**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

**INFORMATION TECHNOLOGY**

**⬩⬩🙠🕮🙢⬩⬩**



**THE PROJECT TWO REPORT**

**BUILDING A CLOUD SURVEY APP**

**LECTURE NAME: M.Sc. Nguyen Duc Khoan**

**STUDENT NAME ID**

**Tran Tan Quy 16110196**

**Nguyen Mai Hung 16110120**

**Ho Chi Minh City, May 2019**

**LECTURER COMMENT**

Signature

**M.Sc. Nguyen Duc Khoan**

**CONTENTS**

[**CONTENTS** 1](#_Toc532818044)

[**1. Project description** 1](#_Toc532818045)

[**1.1. Objective** 1](#_Toc532818046)

[**1.2. User requirements** 1](#_Toc532818047)

[**1.3. Usecase diagram** 2](#_Toc532818048)

[**1.4. Usecase description tables** 2](#_Toc532818049)

[**2. Software design** 6](#_Toc532818050)

[**2.1. Technology** 6](#_Toc532818051)

[**2.2. Software architecture** 7](#_Toc532818052)

[**2.3. Design user interface** 10](#_Toc532818053)

[**2.4. Class diagram** 12](#_Toc532818054)

[**2.5. Data sources** 23](#_Toc532818055)

[**2.6. Configuration** 23](#_Toc532818056)

[**3. Project implementation** 24](#_Toc532818057)

[**3.1. Enviroment** 24](#_Toc532818058)

[**3.2. Work assignment** 24](#_Toc532818059)

[**3.3. Work plan** 25](#_Toc532818060)

[**4. Manual instruction** 26](#_Toc532818061)

[**5. Conclusion** 28](#_Toc532818062)

[**5.1. Results** 28](#_Toc532818063)

[**5.2. Difficulties** 28](#_Toc532818064)

[**5.3. Solutions** 28](#_Toc532818065)

[**5.4. Advantages** 28](#_Toc532818066)

[**5.5. Disadvantages** 29](#_Toc532818067)

[**5.6. Developing ideas** 29](#_Toc532818068)

[**5.7. Source code** 29](#_Toc532818069)

[**REFERENCES** 30](#_Toc532818070)

**LIST OF ACRONYMS**

- IVR: Interactive Voice Response

- PSTN: Public Switched Telephone Network

- POTS: Plain Old Telephone Service

- SIP: Session Initiation Protocol

- VoIP: Voice over Internet Protocol

**LIST OF IMAGES**

[Image 1. Usecase diagram 2](#_Toc532558294)

[Image 2. Three layer architecture 8](#_Toc532558295)

[Image 3. Project architecture 9](#_Toc532558296)

[Image 4. Login form 11](#_Toc532558297)

[Image 5. Products form 11](#_Toc532558298)

[Image 6. Class diagram 12](#_Toc532558299)

[Image 7. Manual instruction 26](#_Toc532558300)

[Image 8. Manual instruction main form 27](#_Toc532558301)

**LIST OF TABLES**

[Table 1. Usecase login website description 3](#_Toc9241469)

[Table 2. Usecase register account description 3](#_Toc9241470)

[Table 3. Usecase Manage IP address description 4](#_Toc9241471)

[Table 4. Usecase Add an IP address description 4](#_Toc9241472)

[Table 5. Usecase Edit an IP address description 4](#_Toc9241473)

[*Table 6. Usecase Manage survey description* 5](#_Toc9241474)

[Table 7. Usecase Add survey description 6](#_Toc9241475)

[Table 8. Usecase Edit survey description 6](#_Toc9241476)

[Table 9. Usecase Delete survey description 7](#_Toc9241477)

[Table 10. Usecase Manage device description 7](#_Toc9241478)

[Table 11. Usecase Create device description 8](#_Toc9241479)

[Table 12. Usecase Edit device description 8](#_Toc9241480)

[Table 13. Usecase Delete device description 9](#_Toc9241481)

[Table 14. Usecase Manage agent description 10](#_Toc9241482)

