ICCS-CWIE007

COOPERATIVE AND WORK INTEGRATED EDUCATION'S REPORT

by

Name (in English) Karan Kumar

Student ID No. 6380812

Enrolls in a Cooperative and Work Integrated Education at Agoda Services Co. Ltd.

Adderss No 999/9 Sub-district Pathum Wan District Pathum Wan

Province Bangkok Postcode 10330

This report is submitted as a part of enrolling in a Co-operative Education of Computer Science Program, Mahidol University International College

ACKNOWLEDGEMENT

I have enrolled in a Cooperative and Work Integrated Education by working at the company named Agoda Services Co. Ltd. From Date 7 Month August Year 2023 to Date 15 Month December Year 2023

I have gained much professional experience and valuable knowledge which could not be found in classrooms. This experience and knowledge will be adapted and applied to my learning and future carreer path. Though, this Cooperative and Work Integrated Education Education's Report could not be done without the help, coorperation and encouragement from many collaborations. So, I would like to express my grateful feeling toward people as follows.

1. Ankita Kujur
2. Pakorn Thanaprachanon
3. Ayon Bhattacharya
4. Venkatesh Harisankar
5. Wisit Saengkaew
6. Sittikorn Hirunpongthawat
7. Nunticha Prasertadisorn
8. Nattapoom Dumrunglaohapun
9. Witsanu Sirichanyaphong
10. Sorawit Namseetan

I also would like to thank the entire Fintech Team and I also would like to thank you to those who are not mentioned in this report for your guidance, suggestion and understanding towards completing this report.

Thank you for your every kind of support, I really do appreciate.

Signature...

Karan Kumar

Date............/................./................

Abstract

The company Agoda Services Co. Ltd. is the company that is an online travel agency where customers are allowed book for hotels, private stays, flights (with hotels included), activities, airport transfer and many more. There are also services for hotels to help communicate with the company to help with promotion and discounts.

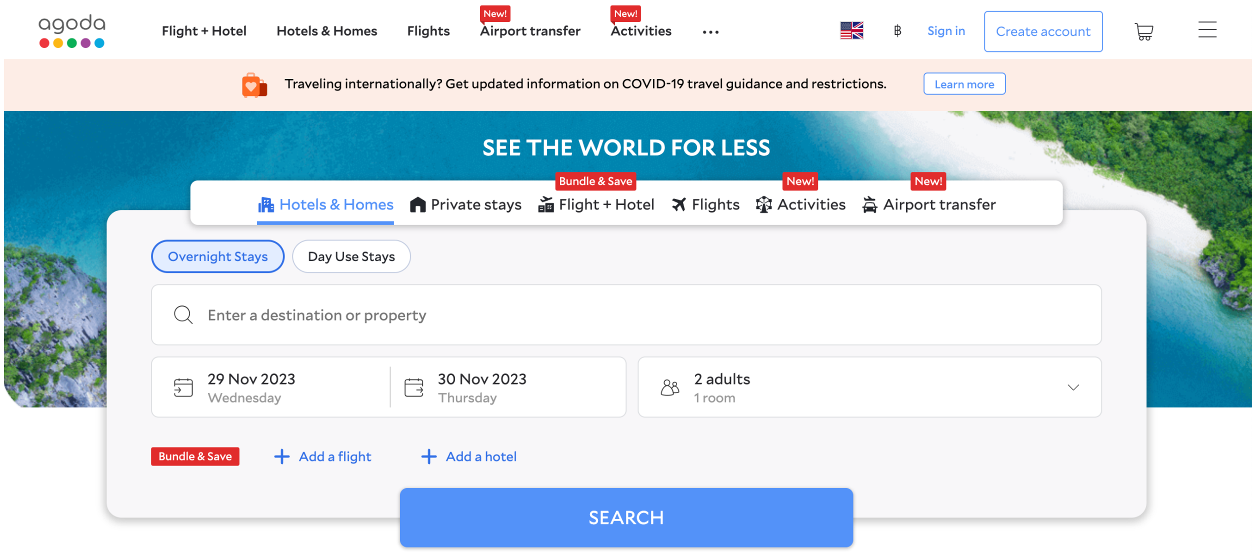
I have enrolled in a Cooperative and Work Integrated Education by working at the company named Agoda and I am assigned to work in a Department of IT Finance that is split into multiple teams which are Fintech Tax, Fintech Infra, Fintech Tactical, Fintech Funnel and Fintech Data where the team I mostly was managed by was Fintech Tactical team. By doing so, I have learned a great foundation to what it takes to become not only a knowledgeable backend engineer but also how to develop my soft skills for better communication with my fellow team members. Furthermore, I have also learned how to be better at multitasking and time management as there can always be urgent tasks to prioritize or meetings to attend which can overwhelm one quickly if they don't have a plan on what to do.

Lastly, I have also kept track of my daily performance to use that information to analyze and improve my weaknesses resulting to a greater understanding on what role I fill in the team and what improvement and adjustments I can make not only as a better team member and engineer but also a person. This experience has made me grow a lot and now I have a good foundation and idea what I can expect in my future companies I work in.

1. Introduction

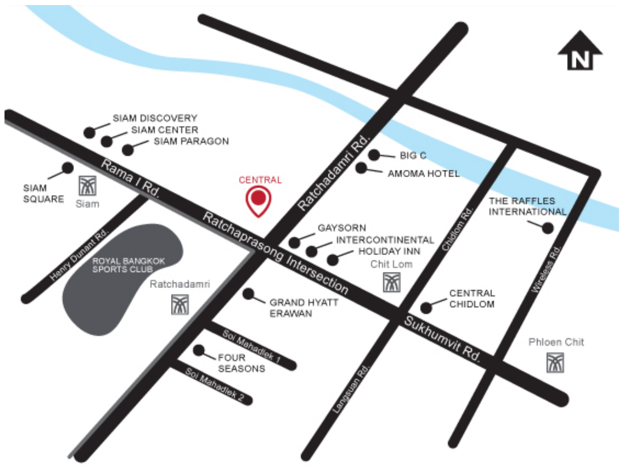
1.1 Workplace Information

Agoda Services Co. Ltd is one of the largest online travel agencies that provides a platform for users to book for hotels, properties, flights, activities and much more. It has networked with over 2.9 million properties globally consisting of over 200 countries and territories.


Traveling internationally? Get updated information on COVID-19 travel guidance and restrictions.

As the company operates globally it has catered to customers globally as well by providing reservation in 39 different languages that understands the local culture uses local connection to its best to provide the best prices affordable of all kinds of customers.

Regards of the company's operations, it has employed over 7000 professionals in this area consisting of 90 different nationalities which promotes a great diversity and creativity to provide the best experience for customers and feel at home when using the service.



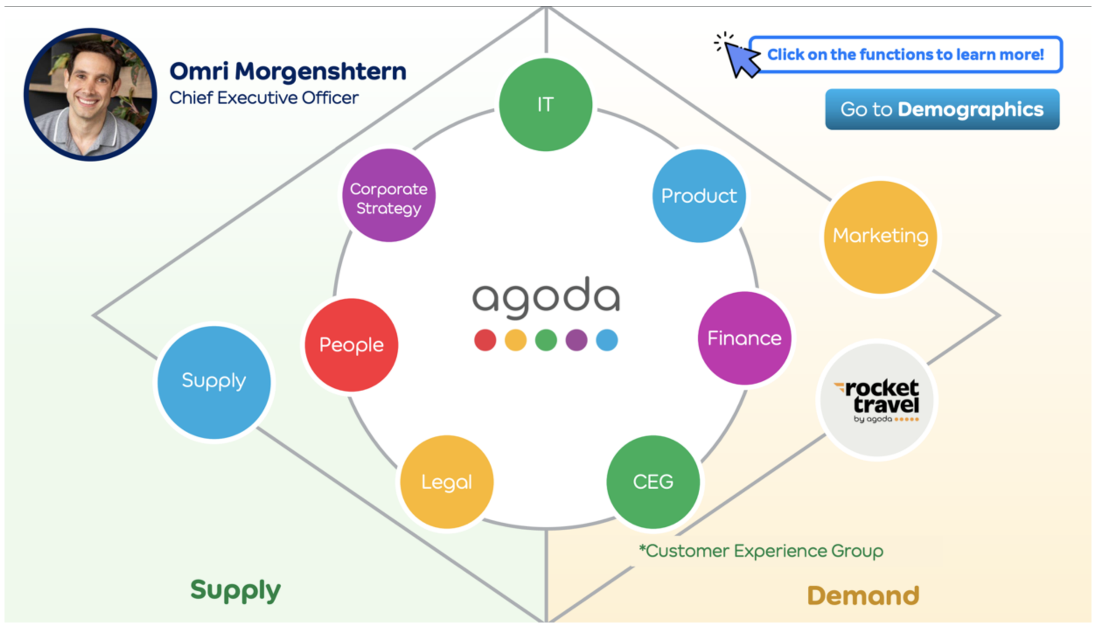
Located at the heart of Bangkok, Central World Offices allowing many employees to have a convenient way to transport to but also experience the culture of the country has to offer for many of the employees who are from overseas.

1.2 Organization Overview

There are many departments in Agoda all assigned with their responsibilities to help Agoda keep their business running daily, since some of the departments are more demanded in the company the ratio in number of employees between different departments differ. All of the departments are overseen by our current CEO, Omri Morgenshtern.

