6-1-2015

Code girl

Tracey Acosta
Santa Clara University

Amanda Holl
Santa Clara University

Paige Rogalski
Santa Clara University

SANTA CLARA UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING
WEB DESIGN AND ENGINEERING

Date: June 1, 2015

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Tracey Acosta
Amanda Holl
Paige Rogalski

ENTITLED

Code Girl

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

BACHELOR OF SCIENCE COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING

Dan C.

Thesis Advisor

Thesis Reader

Department Chair

Code Girl

by

Tracey Acosta
Amanda Holl
Paige Rogalski

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 1, 2015

Code Girl

Tracey Acosta
Amanda Holl
Paige Rogalski

Computer Science and Engineering
Web Design and Engineering
Santa Clara University
June 1, 2015

ABSTRACT

Despite the growing importance of technology and computing, fewer than 1% of women in college today choose to major in computer science.[1] Educational programs and games created to interest girls in computing, such as *Girls Who Code* and *Made With Code*, have been successful in engaging girls with interactive and creative learning environments, but they are too advanced for young girls to benefit from. To address the lack of educational, computer science games designed specifically for young girls, we developed a web-based application called Code Girl for girls age five to eight to customize their own avatar using Blockly, an open-source visual coding editor developed by Google. Girls learn basic computer science and problem-solving skills by successfully using puzzle-piece like blocks to complete challenges that unlock new accessories for their avatar.

In conducting user testing with a Girl Scouts ages six to eight, we assessed the complexity of the application and identified ways make Code Girl more user-friendly and intuitive. The overall feedback we received on Code Girl in user testing was positive, as a majority of the girls expressed an interest in playing the game again and playing more games designed to teach programming. Code Girl thus appeals to the general pastimes of young girls to interest them in computer science from an early age and hopefully inspires them to pursue computing as a career. Before being released to the public, a few improvements are necessary. The application must be made fully responsive, the page load time when deployed must be reduced, and additional challenges and accessories for the avatar should be incorporated, all of which will better reach and engage users in learning about computing, thereby educating and empowering them even more.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Computing continues to grow in importance, yet few girls pursue this field. Today, less than 1% of women in college are majoring in computer science, making computing a heavily male-dominated field.[1] Girls have the potential to contribute immense creativity and ingenuity to the future of computing and technology, but few of them are introduced to opportunities to get involved. Exposing girls to programming at a young age fosters their interest in the subject, so that when they are older they may consider pursuing a career in computing.

In an attempt to address this problem, many educational programs designed to introduce girls to computing have been implemented in schools and camps across the country. Non-profit organizations such as *Girls Who Code* hold summer coding programs that educate high school girls about computer science and the skills they need to pursue computing. In addition, programs like *Scratch* have been developed to begin teaching children aged eight and older how to code through games. *Scratch* and similar programs such as *Alice* have children use a visual coding editor to solve puzzles and create games, improving their ability to reason through problems and design projects. *Made With Code* recently launched a series of *Scratch*-like applications aimed at engaging girls in coding by allowing them to customize a picture of themselves, draw an avatar, design a bracelet and more with code. These interactive and customizable coding environments pique the interests of children by giving them control and the ability to create real products.

Although educational programs, such as those organized by *Girls Who Code*, and visual-based programs like *Scratch* and *Alice* interest children and teenagers in computing, they are not enough. Educational programs are usually designed for children in middle school and up, and applications like those created by *Made With Code* are geared toward girls eight and older. Few programs exist for children younger than eight and those that do, such as *Tynker* games, are either aimed at boys or are designed to be unisex, which make them uninteresting to girls. As such, girls have less exposure to coding from an early age and become interested in other areas, perpetuating the lack of women in computer science and related fields.

Our solution, *Code Girl*, introduces girls ages five to eight to computing through a education computer-science game designed to best appeal to their general interests while simultaneously teaching them coding. Many young girls love to play with dolls and spend hours dressing their dolls up, so we created a web-based application that allows girls to customize their own avatar using a visual coding editor. To unlock features and accessories for their avatar, the girls have to complete various challenges and puzzles designed to develop their ability to solve problems using code. For example, before a girl is able to put shoes on her avatar, she has to successfully move a object through a maze by putting together blocks of code. We included a story-telling aspect in our application by making the avatar a superhero named Grace, whom the user then dresses up to assume any identity she chooses. Grace also has a robot sidekick named Ada who is incorporated into many of the challenges, giving the application a cohesive superhero theme and story that will engage users. To allow users to progress through the application at their own pace, girls can create an account, with their parents permission, so they can save their progress in designing their avatar at any point in the game. When they are finished creating their avatar, the girls are able to save their avatar as an image, which they can then print them out to share with their friends, hopefully inspiring others to learn more about Code Girl and consequently computer-science. Given the young age of our users, we designed our application to be simple, colorful, easy to read, and easy to interact with, making it as user-friendly and engaging as possible.

We assessed the success of our project by partnering with Girl Scouts and analyzing how interested the girls were in learning more about programming after using our application. By designing an application specific to the interests of five to eight year old girls, we hope to interest this underrepresented group in coding from a young age and inspire them to pursue computing later on.

1.2 Background

As the initiative to introduce young children to programming grows, so do the number of available tools. Before designing Code Girl, we investigated a few of the previously mentioned platforms and applications, along with other popular online games for children, to understand and assess what they do and do not offer.

The first application, *ScratchJr*, is an iPad App where children ages five to seven can program their own interactive stories and games by dragging and dropping different puzzle-piece shaped blocks. Although *ScratchJr* has the customization options for the age group we are looking for, it does not specifically target young girls and limits the users to iPads. This will not work for our project because we want young girls to feel comfortable using a computer, not an iPad, through our application.

Code.orgs Studio is another platform that teaches basic computer skills through games and puzzles designed for ages four to six plus. Course 1 and Course 2, the twenty hour courses geared towards our target age group, are specifically designed to be taught in classrooms and include full lesson plans for the teacher. As a specific curriculum, *Code.orgs Studio* is designed to be unisex in an attempt to appeal to an entire classroom. Our project hopes to captivate girls and motivate them to play and learn outside of the classroom.

The hardware and software combination of *Leapfrog* is well known for its educational games that are designed to begin teaching kids math, problem solving, language and more. An appealing attribute of *Leapfrog* is the use of popular cartoon characters in its games and its diverse selection of products and games for a range of ages. However, *Leapfrog* currently only has one game to teach computer programming and the characters are vikings, which appeal more to boys. The age group for *Leapfrog* games aligns with our target audience, but the content is not geared towards teaching basic computer engineering principles.

MadeWithCode is an initiative by Google to get more girls in middle school and above excited about coding. On their website, they have a few beginner projects that use Blockly, a programming language made up of puzzle-piece shapes, to do tasks such as create an avatar using shapes, connect together virtual instruments to create a song, decorate a picture, or animate your own gif. Since *MadeWithCode* is one of the few platforms geared towards girls, we hoped to use some of their design ideas to appeal to a younger audience such as the colors and the simplicity of using Blockly to piece things together.

Another application, *Polly Pockets* online games, is a series of games featuring the main character and her friends in various locations and situations. Since she is a popular doll, with her own elaborate and engaging online world, we wanted to use the idea of customization of a character. The downside of *Polly Pockets* games though, is that there is too much of a focus on activities such as shopping and less focus on educational enrichment, which is the main objective of our project.

These platforms all have something to offer in their own way, yet none solve the problem of specifically engaging girls ages five to eight in basic computer and problem solving skills. We thus created our requirements by taking the advantages of each of these platforms and combining them into a unique solution.

1.3 Requirements

For our solution, we identified and met the following functional and non-functional requirements, which describe the necessities of our application. The set of functional requirements define what must be done by the system, and the set of nonfunctional requirements define the manner in which the functional requirements need to be achieved. In addition, we identified and met design constraints, which are similar to non-functional requirements, but they limit the way the solution is designed and implemented.

1.3.1 Functional Requirements

The functional requirements for our solution are summarized below. Our application is interactive, as users snap together blocks of code to customize an avatar and immediately see the results of their actions, in addition to receiving dynamic instructions on how to play the game. Users also connect blocks of code to solve challenges, which upon successful completion unlock new features for the users avatar. The avatar creation and challenges teach users basic logic and computer science skills, such as iteration, loops, and conditional statements. To allow our users to progress through the application at their own pace, girls can create an account, with their parents permission, so that they can save their progress in designing their avatar and completing challenges and return to the application later. When they are finished customizing their avatar, the girls can save their avatar as an image, which they can then share with their friends and inspire others.

1. The application is interactive
2. The user snaps together puzzle pieces to customize an avatar
3. The avatar created by the user is saveable
4. The application provides instructions
5. The user solves challenges by snapping together puzzle pieces
6. The user unlocks new accessories for her avatar by successfully completing challenges
7. The application saves user progress
8. The application teaches basic logic skills including positioning, loops and sequences

1.3.2 Non-functional Requirements

In addition to functional requirements, we outlined and achieved the following non-functional requirements, summarized below. Given the young age of our users, we designed our application to be colorful, easy to read and easy to interact with, making it as user-friendly, engaging, and intuitive as possible. Our application is also compatible with desktops, laptops, and tablets, making it competitive with applications currently available.

