1. Introduction

1.1. Project, Context and Goals

In this project, our team called MNRP Tech was assigned the company called LeapIn! with our client contact being Jane Sheehy. For the rest of this report, we may refer to the client contact as Jane.

LeapIn! provides Plan Management services for people part of the National Disability Insurance Scheme or NDIS which is a scheme sponsored by the Australian Government to fund people with disabilities. Therefore, Leap In! operates in the Disability Services Sector and provides a platform for participants of the program to be able to manage their NDIS Plans.

For this project, the client sought to develop a provider feedback capturing and surfacing mechanism so the project was termed "Provider Ratings". Such a mechanism was not limited to designing, developing and deploying a system on the website and on the mobile applications to let users give feedback about the services of a certain provider. It also required storing that feedback in the database of the company in the backend and then surfacing the feedback as per the requirements of the client. Therefore, apart from web, Android and iOS development, the development team had to also work with Application Programming Interface (API) communications to establish contact between front-end and back-end.

1.2. Project Outcomes

1.2.1. Feedback Eliciting Mechanism

After multiple iterations of client feedback implementation, the team successfully developed a mechanism to capture qualitative and quantitative feedback from its users. Quantitative feedback was captured in the form of Star Ratings and a numerical scale of 1-10. These systems were placed in two different parts of the website as per the requirements of the client. The styling was all set to match the rest of website including colors, font size and style, dialog boxes and prompt buttons just to name a few. The systems were approved by the client.

1.2.2. Storing the collected feedback

The feedback captured from the developed mechanisms was stored in the backend successfully. The team was given access to the database of the client and therefore, created a new Table in the database to store the feedback. This was all approved by the client and the outsourced I.T team of Leap In!.

1.2.3. Surfacing the feedback

Finally, the collected feedback was displayed on two different places on the website. As per the requirements, only the most recent feedback was displayed.

2. Project Set-up

2.1. **Project Management Approach**

For project management purposes, we followed the Agile methodology. It was suitable for our project since it is incremental and iterative in nature. This was useful since our project had three stages which were collecting the user feedback, storing the feedback appropriately and surfacing it. Plus, the success of each stage depended on the one before it which means we may have needed to revisit the functionalities that were developed earlier for various reasons for instance resolving a bug or changing the user input type. Therefore, the iterative nature of Agile helped us a lot.

Moreover, we also needed to get the functionality and design approved by the client for each stage before moving on to the next one. This is where the incremental development of product came into play where we released the minimal viable product (MVP) to get client feedback and implement that feedback the following week thereby completing a Sprint. Once two Sprints were completed, we would have one stage of the final artefact completed with client feedback implementation and testing thereby completing a release.

As for the management of the Sprint and the processes used, we had internal meetings every Friday morning and client meetings every Wednesday at 11.30am for updating and feedback elicitation purposes. The client, Jane Sheehy assumed the role of Product Owner while Mahad was the Project Manager and Client Liaison. The Development (Dev) team included Nihar, Sangpil and Ricky where Nihar and Phil were the front-end developers and Ricky was the backend developer. For mobile, Nihar worked on Android while Ricky and Sangpil worked on iOS. A lot of user testing and tech support was provided by Mitra which is the offshore technical team for LeapIn!.

2.2. **Client Expectations**

In our project, our client Jane Sheehy was the product owner. She set direction for the project, determined the scope as needed, provided feedback and validated our releases. The offshore tech team, Mitra, could also be considered our client to some extent but they did have many expectations apart from reviewing the code and providing feedback.

Our primary channel of communication with the client was Slack. We set up workspaces that included a general workspace which was the primary channel of communication for everything project related. Then, there was also a tech workspace which was mainly used by our dev team to communicate with Mitra. Figure 2.2.1 shows the layout of our Slack workspace.

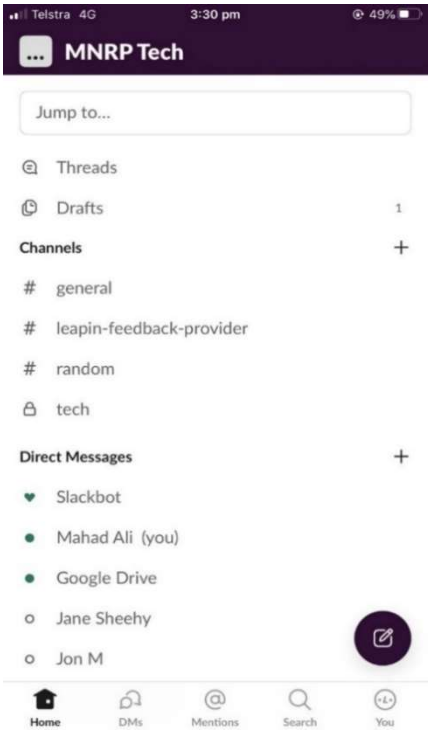


Figure 2.2.1: Slack Workspace Layout

To manage client expectations, we first set up a weekly meeting time with the client and set up recurring Outlook Calendar invites. (See Figure 2.2.2) This time was then pushed ahead by half an hour since it worked better for the client.

;

