplement with each other more conveniently.

5: **Need Additional Assistance:** If the learner is having difficulties, they can click this button which will pop up additional helpful notes and tips that will help the learner. If they are still having issues, within this component is an "ask an expert" module, which opens up a Chat window with someone knowledgeable about the subject matter who can guide the learner along.

<u>Evaluation</u>

Learning Goal Evaluation

According to the learning goal specified through our GYL learning system, the evaluation will focus on if the learners have gained sufficient knowledge of CSS and HTML upon the completion of our program and are well-prepared to both efficiently and effectively collaborate with other programmers, designers, and product managers in the development team. To evaluate whether the learning goal has been fulfilled is to assess our learners' capability through reviewing their work. Besides, evaluating the appropriateness of the goal according to our curriculum and the learners reactions and feedbacks is also indispensable as it would be helpful to adjust the learning goal to better motivate the learners.

Given that the learning goals and objectives differ depending on the track, the evaluation of the accomplishment of learning goals differ as well. While there is a lot of overlap, there are distinct goals and objectives depending on the learners track, and obviously the learners ultimate goal beyond the structure of the course.

The **career track learner evaluation goals** focuses on how the learner will be able to progress in learning programming methods and languages beyond the course. Therefore, the goals for this track focus on how well the learner utilized external resources (ie: StackExchange, Github etc..), and how well they integrated them into their project.

The **non-career track learner evaluations goals** focuses on how well they understand how their knowledge will play a role in their future goals. As the different units/aspects evaluate learners are detailed below. (See Learning Goals for both Tracks)

Learning Environment Evaluation

Our evaluation of the system will be iterative. Given feedback from students, professor collaborators and industry partners, the system will continue to be updated based on that feedback.

Just like learning isn't a stagnant process, learning systems need to be able to grow and adapt as well. To that end, GYL Learning will continue to grow based on feedback from our collaborators, students, and community as well as advances in the nature of programming and web design. The efficacy of the learning system will be assessed on multiple different fronts, mainly through system usage data, "Ask an Expert" usage, and user provided data via survey.

System Usage Data

The system will automatically locate basic errors for opening and closing of any section, based on the algorithm. For example, if the learner opens <div> section, but forgets to close it(</div>), the algorithm will point out an error to close the section. The GYL system automatically collects errors that occur when the learner runs their program. This data is collected and analyzed by section, by module and by unit.

The data is used to analyze which aspects of the learning system need to be adjusted, and overtime this data can be highly granular and customized for an adaptive learning environment, customized to the individual learner.

For example, let's say the data shows that while implementation of a certain method is done correctly immediately, the learner slowly shows decrease in using/understanding a certain method. We would adapt the GYL system to include "refresher" sections as needed. This intervention can be implemented system wide, or if learners in general are not struggling with that concept- only if they show repeated issues with a particular concept or method.

There is also a "**Need additional Assistance**" button that exists in all of the modules, it provides additional context, things to bear in mind and general additional assistance for the learners. Depending on how often it's clicked, and what the errors are before and after the learner asks for additional assistance, the content in the module itself may be adapted (certain instructions may be emphasized or changed entirely), or the help text may be edited as well.

If the learner utilizes the "**Ask an Expert**" Chat button, the expert submits usage data as to what the learner needed help with. This data will be used to improve the content in the "Need Additional Assistance" tabs, to enhance and improve the learning throughout the system.

Survey of User Experiences

At the end of every module, the learners are asked to reflect in the "summary" section about what they've learned and if they've struggled with anything in particular. This information will be used to clarify the sections and modules to make it clear to learners.

Learner Evaluation

Peer-based Evaluation

The learner evaluation will be a two fold methodology. As our learning goals include the ability to grow and collaborate on others code, part of the evaluation process does as well.

Later modules include "peer-based evaluation" where a code is critiqued and assessed by another member of the class. The learners then have the opportunity to improve on their original design based on the critique of their classmates.

Learner Evaluation of Cumulative Project by Expert

Our evaluation method is not based on simple memorization of concepts and material, replicated exactly as it exists in the learning material. The GYL system embraces the "messiness" of the real world environment- and that's exactly how we model our evaluation.

