Programming For Underrepresented High School Students – Requirements

Isak Foshay, Rebecca Croysdale, Cristian Mann, Roy Simons

Spring Term: May 26, 2020

Team: CS67 "Get Good, Kids!"

CONTENTS

I Introduction 3

I-A System purpose . . . 3

I-B System scope . . . 3

I-C System overview . . . 3

I-C1 System context . . . 3

I-C2 System functions . . . 3

I-C3 User characteristics . . . 3

I-D Definitions . . . 3

II System requirements 3

II-A Functional requirements . . . 3

II-B Usability requirements . . . 5

II-C Performance requirements . . . 5

II-D System interface . . . 5

II-E System modes and states . . . 6

II-F Environmental conditions . . . 6

II-G System security . . . 6

II-H Information management . . . 6

II-I Policies and regulations . . . 6

II-J System life cycle sustainment . . . 6

II-K Packaging, handling, shipping and transportation . . . 6

III Change Log 7

I. Introduction

A. System purpose

Teaching coding to high school students that have limited or no access to internet, computer science concepts, or computer labs. The goal of teaching these high school students is to increase their interest in pursuing an education and career in computer science and technology.

B. System scope

The system will supplement the students' computer science education with engaging lessons that gradually increase in difficulty. The lessons will increase in difficulty and in the amount of code that has to be written to complete the lesson. The system will also allow the teacher to create new lessons and adjust or remove current lessons. The teachers will be able to view the student's code as well as run the code through a grading script.

C. System overview

The system will be a collection of lessons to teach basic coding skills and convey programming concepts. New lessons can be added and current lessons can be adjusted by the teacher. The system will hold user data including user info like username and password, as well as lesson progress and the score generated from the teacher created grading script. The students' lesson progress and score can be accessed by the teacher, and if needed, the teacher can delete a student's account.

1) System context: The system should be usable in public high schools that have limited to no access to internet or technology. The web application should be able to run on older systems. The web application should be view-able and easily used on both mobile devices and desktop devices.

2) System functions: A student should be able to send their code to be graded and receive a numerical score in response based on a script held in the server for each lesson. Each student should have a score stored for each existing lesson. If the student completes the lesson correctly they should receive 100 points. Depending on the written grading script of the lesson, the student should be able to receive partial credit for work that is not fully correct. If the code does not match any partial or full credit options in the script, or if a student has not attempted a lesson, the student will receive 0 points.

3) User characteristics: Primary users of this system will be high school students who do not have access to quality technical education. The users can have experience with some tech devices, like mobile phones, but it is expected that most users have limited availability to tech. Secondly, it's created for teachers that teach the high school students. The application is designed so that it is easy for teachers to create lessons even if they do not have experience with coding.

D. Definitions

Artificial Intelligence: AI

II. System requirements

A. Functional requirements

Select a topic: Once a user has registered a logged in, they can select a page to view from the home menu or from the header menu. Clicking on an option will bring the user to the selected page.

Student: The pages include the lesson menu, student's grades, and the option to log out.

Teacher: The pages include manage students which allow the teacher to check students progress, manage lessons where lessons can be added or adjusted, and the option to log out.

Select a lesson: Student: Students will see an option named "Lessons". After clicking on it, they will see a list of the existing lessons as buttons and can pick a lesson to work on by clicking on the name of the lesson.

Teacher: Teachers will see an option named "Manage Lessons". After clicking on it, they will see a table containing existing lessons, as well as a button to add new lessons.

Lesson: Student: Upon selecting a lesson, the student is presented with what is expected of the student, a hint to help the student complete the task, and a Blockly environment. If the student has already worked on this lesson, their saved progress will load from the database.

Teacher: The teacher can write what is expected of the student, give a hint for the student, add what the answer should be as a helpful reminder, and interact with a Blockly environment to write the grading script. The grading script takes in the output of the student code, compares it to different teacher defined options, and then outputs a corresponding grade from 0 to 100.