1. The application is user-friendly
2. The application is intuitive
3. The application is compatible with desktops, laptops and tablets

1.3.3 Design Constraints

With regards to design constraints, summarized below, we identified and met only two. First, our application is web-based because of senior design requirements for the web design and engineering major, but this has the advantage of making our application widely available to potential users. Most importantly though, given the young age of our users, our application is compliant with the Children's Online Privacy Protection Act (COPPA), which constrained the manner in which we gathered and stored user information.

1. The application is web-based
2. The application is compliant with the Children's Online Privacy Protection Act

1.4 Conclusions

Through research into existing applications, we were able to take the design and learning outcomes we thought were most beneficial for engaging and educating young girls and craft them into requirements for *Code Girl*. The functional requirements include the main activities the user is able to do, as well as what we want to teach young girls through playing with our application. These requirements and ideas then gave way to designs of how these activities fit together and how the user might interact with the system.

Chapter 2

Design

2.1 Conceptual Model

Before implementing our application, we created conceptual models, in the form of state diagrams, activity diagrams, and mockups, to help us determine and visualize the flow of system and the design of the application.

2.1.1 State Diagram

From the functional requirements, we determined the main states of the system and how they interact with each other. The state diagram, Figure 2.1, shows the three states and how the system transitions from one state to another. At any given time, the user can only be in one of these states. For example, while a user is playing a challenge, she cannot simultaneously edit her avatar. The user must successfully complete a challenge in order to unlock a new challenge and accessories. From there she can either play the new challenge or work on building her avatar.

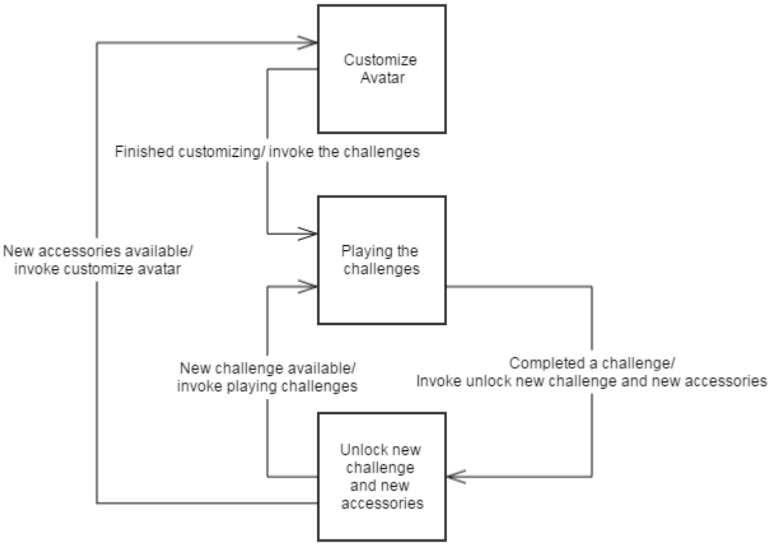

Customize Avatar
Finished customizing/ invoke the challenges
New accessories available/ invoke customize avatar
Playing the challenges
New challenge available/ invoke playing challenges
Completed a challenge/ Invoke unlock new challenge and new accessories
Unlock new challenge and new accessories

Figure 2.1: A state diagram that shows the three main states of the system.

2.1.2 Activity Diagrams

Each state is made up of many actions. An activity diagram combines these actions into main activities done by the user. The first activity diagram, Figure 2.2, focuses on showing the overall activities of a user of our system. Once a user accesses the system by signing up or signing in, she is taken to the main avatar customization page, where she can build her avatar. On this page, there is a progress bar indicating if the user has challenges available to play. If she does, she may successfully complete them to unlock accessories for her avatar. At any given point, the user can save her progress and continue playing or log out.

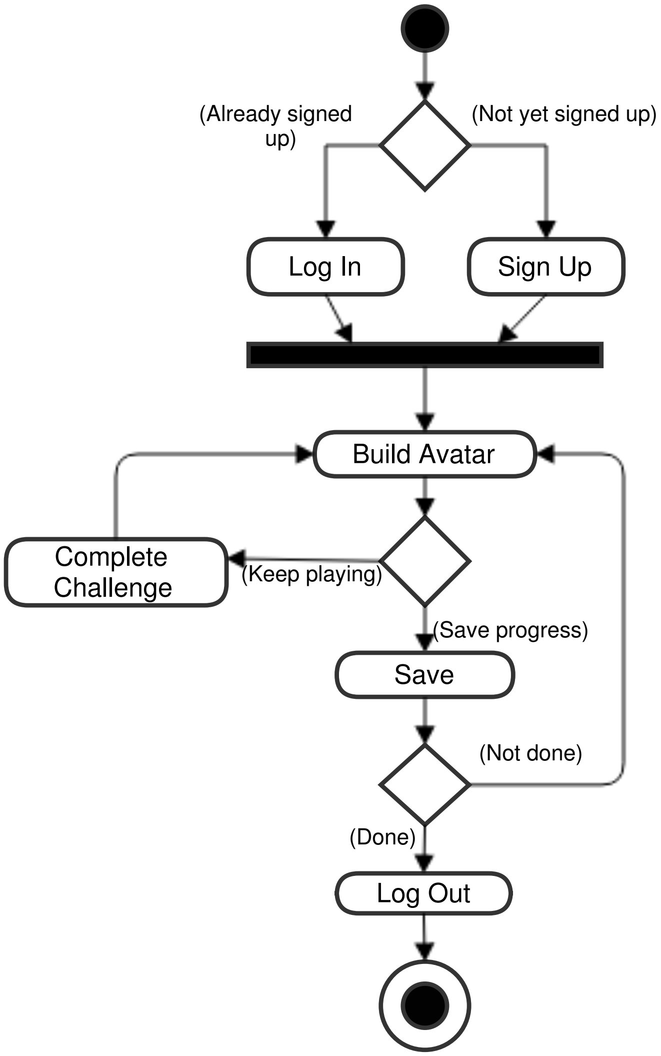


Figure 2.2: The activity diagram of the general application.

While the general activity diagram of the application gives a high-level view of the activities, it does not go into detail about how the user builds an avatar or completes challenges.

Figure 2.3 shows the actions a user takes to create an avatar. These actions focus on the user interaction with puzzle-piece shaped blocks. As the user selects, drags and drops, and arranges these blocks on the canvas she can see the customization of her avatar taking place. If the user wants to try and unlock new accessories, she can choose to play a challenge; otherwise, she can continue to edit her avatar.

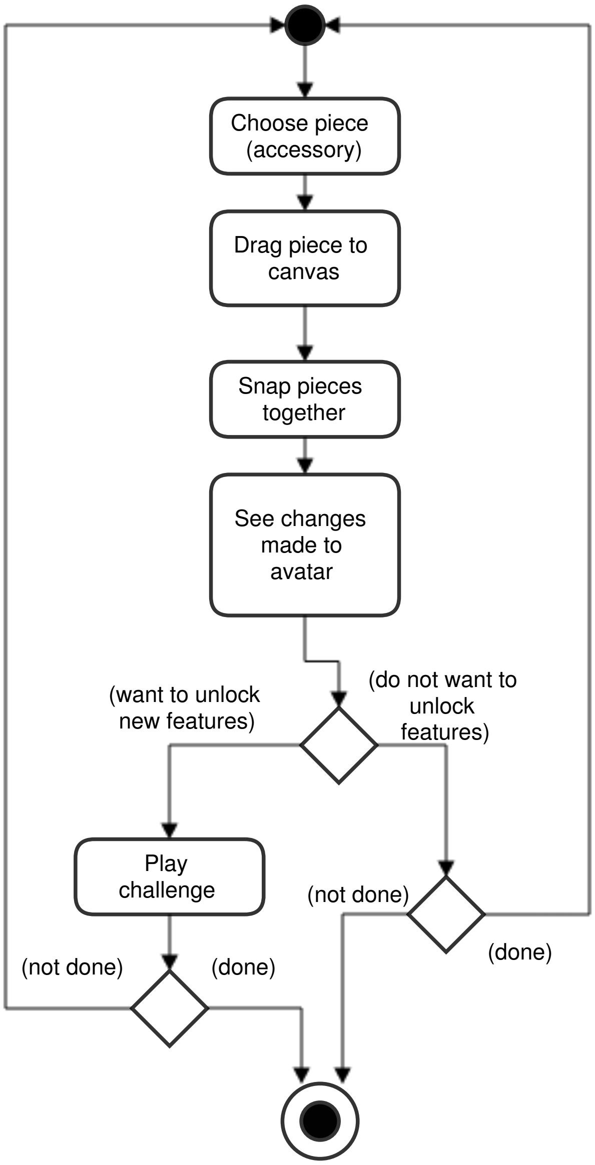


Figure 2.3: The activity diagram of building an avatar.

Figure 2.4 depicts the flow of activities of a user completing a challenge. In this example challenge, the user arranges puzzle-piece shapes to direct their character through a maze. The puzzle pieces may tell the character to turn right, or take two steps. If the user successfully completes the challenge, she unlocks new accessories for her avatar.

