**Software Documentation for the Lease Estate**

**Product Name: Lease Estate**

Release Version:

Testing Started:

Testing Completed:

Technical Lead:

Product Owner: BBA

Primary Tester:

Developer:

Types of Test Performed:

**Browser Tested:**

1. Chrome

2. Mozilla Firefox

**Device Tested:**

1. Desktop

2. Laptop

3. Tab

4. Smartphone

**Instance Tested:**

1. Development

2. Testing

3. Production

4. UAT

**Table of Contents**

[Revision History: 4](#__RefHeading___Toc25413_3163193700)

[Design Specification: 4](#__RefHeading___Toc25409_3163193700)

[Abstract: 5](#__RefHeading___Toc25411_3163193700)

[Chapter -1: Introduction 6](#__RefHeading___Toc25415_3163193700)

[1.1 Review of the current state of leasing operation 6](#__RefHeading___Toc25654_3163193700)

[1.2 Motivation 6](#__RefHeading___Toc25656_3163193700)

[1.3 Aims and Objectives 6](#__RefHeading___Toc25658_3163193700)

[1.4 Solution Methodology 7](#__RefHeading___Toc25660_3163193700)

[Chapter -2: Background Study 8](#__RefHeading___Toc25662_3163193700)

[2.1 Lease Management System (LMS) 8](#__RefHeading___Toc25664_3163193700)

[2.2 The reasons why we use this software 8](#__RefHeading___Toc25666_3163193700)

[2.3 User’s view 8](#__RefHeading___Toc25668_3163193700)

[2.4 Lease Control 8](#__RefHeading___Toc25670_3163193700)

[2.4.1 Leaser Information 4](#__RefHeading___Toc25672_3163193700)

[2.4.2 Decisions and steps 4](#__RefHeading___Toc25674_3163193700)

[2.4.3 Lease Holder Relationship 4](#__RefHeading___Toc25676_3163193700)

[2.4.4 Benefits 4](#__RefHeading___Toc25678_3163193700)

[2.4.5 Reports and analysis 5](#__RefHeading___Toc25680_3163193700)

[2.5 LMS features 5](#__RefHeading___Toc25682_3163193700)

[Chapter-3: Software Requirement Analysis 6](#__RefHeading___Toc25684_3163193700)

[3.1 Introduction 6](#__RefHeading___Toc25686_3163193700)

[3.2 Requirement Analysis: 6](#__RefHeading___Toc25688_3163193700)

[3.3 Software Requirement Specifications(SRS) 6](#__RefHeading___Toc25690_3163193700)

[3.3.1 SRS scopes 6](#__RefHeading___Toc25692_3163193700)

[3.3.2 Functional Requirements 6](#__RefHeading___Toc25694_3163193700)

[3.3.3 Non-functional Requirements 7](#__RefHeading___Toc25696_3163193700)

[3.4 Interface Requirement: 7](#__RefHeading___Toc25698_3163193700)

[3.4.1 User Interfaces 8](#__RefHeading___Toc25700_3163193700)

[3.4.2 Hardware Interfaces 8](#__RefHeading___Toc25702_3163193700)

[3.4.3 Software Interfaces 8](#__RefHeading___Toc25704_3163193700)

[3.4.4 Communications Interfaces 8](#__RefHeading___Toc25706_3163193700)

[3.5 SRS Specification Table 8](#__RefHeading___Toc25708_3163193700)

[3.6 Analysis 9](#__RefHeading___Toc25710_3163193700)

[Chapter - 4: System design 11](#__RefHeading___Toc25712_3163193700)

[4.1 Algorithm Design 11](#__RefHeading___Toc9252_3251384445)

[4.2 Process Flow Diagram: 12](#__RefHeading___Toc51225_4288561874)

[Chapter -5: Testing Documentation 13](#__RefHeading___Toc25734_3163193700)

[5.1 Introduction 13](#__RefHeading___Toc25736_3163193700)

[5.2 Scope 13](#__RefHeading___Toc25738_3163193700)

[5.3 Quality Objective 13](#__RefHeading___Toc25740_3163193700)

[5.4 Testing Methodology 14](#__RefHeading___Toc6771_1593851831)

[5.5 Testing web pages with Result 15](#__RefHeading___Toc9254_3251384445)

# Revision History:

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Design Specification:

**Overview:**

Estate Management is the important part of Management Control System(MCS). This Estate Management contains two parts. One is server side and the other is client side. For controlling this environment, there are front-end and back-end section. In each section, there are used a multiple number of components to handle various management purposes.

**The back-end section includes the followings tools and components:**

1. Express.js

2. MS Server Version – 2.0

3. DB → Oracle

**The front-end section includes the followings tools and components:**

1. React.js

2. Bootstrap champ

3. Domain → Friendly

# Abstract:

Estate Lease Management System defines the state of the entire part of Lease information which contains several terms of lease like Lease holder and their validation whether the lease is available or not. For Lease management procedure, there should be created lease category to summarize the lease information. After completion of each lease, a report will be generated to get idea about the overall lease status. This system can provide all of the demands in which one can take a part of controlling the management of the Estate Lease System. As the Lease System connects to several portion of the project, the primary objective is to build a control panel that can manage the Lease information, update them according to the requirements and cover the area that is connected to the Lease Information.

# Chapter -1: Introduction

Lease Management System clarifies the current state of the Lease Information.

It provides a user to operate some actions on the lease. In this section, our system can control the each process and execute the multiple actions. All of the lease ingredients connects each other to perform several tasks. An amount of components create integrity, scalability, confidentiality in each function of the Lease System.

## **1.1 Review of the current state of leasing operation**

Leasing operation in the current scenario, most the companies follow categorical order to find an absolute an optimal output in which they can extract the best benefit from these. In our Lease System, we also follow these systems, but besides those we allow several types of holders available as well as lease availability.

## **1.2 Motivation**

Leasing can be done manually earlier. But in the recent era, software can make it easy. So, such types of system having hardware and software components make it easier. So, this kind of thinking make us to move on the real time expected scenario about Lease Estate Information.

## **1.3 Aims and Objectives**

Some goals are focused which are mainly have to be achieved.

• Platform Independent

• Auto Synchronizing

• Friendly Environment

• Domain and Hosting

• Lease Control System

• Leaser History

• Generating Report

• Add-nos

• multi-functional lease

## **1.4 Solution Methodology**

In this section, analyzing the requirement and find solutions by following waterfall method. finding some alternative solution for reducing cost and time lost with efficient way. Implementation is done by using Bootstrap, ODBS, AJAX, and some types of framework like Express.js, Node.js react.js etc .

# Chapter -2: Background Study

This is the system where the controlling system will be formed by a software.

## **2.1 Lease Management System (LMS)**

LMS stands on Lease Management System. This software system allows a user to add lease, manage lease components and automatically create a lease report and lease dashboard.

## **2.2 The reasons why we use this software**

In a huge amount of lease area, leasing is difficult to manage all the things. while using this software the leasing makes more easier than manual process. So, for the big data analysis, this software plays a great role to that environment.

## **2.3 User’s view**

User’s are the key former of the system. So, it needs to handle the user’s con- trolling process while they want any actions in the system. A software system should be built for maintaining the User’s viewing system.

## **2.4 Lease Control**

Without a good controlling system of lease, the management procedure can’t run smoothly. The management of lease controlling provides a better under- standing and perform leasing task orderly. It allows one to operate lease easily.

### **2.4.1 Leaser Information**

Every Lease Holder information will be manage in lease category. This infor- mation can be used by Lease Holder Section.

### **2.4.2 Decisions and steps**

From the following data and relational database, the necessary step should be taken by the user that can carried out Lease information , dashboard and lease Report.

• Manage Category

• Lease Dashboard

• Add Lease

• Lease report

### **2.4.3 Lease Holder Relationship**

Lease Holder type can be formed with different section like that can be brought about from out of the estate lease system. So, A leaser can a broad relationship in the entire system by a specific manner.

### **2.4.4 Benefits**

• Save time - Manually it takes a lot of time while storing the lease information

• Resources - except this software system, the gathering of lease information will be difficult as there need a lot of man power resources in rural and urban areas.

• Reusable - the system can be reused in the similiar types of management.

• Decision making in future - one can make a decision in future whether the system is more better than other.

### **2.4.5 Reports and analysis**

An LMS analysis the lease information like lease type, lease start and expiry date and the pay section. By means of this deed, LMS can generate a report and where user’s can manage it.They can export that as a Document and can mail it to corresponding lease Holder.

## **2.5 LMS features**

• User’s Login

• Lease Dashboard

• Add new lease

• Generate Report over Category

• Manage Category

1. Add Pond

2. Add Lands

3. Add others

4. Add Person

5. Add Institute

6. Add Govt./Others

# Chapter-3: Software Requirement Analysis

## **3.1 Introduction**

Basically it deals with establishing a software for storing lease information.

## **3.2 Requirement Analysis:**

The lease management system requires a high functional requirements. The Function of LMS can be controlled by the user. A user can add, delete, update the lease category.

## **3.3 Software Requirement Specifications(SRS)**

• Documentation (user and administration)

• Operational behavior

• Lease Status

• Technical Support

### **3.3.1 SRS scopes**

In according the lease act, the following information should be provided and stored.

1. Unique User Id

2. Lease Validation for adding lease

3. The total available lease on dashboard

4. Lease Type

5. Lease Management Access

6. ponds, lands, others

### **3.3.2 Functional Requirements**

Every section has a specific functionality that’s are given below:

• Login

• User’s Login

• Dashboard - A user can manage it.

• User’s Role - manage all lease categories.

### **3.3.3 Non-functional Requirements**

For an specific criteria, there should be applied non-functional requirements which are shortly described below:

**Reliability**

The operational interface can’t fail within a period of time.

**Availability**

The system is available or not while a user wants access the information.

**Security**

All user can login to the system with secured procedure. Each user has a unique identification system. Admin can store user’s login information.

**Maintainability**

All source code and related document should be controlled under a version of the system.

**Portability**

As the software run into several platform, the platform independence should be ensure. Different OS can access the software easily.

**Usability**

The Software should be easy to use and given a positive experience by the user’s.

## **3.4 Interface Requirement:**

Computer system including software and hardware can turn the system on by interfacing them with human thought.

### **3.4.1 User Interfaces**

• Login Screen

• Home Screen

• Dashboard

• Lease Category

### **3.4.2 Hardware Interfaces**

• Server Config.

• CPU/Processor

• Windows with apache preloaded

• Client Config.

### **3.4.3 Software Interfaces**

• OS = Windows, Linux

• Language = PHP, JS

• DB = Oracle

### **3.4.4 Communications Interfaces**

These interfaces include E-mail, Web Browser, Server Network and use http or ftp for security purposes.

## **3.5 SRS Specification Table**

It describes necessary behavior of the software system and shows in the table

3.1.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Requirement Name** | **Requirement Classified** |
| 1. | User’s Login |  |
| 2. | Add Lease |  |
| 3. | **Dashboard** |  |
| 4. | Report |  |
| 5. | **Add Category** |  |
| 6. | Add pond |  |
| 7. | And land |  |
| 8. | Add others |  |
| 9. | Add Person |  |
| 10. | Add Institute |  |
| 11. | Add Govt./others |  |

## **3.6 Analysis**

This section defines the requirements of the system and how they are depen- dent or independent to each other. At the end of the phase, the document of the requirement will be provided.

**Priority Checklist**

1. Level-1: Must Be Performed

2. Level-2: Must be perform for next level

3. Level-3: Not Mandatory

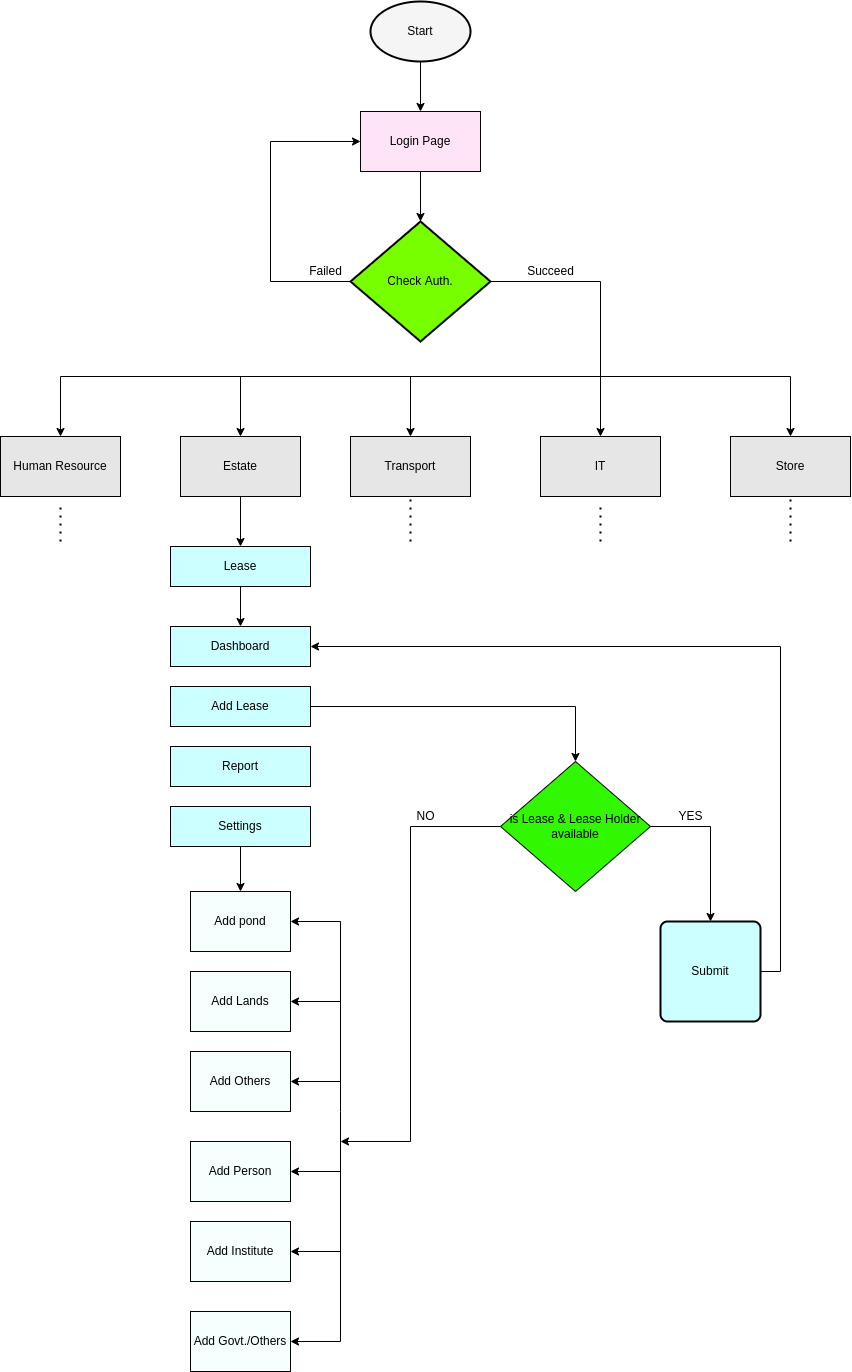
|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | **Requirement Name** | **Functional/Non- functional** | **Priority** |
| 1. | User’s Login | Functional | 1 |
| 2. | Add Lease | Functional | 2 |
| 3. | **Dashboard** | Functional | 3 |
| 4. | Report | Functional | 2 |
| 5. | **Add Category** | Functional | 1 |
| 6. | Add pond | Functional | 1 |
| 7. | And land | Functional | 1 |
| 8. | Add others | Functional | 1 |
| 9. | Add Person | Functional | 1 |
| 10. | Add Institute | Functional | 1 |
| 11. | Add Govt./others | Functional | 1 |
| 12. | Reliability | Non-functional | 1 |
| 13. | Availability | Non-functional | 1 |
| 14. | Security | Non-functional | 1 |
| 15. | Maintainability | Non-functional | 1 |
| 16. | Portability | Non-functional | 1 |
| 17. | Usability | Non-functional | 1 |
| 18. | Database | Functional | 1 |

# Chapter - 4: System design

## 4.1 Algorithm Design

|  |  |
| --- | --- |
| 1. STEP – 1: START 2. STEP – 2: CHECK Authentication 3. STEP – 3: SELECT lease 4. STEP – 4: CALL Fun(X) 5. STEP – 5: if(X = ‘settings’) 6. SELECT Y 7. if(Y = ‘pond’) 8. ADD pond 9. SHOW ponds 10. if(Y = ‘land’) 11. ADD land 12. SHOW lands 13. if(Y = ‘other’) 14. ADD other 15. SHOW others | 1. if(Y = ‘person’) 2. ADD person 3. SHOW persons 4. if(Y = ‘institute’) 5. ADD institute 6. SHOW institute 7. if(Y = ‘govt’ OR ‘other’) 8. ADD govt OR other 9. SHOW govt, others 10. if(X = ‘add lease’) 11. if(lease AND lease holder) 12. ADD newLease 13. if(X = ‘report’) 14. Show report 15. STEP – 6: SHOW dashboard 16. STEP – 7: END |

## 4.2 Process Flow Diagram:



# Chapter -5: Testing Documentation

## **5.1 Introduction**

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities of the product Store Management System of BBA. The plan identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

## **5.2 Scope**

1. In scope:

|  |  |  |
| --- | --- | --- |
| Name | Applicable Roles | Description |
| Add Lease | User | User can add lease for a specific lease Holder. |
| Dashboard | User | Dashboard shows the overall lease status. |
| Report | User | By managing lease, system generate a lease report. |
| Add Category | User | This section can add lease type. |
| Add pond | User | User can Manage Ponds |
| And land | User | User can Manage Lands |
| Add others | User | User can Manage Others |
| Add Person | User | User can Manage Person |
| Add Institute | User | User can Manage Institute |
| Add Govt./others | User | User can Manage Govt./Others |

2. Out scope

• User Interface (UI)

• Hardware

• System security and performance

## **5.3 Quality Objective**

The test objectives are to verify the Functionality of the system Lease Manage- ment System, the project should focus on testing the lease operation such as Add lease, manage lease, all lease info to guarantee all these operation can work normally in real business environment.