Code insertion: Within each lesson there will be a window to insert code containing the Blockly environment. This environment is where the student will put together blocks of code which will be graded and given a score depending on how well they performed. A similar environment is presented when teachers are editing a lesson's grading script.

Code verification: Upon submitting a piece of code it should be verified to contain no syntax / logic errors, which should be done by Blockly itself. Then the code should be run within a virtual environment to avoid crashing the system if a bad actor were to try and tamper with the software. Finally, the teacher has to write a grading script in the Blockly environment that will check the answer that the student has submitted. The grading script does not check what the student's code looks like, only the return value. For example, if the student's code should output an integer "20," the grading script will output a grade of 100 if this is the case, and 0 otherwise.

Lesson progression: The lessons will all be available from the start. The teacher should instruct the students which lesson to attempt. The reason for having all the lessons available is that some students might have prior knowledge, or learn faster than others, so they should have the option to attempt more difficult lessons. Otherwise, experienced students may view the lesson progression as busy work. The risk is that an inexperienced student will attempt a lesson that is too difficult, but they have the option to just exit out of the lesson and attempt a different lesson. Their progress is saved so they can resume the difficult lesson at a later time.

Point system: When completing a lesson a point value should be awarded based on how well they did during the lesson. The amount of points indicate how well a user has done on their lesson. Full points means the user has done optimal work. A score lower than full points will show the user that there is room for improvement for the concept, and that they should try the lesson again. The teacher themselves can assign the amount of points they give for the answer the student provided. Points should be awarded between 0 and 100 if the teacher wishes to correspond with a traditional grading scheme.

Navigate to the user progression: There should be an option in the main menu to navigate to the currently logged in student's grades. They should be able to check which lessons they attempted and how well they performed on each individual lesson. Having the ability to check your score progression, can give the students a feeling of accomplishment if performing well. The teacher also has access to the progression for each student and can even see the Blockly code the student put together for each attempted lesson. Having the ability to check the students code allows the teacher to see if the student

needs help or to check if a student is academically dishonest. This also allows the teacher to assign partial points outside of the app if the student's attempt did not meet any of the grading script criteria.

Tutorial: A detailed tutorial should be provided in the Github of the program. This tutorial can also be handed out to school's that receive this system. It should walk through how to register/log-in, how to create and complete a "Hello World!" lesson, and a more complicated computer science concept lesson. These tutorials should be for both the student and the teacher perspective.

Lesson creator and adjust option: A teacher can create their own lessons and share them for other students to complete. The teacher should also be able to adjust lessons that are already available. Students should not have the option to create or adjust lessons, because the risk of misuse. For example, a student could repeatedly create new lessons and clog the website. When creating or adjusting a lesson, everything within the lesson should be open for change to allow the teacher to personalize the lesson optimally. Blockly should check that the grading script has no syntax / logic errors.

Hints: Teacher have the ability to write hints with each lesson they adjust or create. They can give worded examples, or tell the students which blocks to use and where to get them from. It's the choice of the lesson creator what to put in the hints section. The student users should see the hints once they open the lessons they picked.

B. Usability requirements

The product should appeal to high school students and motivate them to want to learn to code. It should start with accessible lessons, with a low level or even no coding experience. The lessons should gradually increase in difficulty and the amount of coding to be done to complete it. A progression page should be developed for the project which collects minimal personal data and provides the scores achieved per lesson.

C. Performance requirements

Should be able to run on potentially older hardware, meaning the device used to access the app being run from the NUC's server. For lessons regarding AI, an Intel compute chip can be used to supplement the heavy processing needed for advanced operations. It is expected that the users do not have the newest hardware, which makes it a poor choice to design the program for new hardware. Test should be run on older hardware to conclude if the program is able to run, if it runs the performance of the program should be checked.

