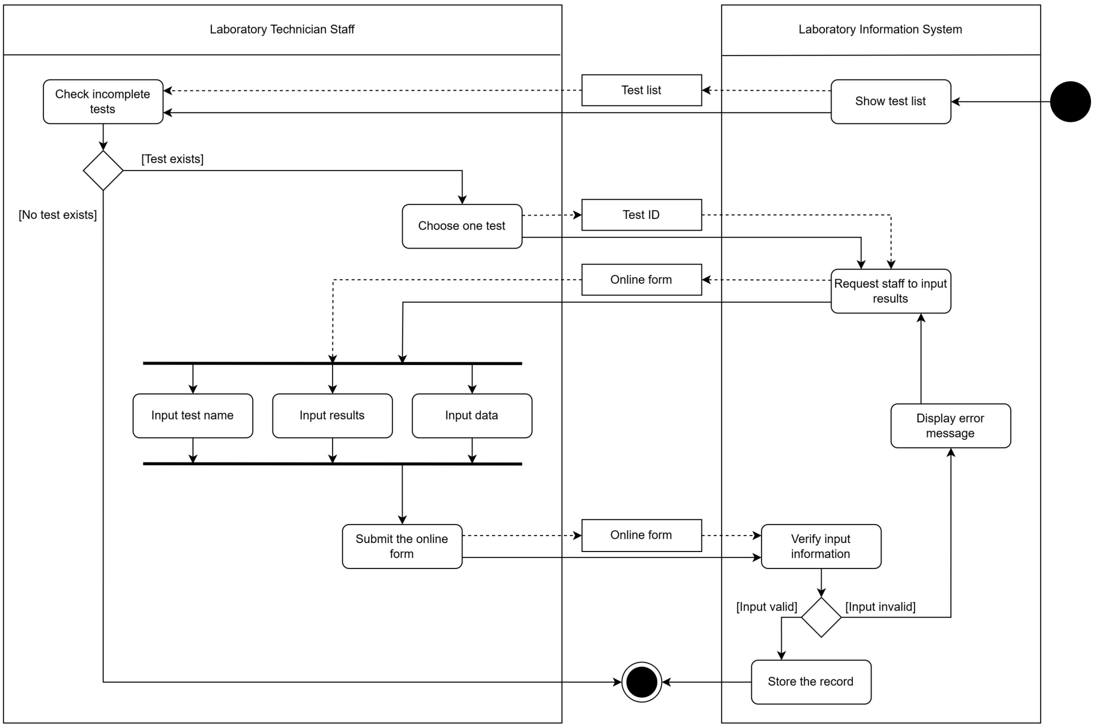
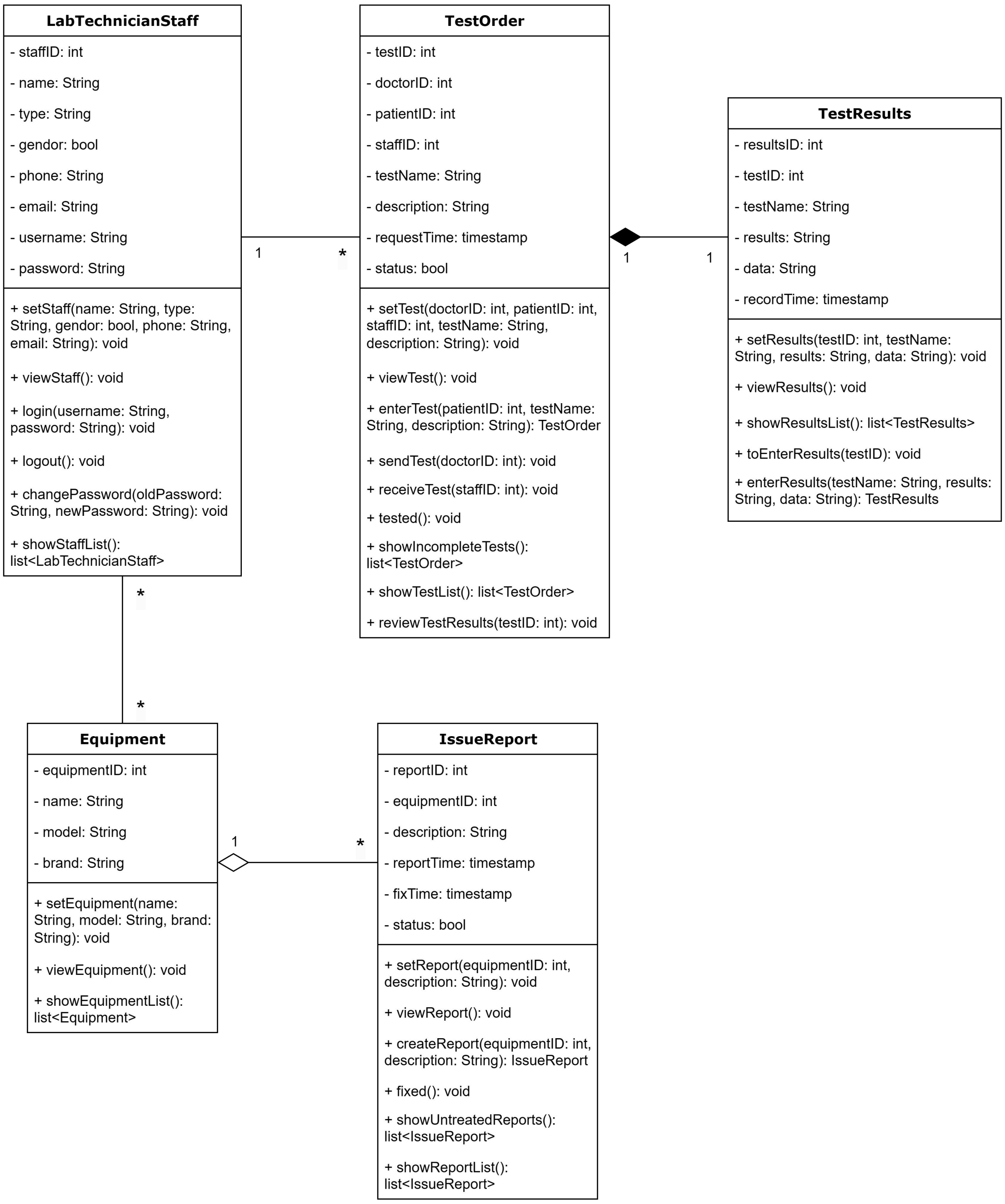
Task 1: Software Modelling and Specification (Subsystem: HealthGuardian Lab)

