TEST SUMMARY REPORT

KAU Hospital System (KHS)
Independence Verification and Validation

Version : 1.0.0
Date : XX/XX/2024

Project Title: KHS Independence Verification and Validation		
Date: 6/04/2024	Test Summary Report ID: KHS_TSR_1_1.0.0	

Document Control

Document Name	KHS Test Summary Report
Reference Number	KHS_TSR_1
Version	1.0.0
Project Code	SAADA_KHS
Status	In_use
Date Released	15th of June, 2024

Name	Position	Signature
Prepared By: Adiba Alya	Test Manager	
Prepared By: Amylia Natasya	Test Analyst	
Reviewed By: Danisha Azra	Test Lead	
Approved By: Shahrul Amin	Test Automation Engineer	
Approved By: Nur Aisyah	Tester	

Project Title: KHS Independence Verification and Validation		
Date: 6/04/2024	Test Summary Report ID: KHS_TSR_1_1.0.0	

Version History

Version	Release Date	Section	Amendments
1.0.0	15/06/2024	All	Original Document

Distribution List

Version	Release Date	Controlled Copy No	Recipient Name	Department	Issue Date	Return Date
1.0.0	06/04/2024	01		Testerontop Test Team	06/04/2024	

Project Title: KHS Independence Verification and Validation		
Date: 6/04/2024	Test Summary Report ID: KHS_TSR_1_1.0.0	

Table of Content

Project Title: KHS Independence Verification and Validation		
Date: 6/04/2024	Test Summary Report ID: KHS_TSR_1_1.0.0	

Test Summary Report

1.0 Introduction

Purpose of Testing

The purpose of this testing is to verify that all system functions, such as adding a doctor, adding a nurse, adding a patient, adding medicine, and adding a room into the database, operate correctly and adhere to the defined constraints. Also to evaluate the system's capability to manage erroneous input effectively by providing informative error messages and preventing invalid data from being inserted into the database.

Objectives

i. To achieve a minimum of 80% code coverage through automated unit tests to ensure most of the codebase is exercised and potential issues are identified early in the development process.
ii. To ensure all features - add doctor, add nurse, add patient, add medicine, add room work well.
iii. To ensure all the errors can be handled by this system.

References

The following IEEE standards have been referenced in preparation of this document:

i. IEEE 829-2008 Standard for Software and System Test Documentation

The following documents provide the test basis for this test design:

i. KHS Software Requirement Specifications (KHS_SRS_1.0)
ii. KHS System Design Specifications (KHS_SDS_1.0)
iii. KHS System Design Specifications (KHS_SDS_1.0)

2.0 Testing Approach

Testing Techniques

i. Equivalence Partitioning
ii. Boundary Value Analysis
iii. Decision Table
iv. Use Case Testing

Testing Methodology

i. Stub and Driver

Tools

i. JUnit for automated testing to ensure consistency and repeatability.

3.0 Test Results

All critical function passed the unit test with 80% and above by using the number of test case

The status of the test are as follow:

Number of Test Case Planned to be Completed	Number of Test Case Remaining to be Executed	Number of Test Case Completed
145	0	145

Table 3.1 shows the status of test case

The Test Coverage is shown as below:

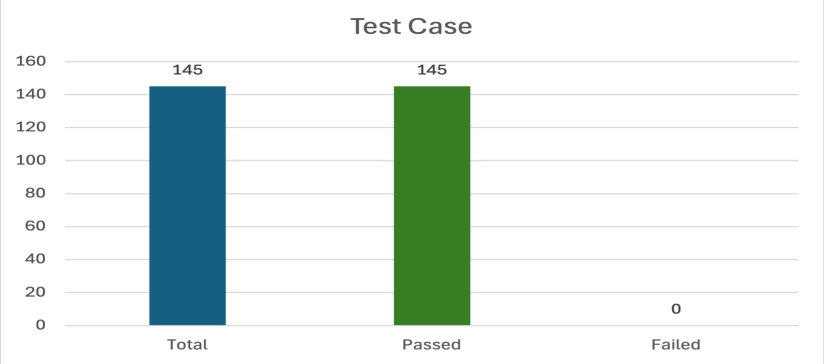

Figure 3.1 The graph shows the pass and fail test case

Coverage Information

We used stub and driver techniques. First we tested the driver which is add doctor, add nurse, add medicine, add room and add patient method by using JUnitClass and then we used stub method to test the main method by using console to test that coverage.

