

SCHOOL OF ENGINEERING AND TECHNOLOGY

**COURSEWORK FOR THE BACHELOR IN SOFTWARE ENGINEERING (HONS); YEAR
2/3**

ACADEMIC SESSION APRIL 2024; SEMESTER 6,7

CSC3209: SOFTWARE ARCHITECTURE AND DESIGN PATTERNS

DEADLINE: 2ND AUGUST 2024 10:00 am

GROUP: 9 (E-Hailing)

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 20% to your final grade.
- This is a group assignment. Each group consists of 5-6 members

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 60%.
- Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.

Students' declaration:






	(Name)	(ID)	(Signature)
We	1) Aisha Sofia Binti Najidi	20065231	
	2) Emily Teng Jie Qi	20066502	
	3) Justin Phang Sheng Xuan	20054607	
	4) Catherine Shagembe Mipawa	20095089	<div>C.S.Mipaw</div>
	5) Alkin Wong Wye Kin	20040275	

received the assignment and read the comments

Academic Honesty Acknowledgement

“We (names stated above) verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties (*refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme*) for any kind of copying or collaboration on any assignment.”

- 1) Aisha Sofia Binti Najidi
- 2) Emily Teng Jie Qi
- 3) Justin Phang Sheng Xuan
- 4) Catherine Shagembe Mipawa
- 5) Alkin Wong Wye Kin

  ,   

07/08/2024 (Student's signature / Date)

General description for SADP Assignments

For this assignment, you are required to form a group to go through a software architecture design process. The targeted goal is to design and build an architecture and a prototype of the designed system. The target system should be a modern system such as e-Hailing (Grab, Uber) or an e-Wallet (Boost, Touch and Go) system. The system must support real time exchange of information among clients. There should be mandatory functions that involve communication, security, performance, availability and some other important quality attributes. The other functions depend on the scenarios and requirements on which the group has decided.

Description of the assignment (20 marks)

This is the second project assignment, and it is the subsequent assignment of the first project assignment (Assignment1). The goal of this assignment is to apply the utility tree technique to prioritize important and relevant requirements from the list of requirements from Assignment #1. Based on the shortlisted and prioritized list of requirements, design a software architecture for the desired modern system. Also present the prototype of the system.

The breakdowns on the required contents are:

1) Utility Tree (5 marks)

Based on the gathered requirements, use the Utility Tree technique to analyse and prioritize the requirements to shortlist the most important requirements/ASRs.

2) Architecture Design (10 marks)

Design the architecture and provide the required documentation. In your design, you must address the following:

- You must employ the design concepts (architectural design patterns, architectural views, architectural structures) that have been discussed in the lecture notes. You must employ at least two architectural design patterns, structures and views. You must employ at least one module view and one component and connector C&C view. **Those selected design concepts should address the identified and analysed requirements obtained in Assignment#1** (functional and quality attributes) and addressed in the utility tree.
- You must present a mapping between the selected architectural views. This is because the views present different perspectives of the same architecture, and they should be related.
- **You must employ at least two notations to document the behaviour (e.g. sequence diagrams, activity diagrams...).**
- Based on the designed architecture, present the prototype of the system.

3) Self-reflection (5 marks)

Based on the roles each has chosen or been assigned to, write a reflection on the role and influence on the prioritization of requirements and selection of design concepts (i.e. architectural patterns and views).

The prioritization of requirements needs to be well justified based on roles and needs. Similarly, the architecture design decisions should also be supported by justification.

Expected outcome:

The outcome is expected to be written in a Word/PDF Document report. The report should consist of THREE sections: a) The utility tree and the prioritization of requirements b) The final architectural design and documentation that addresses the required points in the second question and the system prototype c) The justifications and reflection on the requirements and design decisions and concepts.

	Excellent	Good	Adequate	Insufficient
1) Utility tree and prioritization (5)	Excellent use of utility tree and prioritization of requirements are presented. The justifications were excellent and led to the architecture	Good use of utility tree and prioritization of requirements are presented. The justifications were good and can led to the architecture	Adequate use of utility tree and prioritization of requirements are presented. The justifications were acceptable and may lead to the architecture	Inadequate use of utility tree and prioritization of requirements are presented. The justifications were below acceptable and do not fully lead to the architecture.
2) Architecture design (10)	<p>Excellently justified and designed architecture.</p> <p>Excellent derivation of the design concepts from the requirements and observation of identified the problems/scenarios.</p> <p>Excellent presentation of the design concepts, combination and notations to document behaviour.</p> <p>More than two design concepts and notations are employed in designing the architecture.</p>	<p>Well justified and designed architecture.</p> <p>Good derivation of the design concepts from the requirements and observation of identified the problems/scenarios.</p> <p>Good presentation of the design concepts, combination and notations to document behaviour.</p> <p>Two design concepts and notations are employed in designing the architecture.</p>	<p>Adequately justified and designed architecture.</p> <p>Adequate derivation of the design concepts from the requirements and observation of identified the problems/scenarios.</p> <p>Adequate presentation of the design concepts, combination and notations to document behaviour.</p> <p>Less than two design concepts and notations are employed in designing the architecture.</p>	<p>Inadequate justified and designed architecture.</p> <p>Inadequate derivation of the design concepts from the requirements and observation of identified the problems/scenarios.</p> <p>Inadequate presentation of the design concepts, combination and notations to document behaviour.</p> <p>Less than two design concepts and notations are employed in designing the architecture.</p>
3) Self reflection (5)	Excellent reflection that demonstrated excellent appreciation of utility tree, prioritization of requirements and selection of design concepts. Student can demonstrate complete understanding on translating requirements to architecture design.	Well-presented reflection that demonstrated good appreciation of utility tree, prioritization of requirements and selection of design concepts. Student can demonstrate good understanding on translating requirements to architecture design.	Adequate presentation of reflection that demonstrate sufficient appreciation of utility tree, prioritization of requirements and selection of design concepts. Student can demonstrate sufficient understanding on translating requirements to architecture design.	Inadequate presentation of reflection that demonstrate poor understanding of the use of utility tree and selection of design concepts. Student has not demonstrated any ability to design an architecture based on requirements.

Section 1: Introduction

Purpose

The purpose of this document is to provide a comprehensive overview of the architectural decisions and design choices made for the development of our e-Hailing system. It aims to detail the system's structure, the rationale behind the chosen architectures, and how these decisions meet the requirements and constraints of the project. This document serves as a guide for stakeholders to understand the architecture, its components, and their interactions.

Scope

This document covers the architectural aspects of the e-Hailing system, including the overall system architecture, detailed descriptions of the Service-Oriented Architecture (SOA), Layered Architecture, and the Pipe and Filter pattern used for specific services. The scope is limited to the architectural design and does not delve into the implementation details or specific coding aspects.

Document Organisation

The document is organized into the following sections:

1. **Introduction:** Provides the purpose, scope, document organization, view description, and how stakeholders can use the document.
2. **System Overview:** Offers a high-level overview of the e-Hailing system, including its main components and interactions, scenarios, functional requirements, quality attributes, and utility tree.
3. **Views:** Describes the chosen architectural patterns, including Layered Architecture, Service-Oriented Architecture (SOA), and Pipe and Filter pattern.
4. **Mapping Between Views:** Details how the different architectural views are related and integrated.
5. **Rationale:** Explains the rationale behind each architectural decision.
6. **Directory:** Explains the technical terms and lists references
7. **Self-Reflection:** Self Reflection of Group members

How Stakeholders can use

- **Developers:** Use this document to understand the architectural framework and the relationships between different components
- **Project Managers:** Reference the architectural decisions to ensure the project meets the specified requirements.
- **System Architects:** Utilize the detailed architectural views and justifications to verify the architecture.
- **QA Engineers:** Use the architecture documentation to develop test plans and strategies that cover all system components

Section 2: System Overview

Scenarios

1. Functional Requirements

a. Scenario 1: User Account Log In

Users should be able to authenticate their identity seamlessly using the application, ensuring secure and efficient access to personal information, ride history, and other features. When Alex opens the application, he is presented with a login form requiring his username and password. Upon entering his credentials, the system validates them against stored user data. If the credentials are correct, Alex is redirected to his dashboard where he can manage his profile and book rides; if incorrect, an error message prompts him to try again or use the "Forgot Password" feature. For enhanced security, the application may optionally require two-factor authentication by sending a one-time password (OTP) to his registered email or phone number, which he must enter to complete the login process.