2.1.3 Mockups

Low fidelity mockups provided us with a way to show the user-interface design to visualize how the user will accomplish the main activities in our application and gave us a framework to begin

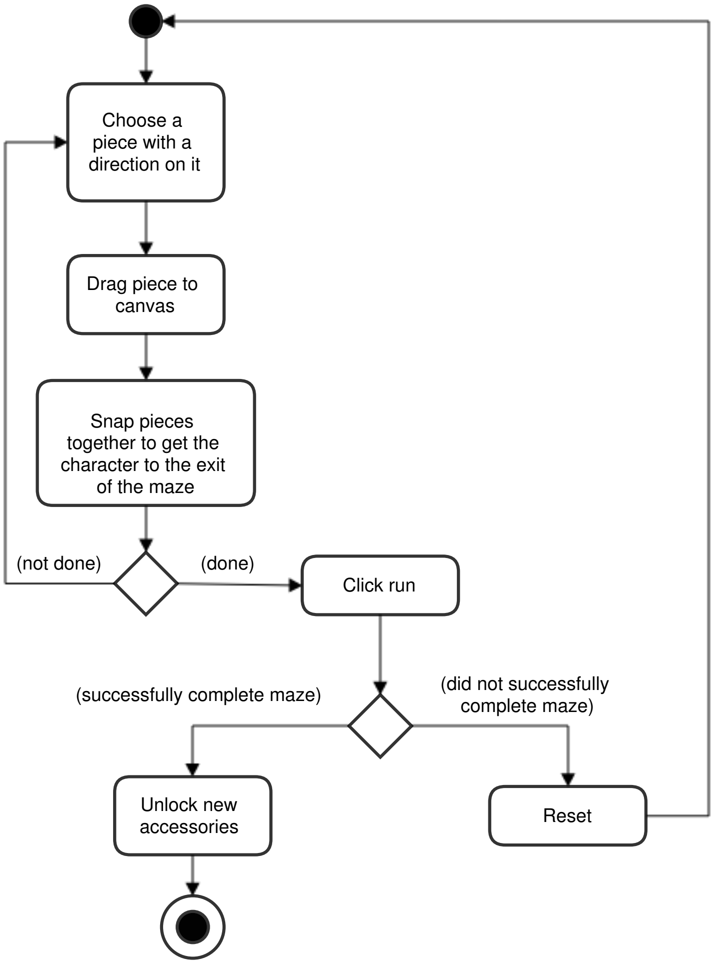

Choose a piece with a direction on it
Drag piece to canvas
Snap pieces together to get the character to the exit of the maze
(not done)
(done)
Click run
(successfully complete maze)
(did not successfully complete maze)
Unlock new accessories
Reset

Figure 2.4: The activity diagram of completing a challenge.

designing and implementing the game. Figure 2.5 shows an initial idea for the design of the login or signup page. We decided to combine these actions into a single page since they relate to the user's account. It is also important to note that because our target audience consists of minors, we ask for the parent's email to make sure the parent gives permission to join, thereby making our application compliant with the Children's Online Privacy Protection act, as specified in our design constraints.

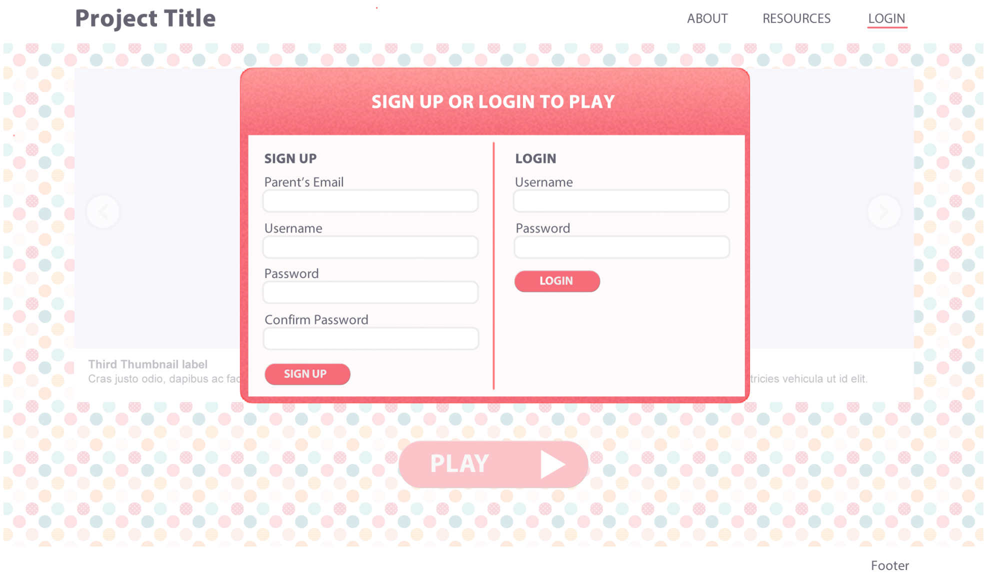


Figure 2.5: A mockup of the signup or login page.

The mockup of the main page of *Code Girl*, where the user will create the avatar, is shown in Figure 2.6. The main page is divided into three sections. To the left is a library of available blocks that the user can use to create the avatar. To the right of the library is the canvas where the user snaps the pieces together. On the far right is where the avatar is shown, so the user can see the changes made as she customizes her avatar.

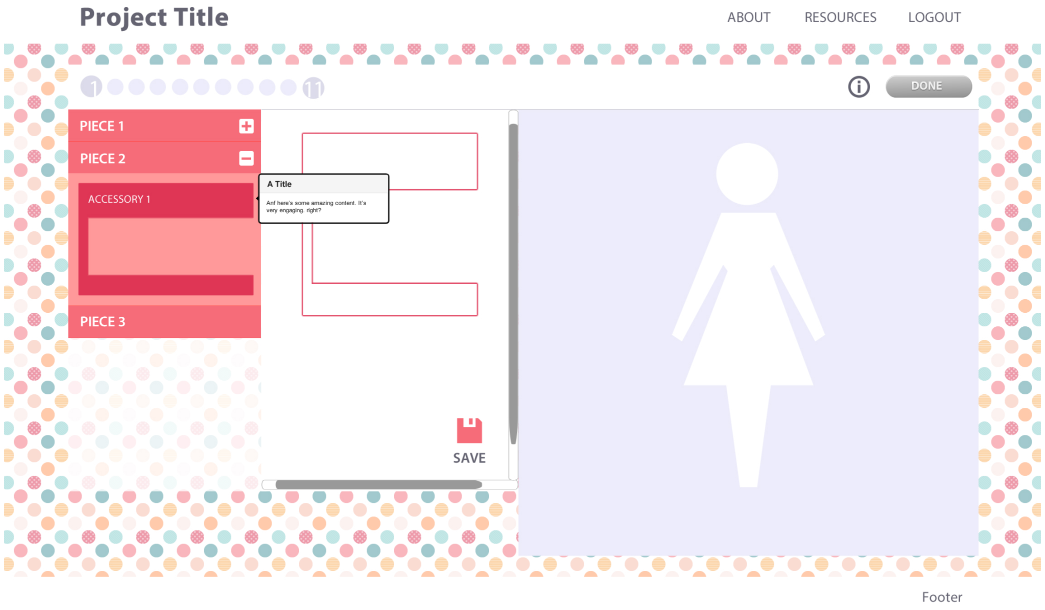


Figure 2.6: A mockup of the main page.

2.2 Use Cases

A use case defines the steps required to accomplish a specific goal, as described in the activity diagrams and visualized in the mockups. The following use cases describe how the user interacts with the system to achieve these goals, including conditions that need to be met, steps required, and common errors that might occur. The use case diagram, Figure 2.7, helps to illustrate the major actions the user will take when using our system.

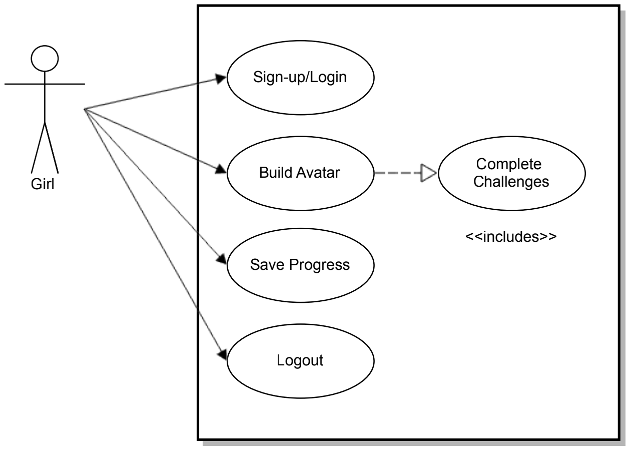


Figure 2.7: The use case diagram.

1. Signup/Login

    - Goal: Sign-up or login to begin building the avatar
    - Actor: Girl (with Parent)
    - Precondition: The user does not have an account if he or she wants to sign-up and the user has an account if he or she wants to login in
    - Postcondition: The user is logged into the system
    - Scenario:

        (a) The user either selects the SIGN UP or LOGIN button

        (b) The user either fills out the form to sign up or the form to login

    - Exceptions:

        (a) The user wants to sign up, but his or her email is already in the system

        (b) The user forgets his or her password to login

2. Build Avatar

   - Goal: A customized avatar/doll has been created
   - Actor: Girl
   - Precondition: The girl has unlocked all of the features and built her customized avatar
   - Postcondition: The girl has built her own avatar using blocks of code
   - Scenario:

     (a) The user chooses a piece that she wants to add to her avatar

     (b) The user drags the piece to the canvas

     (c) The user snaps the piece together with the other pieces on the canvas

     (d) The user sees the changes made to the avatar

     (e) The user plays challenges to unlock more pieces
   - Exceptions:

     (a) The user wants to skip a challenge

3. Save Progress

   - Goal: The user saves the blocks she has put together to build her avatar
   - Actor: Girl
   - Precondition: The girl has at least one block on the canvas for building her avatar
   - Postcondition: The girl's progress has been saved
   - Scenario:

     (a) The girl clicks the "SAVE" icon
   - Exceptions:

     (a) None

4. Logout

   - Goal: The user is logged out of the application
   - Actor: Girl
   - Precondition: The girl is logged in
   - Postcondition: The girl is logged out

- Scenario:
  (a) The girl clicks on the LOGOUT button
  (b) The girl is prompted to save her work
  (c) The girl confirms she wants to logout
- Exceptions:
  (a) None

2.3 Architectural Design

The activity diagrams, mockups, and use cases all present how the user interacts with the system. An architectural diagram shows the components and connections of the system as a whole. Figure 2.8 displays a multi-tier architectural style that models our system. While *Blockly* is client-side, we still need a server and database for logic and storage.