Driver

```
public static class Doctor {

    private int doctorId;
    private char gender;

    public Doctor(int doctorId, char gender) {
        setDoctorId(doctorId);
        setGender(gender);
    }

    public int getDoctorId() {
        return doctorId;
    }

    public char getGender() {
        return gender;
    }

    public void setDoctorId(int doctorId) {
        if (doctorId >= 5000 && doctorId <= 7999) {
            this.doctorId = doctorId;
        } else {
            throw new IllegalArgumentException("Invalid doctor ID");
        }
    }

    public void setGender(char gender) {
        if (gender == 'M' || gender == 'F') {
            this.gender = gender;
        } else {
            throw new IllegalArgumentException("Invalid gender");
        }
    }

    @Override
    public String toString() {
        return "\tDoctor ID: " + getDoctorId() + "\n\tGender: " + getGender();
    }
}
```

Figure 3.2 The coverage of add doctor

Coverage

DoctorTest (25 Jun 2024 2:06:50 pm)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TA	96.2 %	303	12	315
src	96.2 %	303	12	315
KauHospitalSystem	96.2 %	303	12	315
KAUHospitalSystem.java	84.4 %	65	12	77
DoctorTest.java	100.0 %	238	0	238

Figure 3.2 The percentage of coverage for add doctor

```
173
174     public static class Nurse {
175
176         private int nurseId;
177         private char gender;
178
179         public Nurse(int nurseId, char gender) {
180             setNurseId(nurseId);
181             setGender(gender);
182         }
183
184         // Getters
185         public int getNurseId() {
186             return nurseId;
187         }
188
189         public char getGender() {
190             return gender;
191         }
192
193         // Setters
194         public void setNurseId(int nurseId) {
195             if (nurseId >= 8000 && nurseId <= 9999) {
196                 this.nurseId = nurseId;
197             } else {
198                 throw new IllegalArgumentException("Invalid nurse ID");
199             }
200         }
201
202         public void setGender(char gender) {
203             if (gender == 'M' || gender == 'F') {
204                 this.gender = gender;
205             } else {
206                 throw new IllegalArgumentException("Invalid gender");
207             }
208         }
209
210         @Override
211         public String toString() {
212             return "\tNurse ID: " + getNurseId() + "\n\tGender: " + getGender();
213         }
214     }
```

Figure 3.3 The coverage of add nurse

Coverage

NurseTest (25 Jun 2024 2:09:17 pm)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TA	96.2 %	303	12	315
src	96.2 %	303	12	315
KauHospitalSystem	96.2 %	303	12	315
KAUHospitalSystem.java	84.4 %	65	12	77
NurseTest.java	100.0 %	238	0	238

Figure 3.4 The percentage of coverage for add nurse

```java
public static class Medicine {

    private int medicineCode;
    private double price;

    public Medicine(int medicineCode, double price) {
        setMedicineCode(medicineCode);
        setPrice(price);
    }

    // Getters
    public int getMedicineCode() {
        return medicineCode;
    }

    public double getPrice() {
        return price;
    }

    // Setters
    public void setMedicineCode(int medicineCode) {
        if (medicineCode >= 4000 && medicineCode <= 4514) {
            this.medicineCode = medicineCode;
        } else {
            throw new IllegalArgumentException("Invalid medicine code");
        }
    }

    public void setPrice(double price) {
        DecimalFormat df = new DecimalFormat("#.00");
        String formattedPrice = df.format(price);

        // Reject charges with letters or more than two decimal places
        if (!Pattern.matches("\\d+(\\.\\d{1,2})?", formattedPrice)) {
            throw new IllegalArgumentException("Invalid price");
        }

        double parsedPrice = Double.parseDouble(formattedPrice);
        if (parsedPrice < 1.00 || parsedPrice > 50.00) {
            throw new IllegalArgumentException("Invalid price");
        }

        this.price = parsedPrice;
    }

    @Override
    public String toString() {
        return "\tMedicine Code: " + getMedicineCode() + "\n\tPrice: " + getPrice();
    }
}
```