Departments in Agoda consists of

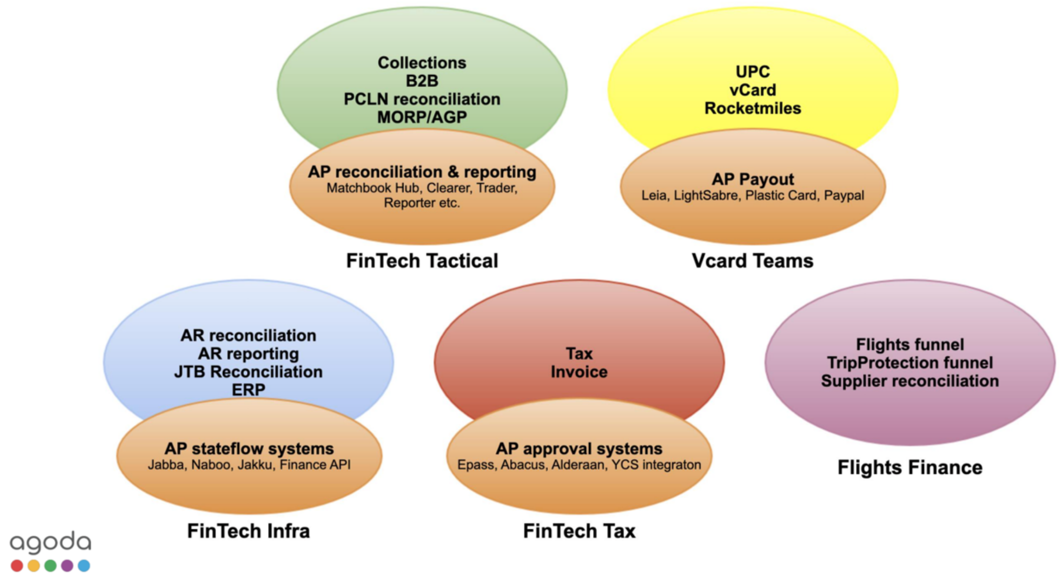
- Information Technology or Tech (23%)
- Product (4%)
- Finance (3%)
- Customer Experience Group (51%)
- Legal (1%)
- People (3%)
- Corporate Strategy (< 1%)
- Supply (12%)
- Marketing & RTA (4%)



1.2 Department Information

I was assigned to be in the department of IT but to be more specific, the IT-Finance department or Fintech. The primary responsibility of this department to help ease the process and help the Finance team go through by developing and maintaining financial systems for Agoda.

The Fintech department is split into multiple sub teams for easier organization and specialization of financial tools.



1.3 Fintech Core Intern Team Information

I was assigned to be a part of the Fintech Core Intern Team or ICore team for the duration of my internship to collaborate and create new projects or help maintain existing systems and services e.g., Investigate bug on one of the services repositories. The team consists of software engineering interns of Agoda. The intern at the current moment is managed by the Fintech Tactical Team or in short, Tactical. Throughout my internship there were 4 full time intern and 1 part time but note that this number will never be the same as it depends on the period and hiring decisions by the Tactical team.

The ICore team were assigned projects that covered both frontend and backend to maximize learning and experience.

Mentors:

- Ankita Kujur (Software Engineer)
- Pakorn Thanprachanon (Associate Software Engineer)

1.4 Company Contact Information

- Company Name: Agoda Services Co., Ltd.
- Address: No 999/9 Road Rama 1 Sub-district Pathumwan District Pathumwan Province Bangkok Postcode 10330
- Mobile no: 02-625-9200
- Telephone no: 02-016-4245

1.5 Co-operative Education Program Information

My position assigned during the Co-operative Education Program was Software Engineer Intern where the official period of this program is from August 7th, 2023, to December 15th, 2023 (20 Weeks).

The Co-operative Education Program was full time therefore the official working hours were 9:30 AM – 6 PM from Monday to Friday. Regardless of the official working hours set, the only requirements were to complete 8 hours of work every day excluding break and standup everyday with the team.

Agoda does support flexibility but following the set working hours is highly encouraged for the most productive experience with the team an employee has been assigned. Furthermore, Agoda allowed working from home 3 days a week and working onsite 2 days a week to help work-life balance and communication with team.

The ICore Team and the Tactical team have set a routine to come and work in the office on Tuesday and Wednesday while the rest of the days can be work from home.

1.6 Co-operative Education Objectives

- To develop soft skills such as communication and time management
- To be more exposed and develop skills with different tools used in software development.
- To be able to apply Computer Science foundation into real life applications through being assigned hands-on projects and tasks.
- To learn and connect with more experienced software engineers.
- To understand and be a part of Agoda's work culture.

Chapter 2. Literature Review

1. Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) is the primary tool for data storage best suited for Hadoop applications. This distributed file system takes advantage of NameNode and DataNode architectures to allow a for better performance in accessing data across different Hadoop clusters. It provides a reliable solution for managing pools of big data.

This tool was used to help deploy the services Resource Monitoring Notifier Service and Scheduler Job Notifier Service along with other services that exist in Generic Monitoring Service project where it gets assembled as a .jar file and place under the path in file system of HDFS navigated with the help of HUE.

HUE or Hadoop User Experience is a graphical interface that makes it easier for the user to interact with the Hadoop. Users can easily browse, edit and add files with this tool. The reason the Generic Monitoring Services .jar file can run every 10 minutes automatically is because it has access to creating oozie workflows and coordinator.

2. GraphQL

GraphQL is an open-source data query and manipulation language for implementation Web server-based APIs. Great thing about GraphQL is that is a syntax developers can use to ask for specific data and return the same exact customized data from multiple different sources. This tool has been a great alternative to the traditional REST-APIs.

I have used a lot of this tool during the project of implementing AP Reconciliation Approval Adjustment Page to help fetch different kinds of data in a customized fashion.

Chapter 3. Methodology

3.1 Agile Process and their ceremonies

Agile is the project management approach used and integrated deeply into not only our team but many IT Teams here in Agoda. Agile process being used ensures projects and tasks are done on time by setting deadlines and estimated timeline for each of the small tasks to help successfully build the project. Additionally, it allows for greater collaboration as everybody in the team would know the current state of everybody else and help out where needed. The Fintech ICore team integrated 5 methods of Agile Process into our team:

- Daily Stand-up – A short meeting usually taken place in the beginning of every workday to allow every person in the team to update their progress on the current tasks and if any assistance is needed from the mentors in our team. Our team had daily standup every day at 9:30 am.

- Sprint Retrospective – Occurs simultaneously with happiness and sprint planning, where the interns can reflect on what went well for the team or individually, what could be improved on and if there's any brilliant idea to share to the team to make the next sprint or one of our services better.

- Happiness – Occurs simultaneously with sprint retrospective and planning, this is where the interns in ICore Team can give feedback to the mentors of their working condition the past sprint through a small survey covering the following areas:

    - Estimation – Was the individual satisfied with points estimation given to their work on Jira.
    - Work Duration - Was the individual satisfied with their working hours.
    - Sprint Goals – Did the individual feel the sprint goal was achieved.
    - Personal Goals - How satisfied is the individual with themselves.

- Time for MR Reviews – How much time was given to the individual to help get MR reviewed.
- Support from other teams – How much support did the individual get from other teams.
- Learning – How much did the individual learn.

- Sprint Planning – Held at the end of every sprint or every 2 weeks, this is where the interns and mentors go through each uncompleted tickets of the previous sprint and asks for progress and if it needs to be carried out to the new sprint. Once that is done, the mentors evaluates if there are enough tickets for the new sprint and tries to add new projects or tickets to make sure everyone in the team has equal and adequate workload.

- Sprint description Breakdown – Held the day after sprint planning this is where the interns gather up in a short meeting to try out fill out and give context to every ticket added to the new sprint so any intern once they have enough bandwidth can pick it up and not have to bother the mentors for briefing or more information.

3.2 Application & Tools used for Communication.

3.2.1 Microsoft Teams

Microsoft Teams served as the primary tool to hosts virtual meetings in Agoda, it also serves as a way of communication to different employees for some of the teams in Agoda, but it wasn’t the case for our team. Microsoft Teams also allowed to manage and schedule different calendars to make sure there weren’t conflicting meetings. Lastly, it had a deep integration with Agoda where each of the meeting rooms in the office would sync with Microsoft Teams to ensure the teams that booked a certain had full access to only the room onsite and online.

3.2.2 Slack

Slack is the primary tool for communication outside of meetings for employees for Agoda but mostly in IT department. It allows for two direct communications between employees which is through direct message or voice call (known as huddle).

Another great thing about Slack is that it allows to create text channels for teams to communicate together to help resolve issues and brainstorm or larger channels for support or announcements. Lastly, slack also supports many automated messages to be sent through their own Slack API for convenience and efficiency of many teams in the company e.g., server status checker bot that pings the appropriate team if it’s down or spiking.

3.2.4 Microsoft Outlook

Outlook is the primary medium for company's announcement or more formal communication to be done e.g. permission to access. It's integration with Microsoft Teams also makes sure that no employees miss upcoming meetings.

3.3 Application & Tools used for Development.

3.3.1 Jira

Jira is a propriety tool used by many Agoda IT teams as it is one of the best tool for agile project management therefore this allows each teach members to manage and plan their tasks during the current sprint. It provides support to be integrated with GitLab as well to further ease management of work and development of a project.

3.3.2 GitLab

GitLab is a web-based Git repository that also provides a DevOps platform that allows developers at Agoda to develop, maintain and secure a software. Since it has DevOps platform it allows for convenient development for CI/CD to automate some of the key steps e.g. deploying to product.

3.3.3 Confluence

Confluence is a crucial tool for many developers in Agoda especially for those who are new to the services and company, it houses documentation for many of the services that is built in-house by Agoda which could be seen as a knowledge sharing platform. It is also organized based on department and teams to easily navigate relevant documents.

3.3.4 IntelliJ IDEA

IntelliJ is an integrated development environment best suited for developing software written in JVM-based languages e.g. Java, Scala and Kotlin

3.3.5 WebStorm

Webstorm is an integrated development environment for JavaScript and technologies relating to frontend web development providing very convenient development experience.

3.3.6 DataGrip

Datagrip is an integrated development environment designed to work with databases hosted locally, on a server or on the cloud aiding in faster development in writing queries and scripts e.g. SQL

3.3.4 Docker

Docker is a service platform that virtualizes OS-level to deliver software in packages called containers, this greatly helps developers run and test their services locally without having to deploy. Additionally, OS-level virtualization also ensures the services behaves the same in every environment run on through containers.

3.3.5 Postman

Postman is heavily used here in Agoda to help developers to test and build their APIs to ensure these endpoints behaves as expected. It is integrated with many different protocols for sending requests as well e.g., HTTP, GraphQL and gRPC requests. What makes it so heavily utilized it as well that it doesn't require any code to be written for sending requests.