## **5.4 Testing Methodology**

**=> Test Models**

• **Waterfall model** – a dynamic approach where each step depends on the previous one, where the developers follow some sequence of steps progressively downwards forwards to achieve the goal. It is like a waterfall approach containing several phases.

=> **White box Testing**

• Tester has access to the full view of code.

• It is known as transparent or glass box testing

• Avoiding errors and wasting time at an early stage

• White box testing contains various tests - (a) Basic Path testing

(b) Loop testing

(c) Condition testing

(d) Memory perspective testing

(e) Performance testing

**=> Test Levels**

• Integration testing

• System testing

• Cross Browser Compatibility testing

• UAT

• API testing

=> **Test Deliverable**

(a) Before Testing

• Test plans document.

• Test case document

(b) After Testing

• Test Results/reports

• Defect Report

**•** Installation/ Test procedures guidelines

=> **Testing Tools**

|  |  |  |
| --- | --- | --- |
| No. | Resource | Description |
| 1. | Server | Oracle Database Server |
| 2. | Network | LAN line. |
| 3. | Computer | PC and other OS configuration |

## 5.5 Testing web pages with Result

**Test Scenario 1: Login Functionality Check**

For All User:

*Positive test case*

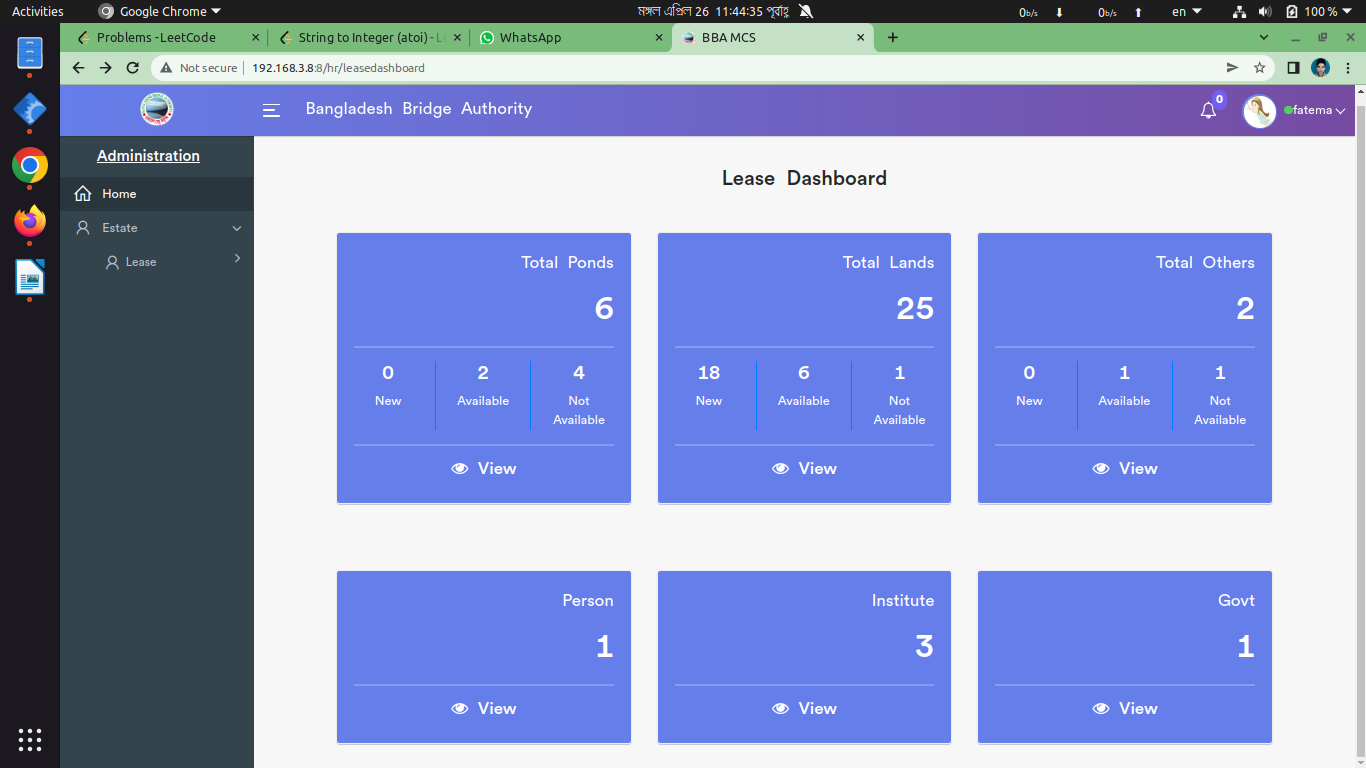
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Login | 1. Visit login page  2. Enter username  3. Enter password  4. Click login  5. Change password (if forgot)  6. Keep login | Data saved successfully | Go to the login page | P |

*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Login | Checked all the necessary given data | Data checked | Login Page | P |

**Test Scenario 2: Lease**

1. Dashboard



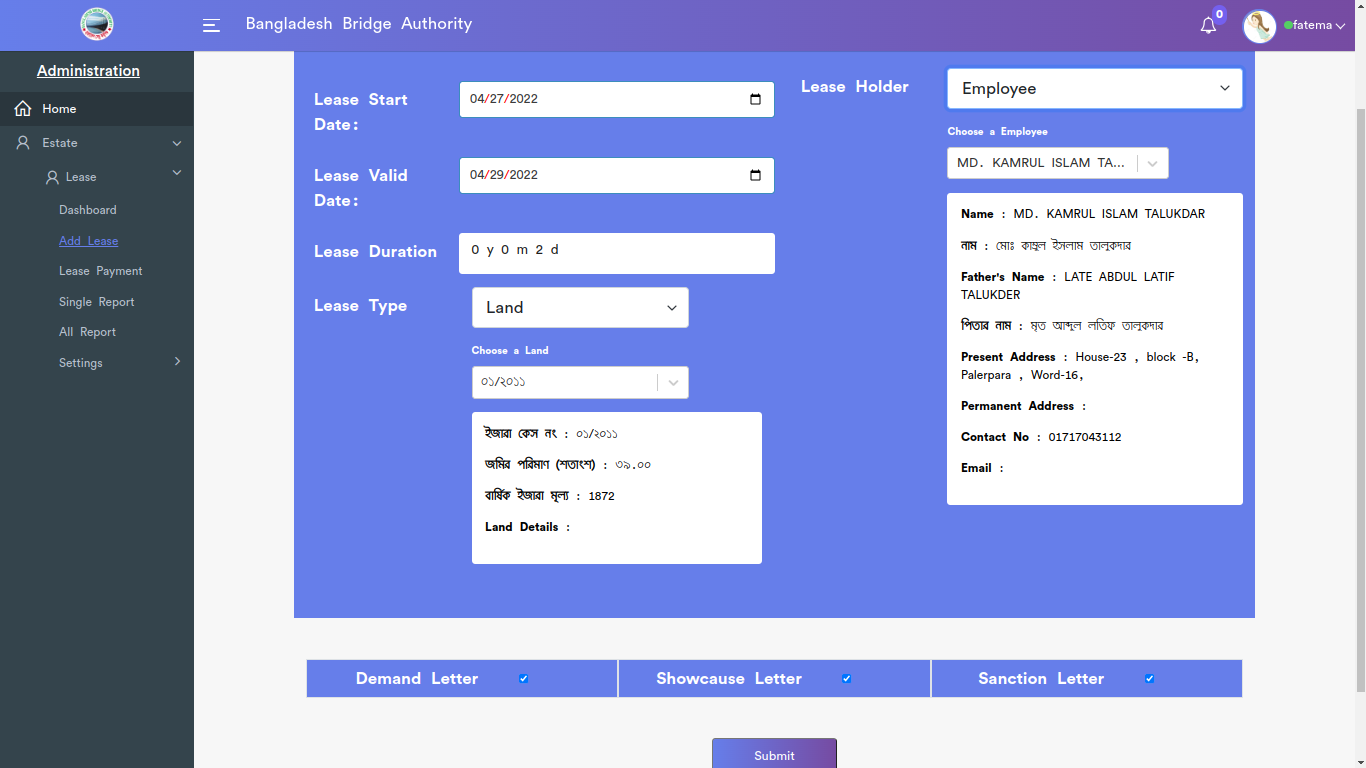
*Positive test case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Total Ponds | Click to view ponds details | Show total ponds | Displayed respective information | P |
| 2 | Total Lands | Click to view lands details | Show total lands | Displayed respective information | P |
| 3 | Total others | Click to view others details | Show total others | Displayed respective information | P |
| 4 | Person | Click to view person’s details | Show total Person | Displayed respective information | P |
| 5 | Institute | Click to view institute details | Show total Institute | Displayed respective information | P |
| 6 | Govt. | Click to view govt details | Show total Govt. | Displayed respective information | P |

*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Total Ponds | Proper information should be displayed | No missing any information | Perfectly done | P |
| 2 | Total Lands | Proper information should be displayed | No missing any information | Perfectly done | P |
| 3 | Total others | Proper information should be displayed | No missing any information | Perfectly done | P |
| 4 | Person | Proper information should be displayed | No missing any information | Perfectly done | P |
| 5 | Institute | Proper information should be displayed | No missing any information | Perfectly done | P |
| 6 | Govt. | Proper information should be displayed | No missing any information | Perfectly done | P |

2. Add lease



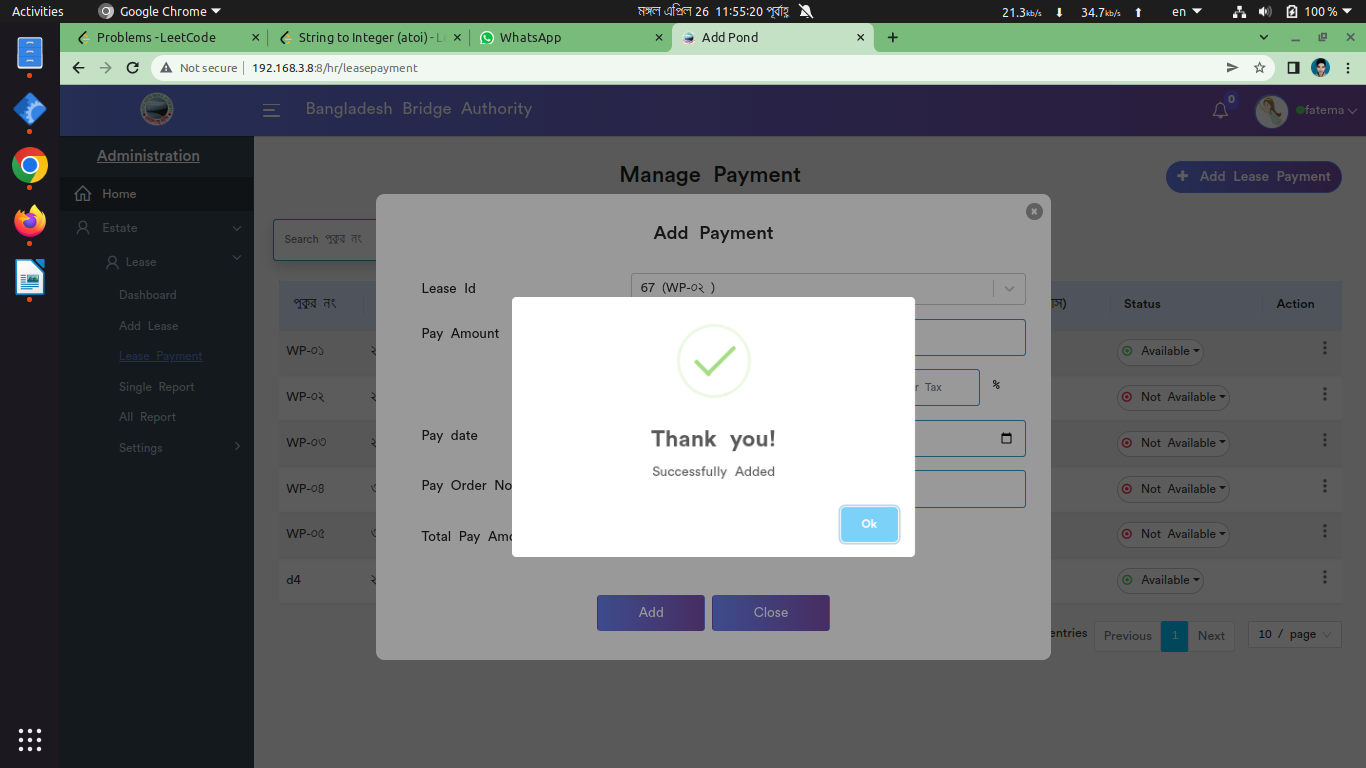
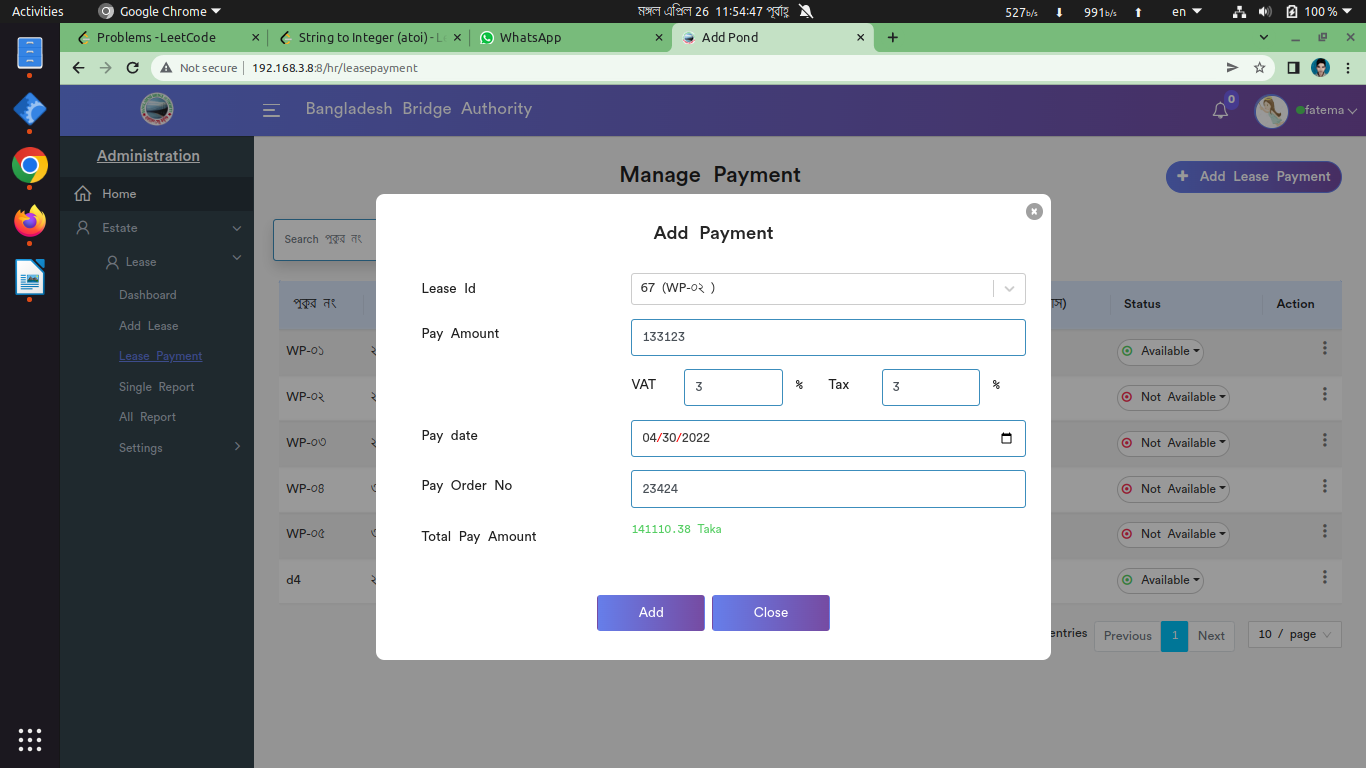
*Positive test case*