Figure 3.5 The coverage of add medicine

Coverage

MedicineTest (25 Jun 2024 1:56:02 pm)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TA	97.7 %	505	12	517
src	97.7 %	505	12	517
KauHospitalSystem	97.7 %	505	12	517
KAUHospitalSystem.java	87.9 %	87	12	99
MedicineTest.java	100.0 %	418	0	418

Figure 3.6 The percentage of coverage for add medicine

```java
public static class Room {

    private int roomNo;
    private double charges;

    public Room(int roomNo, double charges) {
        setRoomNo(roomNo);
        setCharges(charges);
    }

    // Getters
    public int getRoomNo() {
        return roomNo;
    }

    public double getCharges() {
        return charges;
    }

    // Setters
    public void setRoomNo(int roomNo) {
        if (roomNo >= 1 && roomNo <= 399) {
            this.roomNo = roomNo;
        } else {
            throw new IllegalArgumentException("Invalid room number");
        }
    }

    public void setCharges(double charges) {
        DecimalFormat df = new DecimalFormat("#.00");
        String formattedCharges = df.format(charges);

        // Reject charges with letters or more than two decimal places
        if (!Pattern.matches("\\d+(\\.\\d{1,2})?", formattedCharges)) {
            throw new IllegalArgumentException("Invalid charges");
        }

        double parsedCharges = Double.parseDouble(formattedCharges);
        if (parsedCharges < 100.0 || parsedCharges > 2000.0) {
            throw new IllegalArgumentException("Invalid charges");
        }

        this.charges = parsedCharges;
    }

    @Override
    public String toString() {
        return "\tRoom No: " + getRoomNo() + "\n\tCharges: " + getCharges();
    }
}
```

Figure 3.7 The coverage of add room

Coverage

RoomTest (25 Jun 2024 2:03:56 pm)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TA	96.7 %	500	17	517
src	96.7 %	500	17	517
KauHospitalSystem	96.7 %	500	17	517
KAUHospitalSystem.java	82.8 %	82	17	99
RoomTest.java	100.0 %	418	0	418

Figure 3.8 The percentage of coverage for add room

```
307    public static class Patient {
308
309        private int patientId;
310        private String illness;
311
312        public Patient(int patientId, String illness) {
313            setPatientId(patientId);
314            setIllness(illness);
315        }
316
317        // Getters
318        public int getPatientId() {
319            return patientId;
320        }
321
322        public String getIllness() {
323            return illness;
324        }
325
326        // Setters
327        public void setPatientId(int patientId) {
328            if (patientId >= 1000 && patientId <= 3999) {
329                this.patientId = patientId;
330            } else {
331                throw new IllegalArgumentException("Invalid patient ID");
332            }
333        }
334
335        public void setIllness(String illness) {
336            if (!illness.isEmpty()) {
337                this.illness = illness;
338            } else {
339                throw new IllegalArgumentException("Illness cannot be null");
340            }
341        }
342
343        @Override
344        public String toString() {
345            return "\tPatient ID: " + getPatientId() + "\n\tIllness: " + getIllness();
346        }
347    }
```

Figure 3.9 The coverage of add patient

Coverage

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TA	95.2 %	240	12	252
src	95.2 %	240	12	252
KauHospitalSystem	95.2 %	240	12	252
KAUHospitalSystem.java	83.8 %	62	12	74
PatientTest.java	100.0 %	178	0	178

Figure 3.10 The percentage of coverage for add patient

The coverage for each JUnit class is 100%. However, the KauHospitalSystem class does not achieve 100% coverage because we have not tested the main method, which is responsible for displaying and retrieving attributes to pass to it. Therefore, we test the main method.