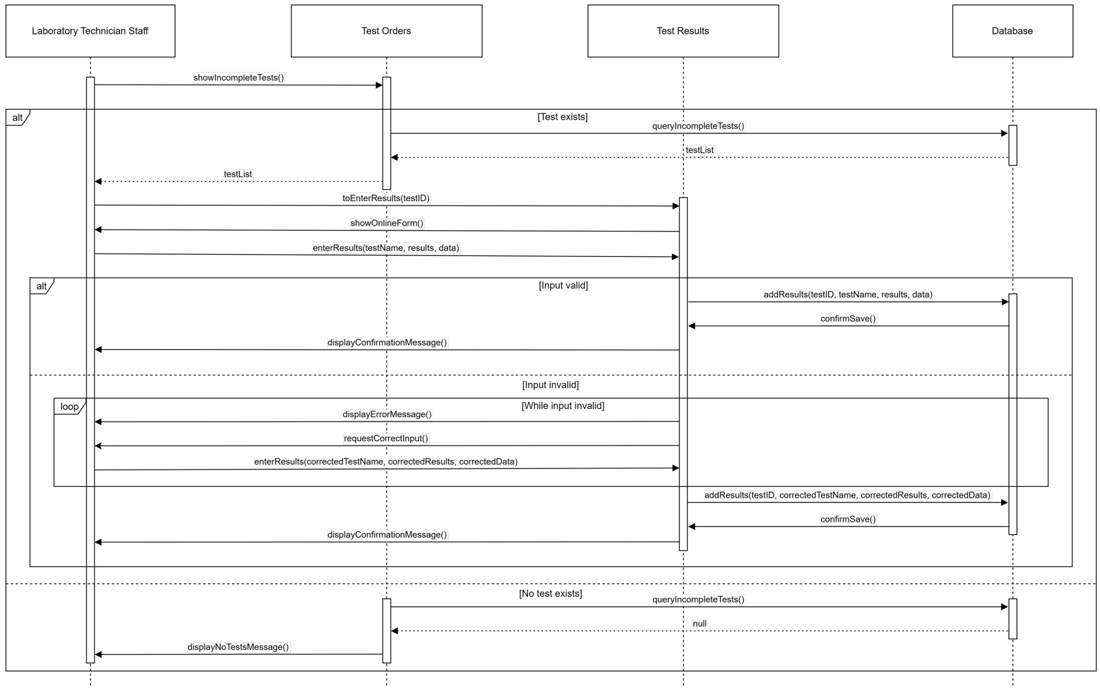
b) Activity Model (Usecase: Data Entry)



