PORTFOLIO

JONATHAN MENG

I'm an aspiring software developer, passionate about translating solutions and ideations into tangible, value adding software. I derive a sense of thrill from technical challenges, and pride myself on my creative problem solving skill set and analytical capability. My strengths in adapting to challenges, empathetic and critical thinking, enable me to uniquely contextualise and bridge the gap between clients, ideas and the end product. I hold myself to a high standard in projects I deliver, fuelled by my tendency to think outside the box, to prod and probe and to play devils advocate. To see if we can do things differently, better. Where people see difficulty, I see a unique problem to be solved.

ROLES AND RESPONSIBILITIES

As the lead developer working in an Agile team, the scope of my responsibilities extended from discovery all the way to product launch. Being placed in such a unique position in an agile environment, initial responsibilities heavily involved collaboration with the Product Owner, Scrum Master and Business Analyst to strategise and prioritise scope, features and capabilities of our proposed software solution. This involved the customisation of product backlogs, release backlogs which culminated in our sprint backlogs. Once sprint planning was complete, these responsibilities would shift towards the management and delegation of these backlogs, with oversight over team progress towards the completion of these backlogs. Within the scope of my development team, technical responsibilities included environment selection, database selection and integration, code refactoring, building and testing among other technical considerations. Further roles undertaken included the mentoring, guidance and growth of my Development Team, in order to facilitate the transition from requirement to product.

ACCOMPLISHMENTS

Management and delegation of development objectives balancing the core concepts of the Iron Triangle; features, time and resources. Throughout the timespan of this project, these three concepts were always critical constraints that I had to juggle as the Lead developer in collaboration with strategic and managerial team members (Product Owner, Business Analyst and Scrum Master), as well as my own development team. From the strategic perspective, liaising with the Scrum Master and Business Analyst towards the implementation of product and sprint backlogs involved constant deliberation over features. Such concerns needed a shared understanding as to the limitations of our team. To facilitate this, I leveraged the action priority matrix in sprint planning meetings, to express where the development team stood in terms of feasibility, whilst ensuring transparency, and maximising impact and progress.

Mentoring Development Team members to ensure sustained high performance during sprint phases. A core responsibility as a Lead Developer was to provide support, training and collaborate with the development team during the execution of our sprints. Initially I noticed an inherent struggle faced by the development team, reflected in the feedback as follows:

"Hey Jono it would be good if you were able to help some of the other developers because I think they are struggling to understand some components of your code" - ***Nathaniel Tran*** *(Scrum Master)*

As with the nature of teams, the issue of disparity of coding skill levels were apparent. Taking on the feedback immediately, I took on the initiative to conduct a meeting with the development team to explain the code base and detail the frameworks I had created in order to ensure a shared awareness amongst the team. Specifically this included explanations regarding the decision to separate Android Activities into View Fragments and universal Navigation bars, as well as explaining the logic behind Activity and Fragment Lifecycles which were significant blockers towards the teams understanding of the code base. From then on, I detailed common components that would be used within the application, (RecyclerViews, ScrollViews, CardViews, Progress Bars, database calls) further documenting and noting resources the team could follow if they were stuck with the implementation of a particular feature, maintained throughout the entirety of the project.

Delivering quality software through the application of code reviews, and focusing on continuous code refactoring. As the code base became more complicated and layered, I sought to incorporate consistent code reviews and code refactoring meetings not only to facilitate knowledge sharing, but to better equip the development team for further feature implementations. Code reviews were conducted at designated checkpoints in the completion of a feature. Occurring regularly, these reviews involved analysing logical implementations, referencing user requirements, consistency in style and stress testing new features. Following best agile practices dictated by Atlassian, a culture of quality was incorporated into our software development process, whereby an emphasis was placed on preventing bugs rather than detection and having the developers hold a shared responsibility for quality assurance. This involved creating core development values of critical thinking, freedom to investigate and risk assessments, thereby enacting these values not only through code reviews as a team, but also consciously shifting our individual mindsets to emphasise quality throughout the development process.

KEY LEARNINGS

The importance of Ceremonies

I initially underestimated the impact of following through with ceremonies, as this development process shift seemed overly pedantic. However, it became evident after our first sprint that we faced challenges in responsibility clashes, conflicting definitions of done, underdeveloped requirements, and breakdowns in collaboration, which these ceremonies were designed to alleviate. The subsequent Sprint Review allowed us to understand and become aware of these issues we faced, through the celebration of our progress. As such we leveraged our Sprint Retrospective to really identify these pitfalls, in order to reassess responsibility and align expectations, ultimately helping us significantly improve productivity as we refined our approach. Committing to this framework and embracing the ceremonies proved to be most valuable in the successful development of this project. As a team, we strived towards three core values, bonding, planning and commitment, where we strived particularly to work/hold ceremonies together in person, promoting accountability through leveraging and maintaining sprint tools such as Miro and Trello Boards.

