ADDIS ABABA UNIVERISITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

LABORATORY INFORMATION MANAGEMENT SYSTEM (LIMS)

Software Design Specification

PREPARED BY: -

1. HILINA AYALEW
2. MICHAEL DESALEGN
3. RIHANNA ABDELA
4. SAMRAWIT MULUGETA
5. SOLYANA MENGISTU
6. TSIYON WULETAW
7. ZEBIBA HASSAN

ADVISORS:

Date: Mar-19-2018

Revision History

Date	Description	Author	Comments
May-13-2017	Version 1.0	HMRSSTZ	
June-5-2018	Version 2.0	HMRSSTZ	

Document Approval

The following Software Requirements Specification has been accepted and approved by the following

Signature	Printed Name	Title	Date

Table of Contents

TABLE OF FIGURES

LIST OF TABLES

ABBREVIATIONS, ACRONYMS AND DEFINITIONS

Terms	**Abbreviations, Acronyms, Definitions**
LIMS	Laboratory Information Management System
SDS	Software Design Specification is a description of the design of software system to be developed
JavaScript	JavaScript is a computer programming language commonly used to create interactive effects within web browsers.
HTTP	application protocol that is the foundation of data communication for the World Wide Web.
HTML5	Hypertext Markup Language is the fifth major revision of the Hypertext Markup Language (HTML).
OS	Operating System
Admin	Abbreviation for Administrator
UML	Unified modeling language
CSS	Cascading style sheet
MVC	Model-view-controller

1 Introduction

1.1 Purpose

The purpose of the System Design document is to translate the business requirements and business processes into a technical design that will be used to develop the application.

1.2 General Overview

The implementation of LIMS will take a three tier architecture (MVC system architecture). Model-View-Controller architectural model is an approach to graphical user interface design that allows for multiple presentations of an object and separate styles of interaction with each of these presentations. We choose this model because it isolates the different responsibility in our system. The controller manages all the user requests when the user clicks on the desktops interfaces to perform the required action. The model is the data and the rules applying to it, the controller prepares any data need by the view. The view provides different ways to present the data prepared by the controller.

The two main reasons why we choose the MVC model is regarding its maintainability and security, which is the non-functional requirement of the system. MVC is a three- tier-architecture for the dissociation of the data access logic, control logic and the user interface. This segregation of the system into different layers will help in the maintenance of the system. MVC also provides a foundation for implementing security components at the interfaces of each level.

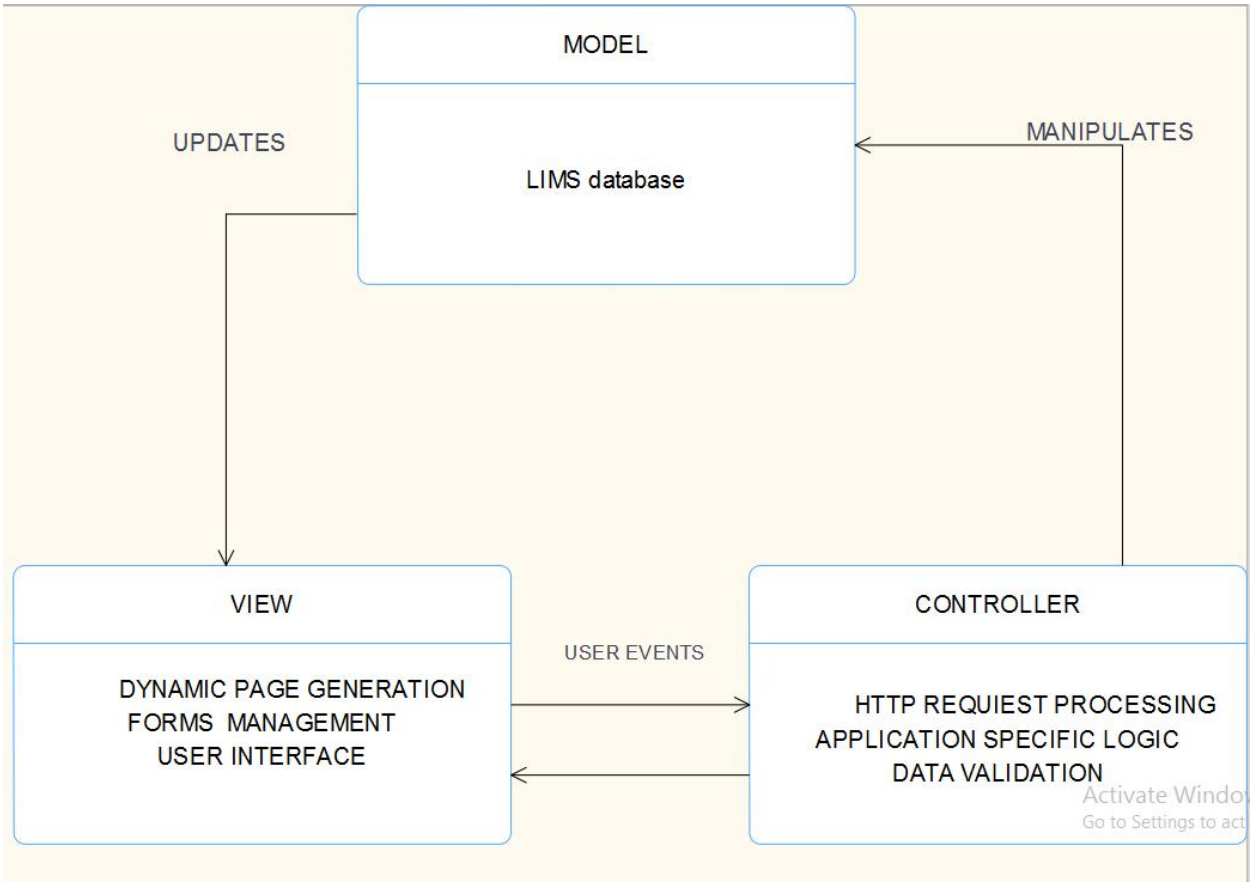


Figure 1 Diagram of interaction within the MVC pattern

1.3 Development Methods & Contingencies

The system will be designed using the object-oriented method and Unified Modeling Language (UML) will be the language used for the system and software design. It will be written in HTML5, CSS3, JavaScript, and Java. HTML: allows clear writing and descriptive codes. CSS: allows the styling of the website. JavaScript: allows the pages to be provided dynamically. Java: allows creating dynamic web pages. The query and updating of user data will be handled using Structured Query Language. MYSQL Community Server will be used. The system will be running on apache web server.

