XIAMEN UNIVERSITY MALAYSIA

Course Code	:	SOF107
Course Name	:	Introduction to Software Engineering
Lecturer	:	Subashini Raghavan
Academic Session	:	2021/04
Assessment Title	:	Requirement Engineering, Software Design & Software Testing
Submission Due Date	:	09/07/2021

Prepared by :

Student ID	Student Name
SWE2009510	Liu Aofan
SWE2009504	Huang Siqi
SWE2009513	Su Yanyu
SWE2009512	Liu Yutong
SWE2009497	Gao Yimin
SWE1909483	Lin Zhiwei

Date Received : 09/07/2021

Feedback from Lecturer:

Mark:

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature:

Date:
09/07/2021

Table of Contents

Grab——Super App in Southeast Asia

1 Summary and Purpose

At one time, due to unreasonable highway design, backward public transportation such as the subway was not yet popular, traffic and travel problems in Southeast Asia have been plagued for a long time, which led to taxis becoming a pinup way for people to travel.

Grab was originally a platform to provide the concept of "taking a taxi through the network", and its core service is to provide a taxi dispatching service with a GPS enhancement function so that passengers can take taxis easier. But now, Grab's application also provides daily services including ordering food, delivering food, buying insurance, and digital payments(Abdualla et al., 2016).

Grab was born from an idea that Chen Bingyao (Anthony Tan) came up with when he was studying at Harvard: to develop an application that can make users find taxis conveniently. After verifying the feasibility and writing a business plan together with co-founder Tan Huling, this project landed in Malaysia in June 2012(Dailey et al., 2019). Grab focuses on the cultivation of excellent organizational culture and service centered on user experience and has made great progress. The name of the project was MyTeksi at the beginning, and it was later renamed Grab.

Compared with competitors in the same business field, Grab can stand out from competitors because it has more obvious advantages. In Southeast Asia, taxi passengers often suffer from safety problems, so Grab allows users to share the journey with friends and track the taxis they take in real-time, which makes them feel safe(Fletcher et al., 2020).

The most exciting thing is that Grab adopts a no-bargain pricing method. When the user's order takes effect, the ride price will be fixed, and the fare will not be increased even in unexpected situations such as traffic congestion. This encourages drivers to deliver passengers to their destinations as soon as possible and also saves passengers time and cost. In some Southeast

Asian cities with traffic congestion, this policy perfectly meets the needs of users. In addition, Grab also provides cash payment services for users and drivers in response to the lack of local credit cards and the popularity of online payment, which is also popular among users.

2 System Requirement

Now we would like to give a general description of the Grab system requirement based on the literature survey method. The purpose of the system requirement is to provide developers enough details for developing Grab. Contrast to user requirement written for customers which are often in natural language and do not contain technical details, system requirement is written for the developer and contains functional requirement and non-functional requirement. It is clear and more rigorously specified than the user requirement.

2.1 Grab analysis

Grab's goal is to use the increasing number of cars to achieve more efficient taxi services. Therefore, to integrate taxi resources and smart terminals to solve users' travel problems, the taxi-booking function is essential for Grab(Gross & Yu, 2001).

In addition to the taxi-hailing platform, we also need a lot of additional features to improve the user experience. Many additional functions related to taxi-hailing software shall be added to Grab including GPS, maps, and information service.

From the initial provision of free and convenient ride-hailing services to passengers and drivers to accumulate users, to the final realization of profit through software value-added services, third-party payment platforms, local information services, and entry value(Jha & Mahmoud, 2019).

At the same time, Grab will provide the following third-party services:

- advertising services
- broadcasting services
- payment services
- map services
- traffic management services.

All the functioned mention above works in consistency to consist the Grab.

2.2 Function requirement

Grab, this amazing mobile application has successfully spawned "leapfrogging", that is, the rapid popularization of cutting-edge services and technologies in emerging economies and made people's lives more convenient and efficient(Jindal, Malhotra, Jain & Bansal, 2021). As the super app in Southeast Asia, there are various types of functional requirements in Grab ranging from buying daily things to booking a ride. Here, we make a mind map (Figure 2.2 - 1) to better illustrate the functional requirement in Grab.

