技能交換平台

測試文件

專案名稱	Exchange
撰寫日期	2017/01/11
發展者	吳宇鴻、鍾子健、施博文、蔡昌廷、羅祐任

版次變更記錄

版次	變更項目	變更日期
1.0	初版	2016/12/31
1.1	新增細部內容	2017/01/15
1.2	Bug回報與完成度更新	2017/01/17

目錄

版次變更記錄 2

1 測試目的與接受準則(OBJECTIVES AND ACCEPTANCE CRITERIA) 4

1.1 系統範圍(SYSTEM SCOPE) 4

1.2 測試接受準則(TEST ACCEPTANCE CRITERIA) 4

2 測試環境(TESTING ENVIRONMENT) 5

2.1 硬體需求(HARDWARE SPECIFICATION AND CONFIGURATION) 5

2.2 軟體需求(SOFTWARE SPECIFICATION AND CONFIGURATION) 5

2.3 測試資料來源(TEST DATA SOURCES) 5

2.4 測試工具與設備(TOOLS AND EQUIPMENTS) 5

3 測試案例(TEST CASES) 6

3.1 單元測試(UNIT TEST) 6

3.2 系統測試(SYSTEM TEST) 38

4 測試工作指派與時程(PERSONNEL AND SCHEDULE) 47

4.1 測試成員(PERSONNEL) 47

4.2 測試程序與時程(TEST PROCEDURE AND SCHEDULE) 47

5 測試結果與分析(TEST RESULTS AND ANALYSIS) 48

5.1 測試結果(TEST RESULTS) 48

5.2 缺失報告(DEFECT TRACKING) 51

追溯表(TRACEABILITY MATRIX) 55

1 測試目的與接受準則(Objectives and Acceptance Criteria)

1.1 系統範圍(System Scope)

本系統以技能交換為主軸，有別於以往同類型網站條列式呈現技能的方式，本系統使用自動配對功能隨機配對彼此的技能，使用者可以不必侷限於自己腦海中的技能，透過隨機配對功能在不預設立場的情況下挑選自己有興趣的技能。我們的系統採用自行設計的配對演算法，參考使用者的興趣以及相鄰地區來鎖定隨機挑選的範圍。本系統將系統分為登入模組、首頁模組、帳戶模組、配對模組、技能模組、交流模組、評價模組、資料庫模組、聊天模組等九大模組。目前已完成各模組實作並設計對應之單元測試與系統測試之設計案例，整合部分則已完成主要功能的整合。

1.2 測試接受準則(Test Acceptance Criteria)

本測試計劃需要滿足下面的測試接受準則：

(1) 測試程序需要依照本測試計劃所訂定的程序進行，所有測試結果需要能符合預期測試結果方能接受。

(2) 以系統測試案例為單位，當系統測試未通過時，需要進行該單位的單元測試，其接受的準則如第一項中所規定的相同。

(3) 若模組中的任一單元需要測試，則該模組的其他單元應加入單元測試，其接受的準則如第一項中所規定的相同。

(4) 時間寬裕的情況下，為確保系統品質，並避免系統測試下未發現的錯誤，各模組開發應各自執行單元測試，其接受的準則如第一項中所規定的相同。

2 測試環境(Testing Environment)

2.1 硬體需求(Hardware Specification and Configuration)

項次	名　　稱	數量	規　　格	備　　註
1	伺服器	1	Msi GE602O1	client
2	電腦	1	能上網之電腦	server

2.2 軟體需求(Software Specification and Configuration)

項次	名　　稱	數量	規　　格	備　　註
1	瀏覽器	1	Chrome	client
2	資料庫軟體	1	mySQL	Server

2.3 測試資料來源(Test Data Sources)

本測試計畫的測試資料來源可分成下面的資料來源：

(1) 自行建立測試用之註冊會員基本資料

(2) 自行建立測試用之興趣技能與技能

(3) 自行建立測試用之技能交流關係

2.4 測試工具與設備(Tools and Equipments)

本測試計劃分為單元測試以及系統測試。單元測試的執行使用 Junit 進行，並事先進行分析所需的測試案例，其包含測試輸入、預測輸出、實際輸出。系統測試的執行採用使用者直接操作的測試方式，預先進行分析所需的測試案例，其包含操作流程、預期結果、實際結果。單元測試與系統測試的準備與結果皆使用 Microsoft Office Word 撰寫成測試文件。

3 測試案例(Test Cases)

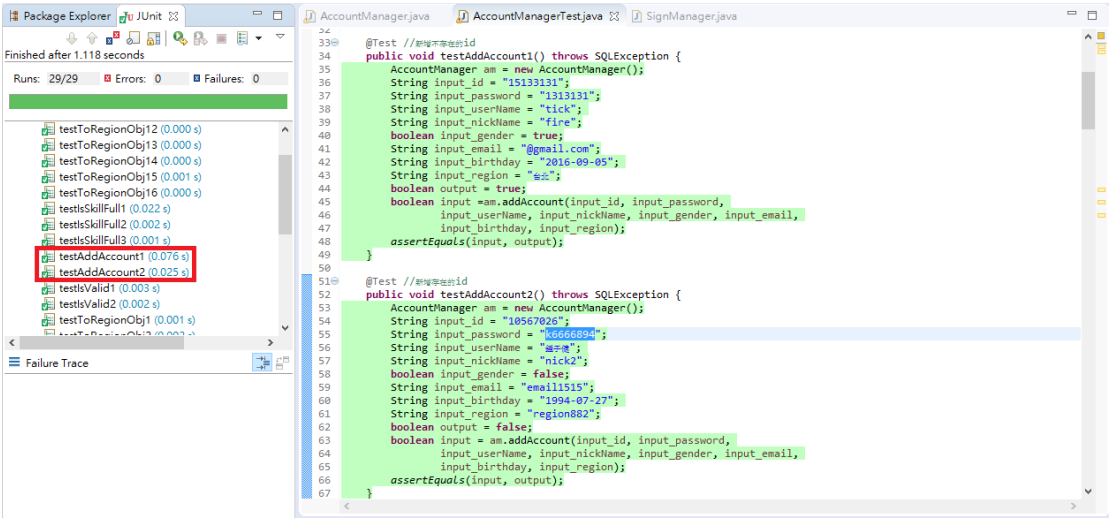
3.1 單元測試(Unit Test)

➢ 帳戶模組單元測試

• AccountManager.addAccount()

Test Case 1	
Test Case Id	JUT-001
Test Function Name	testAddAccount1()
Test proposal	嘗試建立不存在資料庫的帳號資料
Input	input_id = "15133131" input_password = "1313131” input_userName = "tick" input_nickName = "fire" input_gender = true(男性) input_email = "@gmail.com" input_birthday = "2016-09-05" input_region = "台北"
Expected Output	建立成功，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-002
Test Function Name	testAddAccount2()
Test proposal	嘗試建立已存在資料庫的帳號資料
Input	input_id = "10567026" input_password = "k6666894" input_userName = "Serena" input_nickName = "nick2" input_gender = false(女性) input_email = "email1515" input_birthday = "1994-07-27"; input_region = "region882"
Expected Output	建立失敗，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass



addAccount() Code

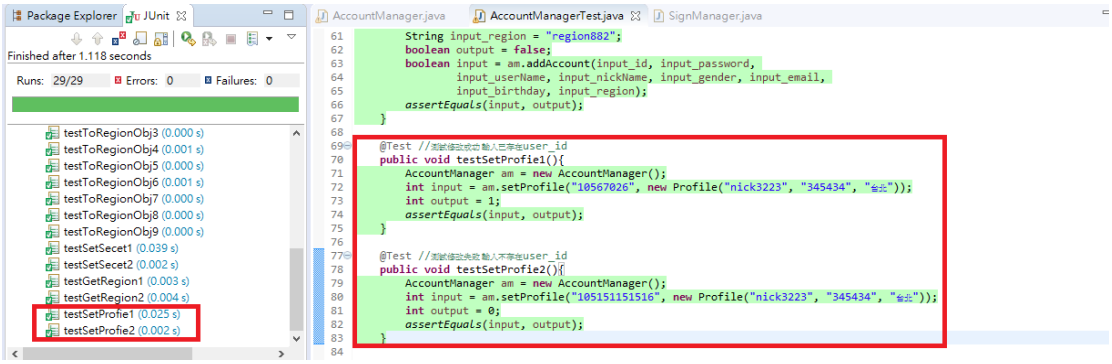
```
public boolean addAccount(String id, String password, String userName, String nickName, boolean gender,
        String email, String birthday, String region) throws SQLException {
    int result = 0;
    Date recentLog = new Date();
    java.sql.Date sqlStartDate = new java.sql.Date(recentLog.getTime());
    int skillMax = 3;
    int skillNumber = 0;
    int gender_int = 0;
    if (gender) {
        gender_int = 1;
    } else {
        gender_int = 0;
    }
    String query = "INSERT INTO accounts VALUES ('" + id + "', '" + password + "', '" + userName + "', '" + nickName
            + "', '" + gender_int + "', " + "'" + email + "', '" + birthday + "' ,'" + region + "', '" + skillNumber
            + "', '" + skillMax + "', '" + sqlStartDate + "')";
    result = DataBaseAdmin.updateDB(query);
    return (result == 0) ? false : true;
}
```

- AccountManager.setProfie()

Test Case 1	
Test Case Id	JUT-003
Test Function Name	testSetProfie1()
Test proposal	嘗試修改已存在資料庫帳號的個人資料
Input	input_id = "10567026" input_nickName = "nick3223" input_email = "345434" input_region = "台北"
Expected Output	修改成功，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-004
Test Function Name	testSetProfie2()
Test proposal	嘗試修改不存在資料庫帳號的個人資料
Input	input_id = "105151151516"

	input_nickName = "nick3223" input_email = "345434" input_region = "台北"
Expected Output	修改失敗，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass



setProfie() Code

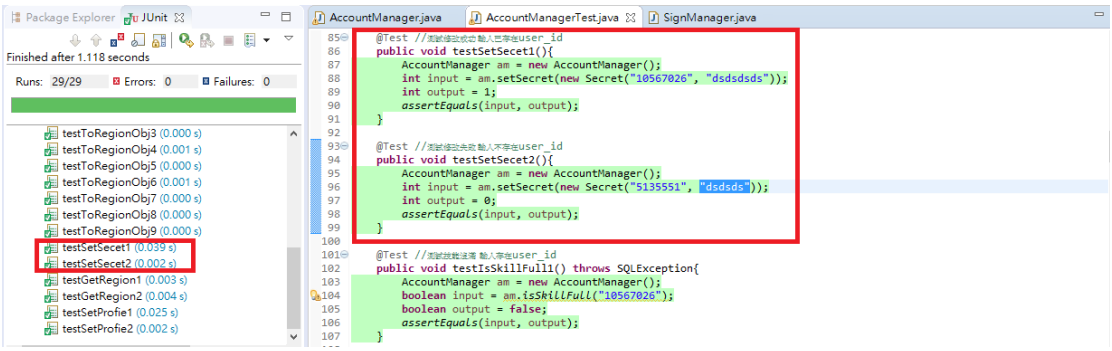
```
// Profile修改
public int setProfile(String id, Profile profile) {
    int result = 0;
    String nickName = profile.getNickName();
    String email = profile.getEmail();
    String region = profile.getRegion();
    try {
        String query = "UPDATE accounts SET nick_name = '" + nickName + "', email = '" + email + "', region = '"
                + region + "' " + "where user_id = '" + id + "'";
        result = DataBaseAdmin.updateDB(query);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;

}
```

- AccountManager.setSecret()

Test Case 1	
Test Case Id	JUT-005
Test Function Name	testSetSecret1()
Test proposal	嘗試修改已存在資料庫帳號的密碼
Input	input_id = "10567026" input_password = “dsdsdsds”
Expected Output	修改成功，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-006
Test Function Name	testSetSecret2()
Test proposal	嘗試修改不存在資料庫帳號的密碼
Input	input_id = "5135551"

	input_password = "dsdsds"
Expected Output	修改失敗，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass



setSecret() Code

```
// Secret修改
public int setSecret(Secret secret) {
    int result = 0;
    String id = secret.getId();
    String password = secret.getPassword();
    try {
        String query = "UPDATE accounts SET password = '" + password + "' " + "where user_id = '" + id + "'";
        result = DataBaseAdmin.updateDB(query);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;

}
```