[Table 15. Usecase Create agent description 10](#_Toc9241483)

[Table 16. Usecase Edit agent description 11](#_Toc9241484)

[Table 17. Usecase Delete agent description 11](#_Toc9241485)

[Table 18. Usecase View reports description 12](#_Toc9241486)

[Table 19. Usecase Logout description 12](#_Toc9241487)

[Table 20. Usecase Take survey description 12](#_Toc9241488)

[Table 23. List of classes 19](#_Toc9241489)

[Table 24. Methods in iDataAccess 20](#_Toc9241490)

[Table 25. Methods in SQLServer 21](#_Toc9241491)

[Table 26. Methods in MySQL 22](#_Toc9241492)

[Table 27. Methods in ACCESS 23](#_Toc9241493)

[Table 28. Methods in PostgreSQL 24](#_Toc9241494)

[Table 29. Methods in BLL 25](#_Toc9241495)

[Table 30. Methods in ProductsBL 25](#_Toc9241496)

[Table 31. Methods in LoginBL 26](#_Toc9241497)

[Table 32. Methods in frmLogin 27](#_Toc9241498)

[Table 33. Methods in frmProducts 28](#_Toc9241499)

[Table 34. Work assignment 31](#_Toc9241500)

[Table 35. Work plan 32](#_Toc9241501)

# **CONTENTS**

# ***1. Project description***

In every service that we use nowadays, we always want the best service quality for every single service and every company also wants to improve their service quality as best as they can, especially customer services. Recognizing the needs, we are building a system that can help those companies to meet their needs. Our main targets that we are aiming to are call centers which don’t have a way to get customers feedbacks about their service.

**What is call center?**

Call center is a centralized office used for receiving and transmitting a huge amount of calls by telephone, their primary employees called ‘agents’. Agent is the person who takes responsibility for answering calls from their customers. So technically, every agent’s performance for every call will decide their service quality. The better agent’s performance, the better their service quality.

To improve the service quality, they might need something that can help them gather feedbacks from their customers about agent’s performance after every call so we are building ‘Cloud Survey App’ to help them do it.

## **Objective**

Cloud Survey App will allow call center to register to use its services via the web app. They can create/customize a survey, which is an interactive voice response (IVR) so that they can use this survey to gather feedbacks from customer after every call. After an agent or a customer hangs up a call, the survey will play and ask customer to rate an agent whom they just spoke with. Call center can base on the rating report to check agent’s performance.

## **1.2. User requirements**

- Call center can publish, modify IP address, agents information and devices information of their call center to use Cloud Survey App services.

- Call center can create/customize their survey which will be used for gathering feedbacks from their customers.

- Call center can view rating reports of agents in order to increase their service quality.

- Call center’s customers can answer and interact with the survey to rate agent.  
E.g: press button 1 on phone to rate 1 star for the agent.

## **1.3. Usecase diagram**

Image 1. Usecase diagram



## **1.4. Usecase description tables**

Table 1. Usecase login website description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Login website | | |
| Description | Allow user to log in the web app | | |
| Actor | Unauthorized user | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Click ‘Sign in’ button. |  |
|  | 3 |  | Show login interface. |
|  | 4 | Enter username and pass word then click ‘login’ button. |  |
|  | 5 |  | Process username password and forward to main UI. |
| Preconditions | - Must have an account.  - Internet connection required. | | |
| Condition affecting termination outcome | - When user signs in successfully, show the main homepage and functions.  - When user signs in fail, show error. | | |

Table 2. Usecase register account description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Register account | | |
| Description | Allow user to register a new account to login and use the services. | | |
| Actor | Unauthorized user | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Click ‘Sign up’ button. |  |
|  | 3 |  | Show register interface. |
|  | 4 | Input information to the register form. |  |
|  | 5 | Click ‘Sign up’ button to create account. |  |
| Preconditions | - Password must be strong.  - Internet connection required. | | |
| Condition affecting termination outcome | - When user signs up successfully, show ‘Sign in’ interface for user to sign in.  - When user signs up fail, show error. | | |