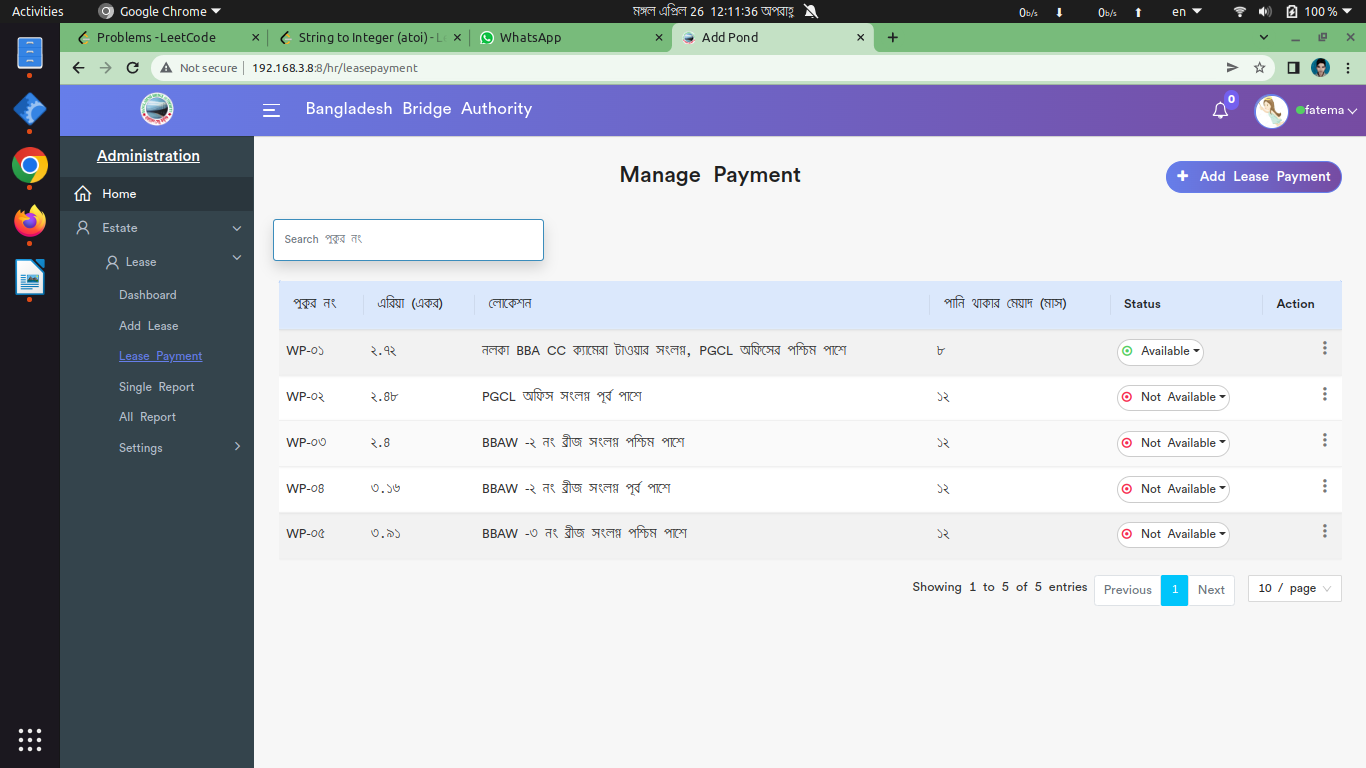
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Lease Information | 1. Select Lease Start Date  2. Select Lease end date  3. select lease type  4. select lease holder  5. Enter total amount  6. Enter paid amount  7. select paid date  8. Enter pay order no  9. click submit | Data saved successfully | Saved in the database | P |

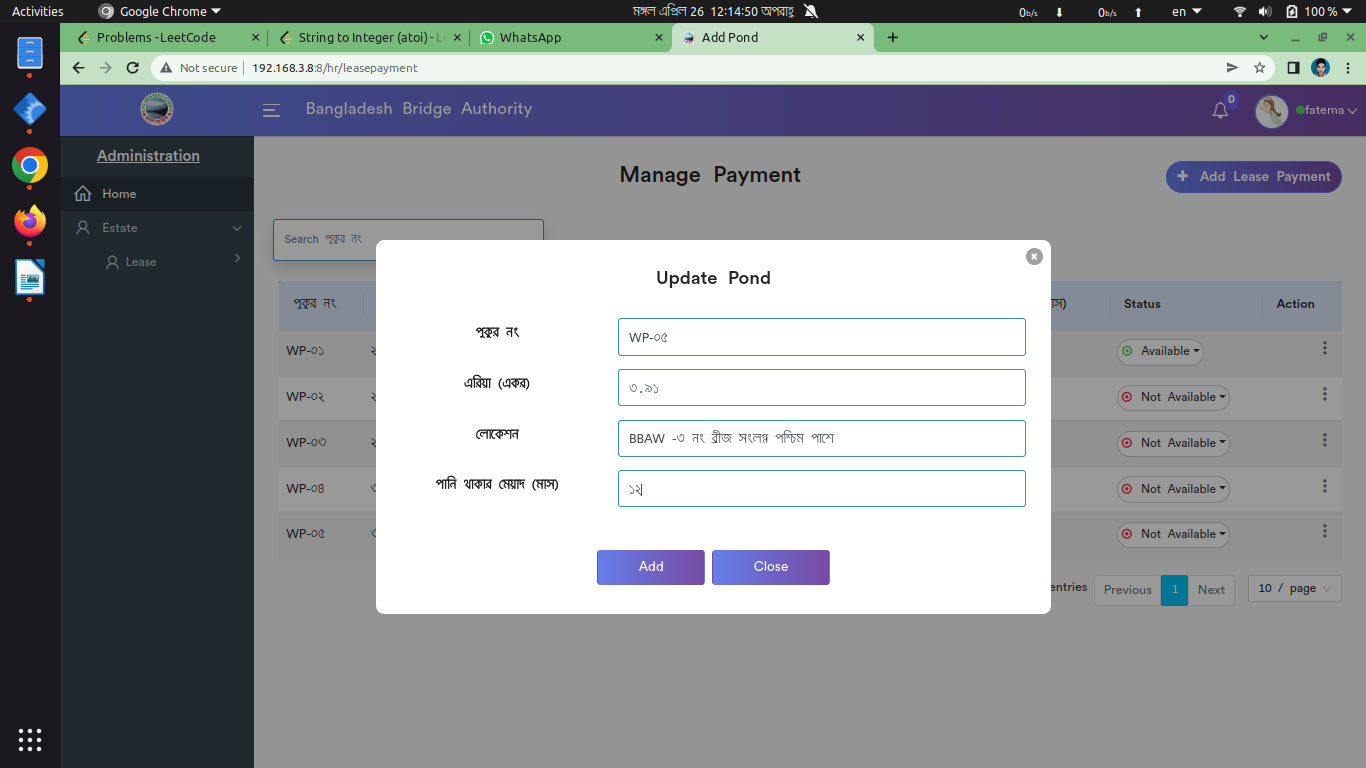
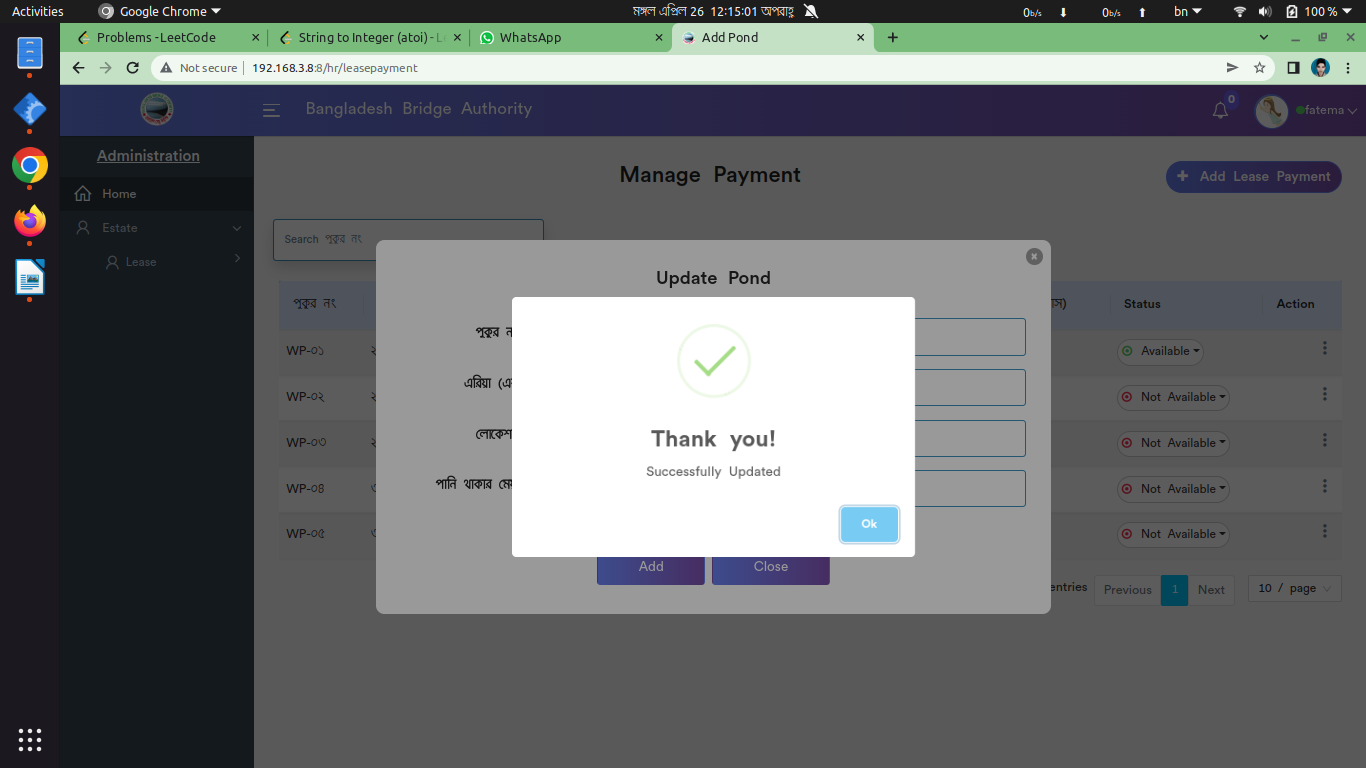
*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Lease Information | 1. Data should properly selected & entered  2. Submit button should perfectly worked | irrelevant information are checked | Works fine | P |

3. Lease Payment







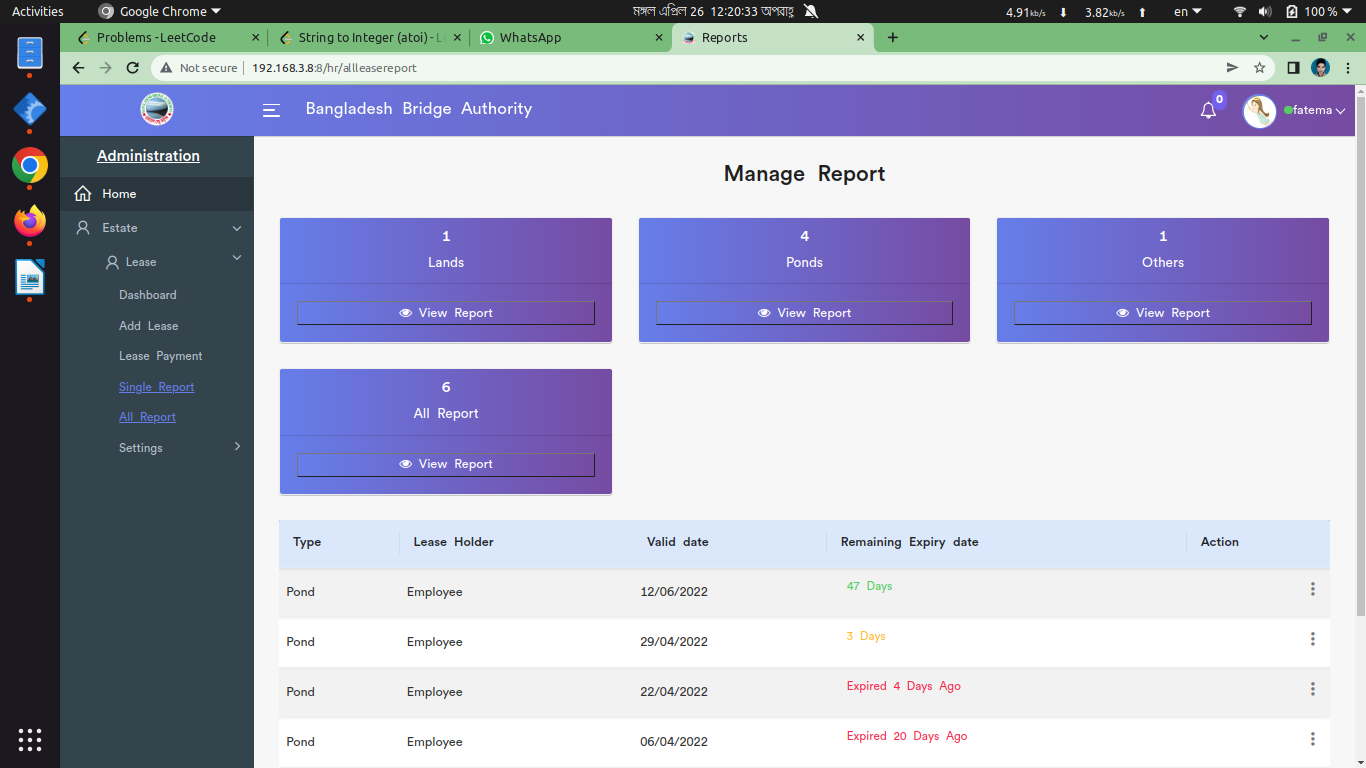
*Positive test case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add Lease Payment | 1. Select Lease Id  2. Enter pay amount  3. Enter vat & tax  4. Select pay date  5. Enter pay order no  6. Click add | Data saved successfully | No error is displayed | P |
| 2 | Manage Payment | Display all the payment details of lease | Displayed all information | No error | P |
| 3 | Update or Delete | 1. Update the existing information of lease payment  2. Delete an specific information | Action button does the specific operation | Works fine & displayed | P |

*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add Lease Payment | All selected information should be valid & Button should perfectly worked | Data are valid and ok | No irrelevant data | P |
| 2 | Manage Payment | Added information should be perfectly displayed | Perfectly all ok | No missing any information |  |
| 3 | Update or Delete | Information should be updated & delete from the database | Operation is perfectly done | No error occurred | P |