Stub

```
KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 1
Enter room number (1-399): 3
Enter charges (100.0-2000.0): 200
Room successfully added

KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 2
Enter doctor ID (5000-7999): 5619
Enter gender (M/F): m
Doctor successfully added

KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 3
Enter nurse ID (8000-9999): 8910
Enter gender (M/F): F
Nurse successfully added

KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 4
Enter patient ID (1000-3999): 1234
Enter illness: fever
Patient successfully added

KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 5
Enter medicine code (4000-4514): 4510
Enter price (1.00-50.00): 20
Medicine successfully added
```

```
KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 6

Rooms:
        Room No: 0
        Charges: 0.0


Doctors:
        Doctor ID: 0
        Gender:


Nurses:
        Nurse ID: 0
        Gender:


Patients:
        Patient ID: 0
        Illness: null


Medicines:
        Medicine Code: 0
        Price: 0.0


KAU Hospital System Menu
1. Add Room
2. Add Doctor
3. Add Nurse
4. Add Patient
5. Add Medicine
6. Display All
7. Exit
Select an option: 7
Exiting...
```

3.11 The console to test stub

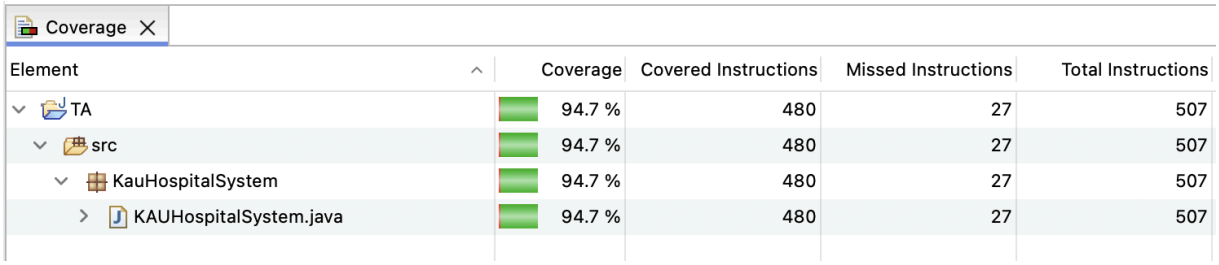

Figure 3.12 The percentage of coverage for stub

Next, we achieved 94.7% coverage in the KAUHospitalSystem class. The remaining percentage is due to untested exceptions. We tested this driver using the terminal.

For the Test Coverage completeness refer to the table below:

Test Documents	% Completed	Date of Completion
Test Plan	94.7%	06/04/2024
Test Design Specification	94.7%	29/04/2024
Test Case Specification	94.7%	28/05/2024
Test Procedure Specification	94.7%	09/06/2024
Test Log	94.7%	15/06/2024

Table 3.2 shows the Test Coverage completeness completeness

4.0 Recommendation

i. Enhance Test Coverage

Increase unit test coverage for critical modules to achieve 100% coverage. It is to make all the statements execute at least once before it passes to the client.

ii. Improve Performance

To shorten the testing period from 1 hour 30 minutes each function to 30 minutes

5.0 Conclusion

Result Summary

The item tested was the Session Verification feature in the KHS 1.1 (KAU Hospital System). The tests were conducted on a system provided by KAU Hospital.

There are a total of 145 test cases planned and executed, with 100% of the test cases passing successfully, achieving complete test coverage. No incidents were recorded during this iteration.

Since all test cases passed without any issues, it is expected that the product has fully conformed to the KHS Software Requirement Specifications, version 1.0. The Session Verification feature demonstrated excellent performance and stability, indicating readiness for the next phase of deployment.

Rationale for Decisions

Due to the absence of any defects at the end of the test iteration, the test team is confident that the Session Verification feature is ready for the next phase. Achieving 100% pass rate across all 145 test cases indicates that the feature meets all specified requirements and demonstrates robust performance and reliability. This thorough testing process ensures a low-risk product, ready for deployment.

Conclusion and Recommendation Based on Test Result

Based on the flawless performance of the Session Verification feature in KHS 1.1, with all 145 test cases passing without incident, the current version of the product is deemed fit for release. To sustain this success, maintain thorough testing practices and implement comprehensive logic checks in future updates to uphold system reliability.