Table 3. Usecase Manage IP address description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Manage IP address | | |
| Description | User can configure IP address of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Choose ‘Domain’ tab. |  |
|  | 6 |  | Show IP address input. |
| Preconditions | - User logged in successfully.  - Internet connection required | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification. | | |

Table 4. Usecase Add an IP address description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Add an IP address | | |
| Description | User can add IP address of their call center for the first time | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Choose ‘Domain’ tab. |  |
|  | 6 |  | Show IP address input. |
|  | 7 | User inputs a new IP address. |  |
|  | 8 | User clicks ‘Save’ button to save the IP address. |  |
| Preconditions | - User logged in successfully.  - Internet connection required.  - IP address must be a valid one. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - If the web app fail to save, show notification.  - If the web app save successfully, show notification.  - If the IP address of user is invalid, show notification. | | |

Table 5. Usecase Edit an IP address description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Edit an IP address | | |
| Description | User can edit IP address of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Choose ‘Domain’ tab. |  |
|  | 6 |  | Show IP address input. |
|  | 7 | User edits the exisitng IP address. |  |
|  | 8 | User clicks ‘Save’ button to save the IP address. |  |
| Preconditions | - User logged in successfully.  - Internet connection required.  - IP address must be a valid one. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - If the web app fail to save, show notification.  - If the web app save successfully, show notification.  - If the IP address of user is invalid, show notification. | | |

*Table 6. Usecase Manage survey description*

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Manage survey | | |
| Description | User can manage survey of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Survey Management’ button on homepage. |  |
|  | 4 |  | Show ‘Survey Management’ page. |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification. | | |

Table 7. Usecase Add survey description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Add survey | | |
| Description | User can add new survey | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Survey Management’ button on homepage. |  |
|  | 4 |  | Show ‘Survey Management’ page. |
|  | 5 | Click ‘Create new’ button. |  |
|  | 6 |  | Show modal for user to input information. |
|  | 7 | User inputs information |  |
|  | 8 | Click ‘save’ button |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest survey added.  - When the web app fail to save, show notification. | | |

Table 8. Usecase Edit survey description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Edit survey | | |
| Description | User can edit existing survey | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Survey Management’ button on homepage. |  |
|  | 4 |  | Show ‘Survey Management’ page. |
|  | 5 | Click ‘Edit’ button. |  |
|  | 6 |  | Show modal for user to change information. |
|  | 7 | User changes information. |  |
|  | 8 | Click ‘save’ button. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest survey edited.  - When the web app fail to save, show notification. | | |

Table 9. Usecase Delete survey description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Delete survey | | |
| Description | User can delete existing survey | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Survey Management’ button on homepage. |  |
|  | 4 |  | Show ‘Survey Management’ page. |
|  | 5 | Click ‘Delete’ button. |  |
|  | 6 |  | Show message to ask user if they want to delete or not. |
|  | 7 | Click ‘Yes’ to delete, ‘No’ to cancel delete. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app deleted successfully, reload the page.  - When the web app fail to delete, show notification. | | |

Table 10. Usecase Manage device description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Manage device | | |
| Description | User can manage device of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Device’ tab. |  |
|  | 6 |  | Show device list. |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification. | | |

Table 11. Usecase Create device description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Create device | | |
| Description | User can add new device of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Device’ tab. |  |
|  | 6 |  | Show device list. |
|  | 7 | Click ‘Create new’ button. |  |
|  | 8 |  | Show modal for user to input information. |
|  | 9 | User inputs information for device. |  |
|  | 10 | Click ‘Save’ button. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest device added.  - When the web app fail to save, show notification. | | |

Table 12. Usecase Edit device description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Edit device | | |
| Description | User can edit existing device of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Device’ tab. |  |
|  | 6 |  | Show device list. |
|  | 7 | Click edit button. |  |
|  | 8 |  | Show modal for user to edit information. |
|  | 9 | User edit information for device if needed |  |
|  | 10 | Click ‘Save’ button. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest device edited.  - When the web app fail to save, show notification. | | |