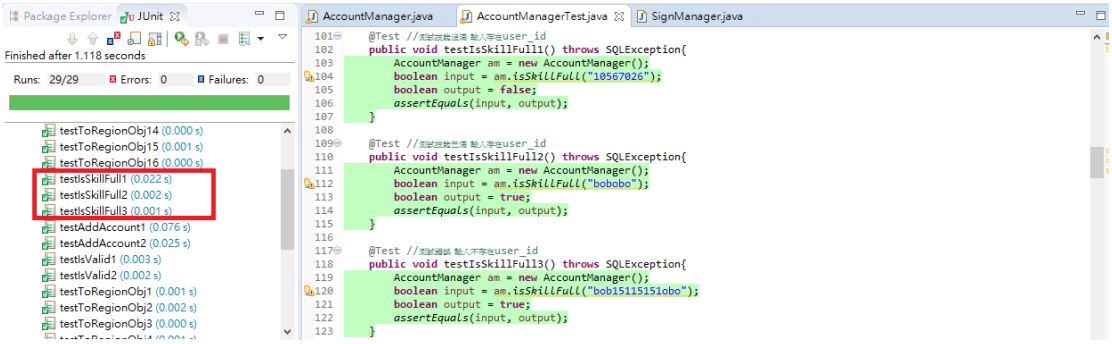
- AccountManager.isSkillFull()

Test Case 1	
Test Case Id	JUT-007
Test Function Name	testIsSkillFull1()
Test proposal	確認已存在資料庫帳號技能數量尚未達到上限
Input	input_id = "10567026"
Expected Output	確認技能數尚未達到上限，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-008
Test Function Name	testIsSkillFull2()
Test proposal	確認已存在資料庫帳號技能達到上限
Input	input_id = "bobobo"
Expected Output	確認技能數達到上限，回傳 true 值
Real Output	回傳 true 值

Test Result	Pass

Test Case 3	
Test Case Id	JUT-009
Test Function Name	testIsSkillFull3()
Test proposal	確認不存在資料庫帳號技能沒達到上限
Input	input_id = "bob15115151obo"
Expected Output	帳號不存在，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass



isSkillFull() Code

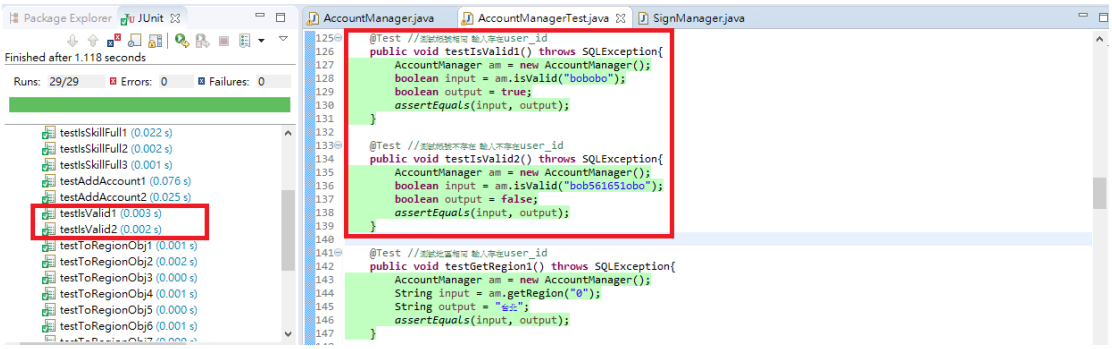
```
public static boolean isSkillFull(String id) throws SQLException {
        boolean result = true;
        String query = "select * from accounts where user_id = '" + id + "'";
        ResultSet rs = DataBaseAdmin.selectDB(query);
        if (rs.next())
            if (rs.getInt("skill_number") < rs.getInt("skill_max"))
                result = false;

        return result;
    }
```

- AccountManager.isValid()

Test Case 1	
Test Case Id	JUT-010
Test Function Name	testIsValid1()
Test proposal	驗證登入輸入的帳號在資料庫內存在
Input	input_id = "bobobo"
Expected Output	確認技能數達到上限，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-011

Test Function Name	testIsValid2()
Test proposal	驗證登入輸入的帳號在資料庫內不存在
Input	input_id = "bob561651obo"
Expected Output	確認技能數達到上限，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass



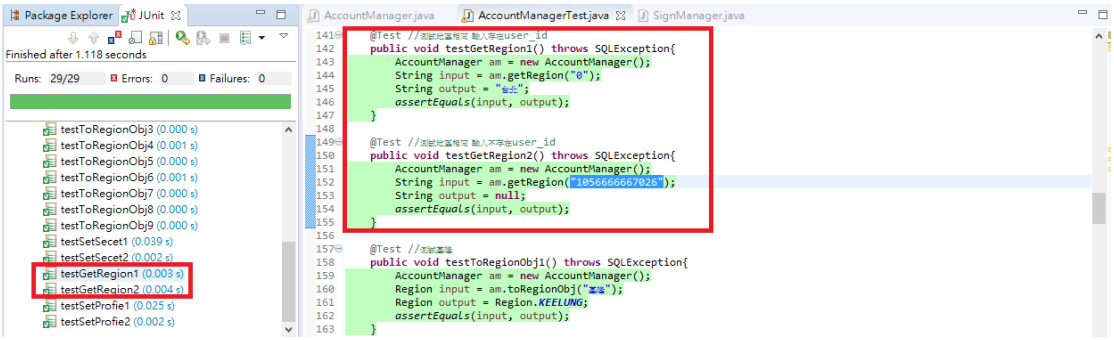
isValid() Code

```
// 驗證帳號登入
public boolean isValid(String id) {

        String query = "SELECT * FROM accounts where user_id='" + id + "'";
        try {
            ResultSet rs = DataBaseAdmin.selectDB(query);
        //System.out.println("[" + id + "] -> [" + rs +"]");

            if (rs.next()) {

                if (id.equals(rs.getString("user_id")))
                    return true;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return false;
    }
```

- AccountManager.getRegion()

Test Case 1	
Test Case Id	JUT-012
Test Function Name	testGetRegion1()
Test proposal	測試從資料庫得到的地區資料正確
Input	input_id = "0"
Expected Output	“台北”
Real Output	“台北”
Test Result	Pass

Test Case 2	
Test Case Id	JUT-013
Test Function Name	testGetRegion2()
Test proposal	測試從資料庫得到的地區資料不存在
Input	input_id = "1056666667026"
Expected Output	null
Real Output	null
Test Result	Pass




```
public String getRegion(String id) {
        String query = "SELECT * FROM accounts where user_id = '" + id + "' ";
        //System.out.println(id);
        //System.out.println(query);

        ResultSet rs = DataBaseAdmin.selectDB(query);
        try {
            rs.next();
            // System.out.println("[region]->"+rs.getString("region"));
            //System.out.println(rs.getString("region"));
            return rs.getString("region");
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return null;
    }
```

- AccountManager.toRegionObj()

Test Case 1	
Test Case Id	JUT-014
Test Function Name	testToRegionObj1()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“基隆”
Expected Output	Region.KEELUNG
Real Output	Region.KEELUNG
Test Result	Pass

Test Case 2	
Test Case Id	JUT-015
Test Function Name	testToRegionObj2()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“台北”
Expected Output	Region.TAIPEI
Real Output	Region.TAIPEI
Test Result	Pass

Test Case 3	
Test Case Id	JUT-016
Test Function Name	testToRegionObj3()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“桃園”
Expected Output	Region.TAOYUAN
Real Output	Region.TAOYUAN
Test Result	Pass

Test Case 4	
Test Case Id	JUT-017
Test Function Name	testToRegionObj4()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“新竹”
Expected Output	Region.HSINCHU
Real Output	Region.HSINCHU
Test Result	Pass

Test Case 5	
Test Case Id	JUT-018
Test Function Name	testToRegionObj5()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“苗栗”
Expected Output	Region.MIAOLI
Real Output	Region.MIAOLI
Test Result	Pass

Test Case 6	
Test Case Id	JUT-019
Test Function Name	testToRegionObj6()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“台中”
Expected Output	Region.TAICHUNG
Real Output	Region.TAICHUNG
Test Result	Pass

Test Case 7	
Test Case Id	JUT-020
Test Function Name	testToRegionObj7()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“彰化”
Expected Output	Region.CHANGHUA
Real Output	Region.CHANGHUA
Test Result	Pass

Test Case 8	
Test Case Id	JUT-021
Test Function Name	testToRegionObj8()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“南投”
Expected Output	Region.NANTOU
Real Output	Region.NANTOU
Test Result	Pass

Test Case 9	
Test Case Id	JUT-022
Test Function Name	testToRegionObj9()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“雲林”
Expected Output	Region.YUNLIN
Real Output	Region.YUNLIN
Test Result	Pass

Test Case 10

Test Case Id	JUT-023
Test Function Name	testToRegionObj10()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“嘉義”
Expected Output	Region.CHIAYI
Real Output	Region.CHIAYI
Test Result	Pass

Test Case 11	
Test Case Id	JUT-024
Test Function Name	testToRegionObj11()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“台南”
Expected Output	Region.TAINAN
Real Output	Region.TAINAN
Test Result	Pass

Test Case 12	
Test Case Id	JUT-025
Test Function Name	testToRegionObj12()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“高雄”
Expected Output	Region.KAOHSIUNG
Real Output	Region.KAOHSIUNG
Test Result	Pass

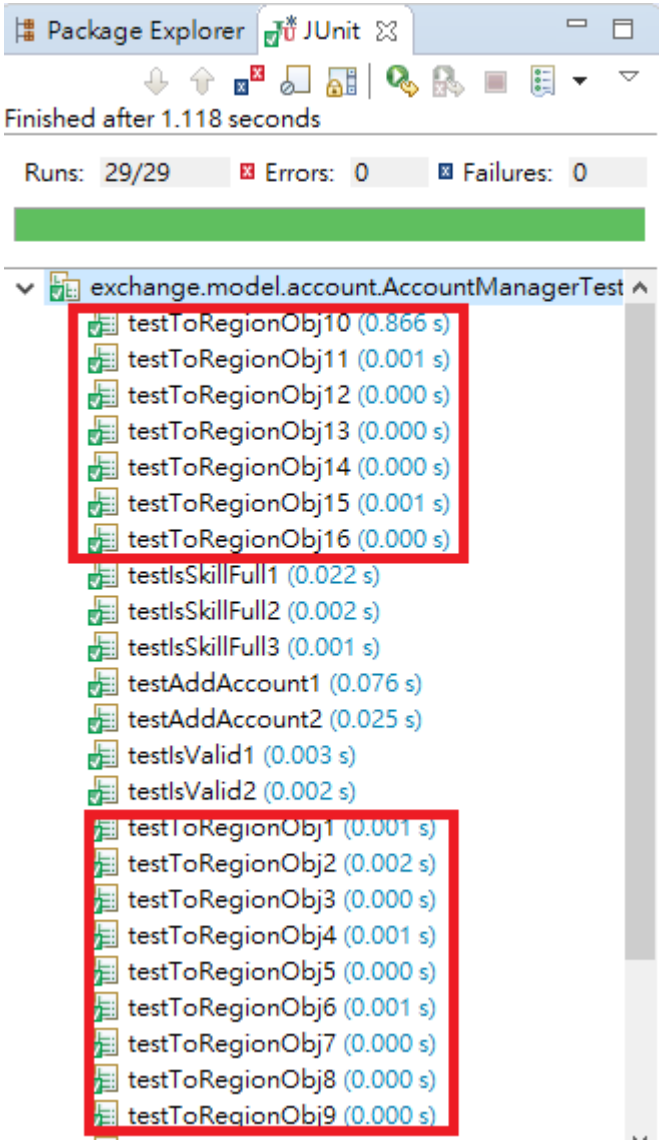
Test Case 13	
Test Case Id	JUT-026
Test Function Name	testToRegionObj13()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“屏東”
Expected Output	Region.PINGTUNG
Real Output	Region.PINGTUNG
Test Result	Pass

Test Case 14	
Test Case Id	JUT-027

Test Function Name	testToRegionObj14()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“宜蘭”
Expected Output	Region.ILAN
Real Output	Region.ILAN
Test Result	Pass

Test Case 15	
Test Case Id	JUT-028
Test Function Name	testToRegionObj15()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“花蓮”
Expected Output	Region.HUALIEN
Real Output	Region.HUALIEN
Test Result	Pass