4. All report



*Positive test case*

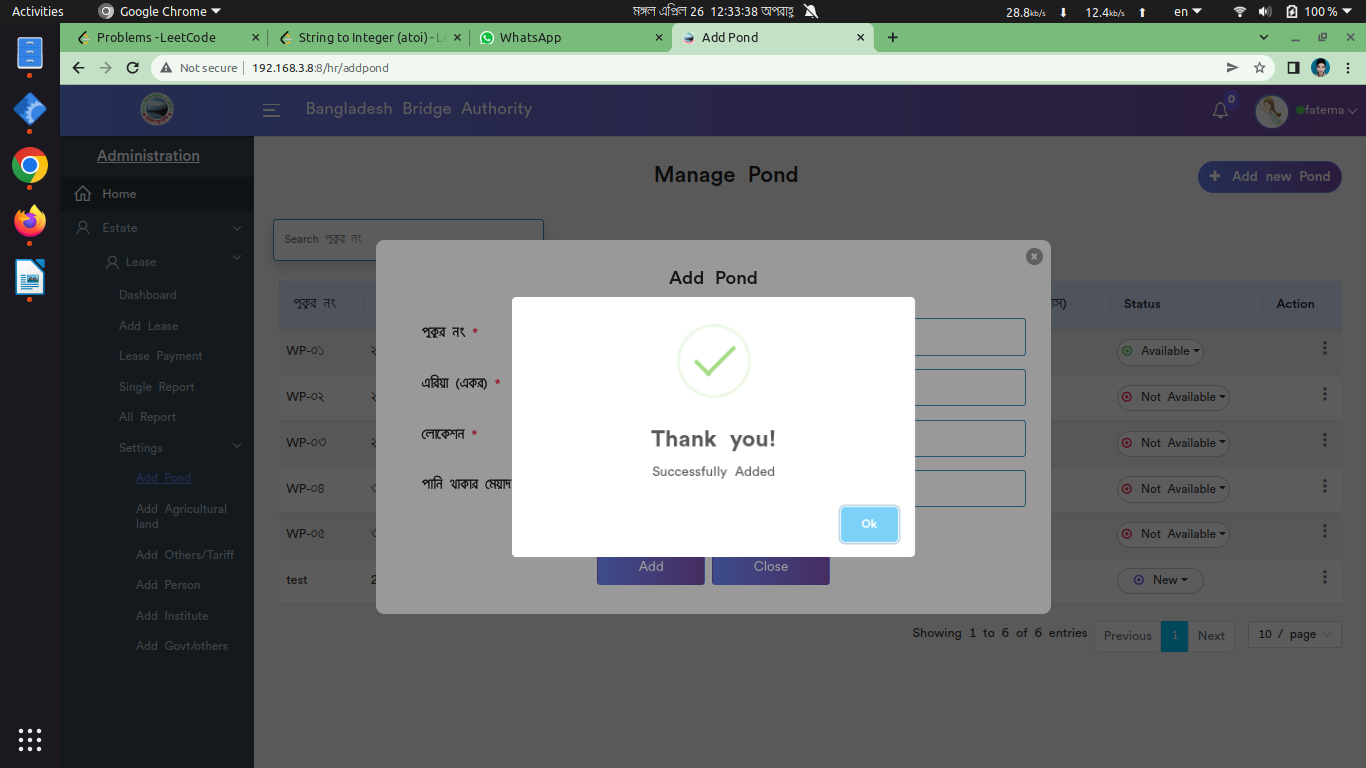
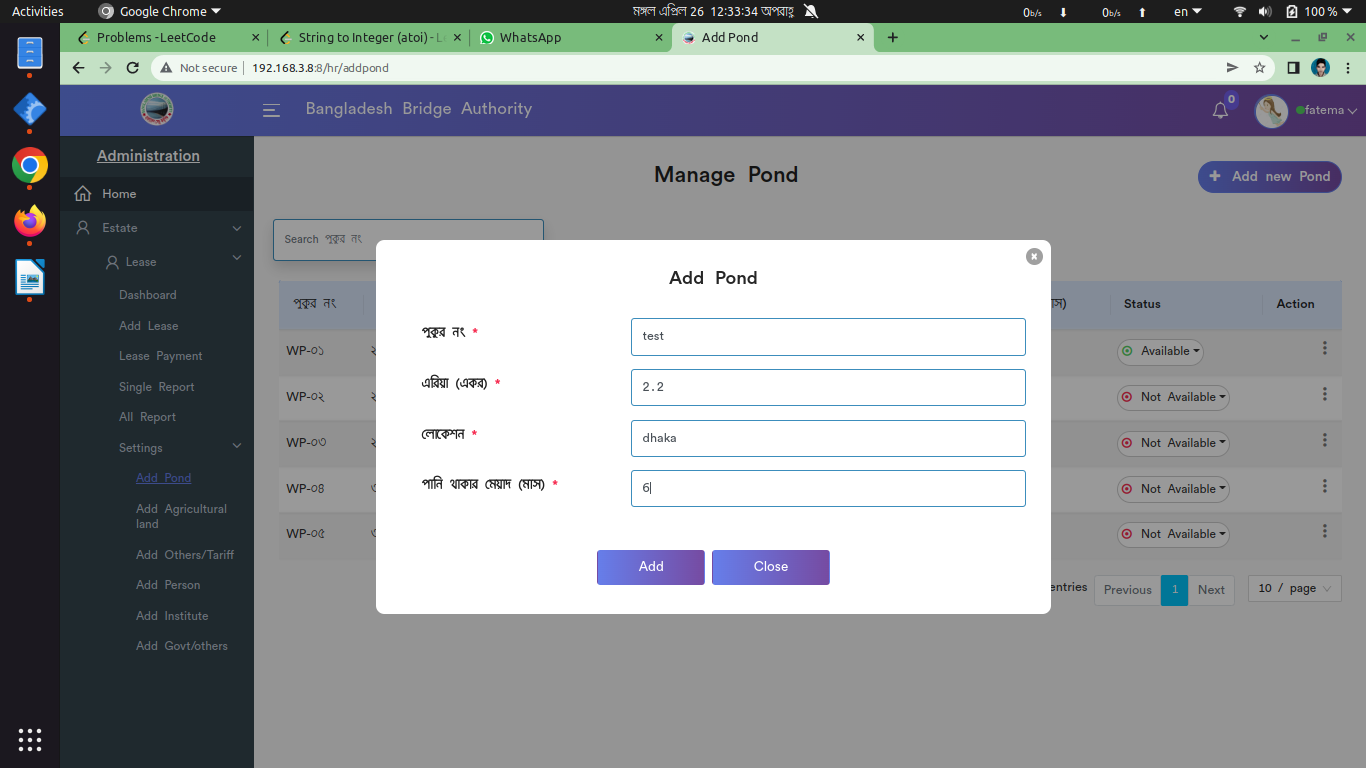
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Manage Report | Display all the lease dashboard | Report is visualized | No missing | P |
| 2 | View Report | Click view report to see corresponding report | Show any report | No data is missing | P |
| 3 | Delete | Delete a specific item from the report | Deleted | Delete form the list successfully | P |

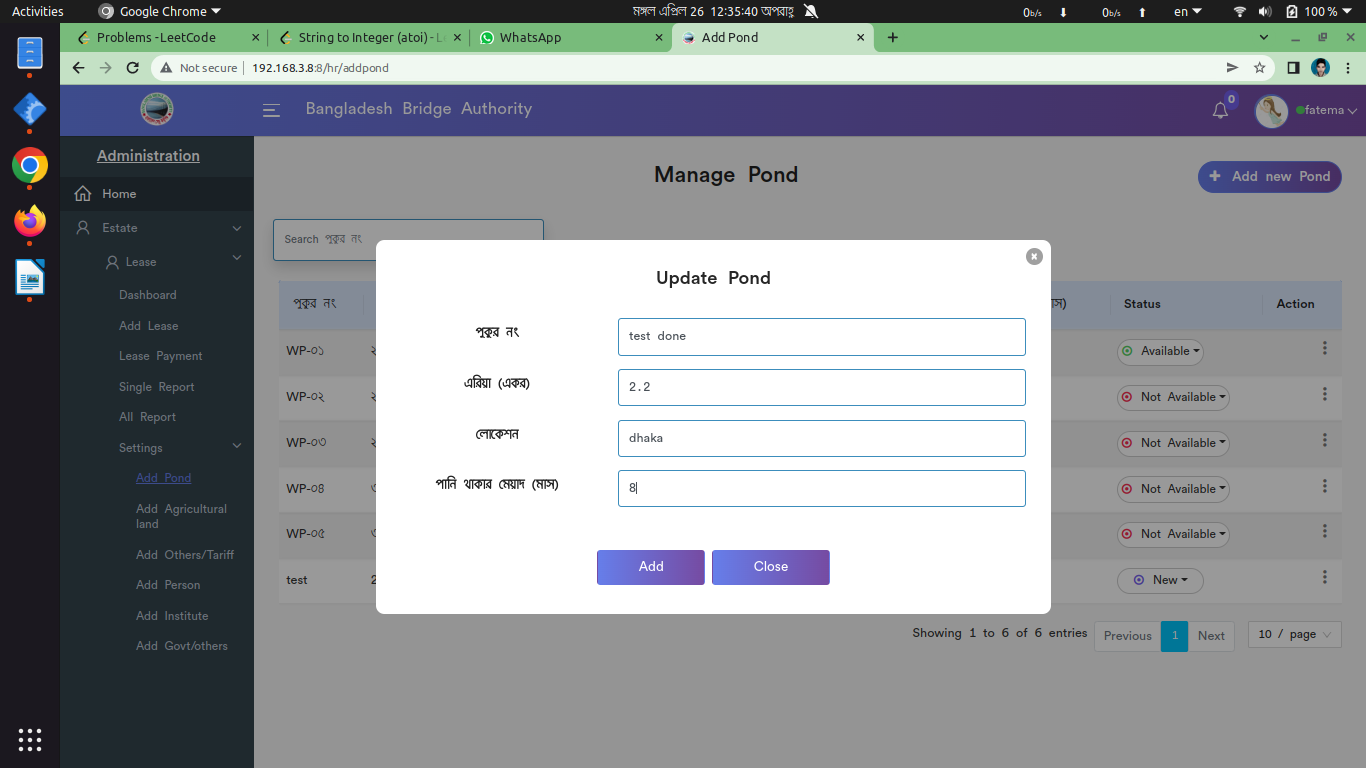
*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Manage Report | No information is missing | Works fine | No error is occurred | P |
| 2 | View Report | Must show the specific report | Works fine | No error is occurred | P |
| 3 | Delete | Delete button must work fine | Works fine | No error is occurred | P |

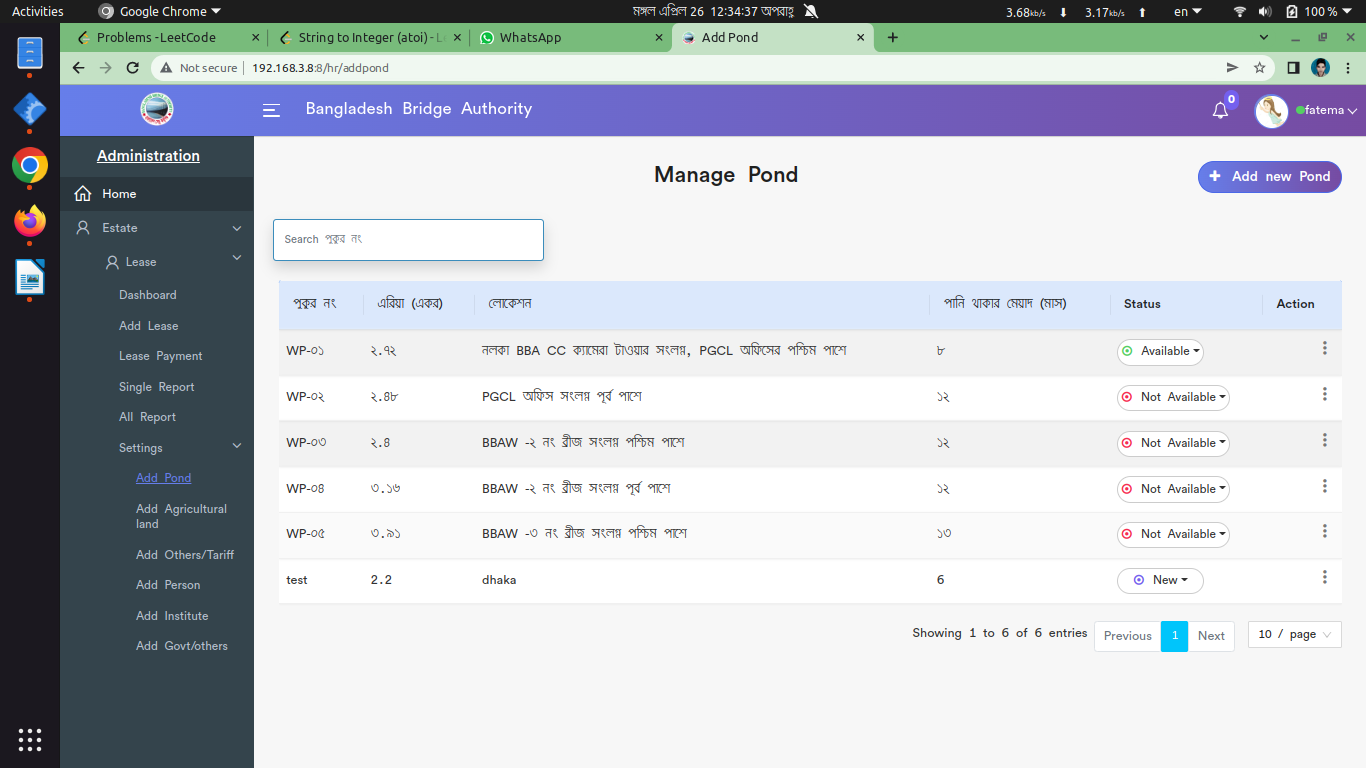
5. Settings

i. Add Pond









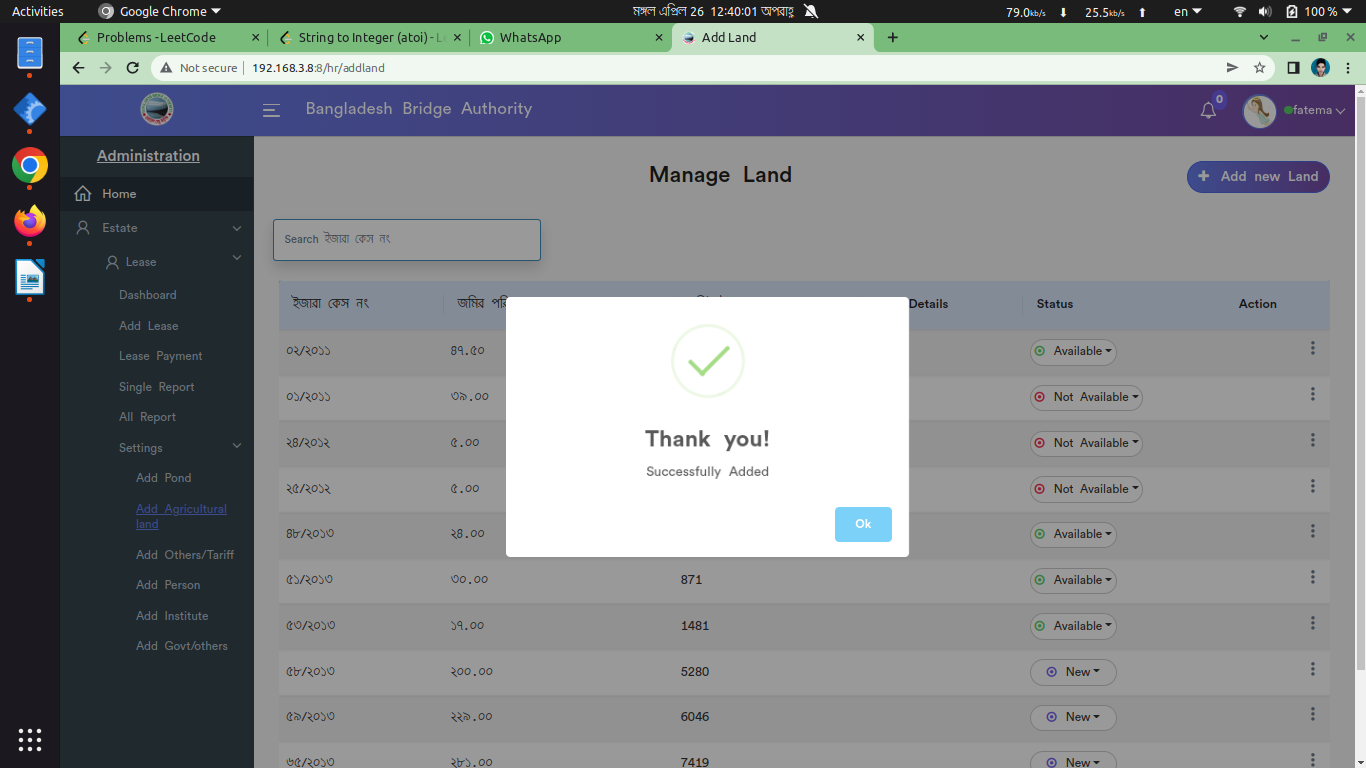
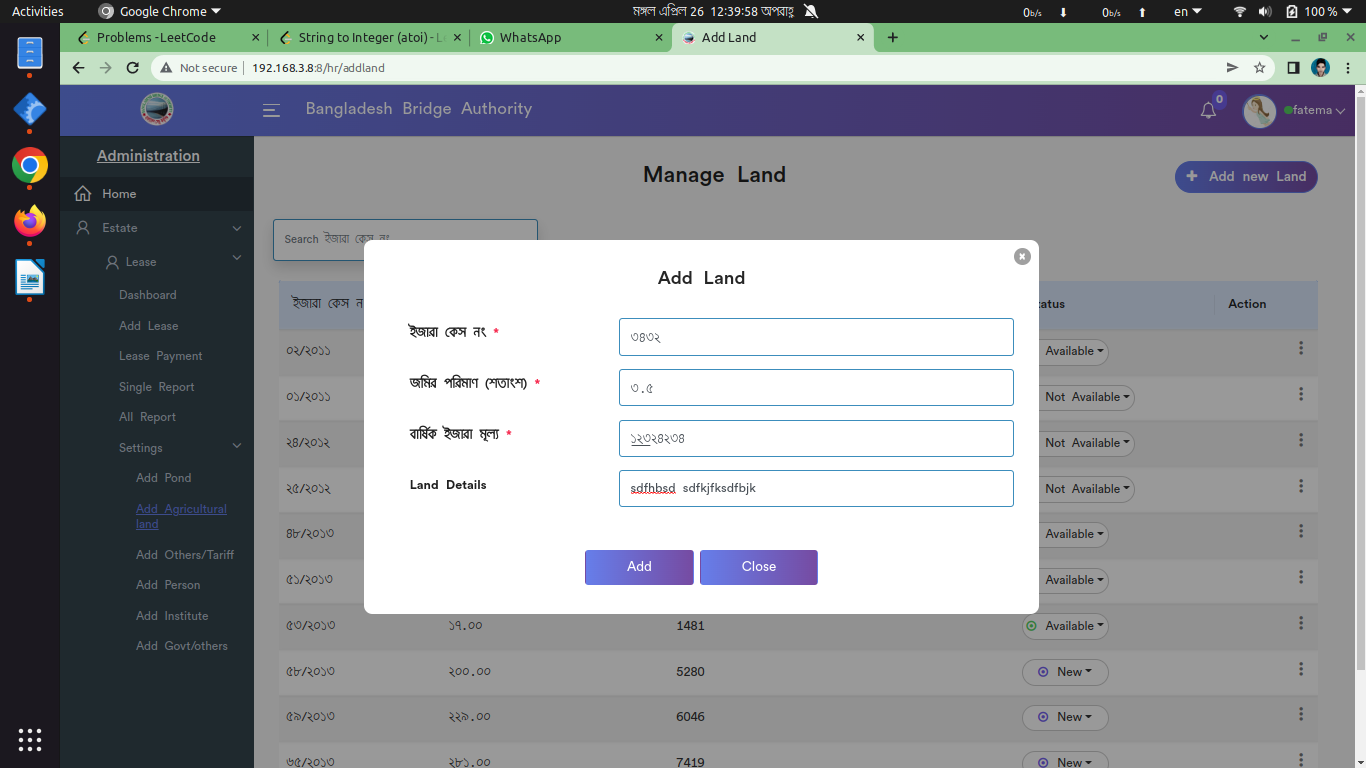
*Positive test case*

