Scenario 1: Technical Document

Kang Shen, Sahil Gaikwad, Helena Tai, Aryan Jain

10 February 2023

Contents

1 Context

1.1 Introduction

For this coursework, our group is developing a smartphone application that incorporates user intuition and interaction for teaching calculus. Calculus is arguably one of the most important branches in the field of mathematics. Initially invented in the 17th century by Isaac Newton and Gottfried Leibnitz (Coolman, R. 2015), calculus is applied in a lot of sectors, which makes it important to understand the fundamentals properly. Calculus is split into two major branches, differential calculus and integral calculus. Applications of differential calculus include the calculation of velocity and acceleration as well as the calculation of rates of change. Applications of integral calculus include computations involving area, volume and the centre of mass.

1.2 Target Audience

Since calculus is first taught at an advanced level in A-Level Mathematics, our group has jointly agreed that our target audience is A-Level students (year 12 and year 13) who are pursuing A-Level Mathematics (and Further Mathematics). Our main intention when we chose this target audience, was to make a teaching tool for teachers and students, which can help strengthen their core knowledge about calculus. Calculus is an essential part of A-Level Mathematics and Further Mathematics because it is the backbone of several crucial topics such as the Taylor Series (or just Maclaurin series) and orders of differentiation (e.g. first-order differentiation and second-order differentiation). Even at university level, there is a whole plethora of subjects that depend on calculus (e.g. Biology where it is used for various statistical analyses, or Engineering where it is used mathematically to help solve various real-world engineering problems). Therefore we deem it important for students to grasp this branch of mathematics.

With advancements in portable technology, all sections of the population demographic have expressed reliance on smartphones in this generation. Arguably the most avid consumers of smartphones are students aged 13 to 20. According to YPulse (2022), 79% of 13 to 20-year-olds say that they cannot survive without a smartphone (YPulse 2022). Since our target audience falls under the 13 to 20 years demographic, it is important that we accommodate the current needs while supporting them in education. This explains our decision to develop a smartphone application, which would allow students to access practice materials anytime and anywhere.

2 Approach

2.1 Content

The topics that would be covered in our tool will be some basics of differential and integral calculus, according to the A-Level Mathematics syllabus, such as:

1. First and Second Derivatives
2. Tangents, Gradients and Normals
3. Introduction to Integration
4. Area under the Curve

Students would have the option to select the mode of the practice exercise according to their preferences, which are multiple-choice questions or fill-in-the-blank questions. There would also be an option for the difficulty levels: easy, medium or hard.

For “Area under the Curve” particularly, students will be required to shade in the area under a given graph, to demonstrate their knowledge of integration bounds (upper and lower).

2.2 Use of Language

Since we are doing Android application development for this project, our team has chosen to use Java as the language for development. Even though Android apps can be developed using a variety of languages such as Kotlin, C++, Python, or Swift, we made the decision to use Java for a variety of reasons (Prabhu, J. 2020):

1. It is an object-oriented programming language meaning it allows for code to be more modular (which makes testing and debugging easier in a team environment), and reusability of code without having to repeat code lines, which allows for a cleaner presentation of code and reduces clutter.

2. Java is one of the oldest languages so it has extensive community support and available help, which is always a positive factor when considering a language for team development, because it reduces reliance on language/library documentation alone for help, since there is community support available.

3. It is a more secure language than others because it uses features such as security manager to ensure that no one can access code without the right authority (Kalwan, A. 2019)

4. Java is highly resourceful for android development than other languages such as Python. This makes android app development easier because many UX and functionality features that are necessary for a good app are available to use through Java and has immense support for it.

2.3 UX and Functionality Features

The User Experience (UX) of an app is a critical aspect of its design and development. It refers to the overall feel and interactivity of the app, and it can make or break the user's experience (Mor-Samuels, E. 2023). With millions of apps available, users have high expectations when it comes to the quality of an app's UX. In today's fast-paced digital world, users have limited time and attention, and they tend to only use a few apps regularly. On average, a smartphone user interacts with only 5 apps in heavy use on a daily basis (Perez, S. 2015), which highlights the importance of having a well-designed UX for an app.