3.6 Projects and Contributions

3.6.1 Scheduler Job Notifier Service

The Scheduler Job Notifier Service or known simply called as Scheduler Bot within the company is the very project I have been assigned to implement starting from scratch. This project asks to implement an alerting framework that alerts an appropriate team through the corresponding text channel with the help of Slack API regarding process or report failure.

To know when to alert what this service is basically is does is that it gets triggered by a scheduler every 10 minutes to keep fetch the statuses of the processes and alerts the team if needed based on the statuses received and the time it runs.

The benefits of having this framework implemented in Agoda are of the following.

- Maintainability – As all the configurations regarding the service is stored in the same project, it is easy to be able to rename the process name or their set schedule for expected run time and finish time. Not only edit the current existing process but adding and removing is as easy as editing in the appropriate place. Additionally, adding another team to alert or channel to track different process can also be done easily by specifying which team to tag during alerts and which channel the message should be sent and updated. For the channel to be sent the user must make sure that the channel to be sent is already present in the Consul config.

- Scalability - If there is another endpoint or another service to fetch the status for processes it can easily by done by implementing a different client (following the structure of existing clients e.g., SchedulerJobWhitefalcon or SchedulerJobLogKestrel) and adding the appropriate endpoint details to the config. With the updated configuration the service is now equipped to fetch and monitor processes from this endpoint alongside the existing sources.

- Faster response time – As majority if not all of IT Teams in Agoda communicate through Slack daily and have a lot of tasks on plate which means that they do not have the time to monitor dashboards or check logs to make sure if a process ran successfully. As Scheduler Job Notifier Service has integration with Slack API, it would alert the appropriate team right away if a failure related issue occurred with one of the processes being monitored by the service by pinging the team or a user. Pinging the team on Slack would allow actions be taken much faster as pinging with an appropriate causes a notification to be sent. More regarding alert will be talked in the implement of the service.

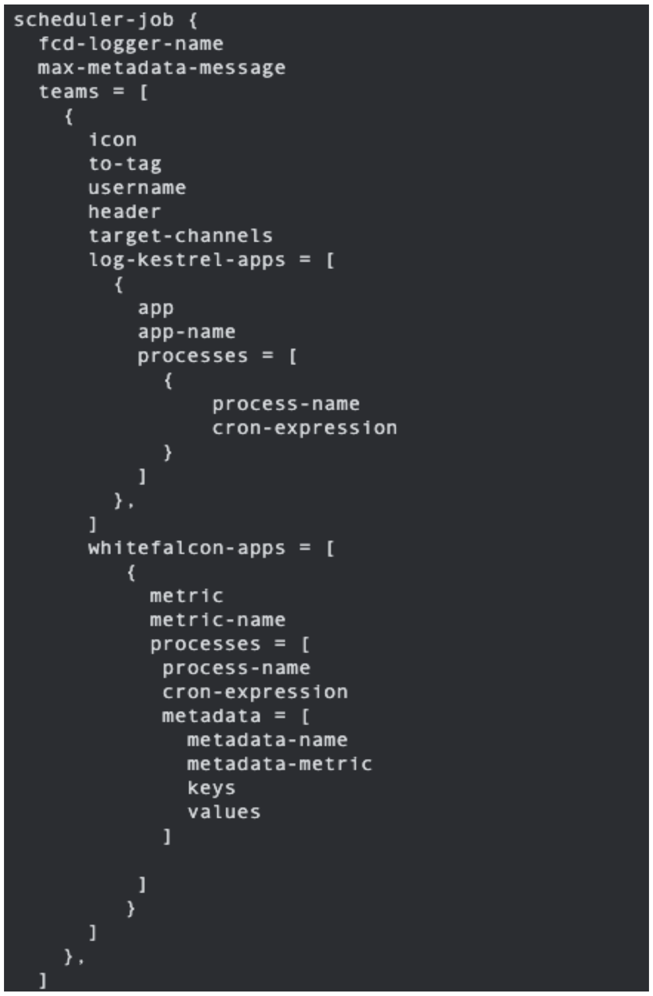
This framework will be implemented and be a part of Generic Monitoring Services that is a repository that has been created by the Fintech ICore Team to host many services integrated with Slack API for different alerting purposes that has long term planned to be used by many teams in Agoda.

At the current moment it has integration to receive statuses of reports and processes through 2 built-in house APIs from Agoda which are Log Kestrel and Whitefalcon.

Configuration:

This is the crucial part of the service as it will provide the appropriate information the service so that it can be able to run the service and track the process correctly and alert at the right time.

One notable thing that was kept in mind during the design of the configuration was to ensure that the service remains as generic as possible and scalable as possible therefore this following format was followed.



```
scheduler-job {
  fcd-logger-name
  max-metadata-message
  teams = [
    {
      icon
      to-tag
      username
      header
      target-channels
      log-kestrel-apps = [
        {
          app
          app-name
          processes = [
            {
                process-name
                cron-expression
            }
          ]
        },
      ]
      whitefalcon-apps = [
         {
           metric
           metric-name
           processes = [
            process-name
            cron-expression
            metadata = [
              metadata-name
              metadata-metric
              keys
              values
            ]

          ]
         }
      ]
    },
  ]
```

To briefly explain these dynamic configs, we will first start with the outer most configuration which are teams. This loosely translates to service as this whole Scheduler Job Service can run many sub services all running along side each other where it will have its own.

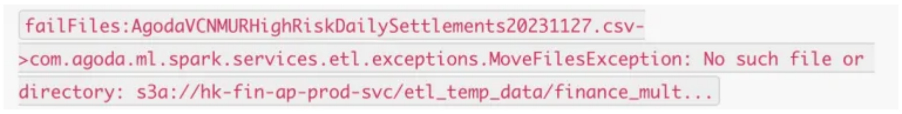
- icon = profile picture of Slack bot
- to-tag = who to tag when an alert is thrown by the processes
- username = username of the Slack bot
- header = what should be the base header for the bot message when displaying the lists of processes
- target-channels = which channels in slack should the message be sent to
- log-kestrels-apps = processes that are logged in Log Kestrel depending on the app
- white-falcon-apps = processes that are logged in Whitefalcon depending on the metric

To further explain the configs that are stored in different clients currently supported by the service, we will first start off with Log Kestrel Configs.

Log Kestrel Applications configuration firstly consist of the app which helps us in querying and filter to help find the processes data in Log Kestrel API, secondly is the app-name this is just for the display message to help with readability in the actual message. Lastly, are the processes that are going to be tracked where it will consist of the process name which also helps in further querying and filtering, and the CRON expression this helps lets us know the expected start and finish time of the processes so that the appropriate alert can be thrown if it hasn't finished by the finished time. One neat feature implemented is that the days that the CRON expression does not satisfy the processes will not be monitored this is to help not have redundant processes displayed in the message that's not even going to run and help with performance so the service can execute faster.

Next client configurations are Whitefalcon Applications. Whitefalcon applications requires a lot of configurations to be able to fetch appropriate status as many services that uses Whitefalcon measurement integrated has their own different keys to display so an attempt was to make it generic as possible. Firstly, is the metric this is to filter and locate where the measurements will be captured in Whitefalcon API next are the processes that will be tracked which follows the same format and reasoning as Log Kestrel however the difference this time is to fetch metadata. Unlike Log Kestrel where metadata is stored within the same logs returned in this case a different metric is needed to help fetch the appropriate metadata for the processes and there could be multiple metadata that would like to be queried. The reasoning behind of keys and values being placed in the configuration is that as I mentioned that output varies greatly in Whitefalcon for different metric filtering the appropriate information of each process also is unique for different metric.

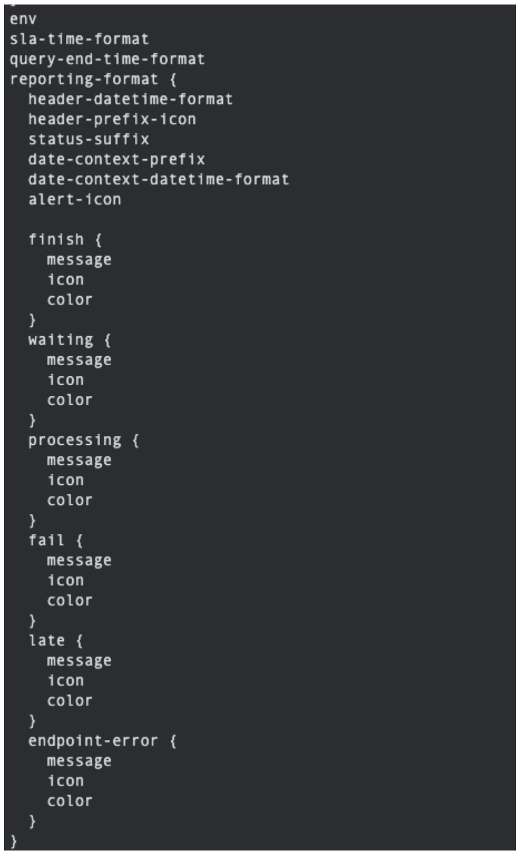
The fixed configs that are stored regardless of the service or application are the FCD Logger Name which is what Log Kestrel is currently integrated with which is FCD to help fetch statuses of the processes easily therefore it's remained fixed. Another is Max Meta Data Message which is to ensure that consistency of metadata character amount is kept within every service and applications like the following picture.



```
failFiles:AgodaVCNMURHighRiskDailySettlements20231127.csv-
>com.agoda.ml.spark.services.etl.exceptions.MoveFilesException: No such file or
directory: s3a://hk-fin-ap-prod-svc/etl_temp_data/finance_mult...
```

The rest of the config are dynamic and are needed to be set when a different new service needs to be added such as when needed to track different log kestrel apps or white falcon apps or have a different icon. These services will run along side yet treated independently so updating and sending the message would have their own service to do it.

There's also configurations set for formatting on the overall messages therefore the following configuration template was used.



```
env
sla-time-format
query-end-time-format
reporting-format {
  header-datetime-format
  header-prefix-icon
  status-suffix
  date-context-prefix
  date-context-datetime-format
  alert-icon

  finish {
    message
    icon
    color
  }
  waiting {
    message
    icon
    color
  }
  processing {
    message
    icon
    color
  }
  fail {
    message
    icon
    color
  }
  late {
    message
    icon
    color
  }
  endpoint-error {
    message
    icon
    color
  }
}
```

The configurations are only used in SchedulerJobNotifier implementation therefore their explanation and usage would be provided when the notifier implementation is discussed about.