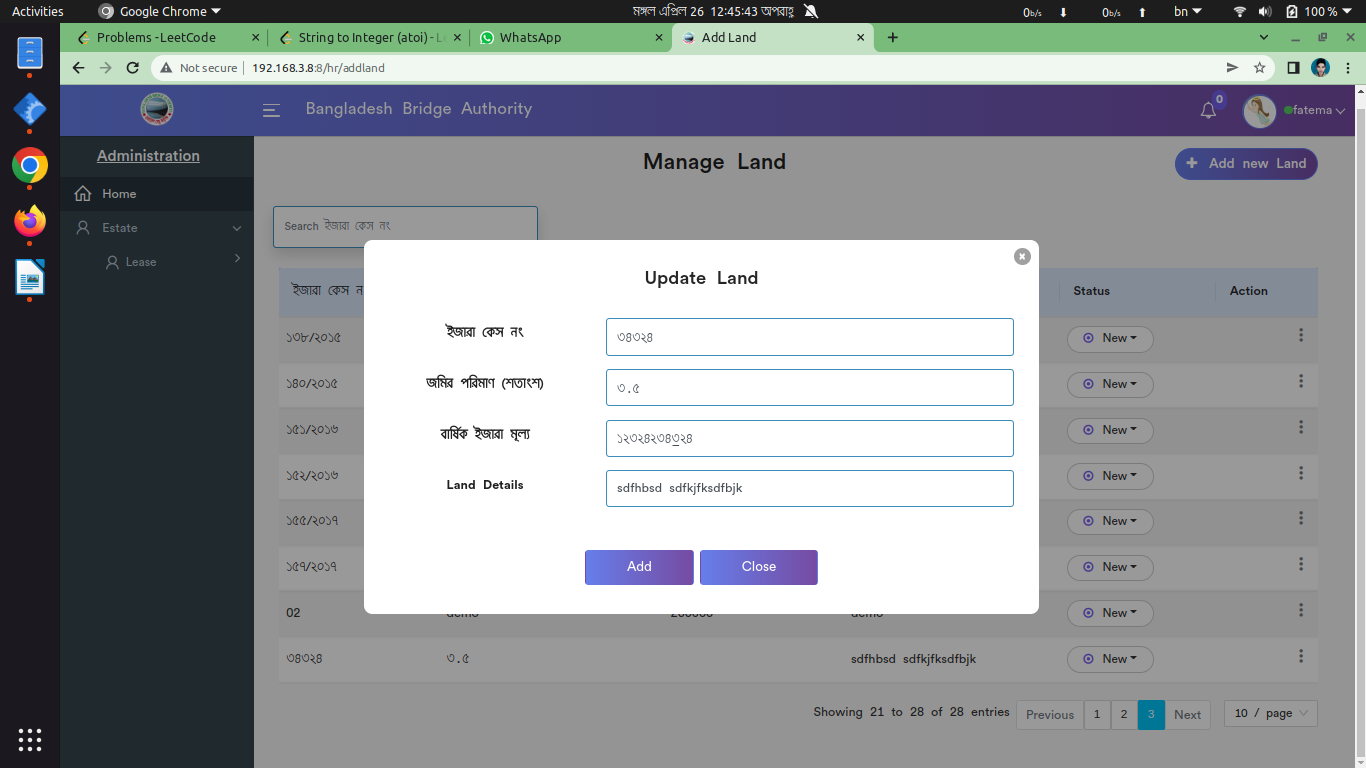
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new pond | 1. Enter পুকুর নং, এরিয়া, লোকেশন, পানি থাকার মেয়াদ  2. click add | Data saved successfully | No error is displayed | P |
| 2 | Manage pond | Display all the pond information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

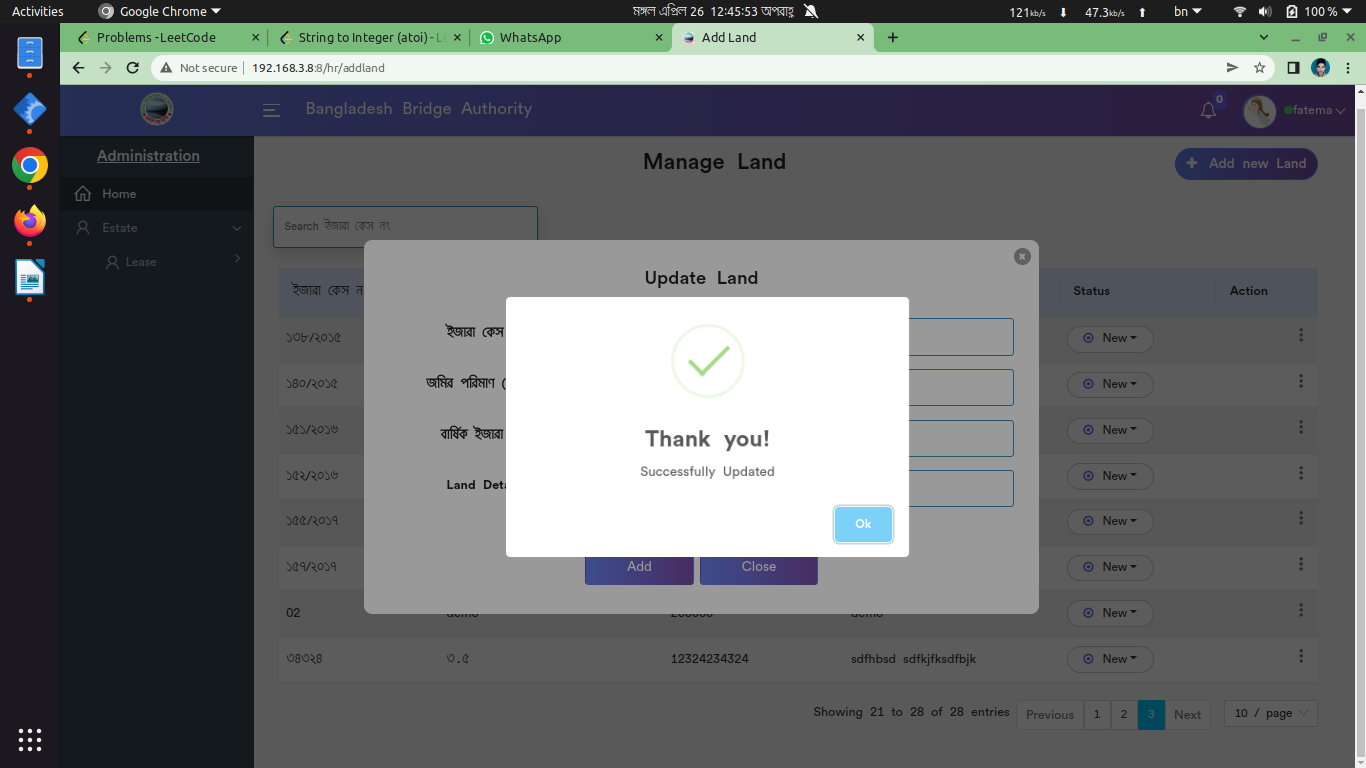
*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new pond | Data should perfectly store and button should work fine | Data are valid and ok | No irrelevant data | P |
| 2 | Manage pond | No information is missing | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |

ii. Add Lands







**

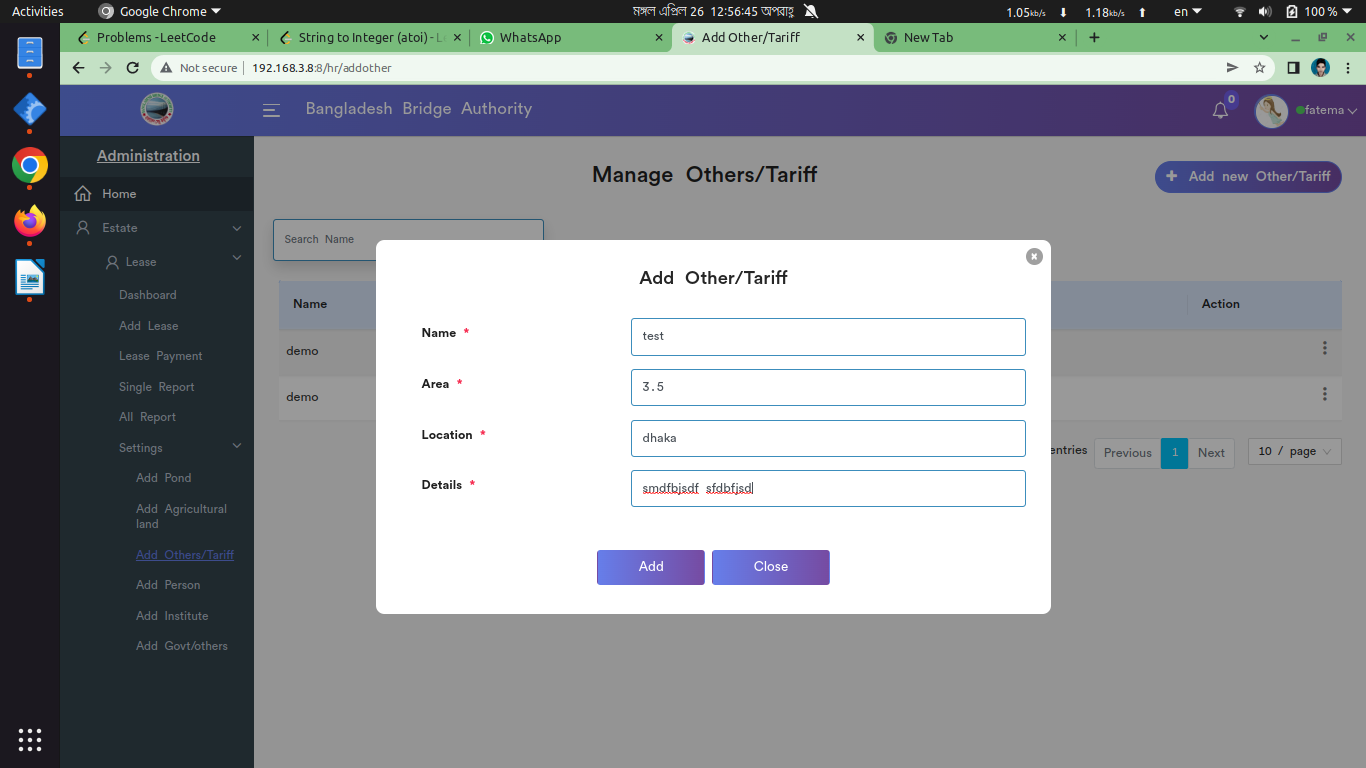
*Positive test case*

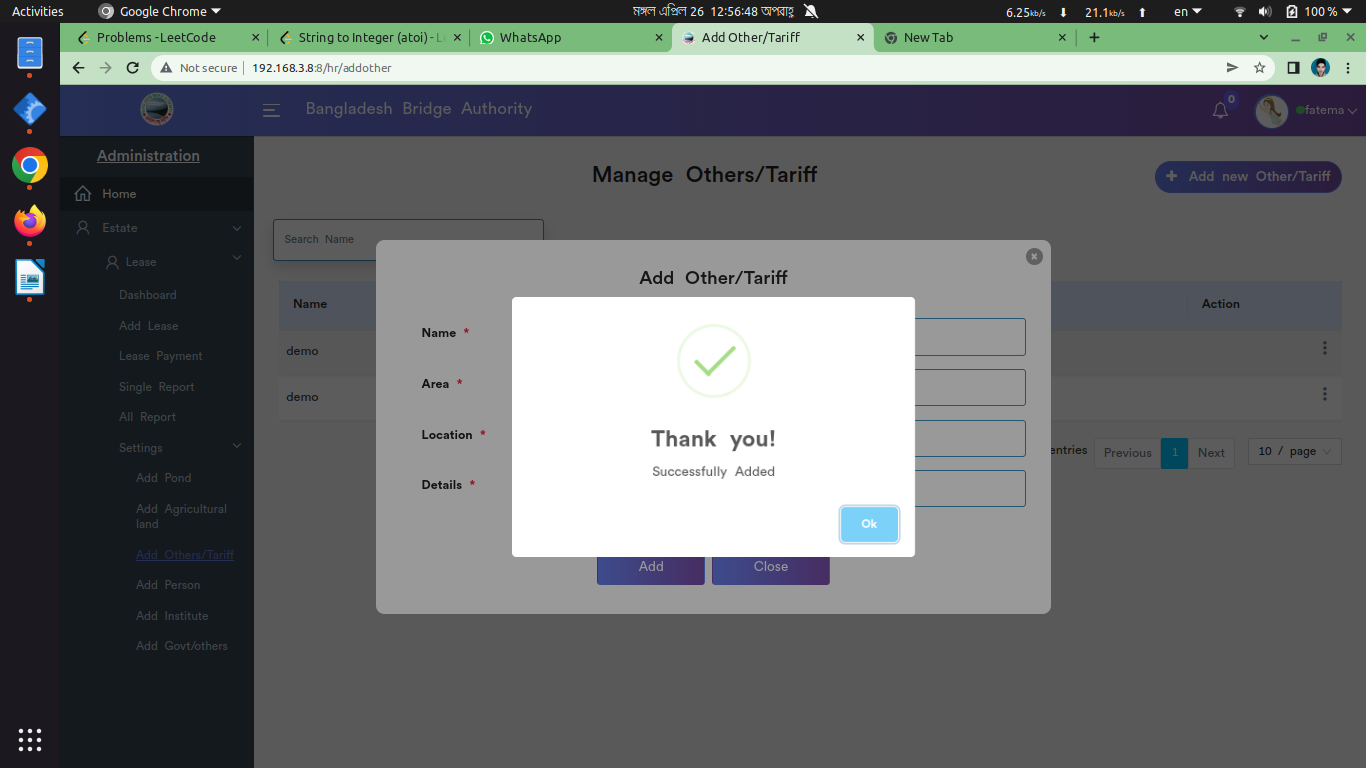
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new land | 1. Enter ইজারা কেস নং, জমির পরিমাণ, বার্ষিক ইজারা মূল্য, land details  2. Click add | Data saved successfully | No error is displayed | P |
| 2 | Manage land | Display all the land information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

