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!

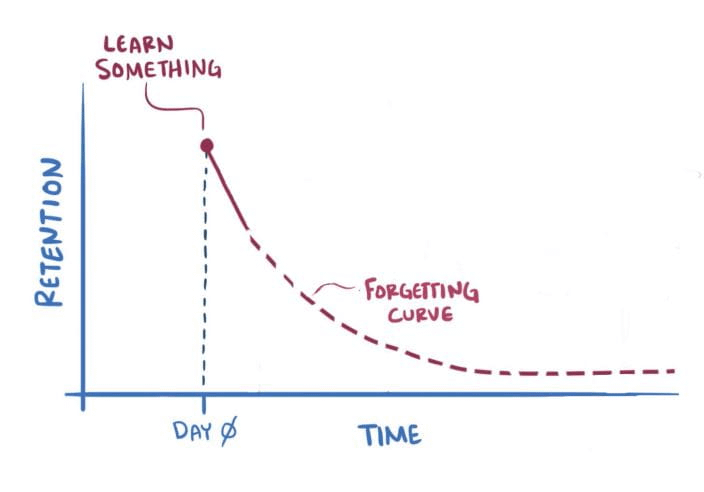
Codecademy:

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

A diagram of a graph

Description automatically generatedThe algorithm will then calculate the user’s ‘strength’ out of 100 for each concept using the above variables. These variables will be stored in a variety of databases storing data about the user used to optimise their app experience. Using this strength, the algorithm will then compute what date this material should be reviewed. This will be implemented through ‘review’ lessons that will pop-up, providing past mistakes or difficult content based on the concepts that need to be reviewed. The results from this lesson will alter the ‘strength’ for each concept, and thus the algorithm will compute the date when the concept needs to be reviewed once more.

Difficulty-ability rating algorithm using stochastic gradient descent:

A diagram of a normal distribution

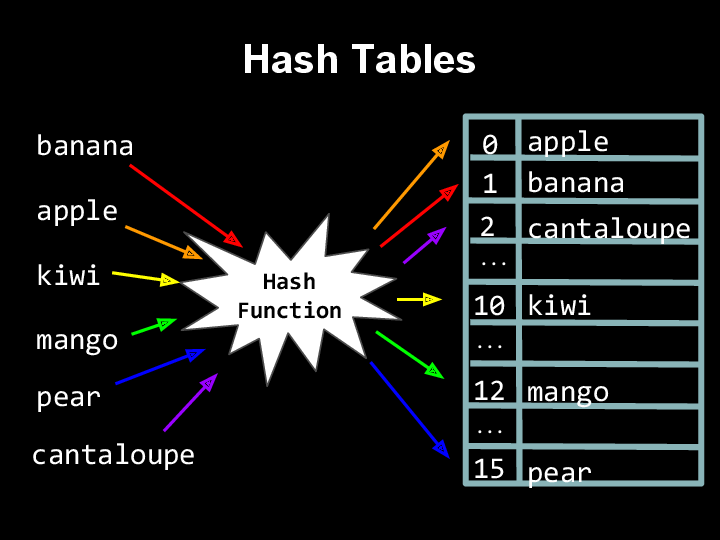
Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceThe ELO rating algorithm is a method for calculating the relative skill levels of players in games like chess or Esports. However, it has also been shown to be useful in machine learning algorithms. For example, in my program I will be using a stochastic gradient descent on 2 parameters: the difficulty of the exercise, and the ability of the user, for each exercise completed. This results in a generalisation of the ELO rating system, however, instead of having their ability increase when a player wins a game and the opponent’s decrease, my app will have the user’s ability decrease when they get an exercise wrong and the difficulty of the exercise increase. There can be dramatic changes in these ratings, caused by users with low ability ratings completing exercises with high difficulty ratings, but if a user with a high ability rating completes an exercise with a low difficulty rating, the ratings will not change much.