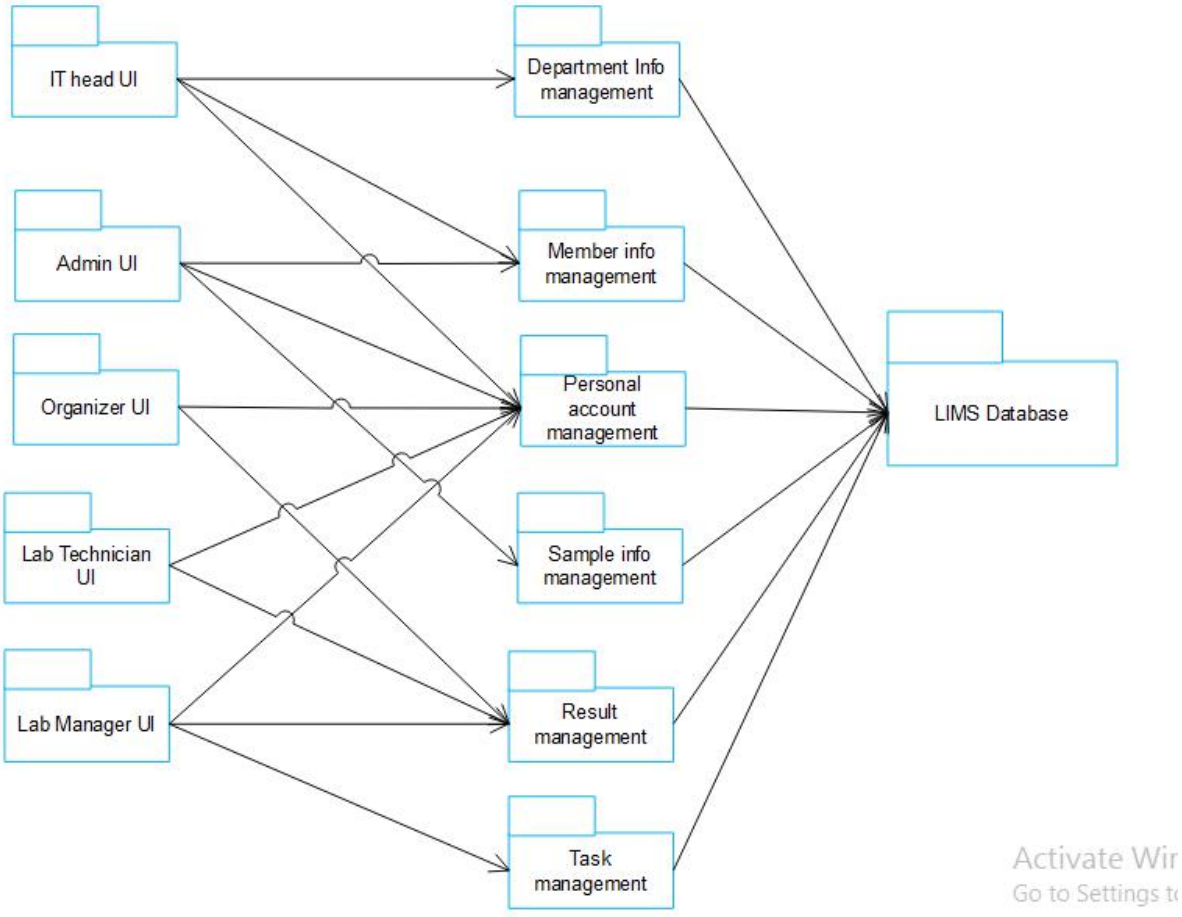
2 System Architecture

2.1 Subsystem decomposition

Layer 1

Figure 2 Layer 2 diagram

Layer 2



Layer 3

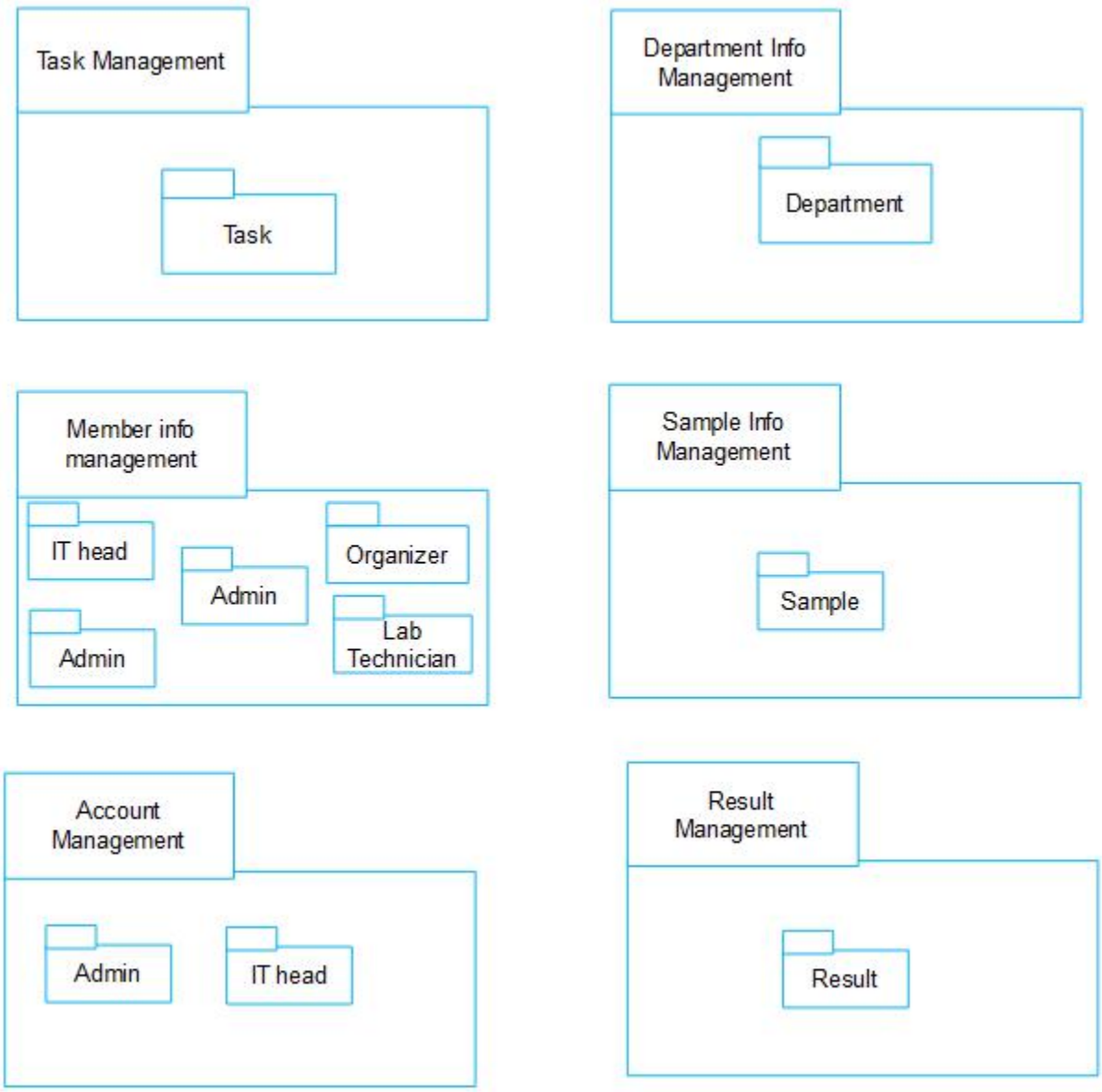


Figure 3 Layer 3 diagram

2.2 Hardware/software mapping

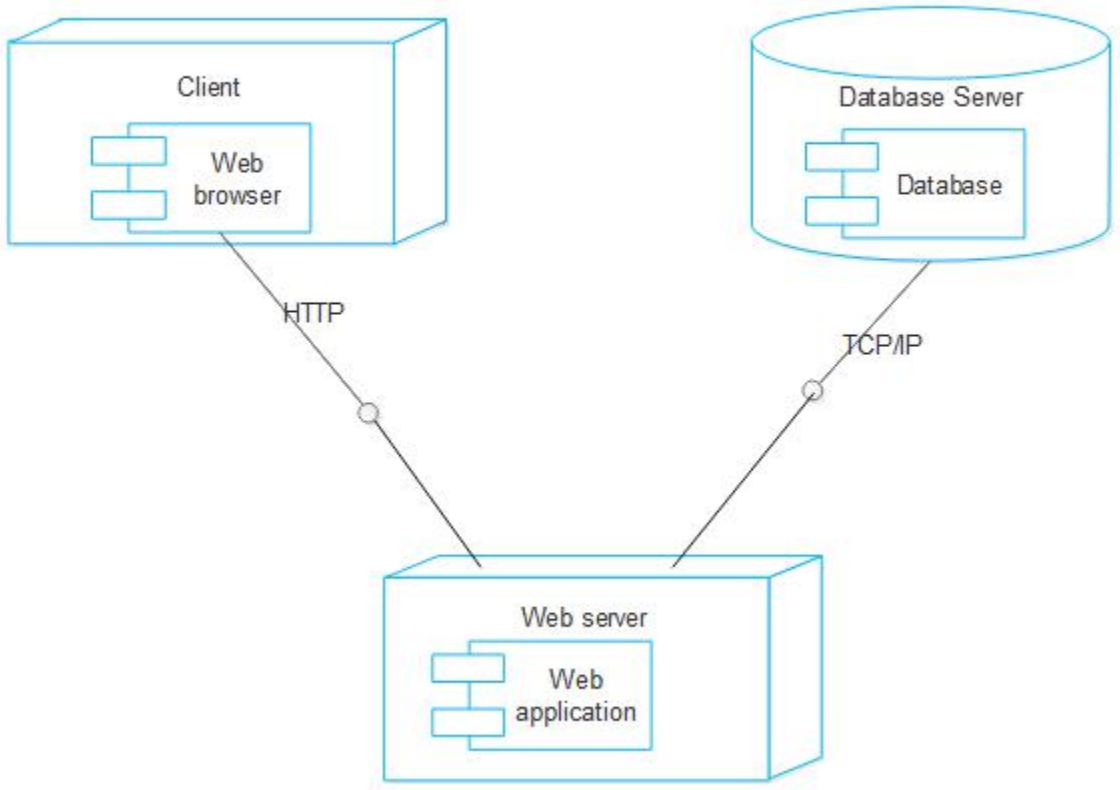


Figure 4 Deployment Diagram

3 Object model

3.1 Class Diagram

Figure 5 class diagram

3.2 Sequence Diagram

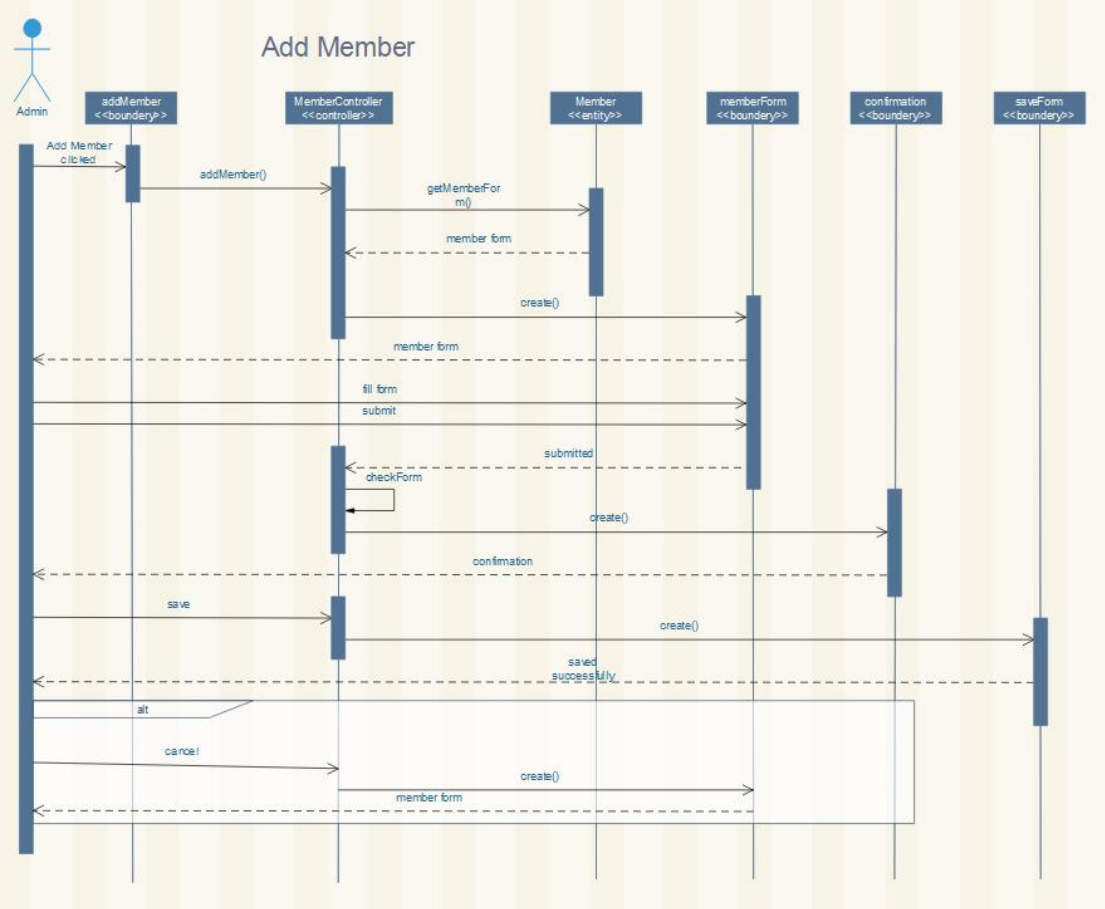


Figure 6 Add Member sequence diagram

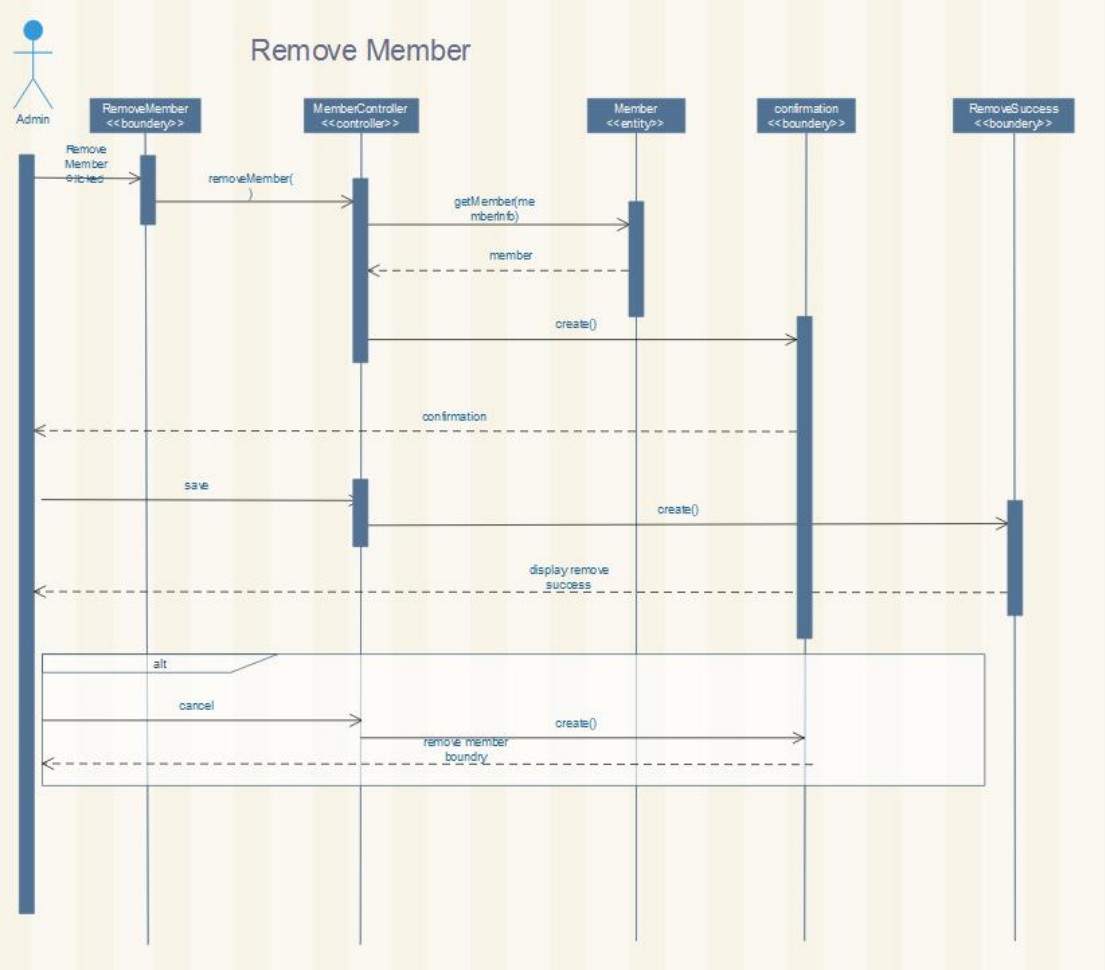


Figure 7 Remove Member sequence diagram

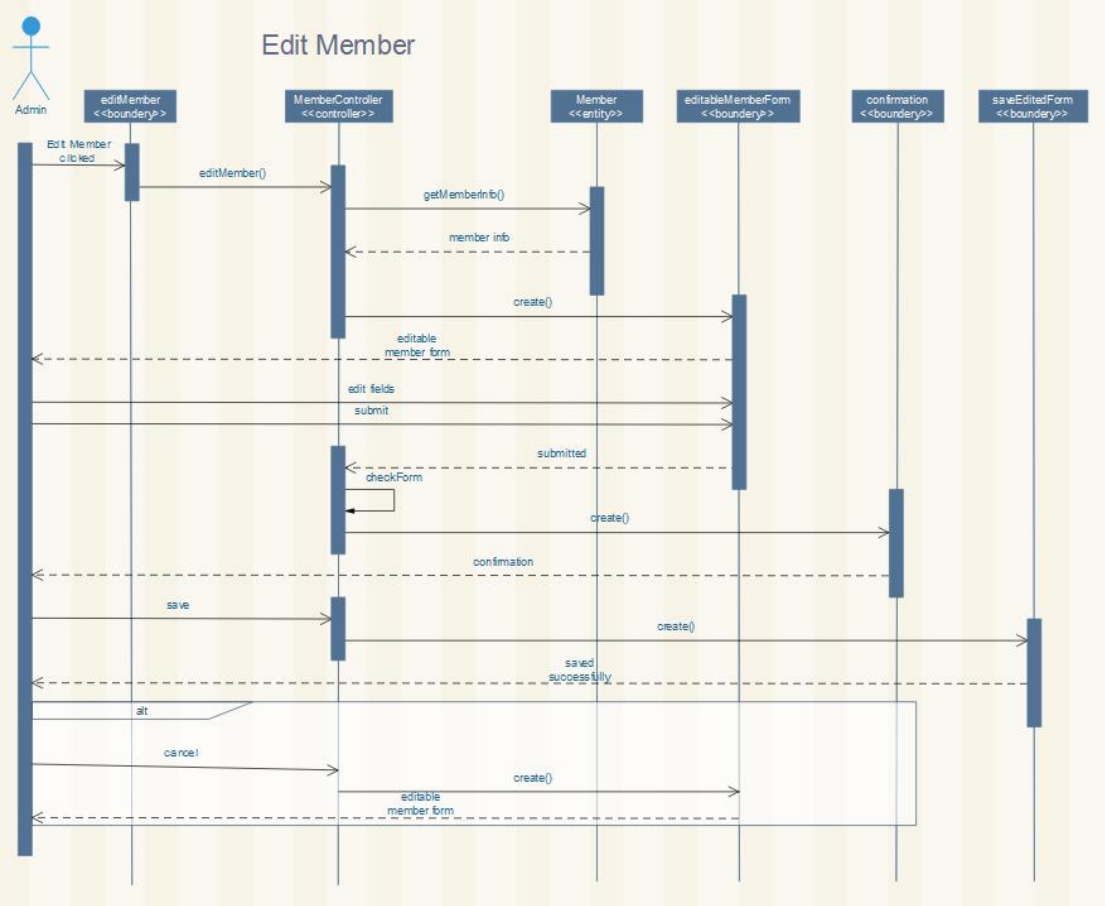


Figure 8 Edit Member Sequence Diagram

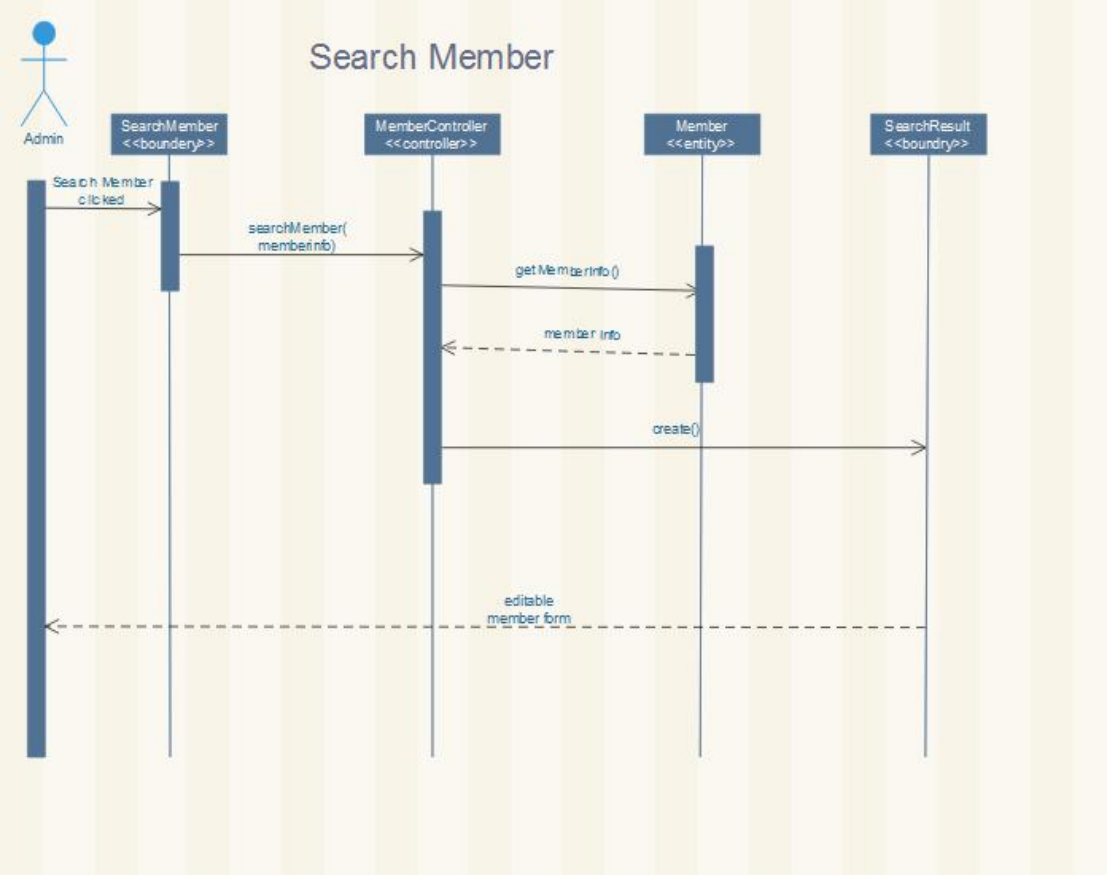


Figure 9 Search Member

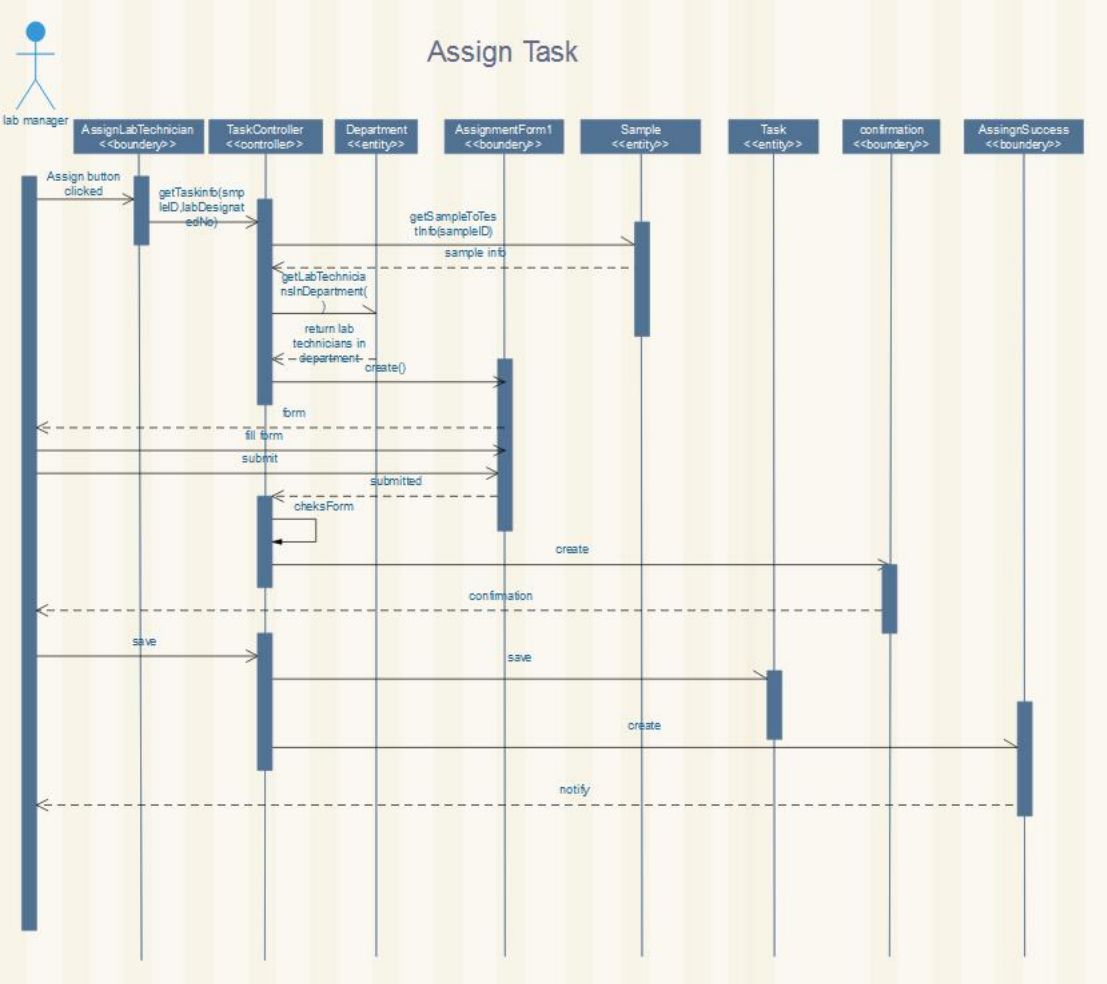


Figure 10 Assign task sequence diagram

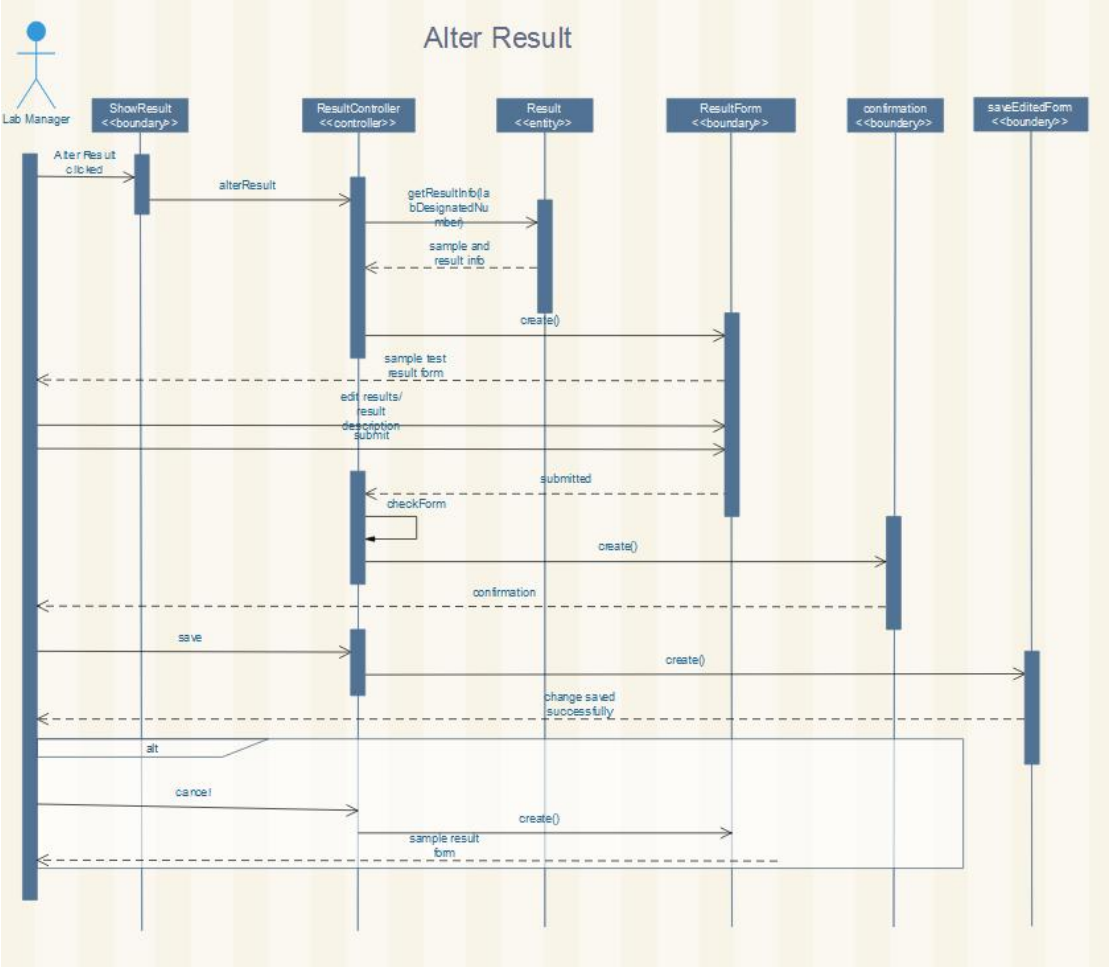


Figure 11 Alter Result sequence Diagram

Member

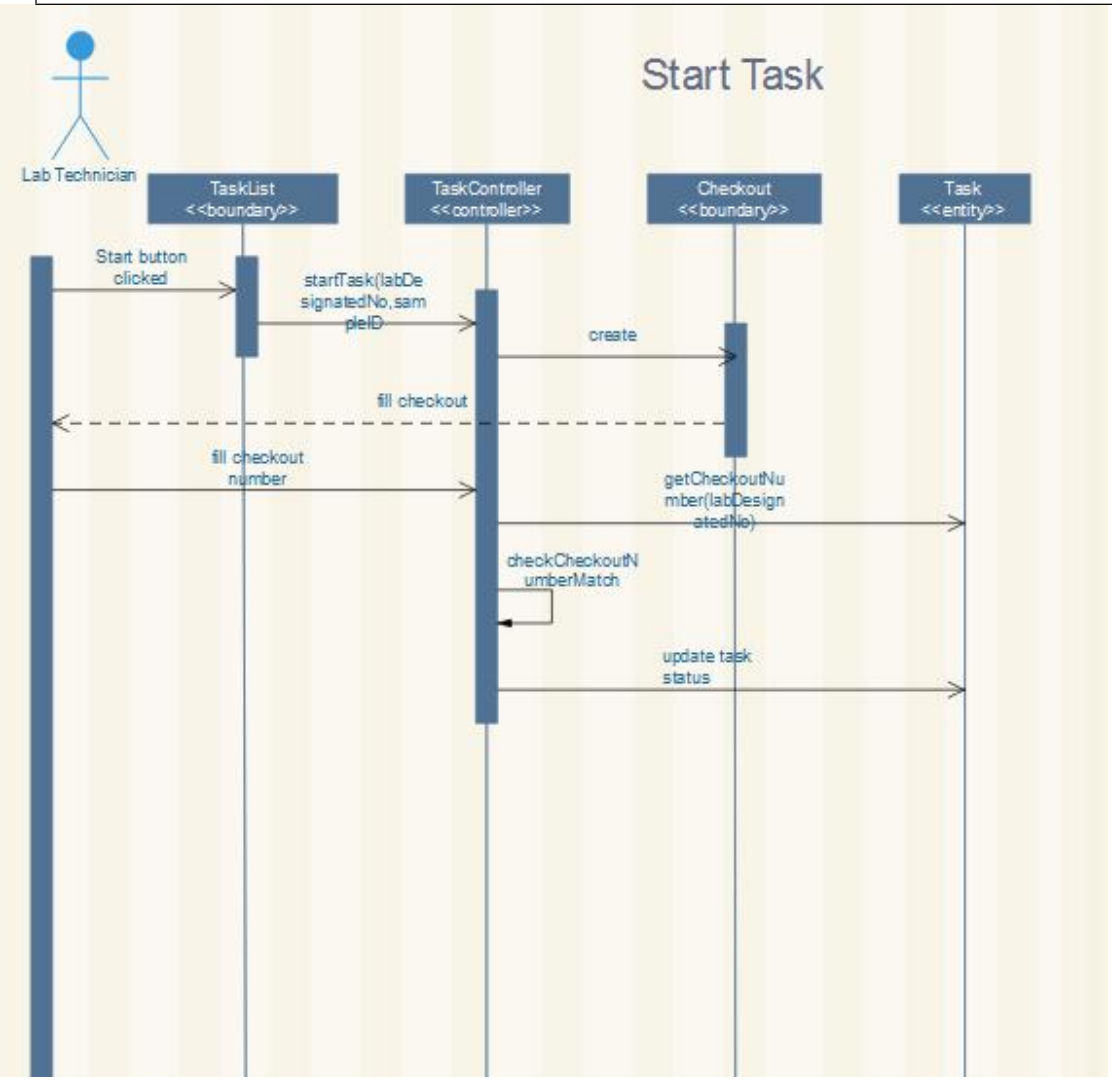


Figure 12 Start task sequence diagram

4 DETAILED DESIGN

This section will provide the detailed implementation of the classes from the class diagram, sequence diagram and state diagram

4.1 Member Class

-memberId: String -password: String -firstName: String -lastName: String -dateOfBirth: Date -sex: String -phoneNo: String -email: String -address: String -department: String -typeOfUser:String