A good UX design should be intuitive, straightforward, and efficient, allowing the user to perform their desired tasks quickly and effectively. The design should also be aesthetically pleasing, with a consistent and visually appealing layout and colour scheme. The goal of an app's UX design is to make the user feel comfortable and confident when using the app, encouraging them to spend more time interacting with it.

For the app that we are building, these are some of the proposed features to enhance the user's experience:

1. User Authentication: User authentication ensures the safety and privacy of any given user, however in the context of educational tools, user authentication allows access to the history of the user on the app, so we can provide relevant analytics such as leaderboard scores, right versus wrong ratio, and recommendations.

2. Leaderboard ranking: Given that this application will have multiple users, we want to ensure that there is an incentive for students to join our app, and use it. Therefore, we are proposing to build a system where users can compete based on how many correct answers they have given, and give relevant leaderboard rankings.

3. Animations for guidance: Using visual cues such as animations or text transitions, we can improve the quality of teaching students core concepts of calculus.

4. Options for difficulty level: To maintain the user intuition of the app, the content must be doable for everyone (where each student fits in a wide spectrum of intelligence), therefore we propose different difficulty levels where questions for the user can be user-filtered by difficulty.

By incorporating these features, we aim to provide an outstanding user experience that will increase user engagement and retention.

2.4 Algorithms with Data Structures

To provide our UX features and general functionality, we are proposing the following algorithms and data structures, which will be designed to boost efficiency in terms of time complexity and space complexity.

2.4.1 Authentication

In order to have a user authentication system, there needs to be a robust backend system that handles all queries quickly and efficiently. In order to fulfil this, we are proposing to build an API that stores authentication usernames and passwords, alongside the respective user's leaderboard rankings. These are the following algorithms we propose:

1. API creation and access: This algorithm automatically establishes connection with the API (and reestablishes connection if it is lost during user interaction). Since we are planning to use HTTP as our protocol, GET and POST requests will be sent to retrieve or send data elements to the API (for authentication or storage of credentials).

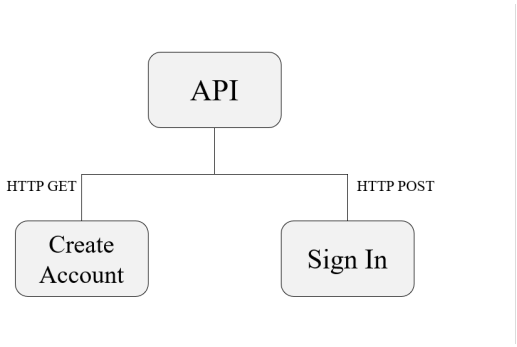


Figure 1: Visual representation of API use

In terms of the contents of the header packet for POST, this will be the proposed layout in JSON format:

```
POST /authenticate,
Content-Type: application/json
{
    "username": "hashed username",
    "password": "hashed password"
}
```

For this project, we are planning to build a REST API that will use these endpoints (specific locations for retrieving or altering data):

- Register: For user registration, where the client can send a POST request with the user's details to this endpoint to create a new user account.
- Login: For user login, where the client can send a POST request with the user's credentials to this endpoint to receive a token for authentication.
- User ID: For retrieving user details, where the client can send a GET request to this endpoint with the user's ID to receive the user's details.
- Leaderboard: For retrieving the leaderboard, where the client can send a GET request to this endpoint to receive a list of users ranked by score.
- Score: For updating a user's score, where the client can send a PUT request to this endpoint with the user's ID and the new score.

In addition to the client-side implementation, we will also implement the server architecture necessary to support the REST API. This will include a server framework, and a database management system. In terms of the server framework, we are planning to use the Spark library because It is well-suited for the server-side implementation of a REST API due to its in-memory processing, distributed computing, high-level Java API, and scalability (Bennett, R. 2021).

2. Hashing algorithm: Hashing algorithms ensure the security of credentials because they are one-way functions, meaning that once the password is hashed, it cannot be easily reversed to retrieve the original password. This protects the passwords from theft or unauthorised access. For this project, we will implement the Bcrypt hashing algorithm (Sewell, M. 2021) which is a commonly used hashing algorithm due to its resilience against brute-force attacks (as it adds computation power each time it runs, preventing attackers from trying thousands of combinations at one time).