Stochastic gradient descent is an optimization algorithm that finds the model parameters corresponding to the best fit between the predicted and actual outputs. It is a form of gradient descent that is much faster than traditional gradient descent; traditional gradient descent uses all of the data, whereas stochastic gradient descent uses randomly selected very small samples of data (sometimes even only 1 piece) to calculate the line of best fit. The predicted difficulty values along this line then let the algorithm know which exercises to place next in the lesson for the us er to complete. Stochastic gradient descent is important to use in this situation instead of gradient descent as it will take much less time and therefore is a reasonable algorithm for calculating these predictions after every exercise.

Hash tables + Hashing:

A computer screen with text and numbers

Description automatically generatedHash tables are data structures that create maps between keys and values. They use a hashing algorithm to calculate the index a certain piece of data, or ‘key’, should be placed into in the table. This immensely speeds up the search for any item of data in the table as the ‘key’ can be put into the hashing algorithm to calculate its index, and then the data at that index in the table can be found immediately. Hashing algorithms are a calculation that takes place on the ‘key’ to provide the index of the ‘key’ in the table. Often hashing algorithms use one-way encryption, which increases the security of the table. Unfortunately, sometimes hashing algorithms can lead to collisions where the hashing algorithm produces the same index value for multiple different keys. These collisions can be handled using a variety of methods, such as rehashing, linear probing, and sep arate chaining. Separate chaining is the best of these methods, inserting all elements that hash into the same slot index into a linked list. This, however, leads to another list that may have to be search through, and some insecurity around the key. A way to reduce the time taken to search through the second list, it could be sorted every time another value is appended to it and then a binary search could be used to find the correct value.

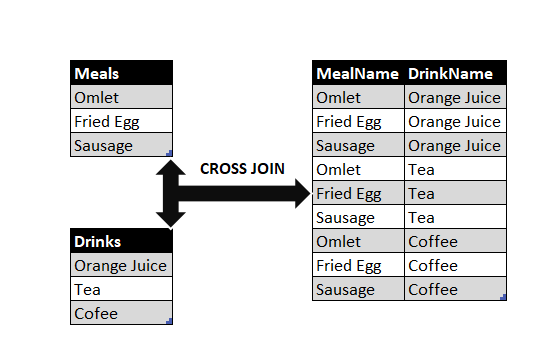
In my project, I intend to use hashing to store the passwords of the users of my application, storing the hashed value in a table with the username. This code is an example of a hashing algorithm I could use, as well as a simple hash table created from a static 1D array, however, as my project progresses, I will use a class to model my hash table instead.

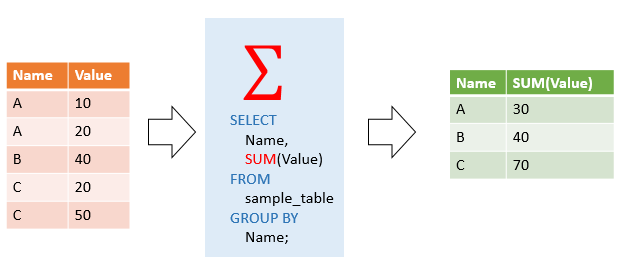
A screen shot of a computer

Description automatically generatedCross-Table parameterised SQL and Aggregate functions:

A screenshot of a computer

Description automatically generatedA computer code with text

Description automatically generated with medium confidenceCross-tabulation organises a dataset in table format. The data is organised by specific table headings, and then only shows 1 type of data for these specific table headings. Aggregate functions perform calculations on values, like SUM(), and output a singular value.