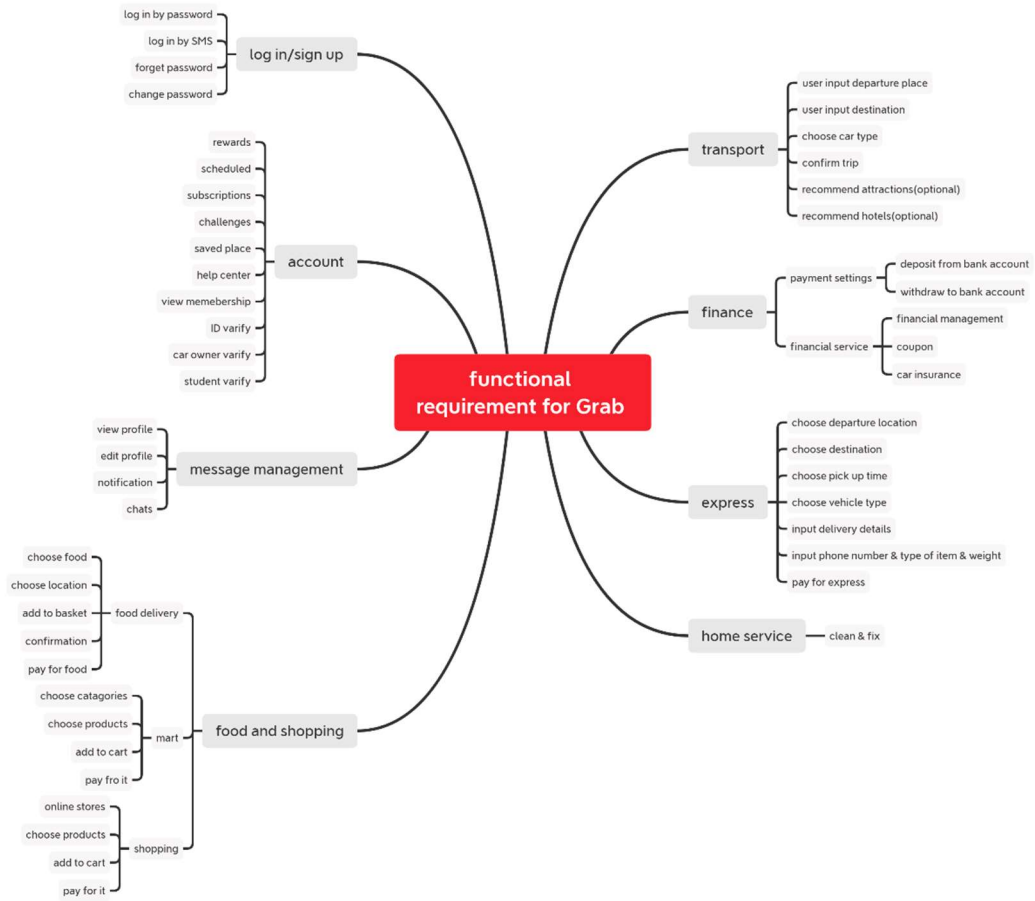


Figure 2.2 – 1 functional requirement for Grab

All these functionals mentioned above are functional requirements that refer to services that the software is supposed to provide to the user. These functions in Grab can be roughly divided into 8 categories: login/up, account, message management, food and shopping, transport, finance, Express, Home Service.

To be brief, in this essay, we will take transportation function as an example and try to translate it into a system requirement:

1.1 Users should be able to input departure place, destination, car type, departure time in Grab

1.2 The system is supposed to recommend the near attractions and hotels to the user

1.3 Each user shall be able to pay for transportation using the amount in their wallet.

1.4 The system should be able to check and deduct money in the user's Grab wallet with permission. If the money is not enough, the app will warn the user and ask him/her to top up the amount (Otherwise, the user can pay directly to the driver by cash).

1.5 After a customer has successfully placed an order, the order can be accessed by multiple drivers in the vicinity. The fastest driver who chooses to accept the order will get it.

1.6 The driver should be able to check the user's location and phone number. Both the user and the driver can contact one another.

2.3 Non-functional requirement

A non-functional requirement is a description of the properties and constraints that are stated to restrict the software. In the meantime, owing to the characteristics of non-functional requirements they may be hard to measure(Khan, Jan, Tahir, Khan & Ullah, 2016) .