Therefore the "Cumulative Project" that the learners spend the Intermediate and Advanced Unit developing, will be evaluated by partner experts in the field. Before it gets to this stage, the project will have been iteratively given feedback on more basic concepts through the system automatic feedback, and the peer evaluation aspects.

Module Evaluation

This is a "Project-based Learning" design. So evaluation of the modules goes on evaluating how the project currently stands up against the learning goal of the section and module. For example, if the module was about proper formatting of heading and body sections (a fairly early module), the evaluation will be as follows:

- Did the project reflect proper head and body tags
- Were any sections mislabeled or formatted
- Was learning from previous modules incorporated.

Early modules will be checked by the IDE/GYL system itself. Since these modules focus on syntax based examples, the IDE system can check both for "RunTime" errors (errors that prevent the program from running) and some style errors (errors that won't prevent the page from loading, but may cause other issues (ie: take an excessive loading time).

Later modules will be much more "loosely-structured" where the exact parameters are up to the students themselves. Thus while certain "style" aspects can be automatically checked by the IDE system itself, the more advanced aspects will be evaluated by their peers.

Peers will be matched automatically with each other, and asked to evaluate the quality of code of the learning and make comments as to how it can be improved.

General errors will be caught by the IDE before it gets to this stage, allowing the learners to focus on the higher level analysis required. This stage of evaluation is saved until the learners pass the first few modules. This ensures that they have the sufficient skills and prior knowledge to effectively evaluate their peers code.

This Project based learning design takes into account not only what the learning goal/task was in the particular module, but how well the module learning goal/task was incorporated into the project. Thus the evaluations is based on the final product delivered through project work by peers.

Collaborative Aspect

Learners not only build on their own projects, but fork off a peer's code and improve on it as well. Different goals for the collaborative section include:

- Incorporating input from a previous page into a new page (ie: a blog post).
- Fixing errors and mistakes in the peers code
- Taking the "basic" code the peer developed and adding additional features, aesthetics and options.

Notice the “ill-structured” nature of this assessment. We’re not telling the learners exactly what to do, rather, we encourage learners to explore external resources (that we support) to gather ideas, and figure out new exciting methods in web design.

<u>Assessment of the collaborative work:</u>

Since we do not tell the learners exactly what to do, the assessment criteria are more loosely structured which includes:

- Did the learner meaningfully improve on the original project?
- Did the learner fix errors and mistakes in the original project?
- Did the learner incorporate previous module’s lessons as well as research outside material for ideas?

Overall Project Assessment

<u>Website Functionality /Learning Goals</u>

Since this is Project Based Learning, the overall project assessment incorporates all of the previous modules. The goal of this evaluation is not if it matched arbitrary design parameters, but how they will function as working in design in the future. The below goals both assess the accomplishments of the functionalities of the learning system, as well as how well the website functions as a website (in terms of both functionality and aesthetic appeal).

It will tie together all that’s learned in all the modules and be a fully dynamic aesthetically pleasing, functional website that can be proudly displayed on one's portfolio.

- Website incorporates material from most, if not all modules and sections.
- Website has interesting, dynamic content on all pages.
- Website has at least one “user generated” page, where content inputted by the user of the website is utilized and displayed (ie: a blogpost)
- Website integrates style guides in the entirety of the website.
- Website has at least 5 pages/modules.
- Website works well on computer, mobile, and tablet computers.
- Website demonstrates at least one concept or method that was **not** taught in class

<u>Expert Assessment</u>

Given that we want our learners to understand the nuances and intricacies of applying their knowledge in the “real-world”, the cumulative project will be evaluated by an expert in web-design. Rather than looking for any specific aspect or functionality, they will be evaluating and providing feedback on the website as a whole. For example, the expert will be checking “is the purpose of the site easily apparent” and “is it easy to navigate.”

The learner will then be directed to the modules that address the experts concerns. (For example, issues with text being to small, will direct the user to the “how to change text size” module as well as the “Style Guides” module for formatting particular text.

This will give our learners the knowledge they need to “fix” or modifying any parts of the website so that it is in line with what the industry expects of designers and programmers in the field, so they know and can showcase their cumulative project on their portfolio.

Conclusion

Programming and understanding the way it works is becoming more and more essential for thriving in the modern world. Many are exploring this field from a different perspective. As the industry moves towards a more collaborative mentality.