c) Structural Model

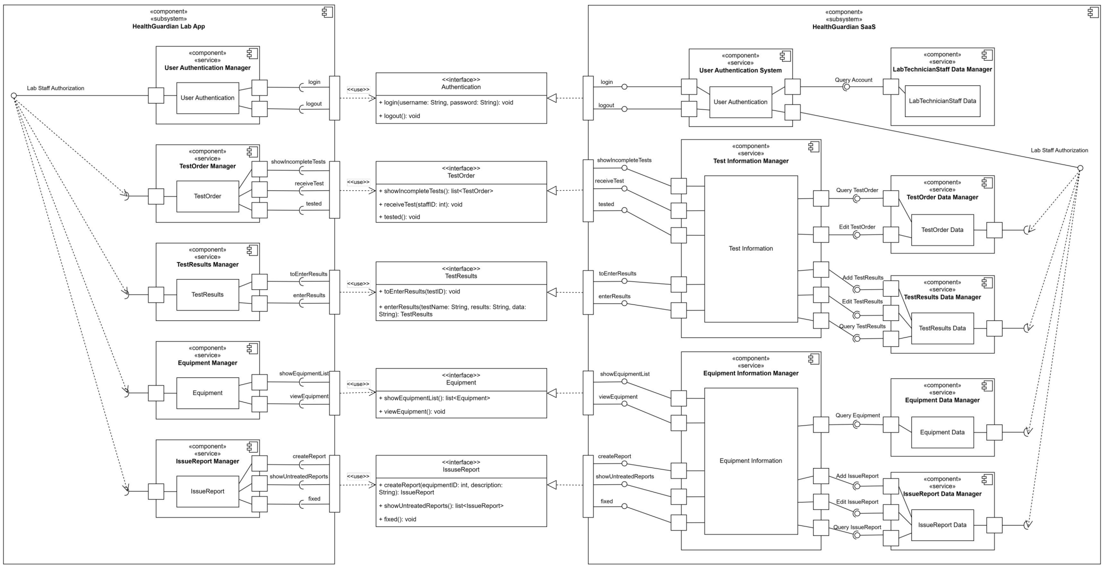

LabTechnicianStaff
- staffID: int
- name: String
- type: String
- gendor: bool
- phone: String
- email: String
- username: String
- password: String
+ setStaff(name: String, type: String, gendor: bool, phone: String, email: String): void
+ viewStaff(): void
+ login(username: String, password: String): void
+ logout(): void
+ changePassword(oldPassword: String, newPassword: String): void
+ showStaffList(): list<LabTechnicianStaff>
1
*
TestOrder
- testID: int
- doctorID: int
- patientID: int
- staffID: int
- testName: String
- description: String
- requestTime: timestamp
- status: bool
+ setTest(doctorID: int, patientID: int, staffID: int, testName: String, description: String): void
+ viewTest(): void
+ enterTest(patientID: int, testName: String, description: String): TestOrder
+ sendTest(doctorID: int): void
+ receiveTest(staffID: int): void
+ tested(): void
+ showIncompleteTests(): list<TestOrder>
+ showTestList(): list<TestOrder>
+ reviewTestResults(testID: int): void
1
1
TestResults
- resultsID: int
- testID: int
- testName: String
- results: String
- data: String
- recordTime: timestamp
+ setResults(testID: int, testName: String, results: String, data: String): void
+ viewResults(): void
+ showResultsList(): list<TestResults>
+ toEnterResults(testID): void
+ enterResults(testName: String, results: String, data: String): TestResults
*
*
Equipment
- equipmentID: int
- name: String
- model: String
- brand: String
+ setEquipment(name: String, model: String, brand: String): void
+ viewEquipment(): void
+ showEquipmentList(): list<Equipment>
1
*
IssueReport
- reportID: int
- equipmentID: int
- description: String
- reportTime: timestamp
- fixTime: timestamp
- status: bool
+ setReport(equipmentID: int, description: String): void
+ viewReport(): void
+ createReport(equipmentID: int, description: String): IssueReport
+ fixed(): void
+ showUntreatedReports(): list<IssueReport>
+ showReportList(): list<IssueReport>

d) Behaviour Model (Usecase: Data Entry)



Task 2: Software Architectural Design (Subsystem: HealthGuardian Lab)

b) Architecture of the Subsystem



Component Tables

Component	User Authentication Manager
Description	This is a HealthGuardian Lab App running on the Cloud to process user authentication and authorization.
Type	HealthGuardian Lab App
Required Interfaces	login, logout
Provided Interfaces	Lab Staff Authorization

Component	TestOrder Manager
Description	This is a HealthGuardian Lab App running on the Cloud to process function related to the TestOrder class.
Type	HealthGuardian Lab App
Required Interfaces	showIncompleteTests, receiveTest, tested, Lab Staff Authorization
Provided Interfaces	None

Component	TestResults Manager
Description	This is a HealthGuardian Lab App running on the Cloud to process function related to the TestResults class.
Type	HealthGuardian Lab App
Required Interfaces	toEnterResults, enterResults, Lab Staff Authorization
Provided Interfaces	None

Component	Equipment Manager
Description	This is a HealthGuardian Lab App running on the Cloud to process function related to the Equipment class.
Type	HealthGuardian Lab App
Required Interfaces	showEquipmentList, viewEquipment, Lab Staff Authorization
Provided Interfaces	None

Component	IssueReport Manager
Description	This is a HealthGuardian Lab App running on the Cloud to process function related to the IssueReport class.
Type	HealthGuardian Lab App
Required Interfaces	createReport, showUntreatedReports, fixed, Lab Staff Authorization
Provided Interfaces	None

Component	User Authentication System

Description	This is a microservice running on the Cloud to process user authentication and authorization.
Type	Microservice
Required Interfaces	Query Account
Provided Interfaces	login, logout, Lab Staff Authorization

Component	Test Information Manager
Description	This is a microservice running on the Cloud to process function related to the TestOrder and TestResults classes.
Type	Microservice
Required Interfaces	Query TestOrder, Edit TestOrder, Add TestResults, Edit TestResults, Query TestResults
Provided Interfaces	showIncompleteTests, receiveTest, tested, toEnterResults, enterResults