Additionally, the application provides a "Remember Me" feature to streamline future logins by securely storing his credentials, ensuring convenience without compromising security. In case Alex forgets his password, he can use the "Forgot Password" link, which prompts him to enter his registered email. The system then sends a password reset link, allowing him to set a new password and regain access to his account. This comprehensive login authentication process ensures that users have a secure, user-friendly experience while accessing the application's features.

b. Scenario 2: Users should be able to book rides on the app

Users should be able to book rides seamlessly using the e-hailing application, providing a more convenient and efficient transportation solution compared to taking public transportation. The booking procedure begins when the user launches the application and picks their preferred pickup and end destinations. After confirming the details, the system will then use its GPS technology to locate available drivers in the vicinity and assigns the nearest driver to the user's request. Once a driver has been assigned, the user will receive a confirmation notification along with the driver's information such as their name, profile picture, car description and the estimated time of arrival (ETA).

For example, Jane opens the e-hailing application on her smartphone and enters her pickup location (office) and destination (home), in addition to her preferred car type. Before confirming her booking, the system will first display the ETA and fare of the whole journey and her preferred payment method (Card, E-wallet and cash). If Jane is satisfied with given circumstances, she will then proceed to confirm her booking. The system then processes her request and assigns a nearby driver, John, who's vehicle satisfies Janes preference. After John has read the journey details and accepted her request, Jane will get a confirmation notification. During this time, the

payment placed on hold. Jane will be able to track his current location and ETA as he travels to the prior specified pickup location. Throughout the journey, both the user and the driver can contact each other via in-app messaging or phone calls for any updates of change in plans. After arriving at her destination, the driver will confirm that the journey has been completed and if Jane has chosen online payment (card, online banking) as her payment method, the held transaction is finalized and processed accordingly.

c. Scenario 3: Integration with third party services

The integration of other third-party services among an e-hailing application may be a more advantageous decision over creating proprietary features from scratch. This is because some of the services require logics that are already well-established prior the creation of the application, to which the majority may find it to be more reliable in comparison to a proprietary feature. In most cases, developers are also required to allocate most of their time to focus on the software's core features and may find redundant to reinvent the wheel. Hence, to counter the limitations of time and costs of development, trusted 3rd party services are incorporated inside the systems mainly through other software companies' API.

A perfect example of this would be integrated payment services during the transaction process of booking a ride. For example, Jane chose to pay for her ride through her credit card. As she selects the option, she is redirected to a page that will require her to fill in the card's details. After that, Jane will then be redirected to her device's browser which prompts her to verify her card details in her chosen bank's website itself. For example, if Jane is using a UOB credit card for her upcoming payments, Grab will open a new browser page which will prompt her to verify her card details on UOB's website itself through methods like OTP (One-time password) verification. After the process is complete, Jane will be directed back to Grab and be notified of her payment's confirmation. It is worth noting that the integration of 3rd party services is crucial as it ensures user's privacy and establishes a trustworthy platform, especially when it comes to processing confidential information like this.

d. Scenario 4: Real time tracking for rides/passengers

Real-time tracking for rides and passengers is a critical feature in an e-hailing system that significantly enhances user experience, safety, and operational efficiency. This functionality allows users to monitor the live location of their ride from the moment a driver is assigned until the passenger reaches their destination. When a user requests a ride, the system immediately begins to track the assigned driver's location using GPS technology. The user can view the driver's approach in real-time on a map interface within the app, providing an estimated time of arrival (ETA) and continually updating as the driver progresses toward the pickup location. Once the passenger is picked up, the real-time tracking feature continues to monitor the journey, displaying the vehicle's current location, route, and updated ETAs to both the passenger and any connected parties, such as family or friends, who have been granted permission to track the ride for safety purposes. Findings have revealed that sexual harassment is a top concern among passengers, making

the importance of real-time tracking with GPS for location monitoring even more critical (Salleh et al., 2024).

Jane books a ride to her office through the e-hailing app during the morning rush hour. Once a driver is assigned, Jane can see the driver's current location and estimated time of arrival (ETA) on a map interface within the app. The driver, David, is also provided with Jane's location and the optimal route considering real-time traffic conditions. As David approaches, Jane receives notifications about his proximity and arrival. During the ride, Jane's friends, with whom she has shared her ride details, can track the car's location in real-time, ensuring her safety. The system continuously updates the ETA based on the traffic and route changes, keeping both Emily and David informed. This real-time tracking not only enhances Jane's sense of security but also improves the overall efficiency of the ride by optimizing route and time management.

e. Scenario 5: System Notifications

Ensuring timely, accurate, and actionable system notifications is a critical aspect of user engagement and satisfaction in e-hailing applications. Users should be able to receive notifications regarding any of their rides and account activities, such as booking confirmations, notification on drivers' arrival, notification on ride completion, payment status, promotion offers as well as during system updates and or when there is a scheduled maintenance. Upon booking of ride, the system processes the booking requests, verifies the user details and assigns the available driver based on pickup location and once booking confirms the system push notification with ride details to the user. The system then uses GPS and traffic data to continuously track the rider's location and progress and sends notification to the user about the location of the driver and any unexpected occurrence such as delayed rides or traffic within the area. Upon ride completion the driver confirms the ride completion through the app and notification is sent to the user with the trip details.

Example Jane books a ride to get home after shopping for groceries at the Jaya grocer. Immediately after booking, Jane receives a notification that confirms her ride with the driver Jacob, along with Jacob's details and the Estimated Time of Arrival. As Jacob approaches near the Jaya grocer pick up location, Jane gets another notification alerting her that Jacob will arrive within five minutes. During the ride, there was traffic causing delay, Jane received real-time notification on the new Estimated Time of Arrival and the reason for the delay. Upon completion of the ride, she gets an email notification summarizing her trip and confirming that her payment via credit card has been processed successfully. Additionally, Jane occasionally receives notifications about discounts and coupons available for her next ride, enhancing her user experience and loyalty to the app.

2. Quality Attributes

a. Scenario 1: Performance

Performance is critical for the application to deliver a seamless experience to users, ensuring efficient operations even during peak usage times. Alex, a frequent user of

Grab, relies on its responsiveness and reliability especially during rush hours or events.

When Alex opens the app during rush hour, he expects swift loading times and immediate access to available rides without delays. The system achieves this by optimizing server-side processes and leveraging efficient database queries to handle multiple concurrent user requests efficiently. During peak times, load balancing mechanisms distribute traffic across multiple servers to prevent overload and maintain responsiveness.

For instance, during a major event, such as a national holiday, where Alex needs to book a ride home, the system anticipates increased demand and scales up resources accordingly. Additional servers are deployed to handle the surge in requests, ensuring that Alex and other users can seamlessly book rides without experiencing slowdowns or service disruptions.

b. Scenario 2: Availability

System high availability is crucial to providing a seamless and reliable transportation service, as it ensures the accessibility of the services to the users whenever they need it especially during high-demand periods or emergencies. To achieve high availability involves minimizing downtime and ensuring the system can handle varying loads efficiently. To ensure high availability the e-hailing system is designed with redundancy and load balancing that can maintain a high level of uptime, with minimum interruptions or downtime, providing users with a seamless and dependable transportation experience. During high-demand events, load balancers distribute traffic across multiple servers, preventing overloads. In case of server failure, systems failure mechanism redirects to backup servers, ensuring an uninterrupted service. During off-peak hours a scheduled maintenance is performed, and an advance notification is sent to users. Real-time monitoring detects and addresses any anomalies proactively in the system, maintaining continuous availability and reliable user experience, allowing users to book and track rides seamlessly even during peak times.

Example during a concert in Bukit Jalil, thousands of attendees including Jane used an e-hailing app to book rides to avoid troubles of finding parking spaces. As an event approaches to the end, Jane opens the app to request a ride back home. The system anticipated a high demand, it had already added extra servers to scale up and be able to handle the expected surge. The redundancy within the system ensures seamless takeover of all active servers in case of failures of any server. Thus, the load balancers distributed Jane's request along with a thousand of others evenly across all the active servers and Jane's request was processed without any delay. Jane receives a confirmation with the driver's details and estimated time of arrival, and she experiences a smooth and reliable booking process despite the high volume of concurrent ride requests.

c. Scenario 3: Security

The security quality attribute of an e-hailing system is fundamental to its integrity and trustworthiness, ensuring that both passengers and drivers can use the service

with confidence. The e-hailing platform includes robust security features to protect user data and prevent unauthorized access.