+setPasword(newPassword:String):void +getPassword(memberId: String):String +setMemberId(newMemberId:String):void +getMemberId():String +setFirstName(newFirstName:String):void +getFirstName():String +setLastName(newLastName:String):void +getLastName():String +setDateOfBirth(newDateOfBirth:Date):void +getDateOfBirth():Date +setSex(newSex:String):void +getSex():String +setPhoneNo(newPhoneNo:String):void

+getPhoneNo():String +setEmail(newEmail:String):void +getEmail():String +setAddress(newAddress:String):void +getAddress():String +setDepartment(newDepartment:String):void +getDepartment():String +setTypeOfUser(category:String):void +getTypeOfUser():String

Table 1 Member Class

Attribute	Type	Visibility	Invariant
memberId	String	private	memberId<> NULL should contain three letters followed by six numbers separated with two slash.
password	String	Private	password<> NULL should contain 8-20 characters
firstName	String	private	firstname <> NULL shouldn't contain special characters and integers and must be between 2-20 letters.
lastName	String	private	lastName <> NULL shouldn't contain special characters and integers and must be between 2-20 characters
dateOfBirth	Date	Private	dateOfBirth<>NULL and the date should be in 'dd-mm-yyyy' format, where d is for date, m for month, y for

			year respectively.
email	String	private	Email <> NULL ✓ Must contain @ ✓ Must contain. (dot) ✓ Position of @ >1 ✓ Position of (dot)>position of @ + 2 Position of (dot)+3<= total length of email address and the total character of the Email is at least 5 characters