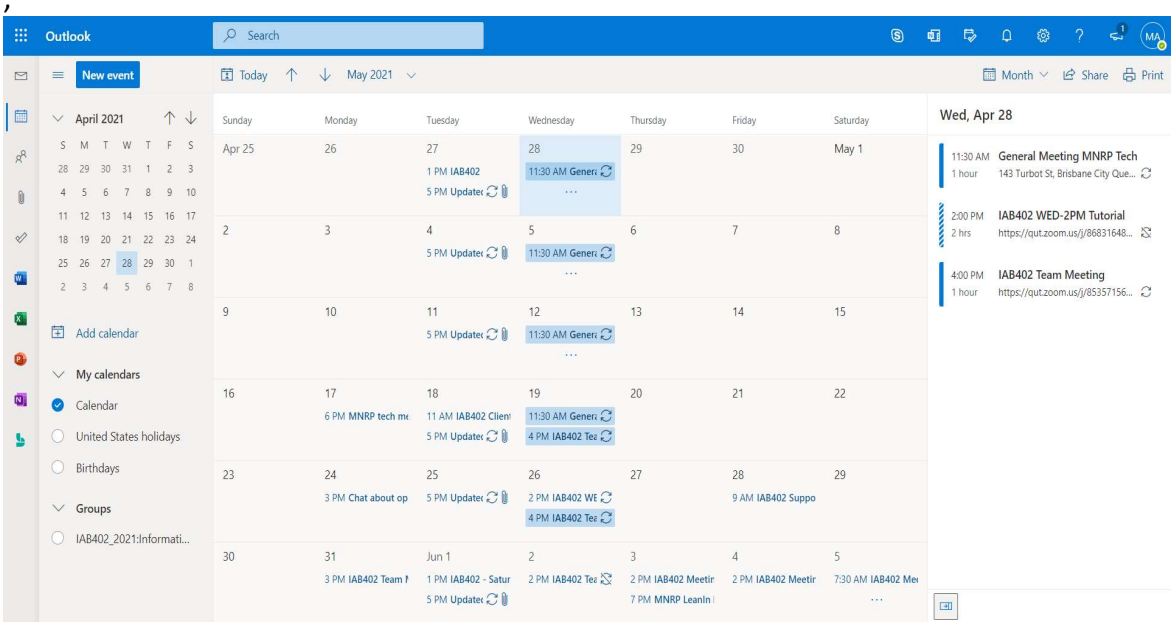


Figure 2.2.2: Recurring Outlook Calendar Invites on Wednesdays. Meeting Description on the side pane.

In the meetings, we would update the client of the work that had been done until the previous week, overview of the work completed in that week and the work to be completed in the following week. We would also remind the client at the start of every meeting about which Release, Sprint and Week we were in. Plus, we would

also update the client about any meetings held with Mitra since the previous client meeting and the meeting outcomes. All these updates were compiled in a PowerPoint Presentation and was presented to the client. Figure 2.2.3 shows a sample of a slide from our slides from Week 6.

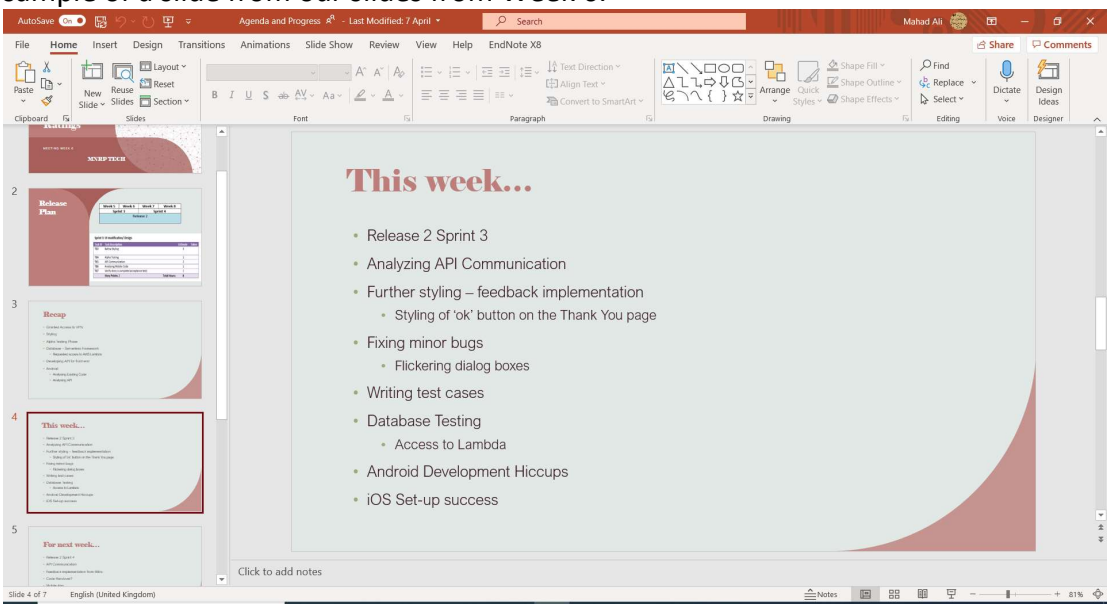


Figure 2.2.3: Slide 4 from Week 6 Client Presentation – "Agenda and Progress".

There were also times where the client had to cancel meetings last minute in which case, we would write up a Progress Update Report and include all the things we would normally talk about in an actual presentation. More information on how we managed such cancellations and their implications can be found in Section 3.3. Figure 2.2.4 displays sample of one such report from Week 7.

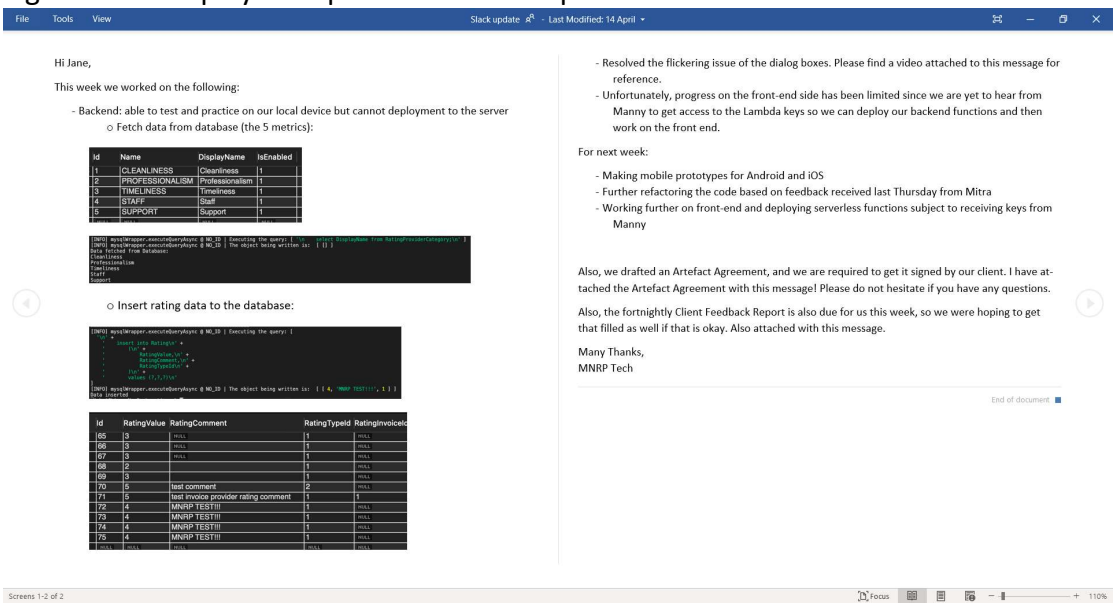


Figure 2.2.4: Week 7 Progress Update Report.