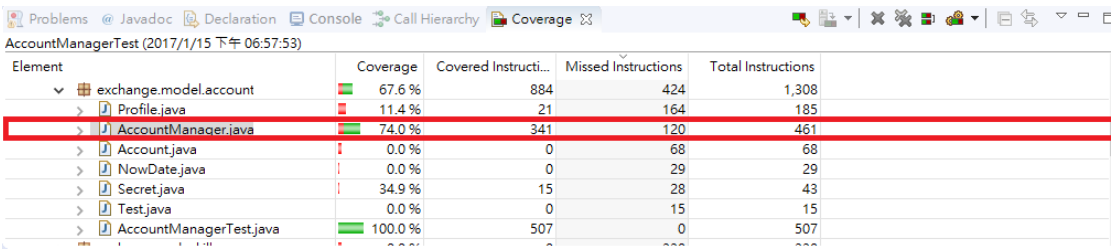
Test Case 16	
Test Case Id	JUT-029
Test Function Name	testToRegionObj16()
Test proposal	測試輸入的地區與輸出的 RegionObj 相同
Input	“台東”
Expected Output	Region.TAITUNG
Real Output	Region.TAITUNG
Test Result	Pass


Package Explorer
JUnit
Finished after 1.118 seconds
Runs: 29/29
Errors: 0
Failures: 0
exchange.model.account.AccountManagerTest
testToRegionObj10 (0.866 s)
testToRegionObj11 (0.001 s)
testToRegionObj12 (0.000 s)
testToRegionObj13 (0.000 s)
testToRegionObj14 (0.000 s)
testToRegionObj15 (0.001 s)
testToRegionObj16 (0.000 s)
testIsSkillFull1 (0.022 s)
testIsSkillFull2 (0.002 s)
testIsSkillFull3 (0.001 s)
testAddAccount1 (0.076 s)
testAddAccount2 (0.025 s)
testIsValid1 (0.003 s)
testIsValid2 (0.002 s)
testToRegionObj1 (0.001 s)
testToRegionObj2 (0.002 s)
testToRegionObj3 (0.000 s)
testToRegionObj4 (0.001 s)
testToRegionObj5 (0.000 s)
testToRegionObj6 (0.001 s)
testToRegionObj7 (0.000 s)
testToRegionObj8 (0.000 s)
testToRegionObj9 (0.000 s)

toRegionObj() Code

```
    public Region toRegionObj(String region) {

        switch (region) {
        case "基隆":
            return Region.KEELUNG;
        case "台北":
            return Region.TAIPEI;
        case "桃園":
            return Region.TAOYUAN;
        case "新竹":
            return Region.HSINCHU;
        case "苗栗":
            return Region.MIAOLI;
        case "台中":
            return Region.TAICHUNG;
        case "彰化":
            return Region.CHANGHUA;
        case "南投":
            return Region.NANTOU;
        case "雲林":
            return Region.YUNLIN;
        case "嘉義":
            return Region.CHIAYI;
        case "台南":
            return Region.TAINAN;
        case "高雄":
            return Region.KAOHSIUNG;
        case "屏東":
            return Region.PINGTUNG;
        case "宜蘭":
            return Region.ILAN;
        case "花蓮":
            return Region.HUALIEN;
        case "台東":
            return Region.TAITUNG;
        }

        return null;
    }
}
```

帳戶模組覆蓋率



➢ 登入模組單元測試

- SignManager.check()

Test Case 1	
Test Case Id	JUT-030

Test Function Name	testCheck1()
Test proposal	輸入已存在資料庫的帳號密碼，驗證登入
Input	input_id = “10567026” input_password = “dsdsdsds”
Expected Output	帳號密碼存在且正確，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-031
Test Function Name	testCheck2()
Test proposal	輸入不存在資料庫的帳號與已存在資料庫的密碼，驗證無法登入
Input	input_id = “105676” input_password = “dsdsdsds”
Expected Output	帳號不存在，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

Test Case 3	
Test Case Id	JUT-032
Test Function Name	testCheck3()
Test proposal	輸入已存在資料庫的帳號不存在資料庫的密碼，驗證無法登入
Input	input_id = “10567026” input_password = “kc5674”
Expected Output	帳號驗證成功，密碼不存在，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

Test Case 4	
Test Case Id	JUT-033
Test Function Name	testCheck4()
Test proposal	輸入不存在資料庫的帳號與密碼，驗證無法登入
Input	input_id = “105026” input_password = “kc5674”
Expected Output	帳號不存在，回傳 false 值

Real Output	回傳 false 值
Test Result	Pass

check() Code

```
public boolean check(Secret secret) {
    int result = 0;
    AccountManager am = new AccountManager();
    SignManager sm = new SignManager();
    Date recentLog = new Date();
    java.sql.Date sqlStartDate = new java.sql.Date(recentLog.getTime());
    //System.out.println("[帳號] -> ["+ secret.getPassword() +"]");
    if (am.isValid(secret.getId()) == true) { // 帳號是否存在
        //System.out.println("[帳號存在，比對密碼]");
        if (sm.CheckPassword(secret) == true) { // 密碼是否相同

            String query = "UPDATE accounts SET recent_Log = '" + sqlStartDate + "'";
            result = DataBaseAdmin.updateDB(query);
        }
    }
    // DataBaseAdmin.closeConnection();
    return (result == 0)? false:true;
}
```

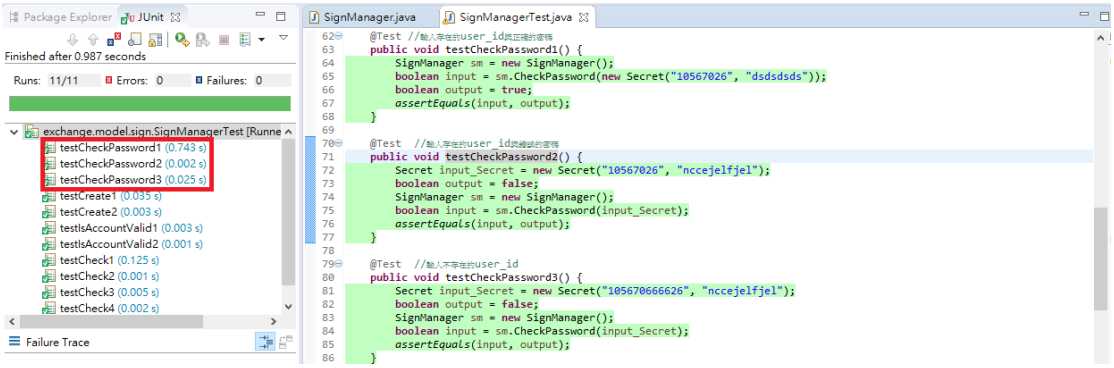
- SignManager.checkPassword()

Test Case 1	
Test Case Id	JUT-034
Test Function Name	test CheckPassword1()
Test proposal	輸入存在資料庫的帳號與密碼，確認密碼驗證正確
Input	input_id = “10567026” input_password = “dsdsdsds”
Expected Output	密碼驗證正確，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-035

Test Function Name	test CheckPassword2()
Test proposal	輸入存在資料庫的帳號與不正確的密碼，確認密碼驗證錯誤
Input	input_id = “10567026” input_password = “nccejelfje1”
Expected Output	密碼驗證錯誤，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

Test Case 3	
Test Case Id	JUT-036
Test Function Name	test CheckPassword3()
Test proposal	輸入不存在資料庫的帳號，無法驗證密碼
Input	input_id = “105670666626” input_password = “nccejelfje1”
Expected Output	無法驗證密碼，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass



CheckPassword() Code

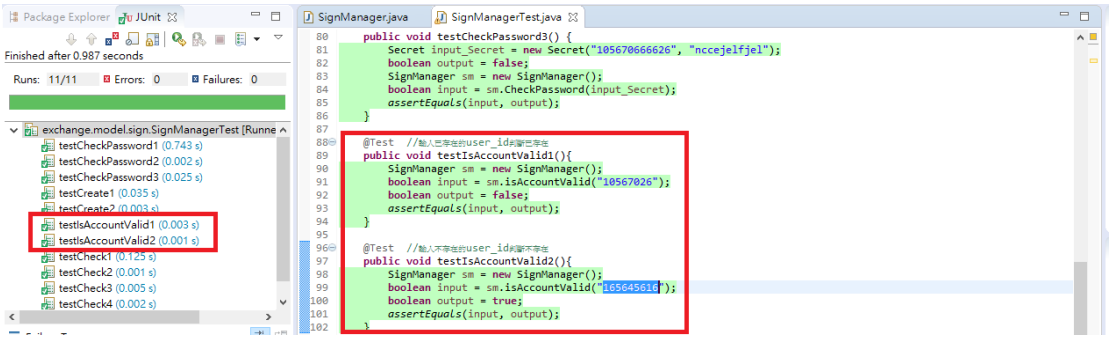
```
public boolean CheckPassword(Secret secret) {
    boolean result = false;
    String query = "SELECT * FROM accounts where user_id='" + secret.getId() + "'";
    ResultSet rs = DataBaseAdmin.selectDB(query);
    try {
        rs.next();
        //System.out.println("["+secret.getPassword()+"] -> ["+ rs.getString("password") +"]");
        if (secret.getPassword().equals(rs.getString("password")))
            result = true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result;
}
```

- SignManager.isAccountValid()

Test Case 1

Test Case Id	JUT-037
Test Function Name	testisAccountValid1()
Test proposal	輸入已存在資料庫的帳號註冊，驗證帳號已存在，無法註冊
Input	input_id = “10567026”
Expected Output	帳號已存在，無法註冊，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-038
Test Function Name	testisAccountValid2()
Test proposal	輸入不存在資料庫的帳號註冊，驗證帳號不存在，可以註冊
Input	input_id = “165645616”
Expected Output	帳號尚未存在，可以註冊，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass



isAccountValid() Code

```
// 確認帳號格式，重複與否
public boolean isAccountValid(String id) {
    boolean result = true;

    ResultSet rs = DataBaseAdmin.selectDB("SELECT * FROM accounts WHERE user_id='"+ id +"'");
    try {
        if (rs.next()) {
            if (id.equals(rs.getString("user_id")))
                result = false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result;
}
```

- SignManager.create()

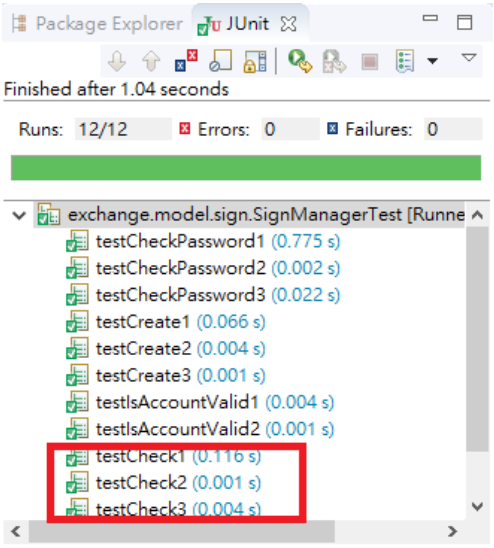
Test Case 1	
Test Case Id	JUT-039

Test Function Name	test Create1()
Test proposal	輸入不存在資料庫的帳號資料，測試註冊成功
Input	input_id = "15561315" input_password = "gd55kdls" input_userName = "ck66ja" input_nickName = " slmcs66mc " input_gender = true(男性) input_email = "djjf66kkfd" input_birthday = "2016-09-06"; input_region = "台北"
Expected Output	註冊成功，回傳 true 值
Real Output	回傳 true 值
Test Result	Pass

Test Case 2	
Test Case Id	JUT-040
Test Function Name	test Create2()
Test proposal	輸入不存在資料庫的帳號資料其中有 null 值，測試註冊失敗
Input	nput_id = null input_password = "gd55kdls" input_userName = "ck66ja" input_nickName = " slmcs66mc " input_gender = true(男性) input_email = "djjf66kkfd" input_birthday = "2016-09-06"; input_region = "台北"
Expected Output	註冊失敗，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass

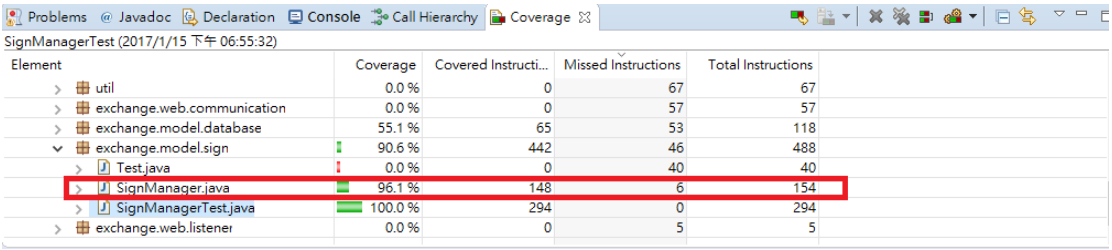
Test Case 3	
Test Case Id	JUT-041
Test Function Name	test Create3()
Test proposal	輸入已存在資料庫的帳號資料，測試建立失敗
Input	input_id = "10567026" input_password = "gd55kdls" input_userName = "ck66ja"

	input_nickName = " slmcs66mc " input_gender = true(男性) input_email = "djjf66kkfd" input_birthday = "2016-09-06"; input_region = "台北"
Expected Output	註冊失敗，回傳 false 值
Real Output	回傳 false 值
Test Result	Pass




```
@Test  //輸入相關資料，建立成功
public void testCreate1(){
    Secret secret = new Secret("15561315", "gd55kdls");
    Profile profile = new Profile("ck66ja", "slmcs66mc",
            true, "djjf66kkfd", "2016-09-06", "台北");
    Date recentLog = new Date();
    java.sql.Date sqlStartDate = new java.sql.Date(recentLog.getTime());
    Account input_Account = new Account(secret, profile, sqlStartDate);
    boolean output = true;
    SignManager sm = new SignManager();
    boolean input = sm.create(input_Account);
    assertEquals(input, output);
  }

@Test //輸入有null 建立失敗
public void testCreate2(){
    Secret secret = new Secret(null, "gddlkdls");
    Profile profile = new Profile("ckja", "slmcsmc",
            true, "djjfkkfd", "2016-09-06", "台北");
    Date recentLog = new Date();
    java.sql.Date sqlStartDate = new java.sql.Date(recentLog.getTime());
    Account input_Account = new Account(secret, profile, sqlStartDate);
    SignManager sm = new SignManager();
    boolean input = sm.create(input_Account);
    boolean output = false;
    assertEquals(input, output);
}

@Test //輸入已存在相關資料 建立失敗
public void testCreate3(){
    Secret secret = new Secret("10567026", "gd55kdls");
    Profile profile = new Profile("ck66ja", "slmcs66mc",
            true, "djjf66kkfd", "2016-09-06", "台北");
    Date recentLog = new Date();
    java.sql.Date sqlStartDate = new java.sql.Date(recentLog.getTime());
    Account input_Account = new Account(secret, profile, sqlStartDate);
    SignManager sm = new SignManager();
    boolean input = sm.create(input_Account);
    boolean output = false;
    assertEquals(input, output);
}
```

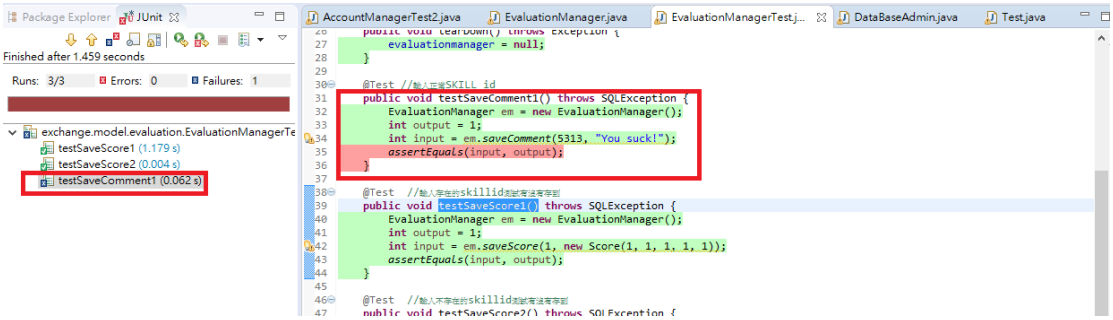
登入模組覆蓋率



➢ 評價模組單元測試

• EvaluationManager.saveComment()

Test Case 1	
Test Case Id	JUT-042
Test Function Name	testSaveComment1()
Test proposal	輸入 skillId、comment，測試建立評價成功
Input	input_skillId = “1” input_comment = ”good!”
Expected Output	建立成功，回傳值為 1
Real Output	回傳值為 0
Test Result	fault



saveComment() Code

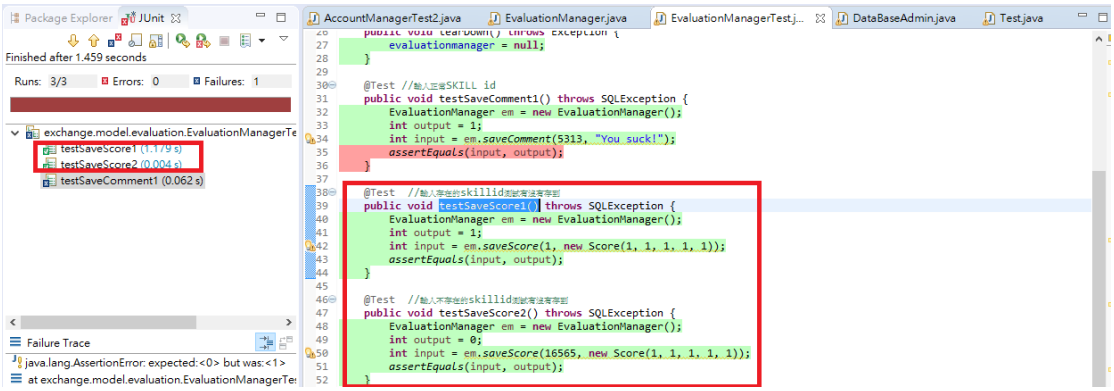
```
public class EvaluationManager{

    public static int saveComment(int skillId, String comment) throws SQLException
    {
        int result = 0;
        Date recentLog = new Date();
        java.sql.Date now = new java.sql.Date(recentLog.getTime());
        try{
            String query = "INSERT INTO comments VALUES ( "+skillId+" , '"+comment+"', '"+now+"' )";
//          String query = "INSERT comments SET skill_id = '"+ skillId +"', comment = '"+ comment +"',date = '"+ now +"'";
            result = DataBaseAdmin.updateDB(query);
        }catch(Exception e){
            e.printStackTrace();
        }
        return result;
    }
```

• EvaluationManager.saveScore()

Test Case 1	
Test Case Id	JUT-043

Test Function Name	testSaveScore1()
Test proposal	輸入已存在資料庫 skillId，測試更新到分數
Input	input_skillId = “1” input_score = Score(1, 1, 1, 1, 1)
Expected Output	更新成功，回傳值為 1
Real Output	回傳值為 1
Test Result	Pass

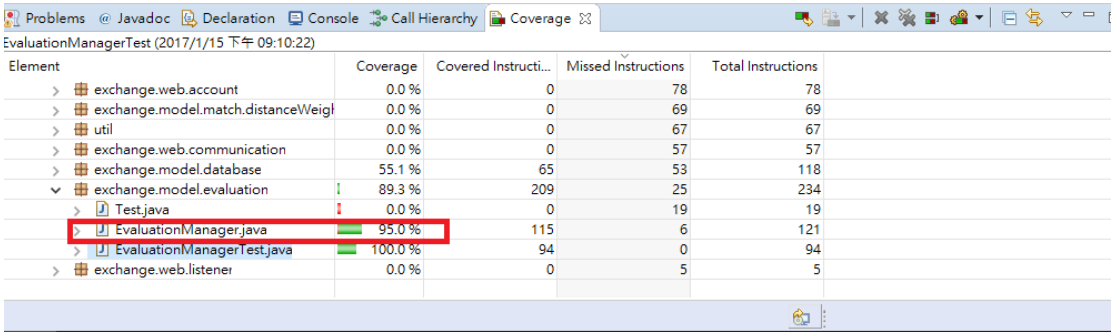
Test Case 2	
Test Case Id	JUT-044
Test Function Name	testSaveScore2()
Test proposal	輸入不存在資料庫 skillId，測試更新不到分數
Input	input_skillId = “11513131131” input_score = Score(1, 1, 1, 1, 1)
Expected Output	更新失敗，回傳值為 0
Real Output	回傳值為 0
Test Result	Pass



saveScore() Code

```
public static int saveScore(int skillId, Score score) throws SQLException
{
    int result = 0;
    Skill skill = SkillManager.findSkill(skillId);
    try{
        String query = "UPDATE skills SET attitude_score = '"+ (skill.getScore().getFrequency() + score.getAttitude()
        + " profession_score = '"+ (skill.getScore().getProfession() + score.getProfession()) +"', "
        + "teaching_score = '"+(skill.getScore().getTeaching() + score.getTeaching())+"',"
        + " frequency_score = '"+(skill.getScore().getFrequency() + score.getFrequency()) +"', "
        + "satisfication_score = '"+ (skill.getScore().getSatisfication() + score.getSatisfication()) +"'"
        + "where skill_id = '"+skillId+"'";
        result = DataBaseAdmin.updateDB(query);
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
```

評價模組覆蓋率

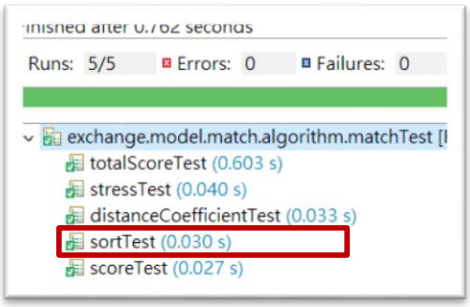


➢ 配對模組單元測試

• BasicAlgorithm.sort()

Test Case 1	
Test Case Id	JUT-045
Test Function Name	sort()
Test proposal	檢查陣列之技能分數，是否由大而小排序。
Input	資料庫成績
Expected Output	true
Real Output	true
Test Result	Pass

```
@Test
public void sortTest() {

        ba = new BasicAlgorithm("vegetable",1);
        cs=ba.getSkillArray("vegetable");
        assertSortSuccess(cs);


}
```

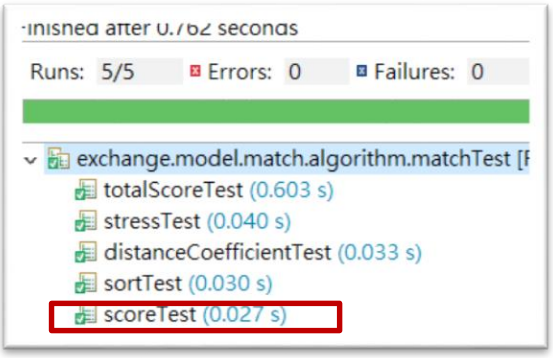



```
private void assertSortSuccess(ArrayList<CandidateSkill> cs){ //測試排序有無錯誤

    for(int i=0;i<cs.size()-1;i++){
        boolean larger=(cs.get(i).getTotalScore()>=cs.get(i+1).getTotalScore());
        assertTrue(larger);
    }

}
```

• BasicAlgorithm.toMatch()

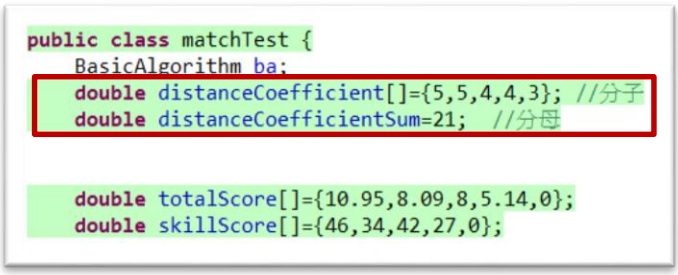
Test Case 1	
Test Case Id	JUT-046
Test Function Name	toMatch()
Test proposal	模擬配對後，如果多次按下 next 超過最大數量時，是否會回傳 null 而不會發生 ArrayIndexOutOfBoundsException。
Input	資料庫取得資料
Expected Output	true
Real Output	true
Test Result	Pass

```
@Test
public void stressTest() {

      ba=new BasicAlgorithm("vegetable", 1);
      ba.getSkillArray("vegetable");

      int queueSize=ba.getSkillQueue().size();
      for(int i=0;i<=queueSize ;i++){


          Skill skill;
          skill=ba.toMatch();
          if(i!=queueSize){
              assertNotEquals(skill, null);
          }
          else{
              assertEquals(skill, null);
          }
      }
```

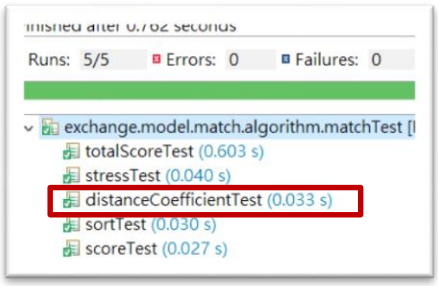


• BasicAlgorithm.computeDistanceWeight()

Test Case 1	
Test Case Id	JUT-048
Test Function Name	computeDistanceWeight(ArrayList<CandidateSkill> cs)
Test proposal	檢查距離權重是否正確
Input	以 vegetable 帳號使用技能編號為 1 的技能進行配對所計算出來的 5 個技能之距離權重
Expected Output	true
Real Output	true
Test Result	pass



