NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2006 - Software Engineering
Lab 4 Deliverables

Lab Group	SDAD
Team	HomeGoWhere
Members	Lim Jing Rong (U2323260D)
	Loo Ping Wee (U2321814J)
	Goh Shuen Wei (U2321093D)
	Ng Jing En (U2321510K)
	Babu Sankar Nithin Sankar (U2323953B)
	Lau Zhan You (U2320608L)

1. Black Box Testing

1.1. Selected Control Class

The control class we have selected for testing will be the **AuthController**.
The AuthController (Authentication Controller) manages user authentication for the application, including user sign up and user login.

During the user sign up process, the user has to input their email, password and full name. After a successful sign up process, the user's details, as well as the hashed password, will be stored in the database. The user can then login to the application with the same email and password used during the sign up process.

During the user login process, the user has to input their email and password that they used during the sign up process. If the email matches an existing user, and the hashed password input matches the hashed password stored in the database, the user will be authenticated and directed to the home screen.

1.2. Equivalence Class Testing

The login and signup process requires discrete values as inputs. As such, Boundary Value Testing will not be applicable

- Login Function
    - **Valid Equivalence Class:** Username and password input values are in correct formats and match an existing user.
    - **Invalid Equivalence Class:** Username and password input values are in incorrect formats or missing or do not match an existing user.

- Sign Up Function
    - **Valid Equivalence Class:** All required fields (Email, Password and Full Name) are correctly filled and email address does not match an existing user.
    - Invalid Equivalence Class: Missing fields (Email, Password and Full Name), invalid email format, or user with the same email address already exists.

1.3. Test Cases and Testing Results

1.3.1. Login

Input Parameters:

1) Email
2) Password

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
Login-01	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword”	Login Success	Login Success	Pass
Login-02	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “wrongpassword”	Login Failed: “Incorrect password”	Login Failed: “Incorrect password”	Pass
Login-03	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “”	Login Failed: “Password is required”	Login Failed: “Password is required”	Pass
Login-04	**(Invalid)** Email: “fakeuser@gmail.com” **(Valid)** Password: “testpassword”	Login Failed: “Email not registered”	Login Failed: “Email not registered”	Pass
Login-05	**(Invalid)** Email: “” **(Valid)** Password: “testpassword”	Login Failed: “Email is required”	Login Failed: “Email is required”	Pass
Login-06	**(Invalid)** Email: “testuseratgmaildotcom” **(Valid)** Password: “testpassword”	Login Failed: “Email format is invalid”	Login Failed: “Email format is invalid”	Pass

1.3.2. Sign Up

Input Parameters:

1) Email
2) Password
3) Full Name

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
Signup-01	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Success	Sign up Success	Pass
Signup-02	**(Invalid)** Email: “repeateduser@gmail.com” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email already exists.”	Sign up Failed: “Email already exists.”	Pass
Signup-03	**(Invalid)** Email: “” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email is required.”	Sign up Failed: “Email is required.”	Pass
Signup-04	**(Invalid)** Email: “testuseratgmaildotcom” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email format is invalid.”	Sign up Failed: “Email format is invalid.”	Pass

Signup-05	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Password is required.”	Sign up Failed: “Password is required.”	Pass
Signup-06	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “1234” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Password must be at least 6 characters long.”	Sign up Failed: “Password must be at least 6 characters long.”	Pass
Signup-07	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword” **(Invalid)** Full Name: “”	Sign up Failed: “Full Name is required.”	Sign up Failed: “Full Name is required.”	Pass

2. White Box Testing

The 2 methods selected for white box testing will be:

1) SubmitReview
2) MakeTerminationRequest

The **SubmitReview** method allows users to leave reviews for the other user they interacted with during a rental.

This includes 2 scenarios:

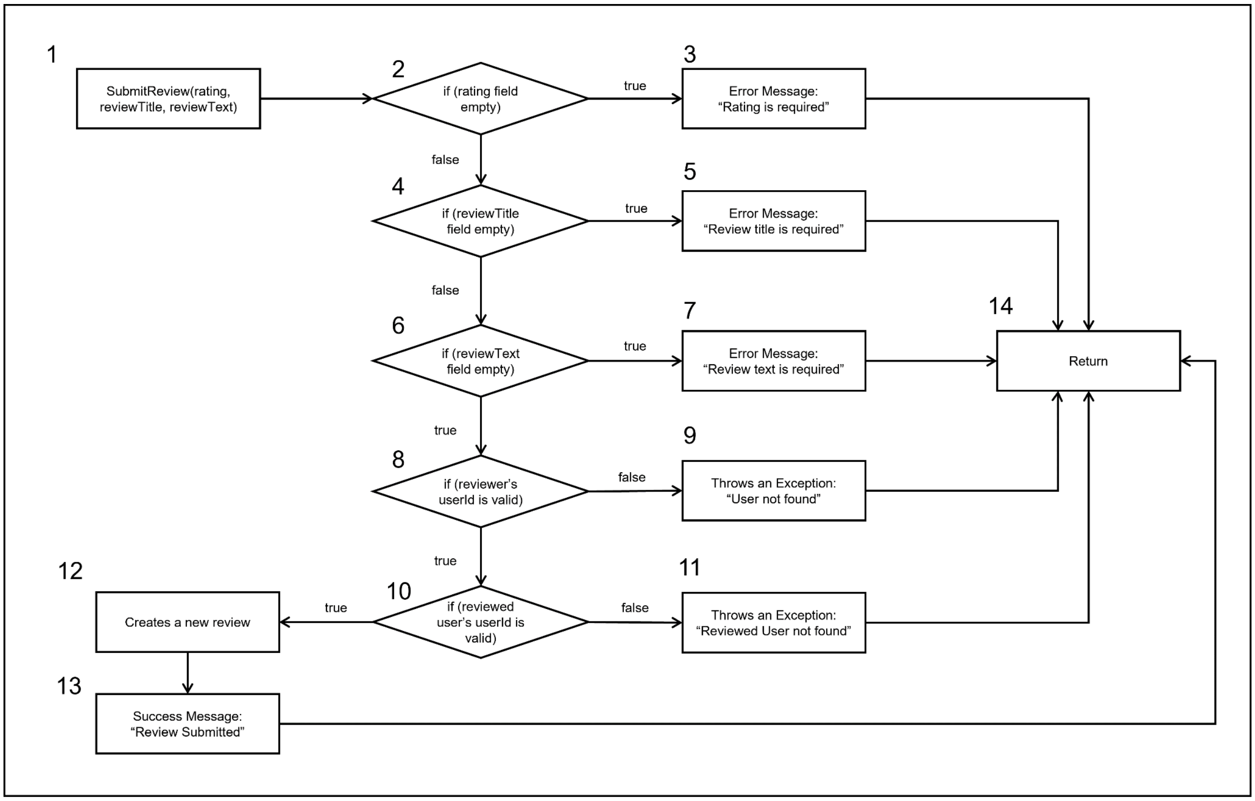
1) Owner leaves review for tenant
2) Tenant leaves review for owner

During the SubmitReview process, the user enters a rating, review title and review text.

The **MakeTerminationRequest** method allows an owner to create a request to terminate a rental contract prematurely, before the actual lease end date specified during the MakeRentalOffer process. During this process, the owner has to enter an amount to refund to the tenant, intended to allow the owner to refund the rental deposit collected earlier. Upon successful creation of the termination request, an automated chat message will be sent to the tenant to approve or deny the request. The termination process will then vary based on the input by the tenant.

2.1. SubmitReview

2.1.1. Control Flow Graph


1
SubmitReview(rating, reviewTitle, reviewText)
2
if (rating field empty)
true
3
Error Message: "Rating is required"
false
4
if (reviewTitle field empty)
true
5
Error Message: "Review title is required"
false
6
if (reviewText field empty)
true
7
Error Message: "Review text is required"
14
Return
true
8
if (reviewer's userId is valid)
false
9
Throws an Exception: "User not found"
true
10
if (reviewed user's userId is valid)
true
12
Creates a new review
false
11
Throws an Exception: "Reviewed User not found"
13
Success Message: "Review Submitted"

2.1.2. Cyclomatic Complexity

Cyclomatic Complexity (CC) = | binarydecisionpoint | + 1 = | 5 | + 1 = 6

2.1.3. Basis Paths

Basis Path #1 (Baseline): 1,2,4,6,8,10,12,13,14
Basis Path #2: 1,2,3,14
Basis Path #3: 1,2,4,5,14
Basis Path #4: 1,2,4,6,7,14
Basis Path #5: 1,2,4,6,8,9,14
Basis Path #6: 1,2,4,6,8,10,11,14

2.1.4. Test Cases and Testing Results

Input Parameters:

1) rating
2) reviewTitle
3) reviewText
4) Reviewed user’s userId (Automatically passed in by referring to the profile page the reviewer navigated from)
5) Reviewer’s userId (Automatically passed in by referring to the currently logged in user’s userId)

Test Case Name	Test Input	Expected Output	Actual Output	Test Result

Review-01	rating = 5
	reviewTitle = “Test Review”
	reviewText = “This is a test.”
	
	revieweeId = 1
	reviewerId = 2
		Success Message: “Review Submitted”	Success Message: “Review Submitted”	Pass

Review-02	rating =
	reviewTitle = “Test Review”
	reviewText = “This is a test.”
	
	revieweeId = 1
	reviewerId = 2
		Error Message: “Rating is required”	Error Message: “Rating is required”	Pass

Review-03	rating = 5
	reviewTitle = “”
	reviewText = “This is a test.”
	
	revieweeId = 1
	reviewerId = 2
		Error Message: “Review title is required”	Error Message: “Review title is required”	Pass

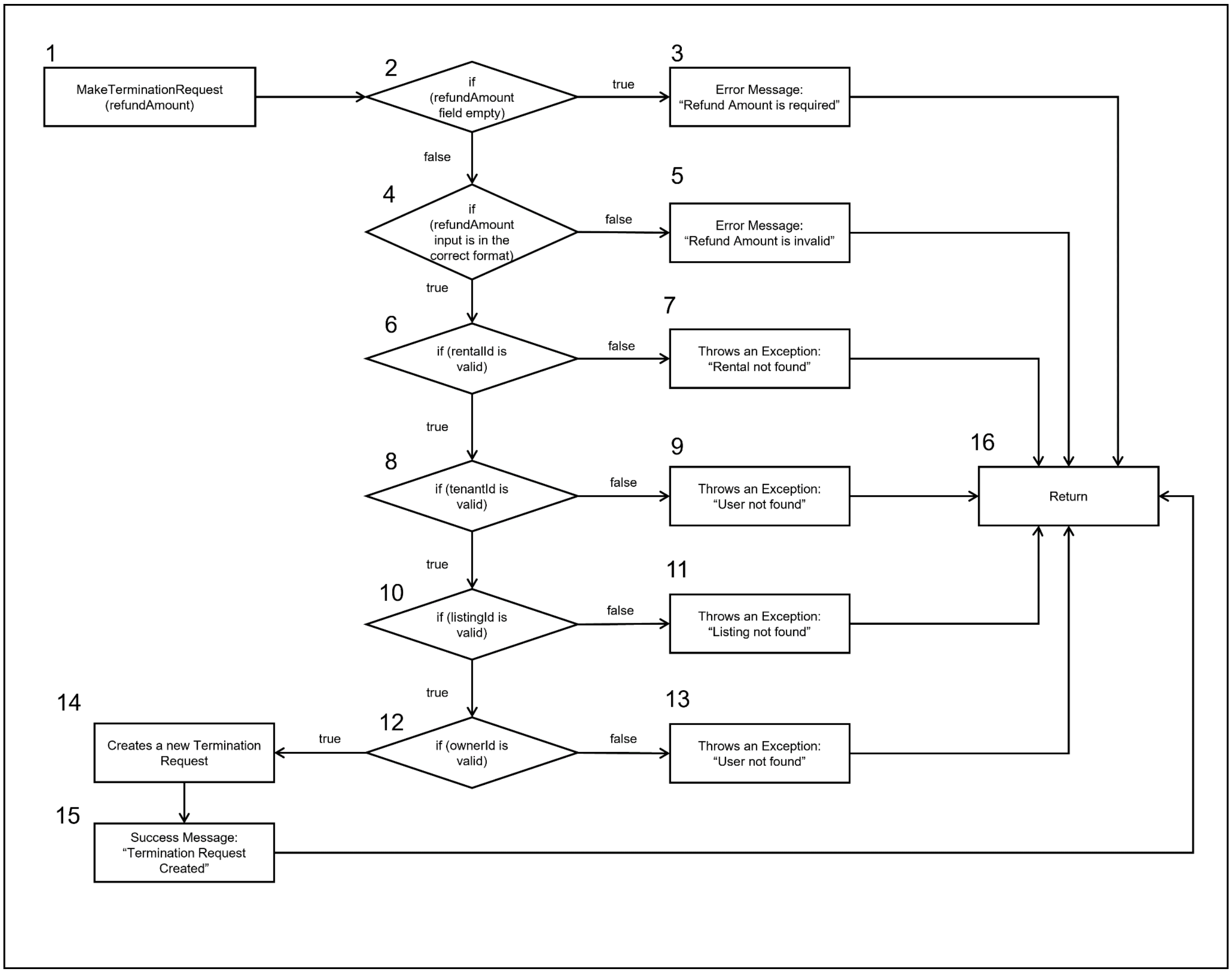
Review-04	rating = 5
	reviewTitle = “Test Review”
	reviewText = “”
	
	revieweeId = 1
	reviewerId = 2
		Error Message: “Review text is required”	Error Message: “Review text is required”	Pass