Then, we would showcase the work completed in that week to the client including displaying the working progress of the MVP of the current release, any bugs encountered, and the steps being taken to resolve them, either demonstrating the

feedback implemented from the previous week or eliciting feedback and finally, validating the MVP.

2.3. Team Collaborations

As mentioned earlier, we had internal team meetings set for every Friday around 9 in the morning. This is where we would usually communicate how we would follow up on the outcomes of the client meeting from two days prior on the Wednesdays. This would include allocating tasks based on expertise and preference, updating the project manager and discussing availabilities if a meeting needed to be set up with Mitra. Moreover, the Dev team would also take this time to explain technical details to the project manager since a lot of details about the project were very techs savvy.

Also, we also had an internal group chat set up on Messenger. This was mainly used to communicate updates once a task was done. Usually, Sunday was the day when everyone was required to send their updates on the chat regardless of whether the tasks were complete or not. This way, all team members had a clear idea about the progress thus far and if a team member needed assistance in any way. Sometimes Messenger chat was also used to establish our availabilities where we would make liberal use of the inbuilt poll option to figure out the best times that suited everyone. The polls were also used to make any other decisions collectively as a team. Plus, the chat was also used to raise concerns or query each other about anything project related. Finally, we would also use the chat to talk about job opportunities for students and sometimes, to give each other weather updates. This work life balance curated a positive work culture.

Moreover, we had also set up a shared OneDrive folder where we upload relevant project material like Social Contract or Artefact Agreement to ensure everyone had access to such important documents. The weekly client presentations were also uploaded ahead of time so team members could have a look and provide feedback as needed. For coding the team used BitBucket to collaborate and all the meetings, be it client or internal, took place over Zoom.

2.4. Communication Plan

STAKEHOLDER	ROLE	FREQUENCY	CHANNEL	Goals	Tools
Jane Sheehy	Client, Product Owner	Once a week	Client's office, Zoom, Slack	1) To update client on Progress 2) To validate MVPs 3) Elicit feedback on MVP 4) Send relevant files, documents and reminders	PowerPoint Presentations, Outlook Calendar

Mitra	Client I.T Team, Tech support	As needed, usually once a fortnight	Zoom	1) Validate Developed Code 2) Elicit feedback on Code	BitBucket, Outlook Calendar
Shea Burland	Tutor, Scrum Master	At least once a fortnight, as needed	Zoom, Email	1) To demonstrate how the progress is on track 2) Clarifying key points around project management 3) Communicating risk occurrences	Outlook Calendar
Internal Team	Project Manager, Front End Developers, Backend Developer	Twice a Week	Zoom, Messenger Chat	1) To divide and allocate tasks 2) To update other members of the progress 3) Making Team Decisions 4) Garnering team member Availabilities as needed 5) To ask relevant questions about the tasks 6) To raise issues	OneDrive, BitBucket

3. Project Plan and Risk

3.1. Project Planning and Progress

To manage and plan the project, the team came up with Sprint and Release plans at the start of the semester which comprised of a set of requirements laid out by the client. A set of these requirements, which would aim at the functionality, design, testing and validation of an MVP, was called a Sprint. As mentioned earlier, a Sprint would usually last for about two weeks and the completion of two Sprints would mark the end of a Release. We originally aimed at completing each stage with each release. In other words, during the first release, over four weeks, we would focus on the feedback input mechanism. The next release would cater to storing the feedback and the last stage of the project, feedback surfacing would be tackled in the last release thereby completing the project within 12 weeks. Figure 3.1.1 shows once such example of a sprint plan is attached in Figure 3.1.1 below.

◢ **Sprint 2: Make screens for the web development using react**

Task ID	Task Description	Estimate
T07	Clone the client's BitBucket Repo and create test branch	2
T08	Analyse the client code	1
T09	Create react screen components.	2
T10	Checking database to store star ratings	1
T11	Creating a table to store Star Ratings	1
T12	Validate with Jane (acceptance test)	1

Figure 3.1.1: Sprint 2 Release 1

As for the tracking of the project, we originally planned on assessing the sprint plans once a fortnight under the assumption that every release would deliver products as anticipated in our plans. Due to various reasons, the project plans had to be reassessed every week based on the progress we had made and the then outstanding work indicated by burndown chart. The factors that influenced the revision of sprint plans and their further implications have been elaborated in Section 3.2 and 3.3 while the burndown chart can be found Section 4.4.

3.2. Risk Management

Figure 3.2 shows the Risk Register made at the start of the project where risks associated with the project were highlighted, their impact and likelihood assessed and the strategies to either contain, mitigate or deal with the risk occurring were put into place. It is also worth mentioning that some of the risks, particularly the ones related to the client and their requirements were identified based on the events of phase 1 of the project. Plus, some of them were also related to the impact COVID-19 had on our project in Phase 1.

Out of the risks accounted for, four of them came into play during the project. Their identification, analysis and response been summarised below.

3.2.1. Last Minute Meeting Cancellations:

We had understood from Phase 1 that our client was a busy person and that they would have to cancel meetings last minute. Therefore, we were always ready to compile a report for to update our client and send it to them via Slack. This often meant that the client would not be able to provide us with feedback for that week so that had to be followed up on the next week. In order to still run the project efficiently, we would then start working on tasks from the next sprint in line in order to even out the workload for those weeks.