Table 2 Attribute description of Member Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setPasword	Public	void	newPassword :String	Authorized personnel want to add new password and The password should be valid	New password will be set.
getPassword	Pubic	String	--	Password stored for the member in the database.	Password will be retrieved.
generateMemberId	Public	void	newMemberId:String	All the member form is filled with a valid inputs	New member id will be set.

getMemberId	Pubic	String	--	Member id stored in database.	Member id will be retrieved.
setFirstName	public	void	newFirstName:String	Authorized personnel want to add first name for the member.	New first name will be set.
getFirstName	public	String	--	First name stored in database.	First name will be retrieved.
setLastName	public	void	newLastName:String	Authorized personnel want to add last name for the member.	New last name will be set.
getLastName	public	String	--	Last name stored in database.	Last name will be retrieved.
setDateOfBirth	public	void	newDateOfBirth:Date	Authorized personnel want to add date of birth for the member.	Date of birth will be set.
getDateOfBirth	public	String	--	Date of birth stored in database.	Date of birth will be retrieved.
setSex	Public	void	newSex:String	Authorized personnel want to add gender for the member.	Gender will be set.
getSex	public	String	--	Gender should be stored in database.	Gender will be retrieved.

setPhoneNo	public	void	newPhoneNo :String	Authorized personnel want to add phone number for the member.	Phone number will be set.