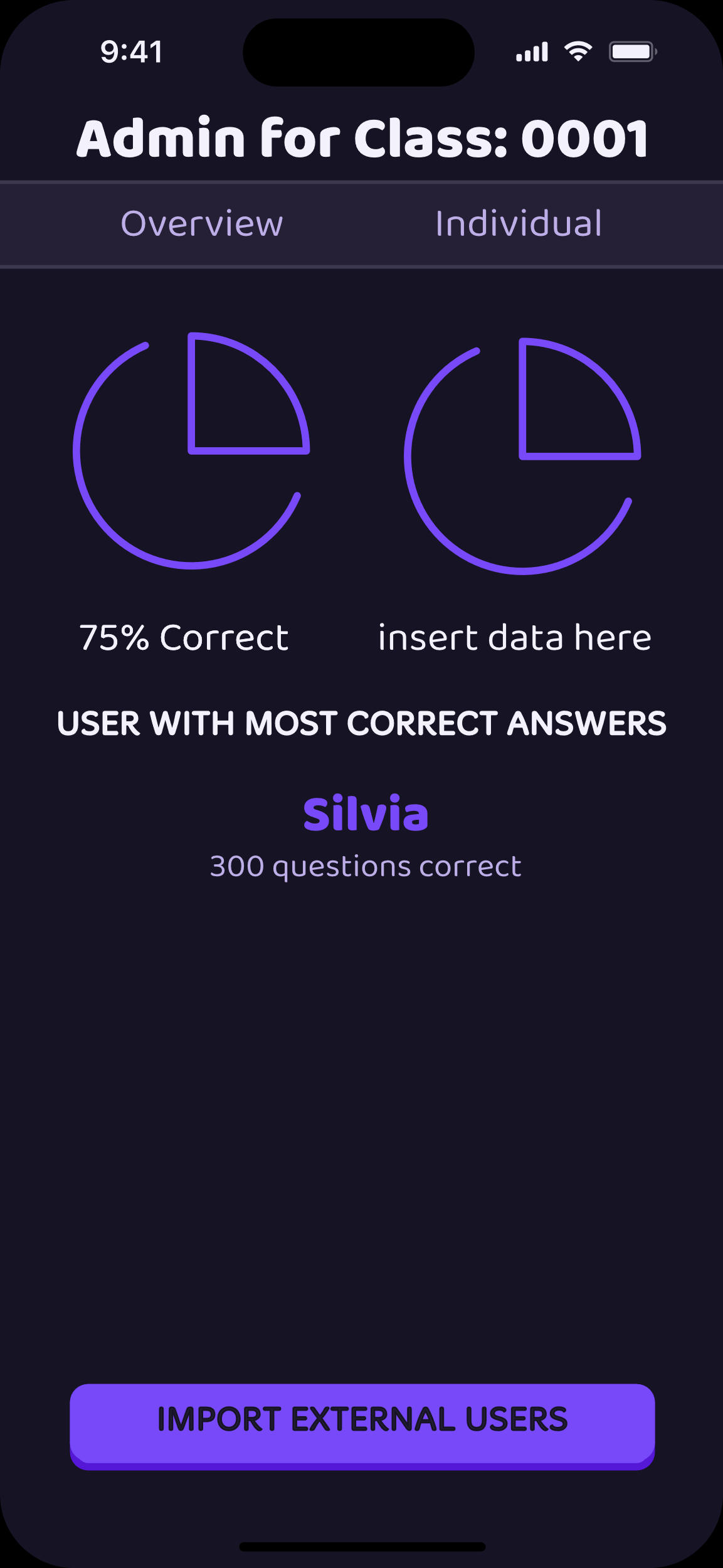
In the example shown, I created a small example table called ‘languages’ to show the exp that each user collected. By using a ‘CASE’ method, the table has been organised by UsernameID. This is the specific table heading organising the table. The 1 type of data shown for this heading is the amount of exp earned, grouped by the language they earned this exp in, as well as the total exp in the last column of the table.

In my project, this method will be used to calculate the total exp points for each user: by week for the weekly leaderboard, by language course and in total.

Complex user-defined algorithms:

The complex user-defined algorithms I will be using are the spaced-repetition optimisation algorithm and the difficulty-ability rating algorithm using stochastic gradient descent. They will be used to optimise the user’s experience within the app and to optimise their learning.

Client information:

My Client, Mr Tibble, has told me he would also like to be able to import users from an external site he has with existing data about the users. To do this, I will be making an admin page for myself and teachers to use in order to be able to set up user accounts and access information about them. The admin page will contain an ‘external user’ button which, upon being clicked, will have the teacher upload a text file and then will read contents from the text file to populate the fields in the user database along with some test results they already have to allow them to start with some experience points and for the algorithms to adjust to their skill levels. I will also add a ‘class’ feature, so that teachers can only see the results of the users in their class, which will be operated by having a teacher login and input a specific ‘class code’ to be able to see the admin page for their class.