3.2.2. Falling behind on Project Plan:

Despite our efforts to be consistent and timely, the project did start to fall behind towards the end of the semester. This was mainly because we had to wait for many days, if not a whole week for the client technical team, Mitra to

grant us relevant privileges and accesses for instance the ability to execute code on their repository or AWS Lambda Keys to access backend. We would then have to either message the client to request a nudge to the technical team. This slowed down our progress since we could not test our MVPs from the first two phases in conjunction with each other until week 9. Consequently, we had gone over our expected time of 8 weeks of development but since we had left at least 3 weeks for user testing and client demonstration, we managed to speed up the development by writing code in local environments and then transferring it later on. This greatly reduced the impact of the risk and helped us deliver a fully working product by the end of week 12 with client showcase in week 13 since our week 12 meeting was cancelled.

3.2.3. Team Member Suffer Unforeseen Trauma

One of our team member suffered the death of a close family member and as a result, could not attend the client meeting that week. However, this was swiftly communicated across to the team and the client. As a result, other team members undertook different parts of the role of the affected team member. Therefore, it did not hinder progress at all and was dealt with professionally and proactively.

3.2.4. Changing Client Requirements

After week 8, the client decided to make some changes to the requirements of how the feedback would be surfaced. Instead of aggregated data, they wanted to display the most recent data. This change was a minor styling change and therefore, was accommodated with a little more effort. It did not hinder the progress either since that week we were waiting on Mitra to grant us access to the client server so we had the time and the space to accommodate the design and functionality of this new feature.

Risk	**Impact**	**Likelihood**	**Mitigation**
Border Closure	Moderate; Client not being able to come to work	Likely	Transition back to online meetings
Team member contracts COVID	Minor; The affected team member will have to isolate and may not be available for meetings in-person	Unlikely; Brisbane has been COVID-safe for a while	Include the affected team members using online platforms
Unforeseen/last minute meeting cancellations	Moderate; It will impact the Sprint timeline	Moderate	1) Rescheduling the meeting with the client as soon as possible 2) Collaborating with the client

			via text-based mediums like email or slack
Falling behind on Sprint Plan	Major	Unlikely since we already have a Sprint plan developed and host weekly internal meetings to track progress	Keeping track of Sprint plans and recording Sprint metrics to measure productivity
Unforeseen trauma or sickness suffered by team member	Major	Neutral; hard to say since such events are unforeseen	1) Early and honest communication with the rest of the team 2) Communicating with the client and the capstone faculty 3) Supporting the affected team member and working as a team to overcome the obstacle 4) Redistributing the workload
Variable and/or ambiguous client requirements	Major	Moderate; Client may want to change or edit some features so improve their product	1) Eliciting and documenting requirements 2) Verifying the requirements with the client during meetings before the Sprint 3) Requesting feedback at least every week

Figure 3.2: Risk Register made at the start of the project

3.3. Project Experience

As with any project, we faced some issues during the course of the development which have been discussed below.

3.3.1. Learning a new language altogether

SITUATION: None of our developers had any experience with or knowledge of AWS Lambda and React and the client required us to develop using those languages

TASK: The Developers were expected to learn the language and code using them.

Action: The team learnt the language online from YouTube and Stack Overflow.

Result: The team was able to successfully build the product

Learning: The team members had acquired a new and valuable skill considering that AWS owns more than 50% of the cloud market. Plus, they were forced into project-based learning and now have the ability to demonstrate their learnings as well. For next time, it may be beneficial to check with the client during the design phase about the languages or tools being employed in the project in order to get a head start on acquiring the required skill set.

3.3.2. Limited support from the client Tech team

Situation: The tech team of client, Mitra was offshore, operated in a different time-zone and had high response times.

Task: The client expected us to seek support from Mitra over unresolvable debugging issues and access rights.

Actions: Different actions were taken to combat this. Firstly, the team had to almost completely rely on Stack Overflow to resolve bugs that were actually originating due to the outdated version of the coding environments employed by the tech team. More on this to come in the next section 3.3.3. Plus, we had to revise and edit our Sprint plans to ensure progress did not come to a halt and we were making progress wherever possible while waiting on a response. Also, we had to often request the client to relay our message to Mitra in order to speed up the process.

Result: We eventually received the required access and guidance. Despite the fact that it delayed the progress, we were still able to deliver a working product.

Learning: We adapted different strategies to overcome this issue but what we found was that getting a superior, in our case Jane, to relay the message across was the most efficient way. We could have also tried obtaining the phone number for a member of the team but that could also be seen as

unprofessional since Mitra was not our direct client.

3.3.3. Different configurations on Android Studio

Situation: Mitra was not running the latest version of Android Studio which is why some inbuilt automation tools like *Gradle* or instance Identifiers like the *FirebaseinstanceID (FBID)* were causing problems. The problems ranged from our developers not being able to log in to the test developer account or running into unexpected errors which slowed down the progress as well.
Task: We were required to contact Mitra to let them know about this problem. Upon finally being able to reach them, we were instructed to download the older versions and configurations.
Action: We downloaded and imported the older versions into our Android Studio.
Result: We were able to finally resolve the errors and continue to test our code. However, we did lose about an entire week due to this towards the end which further extended the delivery date of the project. We communicated this to our client and they were very understanding and if anything, they mentioned they were impressed with our proactive efforts.
Learning: We realised that this error, before anything else, was a communication error on both the sides. Neither did we, as the contracted developers, probe and inquire about the development environment versions nor was it communicated to us in any way by the client. Therefore, for next time, if a project requires modification to the existing platform in any way, it would be worth getting the details of the development environment and the relevant settings or configurations.

4. Artefact

4.1. Functionality

In phase one, we had agreed with the client to collect the feedback from two different parts of the website or app. These were the Budget Category landing page and the Invoice page.

In this phase, we had to develop two different types of feedback captures. For the Budget category landing page, we were required to prompt the user with Star-Based ratings. These empty stars would be displayed along the name of every provider that the user has used the services of and so they could rate them out 5 stars in whole numbers. This feedback would then be stored in the Backend and used for different purposes outside of the scope of this project. As for surfacing, the most recent Star Rating would be displayed.