Component	Equipment Information Manager
Description	This is a microservice running on the Cloud to process function related to the Equipment and IssueReport classes.
Type	Microservice
Required Interfaces	Query Equipment, Add IssueReport, Edit IssueReport, Query IssueReport
Provided Interfaces	showEquipmentList, viewEquipment, createReport, showUntreatedReports, fixed

Component	LabTechnicianStaff Data Manager
Description	This is a microservice running on the Cloud to process data

	of the LabTechnicianStaff class.
Type	Microservice
Required Interfaces	None
Provided Interfaces	Query Account

Component	TestOrder Data Manager
Description	This is a microservice running on the Cloud to process data of the TestOrder class.
Type	Microservice
Required Interfaces	Lab Staff Authorization
Provided Interfaces	Query TestOrder, Edit TestOrder

Component	TestResults Data Manager
Description	This is a microservice running on the Cloud to process data of the TestResults class.
Type	Microservice
Required Interfaces	Lab Staff Authorization
Provided Interfaces	Add TestResults, Edit TestResults, Query TestResults

Component	Equipment Data Manager
Description	This is a microservice running on the Cloud to process data of the Equipment class.
Type	Microservice

Required Interfaces	Lab Staff Authorization
Provided Interfaces	Query Equipment

Component	IssueReport Data Manager
Description	This is a microservice running on the Cloud to process data of the IssueReport class.
Type	Microservice
Required Interfaces	Lab Staff Authorization
Provided Interfaces	Add IssueReport, Edit IssueReport, Query IssueReport

Task 3: Software Testing (Subsystem: HealthGuardian Lab)

a) Unit Test Plan (Class: IssueReport)

1. createReport(equipmentID, description)

Test Case	Method and Parameters	Expected Output
Successful Report Creation	createReport(123, "Microscope not focusing properly")	IssueReport object
Report Creation with Invalid Equipment ID	createReport(-1, "Defective cable")	Error message
Report Creation with Empty Description	createReport(123, "")	Error message

2. showUntreatedReports()

Test Case	Method and Parameters	Expected Output
Retrieve Untreated Reports	showUntreatedReports()	List of IssueReport objects
No Untreated Reports Available	showUntreatedReports()	Empty list

3. fixed()

Test Case	Method and Parameters	Expected Output
Mark Report as Treated	fixed()	Method call with no return value
Attempt Fixed Without Reports	fixed()	Error message

b) System Test Plan (Usecase: Instrument Troubleshooting)

1. **Scenario:** Reporting an Equipment Issue

Test Progress:

Actor (Lab Technician Staff)	System (HealthGuardian SaaS)
1) Identifies an issue with laboratory equipment and logs into the system.	2) Authenticates the staff member and provides access to the issue reporting tool.
3) Uses the issue reporting tool to create a new report, entering the equipmentID and description of the issue.	4) Validates the equipmentID and processes the report, creating an IssueReport entry.
5) Submits the report and awaits confirmation.	6) Stores the new report in the system and confirms the successful submission to the staff member.
7) Receives the IssueReport ID for future reference.	8) Logs the submission of the new IssueReport for tracking and resolution purposes.

Test Data:

Input: equipmentID = 123, description = "Microscope not focusing properly"

Expected Output: IssueReport ID and confirmation of report submission

2. **Scenario:** Viewing and Addressing Untreated Reports

Test Progress:

Actor (Lab Technician Staff)	**System (HealthGuardian SaaS)**
1) Logs into the system to check for untreated issue reports.	2) Authenticates the maintenance staff and retrieves a list of untreated reports.
3) Reviews the list of reports and selects one to address.	4) Displays the selected IssueReport details.
5) Marks the issue as resolved after performing maintenance.	6) Processes the fixed method to update the report's status in the system.
7) Confirms the issue has been addressed and logs out.	8) Confirms the issue resolution and logs the completion of the maintenance task.

Test Data:

Input: None required for viewing; maintenance staff action for resolving issues

Expected Output: Updated system state reflecting the addressed issues