Adjustment pains from waterfall to agile

A key obstacle in my transition to an agile development process was falling into old waterfall patterns. I often found myself challenging the agile approach, specifically in terms of software development. The agile environment pushes a heavy emphasis on pushing out features and functionality at a rapid pace, whereas I was used to the process of completely fleshing out the requirements of our final idea and executing the development process in one go. As such, it was hard for me to embrace the possibility of redoing a lot of my code. This drastic shift however, prompted me to develop a deeper understanding of Agile as well as the role of a Developer. Through this experience, I learnt the importance of collaboration, knowledge sharing and embracing change and pivots. As the Lead Developer, I leveraged initiatives such as prioritising Quality as a culture, regular code reviews and refactoring in order to facilitate the unpredictability. From a team perspective, these initiatives allowed both myself and the team to have a common mindset and approach to developing our code base, as well building on our expertise within a collaborative environment. From a technical perspective, extensive modularisation, abstraction and upkeep of our code base ensured that modifications or pivots could be executed in a timely manner, reducing spaghetti code and maintaining efficiency and readability on our day to day development tasks.

Trust in the team

Specifically during our first sprint, I reflected upon my role as a developer. At this stage, the foundations of our app structure and the main views in our application were ready to be implemented. The assumption here was that the devs would be able to jump in and start work off the backlog. However, a key issue I immediately encountered was that I found myself tackling most of the tasks on my own as the Lead Developer and not properly delegating backlog items to complete. Feedback from my team reaffirmed this issue:

“Hey Jono, I noticed a disparity between our velocity estimates and actual progress. If you’re able to more efficiently delegate some of the sprint backlog, that would give us more opportunities and time later on to add features” - ***Nathaniel Tran*** *(Product Owner & Scrum Master)*

“Hey man, I think the way we've structured the app is gonna be a problem further down the line. I'm already finding it difficult to understand some of the code.” - ***Calvin Koder*** *(Developer)*

Leveraging a why-why analysis 3 initial reasons for this hurdle were: dev team members aren’t on the same page with advanced GIT collaboration, I didn’t trust the team and ambiguity of functional requirements. Drilling down, root causes found respectively were: not being exposed to a project of this scale, code concepts that were unfamiliar to the team and, too much time being taken to explore new features rather than consolidating existing requirements. In order to improve this aspect, I had to take on the responsibility of mentoring the development team, and find strategies to make the code base as readable and understandable as possible.

PROJECT: DINOsaur

Prioritising people: Innovation for wellness and resilience

- Support students in identifying and navigating their emotions by providing regular opportunities to share and be heard
- Provide teachers with more data and actionable insights to support better decision making in classrooms
- Proactively build collaborative relationships with families and communities to create a shared understanding of how to support student learning, safety, and wellbeing

PROBLEM STATEMENT

"How can we help students navigate their emotions in a remote environment that disconnects the interpersonal relations of students and teachers."

ROLE

My role as the lead developer consisted of the management and maintenance of the development team, motivating, delegating and guiding the development team in the development of our Application. Further, overseeing and implementing front-end and back-end functionalities leveraging core Java principles, frameworks and best practices including material design on our Android Studio app.

RESOURCES

Android Studio
Java
Firebase
Figma
Miro Board
Trello
MS Teams

FEATURES

Our mobile application DINOsaur, seeks to bridge the gap between teachers and primary school students, generating, storing and visualising key quantitative and qualitative data points from students through the incorporation of gamification, to give teachers a comprehensive view of their students' wellbeing.