Table 13. Usecase Delete device description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Delete device | | |
| Description | User can delete existing device of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Device’ tab. |  |
|  | 6 |  | Show device list. |
|  | 7 | Click delete button. |  |
|  | 8 |  | Show message to ask user if they want to delete or not. |
|  | 9 | Click ‘Yes’ to delete, ‘No’ to cancel delete. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app deleted successfully, reload the page.  - When the web app fail to delete, show notification. | | |

Table 14. Usecase Manage agent description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Manage agent | | |
| Description | User can manage agent of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Agent’ tab. |  |
|  | 6 |  | Show agent list. |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification. | | |

Table 15. Usecase Create agent description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Create agent | | |
| Description | User can add new agent of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Agent’ tab. |  |
|  | 6 |  | Show agent list. |
|  | 7 | Click ‘Create new’ button. |  |
|  | 8 |  | Show modal for user to input information. |
|  | 9 | User inputs information for agent. |  |
|  | 10 | Click ‘Save’ button. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest agent added.  - When the web app fail to save, show notification. | | |

Table 16. Usecase Edit agent description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Edit agent | | |
| Description | User can edit existing agent of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Agent’ tab. |  |
|  | 6 |  | Show agent list. |
|  | 7 | Click edit button. |  |
|  | 8 |  | Show modal for user to edit information. |
|  | 9 | User edit information for agent if needed |  |
|  | 10 | Click ‘Save’ button. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app saved successfully, reload and show the latest agent edited.  - When the web app fail to save, show notification. | | |

Table 17. Usecase Delete agent description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Delete agent | | |
| Description | User can delete existing agent of their call center | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the web app. |  |
|  | 2 | Log in the web app. |  |
|  | 3 | Click ‘Configuration’ button on homepage. |  |
|  | 4 |  | Show ‘Configuration’ page. |
|  | 5 | Select ‘Agent’ tab. |  |
|  | 6 |  | Show agent list. |
|  | 7 | Click delete button. |  |
|  | 8 |  | Show message to ask user if they want to delete or not. |
|  | 9 | Click ‘Yes’ to delete, ‘No’ to cancel delete. |  |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | - When the web app fail to load, show notification.  - When the web app deleted successfully, reload the page.  - When the web app fail to delete, show notification. | | |

Table 18. Usecase View reports description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | View reports | | |
| Description | User view reports about their call center agents | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Click ‘Report’ button. |  |
|  | 2 |  | Show ‘Report’ page. |
|  | 3 | User inputs conditions of report |  |
|  | 4 | Click ‘View report’ button |  |
|  | 5 |  | Show report bases on user’s conditions. |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | When the web app fail to load, show notification. | | |

Table 19. Usecase Logout description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Logout | | |
| Description | User can logout of the web app | | |
| Actor | User as known as Call center | | |
| Business event | No. | Agent | System response |
|  | 1 | Click ‘Sign out’ button. |  |
|  | 2 |  | Disconnect user’s account from the web app. |
|  | 3 |  | Redirect to main interface of the web app. |
|  | 4 |  | Show ‘Configuration’ page. |
| Preconditions | - User logged in successfully.  - Internet connection required. | | |
| Condition affecting termination outcome | When the web app fail to load, show notification. | | |

Table 20. Usecase Take survey description

|  |  |  |  |
| --- | --- | --- | --- |
| Usecase name | Take survey | | |
| Description | User can logout of the web app | | |
| Actor | Call center’s customer | | |
| Business event | No. | Agent | System response |
|  | 1 | Answer the call. |  |
|  | 2 |  | Play IVR to ask customer if they want to do survey or not. |
|  | 3 | Customer can agree to do survey or not. If they agree, press 1 otherwise press 2 to cancel. |  |
|  | 4 |  | Play IVR survey of the call center and guide customer to do survey. |
|  | 5 | Customer press desired number on the phone to rate the agent |  |
| Preconditions | an make a call to the customer | | |
| Condition affecting termination outcome | If customer pressed other keys instead of number keys, inform them that they have pressed invalid keys. | | |

**2. System design**

## **2.1. Technology**

With this project, we are going to use 2 main technologies which are Asterisk and ASP.NET Core to build the Cloud Survey App.