Project Title: KHS Independent Verification and Validation		
Date:	Test Summary Report ID: KHS_TSR_1_1.0.0	

Appendix

General Information			
Test Log Scope	This Test Log covered Add Doctor (F001), Add Patient (F002), Add Medicine (F003), Add Nurse (F004), Add Room (F005) as described in Test Plan, KHS_TP_1.0.0		
Test Log Description	The items tested were the Add Doctor, Add Patient, Add Medicine, Add Nurse and Add Room feature in KHS 1.1. This test log records the execution of test procedure (KHS_TPS_1.0.0)		
Version Author	Tester 4, Tester 2	Contact Number	-
Revision Version	1.0	People Responsible	Tester 1
			Tester 2
			Tester 3
			Tester 4
			Tester 5
Activities Execution Information			
Execution Start Date	19/06/2024	End Date	19/06/2024
Execution Start Time		End Time	
Tester Name	Tester 4		
Participant	-		

Procedure Result								
Requirement ID	Test Design ID	Test Case ID	Test Procedure ID	Type of Testing	Tool	Pass/Fail	Test Incident ID	Remark
REQ_IO101 REQ_IO102 REQ_IO103 REQ_IO104 REQ_F101 REQ_F102	TDS2.2.1.1 TDS2.2.1.2 TDS2.2.1.3 TDS2.2.1.4	TC-01-001	TP-01-001	Functional	Manual	Pass	-	-
		TC-01-002				Pass	-	-
		TC-01-003				Pass	-	-
		TC-01-004				Pass	-	-
		TC-01-005				Pass	-	-
		TC-01-006				Pass	-	-
		TC-01-007				Pass	-	-
		TC-01-008				Pass	-	-
		TC-01-009				Pass	-	-
		TC-01-010				Pass	-	-
		TC-01-011				Pass	-	-

		TC-01-012				Pass	-	-
		TC-01-013				Pass	-	-
		TC-01-014				Pass	-	-
		TC-01-015				Pass	-	-
		TC-01-016				Pass	-	-
		TC-01-017				Pass	-	-
		TC-01-018				Pass	-	-
		TC-01-019				Pass	-	-
		TC-01-020				Pass	-	-
		TC-01-021				Pass	-	-
		TC-01-022				Pass	-	-
		TC-01-023				Pass	-	-
REQ_IO201 REQ_IO202 REQ_IO203 REQ_IO204 REQ_F201	TDS2.2.2.1 TDS2.2.2.2 TDS2.2.2.3 TDS2.2.2.4	TC-02-001	TP-02-001	Functional	Manual	Pass	-	-
		TC-02-002				Pass	-	-
		TC-02-003				Pass	-	-

REQ_F202		TC-02-004				Pass	-	-
		TC-02-005				Pass	-	-
		TC-02-006				Pass	-	-
		TC-02-007				Pass	-	-
		TC-02-008				Pass	-	-
		TC-02-009				Pass	-	-
		TC-02-010				Pass	-	-
		TC-02-011				Pass	-	-
		TC-02-012				Pass	-	-
		TC-02-013				Pass	-	-
		TC-02-014				Pass	-	-
		TC-02-015				Pass	-	-
		TC-02-016				Pass	-	-
		TC-02-017				Pass	-	-
REQ_IO301 REQ_IO302	TDS2.2.3.1 TDS2.2.3.2	TC-03-001	TP-03-001	Functional	Manual	Pass	-	-

REQ_IO303 REQ_IO304 REQ_F301 REQ_F302	TDS2.2.3.3 TDS2.2.3.4	TC-03-002				Pass	-	-
		TC-03-003				Pass	-	-
		TC-03-004				Pass	-	-
		TC-03-005				Pass	-	-
		TC-03-006				Pass	-	-
		TC-03-007				Pass	-	-
		TC-03-008				Pass	-	-
		TC-03-009				Pass	-	-
		TC-03-010				Pass	-	-
		TC-03-011				Pass	-	-
		TC-03-012				Pass	-	-
		TC-03-013				Pass	-	-
		TC-03-014				Pass	-	-
		TC-03-015				Pass	-	-
		TC-03-016				Pass	-	-