KEY USERS

"To keep in contact with the teacher, I use Edmodo, Moodle and Microsoft teams" - **Primary School student**

There is no structured curriculum to assess and examine how a student is doing emotionally and socially because of the rushed pivot to online learning. I had to make the most of the technology the school provided us such as Microsoft Teams and other basic tools. However, these only really tackle the educational responsibilities of my role as a teacher. It's hard to really check in and stay on top of students with the disconnect of online learning." - **Primary School Teacher**

DATABASE DESIGN

MYSQL & NOSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or	No effect on the design with

Adopting the Agile process complicated the database selection process. Ultimately we selected **Firebase**. Key influencing factors being:
- Realtime sync and storage of data
- Scalability for high volume CRUD
- Dynamic, modifiable and flexible to incorporate user feedback and pivots

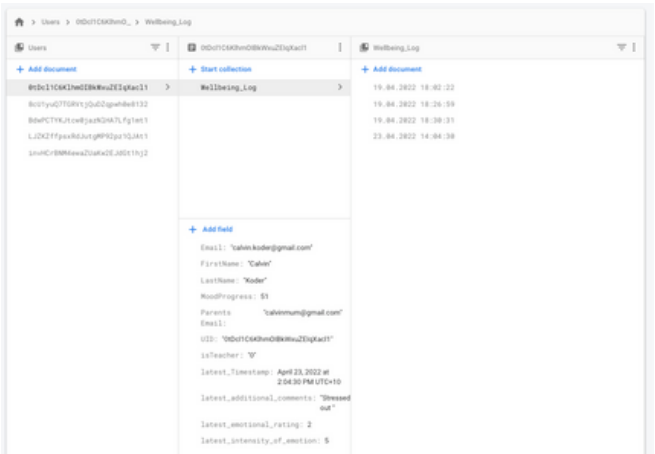
SOFTWARE BUILD SELECTION

Mobile OS selection

Within the mobile application space there are 2 dominant operating systems (OS) being IOS and Android.
The selection process below will be essential in determining the more appropriate OS

Mobile OS Selection Criteria

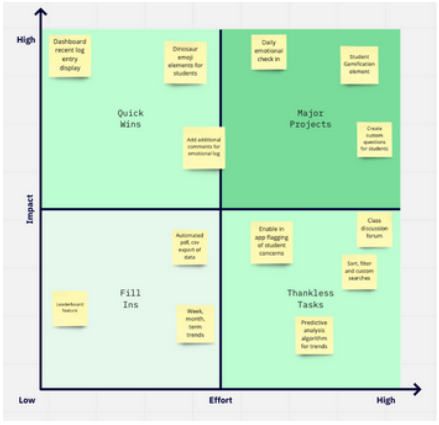
	Number of users (globally)	Open Source	Development Time	Design and development Constraints	IDE	Development team Knowledge
IOS	25.49%	Partially	Rapid development possible "Swift is up to 2.6x faster than Objective-C and 8.4x faster than Python"	- As a developing language, constant updates may see incompatibility between dev versions - Lack of developer expertise - Limited cross platform support - Lack of support for early IOS versions	Yes	Jono (2/10) Calvin (0/10) Jayden (1/10)
Android	69.74% https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/	Yes		- Consistent documentation and large user base - Market is more competitive and saturated - Compatibility concern for multitude of screen sizes	Yes	Jono (7/10) Calvin (6/10) Jayden (6/10)



The selection of our software build allowed the team to highlight strengths and weaknesses individually, as well as in Mobile operating systems. Our selection was quite straightforward, with the Development team having extensive background knowledge in Android Development, leveraging Java concepts and Object Oriented principles. It was a language we were all comfortable and familiar with developing on. From a primary user perspective, we saw a majority of the population in possession of android devices over iOS.

DEVELOPMENT TEAM MANAGEMENT

Maximise productivity and maintain a high functioning team was a core goal of mine. As the Lead Developer, not only did my responsibilities involve writing code, I had to oversee, delegate and manage the Development Team as a whole, aligning the strategic vision whilst implementing strategies to maintain motivation, engagement and mentor the team. As such, this involved meticulous collaboration with the Scrum Master and Business Analyst in the development of functionality, requirements and assessing its impact, with a sustained focus on teamwork and knowledge sharing in consolidating our product and sprint backlogs.



Action Priority Matrix in Sprint Planning

Item Reference	Description	Assigned To	Effort points	Status
https://trello.com/c/aNBh4sN1	Complete 20 min pitch slides	Primary: Nathaniel Supporting: Matthew	5	Completed
https://trello.com/c/yp3VArmB	Implement changes for mood scale + daily Qs	Primary: Jono , Calvin Secondary:	2	Completed
https://trello.com/c/hwrgFHBW	Teacher student list	Primary: Jono , Calvin Secondary:	2	Completed

Snapshot of Sprint Backlog

In order to plan for successful sprints, it was essential for me as the Lead Developer to assess feasibility in terms of our resource pools (development team capability) whilst ensuring desired features were effective and impactful. This extensive collaboration with both the Scrum Master and Business Analyst alongside the Development Team was facilitated with tools such as Action Priority Matrices, to visualise and contextualise our alignment with development strategy. This collaboration contributed to a greater accuracy and feasibility of our sprint backlogs. The biggest impact was the calibration of effort points throughout subsequent sprints, where we successfully completed quite an ambitious sprint backlog, depicted in the velocity chart below.