These configs remain fixed despite the service to make sure there is consistency kept for maintainability and scalability purposes as a user who wants to onboard a different service does not have to edit and configurations regarding the formatting.

Implementation:

Scheduler Job Notifier Service are split into 3 main components which are of the following:

- Scheduler Job Client
- Scheduler Job Notifier
- Scheduler Job Service

Scheduler Job Client

Scheduler Job Client is the primary component in this service as this client is responsible for fetching the statuses of each process and reports with some processing to return the appropriate status in form of a case class, the following statuses case classes that the client can return are of the following:

- Waiting – The process hasn't started.
- Processing – The process is processing.
- Finished – The process has finished running successfully.
- Failed – The process has failed.
- Late – The process hasn't finished or started in it's set schedule.
- Endpoint Error – The endpoint to fetch process's data failed to respond.

To make this service generic so far this client has been implemented further to 2 sub-clients which are Scheduler Job Client Whitefalcon and Scheduler Job Client Log Kestrel so that this service is able to fetch the statuses of different kind of processes or reports that is logged in different services an exposed with each of their own API.

For scalability and maintainability, the following sub-clients must have the following functions so that it can coexist with the previous clients.

Method	Use
getProcessApp(): ProcessApp	Get information regarding the process being monitoring
getStatus(time = DateTime.now())	Get status of the process depending on the time specified if not then current time

getExpectedFinishTime: DateTime	Get the expected finished time of the process (from the configuration)
getFinishedTime: DateTime	Get the finished time of the process
getExpectedStartTime: DateTime	Get the expected start time of the process (from the configuration)
getDuration: Int	Get the minute difference from finished time to start time
getStartTime: DateTime	Get the start time of the process
isStatusGood(time = DateTime.now()): Boolean	Is the status of the process at the current time running fine
isExpected(status: SchedulerJobStatus): Boolean	Is the status provided of this process reasonable and expected
getDataForProcess(status: Option[Int] = None, time: DateTime = DateTime.now()): Option[ProcessData]	Get the response and data of the process being monitored if there exist depending on the specific status or the time.

Scheduler Job Client Log Kestrel

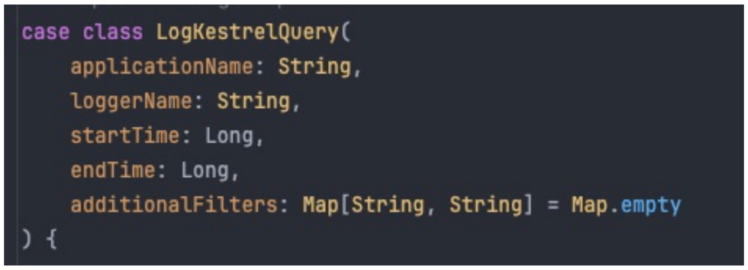
Log Kestrel is one of the services built-inhouse for Agoda teams that allows for metrics and reporting to be integrated to any project such that logs will be printed and viewed through Grafana. Grafana is a dashboard service heavily used by many IT Teams in Agoda for debugging and monitoring purposes as it can visualize and customize filtering for different metrics sent by the service that is integrated.

[Insert Picture of Sample Log Kestrel]

With this service, it comes with its own endpoint that allows to be queried and filtered provided with appropriate and it should respond as a JSON format as the following picture.

[Insert Picture of Postman]

We could use the request format as a case class in our service as it always follows this format therefore this case class would hold the following parameters needed.


```scala
case class LogKestrelQuery(
    applicationName: String,
    loggerName: String,
    startTime: Long,
    endTime: Long,
    additionalFilters: Map[String, String] = Map.empty
) {
```

We also know that as this endpoint would always return in this same format, we can set up a matching case class that could help decode this response and deserialized it as this case class so that it can be computed and understood by the service client.

Now that we can fetch the appropriate data, we can see that we can easily extract the status as it is stored in the response itself. Additionally, with the fetched data also comes with the metadata with response which makes it extracting much simpler.

Scheduler Job Client Whitefalcon

Whitefalcon is one of the services built-in houses in Agoda to help report various kinds of measurements to help with debugging and monitoring purposes. It has been integrated with many services and systems so that it can receive and display real time data with the help of Grafana, a dashboarding tool. All the tools provided by this service are:

- Measurement Tagging
- Aggregation across dimension / time periods / data centers
- Filtering by tag combinations
- Percentile calculation over measurements (on request)

[Sample image for whitefalcon being used]

With this service also comes with it own endpoint to be able to fetch the appropriate measurements of the processes provided the metric.

[Insert picture of postman]

One notable thing about the endpoint is that it always accepts and return in the same format in JSON like Log Kestrel so were we able to write case classes, decoders and encoders to make it much easier to be able to send request and receive response from this endpoint so processing these data would be much more convenient.

[Insert picture of case classes]

Scheduler Job Notifier

Scheduler Job Notifier is the main component responsible for fetching relevant information from Scheduler Job Client and format it as a Slack Message so that it can be sent to the appropriate channels with the appropriate username, icon and header which is extracted from the configurations.

This component takes great advantage of an already built framework called Common Alerting Framework which is developed and maintained by the IScrum Core team to help connect and communicate with Slack API so that it can create an abstraction to be able to perform slack related operations such as sending messages much easier.

There are 2 main unique features in this project especially on Scheduler Job Notifier that is not usually seen in Agoda’s other notifier services which are:

- Ability to update the same message = as this service would get triggered every 10 minutes therefore if it would keep sending message every 10 minutes it can cause a lot of unnecessary spamming in the channel. Brief explanation regarding how the updating functionality works is that it fetches the timestamp of the previous message sent or updated with the same username from Common Alert Library. After it fetches the timestamp, it would compare if the current day of previous message is the same if its not send a new message and if it is updating the message.

- Reply functionality = with this functionality, it can reply to the latest message sent to specifically lists processes that are in need to be looked at and investigated by the appropriate team. Brief explanation is that it would compare the statuses of the processes and if it turns from a status is expected to a status that is not expect it would reply to the same message listing these processes while tagging the appropriate team.

Now that the core unique features are mentioned let's look at the methods defined in this component so that the purpose of this component and further looked at and understood.

- getLatestReportTimestamp

Tries to fetch the timestamp of the latest message sent or message updated by the notifier bot with retry functionality.

- getLatestReport

Tries to fetch the latest message content sent or updated by the notifier bot with retry functionality.

- updateLatestReport

Tries to update the latest message by replacing the old content with the new content with retry functionality.

- replyLatestReport

Replies to the latest message by the same notifier bot in the thread with retry functionality.

- report

Main feature of this notifier component where it would fetch the latest message timestamp and compares the day of the latest message and either sends a new message or update the previous message with the logic explained before.

- submitNewReport

Based on the current statuses, prepares the new report to be sent as a new message with retry functionality

- updateExistingReport

Based on the current and previous statuses extracted from the latest messages it will prepare a new report to be replace the old one by updating the previous message. Along with this, as the previous statuses are fetched it will compare if a process status has changed from expected to unexpected and if it has, it will reply to the same message as a threat by preparing the appropriate log to be sent. Both update and reply have a retry functionality implemented.

- prepareReport

Gets the formatted content and prepares the overall message format based on the processes set in the config and their statuses. One remark is that the processes status are sorted by priority to be investigated in the main message therefore failed or late processes are sorted to be at the top.

- prepareLog

Gets the formatted content and prepares the overall reply message format based on the processes that needs to be investigated and appropriately alerts with the appropriate team in the footer.

- getPreviousStatuses

Gets the previous statuses of the processes which is meant to be the previous execution of the service therefore the latest message sent so far is extracted and with the help of regex and pattern matching on the message, the processes and their corresponding status is extracted to help us compute when to reply.

- getCurrentStatuses

Fetches the current statuses of each process set in the configurations by using the Scheduler Job Client and put them into a dictionary table.

- getFormattedContent

This is the main function that calls the functions getHeaderBlocks, getSummarySection and getStatusAttachments where it gets combined to form the overall message report to be either sent or updated. Additionally, it also assigns the bot more information so that it can be sent which is the icon and username.

- getHeaderBlocks

Prepares the header for the message by fetching the message from getHeaderDetailMessage

- getProcessesErrorSummary

This function only gets called when a process has statuses of the following: Failed, Late or EndpointError and would list processes with their corresponding statuses of the ones that has been listed. Additionally, this function is also responsible for tagging the appropriate team only when a process either is Failed or Late. No tagging take place when there is endpoint error as it does not tell any information about the process so it may or may not need to be investigated.

[Insert Image]

- getSummarySection

Prepares the summary section of the entire report and computes whether there needs the team needs to be tagged depending on the current statuses fetched.

[Insert image]

- getStatusAttachmentsPerGroup

Groups each process based on their source application then carries out retrieving status attachment of each process set from the configuration.

- getSectionAppMessage

Prepares the section division of each different application for easier navigation in the message.

[Insert Image]

- getStatusAttachments

Combines the message format produced from getStatusAppSection and getStatusAttachment of each process to create a block displaying the statuses of each process based on the same group.

[Insert Image]

- getStatusAttachment

Fetches and combines the content from getClientStatusSection and getClientContext to create a block message for each process.

- getClientStatusSection

Prepares the message format to be displayed for the main section of each process which is the process and the status with their appropriate icons.

- getClientContext

Prepares the content to be displayed as the context of each process which in this service would be the metadata.

- getHeaderDetailMessage

Prepares the content for getHeaderBlocks by displaying the appropriate date and time in the message that will be sent along with the timestamp to show when the message was last updated for better understanding of the content being displayed and debugging.

- getClientContextMessage

Prepares the metadata message to be displayed along with appropriate message of start and finish time depending on the status of the process.

[Insert Image]

- processAppMetadata

Prepares further metadata message that could be unique for each process so that it could be displayed which would be by the Scheduler Job Client and is combined with all the other metadata message.

Scheduler Job Service

This component simply connects the two components Scheduler Job Client and Scheduler Job Notifier where would it call the notifier component to report based on the clients added to the config.