The system is contained in four main modules to successfully independently manage different components. The first module contains the custom blocks used to create the avatar. The second module contains the challenges the user can play. Isolating the modules helped us with development and testing that the challenges and avatar worked separately before we integrated them. The logic module on the client side defines the integration between unlocking challenges and new blocks. The server contains the fourth module that defines the logic for saving the user's level and blocks, so that when she logs out and logs back in, the system remembers where she left off. Through hosting our application on Google App Engine we have access to Google Cloud Storage to save and load the program, as well as keep track of user accounts and authenticate our users.

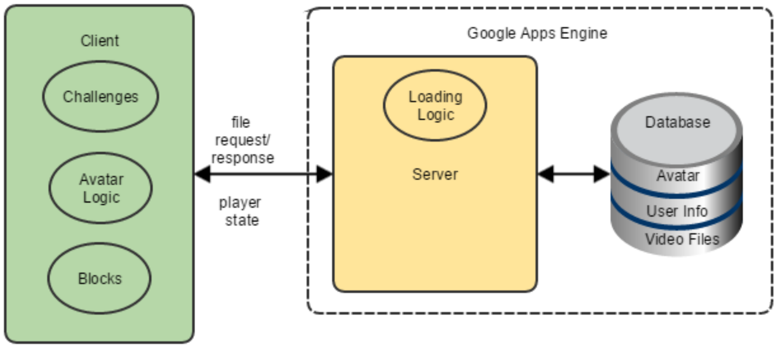


Figure 2.8: An architectural diagram of our system.

2.4 Technologies Used

To accomplish our design and achieve the desired functionalities we have previously described, we used the following web-development technologies and services:

- Blockly: A client-side library that can be used to build programs by snapping together puzzle-piece blocks in a visual editor created by Google. Blockly provides us with the framework for a drag and drop environment and the library of coding puzzle piece blocks used in the games.
- JavaScript: A programming language that enables client-side interaction with the user, browser control, and changes to the content displayed on the webpage.
- JQuery: A JavaScript library that implements common JavaScript functionality to simplify client-side scripting.
- XML: A markup language that specifies how a document is encoded.
- HTML5: The markup language used to render the graphics on the canvas.
- Google Apps Engine: A Platform as a Service that enables developers to build and deploy an application using Googles runtime and development environment. Google Apps Engine provides many features, such as data storage, retrieval, and search.
- NoSQL: A database that enables the storage and retrieval of data, such as account information, but does not use tabular relations
- Python: A scripting language that we used to communicate between the Google App Engine and the database.
- GitHub: a web-based repository service used for revision control and source code management.

2.5 Design Rationale

Before making any decisions regarding the design and implementation of our application, we thoroughly researched how to effectively design games to teach computer science and tested successful applications, such as *Scratch* and *Made with Code*. Our design was thus driven by our goal of educating young girls in a fun and interactive way to inspire them to pursue computer science or technology. To achieve this goal and the requirements we identified, we made several design choices, presented and discussed in this section.

2.5.1 Game Design

We decided to create a game-based application to inspire young girls to pursue computing because game-based learning through applications such as *Scratch* and *Alice* has been proven effective in teaching children computer science concepts and skills.[3] In our application, users snap together pieces of code to create their own avatar and complete challenges that unlock more features for their avatar within the context of a larger story. We chose these activities because research shows that educational games are more effective when they actively engage the users with stories and interactive puzzles. Additionally, in researching non-educational games, such as *Polly Pockets*, that are popular among our target audience of five to eight year old girls, we noticed that many games allowed users to customize their character, indicating that personalization engages young girls. Incorporating challenges that unlock new features will not only engage our users by allowing them to customize their avatar, but will also serve as an incentive for our users to continue playing our game, and thereby learn computer science concepts and skills. By introducing young girls to computer science in a fun and interactive game that is similar to the games they already play, we hope to teach them basic skills and concepts and inspire them to learn more about computing.

2.5.2 Visual Programming Environment

Traditional programming languages and environments are the most effective means of teaching computer science concepts and skills, but they are too complicated for young children to understand. They also are usually taught in a classroom setting or online environment, and difficult for people to learn on their own. Since our target audience is very young and presumably using our application on their own, we chose to use a visual programming environment to best meet their needs and skill level. Visual programming environments are easier to learn in than traditional, language-driven environments, because they not require the users to compile their code or verify syntax. They essentially allow users to write error-free programs without any instruction by abstracting away the syntax of the code, which will benefit our users since their reading, writing, and reasoning skills are still developing. By removing the issues of syntax and coding errors, visual programming environments focus the users attention on the logic of the problem, introducing them to basic computer science concepts, such as loops. Additionally, using visual blocks that users drag-and-drop and snap together to build a program or solve a challenge facilitates interaction and engages the user creatively. The specific visual programming editor we chose to use is Blockly for the following reasons.

2.5.3 Blockly

Created by Google, Blockly is a JavaScript library that developers can use to build a visual coding editor that enables users to write programs by snapping puzzle-piece like blocks of code together. As the blocks are put together, the library generates and executes the corresponding code, showing the results to the user. We decided to use Blockly because of its benefits for both developers and users. Blockly, like *Scratch*, is open source, but Blockly is designed for developers to integrate into their own applications, whereas *Scratch* must be exported or embedded on external websites. Since Blockly is open-source and can be used in custom applications, we can extend its functionality to create an application personalized to our target audiences interests and skills. Blockly was also influenced by *Scratch* and *App Inventor*, so it is easy for young children to use, making it the best tool for engaging our target age group of five to eight year old girls. Blockly, furthermore, meets the design constraint that our application must be web-based and gives our application portability, since it is compatible with Chrome, Firefox, Safari, Opera and Internet Explorer.

2.5.4 Technologies

Our application is built primarily using Javascript, making it a client-heavy application. Blockly, the visual coding editor we chose to use, is 100% client side and written in Javascript, which can then be compiled to Javascript, Dart, or Python, but we chose to only use Javascript, as we are all familiar with the language. To simplify the creation of our application and achieve consistency, we intially decided to use the AngularJS framework. We chose the AngularJS framework over similar frameworks such as Backbone.js and Ember because it is well suited to single page applications. Additionally, AngularJS has been used to develop Blockly applications and does not necessitate a dominant rest API application, which benefits our client-heavy application. AngularJS also does not have as steep of a learning curve as Ember does. Once we began implementation, however, we decided not to use AngularJS because Blockly provided us with a sufficient framework for building our application. When necessary, we used JQuery to simplify the Javascript that we needed to write, since it implements many common functions. To save the users blocks and then restore them when they return, we used calls to export to and return from XML. To enable users to save, load, and share their work, we hosted our application on Google App Engine, which provided these features that we could access using Python and the cloud storage API. Additionally, we used a NoSQL database, provided with Google App Engine, to save the user's progress. We, as such, enable our users to play at their own pace to maximize their learning and enjoyment.

Chapter 3

Project Management

3.1 Test Plan

Our test plan can be broken down into three main sections: testing of the avatar, testing of the games and testing both together. The first part involved unit testing on the processes of signing up, signing in, and using the Blockly pieces. We then conducted functional testing on the different combinations of Blockly pieces and the avatars they produce as we wrote the code to make sure that adding new features did not break existing code. Next, each of the challenges was tested separately. This phase tested that the correct inputs or combination of blocks produced the correct outputs, and therefore solved the maze or puzzle. Finally, we integrated the games with the creation of the avatar to test the system as a whole. An important part of this phase included testing the possible combinations of challenge outcomes, such as successes and failures, and the resulting actions, such as features unlocked. After we tested the system as the developers, we conducted user testing with groups of Girl Scouts ages six to eight. The results of our user testing will be discussed in the following section.

3.2 Test Results

Our user group consisted of nine girls ages six to eight. By observing the Girl Scouts interact with our application, we learned that the directions need to be as simple as possible with easy vocabulary. The younger girls who are just learning to read struggled with words such as ”sidebar”. User testing also taught us that we cannot assume that our users understand how to use a mouse and a desktop, which is what we used for the testing, because the girls did not immediately understand how to ”drag and drop” blocks onto the canvas to customize their avatar. After the Girl Scouts spent half an hour playing *Code Girl*, we asked them questions regarding their previous computer science game experience, their interest in programming before and after the game, and what they would change or add to the game. The overall feedback was positive, with a majority of the girls saying they would play the game again and were interested in playing more games to learn programming. There were two girls who had previous experience with games such as *Scratch* and *Code.org's Studio* who were not as challenged by the games, but they said they really enjoyed getting to create the avatar and wished they could change her hair color and style. User testing helped put us in the mind-set of five to eight year old girls so we could simplify directions and vocabulary to make *Code Girl* even more user-friendly and intuitive.

