1. Introduction

1.1 Project Overview

Peer tutoring mobile app is designed to help match a student who wants to get help on a certain subject and a tutor who are willing to help others on a subject. The tutor may choose to get paid or work as a volunteer. So far it supports Android platform. For backend, we use LAMP in this project, which L stands for Linux, A stands for Apache, M stands for MySQL, and P stands for PHP. For frontend, we use Android Studio to develop and emulate.

1.2 Test Plan Overview

This Test Plan has been created to communicate the test approaches to team members and stakeholders (i.e. Prof. Fahmy, Jason and Michael). It includes the objectives, scope, approach and schedule. This document explicitly identifies how we are approaching Unit Test, Component Test, System Test. This document also clearly defines roles and responsibilities for testing activity schedule. However, specific test cases are not defined in this document.

2. Objective

For verification testing, the test team is responsible for demonstrate to the developer and the customer that the software meets every user requirement.
For validation testing, the goal is to discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification and fix them.

3. Test

3.1 Unit Test

- 3.1.1 Test Tool

For each unit framework, we use JUnit framework to write and run our program tests. With Junit framework, we could write our test cases to test every single function, methods and classes, focusing on their functionality and any possible defects in the object. To cover these two aspects, functionality testing and defect testing are included.

- 3.1.2 Functionality Testing

The purpose of this testing is to show that, when used as expected, every single method and object could offer corresponding desired output. To make our testing efficient and considerable, we write normal independent inputs for every methods and

objects. These test cases should be small and self-validating.

- 3.1.3 Defect Testing

The purpose of this testing is to reveal our possible defects in our objects. We use abnormal inputs to check that every object could process the abnormal inputs correctly. To expose all potential defects, we choose inputs that 1. force the system to generate all error messages 2. cause input buffers to overflow 3. series of inputs numerous times. Besides, we should enforce the output to be invalid.

3.2 Component Test

- 3.2.1 Testing Focus

The focus of component testing is to show that the component interface behaves according to its specification, assuming unit tests on the individual objects within the component have been completed. Interface errors in the composite component may not be detectable by testing the individual objects because these errors result from interactions between the objects in the component

- 3.2.2 Interface Classification

There are different types of interface between program components:
• Parameter interfaces: Data passed from one method or procedure to another.
• Shared memory interfaces: Block of memory is shared between procedures or functions
• Procedural interfaces: Sub-system encapsulates a set of procedures to be called by other sub-systems
• Message passing interfaces: Sub-systems request services from other sub-systems

According to the different nature of interfaces, possible interface errors fall into different categories: Interface Misuse, Interface misunderstanding.

- 3.2.3 Inspections and reviews

Since many of the interface errors fall into interface misuse and interface misunderstanding, Inspections and reviews of component interface can sometimes be more cost effective than testing for discovering interface errors.
Inspections concentrate on component interfaces and questions about the assumed interface behavior asked during the inspection process. Java also allows many interface errors to be trapped by the compiler.

3.3 System Test

- 3.3.1 Focus

System testing tests the integrated system. It focuses on testing the interactions between components. System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces.

System testing also tests the emergent behavior of a system.

- 3.3.2 Verification System Testing

This part focus on checking if system performs correctly using a given set of test cases that reflect the system's expected use. Since our project includes interacting between front-end and back-end, all related functions should be tested. Particularly, each use case specified by the user requirement document should be tested.

- 3.3.3 Defective System Testing

This part focus on exposing defects. Different from Verification System Testing, test cases in this part can be deliberately obscure and need not reflect how the system is normally used. Specifically, all functions must be tested with unexpected input.

✓ Spring 1 main goals:

The first goal is the registration system that will establish the users' database, which will allow user to register and log into the app.
The second one is the function of creating session which allow tutors to create session with corresponding information such as data or subject.

✓ Sprint 2 main goals:

The first goal is the volunteer service which allow tutors to decide if the service is free or they should be paid.
The second one is to allow students to search the wanted subject sessions and do signup for sessions. If nothing is met, they could post the session themselves.
The third one is to allow tutee to rate their tutors.
The fourth one is to allow users to cancel their sessions if they are unable to finish them in the future.
The fifth on is to allow users invite the other users to participant their own sessions.

✓ Sprint 3 main goals:

According to our plans, sprint 3 is all about test our previous functional and non-functional requirements.

Test Type	Test Objects	Owner	Start	Finish	Test Case
Unit Test	Login	Jiuyun Zhang, Junru Wang	3/1	3/3	1. Validation test: Test onCreate(), see whether view can appear or not. 2. Validation test: Test onClick(), see if web can receive data by clicking. 3. Defect Test: pass different responses to onRsponse() for btnLogin in MainActivity to see

					whether it will handle error correctly. 4. Defect Test: pass different parameters to LoginRequest() to test the communication between android studio and database.
	Register	Jiuyun Zhang, Junru Wang	3/1	3/3	1. pass different responses to onRsponse() for btnRegister in RegisterActivity to see whether it will handle error correctly 2. pass different parameters to RegisterRequest() to see if the user info will be saved to database. 3. Try to register with illegal password format to test the password check function
	Session Creation	Xinyu Zhou, Zu Liu, Xinyi Gong	3/1	3/3	1. Validation Test: Test Add() by calling it in test(), passing different parameters to getParams() and see if the return value is correct or not.
Component Test	Login	Jiuyun Zhang, Junru Wang	3/1	3/3	1. Run in simulator and type in valid and invalid username and password. 2. Run in several computers to login in at the same time. 3. Defect Test: Click on button several times and check database. 4. Defect Test: Type in wrong format information.
	Register	Jiuyun Zhang, Junru Wang	3/1	3/3	1. Validation Test: Run in simulator and type in several different types of usernames and passwords 2. Defect Test: Type in several same usernames and passwords to see if it can distinguish an existing account. 3. Validation Test: Run in several computers to register at the same time. 4. Defect Test: Click on button several times and check database.