getPhoneNo	public	String	--	Phone number should be stored in database.	Phone number will be retrieved.
setEmail	public	void	newEmail:Stri ng	Authorized personnel want to add email for the member.	Email will be set.
getEmail	public	String	--	Email should be stored in database	Email will be retrieved.
setAddress	public	void	newAddress:S tring	Authorized personnel want to add address for a new member.	Address will be set for the new member.
getAddress	public	String	--	Address should be stored in database	Address will be retrieved.
setDepartme nt	public	void	newDepartm ent:String	Authorized personnel want to add department for a new member.	Address will be set for the new member.
getDepartme nt	public	String	--	Department should be stored in database	Department will be retrieved.

setTypeOfUser	public	void	category:String	Authorized personnel want to add type of user for a new member.	Type of user will be set for the new member.
getTypeOfUser	public	String	--	Type of user should be stored in database	Type of user will be retrieved.

Table 3 Operation description for Member Class

4.2 Sample Class

Sample
-durationOfTestingTime: Integer -sampleId:String -sampleName:String -paramenter:String -standardReferences:String -validResultRangeOfSample:String -amount:float -typeOfSample:String
+setDurationOfTestingTime(newDurTime:Integer):void +getDurationOfTestingTime():Integer +setSampleId(newSampleId:String):void +getSampleId():String +setSampleName(newSampleName:String):void +getSampleName():String

