AIR University Islamabad

HomeToHome

(A Service Exchange Platform)

Group Members

1. Afia Aziz BSCS-F-23-B-231561
2. Arooba BSCS-F-23-B-231678
3. Zumer Dhillun BSCS-F-23-B-231597

Instructor

Miss Sara Ibrahim

Course Name & Code: Software Engineering (SE-100)
Semester: Spring 2025

1. Risk Management:

a. Risk Identification

Risk ID	Risk Type	Risk Description
R1	Organizational Risk	Limited team capacity or delay in task distribution may delay feature development.
R2	Technology Risk	Lack of two factor authentication system could result in security vulnerabilities.
R3	Requirements Risk	Scope changes like adding real-time chat may require rework and disrupt timelines.
R4	Estimation Risk	Underestimating time and resources for service booking integration may cause delays.
R5	Business Risk	Poor marketing efforts or delayed launch may lead to low user adoption.

b. Risk Indicators and Mitigation Plan

Risk ID	Indicators	Mitigation Plan
R1	Missed internal deadlines; incomplete tasks in project plan	**Preventive:** Assign clear roles and use a shared task board. **Contingency:** Focus on core features and reduce scope if needed.
R2	Multiple login attempts; same user logged in from different devices	**Preventive**: Implement session timeouts and login attempt limits. **Contingency:** Add basic device logging and plan 2FA for future phase
R3	Last-minute feature requests from team or stakeholders	**Preventive:** Freeze feature scope before development; conduct weekly scope reviews. **Contingency:** Allocate a refactoring sprint.
R4	Testing and debugging take longer than expected	**Preventive:** Follow a modular design and perform incremental testing. **Contingency:** Allocate extra time buffer before final submission.
R5	No engagement during demo or feedback rounds; poor online visibility	**Preventive:** Plan a soft launch among peers and create awareness via digital platforms. **Contingency:** Release a short promo video and conduct a mini social media campaign.

2. White Box and Black Box Testing:

a. Understanding Black Box and White Box Testing:

White Box Testing

Definition:
White box testing (also known as **structural**, **glass box**, or **clear box** testing) involves testing the **internal logic and structure** of the code. The tester must have knowledge of the source code and understands how the system works internally.

Focus Areas:

- Code paths
- Conditional statements (if, switch)
- Loops (for, while)
- Function calls
- Error handling

Example: Testing whether all branches of an `if-else` condition are executed properly.

Black Box Testing

Definition:
Black box testing focuses on testing the software from the **end-user's perspective** without any knowledge of the internal code. It evaluates the system based on inputs and expected outputs.

Focus Areas:

- Functional requirements
- Input-output validation
- User interface behavior
- Error messages and boundary conditions

Example: Testing a login form by entering correct and incorrect credentials to see the system response.

Importance in the Software Development Lifecycle (SDLC)

- **Quality Assurance**
  Both white box and black box testing ensure that the software works as expected. White box testing validates the internal logic and code structure, while black box testing verifies the system's behavior from an end-user perspective. Together, they help maintain overall quality.
- **Early Defect Detection**
  White box testing allows developers to catch bugs, logical errors, and faulty control flows early during development. This proactive approach ensures issues are resolved before they become major problems.
- **User Satisfaction**
  Black box testing focuses on the user's perspective. By validating functionality, usability, and expected output, it ensures that the application meets the users' needs and performs correctly in real-world usage.
- **Comprehensive Coverage**
  When both testing types are combined, they offer thorough test coverage—white box for backend logic and black box for frontend functionality. This helps in uncovering hidden defects and improves system reliability.
- **Cost Reduction**
  Detecting and fixing defects early (especially during white box testing) significantly reduces the cost and effort of rework during later stages such as testing or deployment.

