Evaluation Report

Maths Club SAT

Version 1.0.0 approved

Prepared by Garv Shah

Software Development 3/4

29/8/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this evaluation report is to evaluate the efficiency and effectiveness of the Maths Club app, a solution prepared for the CGS Maths Club as a platform. The solution intends to provide a streamlined way for both question creators (admins) and students (members) to interact and consume mathematical content, whether in the form of quizzes or posts.

2. Requirements' Evaluation

Note that not all initial requirements were met in the solution at the time of writing (v1.0.2), as the scope of the project has changed with evolving needs over sprints. These are highlighted alongside an explanation as to why the scope was changed.

2.1 Functional Requirements

Note that FT# represents the Functional Test number for the respective requirement.

Requirement	Evaluation	Testing
Search Engine		
Search for certain sections or individual quizzes/posts	This feature was quite well implemented, using Algolia as a third-party indexer, causing the efficiency of the solution to be far superior to if the algorithms were implemented manually. The requirement is quite satisfactorily met in terms of effectiveness too, since the search engine indexes the whole document, allowing for in text search by content, description, title, or any other metadata available. Sectional searching was removed since it seemed illogical after the app was designed, in that there are only two sections to choose from, but all posts/quizzes can be searched from.	**FT#23**: Pass
Ability to open and enter different parts of the app from the search engine	The ability to navigate the app through the search engine was satisfactorily met, though the scope was later specified to be within the "section" views, since there is not much dynamic content outside of these. There were also a few usability issues initially, as documented through the process journal, due to the manual nature of the animations and transitions. This allowed for search to function as intended, but sometimes made it hard to navigate due to bugs. These were fixed as of v1.0.2, evident by the usability tests, in which no clients experienced issues with the search engine.	**FT#23**: Pass Usability tests also apply, as robustness was improved. A random sample of users were specifically asked to make sure they could navigate the app with the search engine and reported no issues.

Quiz System (Answer Online)		
Have posts that can remain as quizzes for a certain period and then get archived	Satisfactorily met, the quiz system allows for quizzes to have a timeframe, after which they are automatically rendered as posts instead. This lack of server-side change makes the solution quite efficient to implement, with a healthy amount of code, while still being effective client side. Improvements could be made security wise by making this change on the server and restricting answer access up until then, but as discussed in the process journal, this was viewed as excessive for the scope of the app.	**FT#22**: Pass
While posts are an active quiz, members will be able to do the quiz and get points for questions answered	This requirement was satisfactorily completed, as posts can be labelled as "quizzes" and given that they are in the active timeframe, they are rendered as such. Members can also complete the quizzes with the quiz view, providing each question and an answer field, which grabs the LATEX and provides points based on the no. questions answered correctly. There was previously an issue with gaining points, in that if there was a lack of a solution in one question, no questions could successfully be answered. This was fixed in v1.0.1, where the admins were forced (client-side) to provide solutions if the post was not a draft.	**FT#22**: Pass
Solutions only to be released after the quiz period	As mentioned above, this solution was implemented client-side. Solutions are not available to the user when completing the quiz, as part of the UI, but are available through code. This makes the solution quite efficient in terms of code, speed, and resources, and though it is equally effective for most users, it may be vulnerable to someone with malicious intent. As described in the process journal, it was decided that client-side was the more practical solution to meet this requirement.	N/A
Members should be able to answer questions within the quiz interface	This was satisfactorily met within the "quiz view", where users could interact with a specialised maths text field, allowing them to input equations using an onscreen calculator instead of manually typing it out. This is quite an effective solution for answering questions, as it allows for complex answers to be inputted easily, while also being efficient for the user, as they don't have to learn how to input LATEX and/or the computer doesn't have to interpret plain language text.	**FT#22**: Pass
Should be able to differentiate between different math being equal or the same	This requirement is satisfactorily met server-side, with a TypeScript file being written to mark answers. This file turns the LATEX string into a maths expression and can consistently evaluate if expressions are the same or not. This required quite a bit of trial and error in terms of programming, but eventually got to the point where server and client answers could be evaluated for equality in a secure way.	A test post was created, titled "Basic Addition" under Junior. The answer is factorised and if the user entered the expanded form, and it would compare it with the factorised to check for equality.