					5. Defect Test: Type in wrong format information.
	Session Creation	Xinyu Zhou, Zu Liu, Xinyi Gong	3/1	3/3	1. Validation Test: Run in simulator and use different types of input. 2. Validation Test: Run in several computers to register at the same time. 3. Defect Test: Click on button several times and check database. 4. Defect Test: Type in wrong format information.
System Test	Peer Tutoring Mobile App version 1.0	Jiuyun Zhang, Junru Wang, Xinyu Zhou, Zu Liu, Xinyi Gong	3/1	3/3	1. Validation Test: Run in different models of mobiles. 2. Validation Test: Test different languages. 3. Defect Test: Test app behavior when mobile receiving texts or phone calls. 4. Validation Test: Test both horizon and vertical view. 5. Defect Test: Test app behavior when no Internet connection. 6. Defect Test: Test when mobile have insufficient memory.

Test Type	Test Objects	Owner	Start	Finish	Test Case
Unit Test	Voluntary session	Junru Wang	3/20	3/28	1. Validation Test: pass valid requests and print function response. 2. Defect Test: manually pass illegal parameters to function
	Session Search	Junru Wang,Xinyu Zhou,Xinyi Gong	3/20	3/28	1. Validation Test: Pass different responses to test if the onResponse() will handle error correctly. 2. Validation Test: Input different words to test if the onQueryTextChange() in both TutorSearch and TuteeSearch will work normally 3. Defect Test: manually pass illegal parameters to function
	Rate	Jiuyu Zhang, Liu Zu	3/20	3/28	1. Validation Test: Pass different parameters to test if the Feedback() will communicate with database correctly. 2. Defect Test: manually pass illegal parameters to function
	Session Cancel	Jiuyu Zhang, Xinyu Zhou	3/20	3/28	1. Validation Test: Pass different responses to onResponse() for btnCancel to test if it will work normally to cancel a session. 2. Defect Test: manually pass illegal parameters to function
	Appointment	Junru Wang, Xinyu Zhou, Xinyi Gong	3/20	3/28	1. Validation Test: Pass different responses to onResponse() for btnAppointment test if the Feedback() will work correctly to register for a

					session. 2. Defect Test: manually pass illegal parameters to function
	Peer Invite	Xinyu Zhou	3/20	3/28	1. Validation Test: Both tutors and tutees invite an existing user. 2. Defect Test: Try to invite users that does not exist to test if the invitation will cause an exception
Component Test	Voluntary session	Junru Wang	3/20	3/28	Validation testing. The tutor chooses to volunteer for a session.
	Session Search	Junru Wang, Xinyu Zhou, Xinyi Gong	3/20	3/28	1. Validation testing. Tutee search sessions created by tutor using valid keywords. 2. Validation testing. Tutor search sessions created by tutee using valid keywords. 3. Defect testing. Tutee search sessions created by tutor using invalid keywords. 4. Defect testing. Tutor search sessions created by tutee using valid keywords.
	Rate	Jiuyu Zhang, Liu Zu	3/20	3/28	1. Validation testing. The tutor rates on his tutee and a tutee rates on his tutor and then check the database to see if the rate has been recorded. 2. Defect testing. Several tutees rate on one tutor at the same time.
	Session Cancel	Jiuyu Zhang, Xinyu Zhou	3/20	3/28	1. Validation testing. Tutor cancel a session to see if the session will disappear in database 2. Validation testing. Tutee cancel a session to

					see if the session will disappear in tutor's to do list and appear in his not yet list. 3. Defect testing. The tutor clicks on the cancel button several times to see if the app will crash.
	Appointment	Junru Wang, Xinyu Zhou, Xinyi Gong	3/20	3/28	1. Validation testing. The tutee registers for an existing session which has not been registered. 2. Defect testing. The tutee registers for a session which has been registered. 3. Defect testing. Two or more tutees register for a session at the same time. 4. Defect testing. A tutee registers and a tutor cancels at the same time.
	Peer Invite	Xinyu Zhou	3/20	3/28	1. Validation testing. The tutor invites another tutor and a tutee invites another tutee and then check the database to see if the invite component will work correctly. 2. Validation testing. The tutor/tutee check his invitations in his profile to see if his invitations will display correctly 3. Defect testing. The tutor invites another tutor who have not registered. Also, the tutee invites another tutee who have not registered.
System Test	Peer Tutoring Mobile App version 2.0	Jiuyun Zhang, Junru Wang, Xinyu Zhou, Zu Liu,	3/20	3/28	1. Validation testing. Create two users, one as a tutor and the other as a student, then they will test session creation, search,

		Xinyi Gong			register and canceling. 2. Stress testing. All teammates keep using this app to register, login, create session and search to test the stability of the system. 3. Validation testing. Try to hack this app using common approaches such as buffer overflow, SQL injection etc., to test the safety of this system. 4. Validation Testing. Use android simulator to simulate many users using the app at the same time, to test the scalibility of this system.