   To make the authentication experience more secure, we plan to implement salting. Salting is a process used in cryptography to add random data to a password before it is hashed. The purpose of salting a password is to make it more difficult to crack the password if the database of hashed passwords is compromised. When a password is salted, a random value, known as a salt, is generated and then combined with the password before it is hashed. The salt is then stored along with the hashed password. When the user provides their password for authentication, the same salt value is used to hash the provided password, and the resulting hash is compared to the stored hash.

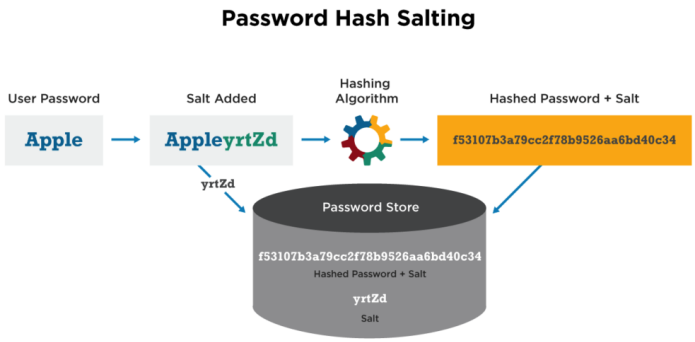


Figure 2: Visual representation of how hashing works

2.4.2 Random question generation

To generate random calculus questions, we first identify the main topics to cover, such as limits, derivatives, and integrals. Next, we create a database of question templates, which include placeholders for variables such as function names, limits, and constants. For instance, a template for a question about limits might be: "Evaluate the limit of the function function as variable approaches value." The questions will be generated by randomly selecting a template from the database and replacing the placeholders with randomly generated values. To vary the difficulty of the questions, we will set constraints on the values of the variables, such as the range of the function names or the limit value. Additionally, we will ensure that the generated questions are solvable by carefully choosing the values for the placeholders (i.e. placeholder values will be chosen from a predefined set of numbers which ensure problems are solvable within the chosen difficulty level). The algorithm for generating these questions will be a crucial part of the app and will require careful consideration to ensure that it is both efficient and effective in producing a wide range of solvable calculus questions.

2.4.3 Parsing user input

As part of our UI design, we are planning to implement a built-in calculator, that gives users access to characters that are not available on a typical android keyboard (e.g. $\frac{\mathrm{d}y}{\mathrm{d}x}$ and $\frac{\mathrm{d}^2y}{\mathrm{d}x^2}$, $\int_a^b f(x)\,dx$). However, even by providing this calculator feature, we need to ensure that we can parse it correctly and convert it to a format where we can do the mathematical calculation (to provide the solution). Parsing, or syntax analysis, is the process of analysing a string of symbols, either in natural language, computer languages or data structures, to determine their structure and meaning (in our case, in the context of mathematical expressions).

The method that we are planning to use is recursive descent parsing.(anonymous007 2022) Recursive descent parsing is a type of top-down parsing (traversing a tree from top to bottom) which involves recursive traversals to analyse a sequence of tokens and determine if it adheres to a given grammar. It consists of mutually recursive procedures (elements that call each other in order to parse the entire user input).

Additional processes such as backpropagation (also known as backtracking) are used to traverse and find alternate procedures if the current procedure fails (i.e. if user input doesn't pass grammar rules).

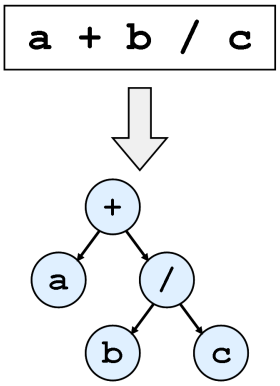


Figure 3: Visual representation of a parsing tree

Recursive descent parsing is a good choice for parsing user inputs in the context of calculus for several reasons:

- Simplicity: Recursive descent parsing is a simple and straightforward parsing technique that is easy to understand and implement. This makes it an attractive option for parsing user inputs, as it can be relatively straightforward to write a parser that correctly handles the grammar of a typical calculus expression.
- Flexibility: Recursive descent parsing is a highly flexible parsing technique that can be adapted to handle a wide range of input formats. This is particularly useful in the context of calculus, where user inputs can vary widely in terms of their format and syntax.
- Robustness: Recursive descent parsing is a robust parsing technique that is capable of handling a wide range of errors in user inputs. This is important in the context of calculus, where users may make mistakes or enter expressions in unexpected ways.