For example, John, a frequent user of the e-hailing service, attempts to log into his account from a new device. Upon entering his username and password, the system prompts him to complete a second step of verification to ensure the security of his account. John receives a one-time password (OTP) on his registered mobile number, which he then enters into the application. Only after successfully inputting the OTP does the system grant him access to his account. This multi-factor authentication process ensures that even if John's password is compromised, his account remains secure, providing an additional layer of protection against unauthorized access.

d. Scenario 4: Interoperability

Interoperability is referred to the degree which two or more software systems can communicate to exchange meaningful information without external efforts from the end user. This means that the chosen system to provide services to the said software should seamlessly provide the necessities without any problem during runtime. In the context of an E-hailing application, the quality attribute of interoperability is crucial as multiple 3rd party services are integrated to fit its functional requirements with minimal flaws as possible. Users of the e-hailing system need not to know where the provided services come from and should be able to use these services without extra efforts of authentication with exemptions of involving confidential information such as bank card details, emails etc.

In a scenario where Jane downloads and uses the e-hailing application for the first time, they are presented with options to create user by either registering natively through the application's sign-up menu, or through 3rd party services like Google sign-in API from Google Identity Services (GIS). Should Jane choose the latter, she will be redirected to Google's sign in page to enter essential details like email and password or proceed with two factor authentication if it is enabled for her Gmail account. After the authentication, Jane will be directed back to the application to enter her preferred username for the newly registered account right away and proceed to fill up her remaining details natively. The integrated Google sign-in services allow for users to directly register a new user with their existing google account without going through the longer process of natively registered accounts and can function faster if such services are integrated properly, which could be an ideal option if Jane is opting for convenience. Such is one of the examples of why interoperability is one of the crucial quality attributes in an e-hailing system.

e. Scenario 5: Usability

The usability attribute plays a crucial part in ensuring that users are able to efficiently and effectively interact with the system. Usability in a e-hailing application is essential for ensuring positive user experience. This is especially important for first-time users or those with limited technical skills. The end-goal of the app is to ensure that the users are able to easily navigate thru the app, allowing them to complete tasks, such as booking a ride, without much frustration and confusion.

For example, Jane, who has never used a e-hailing application before, decides to try using an e-hailing service to book a ride to the airport. When she opens the application, she'll find a large, clear "Book a Ride" button in the homepage. Upon clicking the "Book a Ride" button, the app will automatically detect her location and auto-fills her pickup location, granted that she has turned on her location feature in her phone. She will then be prompted to enter her destination. The application will provide her with suggestions as she keys in her destination, aiding in her selection process. The ride options are the presented clearly with simple descriptions and ride fares. After choosing her preferred ride, the application will then summarize the trip details to allow her to review and confirm the booking in a single tap. The ease of use of the application aids users like Jane to navigate easily throughout the application to perform her tasks.

|

Functional Requirements

Scenario 1: Logging In

The form-based specification below outlines the process and requirements for logging into the application.

Title	Log In
Function	To log in to a user's account user their email and password
Description	The login function allows users to access their account by providing their registered email address and password. Upon successful authentication, the user is granted access to their account.
Pre-condition	The user must have an existing account.
Acceptance criteria	Valid Input: <ul style="list-style-type: none">• The user provides a valid email address and password.• The user is redirected to the home screen of the application. Invalid Input: <ul style="list-style-type: none">• If the email or password is incorrect, an error message is displayed: "Invalid email or password."• If either field is left blank, an error message is displayed: "This field is required."• 3 failed login attempts trigger an account lockout mechanism to prevent brute force attacks.
Post-condition	Successful Login: <ul style="list-style-type: none">• The user is authenticated and has access to their account.• User-specific data (e.g., ride history, amount of money, saved addresses) is loaded and displayed. Failed Login: <ul style="list-style-type: none">• The user remains on the login screen with appropriate error messages displayed.• No sensitive information is revealed.

Scenario 2: Ride Booking

The form-based specifications below outline the process and requirements for booking a ride through the application. This ensures the users can easily and efficiently arrange transportation by entering their preferred pick-up and destination details. Relevant trip details and driver information would be provided as well.

Title	Ride Booking
-------	---------------------

Function	To book a ride from the application
Description	Allows users to enter their preferred pick-up and destination details, view the trip details and have basic access of the drivers' details (e.g. name, car details).
Pre-condition	Users must be logged into their account. The system should have access to the GPS services.
Acceptance criteria	User must have an existing account and is logged in. User must enter their pick-up and destination locations. The system must have access to their devices GPS services.
Post-condition	A booking is created in the system with the user and ride details. User will receive confirmation and real-time tracking information of the driver.

Scenario 3: Real Time tracking

The following table is a form-based specification for the real-time tracking functionality of an e-hailing service

Title	Real Time Tracking
Function	To track the live location of their assigned driver and vehicle.
Description	This functionality allows users to track the live location of their assigned driver and vehicle from the moment a ride is booked until the passenger is dropped off. This ensures transparency, enhances safety, and improves communication between the driver and passenger.
Pre-condition	User is logged into the e-hailing app. A ride has been successfully booked and a driver assigned. Both user and driver have GPS-enabled devices with the e-hailing app installed.
Acceptance criteria	The app displays the driver's real-time location accurately on the map. ETA updates are timely and reflect current traffic conditions. Notifications are sent to the user at key stages (driver assigned, driver approaching, driver arrived). The driver's and passenger's location data are updated consistently without significant delays.
Post-condition	Driver and passenger can communicate through the app. The journey is logged for future reference and safety. Successful in showing/updating the user/drivers real time location

Scenario 4: 3rd party integrated payment services

The following table is a form-based specification of the integrated 3rd party payment services of an e-hailing app. The 3rd party payment service provided by commonly used banks across the e-hailing app's demographic and provide a more trustworthy transaction process without needing to develop a payment system from scratch for the application itself.

Title	3rd party payment services
Function	Providing payment services directly from the user's registered bank based on their bank card, bank account or e-Wallet accounts.
Description	This functionality allows user to enter their payment details by redirecting them to external pages from their bank or e-wallet service providers. This ensures a transparent, trustworthy and more secure transaction process between users and e-hailing application when paying for rides, reloading credits etc.
Pre-Condition	User must have an existing bank account, bank card or e-Wallet accounts. User has chosen the ride option, pick-up location and destination.
Acceptance criteria	The app will redirect the user to a dedicated web page to authenticate the user's bank details. Users will enter the authentication details like their bank account password, OTP (One-time password), mobile phone etc. to verify their payment process.
Post-Condition	Notifications regarding fee charges will be sent from the respective wallet/bank platform they have chosen to make payment with. The user will be directed back to the app itself and will have their payment details saved for next transaction, should they enable it.

Scenario 5: Notification

The following table is a form-based specification for the notification functionality of an e-hailing app. Notifications keeps users informed about status of their rides and account activities in a timely and accurate manner thus maintaining user engagement and satisfaction.

Title	Notification
Function	To provide timely and accurate notification to user about their ride and account activities.
Description	This functionality allows users to receive notification about their rides such as booking confirmation, drivers arrival, ride completion as well as provide users with notification about their account activities such as payment status.

Pre-condition	User has an active booking.
Acceptance criteria	<p>Users receive notifications immediately after booking a ride (i.e. driver details, ETA, and any changes to delay or arrival time).</p> <p>System sends delay notifications if applicable</p> <p>System sends ride completion and payment confirmation notification.</p>
Post-condition	Users informed of all ride-related activities in a timely manner.

|

Quality Attributes

Performance

- Response Time
- Scalability

Source	Internal hardware, System Load
Stimulus	High user activity, multiple concurrent bookings
Artifact	Mobile application, Backend servers, Database
Environment	System operating under peak load conditions
Response	<ol style="list-style-type: none">1. Handling User Requests:<ul style="list-style-type: none">• The system should maintain swift response times even during peak usage.• System should distribute requests evenly across servers to prevent any single server from becoming overloaded.• Use of caching mechanisms to reduce database load and improve response times.2. Performance Optimization:<ul style="list-style-type: none">• The system should scale dynamically to handle increased user load, ensuring consistent performance.• Optimize database queries to be faster and more efficient.
Response Measure	<ol style="list-style-type: none">3. The application responds to user interactions within 1 second.4. The initial loading time of the application does not exceed 3 seconds.5. Support use of at least 50,000 concurrent users during runtime.