3.3 Project Risks

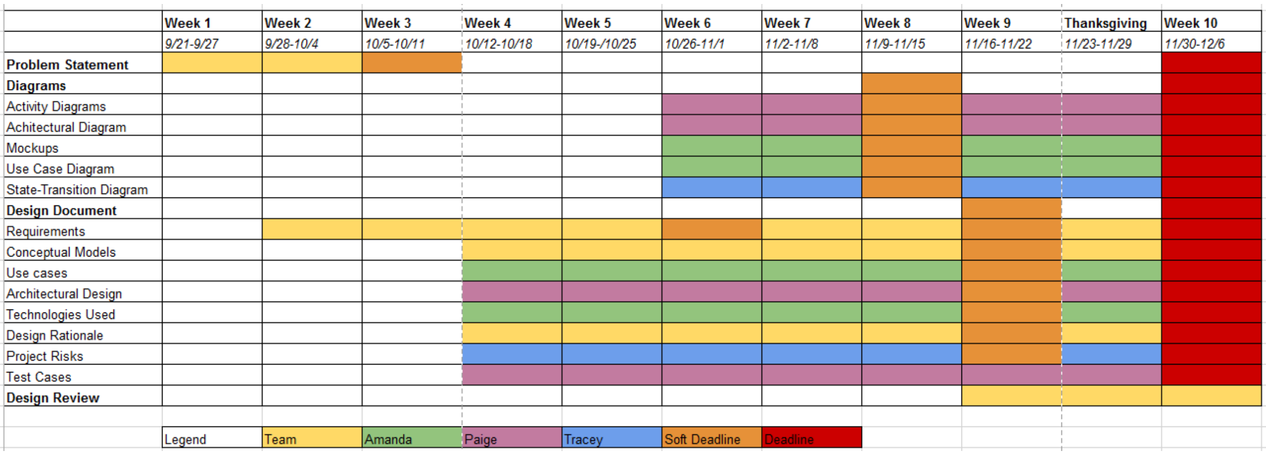
A risk analysis is conducted to identify possible risks and the costs they could have on the system or project as a whole. As shown in Table 3.1, each risk is given a probability from zero to one as well as a severity from zero to ten. The total impact of the risk is then calculated by multiplying the probability and severity together. By considering these risks at the beginning of the project, we aimed to take action to prevent the risks and their associated consequences from happening. Ironically, the biggest risk we encountered was not one we included in our original risk analysis. Originally, we chose PHP as our language to communicate with the server because Google App Engine is compatible with PHP, and it is the language we know best. About three quarters of the way through the project, we realized that Blockly uses Python to communicate with Google App Engine for storage of the user's blocks. In order to successfully reload the users blocks and level when they log in, we would also have to use Python, a language none of us knew. The severity of the problem seemed pretty high, as one of us had to learn a new language to complete the project, and we did not know how easy or quick that would be. Thankfully, we were able to do some quick tutorials and finish the required Python parts of our project to successfully integrate with Blockly. We learned that we should have included a risk about the technology or languages changing, so we would have been more prepared to handle the mini crisis we encountered.

Risk	Consequences	Probability (P)	Severity (S)	Impact (P x S)	Mitigations
Bugs and Errors	We will have to go back into the code, find out what's causing the problems, and fix them	1	3	3	Do ample testing every step of the way to find bugs sooner and use debugging software
We run out of time	Not all desired functions of the project are completed by the design conference.	0.3	9	2.7	Prioritize our work and implement the most important functionality first and perform extensive time management
Incomplete requirements	We do not implement a functionality our system needs	0.2	8	1.6	Review all our requirements to make sure they meet and match our intended outcomes and continuously talk to our advisor regarding the requirements
Changing scope/objectives	We might need to redo sections of code and it can delay our schedule	0.2	5	1	Prioritize requirements and functions to keep the project within the scope and limit the number of functions added after we begin implementation

Table 3.1: Our risk table for the project.

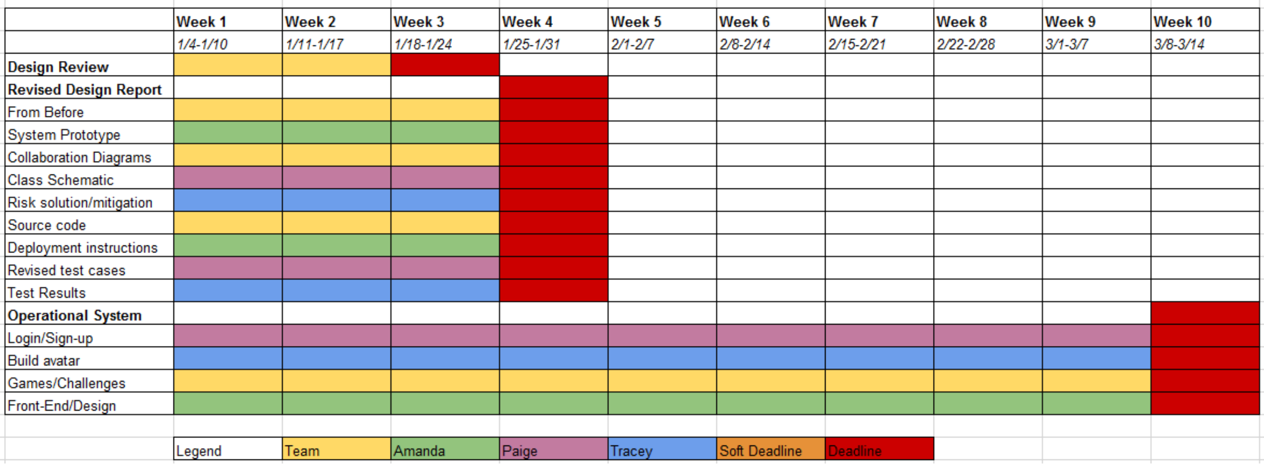
3.4 Development Timeline

One of our mitigation strategies was to prioritize our work and perform extensive time management. We planed to accomplish this by sticking to our Gantt chart. A Gantt chart displays the amount of work done or production that is completed in certain periods of time in relation to the amount planned for in such periods. It is important that we created detailed charts for each quarter to manage our time and energy. The following figures show our proposed development timeline by quarter. Initially, they included the basic deliverables and split them up by team member. However, as we better understood these deliverables, we divided them into specific components to make our Gantt chart even more effective.



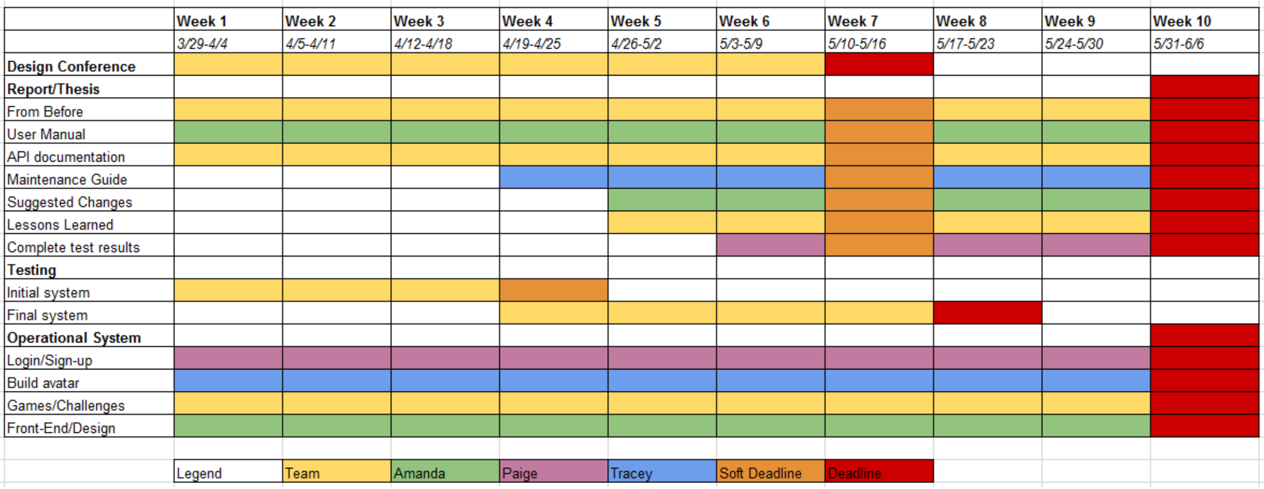
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Thanksgiving	Week 10
	9/21-9/27	9/28-10/4	10/5-10/11	10/12-10/18	10/19-/10/25	10/26-11/1	11/2-11/8	11/9-11/15	11/16-11/22	11/23-11/29	11/30-12/6
Problem Statement	🟨	🟨	🟧								🟥
Diagrams								🟧			🟥
Activity Diagrams						🟪	🟪	🟧	🟪	🟪	🟥
Achitectural Diagram						🟪	🟪	🟧	🟪	🟪	🟥
Mockups						🟩	🟩	🟧	🟩	🟩	🟥
Use Case Diagram						🟩	🟩	🟧	🟩	🟩	🟥
State-Transition Diagram						🟦	🟦	🟧	🟦	🟦	🟥
Design Document									🟧		🟥
Requirements		🟨	🟨	🟨	🟨	🟧	🟨	🟨	🟧	🟨	🟥
Conceptual Models				🟨	🟨	🟨	🟨	🟨	🟧	🟨	🟥
Use cases				🟩	🟩	🟩	🟩	🟩	🟧	🟩	🟥
Architectural Design				🟪	🟪	🟪	🟪	🟪	🟧	🟪	🟥
Technologies Used				🟩	🟩	🟩	🟩	🟩	🟧	🟩	🟥
Design Rationale				🟨	🟨	🟨	🟨	🟨	🟧	🟨	🟥
Project Risks				🟦	🟦	🟦	🟦	🟦	🟧	🟦	🟥
Test Cases				🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟥
Design Review									🟨	🟨	🟨
	Legend	🟨 Team	🟩 Amanda	🟪 Paige	🟦 Tracey	🟧 Soft Deadline	🟥 Deadline				

Figure 3.1: Our Gantt Chart for Fall Quarter.