D. System interface

The system interface for students should consist of a menu for lesson selection. User progression should also be added to the interface as well as a log out option. This should all be accessible through a hamburger menu if the user is logged in. The progress of a lesson should be saved, in case that user switches to a different part of the program, or loses connection. The teacher should see an option to manage lessons or manage students. In the lessons there should be an option to create a new lesson or to adjust an existing lesson by clicking on an entry in the table. In the manage student screen, all important information of the registered students should be available. There should be the option to click on an individual student, to see their individual progress per lesson which if clicked on show the Blockly environment with the student's answer if the lesson was attempted. Finally, the teacher should have the option to delete a student user, in case of a fake account or other various valid reasons.

E. System modes and states

The system will consist of 12 states. The log in screen, register for students, register for teachers, home menu for students, progression screen for students, lesson selection screen for students, current lesson screen, home menu for teacher, progression screen for teacher, manage lesson screen and the lesson create/adjust screen. The two modes that would exist to interact with these states would be as a student or teacher. If no user is logged in, only the log in screen and register screens will be accessible.

F. Environmental conditions

The project is an application which will be used on machines. Since the project is only software there are no environmental conditions.

G. System security

The main goal for system security is to keep the privacy of the users by keeping their personal data safe. This includes passwords and email addresses. User accounts will have their password hashed so, in the case of a security breach, access to their account will be safe. Teachers will also register using a timed-one-time password (TOTP) as dual authentication to ensure access to data manipulating permissions are secured.

H. Information management

Users that have an account will have their username, first and last name, password, score, and progress stored. To elaborate on progress, if a user exits or submits a lesson the current progression of the users blocks in the Blockly environment will be stored in the database. The lessons will be stored in the database, this will include the goal, hint, Blockly environment, answer of the lesson, and grading script.

I. Policies and regulations

Regulate hard coding answers to questions or any potential cheating. Each individual lesson should teach the same concept per user. The users shall have the same task as other users and should learn the same concepts, but can implement their code in different ways into the Blockly environment, while still being able to get full points.

J. System life cycle sustainment

For life cycle sustainment, the lesson creator option for teachers was made, allowing to the teacher to create as many lessons as they want. It could be interesting to eventually give student users the ability to create lessons, so they can see the other side of the lesson. The students will be able to see how their work is graded, which can give them a better understanding of computer science concepts.

K. Packaging, handling, shipping and transportation

The program is a web based application served via an Intel NUC or other wifi hotspot enabled linux system. Thus, the product will either be packaged as an Intel NUC or the program can be found on Github and installed on a linux system. The lessons are part of the application. In addition to the lessons, an Intel compute stick will come packaged for the late AI lessons.

III. Change Log

- Changed project title from "Neuromorphic For Kids" to "Programming For Underrepresented High School Students"
- Changed the project's primary users to not only be created for Native American high school students that have limited access to internet and, but for all high school students have limited access to internet and tech.
- Added teacher to the primary users for this application.
- Changed lessons from being divided in themes, to having all lessons together.
- Replaced the scoreboard plan with a personal progression page, which holds the score of the student per lesson.
- Changed the score system from picking between point or start system, to points system where between 0 to 100 points can be earned per lesson.
- Updated the functional requirements to include the teacher options and abilities. The teacher has the ability to add or adjust lessons. The teacher also has the ability to see each students progression, as well as their implementation of each individual lesson.
- Changed the availability of lessons from unlocking lessons to being all available.
- Changed the grading script to be created by the teacher. The teacher must create a grading script in the Blockly environment, which includes the how they set up the point system.
- Updated the database plan, to also hold the lessons and the students lesson progression.
- Changed from an in game tutorial to highly detailed step-by-step walk through on the GitHub wiki.
- Updated the hint system to a text box with information that is shown at all times in the lesson.
- System interface updated with all the screens the teacher needs, since the teacher is seen as a primary user.
- Added the ability for the teacher to have the option to delete student accounts if needed.
- Updated the system security to use hashing to keep student accounts safe, and to add an OTP for teacher registering.
- Changed that lessons should be teaching the same concepts but are delivered differently to everyone having the exact same lesson, but can implement their code in their individual way.
- Updated the system life cycle sustainment by fully allowing teacher to add lessons and adjust lessons, which will allow the teacher to write as many lessons as they want.