On this admin page, statistics about the users will be displayed, including the top 3 hardest questions they have answered, the question they have answered the most, the user that has answered the most questions in the class, and the average percentage of correct answers in that class. It will also highlight some topics that are particularly difficult for the class so that the teacher can focus on these in class. Other statistics will also be added to this page during development. This page will be updated as the app is used so that whenever the teacher logs in to the admin page the results are up to date.

Survey:

Aim:

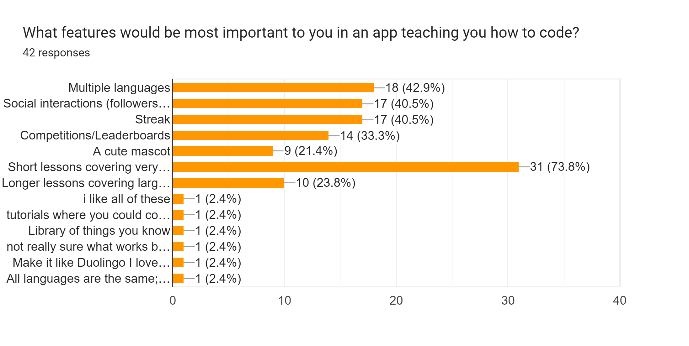
To find out about the experiences of potential users regarding learning to code, I created a survey. This survey was sent out to people in my computing class, people at my school in general, international computing students, professional programmers and other acquaintances who do not know how to code. I got responses from a wide range of ages and genders. I also wanted to know the opinions of users with other learning applications or courses in general to see what I could do better. I wanted to know about what they wanted to see in my application and what features they would want in the user interface.

Results:

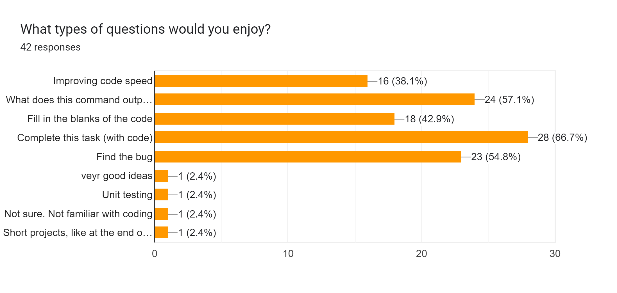
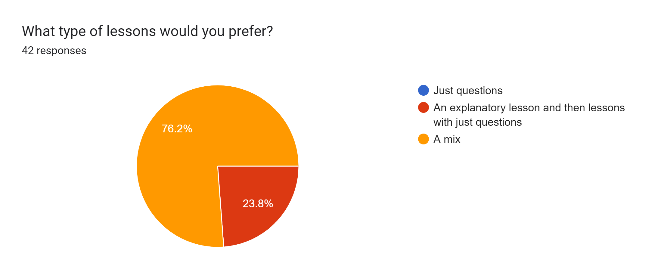
A colorful pie chart with text with Crust in the background

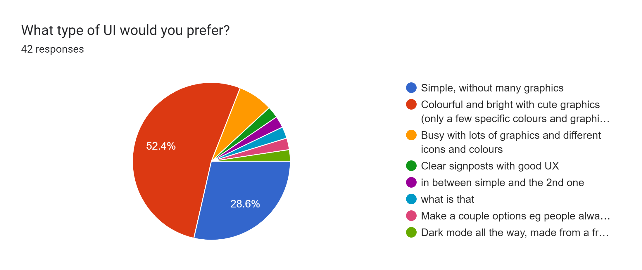
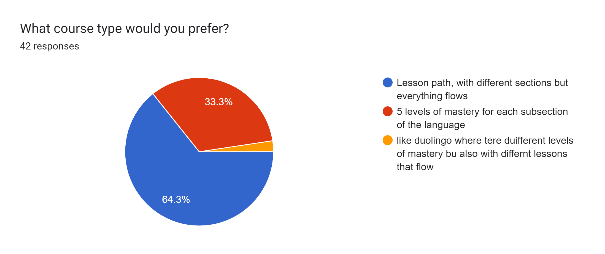
Description automatically generatedI received 44 responses in total, with 63.6% of responses being Female, and with a large range of Ages responding, allowing me to truly configure what I should include in my app for it to be enjoyable for everyone to use.