2.4.4 Leaderboard implementation

In this project, we are also aiming to implement a leaderboard that ranks all the users based on the number of questions they have solved. To achieve that we need several basic operations: insert(record), update(record), search(name), search(rank). We first considered using a priority queue with linked list implementation, however that approach takes O(N) time complexity for both insert and search operations, as we will need to traverse through the entire list until a proper position is found. Then we considered using a priority queue implemented with AVL tree which takes O(log N) for insert and search operations, and hence we can achieve better performance in average case, and also guaranteed performance in the worst case.

Additionally, implementing a leaderboard using an AVL tree has other benefits as well. With the AVL tree, we can easily maintain the balance of the tree, which ensures that the height of the tree remains small and the insert and search operations can be performed efficiently even when the number of users on the leaderboard grows very large (sidhijain 2022).

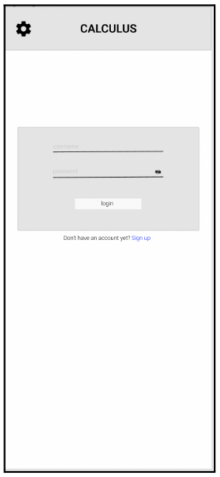
However the downside of using an AVL in a REST API, is that it is unconventional and fairly hard to do. To implement an AVL tree in a REST API, you will have to represent each node in the tree as an endpoint in the API which can be only accessed by HTTP requests. Alternatively, we can use a sObject Tree in REST which is a tree like structure to organise similar data elements in a nested manner.

2.5 Features in/out of the scope of the project

Given the time limitations of this project, it is practically impossible to include all our proposed functionality without compromises, which can result in a lower quality product. While we are actively trying to include many features in this project, such as leaderboard, backend API, random question generations, user input parsing, etc., there are some features that are outside of the scope of this project. One

such feature is creating a "Teaching" page where we plan to provide animations, text transitions, and many other intuitive cues, so that the user can learn the respective topic before answering questions on it.However given the spectrum of features that we are already planning to implement, we believe that the "Teaching" page will be the least of our priorities for this project. Another feature that we are uncertain about is parsing user input. Since parsing is a difficult feature to implement (especially with trees), we are planning on developing a simpler feature for parsing, such as comparing user input values to all possible combinations of solutions (equality operator comparison). This will reduce the complexity of the feature and allow us to focus resources on other development features.

2.6 Design



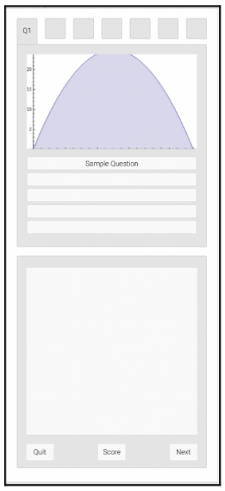
This is the page (**Login Page**) for user authentication. The UI provides entry boxes for the user to insert their authentication credentials (which will be checked using their hashed values against a database of existing credentials). Once the user signs in they are redirected to the **Home** page.

The **home page** provides a variety of functionalities:

- User analytics: This section of the UI showcases the user statistics and their respective position among other players that use this app.
- Random questions: This option redirects the user to the **Test page**, where the user will be given an assortment of random questions (combination of all question topics). The user is allowed to choose difficulty level of the questions
- Questions by topic: This allows the user to choose specific topics that they want to be quizzed on. This option redirects the user to **Questions by topic** page
- Create questions: This functionality is aimed at teachers/ students who want to make a revision tool for themselves. This allows the user to create a question and store it in existing list of questions for the chosen topic. This option redirects the user to **Create Questions** page

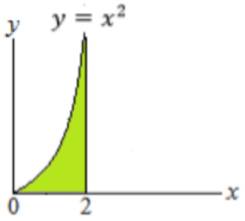
The **Create Questions** page allows the user to create a question for a chosen topic (the dropdown widget is at the bottom of the page). This will queue the user's question in the existing list of questions for the topic