+setParamenter(newParamenter:String):void
+getParamenter():String
+setStandardReferences(newStandardReferences:String):void
+getStandardReferences():String
+setValidResultRangeOfSample(newValidRange:String):void
+getValidResultRangeOfSample():String
+setAmount(newAmount:float):void
+getAmount():float
+setTypeOfSample(newType:String):void
+getTypeOfSample():String

Table 4 Sample class

Attribute	Type	Visibility	Invariant
durationOfTestingTime	String	private	durationOfTestingTime<> NULL should be in hh:mm' format, h for hour, m for minute.
sampleId	String	private	sampleId<> NULL should contain three letters followed by six numbers.
sampleName	String	private	sampleName<> NULL shouldn't contain special characters and integers and must be between 2-20 letters.
paramenter	String	private	paramenter<> NULL shouldn't contain special characters and integers and must be between 2-20 letters.
standardReferences	String	Private	standardReferences <>NULL shouldn't contain special characters and integers, must be

			between 2-20 letters.
validRangeOfSample	String	Private	validResultRangeOfSample<>NULL shouldn't contain special characters and integers, chooses from the list.
amount	float	private	amount<>NULL should only contain float number.
typeOfSample	String	private	typeOfSample<>NULL shouldn't contain special characters and integers, should be from the list.