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
	1/4-1/10	1/11-1/17	1/18-1/24	1/25-1/31	2/1-2/7	2/8-2/14	2/15-2/21	2/22-2/28	3/1-3/7	3/8-3/14
Design Review	🟨	🟨	🟥							
Revised Design Report				🟥						
From Before	🟨	🟨	🟨	🟥						
System Prototype	🟩	🟩	🟩	🟥						
Collaboration Diagrams	🟨	🟨	🟨	🟥						
Class Schematic	🟪	🟪	🟪	🟥						
Risk solution/mitigation	🟦	🟦	🟦	🟥						
Source code	🟨	🟨	🟨	🟥						
Deployment instructions	🟩	🟩	🟩	🟥						
Revised test cases	🟪	🟪	🟪	🟥						
Test Results	🟦	🟦	🟦	🟥						
Operational System										🟥
Login/Sign-up	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟥
Build avatar	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥
Games/Challenges	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥
Front-End/Design	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥
	Legend	🟨 Team	🟩 Amanda	🟪 Paige	🟦 Tracey	🟧 Soft Deadline	🟥 Deadline			

Figure 3.2: Our Gantt Chart for Winter Quarter.



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
	3/29-4/4	4/5-4/11	4/12-4/18	4/19-4/25	4/26-5/2	5/3-5/9	5/10-5/16	5/17-5/23	5/24-5/30	5/31-6/6
Design Conference	🟨	🟨	🟨	🟨	🟨	🟨	🟥			
Report/Thesis										🟥
From Before	🟨	🟨	🟨	🟨	🟨	🟨	🟧	🟨	🟨	🟥
User Manual	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟩	🟩	🟥
API documentation	🟨	🟨	🟨	🟨	🟨	🟨	🟧	🟨	🟨	🟥
Maintenance Guide				🟦	🟦	🟦	🟧	🟦	🟦	🟥
Suggested Changes					🟩	🟩	🟧	🟩	🟩	🟥
Lessons Learned					🟨	🟨	🟧	🟨	🟨	🟥
Complete test results						🟪	🟧	🟪	🟪	🟥
Testing										
Initial system	🟨	🟨	🟨	🟧						
Final system				🟨	🟨	🟨	🟨	🟥		
Operational System										🟥
Login/Sign-up	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟥
Build avatar	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥
Games/Challenges	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥
Front-End/Design	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥
	Legend	🟨 Team	🟩 Amanda	🟪 Paige	🟦 Tracey	🟧 Soft Deadline	🟥 Deadline			

Figure 3.3: Our Gantt Chart for Spring Quarter.

3.5 Societal Issues

As engineers, it is important to consider how the applications we create affect the world around us. We must conduct ourselves in a way that aligns with our ethics as well as the society as a whole. The following societal issues are discussed as they relate to us and our project, *Code Girl*.

3.5.1 Ethics

Because we were the people defining the requirements and specifications for our project, we had to think differently about ethical issues in ways that a group creating something for an outside customer might not. For example, we did not have a company's code of ethics to follow, so we had to decide on our own. Also, unlike most other groups, we had to consider ethics involving children, since we developed our application for a certain age group and tested it with real children. These differences prove we had to carefully consider the ethics of our project.

Ethics in Our Team

Given the nature of our project, we examined the Association for Computing Machinery's (ACM) Code of Ethics and Software Engineering Code of Ethics. [2] The general ACM Code of Ethics talks about morals, professional responsibilities and organizational leadership. These helped us to prevent any ethical issues within our group and foster communication by driving us to be trustworthy and honoring assigned responsibilities. The Software Engineering Code of Ethics can be applied to how we as individuals act as developers. This code focuses mainly on acting consistently with the public interest and upholding integrity. We kept both of these in the front of our minds throughout our senior design project.

Ethics in Our Project

The most important ethical discussion we needed to have involved working with and designing for young girls. We wanted the web application to purposefully appeal to girls, however there could be sexism involved in us deciding what exactly that means. Obviously, not every girl likes pink and princesses, so it was important to offer a wide variety of clothing and accessories for the avatar that will appeal to a multitude of girls. Another ethical dilemma involves internet security because parents need to have control over what their child sees on the internet. We had to take this into consideration if we wanted the child to have a profile where they can sign in and save their progress. There are legalities that go along with asking children to provide an email address and password to sign up, usually a parent must do it for them. Lastly, we wanted to interact with young girls to test our project throughout the design process. We hoped to find out what appeals to them, ask them questions about our design and get feedback on how to make it better. There are legal issues with contacting minors, as well as a Santa Clara University ”human subject review process,” so we partnered with local Girl Scouts to help us. Making sure we uphold our ethics required some research to make sure we follow laws, make our interactions constructive, and make our web application safe and appropriate.

3.5.2 Social Context

From a social standpoint, we aim to interest a very specific group of young girls in computing to help benefit society. Although an interest in playing our game or learning to program does not necessarily mean the user will go on to study computing or have a job in the field, we hope to at least help spark the idea and the possibility at a young age. Computing is growing so fast that there are not enough graduates to fill all the jobs, or enough women in the jobs currently available, exacerbating the gender divide in technology. By interesting young girls in computer science we hope to help fill those jobs with capable, smart women when they grow up.

3.5.3 Political, Economic, Health and Safety, Manufacturability Issues

Code Girl is a free, web-based application, so we do not think there are any relevant political, economic, health and safety or manufacturability issues to discuss.

3.5.4 Sustainability and Environmental Impact

Since *Code Girl* is a web-based application, it is sustainable and environmentally friendly. We do not need to consider how to recycle resources or the impact resources may have on the environment because our application uses the internet and any issues involving the internet are out of the scope of this project. Users do not need any additional hardware or devices to use our application.

3.5.5 Usability

User testing helped put us in the mind-set of five to eight year old girls to help us make *Code Girl* even easier and more intuitive to use. We included animations in the directions to help the young girls better understand how to use our application because their reading levels and vocabulary vary. The user can also slow down the animations in trying to solve the challenges to easily understand and learn what the blocks are doing.

3.5.6 Lifelong Learning

Code Girl has prepared us for a life of learning new technologies as well as fueled a passion in us to help awaken the love of learning in new generations of young girls and programmers. This project has taught us how to quickly learn important and relevant parts of a language on our own. For example, we had to learn Python in a short amount of time to be able to integrate with Blockly and Google Cloud Storage to save what level the user was on. Learning how to distinguish what features of a language are applicable to the functionality we needed to implement proved a useful skill that will help us easily pick up new languages on our own in the future. *Code Girl* also inspired us to extend our passion for encouraging and supporting women in computing to a new generation. Channeling this passion, we want our program to reach as many young girls as possible, and we would love to partner with summer camps, schools and/or the Girl Scouts to include *Code Girl* as part of a curriculum to inspire young girls to learn how to program.

3.5.7 Compassion

Our compassion is evident in our desire to educate and enrich an often overlooked portion of the youth, who are not given many educational programs or games specifically designed for them. By beginning to teach basic computer skills and computer science ideas to young girls, we are giving them the opportunity to develop their interests and skills to help them succeed.

Chapter 4

Conclusion

4.1 The Final Application

When users type in the URL for our website, they are taken to the application's homepage, pictured in Figure 4.1, which introduces them to the superhero story of the game. A slideshow presents Grace, Ada, and the objective of the game, which is to customize a unique avatar while learning computer science concepts.