3.6.2 Resource Monitoring Notifier Service

The Resource Monitoring Notifier Service is the project that I was assigned after having completed the first implementation of Scheduler Job Bot. This project requires an implementation of an alerting flamework that alert an appropriate team through the corresponding text channel with the help of Slack API regarding process or report failure.

As this was implemented when Healthcheck and Scheduler Job Notifier services were already in production, many of the ideas were inspired from those services to create this one. Not only it is inspired this service would run along those services therefore this service gets triggered every 10 minutes to keep fetching the resources usage of different servers and alerts the team if one of the servers has one of the resources exceeding the threshold.

The benefits of having this framework implemented in Agoda are of the following:

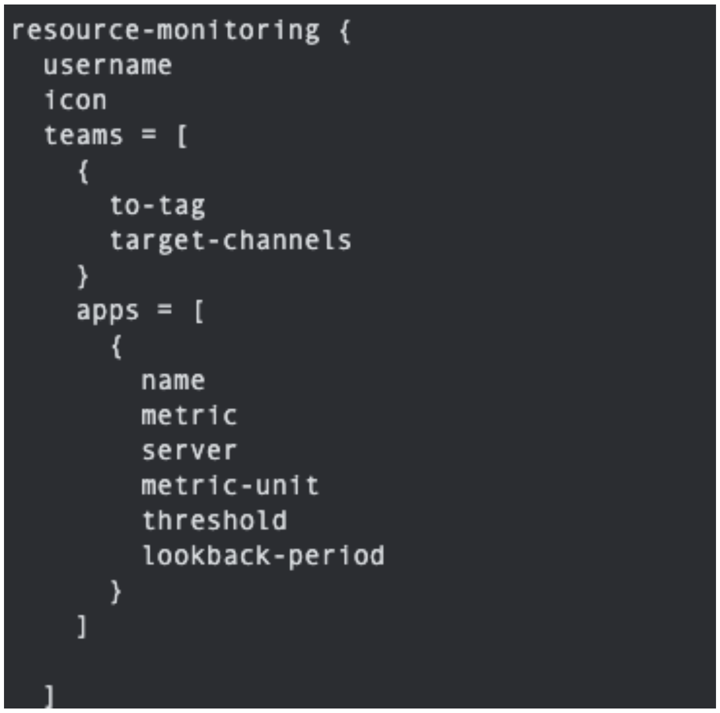
- Scalability
- Maintainability

As this framework is running along and inspired from the services that are already a part of Generic Monitoring Service repository it would be best for this service to co-exist there as well with its similarities.

Configurations

This is the crucial part of the service as it will provide the appropriate information to the service so that it can be able to run the service and track the servers appropriately and alert when needed.

Like all the services that are running in Generic Monitoring Service, this service has also been made sure to keep the service generic and scalable as possible with the following format.


```
resource-monitoring {
  username
  icon
  teams = [
    {
      to-tag
      target-channels
    }
    apps = [
      {
        name
        metric
        server
        metric-unit
        threshold
        lookback-period
      }
    ]

  ]
```

Regarding the dynamics it has similar structures to other services with similar purposes where username and icon are fixed between each service to set the username and icon for the Slack Bot configuration to send. The configs teams represent the sub services be run alongside together to allow to be sent to different team channels and tag different teams as different teams need to maintain different servers.

Now getting to what is unique in this service is how different metrics of a server is configured so each apps is set to be each metric to be measured. These are the following configs needed for each app along with their brief explanation.

- Name = Display name for resource and server being measured to display in the message for readability.
- Metric = Metric that contains the appropriate resource usage measurement to help filter with the Whitefalcon endpoint
- Server = server set to help filter further with the metric provided to specifically get the appropriate resource of the server.
- Metric-unit = unit of the resource being tracked to help with readability in the message

- Threshold = the maximum possible limit the resource metric should be under e.g., CPU should only be under 20%.
- Lookback-period = measurement to fetch the resource usage data in the last lookback-period minutes

There also exists configurations for formatting and behavior of the display but they have the same structure and format as Scheduler Job Service so going into details again won't be necessary. Those are also fixed for every message sent by this service to keep consistency and readability.

Implementation:

Resource Monitoring Notifier Service are split into 3 main components which are of the following:

- Resource Monitoring Client
- Resource Monitoring Notifier
- Resource Monitoring Service

Resource Monitoring Client

Resource Monitoring Client is the primary component in this server as this client is responsible for fetching all the resource usages in the past lookback-period minutes and with further processes it fetches the peak usage reached by the resource then it returns one of the following statuses:

- Stable – The resource did not exceed the threshold, so server is normal.
- High Usage – The resource did exceed the threshold, so the cause needs to be investigated.
- Server Error – The endpoint used to fetch measurements failed.

What was observed so far is that many servers' measurements outputs are reported to Whitefalcon which its purpose and advantages were discussed therefore as of right now the client only fetches these servers resource usage from the Whitefalcon endpoint but as scalability was kept in adding another endpoint to fetch from should not be much of a problem.

As mentioned earlier that Whitefalcon endpoint always return the same format, it can be applied here as well so a decoder and encoder previously implemented was used for better developing experience.

Let's look at each function in this client to further understand the underlying of the Resource Monitoring Client:

Method	**Purpose**
getPeakValue(time: DateTime = DateTime.now())	Sends a request to Whitefalcon endpoint using WhitefalconClient to fetch all the measurements of the past lookback period (in Minutes) from the time given on the server and metric specified. Once fetched successfully the max value is extracted.
isAboveThreshold(value: Double): ResourceMonitoringStatus	With the value given in the argument it will compare with the threshold set from the configuration and returns the appropriate status.

Resource Monitoring Notifier

Resource Monitoring Notifier is a part of the component responsible packaging and delivering a message by building a Slack message format and deliver through Slack API as well to appropriate channels.

It’s development design closely resembles Scheduler Job Notifier that has already been talked in the report therefore further explanations is not needed as the only thing that is unique to this notifier service which is it fetches different kind of metadata to be displayed in each server section. Additionally, no further request needs to be sent to fetch metadata so getPeakValue and the rest of the configurations fixed for the client can provide all the information needed regarding the server.

Resource Monitoring Service

Resource Monitoring Service like Scheduler Job Service component simply connects the two components Resource Monitoring Client and Resource Monitoring Notifier where the service component would call the notifier component to report based on the clients added to the config.

3.6.3 AP Reconciliation Adjustment Approval UI

AP Reconciliation is one of the new built in Fintech team where I have been assigned to help create a UI page that has a table where finance users can perform adjustment approval on different transactions based on their Scenario ID and Status.

This UI was worked on the Finance Backoffice repository. Finance Back Office is a user interface webapp developed for finance users such that they can easily perform financial operations without having to have any knowledge of coding or databases. This application was created using the play framework built with React js and Typescript for Frontend and Scala for backend, it also interacts with other backend services for some of the tools such as Fintech Services that is maintained by The Fintech Tactical Team. This web app was built using the MVC architecture with integration of redux where my project also followed this architecture as well.

Controller

There was only one function that was implemented in the controller which was processApprovalList as further processing needed to be done when the transactions were fetched from the database. The processing that was done within this function was that as the database stored all the possible states of the transactions of the same submission ID grouping was done so that there would be no redundant data, the only transaction state we cared about was its latest state and as timestamp is stored in the transaction we could fetch and get largest timestamp value.

View

This is the crucial part of development as this was a frontend project most of the implementation occurs in view section to make sure the finance users would be satisfied with content and design displayed on the page.

As mentioned before, FBO contained multiple financial tools in its web page therefore AP Reconciliation was the new tool so it had own section in the menu of FBO webpage where picture below could help visualize this menu better.

[Insert picture]

Index.tsx

This is where majority of the computation occurred so that the correct transactions would be displayed based on the filtering.

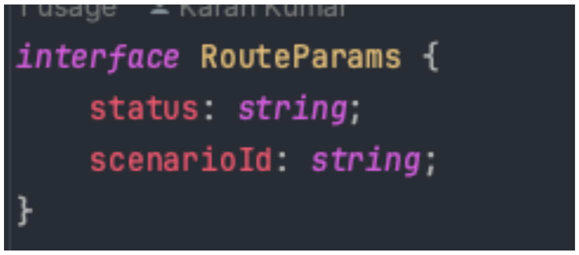
As mentioned before that finance users would need to adjust approvals based on the scenario ID and status, a filtering component was added using a drop-down menu to select these options.

The drop-down options for statuses contained Final Approved Transactions, Rejected Transactions, Pending Approval Transactions and Partially Approved Transactions. These always remained fixed in an array of selectable options. As these were fixed, an Enum was implemented to help ease development experience and improve scalability which would be shown in the model section of the implementation.

On other hand, the drop-down options for Scenario ID were not fixed and instead fetched from a database that contains all possible scenario options. This way it guarantees scalability as adding a new scenario option in the database would allow the web page to sync right away. Fetching the scenario options were done through Redux and GraphQL where it would be dispatched every time the user loads the adjustment approval page.

Filtering

Now that we have understood how these drop downs work, we now try to understand what happens when the options are chosen from these drop-down menus. As mentioned before, that this page lives under a path assigned therefore, filtering was also done with the help of using the path therefore parameters were read every time the approval page was loaded. The following parameters were required in the picture below or else no table would be displayed.



```
interface RouteParams {
    status: string;
    scenarioId: string;
}
```

Once both options were picked using the drop down menus the path would be updated into something as path-till-page/:status/:scenarioId. The reason path filtering was chosen so that when the user refreshes the page it still saves the filtering which helps if fetching data from the server maybe have timed out which occurs often.

Once the data to be displayed in the table last filtering was done which is to display transactions to the user based on their approval level assigned to ensure security as user of certain approval level are only allowed to approve transactions that requires their approval level.

For this filtering, more data fetching was done regarding approval level which depends on the scenario ID chosen with the help of GraphQL and Redux.