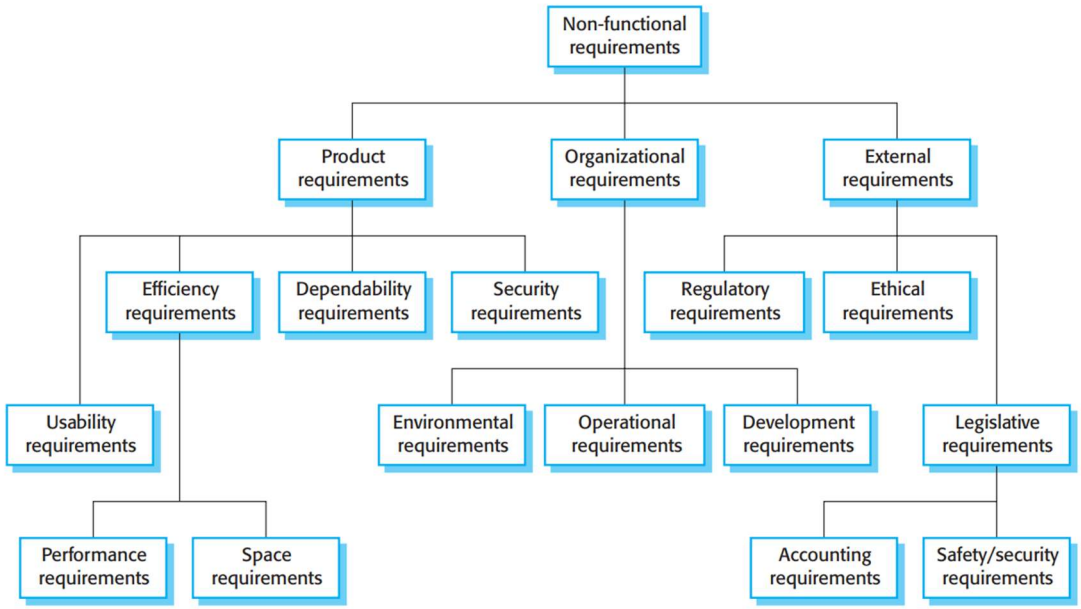


Figure 2.3 - 1 Types of non-functional requirement

Generally speaking, there are mainly three types of non-functional requirements that are most important: product requirement, organizational requirement, and external requirement.

Product requirement refers to the constraint the Grab should have during runtime. It usually provides restrictions on speed or reliability(Malyshev et al., 2015). We think Grab is supposed to satisfy the following product requirement:

- All encryption should use Advanced Encryption Standard (AES encryption),
- Systematic concurrency should up to 10 million users.
- One of the maintenance metrics means time to failure (MTTF) should no less than 5 million hours. And the server shall return the page user-requested within 1 second if the server is operated well.

Next, for organization requirements, they specify how the system will progress into the future. For example, organization requirements may define how the system will be used and the development environment. For the Grab, it may have the following requirements("Non-Functional Requirements Elicitation", 2019).

- The whole system should be developed using the Java programming language.

- Confidential data download time within normal working hours shall no more than 5 times.

Last but not least it our external requirements, it refers to all the requirements given by the external system such as laws or ethics, these are the requirement we need to satisfy during and after the development process.

- All vehicles used on Grab should meet national regulatory requirements.
- All data transmitted in Grab should protect the privacy of users and must not be misused.

3 Important Criteria in Grab

As such a successful software, there is no doubt that Grab has many good Criteria, our team tries to summarize these criteria that Grab should consider in software design.

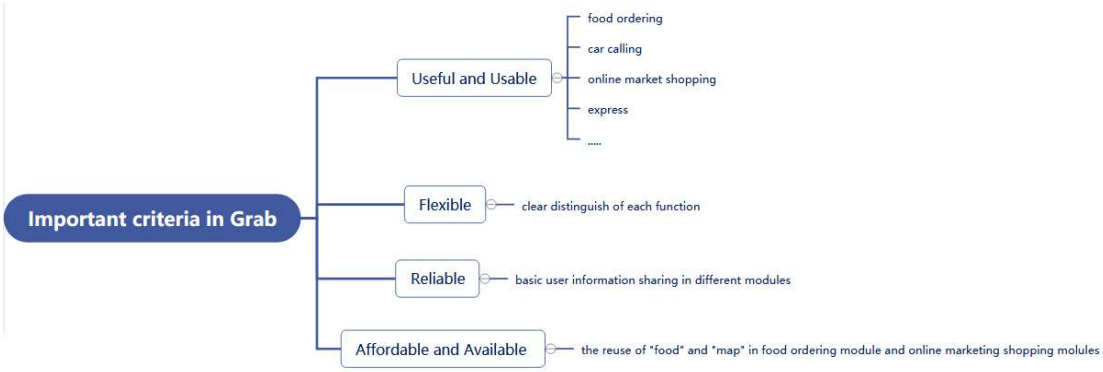


Figure 3 - 1 Important criterion in Grab

3.1 Useful and usable

As mention in the previous part, Grab is supposed to have several functions, including ordering foods, calling cars, online market shopping, express, and so on. Without Grab, users need to download more apps to reach their needs. So, the combination of various functions makes Grab more useful and usable.