I also found it quite interesting to see just how many languages the people who completed my survey knew, and from this I deduced what the most popular languages were and therefore which languages I should teach in my app, Python and HTML/CSS.

A colorful pie chart with white text

Description automatically generatedI was surprised by the difference in ease of coding between those who knew how to code and those who didn’t, however, I feel this may be explained by how intimidating it seems before you know how to code, in comparison to how difficult it actually is once you have learnt the language.

Most liked how current code-learning platforms are easy to access and are very well documented, however many disliked the paywalls behind these platforms and noted a lack of engaging or personalised content. After asking what features would be most important in my app to make it engaging, I found that short lessons covering very small parts of the language each time seemed very popular, as well as having the option to learn multiple different languages, probably for those who are already experienced coders but would like to keep expanding their knowledge. Features from Duolingo like the streak feature and having friends were also popular, as well as a cute mascot. When asking about the specific exercises within lessons, the most popular type was completing tasks with code, consistent with the project-based learning method many people use. A suggestion was a short project at the end of each unit, which I will definitely try to apply. The responses also liked exercises based around finding bugs and stating what commands do. Another interesting suggestion was Unit testing, which inspired me to create an exercise type based around trying to break algorithms and how to prevent this. Considering the lessons as a whole, people definitely wanted explanatory lessons, but preferred when they were integrated with questions simultaneously, and so I will try to implement this by perhaps having a starting page explaining a concept followed by questions.

Responses suggested that people mainly enjoyed minimalistic user interfaces, but preferred those that were colourful to those that were monotonous. They also suggested having a switch for dark mode, which I will be implementing so that more users will feel welcome on my app. The most popular colour schemes included blue, as well as some other cheerful colours like green, purple and pink. I will try to integrate these colours into my user interface. The most popular mascot types were animals, although robotic creatures were also quite popular. To appease both sides, I will be using a snake mascot, inspired by Python, with some cyborg-like elements.

The final question covered the overall course path for each language, and lesson paths with different sections were most popular.

A cartoon character with a blue face

Description automatically generated with medium confidencePrototype:

I have created a simple mock-up of the home screen for my app. It shows the first section and unit of the Python course, covering ‘simple commands’ like ‘print’ and ‘input’, as the survey responses showed that bright colours with minimalistic design are most favoured, I used lots of bright colours and 1 large graphic. Blue was the main colour scheme for the lesson path as that was the most popular colour scheme. The robot is an example of another character that may be featured in lessons, not the mascot specifically. The design is heavily inspired by Duolingo, but with some differences, like the review lesson with the arrows.

If the grey circles are lessons that the user has not unlocked yet and wouldn’t be able to be clicked. The progress bar surrounding the blue circle demonstrates how many lessons the user has completed for that command, and how many are left. The robot would be moving in the actual application, cheering the user on. The numbers by the fire symbols, diamond symbol and heart symbol would change based on the hearts, streak and diamonds the user has.

This is an example of the light mode version of the app, however there would be an option to change it into dark mode, inverting some of the icons’ colours.

Objectives:

Aesthetic:

1. The loading screen will feature a snake (the mascot of the app).
2. The overall colour scheme of the app is purple.
3. The first page will show a picture of the mascot, the name of the app and the slogan. At the bottom of the page there will be 2 buttons, one marked ‘login’, taking you to the login page, and the other marked ‘create account’, taking you to the account creation page.
4. On the account creation page there will be a title at the top saying ‘create your account’, and 3 rectangles to type in the email, the password and to confirm their password. At the bottom of the page will be a ‘create account’ button, taking them to the next page.
5. On this page there will be a title at the top saying ‘fill in your details’, and 5 rectangles asking for the name, username, gender, date of birth and phone number of the user. At the bottom of the page there will be a rectangle marked ‘get started’ to take them to the ‘new user’ page and exercises, and a ‘teacher account’ rectangle, that will take them to the admin page, with a classroom code automatically generated for them and presented at the top of the screen.
6. The log in screen will be simple, just showing 2 rectangles to type your username and password in, and a small rectangle marked ‘log in’ that will take them to the next page. Underneath this rectangle will be a small rectangle marked ‘teacher login’ that will take them to a separate page.
7. On the teacher login page there will be another small rectangle to type in the class code, and a small rectangle marked ‘log in’ that will take them to the admin page.
8. On the normal page there will be a header that shows the number of hearts, streak, language and gems of the player.
9. There will be a footer with different tabs: home, quests, leaderboard, profile and feed,
10. On the home page (marked by a small house icon) there will be lots of circles centred but heading downwards.
11. To the side of the circles there will be graphics depicting robots, snakes, or your avatar.
12. On entering the quests tab (marked by a small chest icon) there will be a list of 3 daily quests, with progress bars to show how much is left to complete them, and a friends quest, showing the avatars of both people and a progress bar with different colours to show how much each friend has completed.
13. On entering the leaderboard tab (marked by a small shield icon) there will be a header showing the league the player is in (a small shield icon in a specific colour for each league) and how much time is left for the weekly league.
14. Below the header there is a numbered list of league participants, ordered by descending experience points for that week.
15. On entering the friends tab (marked by a small snake icon), the header shows the user’s avatar as well as the number of followers they have and how many people the follow and the languages they are learning.
16. Below the header the total streak of the user and their total experience points will be shown, as well as the current league they are in and how many times they have finished in the top 3 of their league.
17. Below this there will be some achievements the user can get by completing a certain number of exercises or tasks, represented by a small graphic for each achievement.
18. On clicking the user’s avatar, they enter an avatar creator, with different tabs to change the body type, skin colour, hair style, eye colour, add any facial accessories or facial hair, and to add any headwear.
19. In the right hand corner of the screen there will be a little gear icon that will take the user to their settings, where they can change their username and password or any other settings.
20. On entering the feed tab (marked by a small bell icon), a set of rectangles will appear, each featuring a milestone or achievement that either the user or someone they follow has achieved, as well as how many people have celebrated this achievement.
21. On entering the shop (marked by a gem icon in the header), some chests with exp boosts will be featured as well streak freezes that can be bought to preserve the users streak.
22. On clicking the hearts (marked by a heart icon in the header), a small drop down menu will be presented with 2 small rectangles, one selling a heart refill for a certain number of gems, and the other indicating that the user can do practice lessons to earn hearts.
23. On the admin page, there will be a header at the top stating ‘Admin page: (class code)’.
24. Underneath this, there will be 2 tabs, one titled ‘overview’, and the other titled ‘individual’.
25. On the ‘overview’ tab, statistics and graphs will be displayed relating to the class as a whole, such as a pie chart for class percentage correct and incorrect, and a list of the hardest topics for the class.
26. On the ‘individual’ tab, statistics and graphs will be displayed relating to the student specifically, such as the amount of questions they have done and the hardest questions they have answered.
27. At the bottom of both pages, there will be a small rectangular button stating ‘import external users’ that upon being clicked, will allow the teacher to import users from their class and set up their accounts for them.

Input:

1. The user will not be able to click anything on the loading screen.
2. To create an account, the user can type in their details using their keyboard, and click the buttons to continue to the next pages.
3. To log in, the user can type their username and password using their keyboard, and then click the button marked ‘log in’ to log in.
4. On the teacher login page, the teacher can type in their classroom code and press enter to go to the admin page.
5. The user can click on any circle on the home screen.
   1. This will cause a small drop down menu to be presented, showing the lesson name and presenting a button the user can click to start the lesson.
   2. If the circle is grey, the user will not be able to click the button to start the lesson, but they can read about the lesson.
6. In the lesson, there are certain buttons the user can click.
   1. The user can not click on anything other than a small green button at the bottom of the screen that will turn green when clickable (the page is over) on the explanatory page.
   2. In the ‘what does this command do/output’ exercise, the user will have 3 options to choose from, and can click on the 3 options and then click a green button at the bottom of the screen to secure their answer.
   3. On submitting an answer in any exercise, a small menu will pop up saying whether they were incorrect or correct, and they can then click another button on the menu to go to the next exercise.
   4. In the ‘fill in the blanks’ exercise, the user will have some options at the bottom of the screen, and can then click and drag these options into any gaps in the code shown, then clicking the small green button at the bottom of the screen to submit their answer once all the gaps have been filled.
   5. In the ‘find the bug’ exercise, the user will be presented with some code and then 3 options at the bottom of the page stating lines in the code where there may be a bug.
   6. After correctly clicking and submitting the line number, the options will be replaced by 3 more options showing the specific command that has an error in it, they can then submit this answer by clicking on a small green button at the bottom of the screen.
   7. In the ‘complete the task’ exercise, the user will be able to type their code into a box and then click a small green rectangle at the bottom of the screen to submit their answer.
      1. If the answer is incorrect, they will be able to fix their code as many times as it takes before they get the answer right.
   8. In the ‘improve the speed’ exercise, the user will be presented with a section of code and a rectangle below to retype the code in a way that will execute faster.
   9. In the ‘testing’ exercise, the user will be presented with some code which will run, they will then be able to type an input which aims to break the code.
   10. After breaking the code they will be asked how to improve the code, with 3 options to choose from, clicking one and then the small green rectangle at the bottom of the screen to submit their answer.
7. The user can click any of the icons in the header or footer.
8. If, upon clicking the heart icon, the user clicks the practice button, they will enter another lesson.
9. If they click the language icon, they can add another language or change the language they are learning.
10. Upon clicking one of the chests in the shop, they will activate an exp boost.
11. In the feed, they can click a ‘celebrate’ button in the bottom left corner of each rectangle to send a celebratory message to the person who completed the achievement.
12. On their profile, they can click on the number of followers they have or the number of people they follow to see the people, and they can click on these people to see their profiles.
13. By clicking the ‘add friends’ button, they can search people up, or simply click on the recommended people they should follow to follow them.
14. On the admin page the user can click on the graphs to make them larger and be able to see more information about them. They can click on the different tabs on the header to change the tab.
15. On the individual tab there will be another header below the two tabs with two arrows and a username between them. On clicking the arrows the teacher can go to the next or previous student and they can look at each student.
16. On clicking the top 3 hardest questions, the teacher will be taken to a preview of the question that they pressed.
17. On clicking the student who has answered the most questions on the ‘overview’ tab, the teacher will be taken to the ‘individual’ tab, to the student’s page.
18. On clicking the ‘import external users’ button, a window will pop up which the user can drag a text file into to import the users’ data.

Output:

1. If the user gets an exercise wrong, the pop-up menu will tell them either the correct answer or what they got wrong.
2. After completing a lesson, a screen will output the % of correct answers, and the total exp received, the time taken on the lesson, as well as a motivating message.
3. After clicking on a chest in the shop, the program will output how long the exp boost will last for.
4. After doing the first lesson of the day, the program will output how long the user’s streak now is.
5. On clicking the ‘add friends’ button in the profile, a list of recommended friends to add will appear, as well as a search button to search up a specific person to follow.
6. If the user types in an incorrect username or password, the program will tell them to try again, making the boxes red.
7. If the user types in an email/username that is already being used on the create account page, the program will tell them that the username/email is taken, and they should input another one. The box will be marked red, and the text will be in red below the rectangles.
8. If the ‘confirm password’ and ‘password’ rectangles are not the same, the program will prompt the user to make them match.
9. On the admin page, when the teacher has imported the external users, the program will present the teacher with a table showing all of the users and any information that they will need to fill in/change. All of the cells can be edited and changed by the teacher and upon clicking the button under the table saying, ‘confirm setup’, the accounts will be created.
10. On the admin page, every time the data changes, the statistics on the page will change automatically, and the graphs will change to adjust to this.