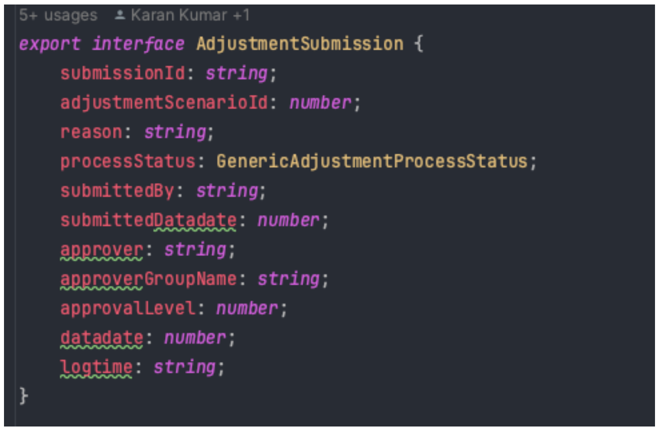
Once the data has been successfully fetched and filtered for the table, it is passed on to the AdjustmentAPApprovalTable component.

AdjustmentAPApprovalTable.tsx

This component handled the design of the table and how should the data be displayed, where if there was no data to be presented, with the help of states, it would be simply empty table with an appropriate message written to let user know. To ensure readability with large number of transactions the table would be split into multiple table pages.

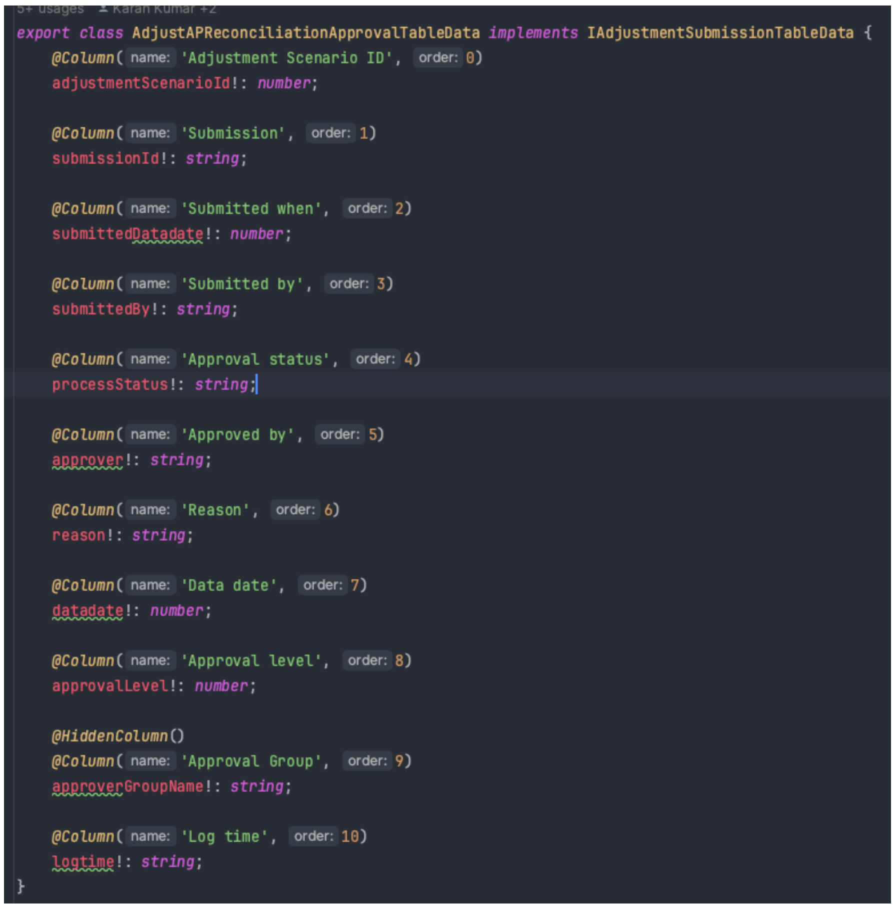
Model

As transactions are fetched from the database, an interface AdjustmentSubmission was created to help map the response to this so that developing experience becomes much more convenient like so:



```
5+ usages  Karan Kumar +1
export interface AdjustmentSubmission {
    submissionId: string;
    adjustmentScenarioId: number;
    reason: string;
    processStatus: GenericAdjustmentProcessStatus;
    submittedBy: string;
    submittedDatadate: number;
    approver: string;
    approverGroupName: string;
    approvalLevel: number;
    datadate: number;
    logtime: string;
}
```

Class called AdjustmentAPReconciliationApprovalTableData was also implemented to help organize the columns that needed to be displayed with their ordering.



```
5+ usages  Karan Kumar +2
export class AdjustAPReconciliationApprovalTableData implements IAdjustmentSubmissionTableData {
    @Column( name: 'Adjustment Scenario ID', order: 0)
    adjustmentScenarioId!: number;

    @Column( name: 'Submission', order: 1)
    submissionId!: string;

    @Column( name: 'Submitted when', order: 2)
    submittedDatadate!: number;

    @Column( name: 'Submitted by', order: 3)
    submittedBy!: string;

    @Column( name: 'Approval status', order: 4)
    processStatus!: string;

    @Column( name: 'Approved by', order: 5)
    approver!: string;

    @Column( name: 'Reason', order: 6)
    reason!: string;

    @Column( name: 'Data date', order: 7)
    datadate!: number;

    @Column( name: 'Approval level', order: 8)
    approvalLevel!: number;

    @HiddenColumn()
    @Column( name: 'Approval Group', order: 9)
    approverGroupName!: string;

    @Column( name: 'Log time', order: 10)
    logtime!: string;
}
```

API

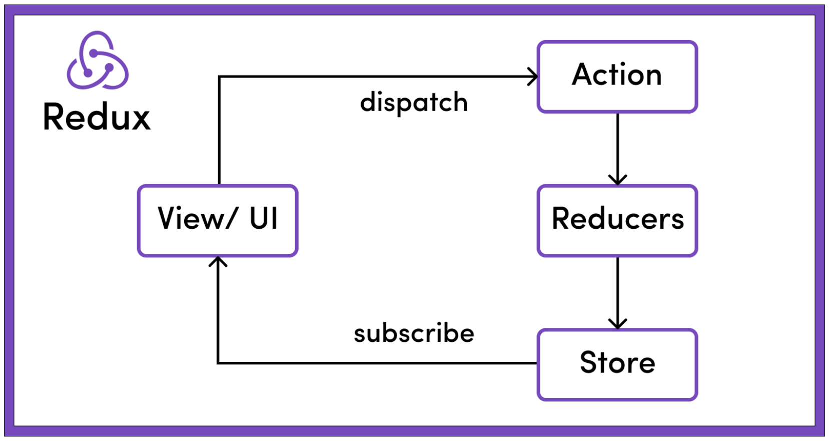
GraphQL API with the help of ApolloClient was used to help fetch 3 different kinds of data that was used in the project that were of the following:

- getGenericApprovalAuthentication = Get the approval levels for each different group or teams based on the scenario ID specified.
- getAdjustmentSubmissionBySecnarioAndStatus = Gets all the transactions that are based on the adjustment status and scenario ID
- getAdjustmentScenarios = Gets all the adjustment scenario possible options

Redux

Redux was used for data fetching for the data as it was the FBO's standard, and it also allows to manage app's state in a single place so that the changes in the app and state could be easier understood and traceable.

https://www.freecodecamp.org/news/what-is-redux-store-actions-reducers-explained/



Action

Actions were used in the approval page as they were the only to call and fetch the data through dispatching. These actions directly called the functions that were implemented using GraphQL APIs to retrive as the payload. The following actions were implemented:

FETCH_ADJUSTMENT_TRANSACTIONS = Fetch all the adjustment transactions based on the status and scenario ID where it would call the function genericAdjustmentTransaction

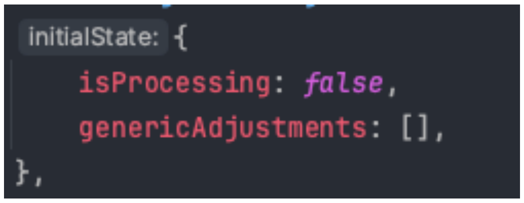
FETCH_ALL_SCENARIO_INFO = Fetch all the scenario ID options using the function getAdjustmentScenarios

FETCH_APPROVAL_INFO = Fetch approval level authentication of every user group and roles based on the scenario ID so the function getGenericApprovalAuthentication is called

Reducers

Reducers are responsible to update from the old state to the new state by taking in two things previous state and action then reduced to the new updated instance of state.

Clearly reducers were used in this page but only one that interacted with many states were GENERIC_ADJUSTMENT_TRANSACTION_DATA.FETCH_ADJUSTMENT_TRANSACTIONS which was responsible for updating the following state below based on 3 conditions:



```
initialState: {
    isProcessing: false,
    genericAdjustments: [],
},
```

Fulfilled = genericAdjustments is loaded with the content from the payload after the controller processApprovalList has been applied and isProcessing can now be set to false

Pending = isProcessing is set to true as the table isn't ready to be displayed yet

Rejected: isProcessing is set to false and empty array of genericAdjustments is set

Stores

Stores were not within the scope of my project therefore it wasn't used.

Chapter 4. Results

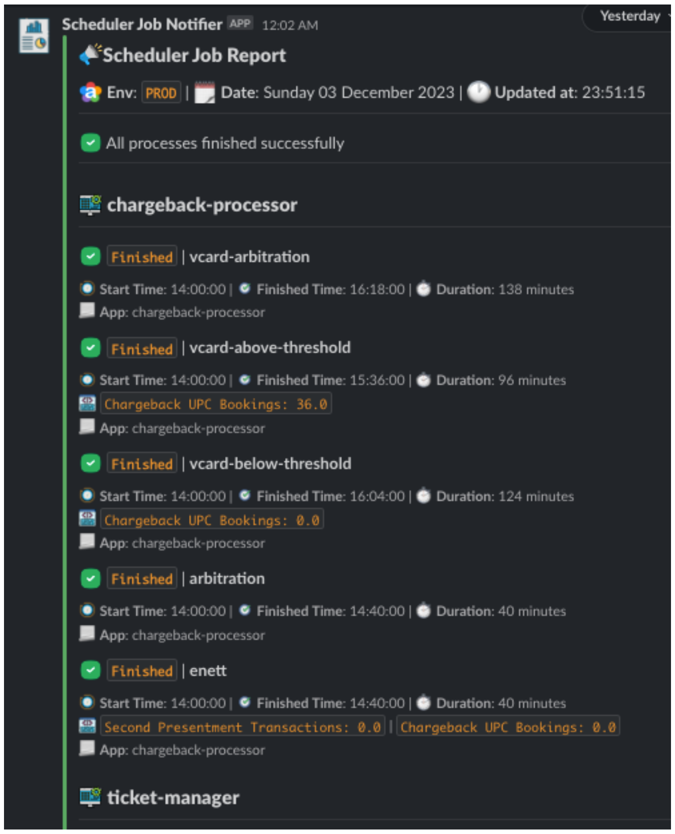
Chapter 4.1 Project Outcomes

Chapter 4.1.1 Scheduler Job Notifier Service

With this completed project it is now deployed on HUE and being run daily to check and monitor production-grade processes every 10 minutes and will help alert the appropriate teams that have onboarded this service into their channel. This would allow those teams to be able to respond to any issues that may occur if the process failed or even fail to run with the reply and customized tag functionality. With its maintainability the users of this service would not need to implement anything extra assuming their measurements are coming from the endpoints Whitefalcon or Log Kestrel, they can simply add their process, their expected start time and finish time in the config in the appropriate place and be ready to go.