QUALITY AS A CULTURE

As the Lead Developer, setting the right internal vision and strategy was essential not only for the strategic vision of the project, but for the personal development of my Development team. Leveraging Atlassian Agile principles to: Create a culture of quality, Push responsibility for testing back to developers, Prevent bugs, not detect them. From a coding standpoint, this meant consistency in naming convention, UI Design and leveraging Object Oriented Design best practices, to create modular, reusable and understandable code to align knowledge, vision and to create a collaborative growth mindset within the Development team.

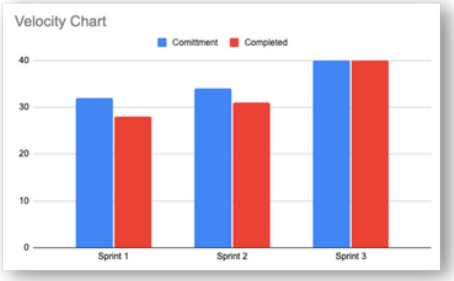
Method Snippet
Modular, reusable and understandable code

Naming Consistency

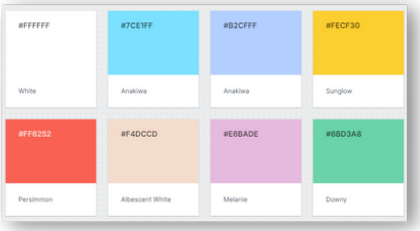
QUALITY SOFTWARE

This focus further extended to developing software with an emphasis on continuous improvement. I incorporated regular code reviews and refactoring meetings to ensure the sustained commitment to this culture. This code review became an enjoyable process within the Development team as we had the chance to showcase our progress and achievements to each other, facilitating knowledge sharing and empowerment in individual abilities. Such review processes would typically consist of:

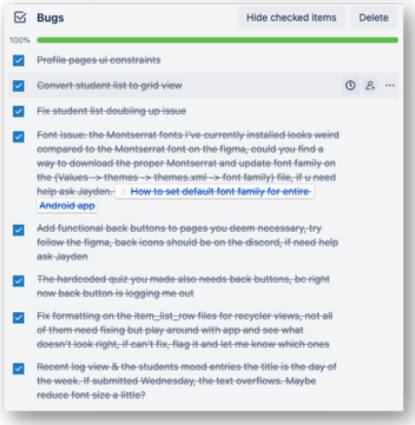
- Developers demo of the feature, concepts used, and interesting things they had learnt
- A recount of controls and how they developed the feature with error prevention and quality assurance in mind
- Followed by stress testing from the dev team and noting additional issues encountered
- These would be collated into bug reports to be fixed and retested



Sprint Velocity charts



UI Design Consistency



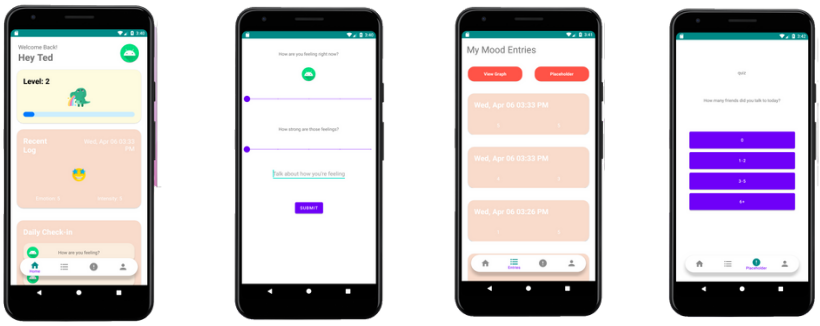
Snippet of Code Review Bug reports

FINAL BUILD

Through several cycles of implementation, testing, feedback and development, the team and myself are proud to document the progress we have made in the development of our solution.

Early Builds

From progress made in early sprints, and our focus on repeated user testing and my commitment on development management, team mentoring and focus on quality has seen rapid incremental changes to our initial build release. As shown on the right, an emphasis was created on creating modular components within the application not only for best practice conventions but to ensure pivots and modifications could be achieved without compromising on time.