Moreover, software that is useful and usable also has the following advantages:

1. Meets the needs of different types of users, to ensure that the functions of APP software meet

the needs of users and realize the value.

2. Practical. It improves the users' experience, is a real mobile phone software designed from the user's point of view, let the user feel the practicality. At the same time, the functional modules of APP are flexible and interesting. Users get more value than expected and attract and retain customers(Ransolin, Saurin & Formoso, 2020).

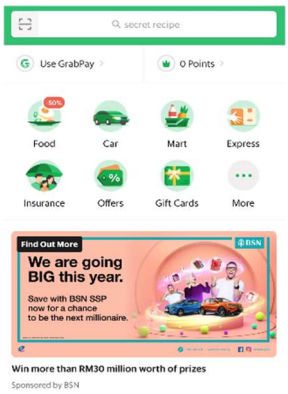


Figure 3.1 - 1 Part the functions in Grab

3.2 Reliable

When the scale of a software system becomes larger and more complex, its reliability becomes more and more difficult to guarantee. The reliability of a software system is also directly related to the reputation of the design and the competitiveness of survival and development(Shahjalal, Hasan, Chowdhury & Jang, 2019).

Software reliability means the ability of the software to avoid possible failures during a test run, and the ability to troubleshoot and troubleshoot failures once they occur. Software reliability must be determined in the design phase, and it is difficult to consider in the production and testing phase.

Grab uses data coupling which greatly increases its reliability. Important data items and arguments are passed between the calling module and the called module. Between different modules, the basic data like address, phone number, and so on can share. But the functions of different modules are different. Therefore, once found failures, they can fix at once and the failure won't affect other parts' design(Stachtiari, Mavridou, Katsaros, Bliudze & Sifakis, 2018).

3.3 Flexible

Flexible means low coupling and high cohesion. The degree of coupling between modules refers to the dependence relationship between modules, including control relationship, calling relationship, data transfer relationship. The more connections it has between modules, the stronger coupling and the worse independence it will have.

Decreasing the coupling degree between modules can reduce the influence between modules, prevent the water wave effect caused by the modification of a module, and ensure the smooth design of the system.

High cohesion means the relationship between the classes. High means the relationship between the classes should be simple, not strong, otherwise, it will run wrong. The operation of one class will affect other classes. Grab uses data coupling and logical cohesion.

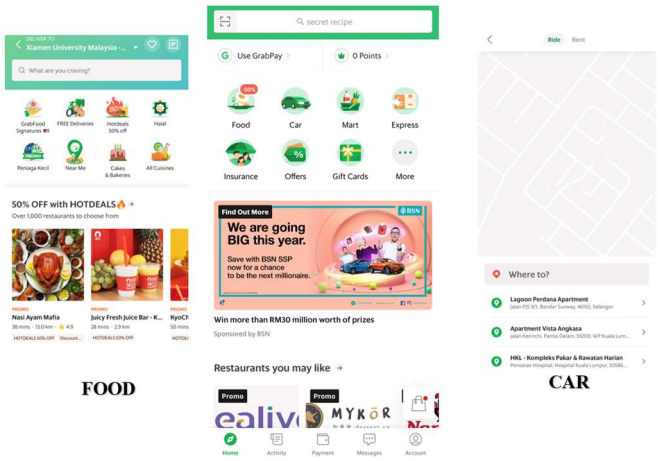


Figure 3.3 – 1 Screenshot of Grab

We can see in the screenshots that Grab has a clear distinction of each function. Click FOOD is only able to order food while click CAR is only able to rent or ride a car. So, it has low coupling and high cohesion. For users, it is easy to satisfy their needs by just clicking the concerning buttons. For programmers, they can fix bugs in certain parts while won't influence the usage of other parts in the application(Tisha & Shibly, 2021).

3.4 Affordable and Available

Software reuse refers to the process of reusing the same or similar software elements in two or more different software development processes. Software elements include program code, test cases, design documents, design processes, requirements analysis documents, and even domain knowledge.