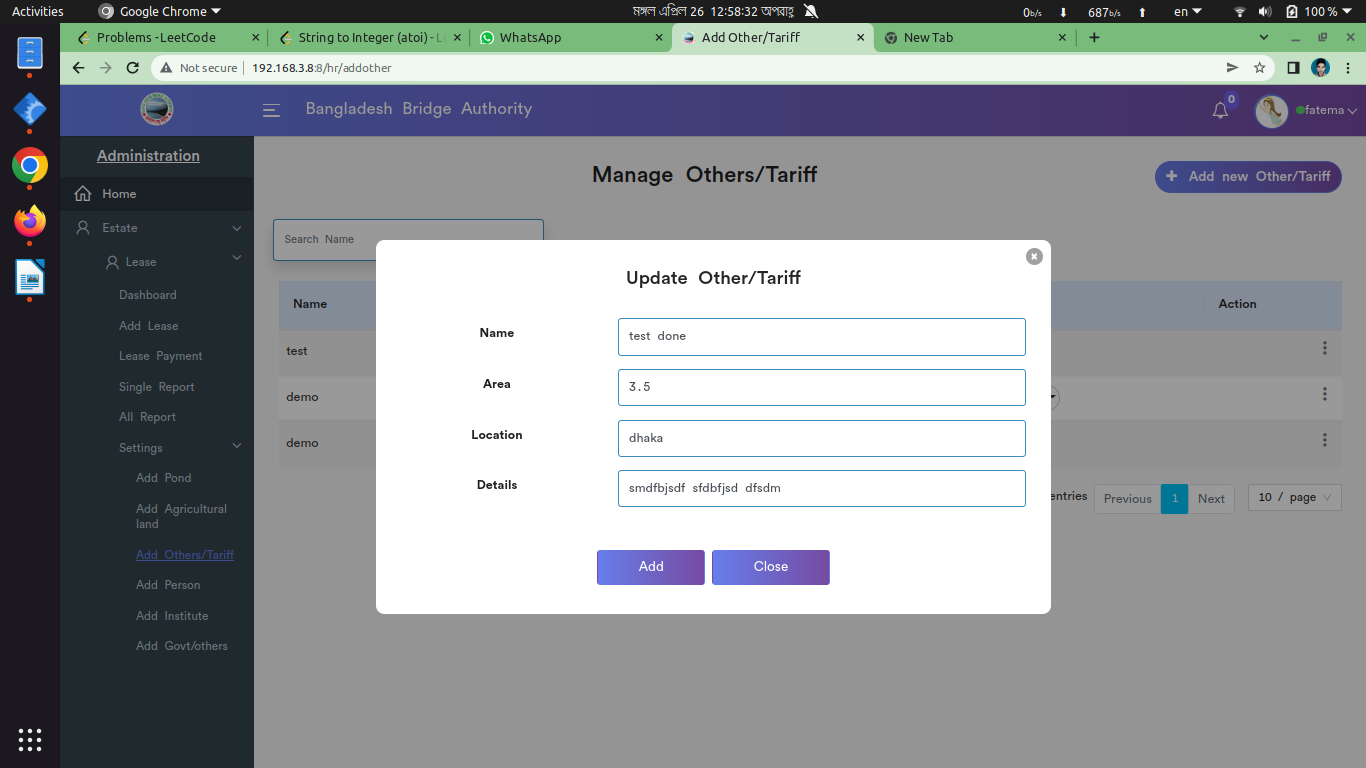
*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new land | 1. Data should be entered  2. add button should perfectly work | Data are valid and ok | No irrelevant data | P |
| 2 | Manage land | Everything should perfectly displayed | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |

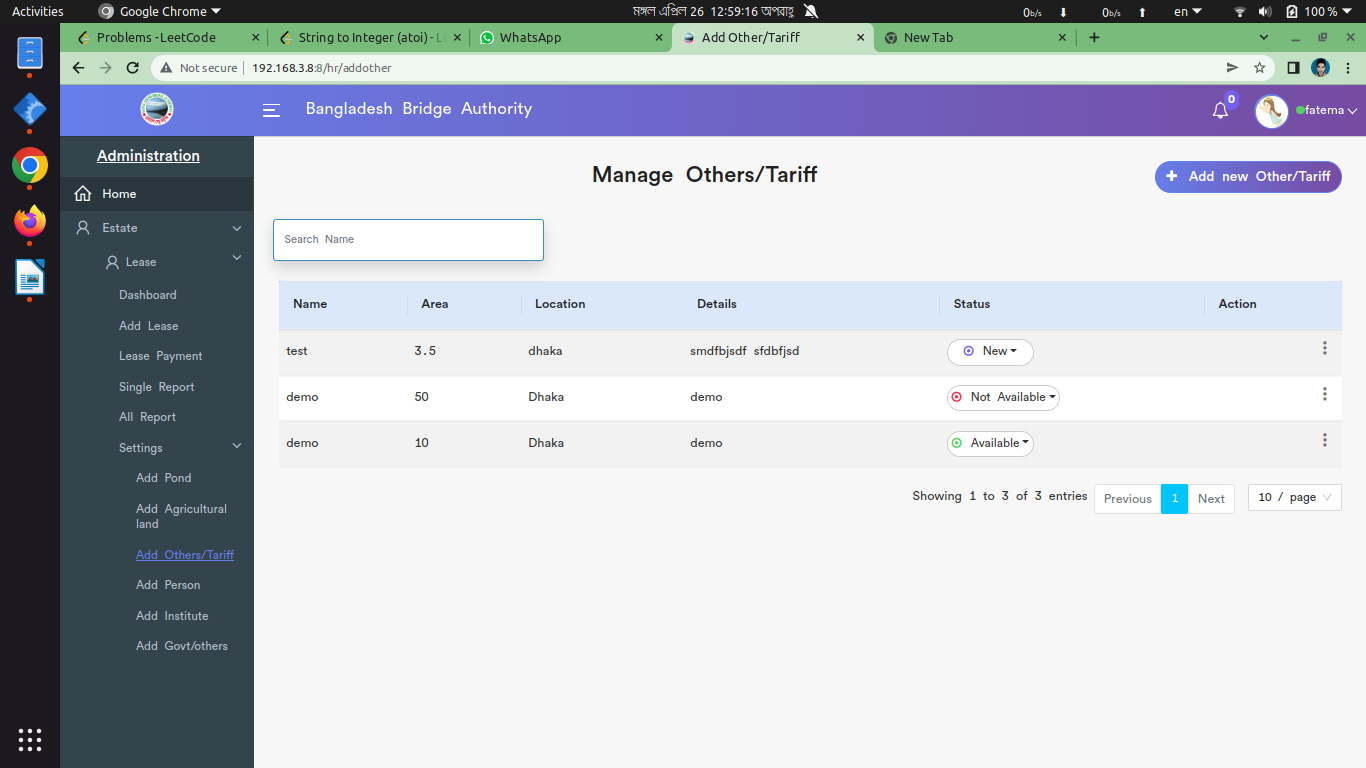
iii. Add others











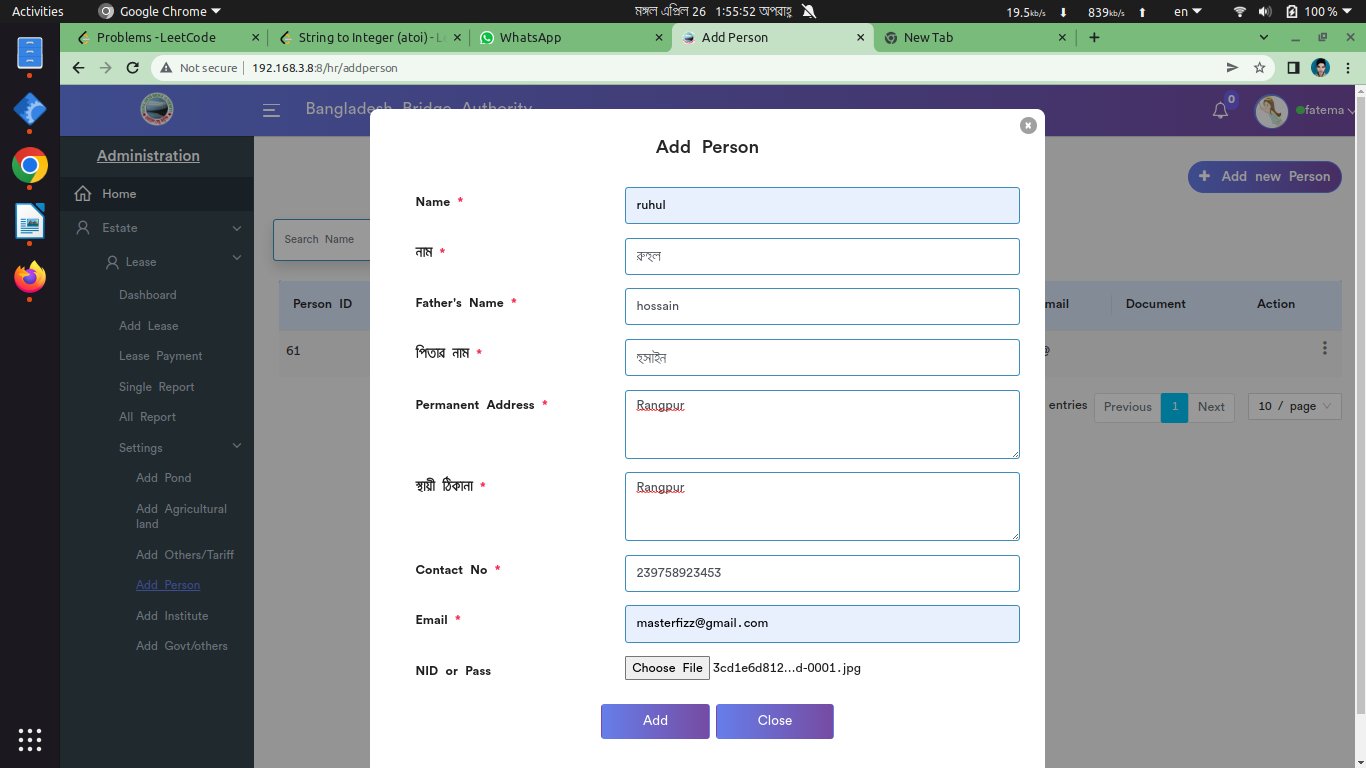
*Positive test case*

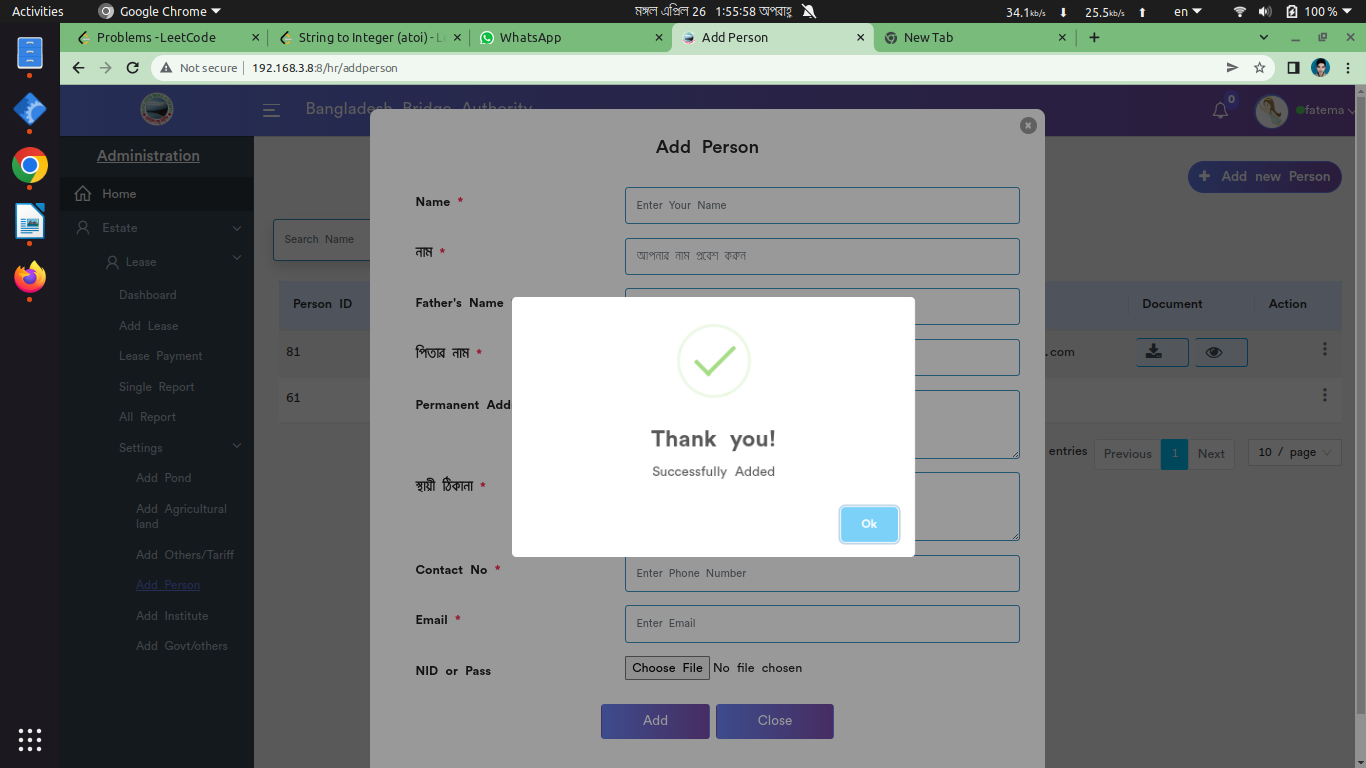
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new others | 1. Enter name, area, location, details  2. click add | Data saved successfully | No error is displayed | P |
| 2 | Manage others | Display all the others information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

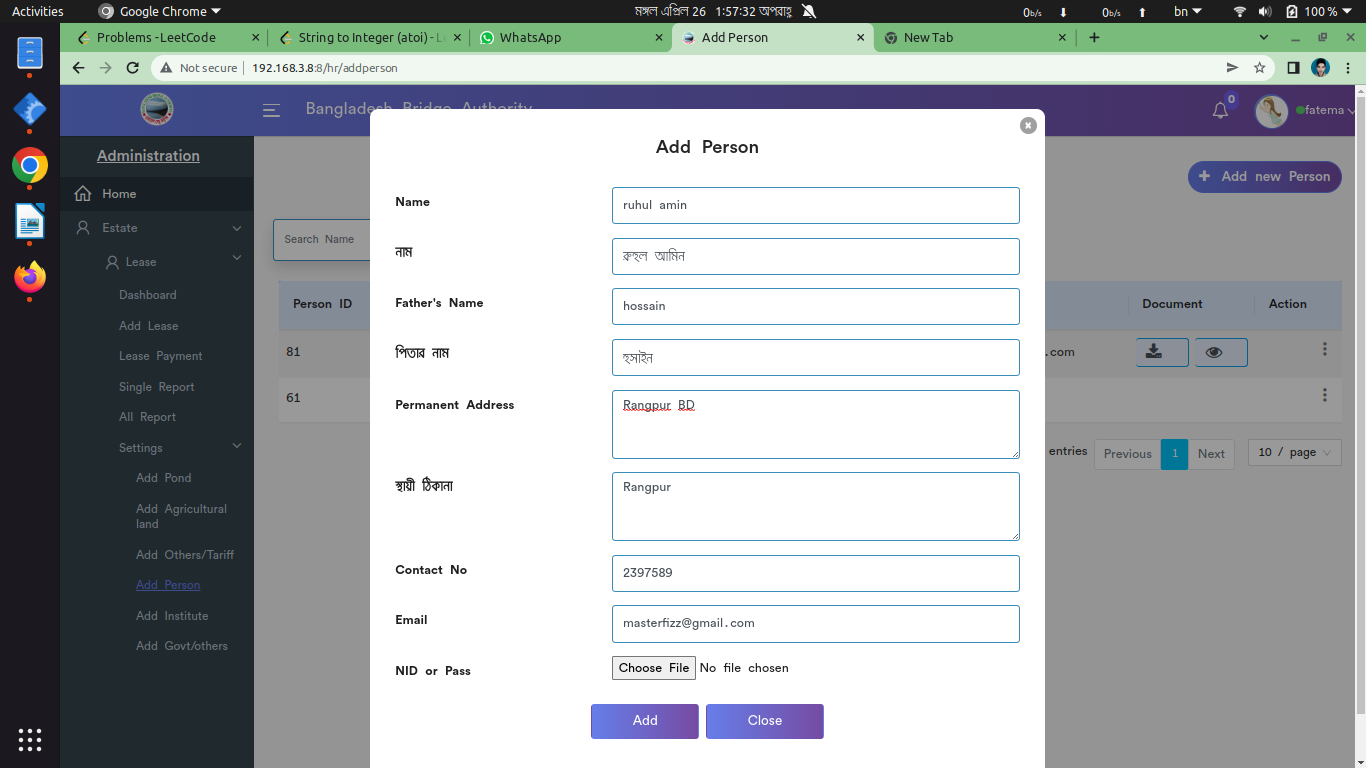
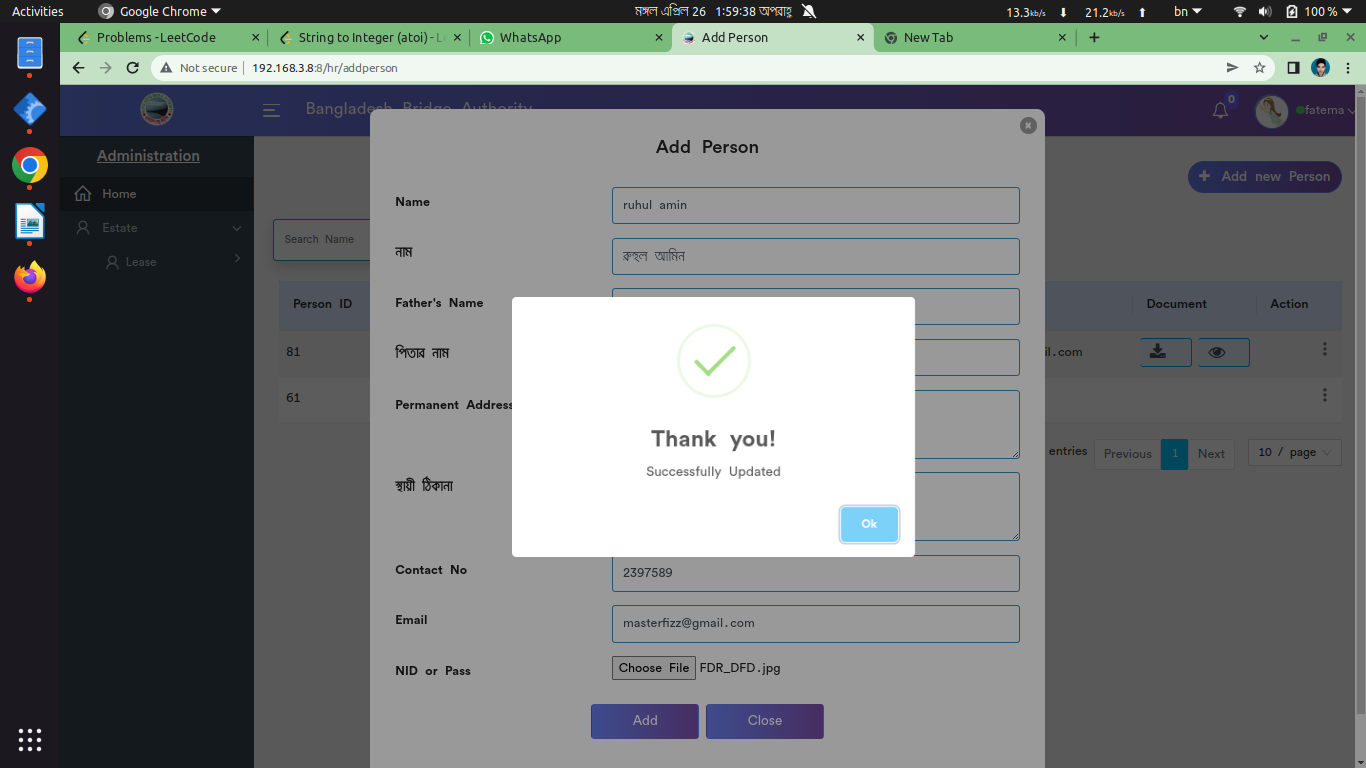
*Negative Test Case*