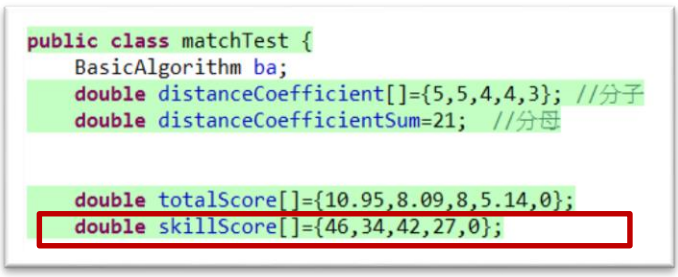
```
public class matchTest {
    BasicAlgorithm ba;
    double distanceCoefficient[]={5,5,4,4,3}; //分子
    double distanceCoefficientSum=21;  //分母


    double totalScore[]={10.95,8.09,8,5.14,0};
    double skillScore[]={46,34,42,27,0};
```



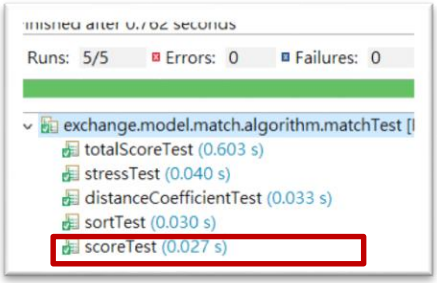

```
@Test
public void distanceCoefficientTest(){
    ba=new BasicAlgorithm("vegetable", 1);
    cs=ba.getSkillArray("vegetable");


    for(int i=0;i<cs.size()-1;i++){
        assertEquals(cs.get(i).getDistanceWeight_numerator(), distanceCoefficient[i],0);
        assertEquals(cs.get(i).getDistanceWeight_denominator(),distanceCoefficientSum,0);
    }
}
```

• BasicAlgorithm.SumEvalScoreWithFitLvWt()

Test Case 1	
Test Case Id	JUT-049
Test Function Name	SumEvalScoreWithFitLvWt()
Test proposal	檢查分數是否正確
Input	以 vegetable 帳號使用技能編號為 1 的技能進行配對所計算出來的 5 個技能之分數
Expected Output	true
Real Output	true
Test Result	pass



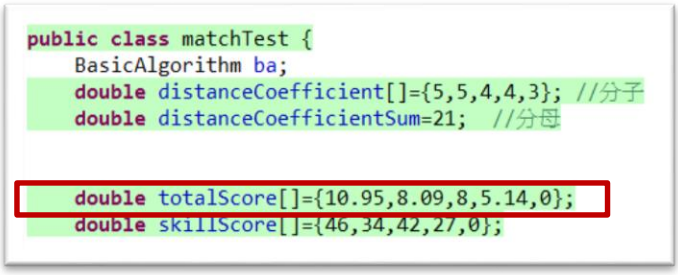
```
public class matchTest {
    BasicAlgorithm ba;
    double distanceCoefficient[]={5,5,4,4,3}; //分子
    double distanceCoefficientSum=21;  //分母


    double totalScore[]={10.95,8.09,8,5.14,0};
    double skillScore[]={46,34,42,27,0};
```



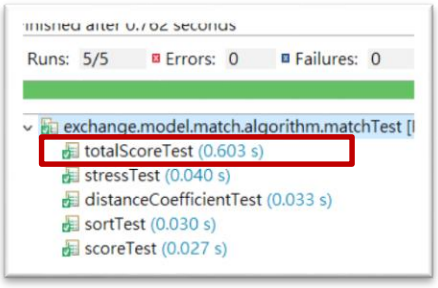

```
@Test
public void scoreTest(){
    ba=new BasicAlgorithm("vegetable", 1);
    cs=ba.getSkillArray("vegetable");

    for(int i=0;i<cs.size();i++){
        assertEquals(cs.get(i).getSkillScore(), skillScore[i],0);
    }
}
```

• BasicAlgorithm.calculateTotalScore()

Test Case 1	
Test Case Id	JUT-050
Test Function Name	calculateTotalScore()
Test proposal	檢查分數乘上距離權重後之總分數是否正確
Input	以 vegetable 帳號使用技能編號為 1 的技能進行配對所計算出來的 5 個技能之總分數
Expected Output	true
Real Output	true
Test Result	pass



```
public class matchTest {
    BasicAlgorithm ba;
    double distanceCoefficient[]={5,5,4,4,3}; //分子
    double distanceCoefficientSum=21;  //分母


    double totalScore[]={10.95,8.09,8,5.14,0};
    double skillScore[]={46,34,42,27,0};
```




```
@Test
public void totalScoreTest(){
    ba=new BasicAlgorithm("vegetable", 1);
    cs=ba.getSkillArray("vegetable");

    for(int i=0;i<cs.size();i++){
        assertEquals(cs.get(i).getTotalScore(), totalScore[i],0.1);
    }
}
```

配對模組覆蓋率

exchange.model.match.algorithm	89.8 %	465	53	518
Test.java	0.0 %	0	52	52
matchTest.java	99.7 %	289	1	290
BasicAlgorithm.java	100.0 %	176	0	176

exchange.model.match.skillScore	95.1 %	176	9	185
SumEvalScoreWithFitLvWt.java	95.1 %	176	9	185

exchange.model.match	89.2 %	272	33	305
Region.java	88.9 %	169	21	190
CandidateSkill.java	88.0 %	88	12	100
Area.java	100.0 %	15	0	15

exchange.model.match.skillRetrieval	98.6 %	435	6	441
FavoriteRegionRetrieval.java	98.8 %	238	3	241
RegionRetrieval.java	98.5 %	197	3	200

exchange.model.match.distanceWeight	100.0 %	69	0	69
NormalizationWeight.java	100.0 %	69	0	69

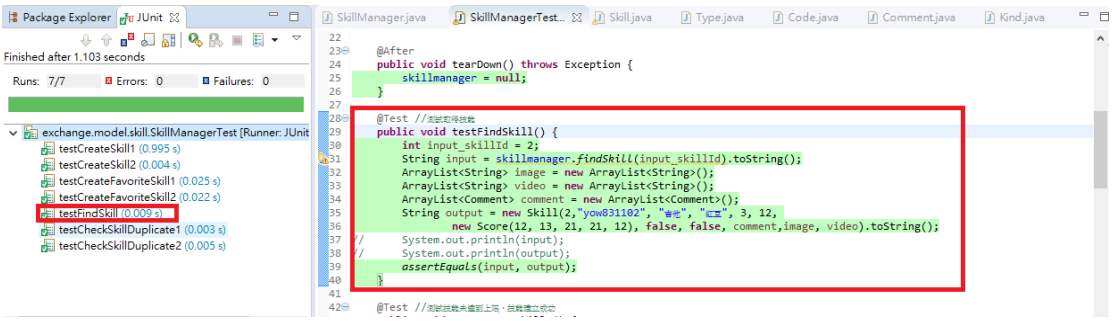
exchange.model.match.regionMatrix	100.0 %	2,143	0	2,143
RealDistanceOrder.java	100.0 %	2,143	0	2,143

➢ 技能模組單元測試

• SkillManager.findSkill()

Test Case 1	
Test Case Id	JUT-051
Test Function Name	TestfindSkill()
Test proposal	測試取得正確帳號資料
Input	Input_skillId = “2” Input_userId = “yow831102” Input_typeName = “吉他” Input_introExpr = “紅豆” Input_skillLevel = 3 Input_times = 12 Input_score = (12, 13, 21, 21, 12) badTag = false warningTag = false comment = empty set image = empty set video = empty set
Expected Output	true
Real Output	true

Test Result	Pass



findSkill() Code

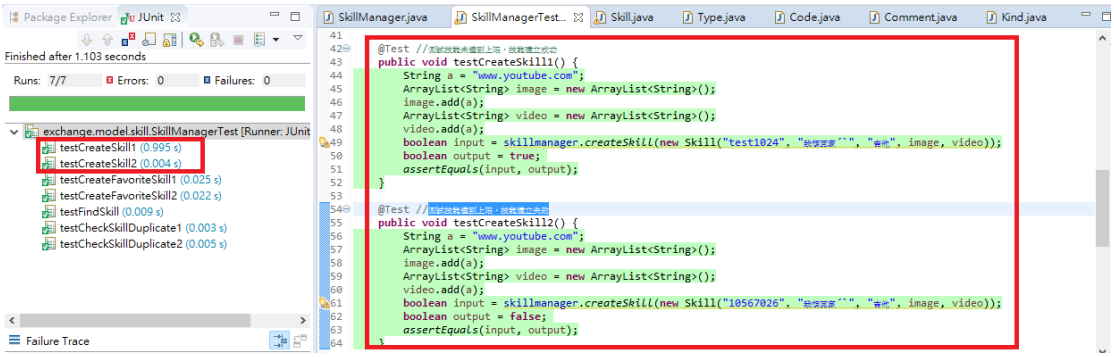
```
    static public Skill findSkill(int skillId) {
        ResultSet rs = DataBaseAdmin.selectDB("SELECT * FROM skills WHERE skill_id = '" + skillId + "'");
        Skill skill = new Skill();

        try {
            if (rs.next()) {

                String userId = rs.getString("user_id");
                String typeName = rs.getString("type_name");
                String introExpr = rs.getString("intro_expr");
                int skillLevel = rs.getInt("skill_level");
                int times = rs.getInt("times");
                Score score = new Score(rs.getInt("attitude_score"), rs.getInt("profession_score"),
                        rs.getInt("teaching_score"), rs.getInt("frequency_score"), rs.getInt("satisfication_score"));
                boolean warningTag = (rs.getBoolean("bad_tag")) ? true : false;
                boolean badTag = (rs.getBoolean("warning_tag")) ? true : false;
                ArrayList<Comment> comment = new ArrayList<Comment>();
                ArrayList<String> image = new ArrayList<String>();
                ArrayList<String> video = new ArrayList<String>();

                rs = DataBaseAdmin.selectDB("SELECT * FROM comments WHERE skill_id = '" + skillId + "'");
                while (rs.next())
                    comment.add(new Comment(rs.getString("comment"), rs.getString("date")));

                rs = DataBaseAdmin.selectDB("SELECT * FROM images WHERE skill_id = '" + skillId + "'");
                while (rs.next())
                    image.add(rs.getString("image"));

                rs = DataBaseAdmin.selectDB("SELECT * FROM videos WHERE skill_id = '" + skillId + "'");
                while (rs.next())
                    video.add(rs.getString("video"));

                skill = new Skill(skillId, userId, typeName, introExpr, skillLevel, times, score, warningTag, badTag,
                        comment, image, video);

            } else {
                System.out.println("資料庫無此筆資料");
                return new Skill();
            }
        } catch (SQLException e) {
```

- SkillManager.createSkill()

Test Case 1	
Test Case Id	JUT-052
Test Function Name	TestCreateSkill1()
Test proposal	測試技能未達到上限，技能建立成功
Input	Input_skillId = “test1024” Input_introExpr = “我想回家ㄟㄟ” Input_typeName = “吉他”

	image = "www.youtube.com" video = "www.youtube.com"
Expected Output	true
Real Output	true
Test Result	Pass