Conclusion:
Both **white box** and **black box** testing are essential components of the software testing process. Together, they improve the overall **reliability**, **performance**, and **usability** of the system, making them vital in delivering high-quality software in any SDLC model.

b. Scenarios and Test Cases:

Test Scenario 1: User Login

Test Scenario ID	TS01	**Test Case ID**	TC01-01, TC01-02, TC01-03
Test Case Description	Validate login functionality with various inputs	**Test Priority**	High
Pre-Requisite	User must be registered	**Post-Requisite**	Redirect to dashboard if successful

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Enter valid email & password	user@gmail.com / password123	User is logged in	User is redirected to dashboard	//	Pass	//
2	Enter wrong password	user@gmail.com / wrong password	Error message displayed	Error message shown	//	Pass	//
3	Submit empty form	"" / ""	Prompt to fill fields	Validation messages shown	//	Pass	//

Test Scenario 2: Search and Book a Service

Test Scenario ID	TS02	**Test Case ID**	TC02-01, TC02-02, TC02-03
Test Case Description	Validate service search and booking flow with valid/invalid inputs and system behavior	**Test Priority**	High
Pre-Requisite	User must be logged in and at least one worker must exist in the system	**Post-Requisite**	Service request is submitted and validated

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Search and book a service successfully	Search Service: "Plumbing"	Matching workers shown Booking form submitted Success message	All steps worked as expected	//	Pass	//
2	Leave the search bar empty and press “Search Icon”	"" (Blank input)	**Prompt:** "Please enter a service to search" OR no results shown	All worker cards shown	//	Fail	System should not show all workers without a specific search
3	Select a past date in booking form	Try choosing a date before today (e.g., "2024-12-01")	Date is not clickable; appears dimmed	Date is visually disabled and unclick able	//	Pass	//

Test Scenario 3: Contact Us Message

Test Scenario ID	TS03	**Test Case ID**	TC03-01, TC03-02, TC03-03
Test Case Description	Validate Contact Us form submission	**Test Priority**	Medium
Pre-Requisite	Contact Us page should be accessible	**Post-Requisite**	Data saved in database and confirmation shown

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fill all fields correctly	Name, Email, Message	Message sent	Message sent successfully	//	Pass	//
2	Submit empty form	None	Required field errors shown	Error messages shown	//	Pass	//
3	Enter invalid email format	"John", "abc@", "Great!"	Error on email field	Email format error shown	//	Pass	//

Test Scenario 4: Worker-User Booking Request Workflow

Test Scenario ID	TS04	**Test Case ID**	TC04-01, TC04-02, TC04-03
Test Case Description	Synchronization and actions for Booking Requests from Customer and Worker side	**Test Priority**	High
Pre-Requisite	User is logged in and submits a request; Worker is registered	**Post-Requisite**	Request status updates on both ends in real-time

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	User submits service request to worker	Select worker, valid form	Request appears in worker's dashboard	Worker sees new pending request	//	Pass	//
2	Worker accepts the request	Click "Accept" on request	Status updated to "Accepted" on both sides	Status reflects "Accepted" on both User and worker panel	//	Pass	//
3	Worker rejects request	Click "Reject"	Status on user side changes to "Rejected"	Rejection visible to user	//	Pass	//

Test Scenario 5: Review Requests by User

Test Scenario ID	TS05	**Test Case ID**	TC05-01, TC05-02, TC05-03
Test Case Description	Validate user ability to view, edit, cancel, and track service request statuses from dashboard	**Test Priority**	High
Pre-Requisite	User has submitted at least one request	**Post-Requisite**	Real-time updates visible to both user and worker after edit/cancel

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	View request status	Open Dashboard	Requests listed with correct status	Statuses match current state	//	Pass	//
2	Edit a request	Click Edit Change Some Data Input Submit	Changes reflected in both user and worker dashboards	Updated info visible immediately	//	Pass	//
3	Cancel a request	Click "Cancel" on an active request	Status becomes "Cancelled" on both sides	Request marked as Cancelled in worker view	//	Pass	//

3. Requirements Traceability Matrix:

Requirements Traceability Matrix:

REQUIREMENT TRACEABILITY	**BUSINESS REQUIREMENTS**				
TEST CASES	**BRD001** (Login)	**BRD002** (Search & Book)	**BRD003** (Contact Us)	**BRD004** (Booking Workflow)	**BRD005** (Review Requests)
TC01-01	✔				
TC01-02	✔				
TC01-03	✔				
TC02-01		✔			
TC02-02		✔			
TC02-03		✔			
TC03-01			✔		
TC03-02			✔		
TC03-03			✔		
TC04-01				✔	
TC04-02				✔	
TC04-03				✔	
TC05-01					✔
TC05-02					✔
TC05-03					✔