Availability

- System Uptime
- Data recovery
- 3rd party service uptime
- Traffic handling.

Source	Physical Infrastructure, Software, Network Connectivity
Stimulus	High user demand, Network outage, Hardware failure
Artifact	Server infrastructure, Load balancer, Database, Network components
Environment	System operating during peak hours, unstable network conditions, data centres
Response	<ol style="list-style-type: none">1. Ensure continuous service availability<ul style="list-style-type: none">- The system should handle disruptions and continue to operate without significant downtime.- System should distribute user requests across multiple servers to prevent overload of a single server.- System perform backup every 1 week.- System should have backup servers that automatically take over in case of failure of primary server.2. Scalability and Performance<ul style="list-style-type: none">- System should be able to scale up and down based on user demand this will ensure consistency in performance.- System should have efficient database queries and indexing to improve response time and reduce latency.3. Ensure disaster recovery and monitoring<ul style="list-style-type: none">- The system should implement a recovery strategy that minimizes downtime and data loss in case of failure.- Real time monitoring and event logging to detect service disruption quickly and logging critical events for analysis and future improvement.
Response Measure	<ol style="list-style-type: none">1. System uptime of at least 99.9% should be achieved to ensure constant system accessibility most of the time.2. System can handle up to 50,000 of concurrent users without degrading performance even during peak hours.3. Database queries are optimized to reduce latency and improve response times.

	4. Detecting and alerting administrators to any service disruptions within 1 minute.
--	--

Security

- Authentication and authorization
- Data Privacy

Source	Malicious Actor, Hardware, Software, Physical Infrastructure, Physical Environment, Compromised user credentials
Stimulus	Unauthorized login attempt, attempt to access sensitive data
Artifact	User authentication module, Access control system, Logging system
Environment	User or driver accessing the system via app or web interface
Response	<ol style="list-style-type: none"> 1. Block unauthorized access <ul style="list-style-type: none"> - The system must prevent any unauthorized access attempt from succeeding - The system should request personal data which shall be used for the purpose of identification only 2. Notify legitimate user <ul style="list-style-type: none"> - An alert should be sent to the legitimate user informing them of the unauthorized attempt. - System should only allow the new user to login if the legitimate user provides proof of identification 3. Log the event <ul style="list-style-type: none"> - The attempt must be recorded in the system logs for further analysis and auditing.
Response Measure	<ol style="list-style-type: none"> 1. Unauthorized access events are logged within 1 second: The logging system records the attempt almost instantaneously, allowing for quick review and analysis. 2. Legitimate user notification is sent within 5 seconds: The legitimate user is promptly informed of the unauthorized attempt, enabling them to take necessary actions to secure their account. 3. After 5 login attempts the user is denied access. 4. 99% of unauthorized access attempts are successfully blocked. 5. System can handle up to 100 unauthorized access attempts per second without failing

Interoperability

- 3rd party service
- API integration
- Cross platform compatibility.

Source	End user (new user), Sign-up menu interface.
Stimulus	End users decided to register a new account for the e-hailing application through third party services, such as Google Sign-in by Google Identity Services (GIS).
Artifact	Sign-up menu interface integrated within the app from GIS.
Environment	When user accesses the e-hailing application for the first time, while having stable internet connectivity.
Response	<ol style="list-style-type: none"> 1. E-hailing app successfully redirects end user to a Google Sign-In page. 2. After entering their google credentials, authentication will be processed, and a new e-hailing account linked to the Google account will be created. 3. GIS page will then redirect the user back to the application and enter their remaining username and account details. 4. For payments, the e-hailing app integrates with third-party payment gateways (e.g., PayPal, Stripe) to facilitate seamless payment processing
Response measure	<ol style="list-style-type: none"> 1. The total time taken for the end user to finish their 3rd party login services should be roughly under 30 seconds long. 2. As the goal is to provide convenience, total time taken for registration should be shorter than registering natively through the e-hailing application itself. 3. Ensure consistent registration process without encountering any errors or additional authentication steps. 4. User successfully create an account linked to their existing email account and proceed to use the application without needing to enter their credentials. 5. Payment processing time should be under 10 seconds to ensure a smooth user experience.

Usability

- User interface interaction
- Task completion
- System completeness

Source	End Users (both existing and new users), System Interface, Physical Device, Network Connectivity
Stimulus	End-users navigating throughout the different sections of the application, perform actions within the application.
Artifact	User interface, Input Validation system, Error Handling Mechanism, Application Workflow
Environment	User/Driver interacting with the application via their smartphones, stable network connectivity
Response	<p>Ease of Use</p> <ul style="list-style-type: none">- The system must ensure that all features of the application is easy to understand and use.- All functionalities should be accessible and understandable without prior training. <p>Quick Completion</p> <ul style="list-style-type: none">- Key tasks (e.g., booking a ride) should be completed swiftly.- The system should load and execute actions without noticeable delays. <p>Error Handling and Feedback</p> <ul style="list-style-type: none">- The system should provide clear and concise feedback on any errors encountered by end-users. <p>Consistency throughout Devices</p> <ul style="list-style-type: none">- User experience should be consistent and predictable across different devices.
Response Measure	<ol style="list-style-type: none">1. An average user satisfaction rate of 4 out of 5 should be achieved for all the app functionalities.2. Maintain a low user error rate across all functionalities.3. Time to complete key tasks should be under 5-7 minutes.4. Performance of the application is regularly evaluated and maintain to improve efficiency.5. Ensure consistency and reliability of user experience throughout different devices.

Utility Tree

Quality attribute	Attribute refinement	ASR
Performance	Response Time	During peak operation time with multiple concurrent users the system maintains swift response time and responds to user with maximum response time of 1000ms. (H, H)
	Scalability	The system can handle increase in user load by scaling dynamically and support the use of at least 50,000 concurrent users during runtime without side effects. (H, H)
Availability	System Uptime	The system developers design to achieve uptime through combination of redundant hardware, failover mechanisms and proactive maintenance. The system achieves 99.99% uptime for the overall system. (H, H)

Justification	Impact on Business Value	Impact on Architecture
The system relies on fast interactions, quick user response enhances user experience as it reduces waiting time.	H Fast response influences user satisfaction and reduces the likelihood of abandoning the application.	H Swift response time under peak load conditions necessitates efficient backend processing, effective load balancing, and optimization mechanisms.
Scalability ensures that the service remains responsive and functional as more users' book rides and interact with the app simultaneously.	H Without the ability to scale, the system risks failing under high usage, leading to downtime, loss of users, and damage to the company's reputation.	H Dynamic scalability requires a scalable architecture, including auto-scaling capabilities, efficient resource management, and robust infrastructure capable of handling dynamic loads. Implementing this will require significant architectural planning.
Ensuring a high system uptime is essential for maintaining continuous service availability, which is vital for operational stability.	H Frequent outages can lead to lost business opportunities, user frustration as well as damage to the brand's reputation.	H This level of uptime requires a sophisticated architecture with redundancy, failover capabilities, and robust monitoring, necessitating significant investment and complexity.

	Data recovery	The system backup data weekly and has a contingency plan for data recovery. (M, M)
	3 rd party service uptime	99.99% availability for 3 rd party services, regularly check subscription status if any. (M, M)
	Traffic handling	During high demand and peak hours, the system handles user requests within 5000ms no matter the amount of concurrent traffics. (M, M)

Regular data backups and a contingency plan for recovery ensure minimal data loss and quick restoration of services in case of failure, preserving data integrity and operational continuity	M Effective data recovery supports regulatory compliance and operational resilience but is less critical for day-to-day user experience.	M Requires additional resources for backup systems and recovery planning, adding complexity but ensuring data integrity and recovery capability.
High availability of 3 rd party services and regular monitoring is necessary to maintain seamless functionality, as these services are integral to the e-hailing app's operations.	M Ensuring reliable 3 rd party services helps maintain overall service quality but may not be as critical as internal system uptime.	M Integration and monitoring of the external services, requiring failover mechanisms and regular status checks.
Efficient traffic handling is essential for maintaining performance for it ensures that user requests are processed promptly, even under high load conditions.	M Efficient traffic handling ensures that user requests are processed promptly, even under high load, which is crucial for maintaining user satisfaction.	M Effective traffic handling requires a scalable infrastructure with robust load balancing and performance optimization. This involves designing the system to handle varying traffic loads without compromising performance.