Test Case 2	
Test Case Id	JUT-053
Test Function Name	TestCreateSkill2()
Test proposal	測試技能達到上限，技能建立失敗
Input	Input_skillId = “10567026” Input_introExpr = “我想回家ノヽ” Input_typeName = “吉他” image = "www.youtube.com" video = "www.youtube.com"
Expected Output	false
Real Output	false
Test Result	Pass



createSkill() Code

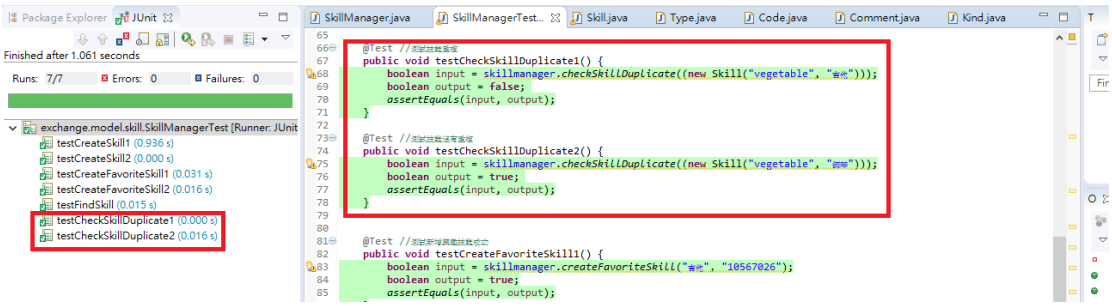
```
static public boolean createSkill(Skill skill) {
    // 判斷是否可新增技能
    int flag = 0;
    int skillId = 0;
    String userId = skill.getUserId();
    String typeName = skill.getType().getTypeName();

    try {
        if (!AccountManager.isSkillFull(userId)) {
            // 資料庫會自動判斷是否已存在
            /*
             * ResultSet rs = DataBaseAdmin.
             * selectDB("SELECT * FROM skills where user_id = '" + userId +
             * "'"); rs.next(); if (skill.getUserId() ==
             * rs.getString("user_id") && skill.getType().getTypeName() ==
             * rs.getString("type_name")){ System.out.println("已建立'"+
             * skill.getType().getTypeName() + "'類別的興趣技能"); return; }
             */

            flag = DataBaseAdmin.updateDB("INSERT INTO skills VALUES('0','" + userId + "','" + typeName + "','"
                    + skill.getIntorExpr() + "','0','0','0','0','0','0','0','0','0')");
            System.out.println("建立技能");

            if (flag != 0) {
                DataBaseAdmin.updateDB("UPDATE accounts SET skill_number = (select skill_number where user_id='"
                        + userId + "')+1 where user_id ='" + userId + "'");

                ResultSet rs = DataBaseAdmin.selectDB(
                        "SELECT * FROM skills where user_id = '" + userId + "' AND type_name ='" + typeName + "'");

                if (rs.next())
                    skillId = rs.getInt("skill_id");

                for (String video : skill.getVideo())
                    DataBaseAdmin.updateDB("INSERT INTO videos VALUES('" + skillId + "','" + video + "')");

                for (String image : skill.getImage())
                    DataBaseAdmin.updateDB("INSERT INTO images VALUES('" + skillId + "','" + image + "')");
                System.out.println("新增技能成功");
            }


        } else {
            System.out.println(userId + " 技能已達上限");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return (flag == 0) ? false : true;
}
```

- SkillManager.checkSkillDuplicate()

Test Case 1	
Test Case Id	JUT-054
Test Function Name	TestCheckSkillDuplicate1()
Test proposal	新增不在資料庫中的技能，技能沒有重複，新增成功
Input	Input_user_id = “vegetable” Input_typeName = “吉他”
Expected Output	true
Real Output	true
Test Result	Pass

Test Case 2	
Test Case Id	JUT-055
Test Function Name	TestCheckSkillDuplicate2()

Test proposal	新增已在資料庫中的技能，技能重複，新增失敗
Input	Input_user_id = “vegetable” Input_typeName = “鋼琴”
Expected Output	false
Real Output	false
Test Result	Pass



skillDuplicate() Code

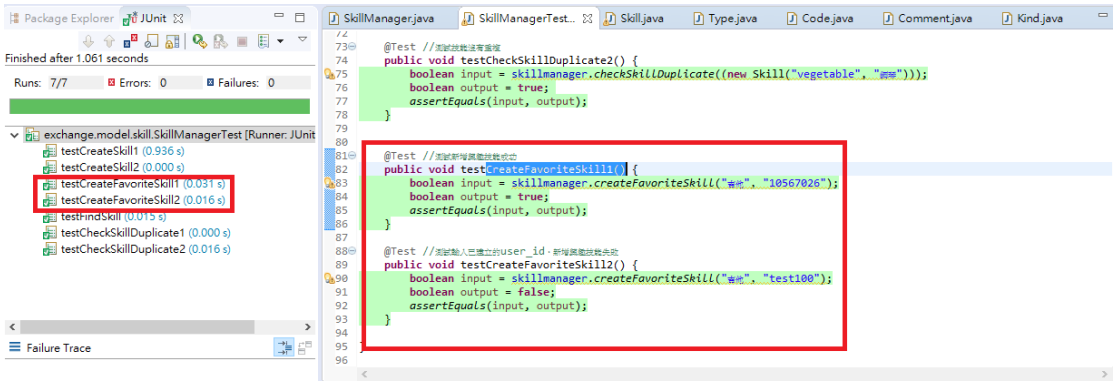
```
//判斷是否是重複技能卡
//要新增的資料沒有在資料庫中 => 回傳TRUE可以新增!
static public boolean checkSkillDuplicate(Skill skill)
{
    int count = 0;
    String userId = skill.getUserId();
    String typeName = skill.getType().getTypeName();

    ResultSet rs = DataBaseAdmin.selectDB(
            "SELECT * FROM skills where user_id = '" + userId + "'");

    try {
        while(rs.next())
        {
            if(rs.getString("type_name").equals(typeName)) count++;
            //System.out.println("["+rs.getString("type_name")+"]->["+typeName+"]");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    //return count;
    return (count == 0)? true:false;
}
```

- SkillManager.createFavoriteSkill()

Test Case 1	
Test Case Id	JUT-056
Test Function Name	TestCreateFavoriteSkill1()
Test proposal	新增不在資料庫中的技能，技能沒有重複，新增成功
Input	Input_typeName = “吉他” Input_user_id = “10567026”

Expected Output	true
Real Output	true
Test Result	Pass

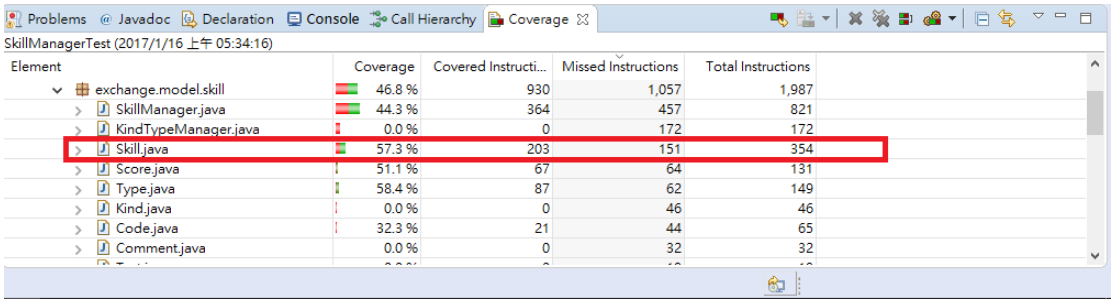
Test Case 2	
Test Case Id	JUT-057
Test Function Name	TestCreateFavoriteSkill2()
Test proposal	新增已在資料庫中的技能，技能重複，新增失敗
Input	Input_typeName = “吉他” Input_user_id = “test100”
Expected Output	false
Real Output	false
Test Result	Pass



createFavoriteSkill() Code

```
static public boolean createFavoriteSkill(String typeName, String userId) {
      int flag = 0;
      //System.out.println("[createFavoriteSkill]:[typeName]->" + typeName + ",[userId]" + userId);
      try{
          flag = DataBaseAdmin.updateDB("INSERT INTO favorites VALUES('" + typeName + "','" + userId + "')");
      }catch (Exception e) {
          e.printStackTrace();
      }
      System.out.println(flag);
      return (flag == 0)? false:true;
  }
```

技能模組覆蓋率



3.2 系統測試(System Test)

Identification	E-STC-001
Name	連結個人頁面系統測試
Tested Target	技能交換平台(Exchange)之導覽列
Reference	E-FR-NB-001
Severity	1
Instructions	點選個人頁面
Excepted Result	進入個人頁面
Cleanup	無

Identification	E-STC-002
Name	連結交流頁面系統測試
Tested Target	技能交換平台(Exchange)之導覽列
Reference	E-FR-NB-002
Severity	1
Instructions	點選交流列表
Excepted Result	進入交流頁面
Cleanup	無

Identification	E-STC-003
Name	登出系統功能系統測試
Tested Target	技能交換平台(Exchange)之導覽列
Reference	E-FR-NB-003
Severity	1
Instructions	點選登出
Excepted Result	進入首頁
Cleanup	無

Identification	E-STC-004

Name	註冊帳戶功能系統測試
Tested Target	技能交換平台(Exchange)之首頁
Reference	E-FR-SU-001
Severity	1
Instructions	1. 點選註冊按鈕 2. 輸入個人資料 3. 點選提交
Excepted Result	1. 註冊成功訊息 2. 顯示首頁
Cleanup	無

Identification	E-STC-005
Name	登入系統功能系統測試
Tested Target	技能交換平台(Exchange)之首頁
Reference	E-FR-SGI-001
Severity	1
Instructions	1. 點選登入按鈕 2. 輸入帳號與密碼 3. 點選登入
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-006
Name	修改密碼功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-AM-001
Severity	1
Instructions	1. 點選修改密碼 2. 輸入密碼、確認密碼 3. 點選確認
Excepted Result	3. 密碼修改成功訊息 4. 顯示個人頁面
Cleanup	無

Identification	E-STC-007
Name	顯示個人頁面系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-SMG-001、E-FR-PMG-003

Severity	1
Instructions	進入個人頁面
Excepted Result	1. 顯示所有已新增的技能 2. 顯示所有已新增的興趣技能
Cleanup	無

Identification	E-STC-008
Name	查看技能資訊功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面與交流頁面
Reference	E-FR-SMG-002
Severity	1
Instructions	點選技能圖示
Excepted Result	顯示技能頁面中的技能資訊
Cleanup	無

Identification	E-STC-009
Name	建立技能功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-SMG-003、E-FR-SA-001
Severity	1
Instructions	1. 按下建立技能按鈕 2. 填寫名稱、種類、程度、經歷欄位 3. 填寫展示資料(嵌入影片、上傳照片) 4. 按下送出按鈕
Excepted Result	個人頁面出現新增的技能
Cleanup	無

Identification	E-STC-010
Name	修改技能功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-SMG-004、E-FR-SMF-001
Severity	1
Instructions	1. 按下修改技能按鈕 2. 修改程度、經歷欄位 3. 修改展示資料(嵌入影片、上傳照片) 4. 按下送出按鈕
Excepted Result	1. 技能修改成功訊息 2. 顯示個人頁面

Cleanup	無

Identification	E-STC-011
Name	技能配對功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-SMG-005
Severity	1
Instructions	按下技能配對按鈕
Excepted Result	顯示配對頁面
Cleanup	無

Identification	E-STC-012
Name	顯示個人資料功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-PMG-001
Severity	1
Instructions	按下個人圖示
Excepted Result	顯示個人資料
Cleanup	無

Identification	E-STC-013
Name	修改個人資料功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-PMG-002、E-FR-PMF-001
Severity	1
Instructions	1. 按下修改個人資料按鈕 2. 修改個人資料 3. 按下送出按鈕
Excepted Result	1. 修改成功訊息 2. 顯示個人頁面
Cleanup	無

Identification	E-STC-014
Name	建立興趣技能功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-PMG-004、E-FR-FSA-001
Severity	1
Instructions	1. 按下新增興趣技能按鈕

	2. 選擇興趣技能資料 3. 按下送出按鈕
Excepted Result	個人頁面出現新增的興趣技能
Cleanup	無

Identification	E-STC-015
Name	刪除興趣技能功能系統測試
Tested Target	技能交換平台(Exchange)之個人頁面
Reference	E-FR-PMG-005
Severity	1
Instructions	1. 按下刪除興趣技能按鈕 2. 按下確認按鈕
Excepted Result	個人頁面顯示所有興趣技能，除了被刪除的興趣技能
Cleanup	無

Identification	E-STC-016
Name	顯示配對技能功能系統測試
Tested Target	技能交換平台(Exchange)之配對頁面
Reference	E-FR-SEG-001、E-FR-SEG-002
Severity	1
Instructions	在個人頁面中按下技能配對按鈕
Excepted Result	1. 顯示所有已新增的興趣技能 2. 顯示依配對演算法挑選出的技能之資訊
Cleanup	無

Identification	E-STC-017
Name	另尋配對功能系統測試
Tested Target	技能交換平台(Exchange)之配對頁面
Reference	E-FR-SEG-003
Severity	1
Instructions	在配對頁面中按下 Next 按鈕
Excepted Result	顯示依配對演算法挑選出的技能之資訊
Cleanup	無

Identification	E-STC-018
Name	結束配對功能系統測試
Tested Target	技能交換平台(Exchange)之配對頁面
Reference	E-FR-SEG-004

Severity	1
Instructions	在配對頁面中按下 Exit 按鈕
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-019
Name	訊息交流功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-001
Severity	1
Instructions	按下訊息交流按鈕
Excepted Result	顯示聊天頁面
Cleanup	無

Identification	E-STC-020
Name	接受邀請功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-002
Severity	1
Instructions	按下接受邀請按鈕
Excepted Result	1. 交流頁面的邀請關係消失 2. 交流頁面顯示交換關係
Cleanup	無

Identification	E-STC-021
Name	拒絕邀請功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-003
Severity	1
Instructions	按下拒絕邀請按鈕
Excepted Result	交流頁面的邀請關係消失
Cleanup	無