In general, reusable elements are also called software components, and the larger the reusable component, the greater the granularity of reuse. Software reuse can make the design more affordable.

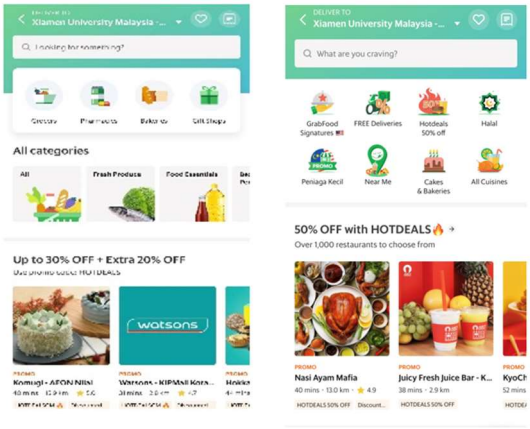


Figure 3.4 – 1 Screenshot of Grab

In GRAB, the “FOOD” and “MART” parts can reuse the software. These two parts are quite similar, both of them are ordering food and deliver to your certain address. Though one is ordering from the restaurant and another is from markets, their main programming is similar. So here software reuse can raise Grab’s affordability.

Through reuse, the development time of the application will decrease, which improve the availability of Grab.

4 Software Testing

Comprising verification and validation, software testing is the evaluation of the software

against requirements gathered from user and system specifications that are conducted at phase level in software development life cycle or module level in program code. The mentioned application, Grab, also needs software testing to help us find whether or not the application is doing what it is supposed to do and find defects of the program(Toth & Vidacs, 2019).

Typically, a commercial software system has to go through three stages of testing, development testing, release testing, and user testing. Now, we are trying to illustrate some testing that is bound to be taken in Grab.

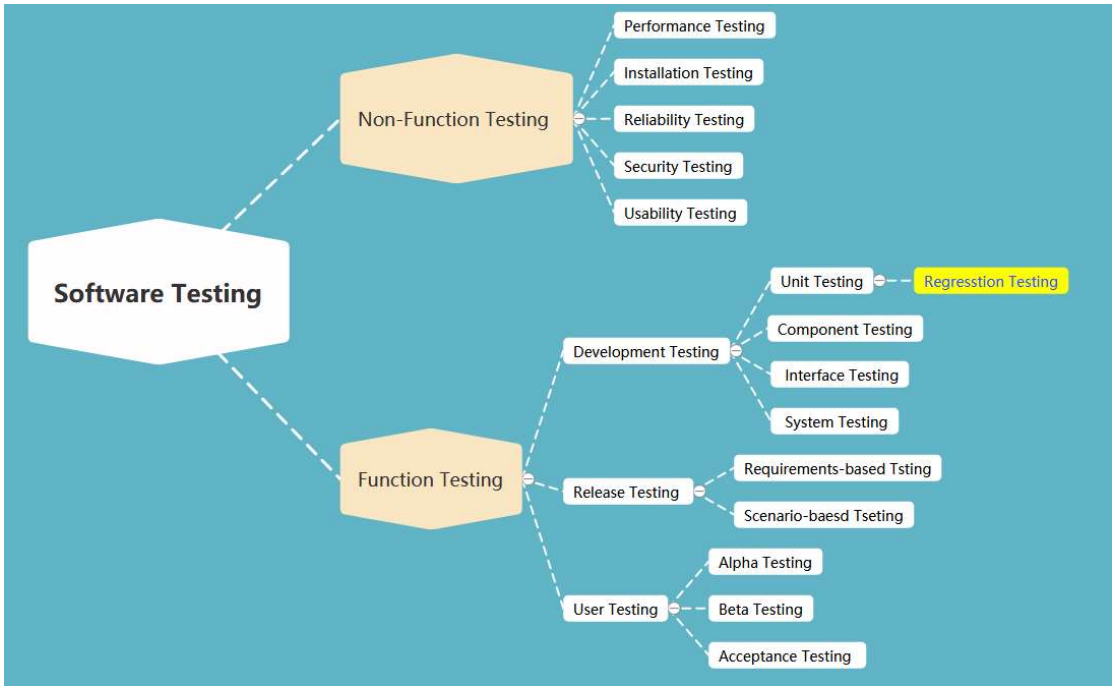


Figure 4 – 1 Software Testing in Grab

4.1 Development Testing

It includes testing activities that are likely to happen during the developing period. System designers and programmers are likely to be involved in this testing phase.

