Institute of Systems Science
National University of Singapore

MASTER OF TECHNOLOGY IN SOFTWARE ENGINEERING

Graduate Certificate Examination

Subject: *Architecting Scalable Systems*

Sample Examination Questions

SECTION A

SECTION A

Question 1 *(Total: 23 Marks)*

Refer to the case study in **Appendix A** and answer the following questions:

a. One of the measures of success of a platform is how it grows its ecosystem by growing its reach with new types of interactions and numbers of customers. One seed that the current platform has is 'Route. Amongst many others, some interactions are 'plan a trip', 'sign up as a driver' and 'complete a trip'.

With respect to the seed and the interactions described above, explain **two** aspects of the architecture that is crucial to the success of the platform.

(2 Marks)

b. A platform consists of reusable or shared components, entities and services using which specific applications and services can be built.

i. Domain entities make up an essential part of the reusable assets of a platform. Analyze **only** the descriptions of the significant use case flows in **Section 6** of the case study and using the Domain Driven Design approach, identify the **domain objects and their relationships**. Capture them in a UML class diagram.

ii. Architecting services that can be shared across applications and exposed to be consumed by external application is an essential characteristic of a platform. Based on the answer to part (i) above and using the Domain Driven Design approach, identify the recommended **shared services** for the platform described in the case study.

(9 Marks)

c. Platforms enable creation of business ecosystems and one such enabler is the services exposed to partners and other participant applications/services of the ecosystem. One such participating application (DaDiCommuter app) is a mobile app developed for use by public commuters.

i. From the list of shared services identified in Q1(b)(ii) identify **two services** that you would expose as APIs to the commuter app . Explain briefly, the functionality in the mobile app that will make use of these services.

ii. For each of the **two** services, which architectural style would you choose from RESTful and GraphQL? Justify your answer.

iii. Pick any **one service** to be exposed in Q1(c)(i) and design the service API according to the architectural style you have recommended for that service in Q1(c)(ii).

(6 Marks)

d. Capacity planning is a hard problem especially when the target is constantly moving. With an ever-increasing customer base for the 'DaDi' platform, there is a constant need for scaling the platform services in order to adhere to the scalability, availability and evolvability of the services. Based on the project growth of the platform, identify **two** key **infrastructure** metrics and **two application** specific metrics and explain why these metrics are important to be captured and monitored.
(Hint: You must identify at least one or two services that will have implications and explain what effect they will have on the platform.)

(4 Marks)

e. After the platform services are implemented, they would be deployed into production environment. Assuming that the reusable services identified in Q1(b)(ii) are to be deployed using containers. One architectural decision is that those services with high communication affinity (defined by the number and size of messages exchanged over time) should be considered to be collocated in the same container. Propose **two services** that are **least** suitable to share the same container. Justify your answer.

(2 Marks)

Question 2 *(Total: 13 Marks)*

Refer to the initial requirements specified in **Section 1** of the case study in **Appendix A** as well as the projected growth of the platform and answer the following questions:

a. Consider the 'DaDi' platform, the initial requirements and system actors. The architectural team has decided that the trip information must be persisted only via its services. Both trip and payment data are transactional in nature. As observed from the significant use cases in Section 6 of the case study, the commuter's trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data will be collected and provided to the partner companies by the platform provider as authorized under the Land Transport Authority (LTA) governance code. Payment records capture information regarding trip, bill amount and payment type. Answer the following questions:

   i. Recommend the **storage type** and a **cloud vendor offering** for persisting the Trip data and Payment data.
   Hint: Storage types could be NoSQL (Key- Value, Graph, Column Family, Document, In-Memory) or NewSQL Stores.
   ii. What **scalability requirements** do you envision for them?
   iii. Design the **sharding** policies as appropriate. Justify your answer.

   (4 Marks)

b. Consider the "Plan a Trip" use case that includes "Compute Optimal Routes" use case. The efficiency of this use case depends on the AI processing model and the data set. Initially, as the platform may not have enough data set, it will probably have to rely on other similar services by making API calls. However, as the platform usage grows, DaDi will leverage on the data collected in the platform and will need to crunch the data to make inferences on optimal routes. Evaluate these **architectural styles** (monolithic, microservices, serverless, lambda, kappa and zeta) for their suitability for the development and deployment of this use case and recommend your choice. The key criteria for your evaluation are extensibility, scalability, technical feasibility, cost and efficiency.

   (5 Marks)