Divided Section for Junior/Senior etc		
There will be a "group" system, where you can create different groups and assign users to them, granting permissions.	This requirement was met quite well, with a group system that each user is assigned to, and sections/permissions the role can access. This is done mostly in the backend (database) as an admin page was not implemented in the UI. Users are assigned to a role in the "roles" collection under the "members" array. These roles are then referenced in the "postGroups" collection, under which roles are assigned to which post group. This serves quite effective in terms of security and is also efficient as all users can automatically be added to the initial "users" array.	N/A
Leaderboards		
There should be a leaderboards page, where points from quizzes can be viewed	Satisfactorily met, a leaderboards page is present with all users and their points from quizzes. This is stored under the "quizPoints" collection, to provide security as users should not be able to edit their own points, and other users should not be able to view sensitive information. This is quite an effective solution, as it allows for fast leaderboard sorting without compromising on security. It is also efficient for the admins due to it robustness, meaning the data on the server does not have to be constantly maintained. Rather, if a user somehow procures an invalid public facing profile state, the app can still render the leaderboards.	**FT#11**: Fail **FT#12**: Pass The leaderboards have seen an unexpected issue with bot accounts, and has remained robust
Clients should be able to click a button to automatically scroll to their leaderboard position	This was not in the solution as of v1.0.2 as it was viewed as an extra. There are a few features that should not have been viewed as *requirements* as they are not required for an effective or efficient solution but are rather add-ons that would be convenient. As such, it could be added in future versions but was removed from the scope of the first two sprints.	N/A
Badge System: members should be able to get badges that will be manually assigned	Similarly, the badge system was also removed from the scope of the app as it was unnecessary for the number of clients in the current Maths Club, and was viewed as excessive, resulting in the requirement being taken out of the scope of the first two sprints. If the app headed in a direction of a different target audience, such as the whole school or even multiple schools, such features could be added.	N/A
Optional Hints		
Question curators should be able to optionally provide hints that users view	Successfully complete, this requirement is integrated into the creation page for admins, where they can optionally add a hint to their questions, which comes up inside the posts, a step between the answer and no guidance. If no hint is provided, no button is rendered. Hints could also be added to the quiz view but have not been within the first two sprints.	N/A
Forums/Ability to Comment on Posts		
Members should be able to comment on posts, facilitating community interaction	All commenting prospects were also removed after the first sprint, as it was viewed as too much for a group of about 20 – 30 people. The removal from the scope was a result of observing people using the app and how they used the app, and the impression was received that it would be more beneficial to foster in-person interaction while discussing problems rather than comments. It could be added in later sprints but was not seen as a requirement for the first two.	N/A
Non-approved members should not be able to comment		

2.2 Non-Functional Requirements

Note that **UT** represents that the requirement was tested through the usability tests, by asking a random sample of people what they thought of the specific criteria.

Requirement	Evaluation	Testing
Usability		
Search Engine should remain relatively fast with increasing size	The search engine is quite well designed in that no matter the size of the indexed database, the search engine will remain extremely fast. This is mostly thanks to Algolia, which handles a lot of the algorithmics regarding speed. So far, despite how big the post was or how many posts were present, the search engine has felt near instant.	**UT**
Search Engine should make it easier to navigate the app and get to where you want to	This was achieved quite well, as all users who used the app and were questioned about the usability of the search engine reported positive experiences. Every user provided feedback that the search engine caused the app to be easier to navigate than the previous solutions.	**UT** Clients also responded well to the Search Engine in the survey
Quiz System should feel fair, promoting a healthy competition, and be easy to use	Though this was achieved, it was harder to measure without extended usage. Long term competitions were not conducted within the first two sprints, and notions of fairness were influenced by the difficulty of the questions, which was a problem induced by the question creators, not the system. The system was reported to be easy to use, helped especially by the on-screen keyboard, but it also hasn't been tested as extensively as it could be, and may be improved by guiding users on difficulty, such as a rating system which could be implemented in future sprints.	**UT** A couple usability issues were found with the Quiz System, due to PICNIC problems from the admins causing no points to be assigned. This hindered healthy competition for a period of the testing.
The GUI should be able to provide easy separation for sections such as Junior and Senior, which should feel intuitive	This was achieved very well, as reported by clients. The sectioning off in the app was a lot more intuitive than the website and systems before, and everybody found the GUI to be helpful and easy to navigate. No issues were reported.	**UT**
The leaderboards must include a sense of competition that encourages people to partake in Maths Club more often	Most people said that the leaderboards were a great addition which encouraged competition, but that could vary with each student. To improve this, the leaderboards could very well be improved in subsequent sprints, such as a publicly viewable profile and/or badges. Competitions could also be held in real life, with in-app rewards, which could help the badges feel more "real".	**UT**
Reliability		
Should reliably be able to handle large file sizes without limits or expiration	This was very partially achieved, due to file handling limits. The initial idea was to use Notion, but that become unfeasible quite quickly. The next solution was to have no dedicated server, and just get admins to paste in links to their files after uploading them to another CDN. The problem with this is that it is not reliable at all (other services expire their files), and it is also a hindrance to the client. The final solution was to just use Firebase Storage (added in v1.0.2), which is very reliable in terms of file uploads, but is feasibly	**UT** This was a part of the usability testing, but also had issues encountered after they were completed by other admins. v1.0.2