Identification	E-STC-022
Name	取消邀請功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-004
Severity	1

Instructions	按下取消邀請按鈕
Excepted Result	技能可以進行配對功能
Cleanup	無

Identification	E-STC-023
Name	顯示評價頁面功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-005
Severity	1
Instructions	按下技能評價按鈕
Excepted Result	顯示技能評價頁面
Cleanup	無

Identification	E-STC-024
Name	技能評價功能系統測試
Tested Target	技能交換平台(Exchange)之技能評價頁面
Reference	E-FR-SEL-001、E-FR-SEL-002、E-FR-SEL-003
Severity	1
Instructions	1. 選擇各項星等 2. 輸入評論 3. 按下送出按鈕
Excepted Result	1. 交流頁面的交換關係消失 2. 兩技能的擁有者無法進行交流 3. 未進行評價之技能應先評價才能再進行交換
Cleanup	無

Identification	E-STC-025
Name	取消評價功能系統測試
Tested Target	技能交換平台(Exchange)之技能評價頁面
Reference	E-FR-SEL-004
Severity	1
Instructions	按下取消按鈕
Excepted Result	顯示交流頁面
Cleanup	無

Identification	E-STC-026
Name	查看技能評論與展示功能系統測試
Tested Target	技能交換平台(Exchange)之技能頁面

Reference	E-FR-SIF-001、E-FR-SIF-002
Severity	1
Instructions	進入技能頁面
Excepted Result	1. 顯示所有的評論 2. 顯示技能的展示
Cleanup	無

Identification	E-STC-027
Name	取消修改功能系統測試
Tested Target	技能交換平台(Exchange)之個人資料修改頁面
Reference	E-FR-PMF-002
Severity	1
Instructions	按下取消按鈕
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-028
Name	取消新增功能系統測試
Tested Target	技能交換平台(Exchange)之興趣技能新增頁面
Reference	E-FR-FSA-002
Severity	1
Instructions	按下取消按鈕
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-029
Name	取消新增功能系統測試
Tested Target	技能交換平台(Exchange)之建立技能頁面
Reference	E-FR-SA-002
Severity	1
Instructions	按下取消按鈕
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-030
Name	取消修改功能系統測試
Tested Target	技能交換平台(Exchange)之技能修改頁面
Reference	E-FR-SMF-002

Severity	1
Instructions	按下取消按鈕
Excepted Result	顯示個人頁面
Cleanup	無

Identification	E-STC-031
Name	送出訊息功能系統測試
Tested Target	技能交換平台(Exchange)之聊天頁面
Reference	E-FR-CP-001
Severity	1
Instructions	按下送出按鈕
Excepted Result	聊天視窗顯示送出的訊息
Cleanup	無

Identification	E-STC-032
Name	5 秒內回應壓力測試
Tested Target	技能交換平台(Exchange)之所有功能
Reference	E-NFR-PR-001
Severity	1
Instructions	使用 JMeter 對各功能進行存取
Excepted Result	各功能系統皆能在 5 秒內回應
Cleanup	無

Identification	E-STC-033
Name	100 位使用者壓力測試
Tested Target	技能交換平台(Exchange)之所有功能
Reference	E-NFR-PR-002
Severity	1
Instructions	使用 JMeter 以 100 的執行緒數量對各功能進行存取
Excepted Result	各個執行緒的請求系統皆能正確且在 5 秒內回應
Cleanup	無

Identification	E-STC-034
Name	顯示交流列表功能系統測試
Tested Target	技能交換平台(Exchange)之交流頁面
Reference	E-FR-CM-006、E-FR-CM-007、E-FR-CM-008
Severity	1
Instructions	進入交流頁面

Excepted Result	3. 顯示所有交換狀態的技能 4. 顯示所有收到邀請的技能 5. 顯示所有送出邀請的技能
Cleanup	無

4 測試工作指派與時程(Personnel and Schedule)

4.1 測試成員(Personnel)

姓名	職責
吳宇鴻	單元測試規劃、單元測試人員
鍾子健	單元測試規劃、單元測試人員
施博文	系統測試規劃
羅祐任	系統測試人員
蔡昌廷	系統測試人員
全體成員	研發人員

4.2 測試程序與時程(Test Procedure and Schedule)

Deliverable	Responsibility	Completion Date
發展測試案例 (Develop Test Cases)	測試規劃人員	106/01/15
複核測試案例 (Test Case Review)	測試人員、研發人員	106/01/15
完成測試執行 (Execute Testing)	測試人員	106/01/15
完成缺失修復 (Complete Defect Reports)	研發人員	106/01/15
提出測試報告 (Final Test Summary Report)	測試人員	106/01/15

5 測試結果與分析(Test Results and Analysis)

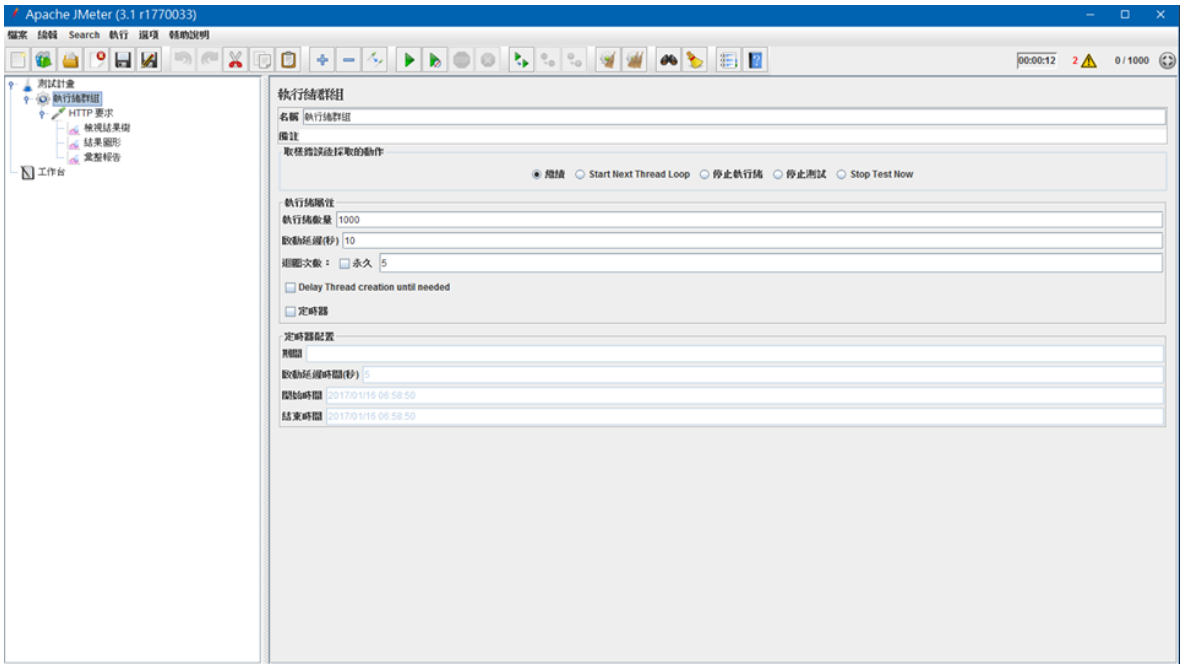
5.1 測試結果(Test Results)

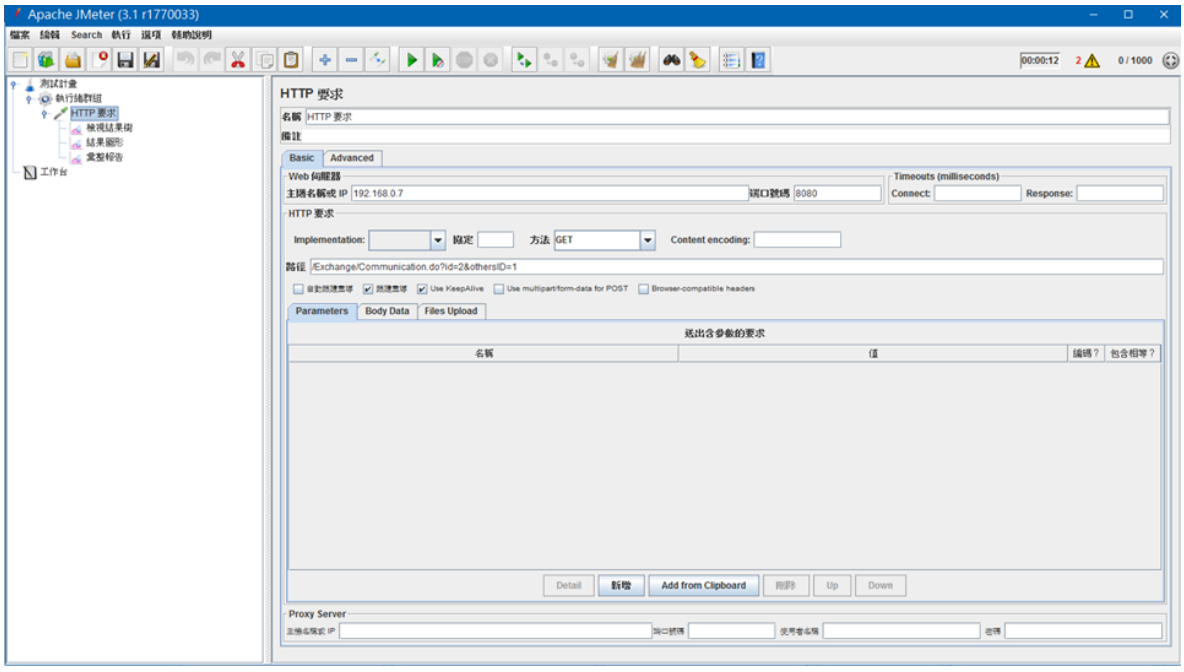
測試案例編號	測試結果 (Pass/Fail)	註解
E-STC-001	PASS	
E-STC-002	PASS	
E-STC-003	PASS	缺失編號：E-DT-002、E-DT-003
E-STC-004	PASS	缺失編號： E-DT-004、E-DT-005、E-DT-006
E-STC-005	PASS	
E-STC-006	PASS	缺失編號：E-DT-007
E-STC-007	PASS	
E-STC-008	PASS	
E-STC-009	PASS	缺失編號：E-DT-008
E-STC-010	FAIL	缺失編號：E-DT-009
E-STC-011	PASS	
E-STC-012	PASS	
E-STC-013	FAIL	缺失編號：E-DT-018
E-STC-014	PASS	缺失編號：E-DT-010
E-STC-015	PASS	缺失編號：E-DT-011
E-STC-016	PASS	
E-STC-017	PASS	缺失編號：E-DT-012
E-STC-018	PASS	
E-STC-019	PASS	缺失編號：E-DT-013
E-STC-020	PASS	缺失編號：E-DT-014
E-STC-021	PASS	
E-STC-022	PASS	
E-STC-023	PASS	
E-STC-024	PASS	缺失編號：E-DT-015、E-DT-017
E-STC-025	PASS	
E-STC-026	PASS	
E-STC-027	PASS	
E-STC-028	PASS	
E-STC-029	PASS	
E-STC-030	PASS	
E-STC-031	PASS	缺失編號：E-DT-016