**2.1.1. Asterisk**

Asterisk is an open source framework for building communications applications. Asterisk turns an ordinary computer into a communications server. Asterisk powers IP PBX systems, VoIP gateways, conference servers and other custom solutions. It is used by small businesses, large businesses, call centers, carriers and government agencies, worldwide. Asterisk is free and open source. Asterisk is sponsored by Digium.[1]



Image 2. Asterisk logo

Asterisk is:[2]

+ Written in C programming language.

+ Primarily running on Linux operating system.

+ Powering Business Telephone Systems.

+ Connecting many different Telephony protocols.

+ A toolkit for building many things:

* An IP PBX with many powerful features and applications.
* VoIP gateways.
* Conferencing system

+ Supporting VoIP phones as well as PSTN and POTS.

+ Speaking SIP, the most common VoIP protocol, among others.

## **2.2. Architecture design**

- Three layer architecture is the main software architecture that we decided to use in this project. This architecture contains 3 layers such as:

+ Presentation layer (UI): contains user interface, windows form where data is presented to users.

+ Business Logic Layer (BLL): contains business logic, validations or calculations related with the data from data access layer.

+Data Access Layer (DAL): contains methods that help DAL to connect to dbms, retrieve data or manipulate data (insert, delete, update,...).

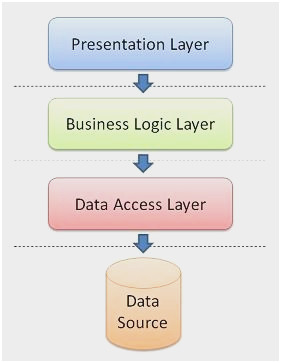


Image 3. Three layer architecture

- Advantages of 3-layer architecture:

+ Dividing into 3 layers helps code looks clearer.

+ Easy to maintain, modify, and improve application without impacting other layers much.

+ Make application more understandable.

+ Reusability: we can reuse middle layer with different interfaces

+ Well hidden database structure, code.

+ High reliability.

- Our project is clearly divided into 3 layers base on the software architecture. It has

Presentation Layer, Business Logic Layer and Data Access Layer.

- In this architecture, there are some important things that we have to keep in mind:

+ Every single layer has its own function, a specific layer is not allowed to do other layers’ functions even though it is possible to do it.

E.g: Presentation Layer shouldn’t do calculating which is belong to Business Logic Layer.

+ Data Access Layer can only to get data from databases, execute queries and can only responses to Business Logic Layer. It won’t be able to contact with Presentation Layer.

+ Business Logic Layer can only interact with Presentation Layer by responding calling function requests from Presentation Layer and data from Data Access Layer, it is not allowed to access databases directly.

+ Presentation Layer can only interact with Business Logic Layer to call functions.

## **2.3. Design user interface**

In software programming, user interfaces have a really big role to determine if a project is worth using. No matter how perfect the back-end programming is, if the UI doesn’t look good, user will not like it. So technically, a good looking UI will bring to users a better experience while using application.

Table 21. UI work assignment

|  |  |  |  |
| --- | --- | --- | --- |
| No | Interface | Designer | Purpose |
| 1 | Homepage |  | In this page, a guest can click sign in or sign up to access to the web app. |
| 2 | Login page |  | In this page, a guest can input email and password to log in. Beside that, s/he can recover password if forgot or can register a new account. |
| 3 | Forgot password page |  | Guest can recover password if forgot. |
| 4 | Register page |  | Guest can register a new account |
| 5 | Main page |  | After login successfully, user can choose tabs to do their work such as Configuration, Survey Management, Report. User can also logout here. |
| 6 | Configuration page |  | In this page, user can work with configuration of domain, device and agent. |
| 7 | Domain configuration |  | User can add/edit IP address of their call center. |
| 8 | Device configuration |  | User can manage, search and filter devices of their call center. |
| 9 | Add device |  | User can add a new device |
| 10 | Edit device |  | User can edit an existing device. |
| 11 | Agent configuration |  | User can manage, search and filter agents of their call center. |
| 12 | Add agent |  | User can add a new agent. |
| 13 | Edit agent |  | User can edit an existing agent. |
| 14 | Survey management page |  | User can manage survey and view report of the survey. |
| 15 | Add survey |  | User can add a new survey. |
| 16 | Edit survey |  | User can edit an existing survey. |
| 17 | Report page |  | User can view reports of their call center. |