Unit Testing: As is known to us all that Grab is a giant app that has a large number of independent functions which is just the characteristic of unit testing.

Component testing: A component test consists of a documented set of instructions on how to carry out a particular set of tests on a given software component and the expected results from

the test.

System testing: It refers to test the interaction between components. It tests Grab's emergent behavior.

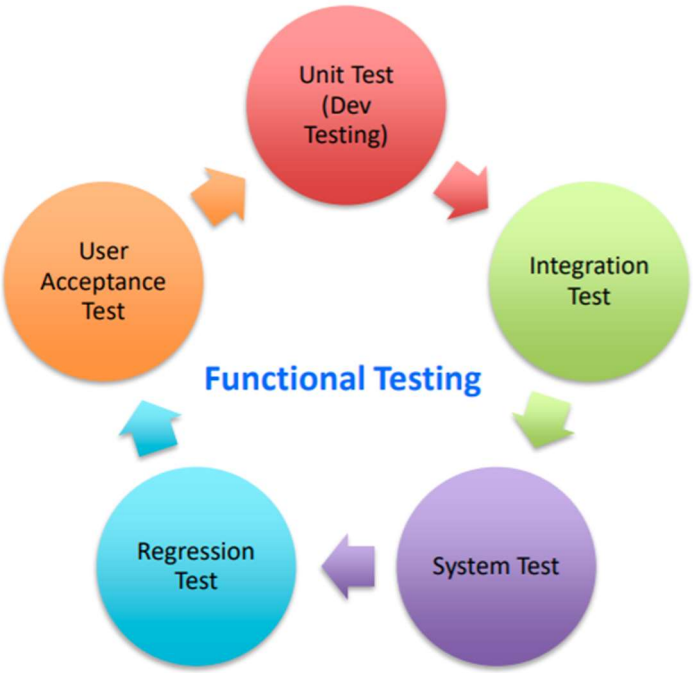
Consisting of so many individual parts, there is no denying that the unit testing is supposed to be taken during the development phase. Component constitutes units and system constitutes components. So, these three kinds of testing are closely related and should be conducted for this project(Zhou, Zhi, Morisaki & Yamamoto, 2020).

4.2 Release Testing

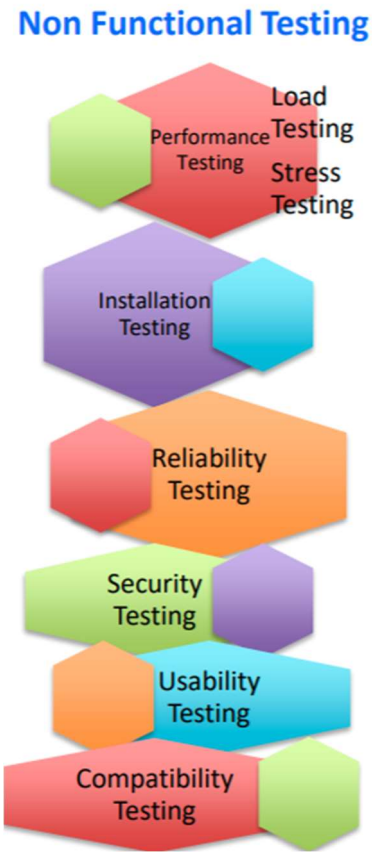
The software in this stage is tested to discover bugs and defects in a particular release of the system. It is usually carried out by a separate team. Here we will try to talk about the testing in the release stage that should be carried out.

Requirement-based testing: The engineers in the separate testing teams check whether Grab satisfies the functional requirements and non-functional requirements stated by the customer. Here is the common requirement-based testing that we think Grab should take.

- **Functional testing**
  - **Unit testing**
  - **integration testing**
  - **system testing**
  - **regression testing**
  - **user acceptance testing**



- **Non-functional testing**
  - **Installation testing**
  - **Performance testing**
  - **Reliability testing**
  - **Security testing**
  - **Usability testing**



Scenario testing: It is of great importance to Grab the image of the typical scenarios of the users which appears in the form of a narrative story.

George is a nurse who specializes in mental health care. One of his responsibilities is to visit patients at home to check that their treatment is effective and that they are not suffering from medication side effects.

On a day for home visits, George logs into the Mentcare system and uses it to print his schedule of home visits for that day, along with summary information about the patients to be visited. He requests that the records for these patients be downloaded to his laptop. He is prompted for his key phrase to encrypt the records on the laptop.