For the second feedback capture, it was placed under a specific invoice. There would be a button to prompt the user for feedback and when clicked on, it would trigger the pop-up box that consists of two steps. The first screen would ask the user to rate the providers based on five different metrics provided by the client. These ratings would also be captured out of 5-stars. Upon clicking next, the

qualitative feedback screen would pop up which ask the user to rate how likely they would be to recommend this provider on a scale of 1-10. Plus, there would be an optional textbox underneath to allow the user to explain their ratings. Once done, the user could then click on Next which would take them to the thank you screen and the feedback provided would be stored.

4.2. Architecture

Our client uses BitBucket to manage the repositories for both, frontend and backend. For frontend, they employ React and React Native while AWS lambda and Serverless framework are used for backend. Therefore, we were required to work with those as well. Figure 4.2.1 shows the overall architecture of our project.

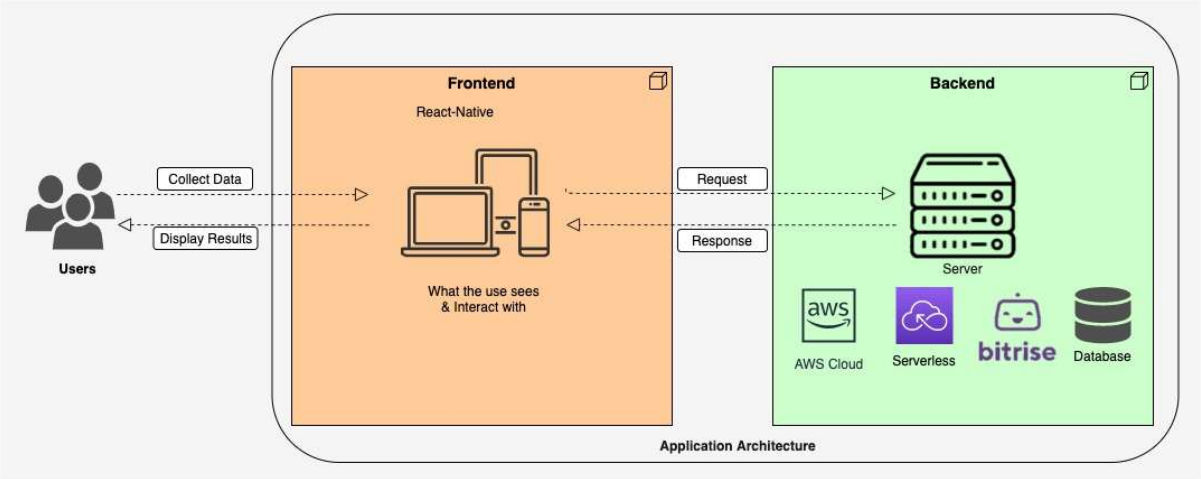


Figure 4.2.1: Overall Project Architecture

For frontend, the dev team developed across different folders inside the *src* folder shown in Figure 4.2.2. The main part of our feedback mechanism was developed in two main screens labelled *PaymentDetails.js* and *SpendingSupportItemDetails.js* in the *Screens* folder. They contain the source code of the Payment History page and the My Budget page which is where the feedbacks are gathered from. *PaymentList.js* in */Components/Molecules* was used to surface the feedback on frontend and *FeedbackPopup.js in /Components/Molecules* contains the mechanism of the feedback popup box that prompts the users to give feedback. For data communication, our team created the *StarRatingAPI.js* in */API* folder to communication with backend through AWS lambda.

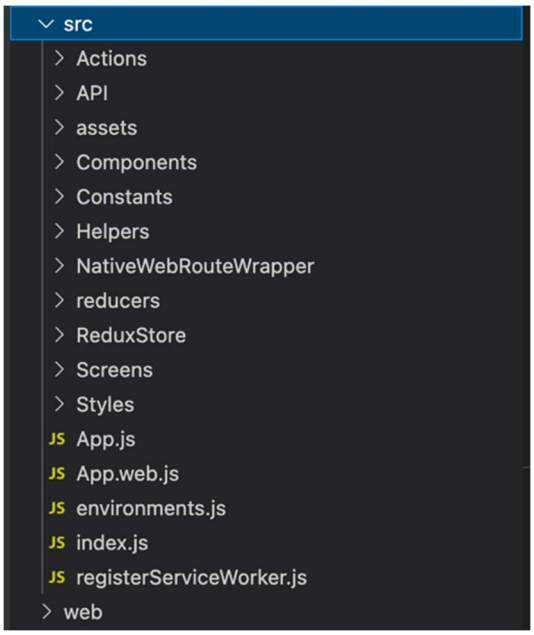


Figure 4.2.2: File Structure for Front-end

Figure 4.2.3 shows the backend file structure for the rating feature. The *ratings* folder is located inside */ServerlessLambda/lib* while the whole backend repository is structured under Serverless framework. Each folder represents a feature of the application and so our team developed inside the *ratings* folder. In the folder, *ProviderRating.js* contains the service class of the rating feature. InsertRating, getRatingMatrix and getRatingData are the serverless functions that operate the under rating mechanism. As for the *Rating.js*, *putRating.js* and *postRating.js*, they contain the script for the feedback mechanism that was originally developed by the client and their team. Moreover, the *serverless.yml* is the config file that coordinates with the service class and all the serverless functions.

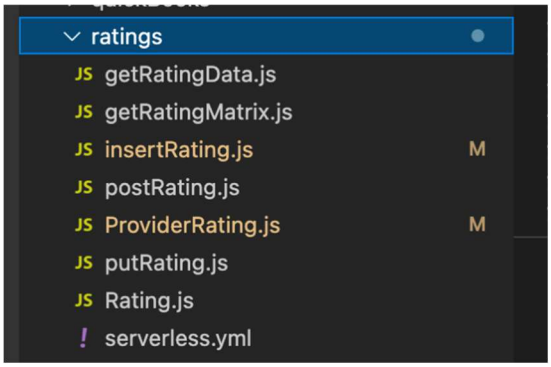


Figure 4.2.3: Backend File Structure

Although our client was using AWS Lambda for handling with the serverless framework, our team was granted access to Bitrise by the client instead of AWS Lambda deployment purposes. This was because Bitrise also does deployment with AWS Lambda so Mitra suggested that it would have been easier for us to work with Bitrise. Bitrise gave us the ability to deploy parts of the code as per our discretion at the time of deployment. Therefore, our team would deploy the whole rating feature to the backend every time we made any changes to the code.