Our system fills an need, that trains novice programmers both in the methods of web-design, as well as the skill necessary to continue to grow as a developer.

References

Akhriza, T. M., Ma, Y., & Li, J. (2017). Revealing the Gap Between Skills of Students and the Evolving Skills Required by the Industry of Information and Communication Technology. *International Journal of Software Engineering and Knowledge Engineering, 27*(05), 675-698. doi:10.1142/s0218194017500255

Ala-Mutka, K. M. (2005). A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education, 15*(2), 83-102. doi:10.1080/08993400500150747

Ben-Ari, M. (1998). Constructivism in computer science education. *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education - SIGCSE '98*. doi:10.1145/273133.274308

Ben-Ari, M. (2004). Situated Learning in Computer Science Education. *Computer Science Education, 14*(2), 85-100. doi:10.1080/08993400412331363823

Conradie, P. W. (2014). Supporting Self-Directed Learning by Connectivism and Personal Learning Environments. *International Journal of Information and Education Technology, 4*(3), 254-259. doi:10.7763/ijiet.2014.v4.408

Erhel, S., & Jamet, E. (2013). Digital game-based learning: Impact of instructions and feedback on motivation and learning effectiveness. *Computers & Education, 67*, 156-167. doi:10.1016/j.compedu.2013.02.019

Giannakos, M. N., Pappas, I. O., Jaccheri, L., & Sampson, D. G. (2016). Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. *Education and Information Technologies, 22*(5), 2365-2382. doi:10.1007/s10639-016-9538-1

Kadijevich, D. M., Angeli, C., & Schulte, C. (2013). *Improving computer science education*. New York: Routledge.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education, 10*(2), 109-128. doi:10.1076/0899-3408(200008)10:2;1-c;ft109

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice, 41*(4), 212-218. doi:10.1207/s15430421tip4104_2

Latchem, C. (2016). Learning Technology and Lifelong Informal, Self-directed, and Non-formal Learning. *The Wiley Handbook of Learning Technology,* 180-199. doi:10.1002/9781118736494.ch11

Marino, V. (2017, September 22). Some People Learn to Code in Their 60s, 70s or 80s [Editorial]. *NY Times*. Retrieved from https://www.nytimes.com/2017/09/22/your-money/some-people-learn-to-code-in-their-60s-70s-or-80s.html

Mayer, R. E. (2003). The promise of multimedia learning: Using the same instructional design methods across different media. *Learning and Instruction, 13*(2), 125-139. doi:10.1016/s0959-4752(02)00016-6

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239-264. doi:10.1080/08993408.2013.832022

Muller, C. L., & Kidd, C. (2014). Debugging geographers: Teaching programming to non-computer scientists. *Journal of Geography in Higher Education, 38*(2), 175-192. doi:10.1080/03098265.2014.908275

Nola, R. (1998). Constructivism in Science and Science Education: A Philosophical Critique. *Constructivism in Science Education,* 31-59. doi:10.1007/978-94-011-5032-3_3

Pozenel, M., Furst, L., & Mahnicc, V. (2015). Introduction of the automated assessment of homework assignments in a university-level programming course. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. doi:10.1109/mipro.2015.7160373

Reiser, R. A., & Dempsey, J. V. (2017). *Trends and issues in instructional design and technology*. Boston: Pearson Education.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172. doi:10.1076/csed.13.2.137.14200

Soloway, E., & Spohrer, J. C. (2009). *Studying the novice programmer*. New York: Psychology Press.

Why We Need Self-Directed Learners. (n.d.). *Assessment Strategies for Self-Directed Learning,* 1-17. doi:10.4135/9781483328782.n1

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education, 12*(3), 197-212. doi:10.1076/csed.12.3.197.8618

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course. *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education - SIGCSE '01*. doi:10.1145/364447.364581

Wilson, B. C. (2002). A Study of Factors Promoting Success in Computer Science Including Gender Differences. *Computer Science Education, 12*(1-2), 141-164. doi:10.1076/csed.12.1.141.8211

Zilan, W., Jing, L., & Pan, X. (2011). How to improving the non-computer science students' computer ability. *2011 6th International Conference on Computer Science & Education (ICCSE)*. doi:10.1109/iccse.2011.6028753