Unit testing also were written to ensure consistent and expected behavior always for all 3 components and the endpoint client e.g., LogKestrel and Whitefalcon

Send/Update a message:



Scheduler Job Notifier APP 12:02 AM
Yesterday

Scheduler Job Report

Env: PROD | **Date:** Sunday 03 December 2023 | **Updated at:** 23:51:15

All processes finished successfully

chargeback-processor

Finished | vcard-arbitration
Start Time: 14:00:00 | Finished Time: 16:18:00 | Duration: 138 minutes
App: chargeback-processor

Finished | vcard-above-threshold
Start Time: 14:00:00 | Finished Time: 15:36:00 | Duration: 96 minutes
`Chargeback UPC Bookings: 36.0`
App: chargeback-processor

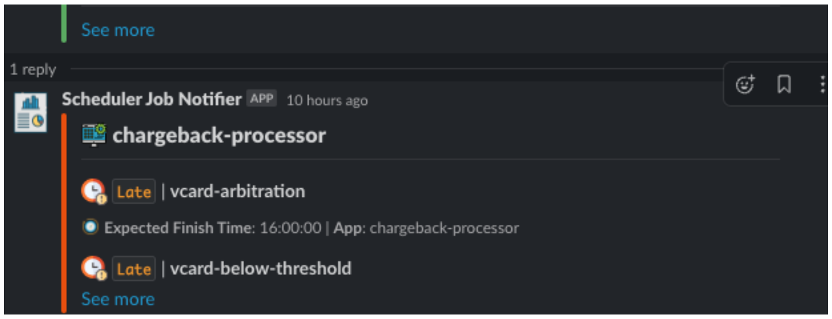
Finished | vcard-below-threshold
Start Time: 14:00:00 | Finished Time: 16:04:00 | Duration: 124 minutes
`Chargeback UPC Bookings: 0.0`
App: chargeback-processor

Finished | arbitration
Start Time: 14:00:00 | Finished Time: 14:40:00 | Duration: 40 minutes
App: chargeback-processor

Finished | enett
Start Time: 14:00:00 | Finished Time: 14:40:00 | Duration: 40 minutes
`Second Presentment Transactions: 0.0` | `Chargeback UPC Bookings: 0.0`
App: chargeback-processor

ticket-manager

Reply to a message:


See more

1 reply

Scheduler Job Notifier APP 10 hours ago

chargeback-processor

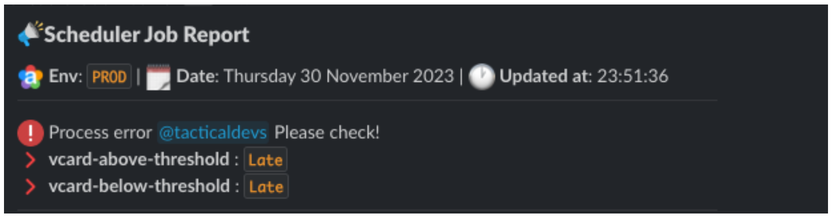
Late | vcard-arbitration

Expected Finish Time: 16:00:00 | App: chargeback-processor

Late | vcard-below-threshold

See more

Alert appropriate team:


Scheduler Job Report

Env: PROD | Date: Thursday 30 November 2023 | Updated at: 23:51:36

Process error @tacticaldevs Please check!

vcard-above-threshold : Late

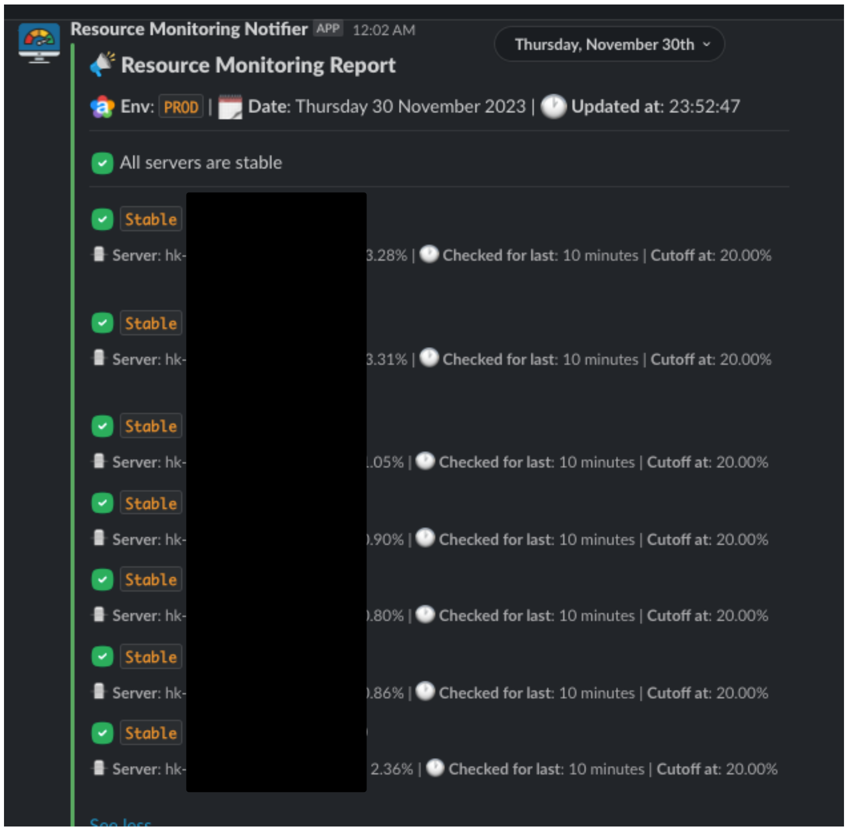
vcard-below-threshold : Late

Chapter 4.1.2 Resource Monitoring Notifier Service

With this completed service implemented it is now deployed on HUE to run alongside other services in Generic Monitoring Service being run every 10 minutes and appropriate team that have onboard this service into their channel and group. After onboarded and adding the servers and different resources to be monitoring it will allow the team to respond to any issues that may occur if any servers start hanging or overloaded at a faster pace thanks to the tagging feature. With its maintainability and scalability, users can easily add new servers and resource metric within the config and add a new endpoint to be fetched from by implementing the new client. After onboarding the new servers on to the configuration, the team can just deploy this new change and the service will be run as usual along with the new configurations.

Unit testing was also written for this service to always ensure expected behavior so no misinformation would be given to the teams and cause any inconvenience.

Send/Update the message.



Resource Monitoring Notifier APP 12:02 AM

Thursday, November 30th

Resource Monitoring Report

Env: PROD | **Date:** Thursday 30 November 2023 | **Updated at:** 23:52:47

All servers are stable

Stable
Server: hk-[redacted]3.28% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Stable
Server: hk-[redacted]3.31% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Stable
Server: hk-[redacted].05% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

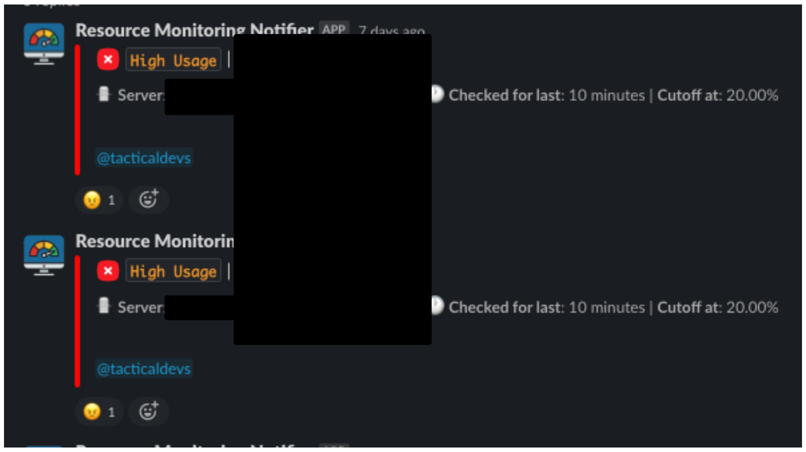
Stable
Server: hk-[redacted].90% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Stable
Server: hk-[redacted].80% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Stable
Server: hk-[redacted].86% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Stable
Server: hk-[redacted]2.36% | **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

Reply to a message.



Resource Monitoring Notifier APP 7 days ago

High Usage |[redacted]
Server[redacted] **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

@tacticaldevs

1

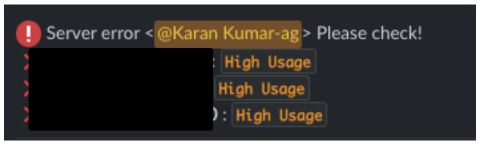
Resource Monitorin[redacted]

High Usage |
Server[redacted] **Checked for last:** 10 minutes | **Cutoff at:** 20.00%

@tacticaldevs

1

Alert the team.