4.3. Technical Description

The main task of the project was to develop a feedback mechanism on top of the existing system and surface the most recent rating for each provider. As per the status quo, the screen that users see are made up of several different components built by the previous developers.

These components are divided according to their size and complexity, ranging from the folder called *Atoms* which contains simple buttons to *Organisms* which would call API's and other components. Figure 4.3.1 shows the *components* folder.

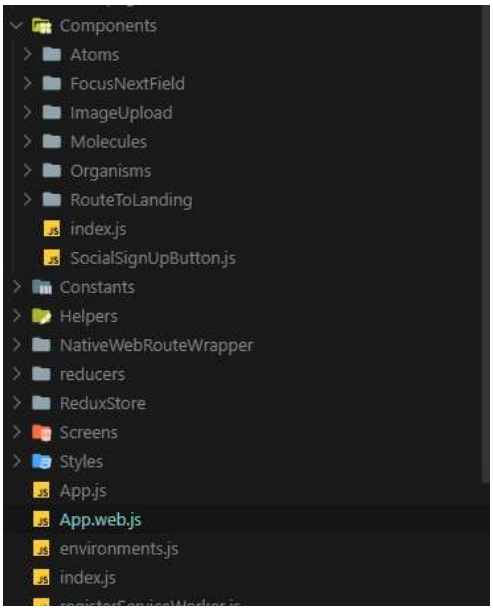


Figure 4.3.1: Components folder

The main popup mechanisms are placed into two different parts of the website and mobile app. These screens are inside the invoice of payment History and the Spending support category as shown in Figure 4.3.2 and 4.3.3.

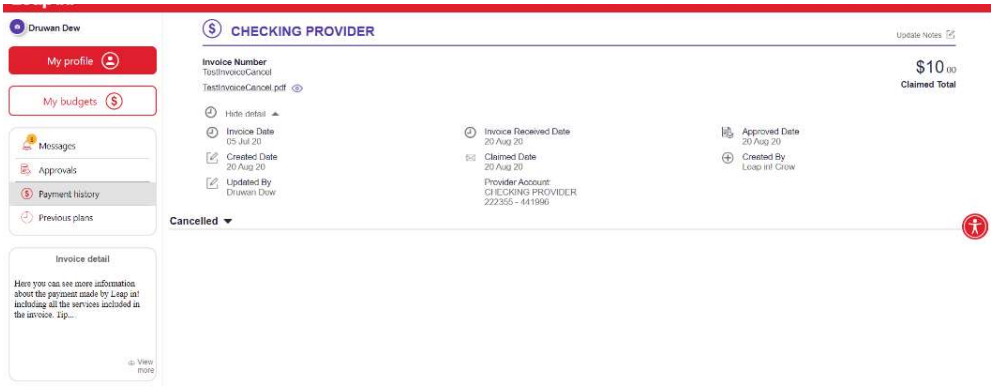


Figure 4.3.2

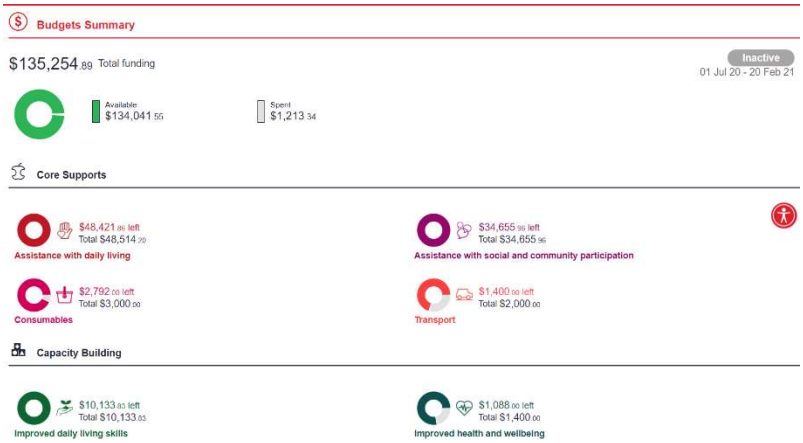


Figure 4.3.3

These screens manage the API calls and handle the PUT and GET request with the backend and pass them to the subcomponents to render and show the information. We render the feedback component inside the screens, and inside the feedback component we render the include the code for the API calls, buttons and inputs. The app is integrated with the two different frameworks i.e., React for the web version and React-Native for the mobile version. The backend part included serverless Lambda with MySQL database.

When we go into any page, the page sends a get request to the backend to get the information that we want to show and renders it inside the frontend function according to our need. When we store the feedback, we store it using the unique provider key, that is stored with the ratings when feedback is input, so it can be easily retrieved. The same procedure is followed when we press the submit feedback button. It retrieves the provider key and stores it in the tables inside the MySQL server. The Backend function developed are deployed, so that the front end components can make use of it. For the deployment, we have two options, either using AWS Lambda or Bitrise. Our team was given access to the Bitrise platform, from where we can deploy the specific function of our need. This is all depicted in Figure 4.3.4

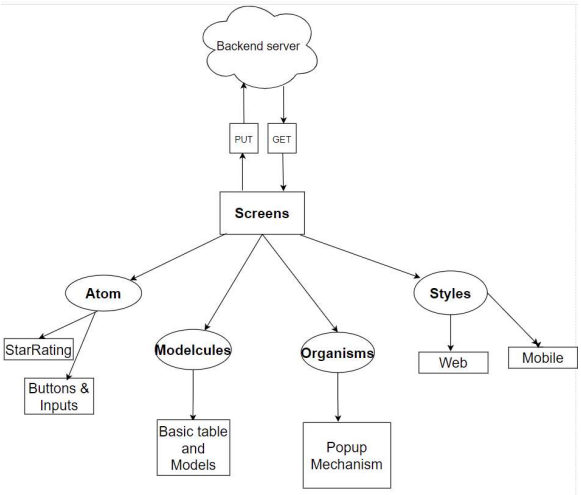


Figure 4.3.4

Thus, once the function was developed and deployed, the frontend API could then establish communication with the backend and either store or grab the feedback the components in the front or client side of the application. Appendix demonstrates an example of the components working.