Security	Authentication and authorization	<p>Upon the user attempt to access the application, the system requests 2 factor authentication during log in and email requirement during sign up.</p> <p>Unauthorized users try to login to the application, the users are logged out within 1 second and notifications are sent to legitimate user within 5 seconds. Attempts are recorded in the system log and the user is denied access after 5 attempts and 99% of unauthorized access is successfully blocked. (H, H)</p>
	Data Privacy	<p>When handling data, the system encrypts data between transactions, transit or during rest.</p>

Enhances security to the system, because it ensures robust user verification and quick detection of unauthorized access.	H Authentication and authorization ensure prevention of data breach thus protecting user trust and company reputation which is critical for business sustainability.	H Implementing 2-factor authentication and real-time logging and notification systems requires a sophisticated and resilient infrastructure to ensure latency and availability.
Protect sensitive data from being accessed by unauthorized users.	H Data privacy is crucial for compliance with legal standards and maintaining user trust, data breach can lead to financial and reputational damage.	M Encryption of data requires additional computational resources to secure data which needs careful design to ensure performance is not significantly degraded.

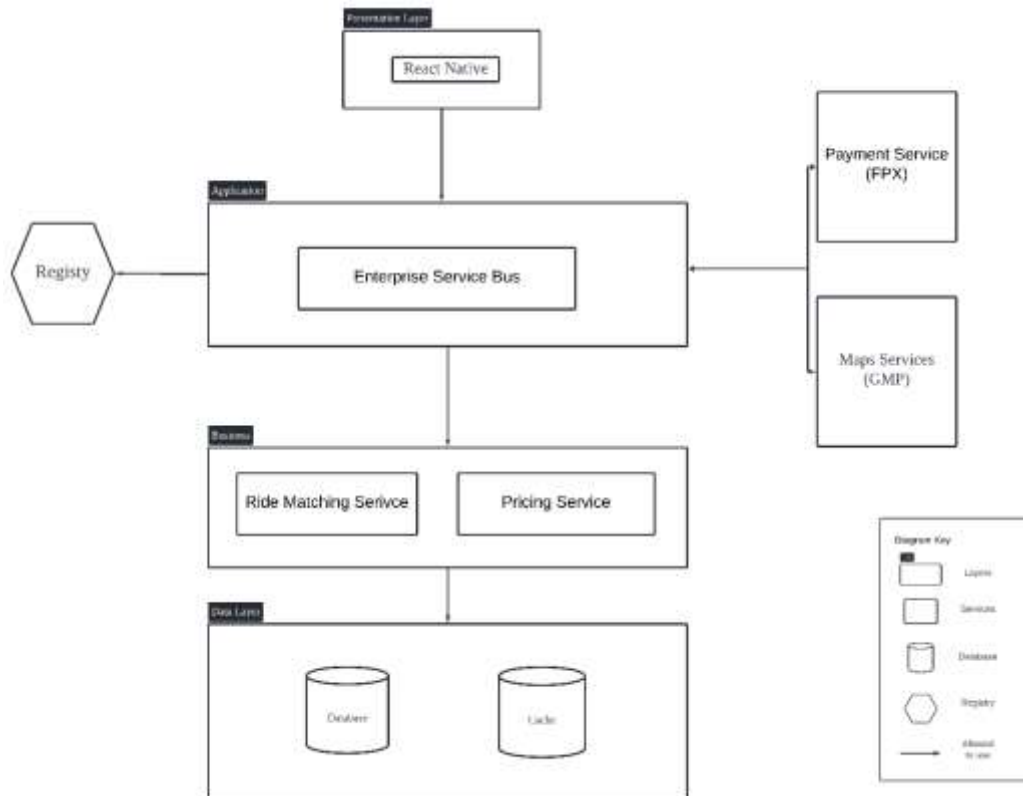
Interoperability	3 rd party service, API integration	When users sign up with and make payments to the third-party services such as Google Sign-In and payment gateways (TNG, Cards) the system enables users to register and make payments within 30 seconds. Google Maps Platform services should also update rider and passenger's location every 1 - 1000ms (H, M)	Reduces entry barrier and offers convenience by allowing registration using google accounts and facilitates seamless payments through trusted gateways, as well as consistent location tracking for users.	H 3 rd party service, API integration increases user acquisition and satisfaction this leads to high conversion rates and better user retention.	H Requires robust integration with Google APIs and payment gateways including handling authentication, data transfer, payments processing and real-time location tracking.
	Cross platform compatibility	When new users sign up the system supports the use of devices across iOS, Android and Web platforms. (H, M)	This is important as it ensures the system is accessible to all devices and platforms and thus facilitates consistent user experience.	H Cross-platform compatibility broadens the user base, enhancing market reach and user satisfaction. Consistent user experience across platforms can lead to higher user engagement and retention, while reliable payment processing increases trust and transaction volume.	M Requires additional effort to ensure compatibility and optimal performance across different platforms with minimal bugs.

Usability	User interface interaction	Upon users interacting with different sections of the application, the system interface should be intuitive, with clear navigation links buttons and a simple layout.	Ensures easy navigation through the app.	H User-friendly interfaces increase user satisfaction and retention rates.	M Requires extensive application of human computer interaction techniques and methodology to verify its capability, such as user acceptance testing and so on.
	Task completion	When users book rides under stable network connection the time taken to book a ride should be a maximum of 1.5 minutes. (H, M)	Quick task completion is crucial for user satisfaction, especially in time-sensitive scenarios like booking a ride.	H Faster task completion times improve user satisfaction and the likelihood of repeated use. Quick and efficient booking processes can be a competitive advantage in the e-hailing market.	M Requires optimizing the data and interaction flow of the software to ensure proper task completion within a timeframe, and efficiency.
	System completeness	The system is fully developed and ready to operate. (H, H)	A fully developed and bug-free system enhances reliability on the system.	H Ensures the system can be confidently launched and marketed	H Requires thorough system testing during late stages to ensure no bugs and issues are present in the system. The system should be compiled and have the necessary services hosted online.

Section 3: Views

Module View (Layer)

a. Primary Presentation



b. Element Catalog

I. Presentation Layer

- a. React Native: The front-end framework used for building the user interface.
 - Responsibilities

- a. Handles user interactions and displays data to users.

II. Application Layer

- a. Enterprise Service Bus: Middleware that facilitates communication between various services.
 - Responsibilities:
 1. Integrates different services and applications.
 2. Ensures that data is routed correctly between services.

III. Business Logic Layer

- a. Ride Matching Service: A service responsible for matching riders with available drivers.
 - Responsibilities:
 1. Processes ride requests.
 2. Matches riders with drivers based on criteria such as location and availability.
 3. Uses the pipe-and-filter architectural style for processing.

- b. Pricing Service: A service responsible for calculating ride prices.

- Responsibilities:
 1. Computes the cost of a ride based on factors like distance, time, and demand.
 2. Applies pricing algorithms and discounts.
 3. Uses the pipe-and-filter architectural style for processing.

iv. Data Layer

a. Database: The primary storage for application data.

- Responsibilities:
 1. Stores persistent data such as user information, ride history, and pricing data.
 2. Ensures data integrity and consistency.

b. Cache: Temporary storage to improve performance.

- Responsibilities:
 1. Caches frequently accessed data to reduce database load.
 2. Enhances response time for read-heavy operations.
- External Services

v. Registry: Service registry for discovering and managing service instances.

- Responsibilities:
 1. Registers service instances.
 2. Facilitates service discovery for load balancing and failover.