Table 5 Attribute description of Sample Class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setDurationOfTestingTime	Public	void	newDurTime:Integer	Authorized personnel want to add duration of time for a new sample.	New duration of time will be set.
getDurationOfTestingTime	Pubic	Integer	--	Duration of time stored for the sample in the database.	Duration of time will be retrieved.
setSampleId	Public	void	newSampleId:String	Authorized personnel want to add sample id for a new sample.	Sample id will be set.
getSampleId	Pubic	String	--	Sample id stored for the sample in the database.	Sample id will be retrieved.
setSampleName	Public	void	newSampleName:String	Authorized personnel want to add sample name for a new sample.	Sample name will be set.

getSampleName	Pubic	String	--	Sample name stored for the sample in the database.	Sample name will be retrieved.
setParameter	Public	void	newParameter :String	Authorized personnel want to add parameter for a new sample.	Parameter will be set.
getParameter	Pubic	String	--	Parameter stored for the sample in the database.	Parameter will be retrieved.
setStandardReferences	Public	void	newStandardReferences:String	Authorized personnel want to add standard of reference for a new sample.	Standard of reference will be set.
getStandardReferences	Pubic	String	--	Standard of reference stored for the sample in the database.	Standard of reference will be retrieved.
setValidRangeOfSample	Public	void	newValidRange:String	Authorized personnel want to add sample validation range for a new sample.	Sample validation range will be set.
getValidRangeOfSample	Pubic	String	--	Validation of range stored for the sample in the database.	Validation of range will be retrieved.
setAmont	Public	void	newAmount:Integer	Authorized personnel want to add amount	Sample size will be set.

				for a new sample.	
getAmount	Pubic	Integer	--	Amount stored for the sample in the database.	Size will be retrieved.
setTypeOfSample	Public	void	newType:String	Authorized personnel want to add sample type for a new sample.	Sample type will be set.
getTypeOfSample	Pubic	String	--	Sample type stored for the sample in the database.	Sample type will be retrieved.

Table 6 Operation description for Sample Class

4.3 Result Class

Result
-sampleId :string -verifierID:string -paramenter:String -labDesignatedNumber:String -resultRange:String[] -resultDescription:String -validation:String
+setValidation(newValidation:String) +getValidation() +setResultDescription(newResultDescription:String) +getResultDescription() +setResultRangeOfSample(newResultRange:String)

+getResultRangeOfSample()

Table 7 Result class

Attribute	Type	Visibility	Invariant
sampleId	String	protected	sampleId<> NULL should contain three letters followed by six numbers.
memberId	String	protected	memberId<> NULL should contain three letters followed by six numbers separated with two slash.
parameter	String	protected	paramenter<> NULL shouldn't contain special characters and integers and must be between 2-20 letters.
validation	String	private	validation<> NULL shouldn't contain special characters and integers and must be between 10-15 characters
labDesignatedNo	String	protected	labDesignatedNo<>NULL must contain only eight numbers.
resultDescription	String	Private	resultDescription<>NULL shouldn't contain special characters and integers.
resultRangeOfSample ToBeTested	List<<listOfTestedRanges>>	Private	resultRangeOfSampleToBeTested<>NULL shouldn't contain special characters and integers.

Table 8 Attribute description of Result class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setValidation	Public	void	newValidation:String	Authorized personnel want to add new validation.	New validation will be set.
getValidation	Pubic	String	--	Validation stored for the sample result in the database.	Validation will be retrieved.
setResultDescription	Public	void	newResultDescription:String	Authorized personnel want to add new result description.	New result description will be set.
getResultDescription	Pubic	String	--	Result description stored for the sample result in the database.	Result description will be retrieved.
setResultRangeOfSample	Public	void	newResultRange:String	Authorized personnel want to add result range for a new sample to be tested.	Sample result range to be tested will be set.
getResultRangeOfSample	Pubic	String	--	Result range stored for the sample to be tested stored in the database.	Result range for the sample to be tested will be retrieved.

Table 9 Operation description of Result class

4.4 Task class

Task
-taskName:String #memberId:String #departmentId:String -status:String
+setTaskName(newTaskName:String):void +getTaskName(): String +setStatus(newStatus:String):void +getStatus(): String

Table 10 Task class

Attribute	Type	Visibility	Invariant
taskName	String	private	taskName<> NULL should contain 10-15 characters
memberId	String	protected	memberId<> NULL should contain three letters followed by six numbers separated with two slash.
departmentId	String	protected	departmentId<>NULL should contain three letters followed by six numbers.
status	String	Private	status<>NULL shouldn't contain special characters and integers and contain 8-20 letters

Table 11 Attribute description of task class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setTaskName	Public	void	newTaskName:String	Authorized personnel want to add new task name.	New task name will be set.
getTaskName	Pubic	String	--	Task name stored for the sample result in the database.	Task name will be retrieved.
setStatus	Public	void	newStatus:String	Authorized personnel want to add new status.	New status will be set.
getStatus	Pubic	String	--	Status stored for the task taking place stored in database.	Status will be retrieved.

Table 12 Operation description of Task class

4.5 Department class

Department
-departmentId:String -departmentName:String
+setDepartmentId(newDepId:String):void +getDepartmentId(): String +setDepartmentName(newDepName:String):void

+getDepartmentName(): String

Table 13 Department Class

Attribute	Type	Visibility	Invariant
departmentId	String	private	departmentId<>NULL should contain three letters followed by six numbers.
departmentName	String	private	departmentName<> NULL should contain 3-20 letters.

Table 14 Attribute description of Department class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setDepartmentId	Public	void	newDepId:String	Authorized personnel want to add new department Id.	New department id will be set.
getDepartmentId	Pubic	String	--	Department id stored in database.	Department id will be retrieved.
setDepartmentName	Public	void	newDepName:String	Authorized personnel want to add new department name.	New department name will be set.
getDepartmentName	Pubic	String	--	Department name stored	Department name will be

				in the database.	retrieved.

Table 15 Operation description of Department class

4.6 SampleToTest Class

SampleToTest
-customerId:String -sampleNameToBeTested:String #sampleId:String -paramenterToBeTested:String -standardReferencesToBeTested:String -amountToBeTested:float -typeOfSampleToBeTested:String -checkOutNo:String
+setCustomerId():String +getCustomerId():String +setSampleNameToBeTested(newNameTested:String):void +getSampleNameToBeTested():String +setParamenterToBeTested(newParamTested:String):void +getParamenterToBeTested():String +setStandardReferencesToBeTested(newStandTested:String):void +getStandardReferencesToBeTested():String +setAmountToBeTested(newAmountTested:Integer):void +getAmountToBeTested():Integer +setTypeOfSampleTested(newTypeTested:String):void

+getTypeOfSampleTested():String +generateCheckOutNo():String +generateLabDesignatedNo():String +getCheckOutNo():String

Table 19 SampleToTest class

Attribute	Type	Visibility	Invariant
customerId	String	private	customerId<> NULL must contain ten numbers.
sampleNameToBeTested	String	private	sampleNameToBeTested <> NULL should shouldn't contain special characters and integers, should contain 2 - 20 letters.
sampleId	String	protected	sampleId<> NULL should contain three letters followed by six numbers.
paramenterToBeTested	String	private	paramenterToBeTested <> NULL shouldn't contain special characters and integers and must be between 2-20 letters.
standardReferencesToBeTested	String	Private	standardReferencesToBeTested<>NULL shouldn't contain special characters and integers, must be between 2-20 letters.
amountToBeTested	float	private	amountToBeTested<>NULL should only contain float number..

typeOfSampleToBeTested	String	private	typeOfSampleToBeTested<>NULL shouldn't contain special characters and integers, should be from the list.
checkOutNo	String	private	checkOutNo<>NULL must contain 10 numbers.