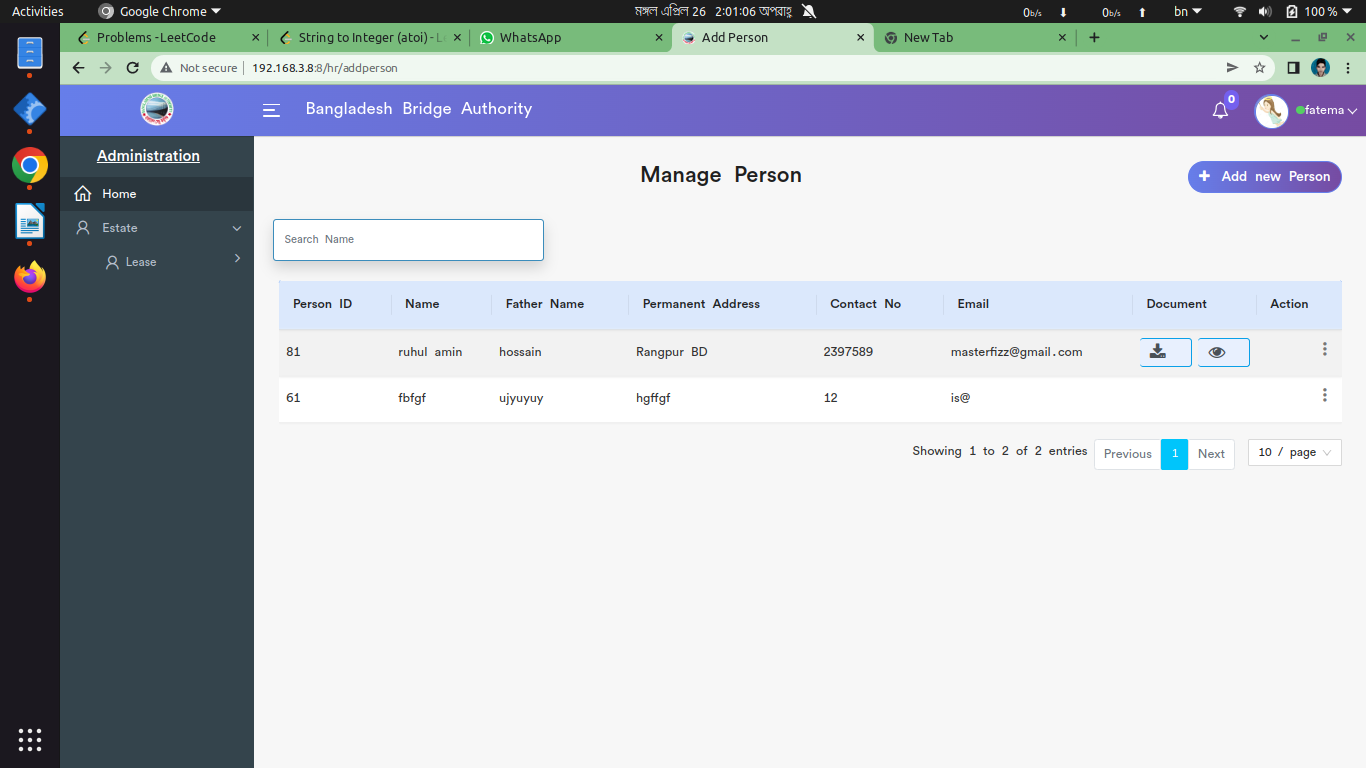
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new others | 1. Data should be entered  2. add button should perfectly work | Data are valid and ok | No irrelevant data | P |
| 2 | Manage others | Everything should perfectly displayed | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |

iv. Add Person









*Positive test case*

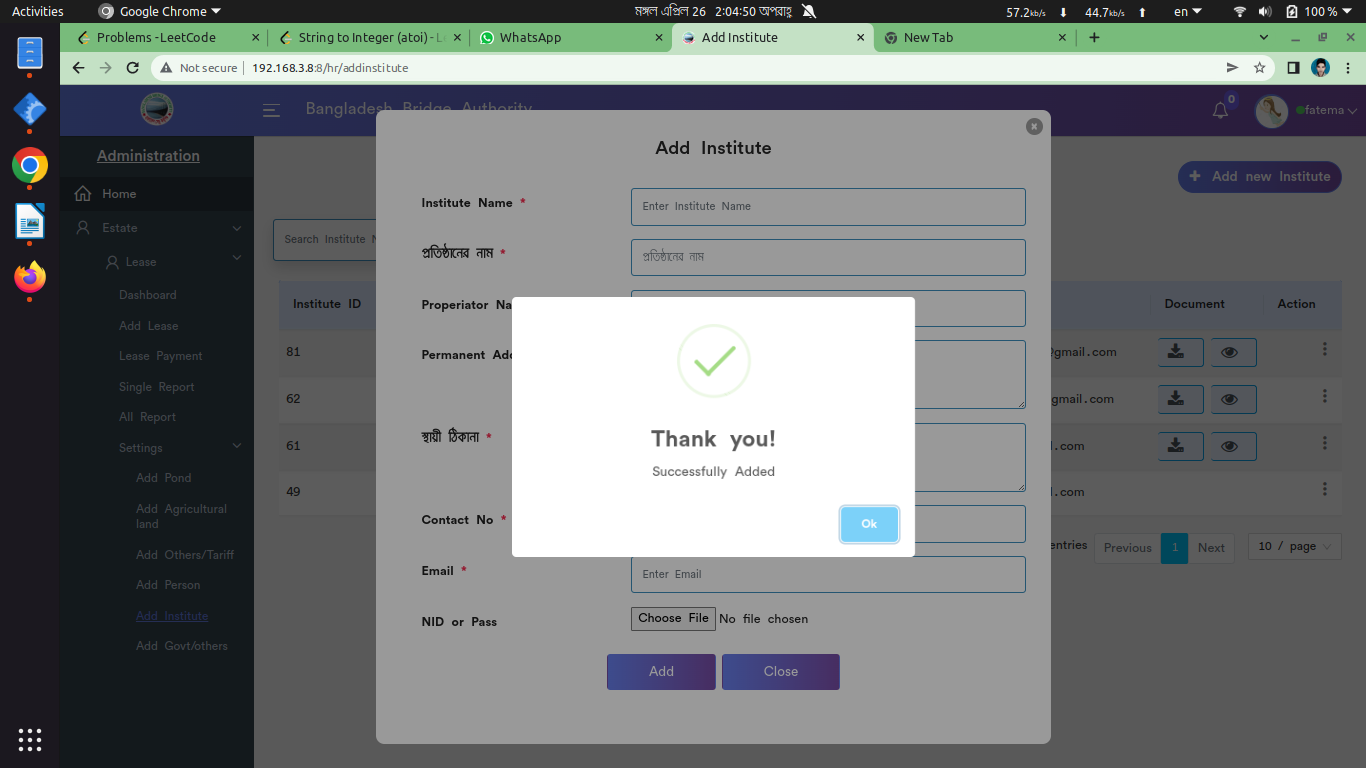
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new person | 1. Enter persons name, persons father’s name, permanent address both in english & bangla  2. Enter contact info and email id  3. upload NID or passport information  4. Click add | Data saved successfully | No error is displayed | P |
| 2 | Manage person | Display all the person’s information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

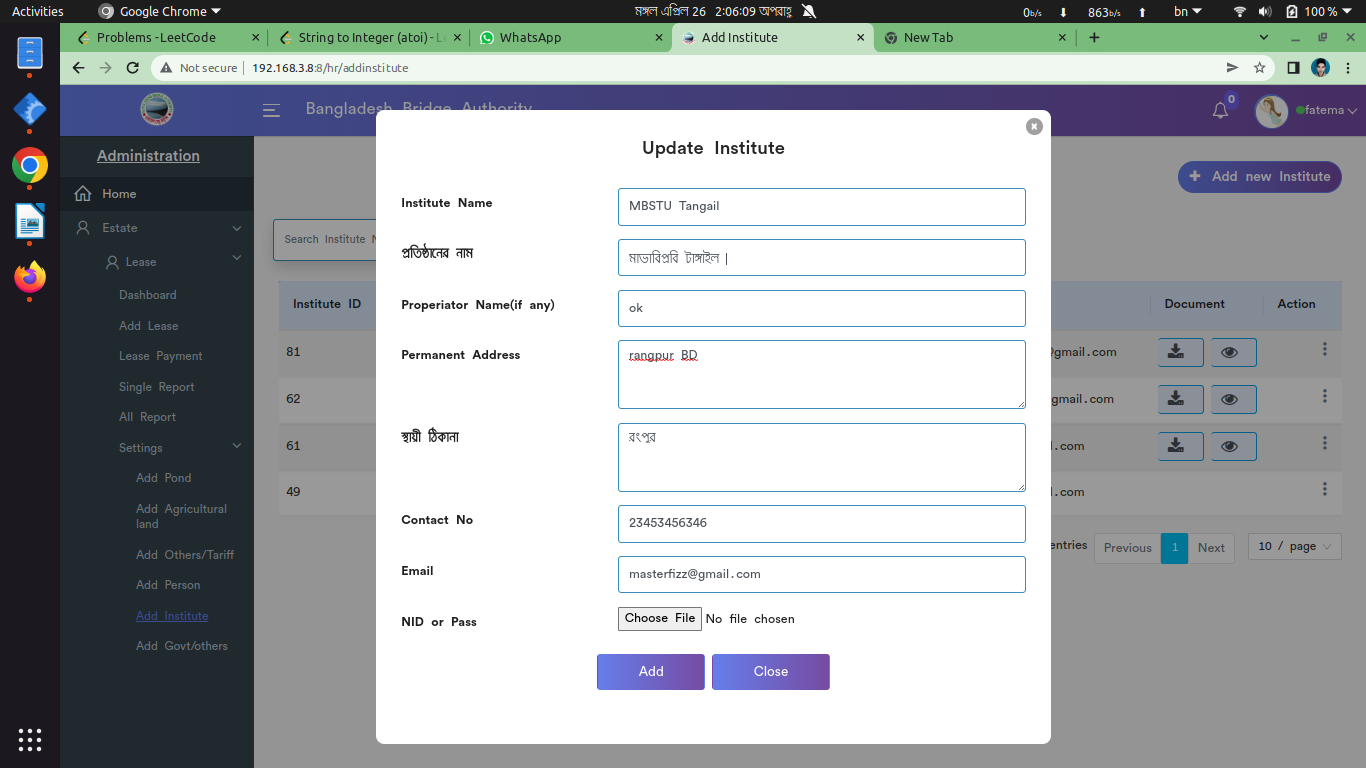
*Negative Test Case*

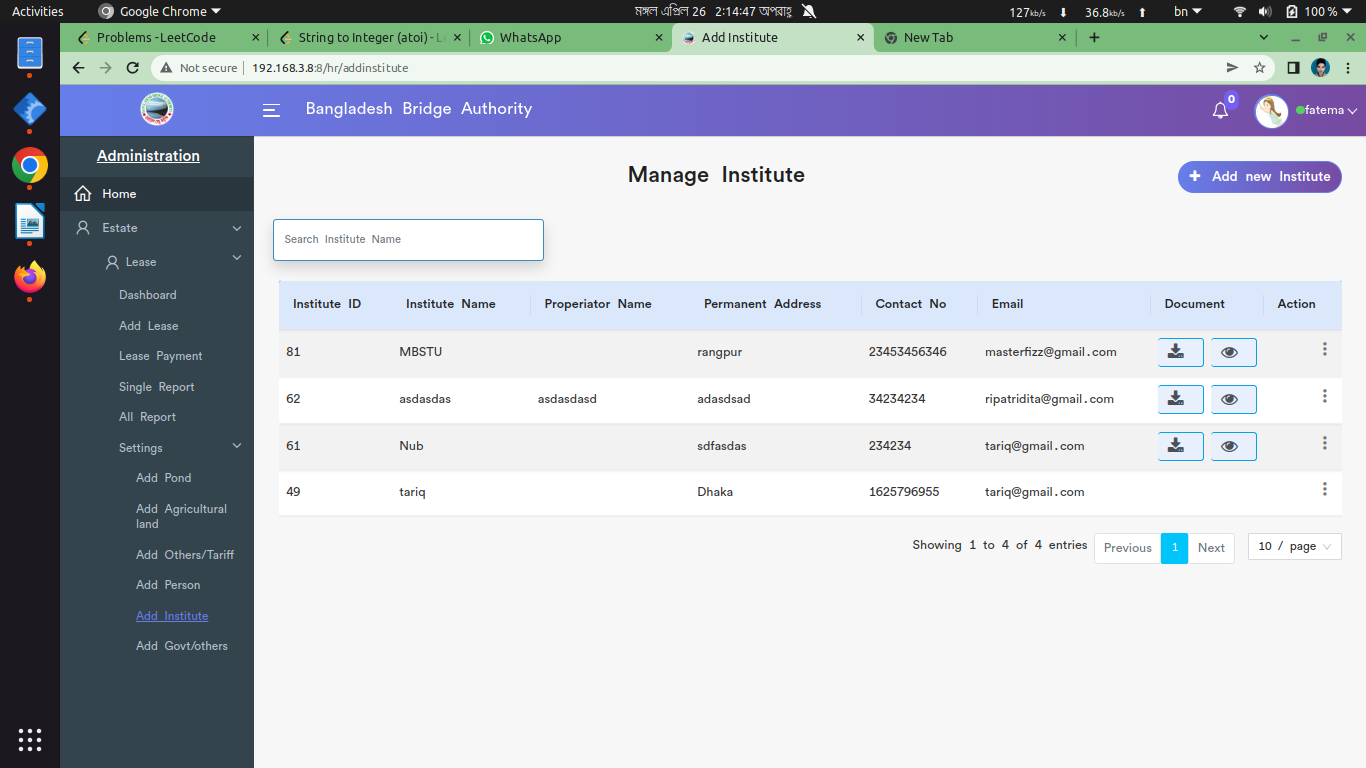
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new person | 1. Data should be entered  2. add button should perfectly work | Data are valid and ok | No irrelevant data | P |
| 2 | Manage person | No error while displaying person’s information | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |

v. Add Institute









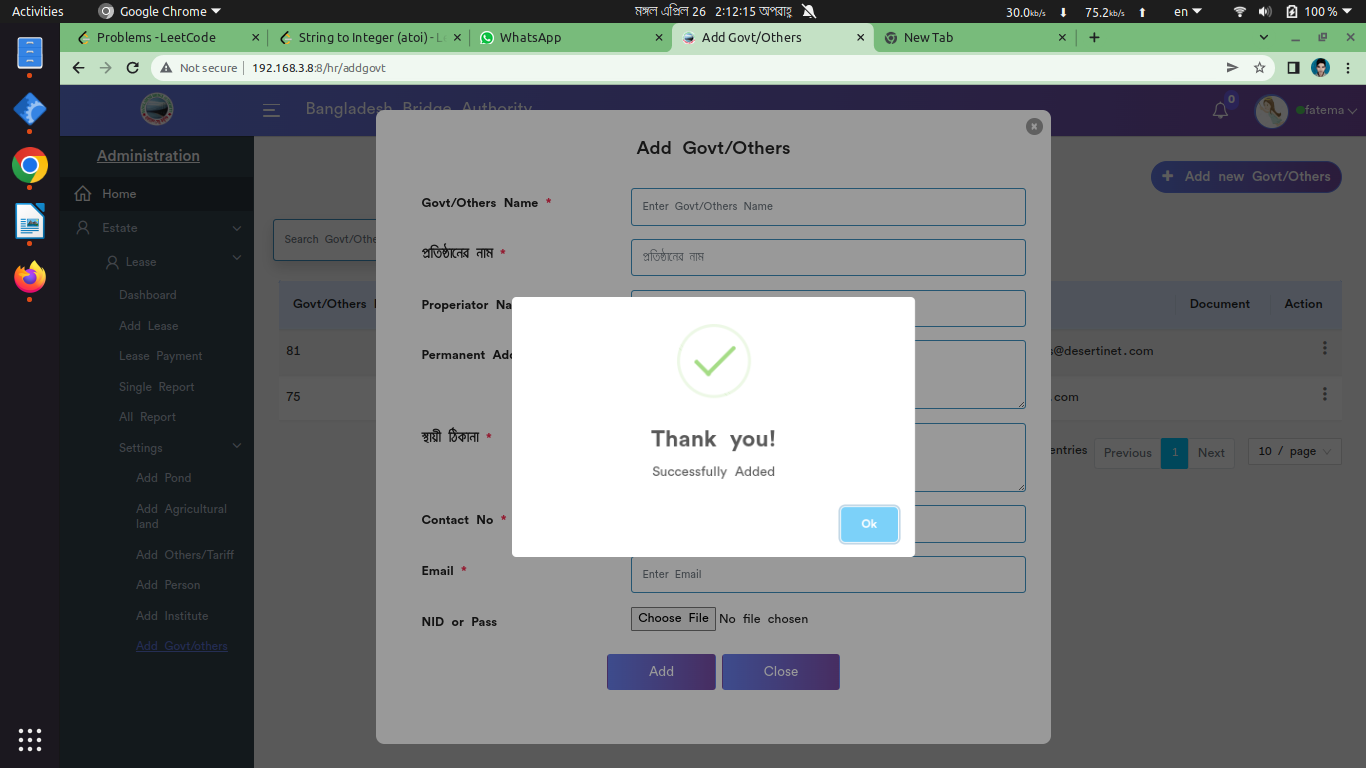
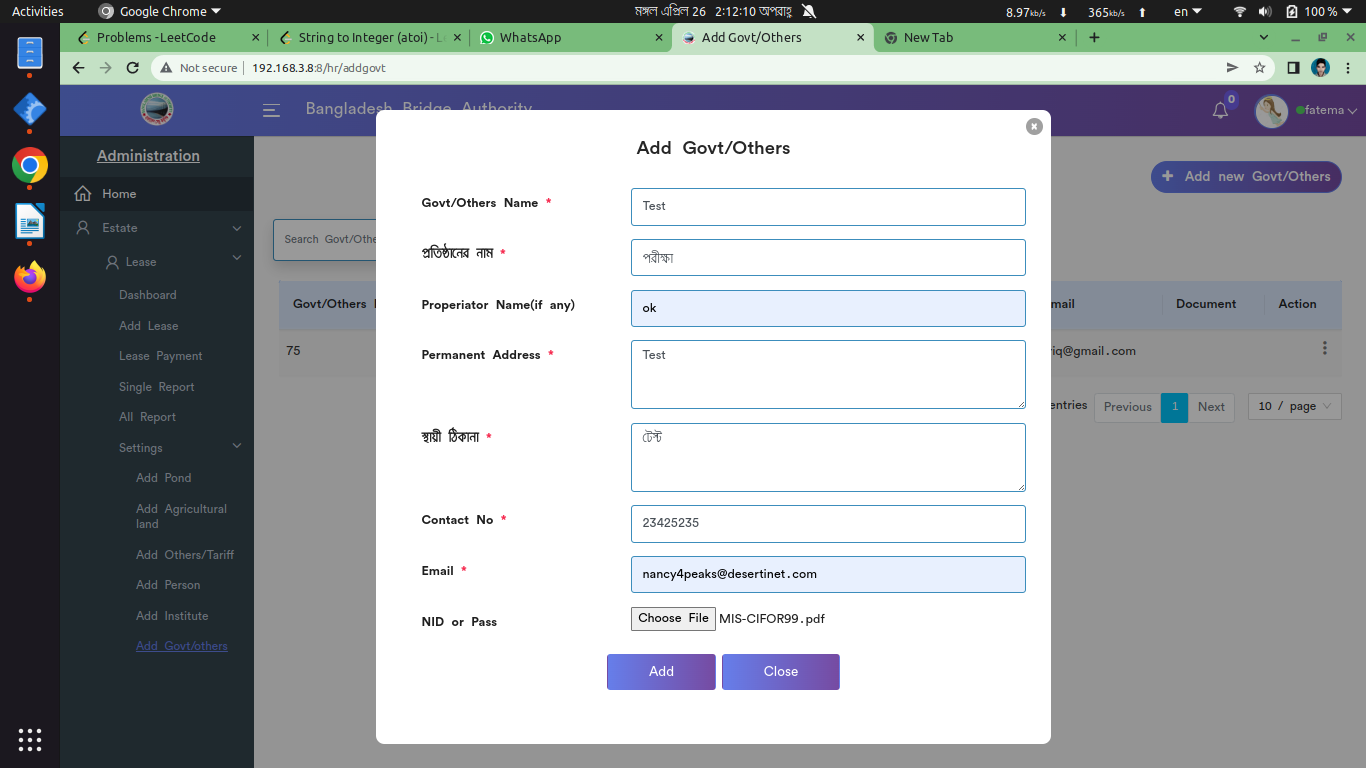
*Positive test case*

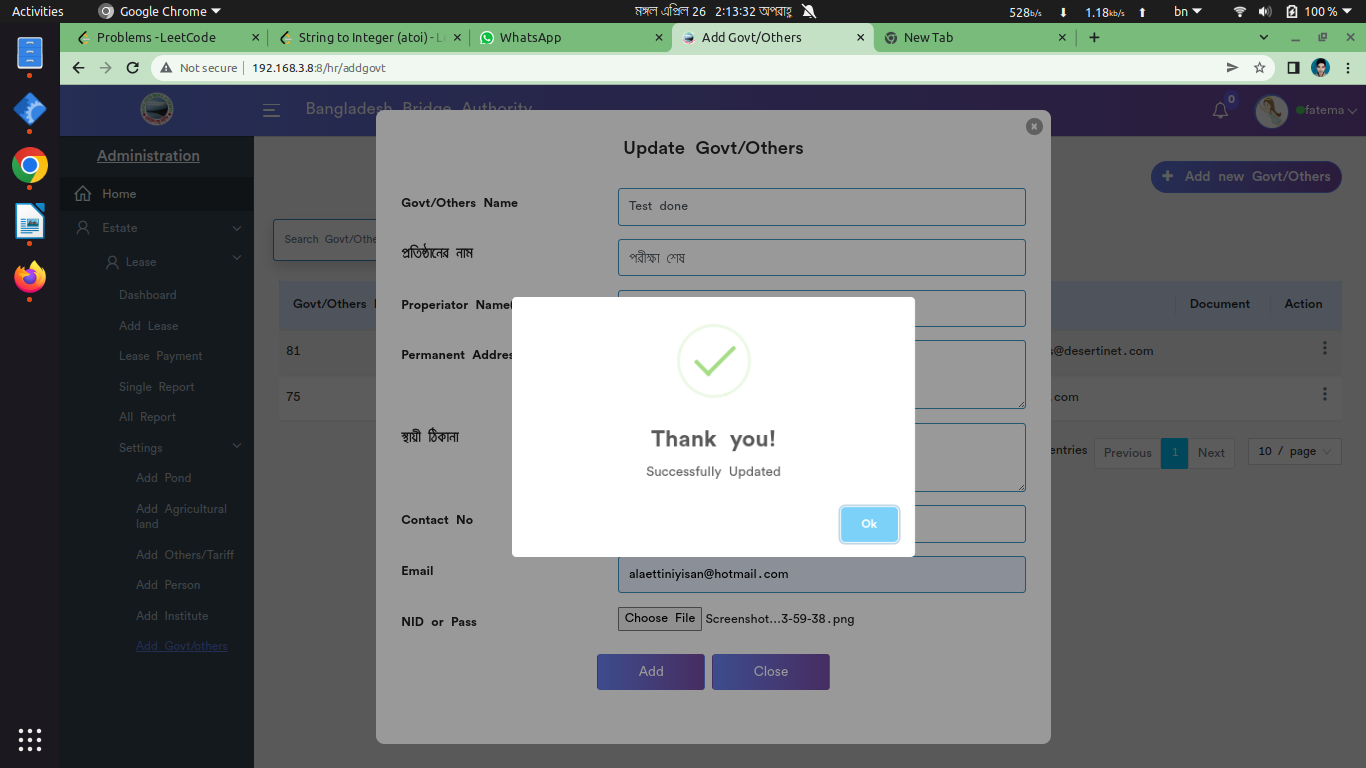
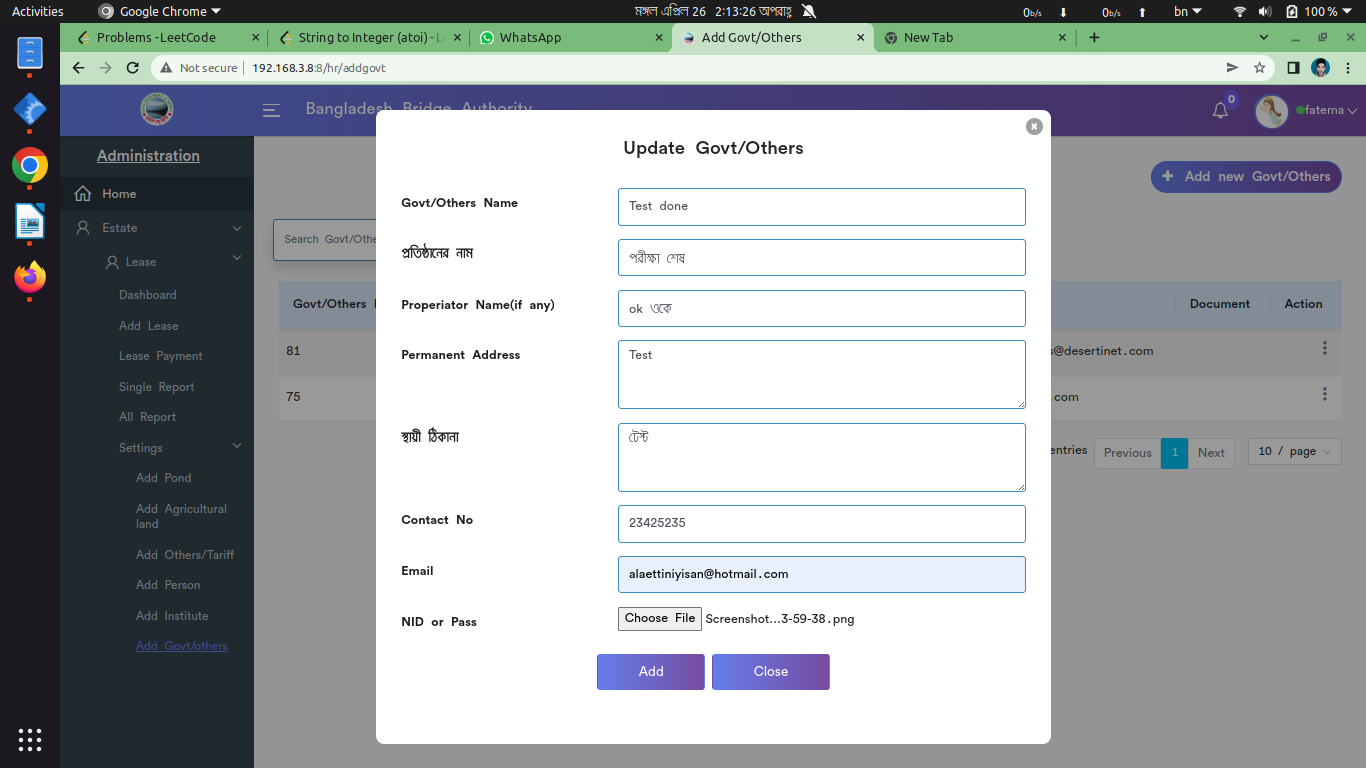
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new institute | 1. Enter institute name & permanent address both in english and bangla  2. Enter properiator name if have  3. Enter contact info & gmail id  4. Enter NID or passport info 5. Click add | Data saved successfully | No error is displayed | P |
| 2 | Manage institute | Display all the institute information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

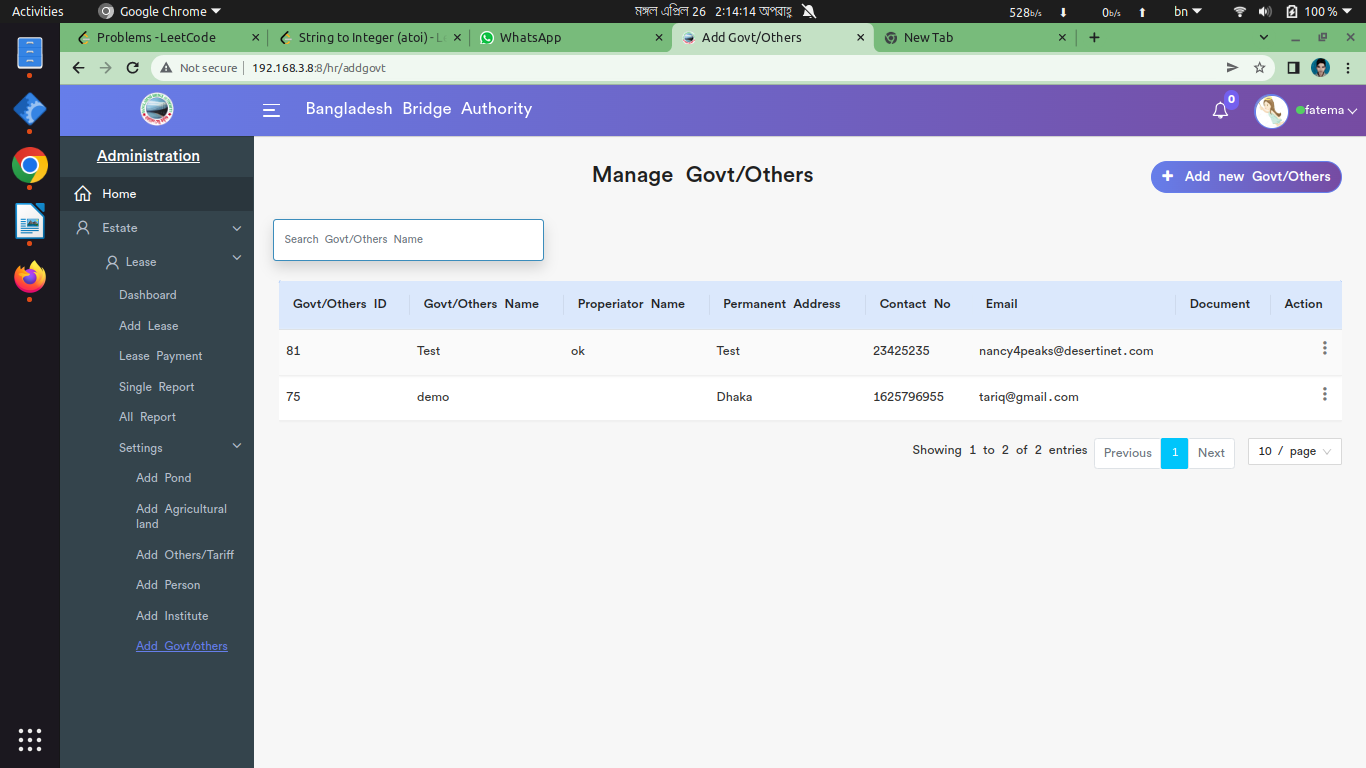
*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new institute | 1. Data should be entered  2. add button should perfectly work | Data are valid and ok | No irrelevant data | P |
| 2 | Manage institute | No error while displaying institute information | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |

vi. Add Govt

**

**

**

*Positive test case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new Govt | 1. Enter Govt name & permanent address both in english and bangla  2. Enter properiator name if have  3. Enter contact info & gmail id  4. Enter NID or passport info 5. Click add | Data saved successfully | No error is displayed | P |
| 2 | Manage Govt | Display all the institute information | Displayed all information | No error | P |
| 3 | Update & Delete | Update the existing information & Delete button delete a specific item | Action button does the specific operation | Works fine & displayed | P |

*Negative Test Case*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case ID | Title | Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Add new Govt | 1. Data should be entered  2. add button should perfectly work | Data are valid and ok | No irrelevant data | P |
| 2 | Manage Govt | No error while displaying Govt information | Perfectly all ok | No missing any information |  |
| 3 | Update & Delete | Update & Delete button must work fine | Operation is perfectly done | No error occurred | P |