	limited in size. Though users can upload as many files as they want, this will eventually start charging money, which clashes with a constraint of the project. All in all, this requirement was achieved in terms of effectiveness and efficiency but is still far from perfect.	was released in response.
Portability		
The software solution should work on most of the intended audience's devices	This requirement was fulfilled by active deployments on the Web, MacOS, iOS and Android. These four main platforms encompass almost all of the userbase, and in cases which they do not, the website is always an accessible solution.	N/A
Robustness		
The system should be able to handle invalid account creation or user input in the quizzes	The system was built with robustness in mind and is therefore quite malleable when it comes to invalid data. The whole app is null safe, and specific areas of the app were built to be able to handle any piece of data being unexpected, namely the leaderboards which can handle a lack of any field, fields being the wrong datatype, fields being infinity or negative and a whole assortment of possible changes. This is useful because quite a few public facing accounts have already managed to become invalid, whether that is due to a network error, or closing the app at just the right time, but the app has continued to work for them and others just fine, correcting itself in due course. This aspect could be improved in certain sections though, specifically in areas that were not programmed in-house but by third parties. Document editing for example is from an external package, and though it has received contributions and changed throughout the project, it is far from safe from errors. This could be a major weak point for content creators.	N/A
Maintainability		
The software solution should have enough documentation and tests written such that maintaining the software after it is written does not require rewriting parts of the software.	The software solution is quite well documented, though unit tests are lacking, due to the scope of the SAT. It was advised that manual unit tests were preferred over automatic ones as they give more feedback to be reported, but this has the drawback of them being slower and more cumbersome. The software's code quality was a focus though, and it is most likely that sections will not have to be rewritten as they have not had to be rewritten yet. There is also extensive internal documentation, with a bit of external documentation created at the start as well. This should aid in maintainability but could be improved by the addition of the admin panel, which was planned but not implemented by the final sprint.	N/A
Security		
The forums should protect and consider the privacy of school students	As the commenting/forums were removed from subsequent sprints, this requirement took on a new form throughout the development process to refer more to the safety and privacy of student data rather than comments on posts. That being said, the app was designed from the ground up with security in mind. Data structures have aided in implementing security rules server-side that deny client access based on privilege, and industry standards by Google were strictly followed to make sure privacy was a priority.	It was hard to test this requirement on a large scope with competent pen-testers, but the author did attempt to penetrate the security.

3. Efficiency & Effectiveness Evaluation

3.1 Efficiency

Overall, the whole solution was quite efficient in its approach to user interaction and speed. Navigating the UI is fast and responsive and doesn't have many unnecessary abstractions that make it hard to navigate. The app is also generally fast in terms of calculations and results, such as the search engine or loading data. Along with this, most elements are also live, utilising streams rather than snapshots, causing the data to be fast and responsive.

The one scenario in which this lacks is when requests are hindered by internet connections or network speeds. For example, internet connections are required for marking questions, as points cannot be awarded client-side owing to security requirements. This additional network step slows the process down. Similarly, if the client's own network speed is slow, the app can feel less efficient.

The UI navigation also has a weak point, that being the actual document editing. A fork of Flutter Quill is the package being used for this implementation and has been chosen because it is the only viable option for document editing in Flutter. The code quality is not ideal, resulting in efficiency that is not reflective of the rest of the app, not only in speed but also in UI navigation, where the toolbar can be buggy. This weak point would need to be ironed out in future iterations for efficiency to be improved.

3.2 Effectiveness

The effectiveness of this solution can be displayed by the results of the usability tests, in which all clients were thoroughly satisfied with the solution, rating it much higher than previous solutions to the presented problem. It effectively solved the issues faced by both target audiences, and though there were a few more issues faced with the admins than the members, as of v1.0.2, both groups can complete their respective tasks with ease.

The solution's effectiveness also relates to its accessibility, as if students cannot access the app in its entirety, it cannot be utilised effectively. As such, it being an extremely portable solution aids well to improve effectiveness, along with the app being licenced to be put on Caulfield Grammar School's "Self Service" platform, meaning the app can be installed on school managed devices. This allows for to be an extremely effective solution for the whole school when it comes to answering Maths Club problems in a remote fashion.

4. Development Model Evaluation

Throughout the project, the Agile development model was used, alongside the Scrum framework. This allowed for rapid development, with a flexibility that was quite suited to the project. As this was an initial prototype, Agile allowed increased communication with the client, and the rapid iterations allowed for them to see requested changes fast. This improved productivity and efficiency, as it made the app focused on the features the client would like to see, making it more effective.

