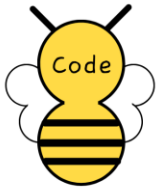
CodeBee



Project Goals

Brought to you by:

Shivaansh Prasann - 400100351

Yachen Wu - 400131465

Sida Wang - 400072157

Weidong Yang - 400065354

Jason Nagy - 400055130

Hongqing Cao - 400053625

Revision History	Date
Revision 0	Oct. 4th, 2020

Table of Contents

Project Description

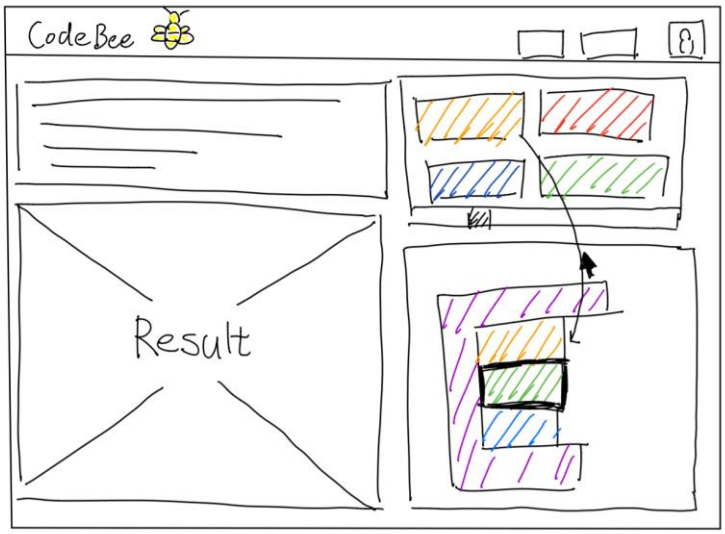
Background

Programming is rapidly becoming a necessary skill for people across multiple disciplines, and it is therefore important to start learning this skill from a young age. Unfortunately, beginners currently face a steep learning curve when it comes to learning the mindset required for programming. The amount of setup required in terms of installing software, along with minute details like language syntax and complicated error messages make it difficult for beginners to get up and running. This only widens the gap between the beginners and experienced programmers during the later years of life. The once "easy" languages like Python and JavaScript have evolved into powerful languages and are not as beginner friendly as they once were.

We have noticed that the approach towards educational content delivery across the world is changing and believe that the approach towards teaching programming needs to evolve as well. This is where CodeBee comes in.

Vision & Solution

CodeBee is an interactive web application which helps young students aged 10-16 learn and implement the basic concepts of programming in an interactive yet playful environment, without having to worry about unnecessary complications like syntax and setup. This method of learning aims to be more hands on, more interactive, more visual, more beginner friendly, and more entertaining. A potential layout is shown below:



Goals

Primary Goals

1. **The application should have an interactive GUI (Graphical User Interface).**
    a. Allow user to create program statements from code blocks.
    b. Provide a visualized live feedback of created statements.

    ### Selling Points

    I. The block-based GUI would allow users without technical experience to get rid of an existing “fear” of programming.
    II. A simplified syntax environment lets users focus on programming principles and logic.
    III. A web-based application would eliminate complex setup of editors, compilers, and tools.

2. **The application should offer a variety of topics to teach.**
    a. The application should offer a level selection page with various difficulty levels.
    b. Include basic concepts ranging from variables and precedence to more advanced topics like data types.

    ### Selling Points

    I. The application offers a reasonable learning curve.
    II. It also minimizes the gap between beginners and experienced programmers.

3. **User data can be saved and loaded.**
    a. The application should have a user authentication system.
    b. The application should be able to save and load the user’s progress.

    ### Selling Points

    I. Saving progress allows users to learn at their own pace.
    II. In case of hardware or network failure, the users would not have to repeat levels to attempt an advanced level.

4. **An in-game reward system to reward players for completing goals.**

    ### Selling Points

    I. A reward system would motivate users to complete the challenges presented to them.

Stretch Goals

1. **The application uses an in-house programming language and compiler.**
    a. The language does not have any complex features and libraries.

    ### Selling Points

    I. Such a language would be more beginner-friendly than “easy” languages like Python or JavaScript.

2. **A sandbox environment for users to use the available code blocks to program whatever they want.**

    ### Selling Points

    I. Users can experiment with the knowledge they have gained to understand concepts better.

3. **Let users to design their own levels.**
    a. Post user designed levels to workshop and share with others.

    ### Selling Points

    I. Having custom levels would allow the application to be used for examination and testing purposes as well.

Terminologies & Abbreviations

Python	An interpreted, high-level and general-purpose programming language.
JavaScript	A dynamic computer programming language.
GUI	Graphical User Interface.
Compiler	A computer program that translates computer code written in one programming language into another language.
Library	A collection of non-volatile resources used by computer programs, often for software development.
Sandbox	A mode of game that allows player to freely build without restriction.