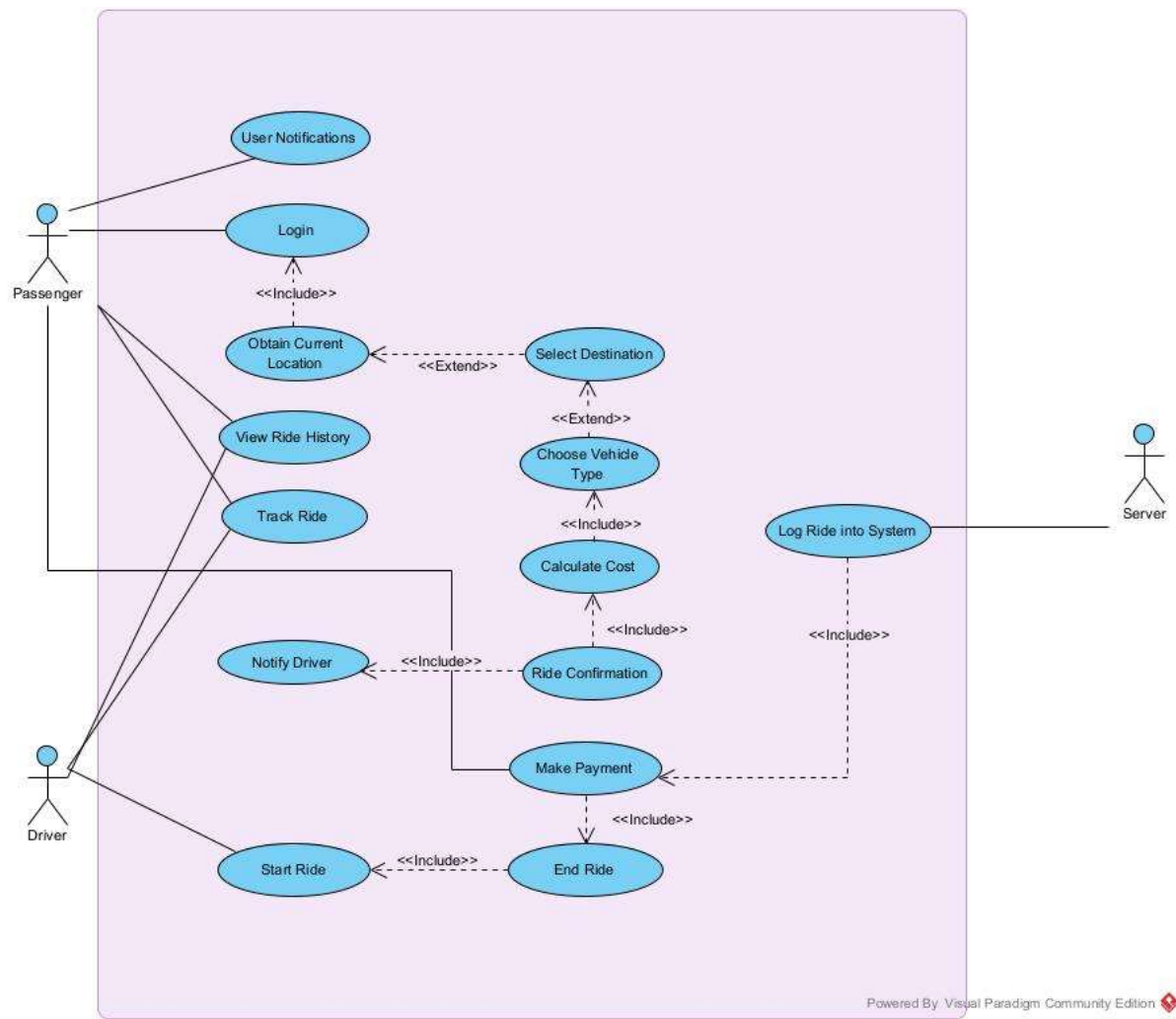
vi. Payment Service (FPX): External payment gateway service.

- Responsibilities:
 1. Handles financial transactions.
 2. Processes payments and refunds.

vii. Maps Services (Google Maps): External mapping service.

- Responsibilities:
 1. Provides geolocation data.
 2. Offers routing and navigation services

c. Use Case Diagram



Rationale

1. React Native

React Native is an open-source UI software framework that facilitates useful tools for developing cross platform applications through a single codebase, thus reducing the development time while allowing for cross platform uses among users with various devices. Not only that, react native also provides interactive user interface that also supports real-time event updates which will be used to request for services represented in the application layer, hence supporting the reasoning behind the usage of layered view.

2. Financial Process Exchange (FPX)

FPX is a Malaysia-based online transaction payment system that is launched in 2004. In collaboration with participating Malaysian banks, FPX promotes the use of real-time payments online by customers through their own bank credentials, providing secure and immediate fund transfers from the buyer's bank account to the merchant's one. In this context, FPX is crucial as it serves as an external payment gateway service for users and riders within the app itself, allowing for quick, transparent, safe and trustworthy transactions throughout its runtime.

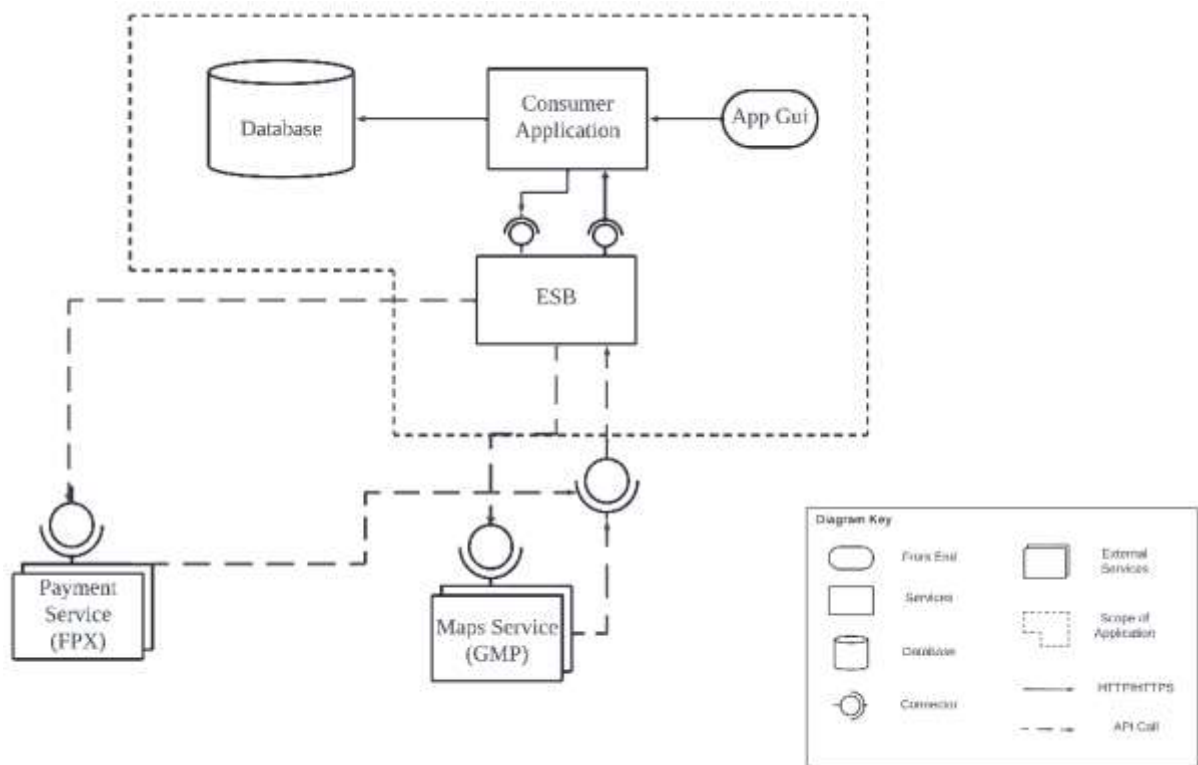
3. Google Maps Platform **API**

Google Maps Platform ~~contains~~is a developer ~~tool~~tool that ~~consist~~consists of APIs and services that provides real-time traffic and global routing services throughout almost, if not all countries across the globe. With that in mind, this service is chosen to compute necessary traffic related information such as travelling distance, estimation time of arrival of destinations, current traffic which are also crucial for calculating derived quantities such as fares based on demand, current time and traffic level.