Image 4. Homepage

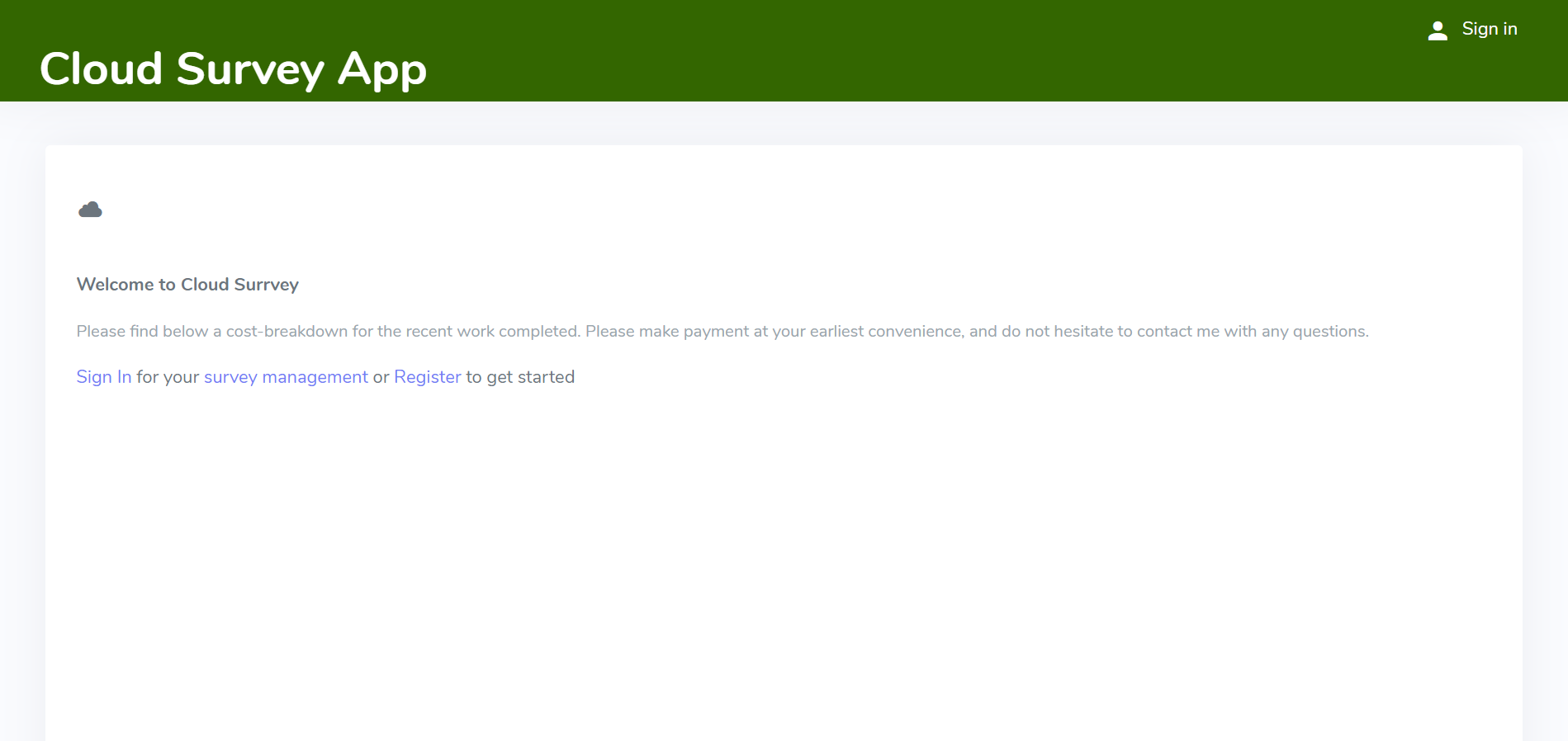


Image 5. Login page