Review-05	rating = 5
	reviewTitle = “Test Review”
	reviewText = “This is a test.”
	revieweeId = 1
	reviewerId = -1
		Error Message: “User not found”	Error Message: “User not found”	Pass

Review-06	rating = 5
	reviewTitle = “Test Review”
	reviewText = “This is a test.”
	revieweeId = -1
	reviewerId = 2
		Error Message: “Reviewed user not found”	Error Message: “Reviewed user not found”	Pass

2.2. MakeTerminationRequest

2.2.1. Control Flow Graph


1
MakeTerminationRequest (refundAmount)
2
if (refundAmount field empty)
true
3
Error Message: "Refund Amount is required"
false
4
if (refundAmount input is in the correct format)
false
5
Error Message: "Refund Amount is invalid"
true
6
if (rentalId is valid)
false
7
Throws an Exception: "Rental not found"
true
8
if (tenantId is valid)
false
9
Throws an Exception: "User not found"
16
Return
true
10
if (listingId is valid)
false
11
Throws an Exception: "Listing not found"
true
12
if (ownerId is valid)
true
14
Creates a new Termination Request
false
13
Throws an Exception: "User not found"
15
Success Message: "Termination Request Created"

2.2.2. Cyclomatic Complexity

Cyclomatic Complexity (CC) = | binarydecisionpoint | + 1 = | 6 | + 1 = 7

2.2.3. Basis Paths

Basis Path #1 (Baseline): 1,2,4,6,8,10,12,14,15,16
Basis Path #2: 1,2,3,16
Basis Path #3: 1,2,4,5,16
Basis Path #4: 1,2,4,6,7,16
Basis Path #5: 1,2,4,6,8,9,16
Basis Path #6: 1,2,4,6,8,10,11,16
Basis Path #7: 1,2,4,6,8,10,12,13,16

2.2.4. Test Cases and Testing Results

Input Parameters:

1) refundAmount
2) rentalId (Automatically passed in by referring to the rental page the owner clicked the "Make Termination Request" button on)
3) tenantId (Automatically passed in by referring to tenantId stored by the specific rental retrieved)
4) listingId (Automatically passed in by referring to listingId stored by the specific rental retrieved)
5) ownerId (Automatically passed in by referring to ownerId stored by the specific listing retrieved)

Test Case Name	Test Input	Expected Output	Actual Output	Test Result

request-01	refundAmount = 1000
	rentalId = 1
	tenantId = 1
	listingId = 1
	ownerId = 2
		Success Message: "Termination Request Created"	Success Message: "Termination Request Created"	Pass

request-02	refundAmount =
	rentalId = 1
	tenantId = 1
	listingId = 1
	ownerId = 2
		Error Message: "Refund Amount is required"	Error Message: "Refund Amount is required"	Pass

request-03	refundAmount = 1000.0.00.0
	rentalId = 1
	tenantId = 1
	listingId = 1
	ownerId = 2
		Error Message: "Refund Amount is invalid"	Error Message: "Refund Amount is invalid"	Pass

request-04	refundAmount = 1000
	rentalId = -1
	tenantId = 1
	listingId = 1
		Error Message: "Rental not found"	Error Message: "Rental not found"	Pass

	ownerId = 2			

request-05	refundAmount = 1000
	rentalId = 1
	tenantId = -1
	listingId = 1
	ownerId = 2
		Error Message: “User not found”	Error Message: “User not found”	Pass

request-06	refundAmount = 1000
	rentalId = 1
	tenantId = 1
	listingId = -1
	ownerId = 2
		Error Message: “Listing not found”	Error Message: “Listing not found”	Pass

request-07	refundAmount = 1000
	rentalId = 1
	tenantId = 1
	listingId = 1
	ownerId = -1
		Error Message: “User not found”	Error Message: “User not found”	Pass