Question1

1. -

* **Availability** - The travel application can recover and is available 24 hours per day which is accessible for anyone regardless the location, technology used without downtime. For example, users can do the task throughout the week at any time during the day.
* **Functional** – The application will perform the functions that it is required to. For example, PlacesAllDay application shall be able to provide travel guidance to its customer
* **Reliable** – The application will not be prone to hardware or software failure and it will deliver the functionality whenever the users want to execute it. For example, PlacesAllDay application’s failure rate shall be less than 0.5% per year
* **Secure** – The application is protected against errors, attacks, and loss of valuable data. For example, PlacesAllDay application shall protect customer personal information during travel visa application services

1. -

Validity -

Consistency -

Completeness -

Realism check -

Veriability -

* The appropriate System organization model will be the **client-server model** for PlacesAllDay mobile application.
* The client server model’s repository is provided as a system server that holds shared data such as travel guidance details, travel visa application information, etc.
* It can provide specific services such as travel guidance, travel consultation and travel visa application which call on these services to access servers through the network.
* The processing of heavy transactions from online users of the PlacesAllDay mobile application can be distributed across a range of processors/servers to avoid overload situations.

Question2

1. The most suitable software process model will be prototyping. Customers always find it difficult to express their real requirements as they don’t have a technology background. Hence, prototyping can be developed to give the concrete impression of the system functionality.

**Justification:**

* + User is unclear about the requirements. Prototypes can be quickly designed and provided to the customer to elicit the requirements.
  + User is unclear about the interaction flows. Interaction flow can be identified through the use of a prototype.
  + User does not have any background or knowledge in dealing with electronic systems. Through the prototype, it can also provide preliminary training to the users from time-to-time.

1. -

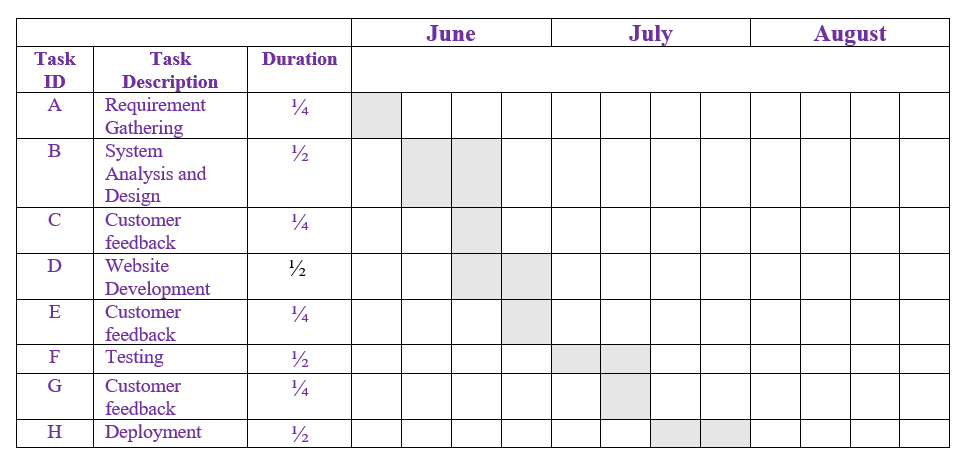
Functional requirements

* The system shall allow the user to search the property based on the criteria specified (i.e price range, property type, location, city, etc)
* The system shall allow user to view the property and make a purchase for the property

Non-functional requirements

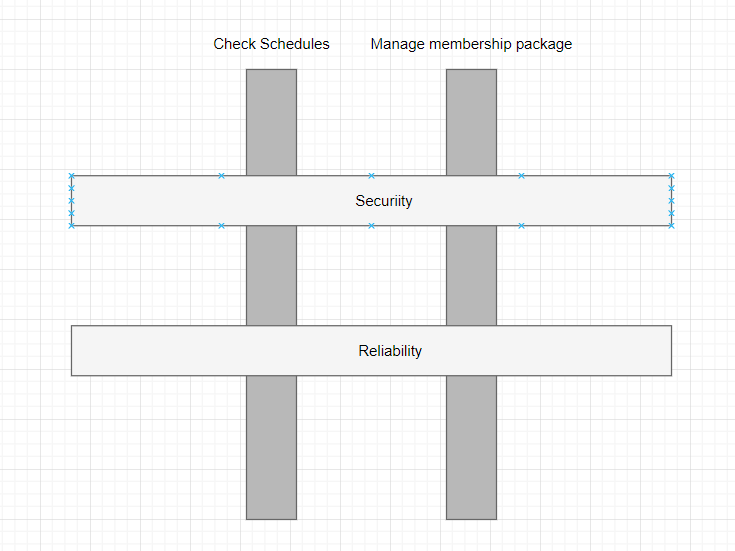
* **Reliability** - The new system’s downtime should not be more than 0.05%.
* **Usability** - The new system should be easy to navigate from page to page and have a very low learning time.

1. -



Question 3

1. -



1. **Functional cohesion.** To illustrate, cohesion focuses on how much the functionality is related to each other within a module. In this case, functional cohesion which has high cohesion is suitable for this system because each module is designed for a specific task. This is due to the fact that low/weak cohesion means that the modules are not logically connected which will make the module complex and reduce system maintainability which will result in higher maintenance cost.
2. -

Stimuli (User clicks on view schedule button) -> Response (display schedule information)

Stimuli (User disconnected from network) -> Response (auto reconnection and prepare log)

Stimuli (User click search button after input keyword (i.e booking id) into search box) -> Response (show search result(s) for booking record)

1. -

Component qualification, component adaptation, component composition and component update

Question 4

1. -
2. -
3. -

| **Characteristic of the legacy system** | **Maintenance cost** | **Comments** |
| --- | --- | --- |
| Developed in 2000 | High | The older age a system, the higher maintenance cost |
| Developed in high level programming language | Low | High level programming language is easy to be understand |
| High dependency between modules | High | When a module is changed, other modules which are dependent on it will be affected |
| Tested for 6 months before deployment | Low | System is tested for long time means it is thoroughly tested with appropriate testing strategies and techniques |
| All version are stored in a configuration management system | Low | All versions are trackable easily through the configuration management system |

Extra notes:

**Low dependency between modules**

* The module is independent or almost independent with other module thus change in a module will have least affect performance to the another module

**Low level programming language**

* Low-level programming language are machine dependent, not portable and hard to debug which cause the system to be hard to maintain

**Don’t have testing before deployment**

* The system might have many errors and bugs which are not identified and fixed in the early phases thus causing the low quality and performance of the system and resulting in higher maintenance cost in order to solve the issue after release for operational use.

FEB 2019

**Online cinema ticket system**

Stimuli (User click on view movie schedule button) -> Response (display movie schedule)

Stimuli (User click on book movie button) -> Response (link to check out page)

Stimuli (Successfully done payment) -> Response (generate online receipt)