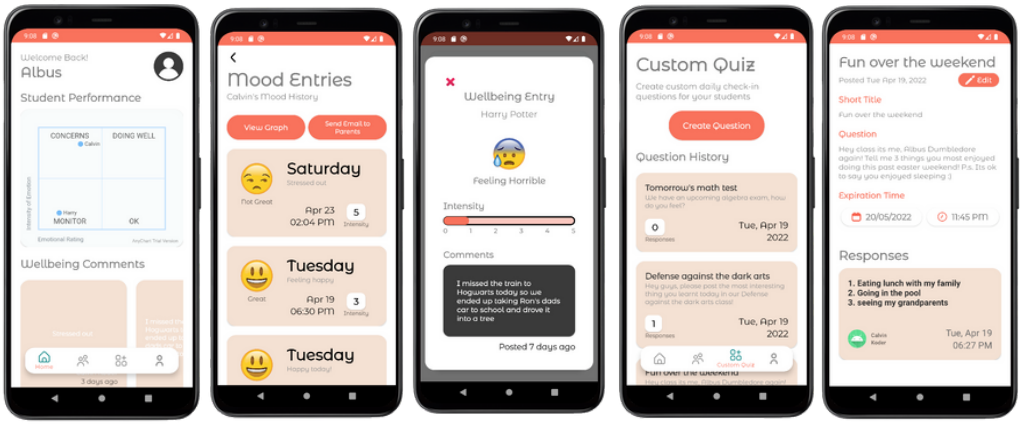


Student Home *Emotional log* *Entries log* *Quiz*

DINOsaur Early builds

Teacher

The teacher side allows teachers to evaluate and be informed by student data. The dashboard incorporates a quadrant view of students last emotional log as well as a live feed of wellbeing comments. These elements were heavily influenced by user feedback, as we tested what teachers really wanted to see.
Further pages highlight key functionalities such as drilling down into a particulars students log to assess historical data, and the creation of custom questions where teachers can customise and view details and student responses.



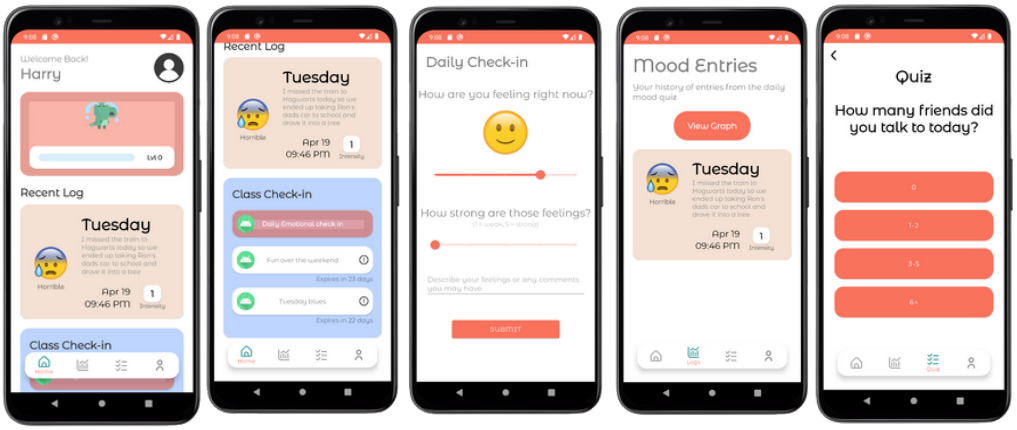
Teacher Home *Student history* *Detail view* *Custom Quiz creation and history* *Custom Quiz detail and responses*

DINOsaur Teacher Functionality

Student

The student side provides the ability for students to generate and record data through 3 primary means:
- Daily Check-in emotional log
- Answering custom questions set by teacher
- Completing the daily quiz

Our gamification feature was structured as the main element within the home screen, to incentivise engagement and sustained use. Through user feedback, these emojis proved to be a huge hit for primary students.



Student Home *Home pt 2* *Emotional Log* *Entries Log* *Quiz*

DINOsaur Student Functionality

Next Steps

As of now our final release has been launched and is being tested by users. So where to from here? Well, we envision several dramatic improvements to our future releases. We identified collaborating with health professionals (psychologists, psychiatrists and youth therapists) to be key in the development in the next stage of our application, to derive professional insight towards refining our current features. This includes the potential addition of specialised modules, refining our depth of data evaluation and actionable insights backed by professional opinion. In the meantime, future implementations include a class leaderboard, student forums, filter and trend analysis and customisable student avatars!