One of the patients whom he visits is Jim, who is being treated with medication for depression. Jim feels that the medication is helping him but believes that it has the side effect of keeping him awake at night. George looks up Jim's record and is prompted for his key phrase to decrypt the record. He checks the drug prescribed and queries its side effects. Sleeplessness is a known side effect, so he notes the problem in Jim's record and suggests that he visit the clinic to have his medication changed. Jim agrees, so George enters a prompt to call him when he gets back to the clinic to make an appointment with a physician. George ends the consultation, and the system re-encrypts Jim's record.

After finishing his consultations, George returns to the clinic and uploads the records of patients visited to the database. The system generates a call list for George of those patients whom he has to contact for follow-up information and make clinic appointments.

A sample of scenario-based testing

4.3 User Testing

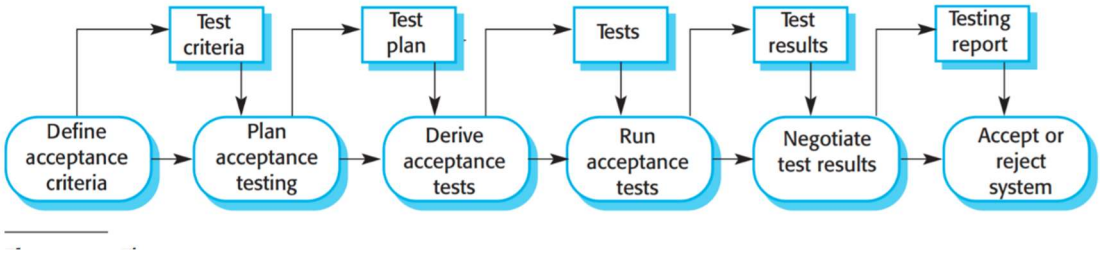
Now, we come to a stage where our users or customers are supposed to provide input and advice on system testing. For a popular app like Grab, user testing is the soul of their success. Generally,

there are three types of user testing we guess the Grab will carry out.

Alpha testing: User and developers are working together to improves the system. It covers the defect that user and developers are not closely related which reduce the chance of unanticipated changes.

Beta testing: Beta testing choose a selected group of customer who is interested in the new version which can be used to discover interaction problems between software and features.

Acceptance testing: As an inherent part of the system development, in acceptance testing, users will use their data to test the system. It usually works in the following flow.



References

Abdualla, H., Ceylan, H., Kim, S., Gopalakrishnan, K., Taylor, P., & Turkan, Y. (2016). System Requirements for Electrically Conductive Concrete Heated Pavements. *Transportation Research Record: Journal Of The Transportation Research Board*, *2569*(1), 70-79. doi: 10.3141/2569-08

Dailey, P., Osborn, J., Ashley, E., Baron, E., Dance, D., & Fusco, D. et al. (2019). Defining System Requirements for Simplified Blood Culture to Enable Widespread Use in Resource-Limited Settings. *Diagnostics*, *9*(1), 10. doi: 10.3390/diagnostics9010010

Fletcher, S., Johnson, T., Adlon, T., Larreina, J., Casla, P., & Parigot, L. et al. (2020). Adaptive automation assembly: Identifying system requirements for technical efficiency and worker satisfaction. *Computers & Industrial Engineering*, *139*, 105772. doi: 10.1016/j.cie.2019.03.036

Gross, D., & Yu, E. (2001). From Non-Functional Requirements to Design through Patterns. *Requirements Engineering*, *6*(1), 18-36. doi: 10.1007/s007660170013

Jha, N., & Mahmoud, A. (2019). Mining non-functional requirements from App store reviews. *Empirical Software Engineering*, *24*(6), 3659-3695. doi: 10.1007/s10664-019-09716-7

Jindal, R., Malhotra, R., Jain, A., & Bansal, A. (2021). Mining Non-Functional Requirements using Machine Learning Techniques. *E-Informatica Software Engineering Journal*, *15*(1). doi: 10.37190/e-inf210105

Khan, F., Jan, S., Tahir, M., Khan, S., & Ullah, F. (2016). Survey: Dealing Non-Functional Requirements at Architecture Level. *VFAST Transactions On Software Engineering*, *9*(2), 7. doi: 10.21015/vtse.v9i2.410

Malyshev, Gorodetsky, Karsaev, Samoilov, Tikhomirov, & Mankov. (2015). Multiagent system for airspace deconfliction. *SPIIRAS Proceedings*, *1*(3), 88. doi: 10.15622/sp.3.5