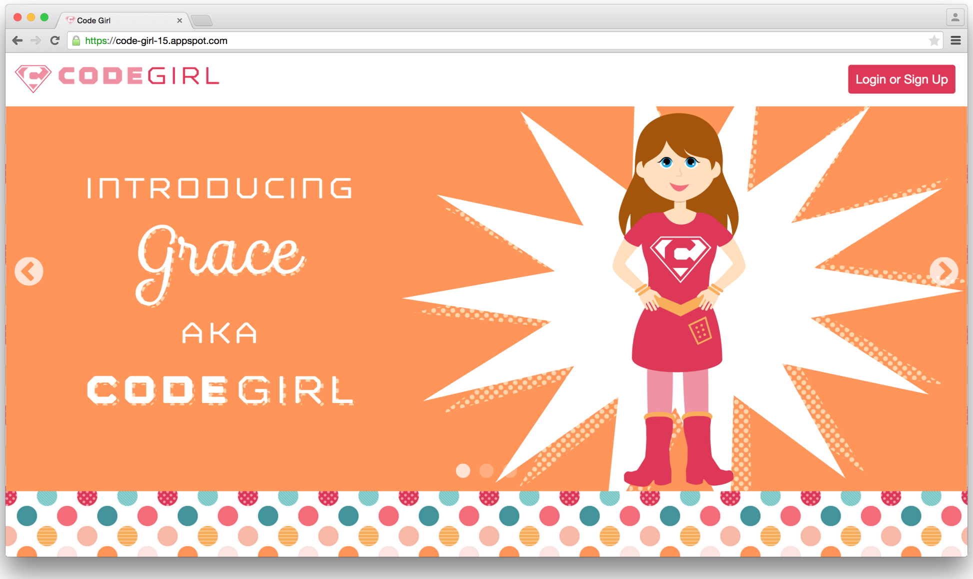
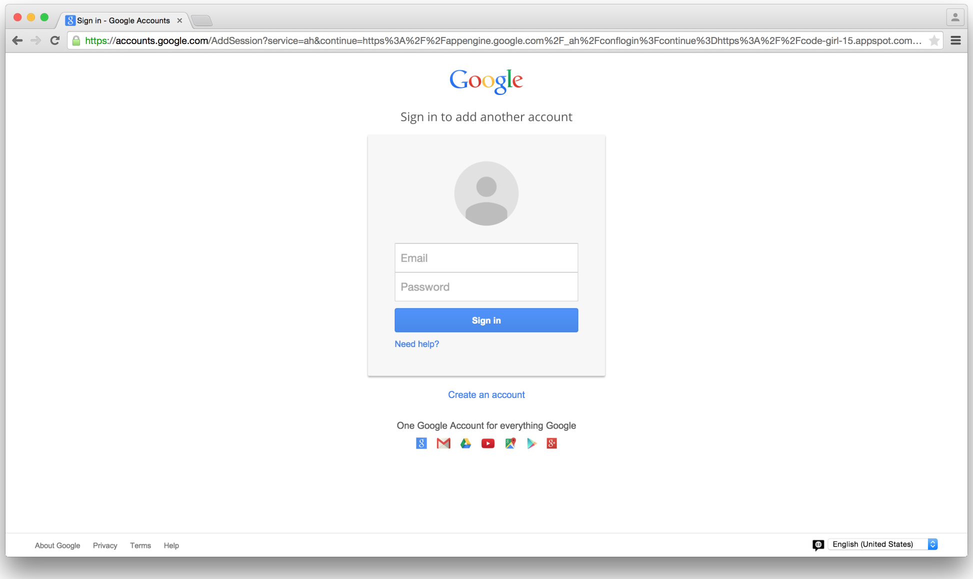


Figure 4.1: Users are initially taken to the application's homepage.

4.1.1 Login

To log in to the application, users select the "Login or Signup" button on the homepage. Because our application is hosted on Google Apps Engine, user registration and authentication is performed through Google Accounts. Consequently, users are taken to the main login page for GMail, in Figure 4.2, where their parents can sign up for an account or log in for them.



https://accounts.google.com/AddSession?service=ah&continue=https%3A%2F%2Fappengine.google.com%2F_ah%2Fconflogin%3Fcontinue%3Dhttps%3A%2F%2Fcode-girl-15.appspot.com...

Sign in to add another account

One Google Account for everything Google

Figure 4.2: Users' parents login or signup via Google Accounts.

4.1.2 Avatar Customization

After users log in to the application, they are taken to the page for the first avatar customization level, show in Figure 4.3, where they can see their avatar and begin customizing her. The page is split into the Blockly canvas on the left, which includes a toolbar where blocks are grouped and selected, and the avatar visualization on the right. They can start by adding a shirt from a toolbar, and this is where our users first get exposed to the Blockly blocks. Users can then add a color block to change the color of the shirt they selected. If they do not like the color or the shirt they selected, they can drag the block to the trash and select a different one. The avatar serves as a reward system to engage girls and make them want to complete the challenges, which introduce them to computer science. In order to get more accessories for the avatar, users need to complete the challenges, so they select the "Unlock Accessories" button above the avatar to go to one.

4.1.3 Challenges

For the first challenge, pictured in Figure 4.4, users need to complete a puzzle, where they connect a block that has the picture of an animal with the block that lists the animal's name and select how many legs the animal has. The purpose of this challenge is to familiarize users with the Blockly

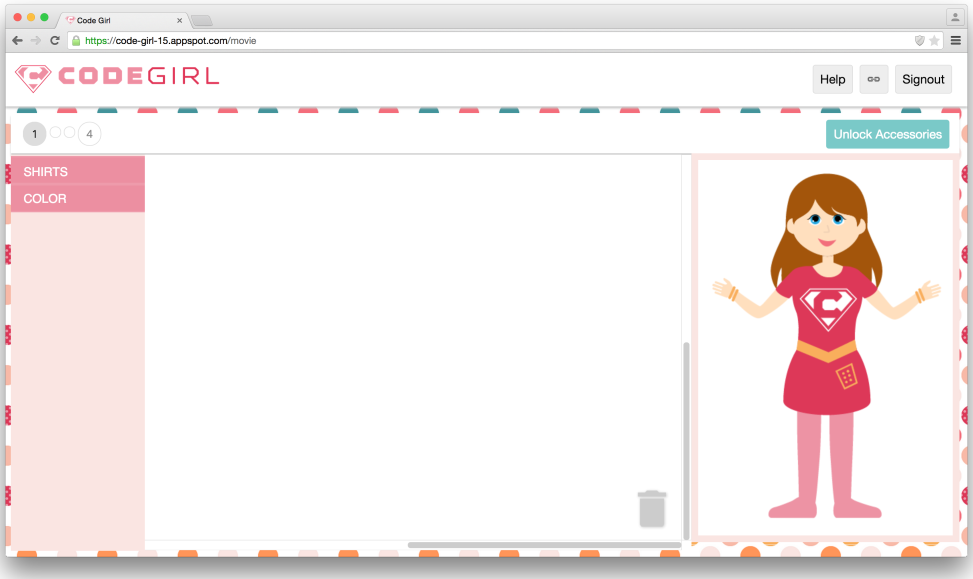


Figure 4.3: Users customize their avatar, on right, using the accessories from the toolbar.

blocks and show them how to make selections within the blocks. Once users think they have the correct answer, they can check the "Check Answer" button, and if they are right, more accessories will be unlocked and the users are redirected to the avatar page.

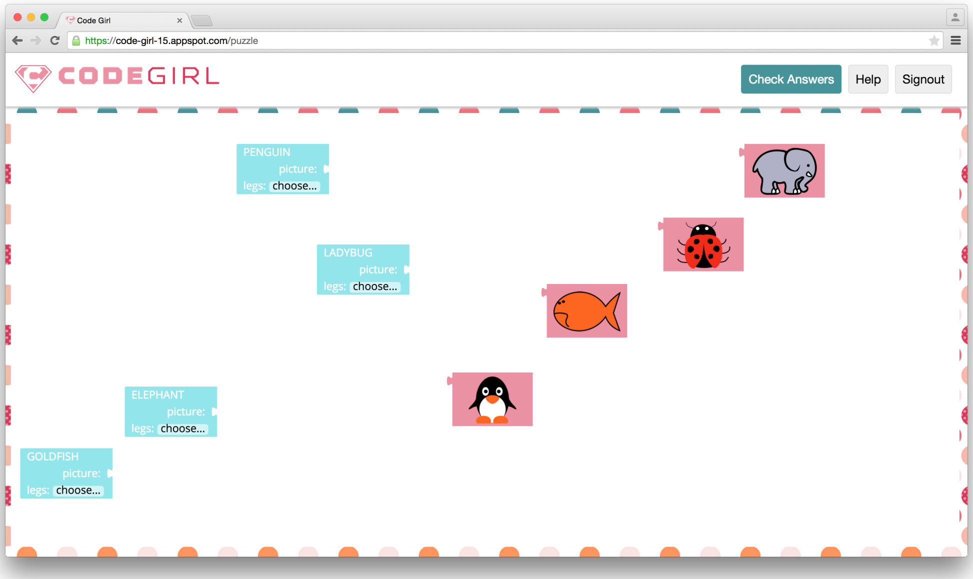


Figure 4.4: Users complete an animal puzzle for the first challenge to introduce them to Blockly.

Now, on the avatar customization page, the users can see a new category of blocks they can use to customize their avatar further. For example, upon completing the puzzle challenge and being taken to the second level in the avatar creation process, users can then add bottoms to the avatar and change their color as well.

In the second challenge, shown in Figure 4.5, we start teaching coding concepts, the first being sequentiality. We give users all of the blocks they need to complete the challenge on the canvas to begin with because we want users to focus on the importance of having code in order, and not on deciding how many steps to take. For this challenge, users need to make the robot sidekick move around in a square, as depicted in the instructions the user sees upon starting the challenge. Once the user has the right answer, they can click the "Run Program" button and see the result of their code.