C&C Views

1. SOA

a. Primary Presentation



b. Element Catalog

- i. Consumer Application: The application used by consumers to interact with the services.
 - Responsibilities:
 1. Provides a user interface (App GUI) for consumers.
 2. Interacts with the database for data retrieval and storage.
 3. Communicates with the ESB to access various services.
- ii. App GUI: The graphical user interface of the consumer application.
 - Responsibilities:
 1. Facilitates user interaction with the consumer application.
 2. Displays data to the user and collects user inputs.
- iii. Enterprise Service Bus (ESB): Middleware that facilitates communication between the consumer application and external services.
 - Responsibilities:
 1. Integrates various services and applications.
 2. Manages data routing, transformations, and message handling.
 3. Connects with external services such as payment and mapping services.
- iv. Database: The primary storage for the application data.
 - Responsibilities:

1. Stores persistent data such as user information and transaction history.
 2. Ensures data integrity and consistency.
- v. Payment Service (FPX): External service for processing payments.
- Responsibilities:
 1. Handles financial transactions.
 2. Processes payments and refunds.
- vi. Maps Service (~~GMP~~GMS - Google Maps ~~Platform~~Service): External service for providing mapping and geolocation data.
- Responsibilities:
 1. Offers geolocation data and mapping services.
 2. Provides routing and navigation features.

Process Flow

1. **Consumer Application** connects to the **Database** for data storage and retrieval.
2. **App GUI** interacts with the **Consumer Application** for user input and display.
3. **Consumer Application** uses the **ESB** to communicate with external services.
4. **ESB** connects to the **Payment Service (FPX)** for processing transactions.
5. **ESB** also connects to the **Maps Service (~~GMP~~GMS)** for geolocation and routing services.

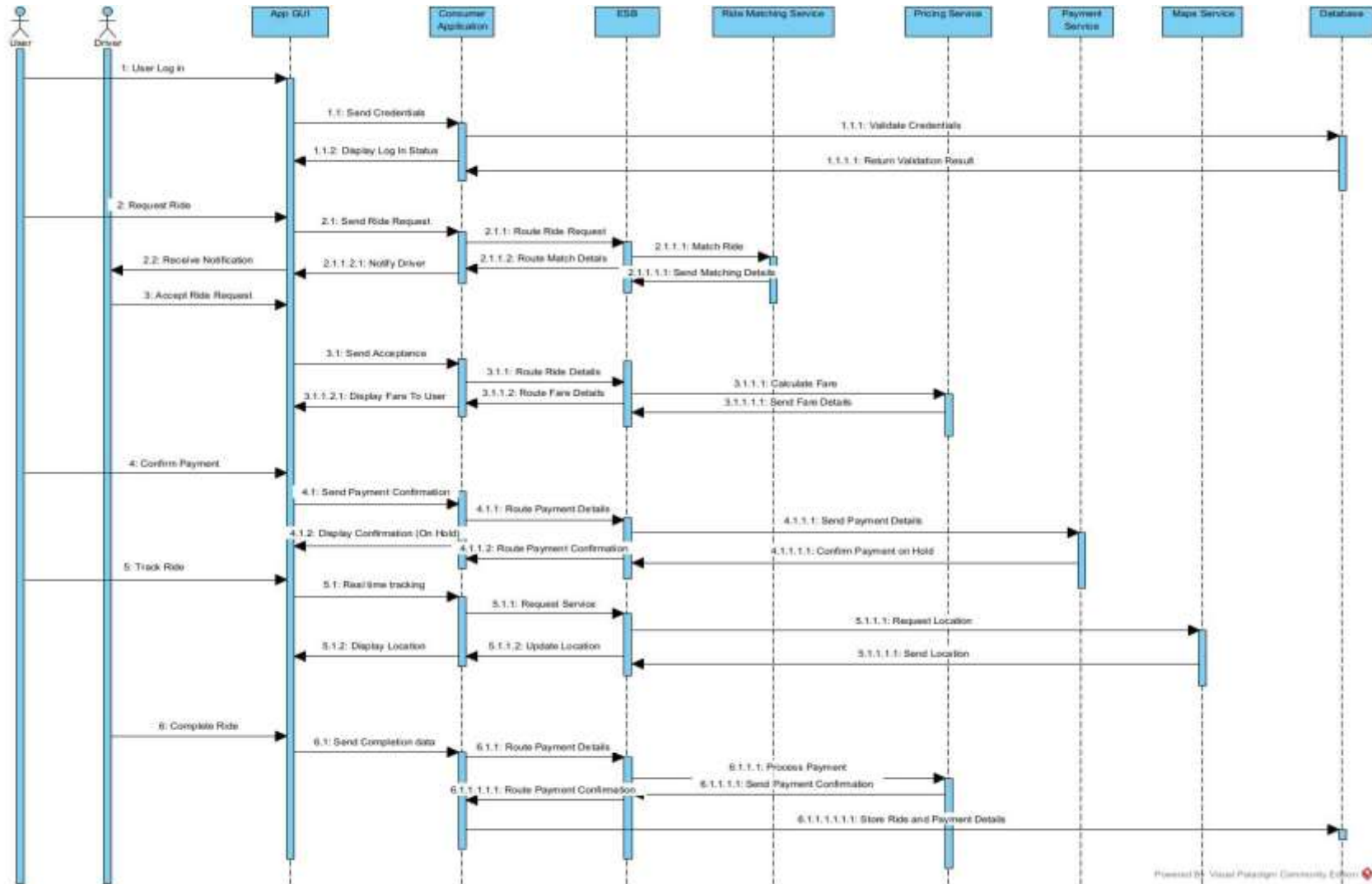
Relationship to Layered Architecture

- **Presentation Layer:** Represented by the App GUI and the Consumer Application.
- **Application Layer:** Primarily the ESB, which integrates the consumer application with external services.
- **Data Layer:** Represented by the Database.
- **External Services:** Payment Service (FPX) and Maps Service (~~GMP~~GMS) interact with the Application Layer through the ESB.

Rationale

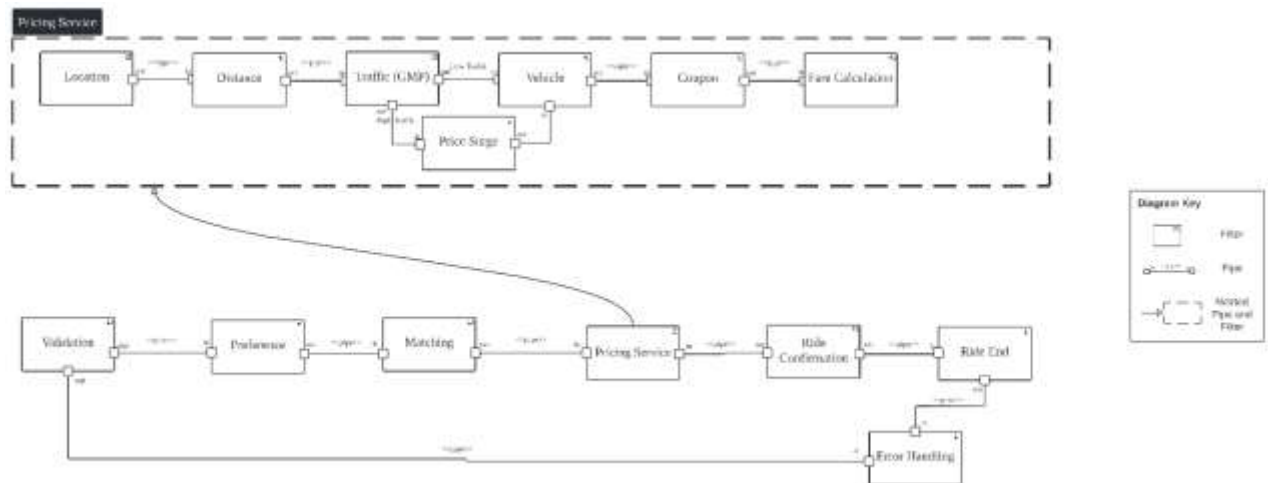
Refer to module view's rationale for chosen services (FPX & Google Maps Platform)

2. Sequence Diagram



3. Pipe and Filter

I. Primary Presentation



II. Element Catalog

Ride Matching Service

- **Validation:** Initial stage that verifies the input data.
 - Ensures that the ride request data is complete and valid before processing further.
- **Preference:** Filters ride requests based on user preferences.
 - Matches ride requests with user preferences such as vehicle type, driver gender, etc.
- **Matching:** Matches riders with available drivers.
 - Finds suitable drivers for ride requests based on location, availability, and other criteria.
- **Pricing Service:** Calculates the fare for the ride.
 - Calculates the ride cost using various factors such as distance, traffic, vehicle type, and applicable discounts. (Nested Pipe-and-Filter)
- **Ride Confirmation:** Confirms the ride booking with the rider and driver.
 - Sends notifications to both the rider and the driver, confirming the match and price.
- **Ride End:** Marks the completion of the ride.
 - Finalizes the ride, logs the ride details, and updates the system status.
- **Error Handling:** Handles any errors that occur during the ride matching process.
 - Manages and logs errors.