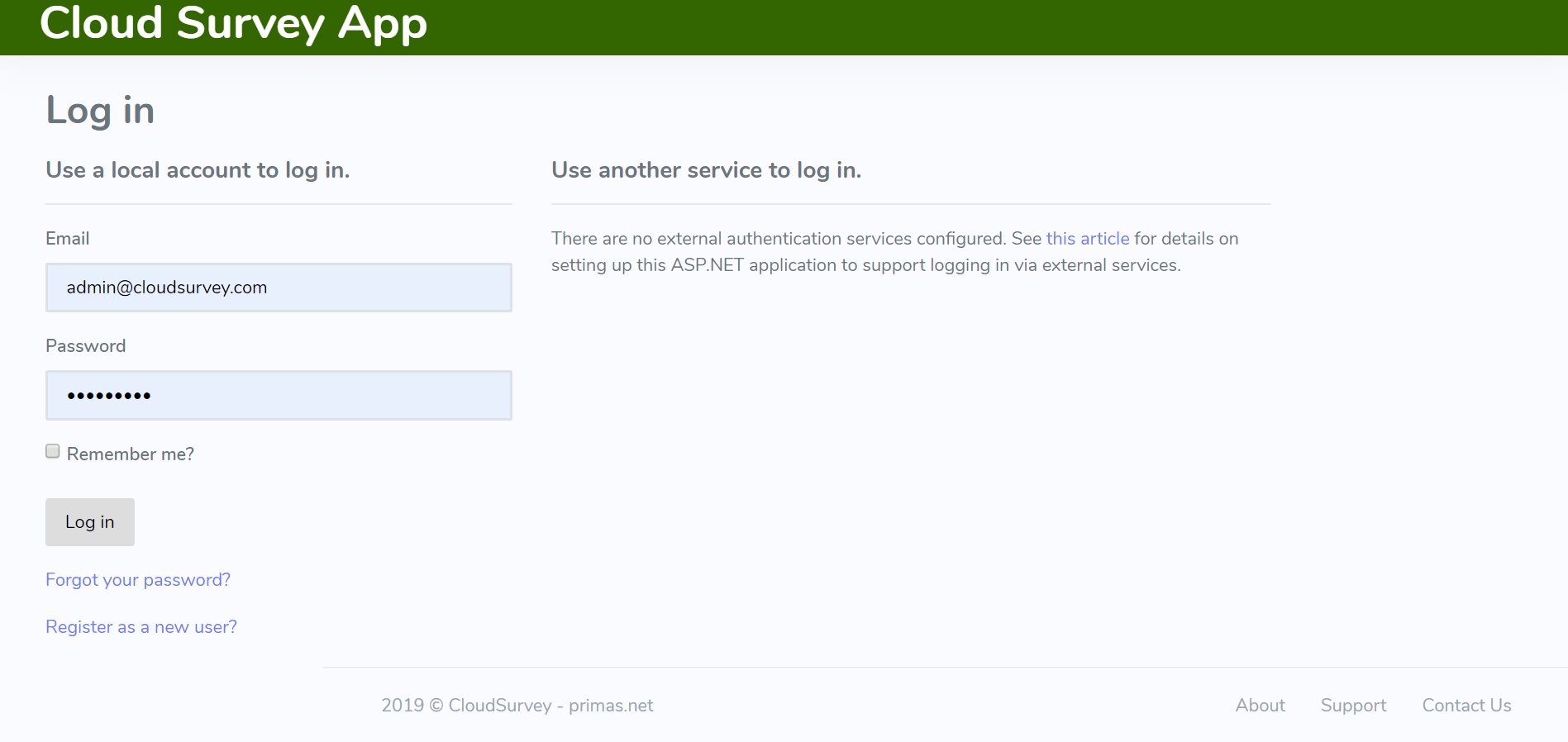


Image 6. Forgot password page