Processing:

1. The program will try to match the user’s username to its password and data.
   1. It will first check through the databases for the input username.
   2. If found, it will generate a hash of the input password.
   3. It will try to match this hash with the respective hash found in the password hash table, and check if the passwords match.
   4. If they do, it will log in the user.
   5. Then, using the user’s username as the primary id, it will load their data stored in all the other databases.
2. When a new account is made, the account database will be updated with this data.
   1. The tables will be updated with the details the user entered.
   2. The hash table will have the new user’s password hashed into it.
   3. If the email and password will be entered first.
      1. The ‘password’ and ‘confirm password’ inputs will be compared to ensure they match.
      2. The email will be searched for in the account tables to see if it is already taken.
      3. If not, a new record will be made in the tables storing the email, with empty cells where the remaining details will be placed.
   4. Then the personal details will be entered.
      1. The username will be searched for in the account tables to see if it is already taken.
      2. If not, the remaining cells will be filled with the details and a new account will have been made.
      3. A new record will be made in all tables, to be able to be filled with results from lessons.
   5. If the user does not finish creating their account, with only the email and password being entered, the record will be deleted after an hour.
3. As they complete exercises, the difficulty-ability rating algorithm and the spaced-repetition optimization algorithm will compute results from the data collected.
   1. It will use the time taken to complete each exercise and the accuracy in the exercises to predict the next exercise and when they should revise the concepts currently being covered.
   2. Once the predictions are released, the application will be updated, and the new exercise will be loaded.
   3. The review will also be scheduled for a certain time, and then the review exercise circle will appear in the lesson path.
4. During the lessons, the user’s data is automatically updated in the background.
   1. If it is their first lesson of the day, their streak will be increased by one.
      1. This will be done by using a flag for whether a lesson has been completed that day in a database storing daily data.
      2. These will then be summed using an aggregate function to produce the daily streak.
      3. If they do not do their streak one day and they do not have a streak freeze, this flag will return false and then all of the daily data will be deleted to be restarted.
   2. Their position in the league will be automatically updated with their exp points after they complete a lesson.
      1. This will be done using cross tables to sum their weekly exp after each lesson.
   3. Quest completion will be automatically updated once the quests are completed.
      1. The quests will be stored in a database and upon updates, the progress bar will be updated and presented after the lesson.
      2. Each quest will have an assigned reward which will be granted upon completion.
   4. Lives are updated every time a mistake is made during the lesson, decreasing by 1.
   5. If a certain achievement is reached, the feed will automatically be updated with the message.
5. If the user does not complete a lesson during the day but has a streak freeze activated, their streak will be preserved.
   1. This is done by checking if there is a streak freeze activated at 00:00 the next day.
   2. If so, the flag will be set to true, however the sum will not be affected, the streak will remain the same.
6. Within the avatar creator, upon saving the avatar, this will be updated throughout the app.
   1. It will be updated on the leaderboard and within lessons.
7. Users who join the league at similar times who gain similar amounts of exp weekly will be put in leagues together.
   1. This will be done using an algorithm that will match players to a league.
8. Quests are generated randomly from a database of preset quests.
   1. Each daily quest has a set difficulty and therefore an increasing reward for the harder quests.
   2. The friend quest partner is generated randomly from the friends list.
      1. The friends list contains people who follow each other.

Modelling:

A blue rectangular shapes on a white background

Description automatically generatedA blue rectangular shapes with white text

Description automatically generated with medium confidenceTo help the design process, I used a top-down design approach to create a hierarchy diagram depicting everything I believe I will need to create during my app’s development. This decomposition will make it easier to tackle individual parts of my app as I can make separate modules/functions for each box. Blue boxes contain UI modules, Green boxes contain Account modules and Purple boxes contain Lesson modules. Orange boxes contain algorithms.

A screenshot of a computer screen

Description automatically generatedA diagram of a chart

Description automatically generated with medium confidence

1. A purple and orange rectangular structure

   Description automatically generated with medium confidenceA diagram of a company

   Description automatically generated

A purple rectangular object with white text

Description automatically generated