Non-Functional Requirements Elicitation. (2019). *International Journal Of Recent Technology And Engineering*, *8*(2S11), 2129-2132. doi: 10.35940/ijrte.b1218.0982s1119

Ransolin, N., Saurin, T., & Formoso, C. (2020). Integrated modelling of built environment and functional requirements: Implications for resilience. *Applied Ergonomics*, *88*, 103154. doi: 10.1016/j.apergo.2020.103154

Shahjalal, M., Hasan, M., Chowdhury, M., & Jang, Y. (2019). Smartphone Camera-Based Optical Wireless Communication System: Requirements and Implementation Challenges. *Electronics*, *8*(8), 913. doi: 10.3390/electronics8080913

Stachtiari, E., Mavridou, A., Katsaros, P., Bliudze, S., & Sifakis, J. (2018). Early validation of system requirements and design through correctness-by-construction. *Journal Of Systems And Software*, *145*, 52-78. doi: 10.1016/j.jss.2018.07.053

Tisha, T., & Shibly, M. (2021). Non-Functional Requirements for Block-chain: Challenges and New Directions. *IOP Conference Series: Materials Science And Engineering*, *1110*(1), 012016. doi: 10.1088/1757-899x/1110/1/012016

Toth, L., & Vidacs, L. (2019). Comparative Study of The Performance of Various Classifiers in Labeling Non-Functional Requirements. *Information Technology And Control*, *48*(3), 432-445. doi: 10.5755/j01.itc.48.3.21973

Zhou, Z., Zhi, Q., Morisaki, S., & Yamamoto, S. (2020). An Evaluation of Quantitative Non-Functional Requirements Assurance Using ArchiMate. *IEEE Access*, *8*, 72395-72410. doi: 10.1109/access.2020.2987964

Contributions

Name and Student ID	Contributions
SWE2009510 Liu Aofan	Image& table making and write system requirement part
SWE2009504 Huang Siqi	Slides making and assistant in checking assignment
SWE2009513 Su Yanyu	Write summary &purpose
SWE2009512 Liu Yutong	Write Software testing part
SWE2009497 Gao Yimin	Write important criteria in Grab
SWE1909483 Lin Zhiwei	Assist in making image used in the assignment

Marking Rubrics

Component Title	**Report**				**Percentage (%)**		
Criteria	**Score and Descriptors**					**Weight (%)**	**Marks**
	Excellent (5)	**Good (4)**	**Average (3)**	**Need Improvement (2)**	**Poor (1)**		
Summary and Purpose	Excellent summary and purpose of the selected application.	Good summary and purpose of the selected application.	Average summary and purpose of the selected application..	summary and purpose of the selected application is not complete and correct.	Almost no summary and purpose of the selected application is written.	**2**	
System Requirements (Functional requirement and Non Functional requirement)	Concise, significant and excellent set of system requirements are determined.	Good set of system requirements are determined.	Almost complete set system requirements are determined.	Insufficient set system requirements are determined.	No set of system requirements are determined..	**3**	
Software Design Criteria	Concise and most relevant software design criteria are determined.	Good set of software design criteria are determined.	Almost complete set of software design criteria are determined.	Insufficient or some missing set of software design criteria are determined.	Almost none or incorrect software design criteria are determined,	**3**	
Software Testing	Complete list of functional and non functional testing are covered. The testing types are very well explained.	List of functional and non functional testing are covered. The testing types are well explained.	Almost a complete list of functional and non functional testing are covered. The testing types are explained.	The list of functional and non functional testing are not well covered. The testing types are somewhat explained.	The list of functional and non functional testing are not covered. The testing types are not explained.	**3**	
					TOTAL		

Note to students: Print out and attach this appendix together with the submission of coursework.

Component Title	**Presentation**		**Percentage (%)**		
Criteria	**Score and Descriptors**			**Weight (%)**	**Marks**
	5-4	**3-2**	**1**		
Delivery	Smooth delivery that holds audience attention.	Fairly smooth delivery that holds audience attention most of the time.	Delivery is not smooth, and audience attention is often lost.	**3**	
Overall presentation	Excellent presentation from all the group members.	Good presentation from all the group members	Poor presentation from all the group members	**4**	
Response to Audience Questions	Answers audience questions clearly and completely	Answers some audience questions, but not clearly and completely	Does not answer audience questions	**1**	
				TOTAL	

Note to students: Print out and attach this appendix together with the submission of coursework.