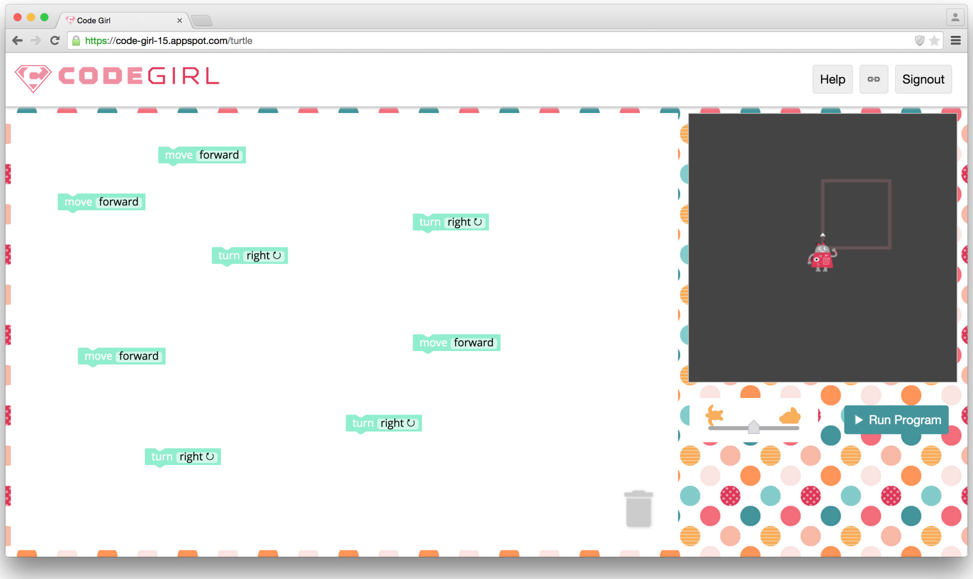


Figure 4.5: Users learn sequentiality by connecting the blocks of code on the canvas.

In the following challenge, seen in Figure 4.6, we increase the difficulty of the challenges by building on the computer science concepts we teach, introducing users to loops as an alternative to sequentiality. This is similar to the previous challenge, but we do not put all the blocks on the canvas because we want users to learn about loops and how they simplify a series of repeated steps. If a user selects the wrong blocks, for example, she on picks a loop and a move forward block, then she can run the program, and from the robot's action, see that she completed the challenge incorrectly. The user is then able to reset the challenge and try it again. Users are also able to slow down the speed the robot moves at, so users can run the program and see exactly what movement corresponds to each block.

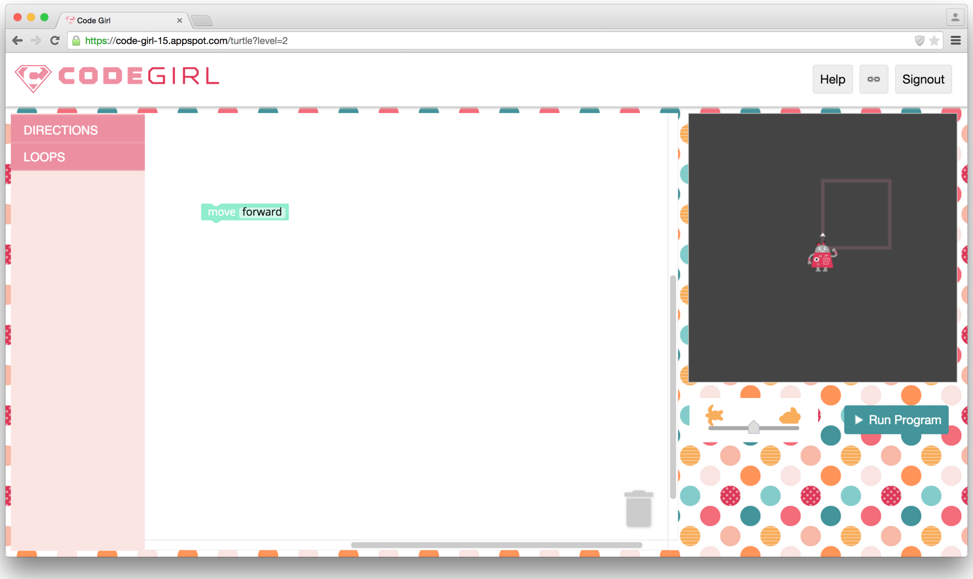


Figure 4.6: Users learn how to use loops in place of iteration to repeat actions.

4.1.4 Saving the Avatar

Once users are happy with how their avatar looks, they are able to save a picture of their avatar by selecting the ”Save Image” button on the final avatar customization level, shown in Figure 4.7, and then print it out to share with their friends. When users are done playing the game, they can logout, and the application will save their progress. Users can also save their progress at any point in the game and return to this point later on.

4.2 Future Work

We are currently working on making the application fully-responsive, so when the application is rendered on smaller screens, the display is not distorted. Now, when the application is viewed on a small display, such as on a tablet, the Blockly canvas becomes very small, making it difficult for users to drag blocks onto it and connect them with other pieces. By making our application more compatible with tables, we are engaging users in a mode they are more comfortable with, since as we learned in user testing, children are less familiar using a desktop and mouse to play games than they are using mobile devices. We are also keeping up to date with the growing popularity of mobile games and making our application competitive with programs like those developed by *Made with Code*. Additionally, when our application is deployed on Google Apps Engine, it currently runs and loads a bit slowly. To optimize our application, we plan to profile our code base to identify

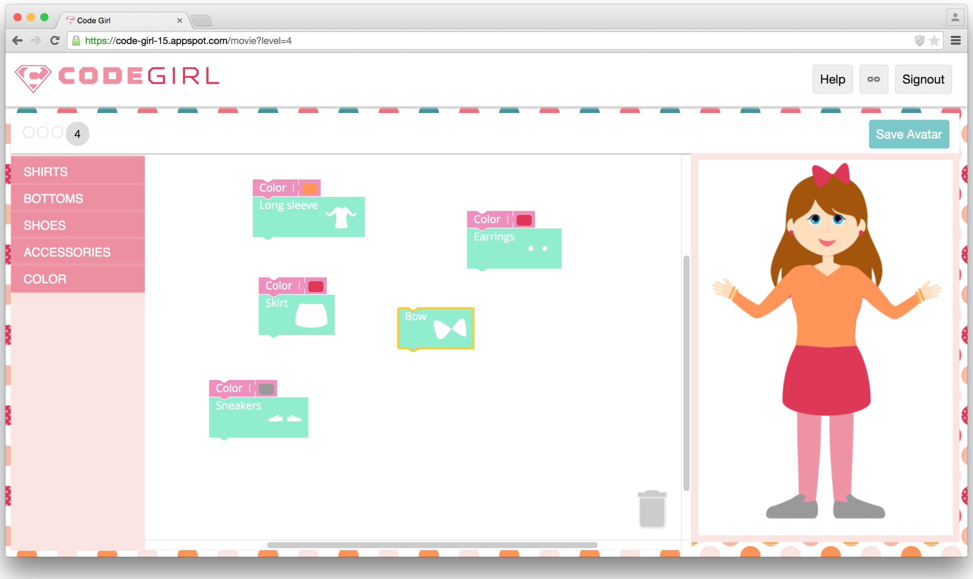


Figure 4.7: Users can save the avatar as image once done customizing.

and resolve performance bottle-necks and use compressed JavaScript files when we deploy the final application on Google Apps Engine. Finally, we are in the process of developing and testing more games and accessories, as requested by the Girl Scouts who used our application.

4.3 Project Assessment

An assessment of the disadvantages and advantages of Code Girl in introducing girls to computer science is provided in the following sections.

4.3.1 Disadvantages

With the target age range of five to eight year old girls we chose for our application, there is a substantial difference in reading comprehension and logic skills among users. As such, the application, in particular the challenges, can be quite difficult for some users, yet very easy for others. We learned in user testing, however, that prior experience with a visual programming environment was a better indicator than age of how quickly users progressed through the application. Additionally, young children are more familiar with playing games on mobile devices than they are with playing games on a desktop or laptop, thus a native mobile application may be a more effective means of engaging them in computer science than a web-based application, such as ours. To address this weakness, we could make a native mobile version of Code Girl, in addition to the web-based application we have already created.

4.3.2 Advantages

Our application appears to be effective at interesting young girls in computer science. As we saw in user testing, the girls were engaged in the game and excited to learn that by completing challenges to customize their avatar, they were learning code. Moreover, all but one of the girls reported that they enjoyed playing the game, and most were actually disappointed that they finished the game quickly, indicating that the superhero story and customization incentive successfully engage our target audience in learning basic computing concepts.

4.4 Lessons Learned

There are few resources designed specifically for young girls that teach basic computer science skills. To fill this gap, we developed Code Girl, which uses Blockly to interest girls in computing using familiar concepts, such as puzzles, customization, and story-telling.

4.4.1 Objectives Met

In our application, successfully completing challenges unlocks new accessories, incentivizing game play, so our users continue learning new concepts while dressing up their avatar as they would a doll, appealing to their unique interests while educating, empowering and inspiring them.

Bibliography

[1] C. Corbett, C. Hill, A. St. Rose. *"Why So Few? Women in Science,Technology, Engineering, and Mathematics"*, American Association of University Women, Washington, DC, 2010.

[2] D. Gotterbarn, S. Rogerson. *"Computer Society and ACM Approve Software Engineering Code of Ethics"*, Computer 32.10, 1999, 84-88.

[3] T. Bell, B. Gibson. *"Evaluation of Games for Teaching Computer Science"*, Proceedings of the 8th Workshop in Primary and Secondary Computing Education. Ed.: ACM. Aarhus, Denmark, 2013.