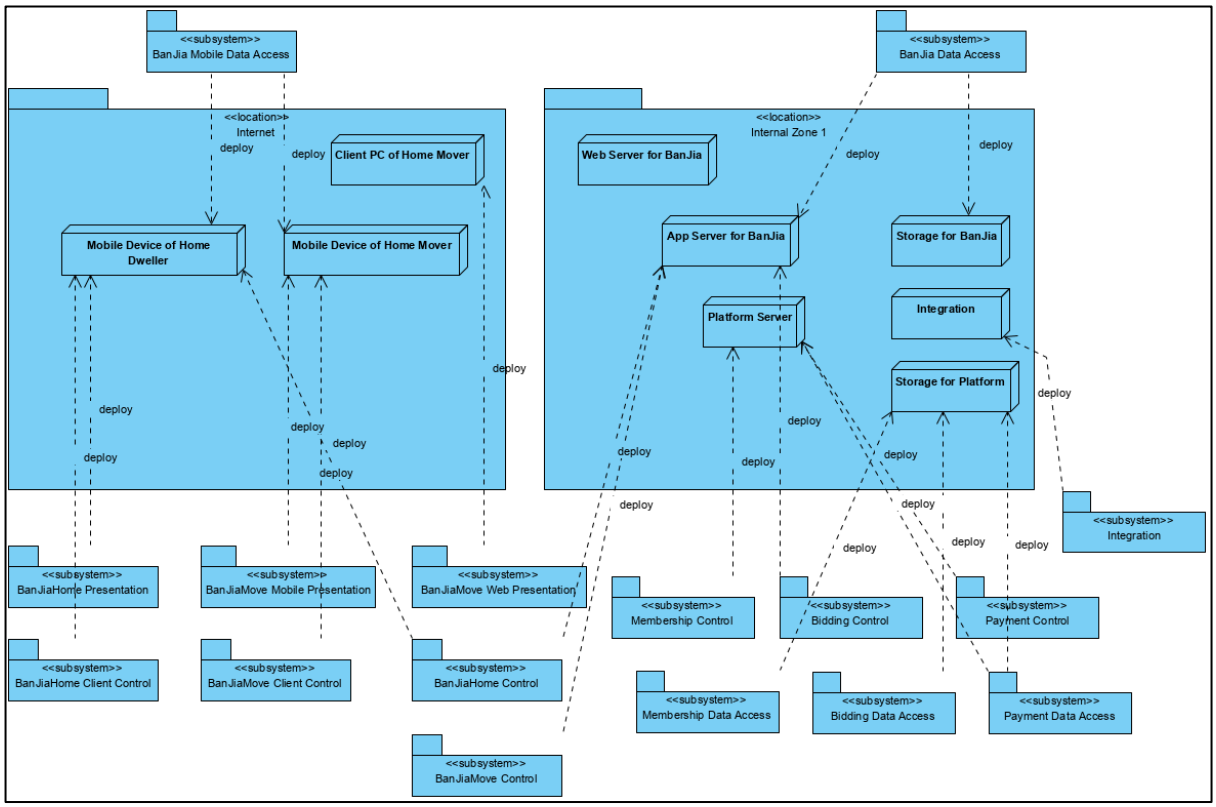
c. The "Hail a Taxi thru App" and "Hail Private Car/Bus" use cases should be uniformly designed across various partners (tenants). They act as an aggregator for services by Private Drivers as well as Taxi App Providers like Grab and Gojek. How would you design the **tenancy model** that works seamlessly for both use cases? If a new tenant requests for a **custom requirement** for the platform to share trip information related to the tenant, explain the **potential impact** on the tenancy model and other aspects of architecture.

   (4 Marks)

Question 3 *(Total: 14 Marks)*

Refer to the additional requirements specified in **Section 2** of the case study in **Appendix A** and answer the following questions:

a. Analyse the following logical detail deployment diagram (note: the functional components are not shown) for the **BanJia** system and identify three undesirable architectural decisions. For each issue, propose the corrective action. Justify your answer.
Hint: If necessary, you may draw a similar diagram to illustrate your corrective actions.



(6 Marks)

b. Based on your corrected detail logical deployment diagram (without showing the outline functional components) for the BanJia system in Q3(a):

i. Craft a **detail physical deployment diagram** to illustrate Cloud deployment using some of the following AWS services. Explain at least **two significant architectural decisions** that lead to your physical architecture. State your assumptions, if any. Explain using a sample scenario how the physical architecture would dynamically scale up and down in response to changes in platform loads.

The following are some AWS services/products that you could consider.

1. Amazon API gateway: A fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
2. Amazon ElastiCache: fully managed Redis and Memcached. Seamlessly deploy, run, and scale popular open source compatible in-memory data stores.

3. Amazon EMR Hadoop: A cloud-native big data platform, allowing teams to process vast amounts of data quickly, and cost-effectively at scale.
4. AWS Lambda service: A serverless service that lets you run code without provisioning or managing servers. Useful to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.
5. Amazon RDS: A Relational Database Service (Amazon RDS) that makes it easy to set up, operate, and scale a relational database in the cloud.
6. Amazon Cognito: A service that lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0.
7. AWS Elastic Beanstalk: An easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.
8. Amazon ElastiCache Redis: Stands for Remote Dictionary Server, is a fast, open-source, in-memory key-value data store for use as a database, cache, message broker, and queue. Redis is a popular choice for caching, session management, gaming, leaderboards, real-time analytics, geospatial, ride-hailing, chat/messaging, media streaming, and pub/sub apps.
9. Amazon DynamoDB: A key-value and document database that delivers single-digit millisecond performance at any scale.
10. Amazon S3: An object storage service that offers industry-leading scalability, data availability, security, and performance.
11. Amazon CloudFront: A fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment.
12. Amazon Simple Notification Service: A highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications. Using Amazon SNS topics, your publisher systems can fan out messages to a large number of subscriber endpoints for parallel processing, including Amazon SQS queues, AWS Lambda functions, and HTTP/S webhooks. Additionally, SNS can be used to fan out notifications to end users using mobile push, SMS, and email.

ii. If the platform is to be migrated to another Cloud provider that has no equivalent services/products for Amazon Lambda service, suggest a migration strategy that require only minimal development effort.

(8 Marks)