		TC-03-017				Pass	-	-
		TC-03-018				Pass	-	-
		TC-03-019				Pass	-	-
		TC-03-020				Pass	-	-
		TC-03-021				Pass	-	-
		TC-03-022				Pass	-	-
		TC-03-023				Pass	-	-
		TC-03-024				Pass	-	-
		TC-03-025				Pass	-	-
		TC-03-026				Pass	-	-
		TC-03-027				Pass	-	-
		TC-03-028				Pass	-	-
		TC-03-029				Pass	-	-
		TC-03-030				Pass	-	-
		TC-03-031				Pass	-	-

		TC-03-032				Pass	-	-
		TC-03-033				Pass	-	-
		TC-03-034				Pass	-	-
		TC-03-035				Pass	-	-
		TC-03-036				Pass	-	-
		TC-03-037				Pass	-	-
		TC-03-038				Pass	-	-
		TC-03-039				Pass	-	-
		TC-03-040				Pass	-	-
		TC-03-041				Pass	-	-
REQ_IO401 REQ_IO402 REQ_IO403 REQ_IO404 REQ_F401 REQ_F402	TDS2.2.4.1 TDS2.2.4.2 TDS2.2.4.3 TDS2.2.4.4	TC-04-001	TP-04-001	Functional	Manual	Pass	-	-
		TC-04-002				Pass	-	-
		TC-04-003				Pass	-	-
		TC-04-004				Pass	-	-
		TC-04-005				Pass	-	-

		TC-04-006				Pass	-	-
		TC-04-007				Pass	-	-
		TC-04-008				Pass	-	-
		TC-04-009				Pass	-	-
		TC-04-010				Pass	-	-
		TC-04-011				Pass	-	-
		TC-04-012				Pass	-	-
		TC-04-013				Pass	-	-
		TC-04-014				Pass	-	-
		TC-04-015				Pass	-	-
		TC-04-016				Pass	-	-
		TC-04-017				Pass	-	-
		TC-04-018				Pass	-	-
		TC-04-019				Pass	-	-
		TC-04-020				Pass	-	-

		TC-04-021				Pass	-	-
		TC-04-022				Pass	-	-
		TC-04-023				Pass	-	-
REQ_IO501 REQ_IO502 REQ_IO503 REQ_IO504 REQ_F501 REQ_F502	TDS2.2.5.1 TDS2.2.5.2 TDS2.2.5.3 TDS2.2.5.4	TC-05-001	TP-05-001	Functional	Manual	Pass	-	-
		TC-05-002				Pass	-	-
		TC-05-003				Pass	-	-
		TC-05-004				Pass	-	-
		TC-05-005				Pass	-	-
		TC-05-006				Pass	-	-
		TC-05-007				Pass	-	-
		TC-05-008				Pass	-	-
		TC-05-009				Pass	-	-
		TC-05-010				Pass	-	-
		TC-05-011				Pass	-	-
		TC-05-012				Pass	-	-

		TC-05-013				Pass	-	-
		TC-05-014				Pass	-	-
		TC-05-015				Pass	-	-
		TC-05-016				Pass	-	-
		TC-05-017				Pass	-	-
		TC-05-018				Pass	-	-
		TC-05-019				Pass	-	-
		TC-05-020				Pass	-	-
		TC-05-021				Pass	-	-
		TC-05-022				Pass	-	-
		TC-05-023				Pass	-	-
		TC-05-024				Pass	-	-
		TC-05-025				Pass	-	-
		TC-05-026				Pass	-	-
		TC-05-027				Pass	-	-

		TC-05-028				Pass	-	-
		TC-05-029				Pass	-	-
		TC-05-030				Pass	-	-
		TC-05-031				Pass	-	-
		TC-05-032				Pass	-	-
		TC-05-033				Pass	-	-
		TC-05-034				Pass	-	-
		TC-05-035				Pass	-	-
		TC-05-036				Pass	-	-
		TC-05-037				Pass	-	-
		TC-05-038				Pass	-	-
		TC-05-039				Pass	-	-
		TC-05-040				Pass	-	-
		TC-05-041				Pass	-	-