Silvia Santamaria Buil

Introduction:

The Problem:

Coding has become an essential skill for everyday life, yet, unless you take an interest in computer science, it is a very difficult skill to learn. In order to make learning to code more accessible, I will create a programming-language learning app that makes learning to code easy by providing engaging and short lessons on coding from the user’s mobile device, so they can learn to code from anywhere.

My client is Mr Tibble, the head of computer science at my school, who would like the app made to ensure that all students have good coding skills regardless of whether they continue to take Computer Science at GCSE or A-Level. The target audience is everyone, as knowing how to code is essential to thrive in modern society. I feel older generations will benefit equally as much as younger generations from my app, as it will help to close the generational technology divide.

The Solution:

My solution to this problem will be an app, based on MFL learning apps, but for programming languages. It will involve gamified exercises that will encourage the user to learn more and spend more time on the app. It will include a streak feature, to encourage players to learn every day, a diamond feature, and a lives feature so that users try harder to get answers correct. They will also be able to compete against other players in weekly leaderboards and get followers/follow people and do ‘friends’ quests’ with them. The UI will be easy to understand so that users of all ages can feel welcome within the app.

The lessons that teach the languages will have many different types of exercises within, as well as a code editor so that the users can test their coding skills within the lessons in the app. I believe that this will help to solve the problem of people not knowing how to code as it will engage users and teach them coding languages.

My project will be limited to only having 1 full course in Python and another smaller HTML and CSS course. Due to memory constraints my app will also be limited to fewer players at first until more memory can be allocated.

Research:

Similar products:

Duolingo:

Duolingo is a mobile application and website that teaches languages (as well as maths and music in recent updates) through the ‘Duolingo teaching method’, which focuses on interactive, personalised and gamified lessons. The backend was originally written in Python, before being rewritten in Scala to improve performance. The frontend was then written in Swift for IOS and Java for Android. The session-generating algorithm uses predictions from their LLM ‘Birdbrain’, to generate exercises that are at an appropriate difficulty for the user, using a system similar to the ‘ELO-rating system’ used in chess. Furthermore, the topics covered in lessons also follow the spaced-repetition teaching algorithm to maximise memorisation.

With 97.6 million monthly users to date, Duolingo attracts users with its free services and bright and colourful UI, as well as its mascot, Duo the owl. After becoming popular in meme culture for the menacing notifications the mascot sends to encourage users to complete their lessons, the app’s social media has become very popular and played into the jokes. Duolingo is the main inspiration for my project, as it has begun to teach subjects other than languages, like maths and music, and so I would like to see if it would be an effective approach for learning programming languages!

Codecademy:

Primarily powered by Ruby and React, Codecademy teaches a huge amount of coding languages to the users of the platform, as well as other important computer science concepts, like cybersecurity and machine learning. Described as an ‘online coding bootcamp’, the website offers many free courses, but the majority are paid, requiring a subscription service to access them. Despite this, the free courses still provide great overviews of different languages, allowing them to be 100% self-studied, letting the user’s decide when and where and how they want to approach their own learning experiences.

Most popular for their web-editors, they provide interactive courses, allowing users to test out the code from their website as soon as they have learnt it. The courses involve a quick lesson page, introducing the key concepts, and then an area to code a task based on the content just learnt. There are also some quizzes when completing topics. For my project, I intend to use an in-app editor for some of the exercises I use, as well as a similar method of teaching the concept before jumping into the questions, since that method seems best for programming languages which can not just be translated into English like MFL languages in Duolingo.

Algorithms needed:

Spaced Repetition Optimization algorithm:

Spaced Repetition is a technique used to optimize memorisation. Study of certain concepts is scheduled with increasingly large gaps between revision of the concept to ensure that the concept is committed to memory. This is because, after learning a concept, memory of that concept will decay quickly until it is forgotten. If you revise that concept the next day, it will take longer for that concept to decay; if you revise that concept again a week later, it will take even longer for that concept to decay and so on. In order to help the memorization of different concepts in the programming-language that the user will be learning, I will implement a Spaced Repetition Optimization algorithm.

This will be done by modelling the decay and then computing the optimum time for future revision of the material. The information it will use to model this is:

* The user’s accuracy of this concept during review
* The time taken for the user to recall the concept during review
* How many times the user recalled the concept in reviews correctly in a row
* How long the current interval since the last review is
* How late is the user in reviewing the material vs when it was scheduled to be reviewed