4.4. Quality and Metric

In this section, summary information of developing Quality and Metric will be presented along with the tools used such as burn-down chart, description of code quality, unit test and progress of the actual development. Our challenge was to create a new feature on their existing website and mobile application (including Android and IOS device). As shown in figure 4.4.1, our weekly challenge was been divided into 6 core processes that were Internal meeting (MNRP IS & CS), Technical design project, Develop Feature (Coding), User Feedback, Unit test and Improvement. Each process was executed every week while the following week except for unit testing which was conducted the following week which included assessing the quality of the code as well. We aimed at finishing major developments until week 8 as shown in the burn down chart in Figure 4.4.2.

Feature	Initial Estimate	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Hours Left
Internal meeting(MNRP IS & CS)	12	1	1.5	1	1	1	1	1	1	3.5
Technical design project	20	4	4	4	3	1	1	1	2	0
Develop Feature (Coding)	40	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4
User Feedback	20	2	2	2	2	2	2	2	2	4
unit test	12	0	0	2	2	2	2	2	2	
Improvement	16	0	2	2	2	2	2	4	4	
Settting	**Start**	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	
Planned Hours		11.5	11.5	11.5	11.5	11.5	11.5	11.5	11.5	
Actual Hours		11.5	14	15.5	14.5	12.5	12.5	14.5	15.5	
Remaining Effort	120	108.5	94.5	79	64.5	52	39.5	25	9.5	
Ideal Burndown	120	108.5	97	85.5	74	62.5	51	39.5	28	

Figure 4.4.1: Burn-down 6 core process

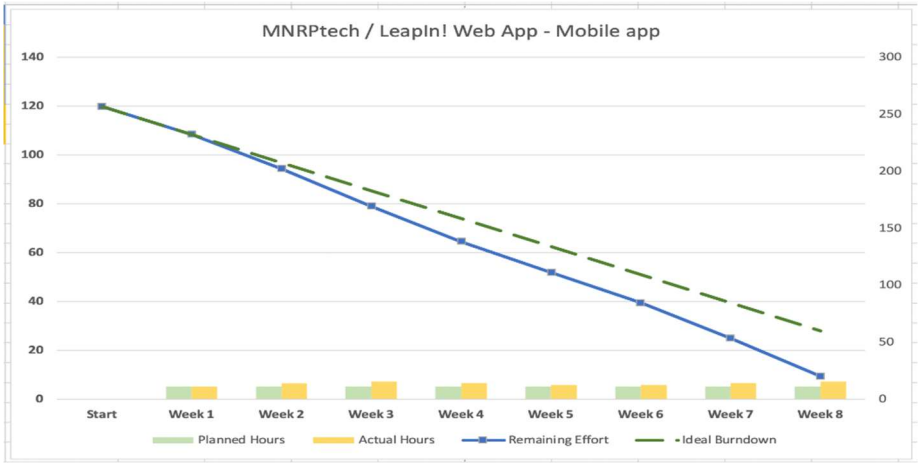


Figure 4.4.2: Burn-down Chart

Internal meetings were conducted before the client meetings every week. Usually, the time was spent on sharing technical feedback and design project structure between all members including sharing ideas, assessing current plans and communication between the project manager and the dev team.

In the beginning of phase 2, the team had to draw an overall technical plan and user interface for this project and present it to the client. This core process conducted big amount of time in the beginning to make sure the client expectations were at par with our understanding of the technical design.

This Feature Development showed how much our time was spent on actual coding on React (Web) and React Native (Mobile). First two weeks were spent setting up Development environment using the tech stack (Bitbucket, lambda, Git, making our Own separated brunch). Then average 4.5 hour per week was spent on developing the required features.

User feedback was one of our critical processes since we are agreed with having weekly client meetings. With the material we prepared, we joined meetings with the presentation slides, the coding structure and progress on code that was then validated by the client.

Further unit tests were done with Mitra and they would give us Technical support and review code to assure the quality of code as well as their Software developing style. One such example is shown in Figure 4.4.3.

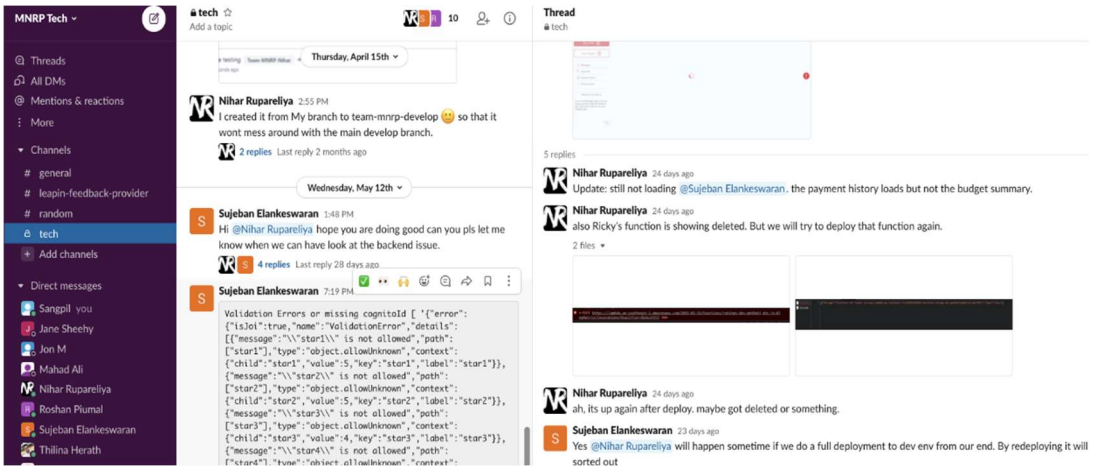
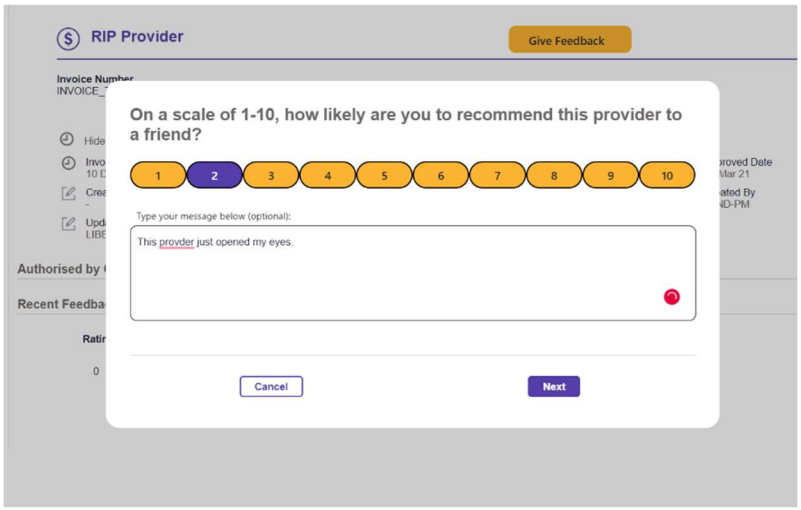