E-STC-032	PASS	
E-STC-033	PASS	
E-STC-034	PASS	
RATE	94.18%	

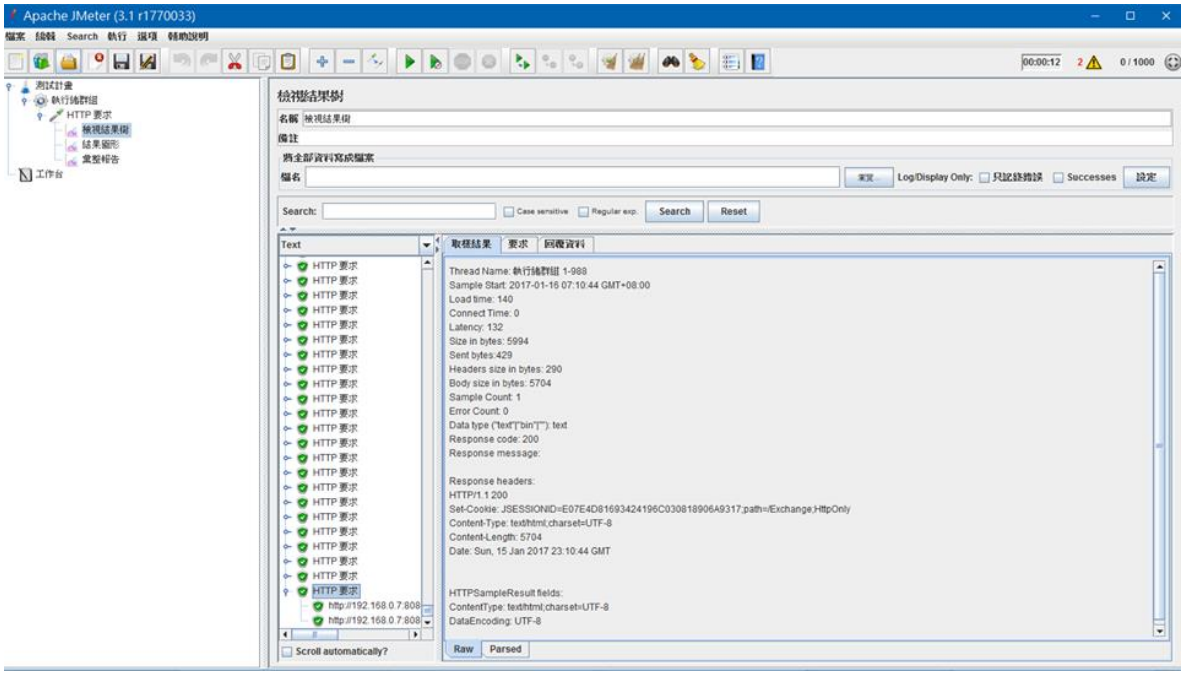
• E-STC-032 與 E-STC-033

設定執行緒數量為 1000(1000 > 100)

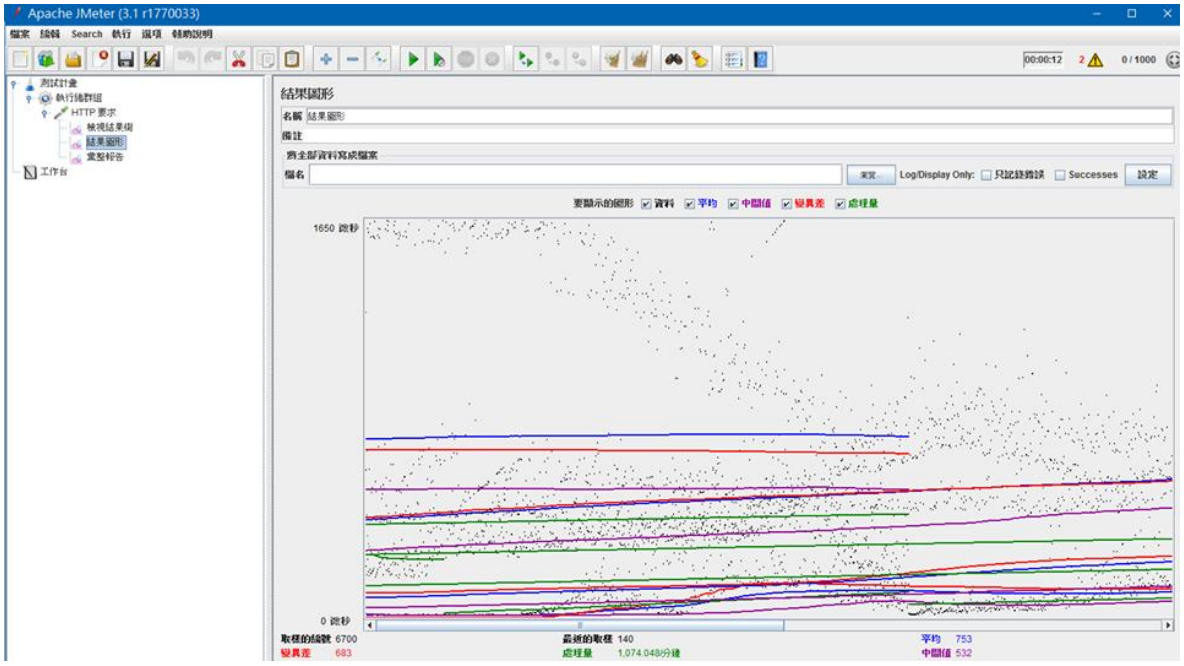


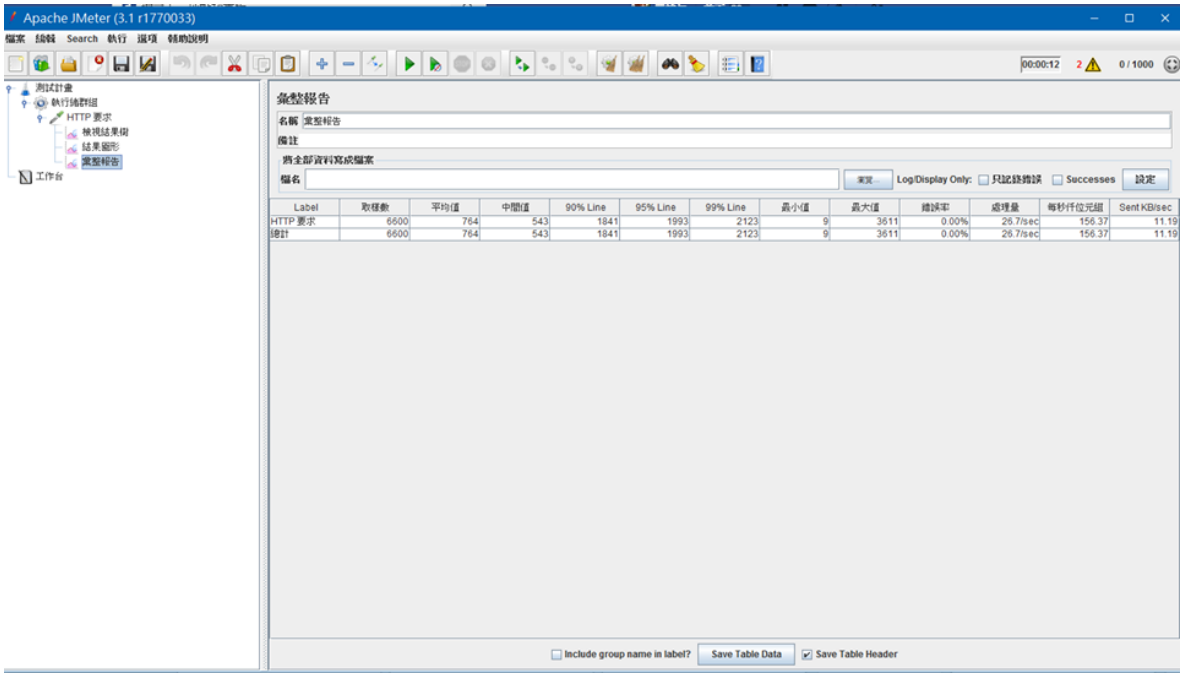


測試結果正確回傳



系統反應平均於 753 微秒





5.2 缺失報告(Defect Tracking)

缺失標號	缺失嚴重性	缺失說明	測試案例編號	缺失負責人	修復狀態	修復說明
E-DT-001	Low	系統的歡迎畫面無法正確導向(HTTP 404)	-	-	Open	-
E-DT-002	High	並非所有帳號皆能成功登入	E-STC-003	鍾子健	Closed	判斷帳號存不存在時，給資料庫錯誤的Query，使取出來的table不存在欲判斷的帳號。
E-DT-003	High	無法連結至特定的 Servlet	E-STC-003	蔡昌廷	Closed	網址路徑須為絕對路徑或相對路徑(開頭不用加上斜線)
E-DT-004	Low	只能正確顯示	E-STC-004	蔡昌廷	Closed	頁面加上

		首頁，但未顯示成功訊息				提示訊息
E-DT-005	High	Eclipse 無法啟動 Tomcat	E-STC-004	蔡昌廷	Closed	web.xml 需要加上一層 <element> 標籤
E-DT-006	High	整合註冊功能時產生 "JDBC Driver not found" 的錯誤訊息	E-STC-004	蔡昌廷	Closed	將 MySQLConnection.jar 放置於 Tomcat 的 lib 資料夾下
E-DT-007	Medium	功能無法使用	E-STC-006	蔡昌廷	Closed	完成整合
E-DT-008	Low	技能的總數未增加一筆	E-STC-009	蔡昌廷	Closed	補上函式呼叫
E-DT-009	Medium	功能無法使用	E-STC-010	蔡昌廷	Ongoing	尚未執行整合
E-DT-010	Medium	不能新增興趣技能	E-STC-014	蔡昌廷	Closed	因為新增興趣技能與刪除興趣技能具有重複的參數
E-DT-011	Medium	不能刪除興趣技能	E-STC-015	蔡昌廷	Closed	因為新增興趣技能與刪除興趣技能具有重複的參數
E-DT-012	High	無法取得下一項技能而產生 null exception	E-STC-017	蔡昌廷	Closed	未處理再配對直至沒有下一項技能的情況
E-DT-013	Medium	功能無法使用	E-STC-019	羅佑任	Closed	聊天模組已完成

E-DT-014	High	無法接受邀請	E-STC-020	蔡昌廷	Closed	doGet 語法錯誤，將網址串接的?寫錯
E-DT-015	High	對方無法評論	E-STC-024	蔡昌廷	Closed	原先設計資料庫僅有一個標籤，一人改動標籤後，另一人將無法再行評論，因此更改資料庫設計為兩個標籤
E-DT-016	Medium	功能無法使用	E-STC-031	羅佑任	Closed	聊天模組已完成
E-DT-017	Medium	評價完未更新技能的等級與帳戶的最大技能數量並判斷是否封鎖或警告	E-STC-031	蔡昌廷	Closed	補上函式呼叫
E-DT-018	Low	只能正確顯示個人頁面，但未顯示成功訊息	E-STC-013	蔡昌廷	Open	-
E-DT-019	Low	單元測試產生error(SQLException)	JUT-010 JUT-011	鍾子健	Closed	ResultSet.next() 未判斷空表格的情況，使用 if(rs.next()) 進行判斷是否從資料庫中取得資料
E-DT-020	Low	從資料庫取得的登入時間數	-	鍾子健	Closed	資料庫中時間的格

		值不符合預期				式無法直接封裝為 Java Date 物件，目前使用 String 物件儲存
E-DT-021	Medium	無法順利修改密碼	E-STC-006	蔡昌廷	Closed	該程式中的 jar 檔引用路徑錯誤
E-DT-022	Low	評價結束，交換對象的頁面仍具有聊天按鈕，並未提示對方須進行評價	E-STC-024	蔡昌廷	Closed	評價結束，將頁面中的按鈕刪除，並顯示需進行評價之提示

追溯表(Traceability Matrix)

Req. No.	Test Case #	Verification
E-FR-NB-001	E-STC-001	Verified
E-FR-NB-002	E-STC-002	Verified
E-FR-NB-003	E-STC-003	Verified
E-FR-SU-001	E-STC-004	Verified
E-FR-SGI-001	E-STC-005	Verified
E-FR-AM-001	E-STC-006	Verified
E-FR-SMG-001 E-FR-PMG-003	E-STC-007	Verified
E-FR-SMG-002	E-STC-008	Verified
E-FR- SMG -003 E-FR-SA-001	E-STC-009	Verified
E-FR- SMG -004 E-FR-SMF-001	E-STC-010	Verified
E-FR- SMG -005	E-STC-011	Verified
E-FR-PMG-001	E-STC-012	Verified
E-FR-PMG-002 E-FR-PMF-001	E-STC-013	Verified
E-FR-PMG-004 E-FR-FSA-001	E-STC-014	Verified
E-FR-PMG-005	E-STC-015	Verified
E-FR-SEG-001 E-FR-SEG-002	E-STC-016	Verified
E-FR-SEG-003	E-STC-017	Verified
E-FR-SEG-004	E-STC-018	Verified
E-FR-CM-001	E-STC-019	Verified
E-FR-CM-002	E-STC-020	Verified
E-FR-CM-003	E-STC-021	Verified

E-FR-CM-004	E-STC-022	Verified
E-FR-CM-005	E-STC-023	Verified
E-FR-CM-006 E-FR-CM-007 E-FR-CM-008	E-STC-034	Verified
E-FR-SEL-001 E-FR-SEL-002 E-FR-SEL-003	E-STC-024	Verified
E-FR-SEL-004	E-STC-025	Verified
E-FR-SIF-001 E-FR-SIF-002	E-STC-026	Verified
E-FR-PMF-002	E-STC-027	Verified
E-FR-FSA-002	E-STC-028	Verified
E-FR-SA-002	E-STC-029	Verified
E-FR-SMF-002	E-STC-030	Verified
E-FR-CP-001	E-STC-031	Verified
E-NFR-PR-001	E-STC-032	Verified
E-NFR-PR-002	E-STC-033	Verified