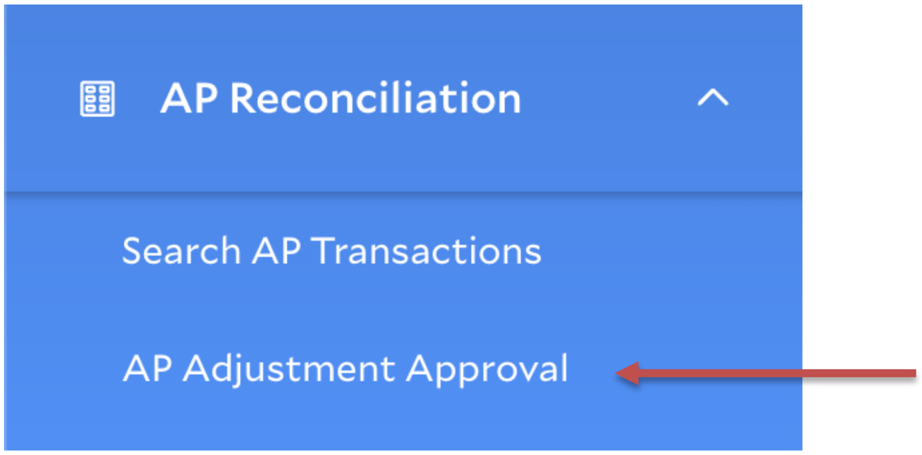


```
Server error <@Karan Kumar-ag> Please check!
[redacted]: High Usage
High Usage
): High Usage
```

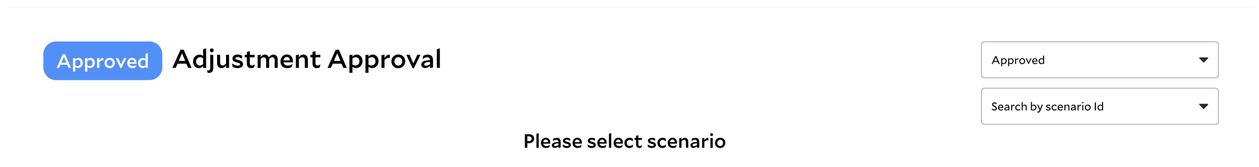
Chapter 4.1.3 AP Reconciliation Adjustment Approval UI

This page has now been completed but it has been yet placed in production for finance users to use as the complete implementation of other sub tools is not yet finalized and ready.

Now the submenu for AP Adjustment Approval will appear under AP Reconciliation tool.



Where once the finance user clicks to use this tool in the submenu this page would show up as the default where the default status would be approved but unselected for scenario Id so the user would be requested to select scenario ID to see the appropriate data.

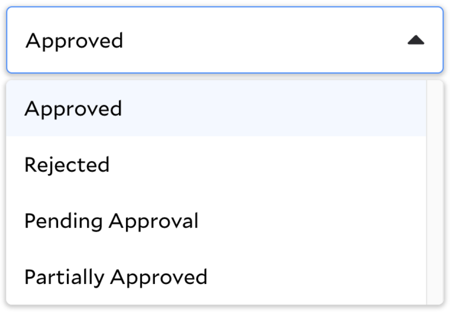

Approved Adjustment Approval

Approved

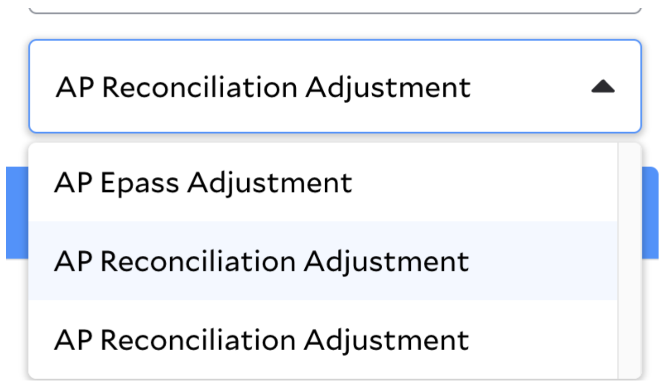
Search by scenario Id

Please select scenario

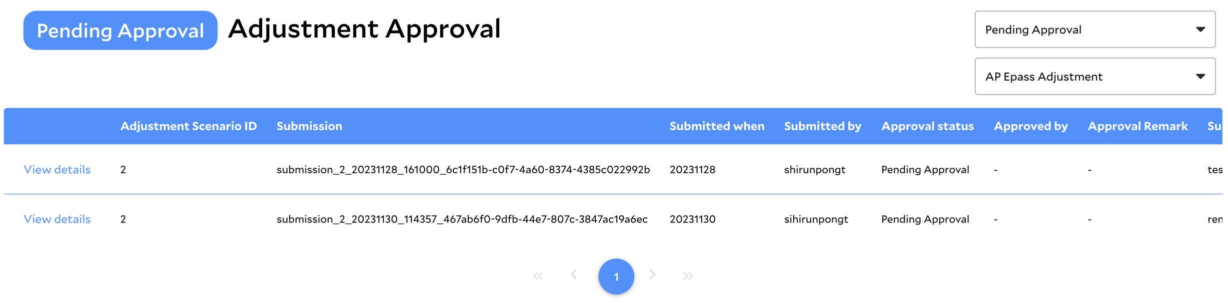
The user may switch the status to their choices with these options.



The options would instantly be loaded for scenario Ids as soon as the default page is loaded so the finance users can choose according to their needs.


AP Reconciliation Adjustment
AP Epass Adjustment
AP Reconciliation Adjustment
AP Reconciliation Adjustment

Once all the parameters are satisfied the table would finally be displayed based on the users' approval level.



	Adjustment Scenario ID	Submission	Submitted when	Submitted by	Approval status	Approved by	Approval Remark	Su
View details	2	submission_2_20231128_161000_6c1f151b-c0f7-4a60-8374-4385c022992b	20231128	shirunpongt	Pending Approval	-	-	tes
View details	2	submission_2_20231130_114357_467ab6f0-9dfb-44e7-807c-3847ac19a6ec	20231130	sihirunpongt	Pending Approval	-	-	ren

Chapter 4.2 Challenges and Issue Analysis

Chapter 4.2.1 Scheduler Job Notifier Service

- **Combining SchedulerJobWhitefalconClient and SchdulerJobLogKestrelClient**
- **Making SchedulerJobWhitefalconClient metadata fetching generic**

Chapter 4.2.2 Resource Monitoring Notifier Service

- **Built-in WhitefalconClient did not support the response format.**

Chapter 4.2.3 AP Reconciliation Adjustment Approval UI

- **Incorporating Redux to fetch data for the table.**

Chapter 5. Conclusion

5.1 Summary of Co-operative Educations

During the immersive Co-operative Education Program hosted by Agoda from August 7th, 2023, to December 7th, 2023, I was given an opportunity to take on a role as a Software Engineer Intern within the Fintech Intern Core Team operating under the experience guidance of Fintech Tactical Team. Throughout this period, I have undergone through practical training by being assigned with projects that has improved the response time regarding the health of servers and different processes ran in Agoda.

In addition to my projects, I also took some of my bandwidth to enhance, testing and debug some of services and repositories across multiple IT-Finance teams. Having worked with many platforms has made me have a better understanding of how the Fintech team operates and what services they provide.

The Co-op experience has exposed me to a myriad of tools both that is either built-in or outsources which significantly expanded my practical knowledge in real-world software engineering. Having been exposed to many tools has also broadened my understand of industry best practices, technology stacks and conventions followed.

Beyond technical skills, being in a collaborative team and environment greatly improved my soft skills especially in time management and communication. This constant collaboration has made me much better at working with new people.

One important highlight during my internship was taking the responsibility of being on-call and scrum master which is a very standard practice for many teams across various tech companies. This experience proved invaluable as it had made me improve my leadership skills and being able to carry on meetings when there no one has anything to say.

Overall, my time being a part of Agoda has been transformative, not only in technically but also acquiring essentials skills to be a successful software engineer and a team member. With all these skills and experience has built a foundation where I am motivated to continue growing and improving within the field of software engineering.

5.2 Challenges and Issues

5.2.1 Unfamiliarity with Agoda's tools and software

The tasks received throughout the duration of the internship often involved the utilization of tools and software which I had no prior experience. The learning curve for some of the tools and concepts has also proven to be very steep which required a huge initial investment of time to gain enough knowledge and proficiency before being able to use them to complete my tasks. Not having foundation of the tools also greatly extended the time required to complete the tasks.

However, despite diverse range of tools being integrated in Agoda has given me the opportunity for continuous learning and growth. Despite the initial hurdles and struggles gone through to finish the tasks being able to adapt to new tools has broadened my technical skills.

5.2.3 Lack of conventions and standards

In the beginning of my Co-operative education at Agoda, I observed a variance in the integration of conventions and standards within software development. One of the most notable one I encountered early on was absence of consistent guidelines for development and collaboration of code such as branch naming or commit messages conventions on Git. This absence means that there a less standardized development experience and can be hard for newcomers to understand and get familiarized with development within Agoda.

However, it is worth mentioning that even though there were some aspects that did not have uniformity, there were also instances that it has established standards that surprised me as not only the fact that it existed but also how strict it could be particularly in areas like stored procedures. These standards presented has made me understood some of the guidelines needed to follow and made me appreciate the development practices within the company.

5.2.4 Long waiting time for permission and access

Throughout the internship, one of the most notable challenges I had gone through was the prolonged waiting time for acquiring permissions or access to specific development features. On several occasion, I experience delays in my work completing spanning hours or even days which has inhibited my productivity despite giving reminders to the righteous person who can grant me access.

This waiting period has not only affected my efficient but can play a huge role underscoring the importance of streamlining access procedures to boost overall team productivity at Agoda.

5.3 Recommendations & Comments

- More team building activities like dinner, board games or other internal team events could be a fun experience for a team to bond and make meaningful connections beyond the confines of a professional relationships. These activities could also make an opportunity for newcomers to forge stronger bonds within the team which greatly accelerates the onboarding experience.

- Organizing dedicated workshops hosted tailored for interns can offer substantial benefits. These workshops can help supplement their knowledge already gained from the university. With the lectures it can also lessen the gap between theory and real-world application so that interns can empower their skills to be applicable and boost their productivity. Lastly, having lectures can also give an opportunity for interns to build connections with other interns and with experts who organize the lectures.