Figure 4.4.3: Slack channel getting feedback of unit testing

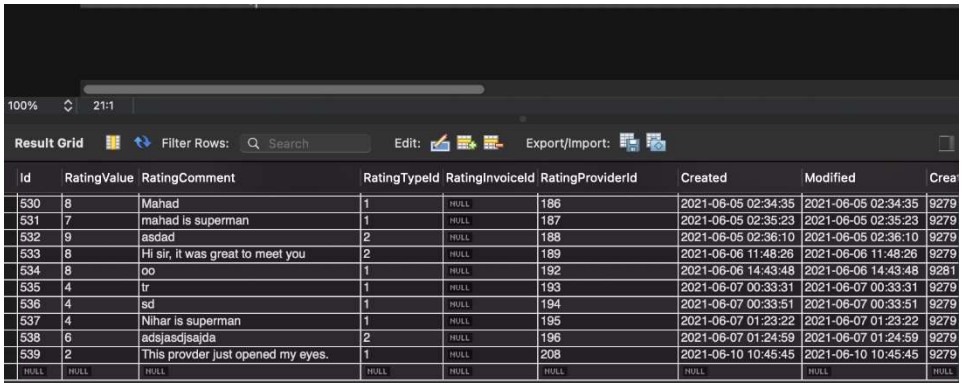
Last process of the week was improvement of design and code from given feedback. Instead of having daily scrum meeting, we held an improvement meeting to share the idea of given feedback from weekly face to face meeting with our client.

To sum up, Quality and Metrics for our project were ideally conducted and we got feedback during face to face meeting by showing either actual design UI or function code to our client. After week 8 we mainly focused on code quality to ensure we were following their standard developing style and organized a code-review with Mitra and dynamically communicate with client.

APPENDIX:



That star given out of 10 is divided by 2 and shown in the recent feedback after you submit the rating.



Result Grid | Filter Rows: Search | Edit: | Export/Import:

Id	RatingValue	RatingComment	RatingTypeId	RatingInvoiceId	RatingProviderId	Created	Modified	Crea
530	8	Mahad	1	NULL	186	2021-06-05 02:34:35	2021-06-05 02:34:35	9279
531	7	mahad is superman	1	NULL	187	2021-06-05 02:35:23	2021-06-05 02:35:23	9279
532	9	asdad	2	NULL	188	2021-06-05 02:36:10	2021-06-05 02:36:10	9279
533	8	Hi sir, it was great to meet you	2	NULL	189	2021-06-06 11:48:26	2021-06-06 11:48:26	9279
534	8	oo	1	NULL	192	2021-06-06 14:43:48	2021-06-06 14:43:48	9281
535	4	tr	1	NULL	193	2021-06-07 00:33:31	2021-06-07 00:33:31	9279
536	4	sd	1	NULL	194	2021-06-07 00:33:51	2021-06-07 00:33:51	9279
537	4	Nihar is superman	1	NULL	195	2021-06-07 01:23:22	2021-06-07 01:23:22	9279
538	6	adsjasdjsajda	2	NULL	196	2021-06-07 01:24:59	2021-06-07 01:24:59	9279
539	2	This provder just opened my eyes.	1	NULL	208	2021-06-10 10:45:45	2021-06-10 10:45:45	9279
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 1 Backend MySQL table

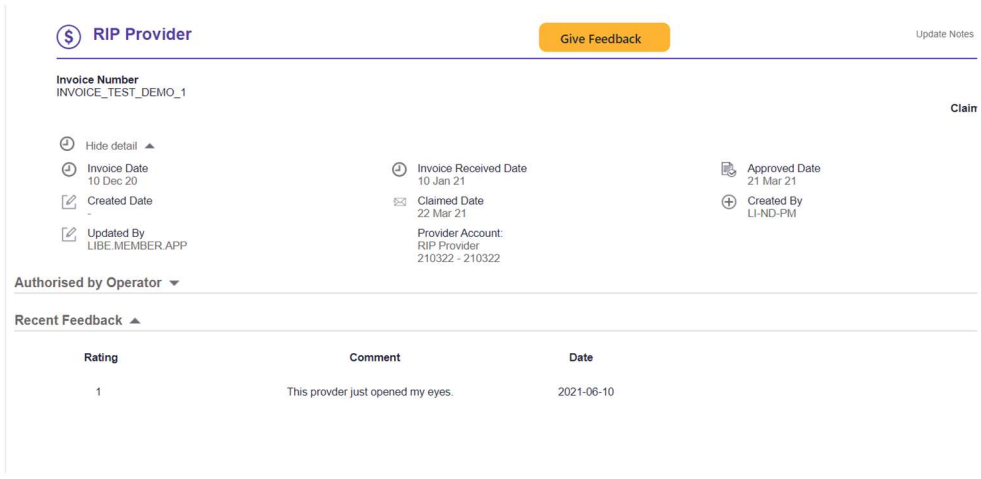


Figure 2 Front-end Sufacing of the feedback