Table 20 Attribute description of SampleToTest class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
setCustomerId	Public	void	newCustomerId :String	Authorized personnel want to add customer id for a new sample receiver.	Customer id will be set.
getCustomerId	Pubic	String	--	Customer id stored for the sample to be tested in the database.	Customer id will be retrieved.
setSampleNameToBeTested	Public	void	newNameTested:String	Authorized personnel want to add sample name to be tested.	Sample name to be tested will be set.
getSampleNameToBeTested	Pubic	String	--	Sample name stored for the sample in the database.	Sample name to be tested will be retrieved.

setParamenterToBeTested	Public	void	newParamTested:String	Authorized personnel want to add parameter for a new sample to be tested.	Parameter to be tested will be set.
getParamenterToBeTested	Pubic	String	--	Parameter stored for the sample to be tested in the database.	Parameter to be tested will be retrieved.
setStandardReferencesToBeTested	Public	void	newStandTested:String	Authorized personnel want to add standard of reference for a new sample to be tested.	Standard of reference for the sample to be tested will be set.
getStandardReferencesToBeTested	Pubic	String	--	Standard of reference stored for the sample to be tested in the database.	Standard of reference for the sample to be tested will be retrieved.
setTypeOfSampleTested	Public	void	newTypeTested:String	Authorized personnel want to add sample type for a new sample to be tested.	Sample type for the sample to be tested will be set.
getTypeOfSa	Pubic	String	--	Sample type	Sample type

mpleTested				stored for the sample to be tested in the database.	for the sample to be tested will be retrieved.
generateCheckOutNo	public	String	--	Sample to be tested form should be filled.	Identifier with eight digits will be generated.
generateLabDesignatedNo	public	String	--	Sample test form should be completely filled.	Identifier with eight digits will be generated.
getCheckOutNo	Public	String	--	Check out number already generated and stored in database.	Check out number will be retrieved.

Table 21 Operation description of SampleToTest class

4.7 ResultController Class

ResultController
#labDesignatedNo:String
+feedResult(labDesignatedNo:String): void +editResult(labDesignatedNo:String):void +viewResult(labDesignatedNo:String):void +deleteResult(labDesignatedNo:String):void

Table 22 ResultController class

Attribute	Type	Visibility	Invariant
labDesignatedNo	String	protected	labDesignatedNo<>NULL should contain eight numbers.

Table 23 Attribute description of Result controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
feedResult	Public	void	labDesignatedNo:String	Authorized personnel want to feed result.	Result will be submitted and stored in the database.
editResult	Pubic	void	labDesignatedNo:String	Result stored in database.	Result will be displayed for editable purpose.
viewResult	Public	void	labDesignatedNo:String	Result stored in database.	Result will be displayed for readable purpose.
deleteResult	Pubic	void	labDesignatedNo:String	Result stored in database.	Result will be deleted and removed from database.

Table 24 Operation Description of ResultController

4.8 AccountManager Class

AccountManager

#memberId:String #password:String
+authenticate(password:String, memberId:String): void +addMember(Member member):void +editMember(memberId:String):void +viewMember(memberId:String):void +deleteMember(memberId:String):void +setPassword(newPassword:String):void +getPassword(): String

Table 25 AccountController class

Attribute	Type	Visibility	Invariant
memberId	String	protected	memberId<>NULL should contain three letters followed by six numbers.

Table 26 Attribute description of AccountController class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addMember	Public	void	List<<Member>>	Authorized personnel want to add a member.	Member will be stored in the database.
editMember	Pubic	void	memberId:String	Member stored in database.	Member will be displayed for editable purpose.
viewMember	Public	void	memberId:String	Member stored in	Member will be displayed

				database.	for readable purpose.
deleteMembe r	Pubic	void	memberId:String	Member stored in database.	Member will be deleted and removed from database.
authenticate	public	void	password:String, memberId:String	Password and member id should already exist and stored in database .	Member will be logged in to account.
setPassword	Public	void	newPassword:St ring	Authorized personnel want to add new password.	New password will be set.
getPassword	Pubic	String	--	Password stored in database.	Password will be retrieved.

Table 27 Operation description of AccountController clas

4.9 SampleController Class

SampleController
#sampleId:String
+addSample(Sample sample):void +editSample(sampleId:String):void +viewSample(sampleId:String):void

+deleteSample(sampleId:String):void

Table 28 SampleController class

Attribute	Type	Visibility	Invariant
sampleId	String	protected	sampleID<> NULL should contain three characters followed by six integers

Table 29 Attribute description of samplecontroller class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addSample	Public	void	Sample sample	Authorized personnel want to add a sample.	Sample will be submitted and stored in the database.
editSample	Pubic	void	sampleId:String	Sample stored in database.	Sample will be displayed for editable purpose.
viewSample	Public	void	sampleId:String	Sample stored in database.	Sample will be displayed for readable purpose.
deleteSample	Pubic	void	sampleId:String	Sample stored in database.	Sample will be deleted and removed from database.

Table 30 Operation description of sampleController class

4.10 TaskController Class

TaskController
#status:String #taskName:String #labDesignatedNo:String
+startTask(taskName:String, labDesignatedNo:String):void +submitTask(taskName:String, labDesignatedNo:String):void

Table 31 TaskController class

Attribute	Type	Visibility	Invariant
status	String	protected	status<>NULL shouldn't contain special characters and integers and contain 8-20 letters
taskName	String	protected	taskName<> NULL should contain 10-15 characters
labDesignatedNo	String	protected	labDesignatedNo<>NULL should contain eight numbers.

Table 32 Attribute description of TaskController class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
startTask	Public	void	taskName:String, labDesignatedNo:String	Authorized personnel want to set status to started.	Status will be started for the task.

submitTask	Pubic	void	taskName:String, labDesignatedNo:String	Task is done.	Task will be deleted and removed from the database.

Table 33 Operation description of TaskController class

4.11 DepartmentController Class

DepartmentController
#departmentId:String
+addDepartment(Department department):void +editDepartment(departmentId:String):void +viewDepartment(departmentId:String):void +deleteDepartment(departmentId:String):void

Table 34 DepartmentController class

Attribute	Type	Visibility	Invariant
departmentId	String	protected	departmentId<> NULL should contain first letter followed by five numbers.

Table 35 Attribute description of DepartmentController class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addDepartment	Public	void	Department department	Authorized personnel want to add a department.	Department will be submitted and stored in

					the database.
editDepartment	Pubic	void	departmentId :String	Department stored in database.	Department will be displayed for editable purpose.
viewDepartment	Public	void	departmentId :String	Department stored in database.	Department will be displayed for readable purpose.
deleteDepartment	Pubic	void	departmentId :String	Department stored in database.	Department will be deleted and removed from database.

Table 36 Operation description of DepartmentController class

4.12 SampleToTestController Class

SampleToTestController
#labDesignatedNo:String
+addSampleToTest(SampleToTest smapleToTest):void +editSampleToTest(labDesignatedNo:String):void +viewSampleToTest(labDesignatedNo:String):void +deleteSampleToTest(labDesignatedNo:String):void

Table 37 SampleToTestController class

Attribute	Type	Visibility	Invariant
checkOutNo	String	protected	checkOutNo<> NULL should contain first letter followed by four numbers.

Table 38 Attribute description of SampleToTestController class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addSampleToTest	Public	void	SampleToTest smapleToTest	Authorized personnel want to add a sample that is going to be tested.	Sample to be tested will be submitted and stored in the database.
editSampleToTest	Pubic	void	labDesignatedNo:String	Sample to be tested stored in database.	Sample to be tested will be displayed for editable purpose.
viewSampleToTest	Public	void	labDesignatedNo:String	Sample to be tested stored in database.	Sample to be tested will be displayed for readable purpose.
deleteSampleToTest	Pubic	void	labDesignatedNo:String	Sample to be tested stored in database.	Sample to be tested will be deleted and removed from database.

Table 39 Operation description of SampleToTestController class

REFERENCES

- Understanding MVC June 1 2017 https://blog.codinghorror.com/understanding-mode-view-controller
- MVC June 1 20l7 https://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488
- what is a UML diagram May 31 2017 https://www.smartdraw.com/uml-diagram
- What is sequence diagram May 31 2017 https://www.smartdraw.com/uml-diagram
- class diagram tutorial May 31 2017 https://www.licidchart.com/pages/uml/class-diagram
- sequence diagram https://en.m.wikipedia.org/wiki/Sequence_diagram
- Software-Engineering Ian sommerville
- UML Distilled, Third Edition - Fowler,Martin