The development model proved quite effective in being flexible enough to change the app requirements after they were initially set. After the first sprint, it was found that several of the initial requirements were not *requirements* for the clients, but rather features that could be added as extras in the future. Agile enabled the project to prioritise features that were important to the client, while still leaving space for improvements.

One factor that negatively impacted the effectiveness of the Agile model was how easily scope creep introduced itself within the development phase of the PSM. Daily scrums lacked more than one individual in this project, making it so the scope was not constantly in check. There were multiple occasions when features were added because clients requested it, or the developer desired it to be in the product but were not necessary at all when conferred with the requirements and priorities. An example of this would be the immense time spent on making the app "live" in the second app, so that every update was in sync and always reflected the latest changes. Incidents like this hindered the efficiency of development and the effectiveness of the model, so it may be worthwhile conducting small retrospectives at the end of every daily scrum, allowing the team to stay true to the scope of the app.

Overall, the Agile development model was most definitely the best fit for this project, owing to the close connection to the client, where gradual improvements and quick turnarounds let changes be made effectively. The flexibility of the model was great as it allowed for the functional requirements to be adapted to changing needs, but this flexibility also introduced scope creep which negatively influenced efficiency of development. Continuous development aided in reducing risk and getting ideas into client's hands fast, but also made it so it was harder to set hard deadlines for the end of sprints, making the timeframe of the project shift a couple weeks past what was ideal.

5. Project Plan Evaluation

5.1 Adjustments

Numerous adjustments were made to the project plan throughout the PSM lifecycle, namely to reflect new needs in new sprints. All of these can be seen within the process journal, but a few noteworthy examples are analysed in this report. There were times when events came up that hindered progress on the project, and as the project plan did not account for these unanticipated occurrences, it was not very flexible to adjust. These included examinations, overseas trips, and camps, all of which were not initially predicted, and due to the lack of "padding" time given in tasks, the project plan suffered very tight deadlines.

There were also occasions when the project plan was changed due to changing client needs, which was a more positive form of change. After every sprint retrospective, the product backlog was revised, as was the project plan, to reflect the changing needs and new deadlines based on sprint goals. This aspect, though effective, could have been improved with more extensive timetabling within sprints, as the project plan only included one hard deadline for the whole sprint when smaller deadlines could have prevented scope creep.

Finally, project plan changes were made due to dependency issues. In certain cases, this was caused by a dependency introducing a new bug that was out of the project's control, or development environments simply not working as expected, which was the case when Cocoapods stopped working. Overall, this type of project plan adjustment was quite hard to avoid, not only due to how unexpected they were, but also how hard they can be to fix. Extra "padding" time could most definitely have helped, and if deadlines did not get so tight due to the delays mentioned above, they could have been mitigated.

5.2 Effectiveness

All in all, the project plan was very effective for the project. There were a range of factors that influenced how effective it was at times, though it always served its purpose of keeping the project on track and increasing efficiency when it came to development.

One factor was how frequently it was being consulted and revised. The project plan was easy to modify at the end of sprints, during sprint retrospectives, but these led to relatively large changes at once that modified the course of the project. The project plan, when combined with the Scrum framework, may be better suited to smaller sprint retrospectives at the end of each daily scrum, or with more than just one individual, so that changes can be more gradual, and the plan is viewed more to stay on track.

Another factor that influenced the effectiveness of the project plan was how well the time designated to a task would reflect the reality of how long it took to complete. Ideally, each task's timeframe should be known beforehand with adequate padding time assigned, but this was not always the case with this project. This could be aided by not planning too far into the future and keeping the plan more relevant to tasks with a known length.

Finally, the project plan was influenced by the viability of the tasks. Certain tasks were not researched in enough depth and were later found out to not be viable for a project of this size. This created a lack of efficiency in the project plan and made it less effective as it was required to go back to the drawing board. An example of this would be the planned Notion integration in the stack. Notion was not a viable CDN, due to how the file links would expire and keep changing, but this was not known at the time of planning, and later had to be changed. Similarly, a Fastlane integration was planned for CI/CD, but ended up taking far too long, not only not being viable (as it was much harder to implement than expected), but far exceeding the scope of the app, considering the size of the target audience.

Factors as the ones provided made the project plan less effective than it could have been, but it still aided well in keeping the project on track, improving efficiency and effectiveness.

6. References

This document refers to https://github.com/garv-shah/software-dev, as this address is where a variety of work on the project exists, including interviews, ideas, and documentation. https://garv-shah.notion.site/Garv-s-Blog-80ae26b22fd04677a0553fa668f6fc03 is also where project updates (process journal) live, under the "Software Development 3&4" tag.