Topics are sorted in order of category (Differentiation or Integration). If a topic is clicked upon, the user is redirected to the **Test** page with the questions from the respective topic they chose



This is the page where the user has to answer questions. The UI for the page supports either built-in calculator input, android keyboard input, or both. The question section of this page supports images in addition to text (which can be used to indicate graphs, shaded regions, etc). The user has the option to go to the next question or quit

Sample question for **Test** page



Question: The following is a shaded graph of $y = x^2$, between the points, $y = 0$, and $y = 2$. Find the area of the graph

Solution: 8/3 (using android keyboard/built-in app keyboard)

2.7 Libraries used for this project

Currently, we are exploring many different libraries for making this app in Java, however for certain UI functionalities such as showing graphs, we plan to use the GraphView library. For other aspects of our project, such as hashing, we plan to use the jBCrypt library. For data structures such as trees, and linked lists, the java built-in data structures are sufficient for our purposes. While we are researching ways to connect to the REST APIs through Android, one potential library that can be used is Retrofit.

3 Conclusion

We have carefully selected the content to cover some basics of differential and integral calculus, according to the A-Level Mathematics syllabus. These topics include First and Second Derivatives, Tangents, Gradients and Normals, Introduction to Integration and Area under the Curve. Students will have the option to select the mode of practice exercises as multiple-choice questions or fill-in-the-blank questions, and can choose from easy, medium or hard difficulty levels.

For implementation, we have chosen to use Java as the language for development as it offers object-oriented programming, extensive community support, high security and immense support for Android development. The UX and functionality of the app will also play a crucial role in the user experience and we have taken this into consideration while developing the app.

In conclusion, we believe that this project will contribute to a better understanding of calculus among students and will prove to be a useful teaching tool for both students and teachers. The combination of technology and education will provide students with an innovative way to learn and we are excited to see the impact it will have.

4 Reference

anonymous007 (2022) Recursive descent parser, GeeksforGeeks. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/recursive-descent-parser/ (Accessed: February 10, 2023).

Bennett, R. (2021) The good, bad and ugly: Apache spark for data science work, The New Stack. Available at: https://thenewstack.io/the-good-bad-and-ugly-apache-spark-for-data-science-work/ (Accessed: February 10, 2023).

Coolman, R. (2015) What is calculus?, LiveScience. Purch. Available at: https://www.livescience.com/50777-calculus.html (Accessed: February 10, 2023).

Kalwan, A. (2019) Why java is secure? top 10 java features that makes it secure, Edureka. Available at: https://www.edureka.co/blog/why-java-is-secure/ (Accessed: February 10, 2023).

Mor-Samuels, E. (2023) App UX: The Fundamentals You Need to know in 2023, AppsFlyer. Available at: https://www.appsflyer.com/blog/tips-strategy/mobile-app-ux/ (Accessed: February 10, 2023).

Perez, S. (2015) Consumers spend 85% of time on smartphones in apps, but only 5 apps see heavy use, TechCrunch. Available at: https://techcrunch.com/2015/06/22/consumers-spend-85-of-time-on-smartphones-in-apps-but-only-5-apps-see-heavy-use/ (Accessed: February 10, 2023).

Prabhu, J. (2020) Kotlin vs Java: Which is the best language for android development?, Tech Blogs by TechAffinity. Available at: https://techaffinity.com/blog/kotlin-vs-java/#: :text=Though%20Kotlin%20is%20not%20as,whopping%2041.1%25%20in%20early%202019 (Accessed: February 10, 2023).

Sewell, M. (2021) Bcrypt: An overview, Medium. Medium. Available at: https://mattfsewell.medium.com/bcrypt-an-overview-e3ef89950189 (Accessed: February 10, 2023).

sidhijain (2022) Time complexities of different data structures, GeeksforGeeks. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/time-complexities-of-different-data-structures/ (Accessed: February 10, 2023).

YPulse (2022) 3 stats on how gen Z is being raised on smartphones, YPulse. Available at: https://www.ypulse.com/article/ 2022/03/29/3-stats-on-how-gen-z-is-being-raised-on-smartphones/ (Accessed: February 10, 2023).