1. BugServiceImpl

TestCaseID	Priority	Preconditions	Input Test Data (Function Parameters)	Steps to be Executed	Expected Results	Actual Results
Bug_TC_01	High	BugDao is initialized	createBug(1, "NullPointerException", "Null reference", "High", 101)	1. Call createBug() with valid inputs. 2. Assert the returned bug ID is correct.	Bug ID is returned and bug is created successfully in the system.	As Expected
Bug_TC_02	High	A bug with ID 1 exists in the system	getBugById(1)	1. Call getBugById() with an existing bug ID. 2. Assert that the correct Bug object is returned.	The correct Bug object corresponding to ID 1 is returned.	As Expected
Bug_TC_03	High	Bug with name "NullPointerException" exists	getBugByName("NullPointerException")	1. Call getBugByName() with a valid bug name. 2. Assert that the correct Bug object is returned.	The correct Bug object corresponding to the name is returned.	As Expected
Bug_TC_04	High	Bug with ID 1 exists and is not closed	closeBug(1)	1. Call closeBug() with an existing and open bug ID. 2. Assert that the bug status is updated to "Closed".	The bug is marked as closed in the system.	As Expecte d

2. EmployeeServiceImpl

TestCaseID	Priority	Preconditions	Input Test Data (Function Parameters)	Steps to be Executed	Expected Results	Actual Results
Emp_TC_01	High	EmployeeDao is initialized	createEmployee("John Doe", "john.doe@example.com", "Developer")	1. Call createEmployee() with valid inputs. 2. Assert that the returned employee ID is correct.	Employee ID is returned and the employee is successfully created in the system.	As Expected
Emp_TC_02	High	Employee with ID 1 exists	getEmployeeById(1)	1. Call getEmployeeById() with an existing employee ID. 2. Assert that the correct Employee object is returned.	The correct Employee object corresponding to ID 1 is returned.	As Expected
Emp_TC_03	High	Employee with ID 1 exists	updateEmployeeEmail(1, "new.email@example.com")	1. Call updateEmployeeEmail() with valid inputs. 2. Assert that the email is updated correctly.	The email of the employee with ID 1 is updated in the system.	As Expected
Emp_TC_04	High	Employee with ID 1 exists	updateEmployeeRole(1, "Tester")	1. Call updateEmployeeRole() with valid inputs. 2. Assert that the role is updated correctly.	The role of the employee with ID 1 is updated in the system.	As Expected
Emp_TC_05	High	Employee with ID 1 exists	deleteEmployee(1)	1. Call deleteEmployee() with valid inputs. 2. Assert that the employee is deleted from the system.	The employee with ID 1 is removed from the system.	As Expected

3. ProjectServiceImpl

TestCaseID	Priority	Preconditions	Input Test Data (Function Parameters)	Steps to be Executed	Expected Results	Actual Results
Project_TC_01	High	ProjectDao is initialized	createProject("New Project", LocalDate.of(2024, 8, 25), 301)	1. Call createProject() with valid inputs. 2. Assert that the project is created successfully.	Project is successfully created in the system.	As Expected
Project_TC_02	High	Project with ID 1 exists	findProjectById(1)	1. Call findProjectById() with valid project ID. 2. Assert that the correct project details are returned.	Correct Project details corresponding to ID 1 are returned.	As Expected
Project_TC_03	High	Project with name "New Project" exists	findProjectByName("New Project")	1. Call findProjectByName() with valid project name. 2. Assert that the correct project details are returned.	Correct Project details corresponding to the name are returned.	As Expected
Project_TC_04	High	Project with ID 1 exists	updateProjectStatus(1, "Completed")	1. Call updateProjectStatus() with valid inputs. 2. Assert that the project status is updated correctly.	Project status is updated to "Completed" in the system.	As Expected
Project_TC_05	High	Project with ID 1 exists	deleteProject(1)	1. Call deleteProject() with valid project ID. 2. Assert that the project is deleted from the system.	Project with ID 1 is removed from the system.	As Expected

4. CredentialsServiceImpl

TestCaseID	Priority	Preconditions	Input Test Data (Function Parameters)	Steps to be Executed	Expected Results	Actual Results
Cred_TC_01	High	CredentialDao is initialized	addCredentials("user1", "Password123")	1. Call addCredentials() with valid username and password. 2. Assert no exceptions are thrown.	Credentials are added successfully, no exceptions are thrown.	As Expected
Cred_TC_02	High	Credentials for "user1" are added	verifyCredentials("user1", "Password123")	1. Call verifyCredentials() with existing username and correct password. 2. Assert that it returns true.	Returns true, indicating credentials are correct.	As Expected
Cred_TC_03	High	Credentials for "user1" are added	verifyCredentials("user1", "WrongPassword")	1. Call verifyCredentials() with existing username and incorrect password. 2. Assert that it throws IncorrectUsernameOrPasswordException.	Throws IncorrectUsernameOrPasswordException indicating wrong password.	As Expected
Cred_TC_04	High	CredentialDao is initialized	addCredentials("user2", "weak")	1. Call addCredentials() with valid username and weak password. 2. Assert that it throws WeakPasswordException.	Throws WeakPasswordException indicating the password is too weak.	As Expected
Cred_TC_05	High	Credentials for "user1" are added	verifyCredentials("user2", "Password123")	1. Call verifyCredentials() with non-existing username and any password. 2. Assert that it returns false.	Returns false, indicating that credentials do not match.	As Expected