Nested Pricing Service (within Ride Matching Service)

- **Location:** Determines the starting location of the ride.

- Identifies and processes the starting point of the ride request.
- **Distance:** Calculates the distance of the ride.
 - Computes the total distance from the starting location to the destination.
- **Traffic (GMS):** Considers current traffic conditions using Google Maps Service.
 - Assesses the traffic conditions to adjust the ride pricing accordingly.
- **Vehicle:** Factors in the type of vehicle chosen for the ride.
 - Applies different pricing based on the vehicle category (e.g., standard, premium).
- **Price Surge:** Applies surge pricing based on demand.
 - Adjusts the base fare considering high demand periods, typically using a multiplier.
- **Coupon:** Applies any available discounts or promotional codes.
 - Reduces the fare based on applicable coupons or promotional offers.
- **Fare Calculation:** Final step in computing the total fare.
 - Totals all factors (distance, traffic, vehicle type, surge, coupons) to calculate the final ride cost.

III. Process Flow

Ride Matching Service:

- Input data goes through **Validation, Preference, and Matching**.
- **Pricing Service** (nested pipe-and-filter) is invoked to calculate the fare.
- The calculated fare is used in **Ride Confirmation**.
- The ride is completed in **Ride End**.
- Any errors are managed by **Error Handling**.

Pricing Service (nested):

- Sequentially processes location, distance, traffic, vehicle type, price surge, coupons, and final fare calculation to determine the ride cost.

Special Arrows

1. Arrow from Pricing Service to Pricing Service (nested):

- **Description:** Indicates that the Pricing Service is a nested pipe-and-filter system within the overall Ride Matching Service.

Section 4: Mapping between Views

Mapping Table

Layered Architecture	SOA Components	Pipe-and-Filter Components	Description
Presentation Layer	Consumer Application (App GUI)	Validation, Preference	User inputs and preferences.
Application Layer	ESB	Validation, Preference, Matching, Pricing Service, Ride Confirmation, Ride End, Error Handling	Core business logic processing.
Business Logic Layer	-	Ride Matching Service	Ride matching logic.
Business Logic Layer	-	Pricing Service	Pricing calculations and adjustments.
Data Layer (Database)	-	-	Data storage and retrieval.
Data Layer (Cache)	-	-	Data caching for quick access.
External Services	Payment Service (FPX), Maps Service (GMP)	Pricing Service	External data integration for pricing. (GMP)

Section 5: Rationale

The architectural decisions for our e-Hailing system were made to address key requirements and constraints while ensuring scalability, performance, and maintainability.

The Layered Architecture is the foundation of our system, providing a clear separation of concerns across different layers. Each layer in the architecture serves a distinct purpose: the Presentation Layer with React Native for the user interface, the Application Layer containing the SOA components, the Business Logic layer which contains the Ride Matching and Pricing Services, and the Data Layer for storage and caching.

Within the Application Layer, we integrated Service-Oriented Architecture (SOA). The Enterprise Service Bus (ESB) acts as the backbone of our SOA, facilitating communication between services and ensuring interoperability. This modularity is crucial for managing the complexity of the e-Hailing system, allowing independent development, maintenance, and scaling of each service. The decision to use SOA within the Application Layer was influenced by the need for high scalability and flexibility.

For the Ride Matching and Pricing services, we utilized the Pipe and Filter pattern. In our system, data flows through multiple stages, including input validation, calculation, and filtering, before producing the final output. The Pipe and Filter pattern enhances modularity, as each filter can be developed, tested, and maintained independently. This modularity also provides flexibility and reusability, as filters can be reused across different services and new filters can be added without affecting existing ones. Additionally, the pattern supports concurrent processing of data, which can improve performance because of its parallelism, especially during high demand periods.

Section 6: Directory

Technical Terms:

- Service-Oriented Architecture (SOA): Architecture pattern focusing on services as fundamental components.
- Layered Architecture: Design pattern organizing system into layers with distinct responsibilities.
- Pipe and Filter Pattern: Architectural pattern where data flows through a series of processing stages.
- API: Application Programming Interface – set of rules for software interactions.
- GPS: Global Positioning System – satellite-based navigation system.
- OTP: One-Time Password – security feature for user verification.
- Load Balancer: Distributes incoming traffic across multiple servers to ensure reliability.
- Redundancy: Backup systems to ensure continuous operation during failures.
- Caching: Storing data temporarily for quick access.
- User Interface Interaction: Intuitive design
- Task Completion: Max 1.5 minutes booking
- System Completeness: Fully developed system
- Layered Architecture: Defines system layers.
- SOA Components: ESB, services
- Pipe-and-Filter Components: Data processing stages.

Acronyms:

- SOA: Service-Oriented Architecture
- API: Application Programming Interface
- GPS: Global Positioning System
- OTP: One-Time Password
- UI: User Interface
- QA: Quality Assurance
- DB: Database
- UX: User Experience
- FPX: Financial Payment Exchange (or Payment Service)
- GMP: Google Maps Platform (Maps Services)
- ESB: Enterprise Service Bus
- ASR: Attribute Satisfying Requirements

References

https://flylib.com/books/en/2.121.1/module_layered_view.html

<https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=146280205>https://flylib.com/books/en/2.121.1/module_layered_view.html

<https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=146280205>

Section 7: Self Reflection

Aisha

Designing the architecture was quite a challenge for me even with all the quality attributes and business requirements laid out for reference, I believe I lacked in depth understanding of how the components work but throughout this assignment I believe I learnt more about how important it actually is to have your architecture decided before starting a coding project as I thought about how much more complicated this project would be in a larger scale. Overall, this assignment was very interesting and despite how challenging it was to complete, I am satisfied with the outcome.

Emily

As the business analysts of this project, my primary task was to gather and prioritize the business requirements. My goal was to ensure that they were effectively incorporated into the system's architecture. One of the critical aspects of my role was utilizing the Utility Tree technique to prioritize the architectural significant requirements (ASRs). This method allowed us to focus on the most impactful features, ensuring our architectural decisions were aligned with both user needs and business goals. This project has enhanced my understanding of the critical role a business analyst plays in bridging business objectives with technical implementation.

Justin

From a software tester's perspective, the planning phase was highly insightful. Reviewing the architectural document helped identify key areas for testing, such as performance, scalability, and security. This allowed me to develop a targeted test plan focusing on benchmarks like response times and system uptime. Understanding these requirements guided the creation of specific test cases for performance, load, and security testing. Overall, this preparatory work was crucial for ensuring a comprehensive and effective testing strategy, setting the stage for a reliable and robust application.

Catherine

My focus as a quality engineer for this project was to ensure that the architectural design and implementation will meet the highest quality, performance and reliability standards. This document helped to identify potential quality issues and thus helped me to define comprehensive future testing in performance, load and security. Collaboration with the system design and development teams in the future will ensure that the system will support seamless and reliable user experience. This project reinforced the importance of early quality consideration in architecture, highlighting the value of a well-thought design in anticipating future testing needs. Overall, it was a valuable learning experience, it enhanced my understanding and importance of quality engineering in planning stages of complex software projects.

Alkin

As the project manager of this project, one of the main obstacles is to establishing good communications between my team members. As software architecting requires not just a single individual to carry out the designing procedures, it is my job to ensure that each progress within project is maintained within the scope and retaining consistency throughout each documentation. Apart from that, I am also required to provide direction and manifest leadership skills, as well as relevant technical skills that are essential for delivering a well-documented architecture of the e-hailing system. As my past experience solely involves learning the use of various programming languages and minimal values of actual software engineering process, this assignment not only taught me the insights of creating a well-planned software, but also other leadership skills that are essential for achieving the necessary results. In conclusion, this project has greatly widened my view on software development disciplines, I hope to apply these skills consistently in future projects and thus satisfied with the overall outcome of the project.