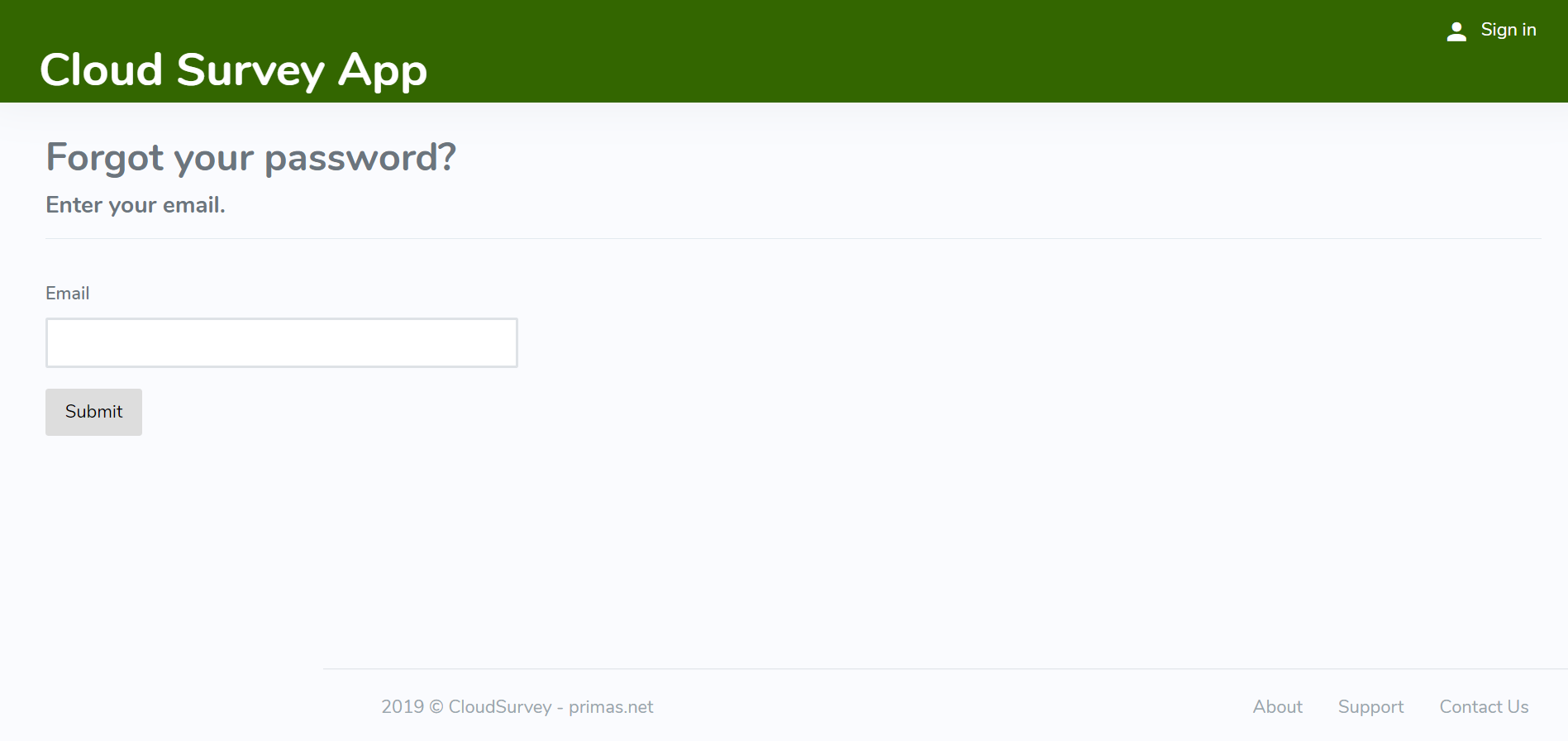


Image 7. Register page

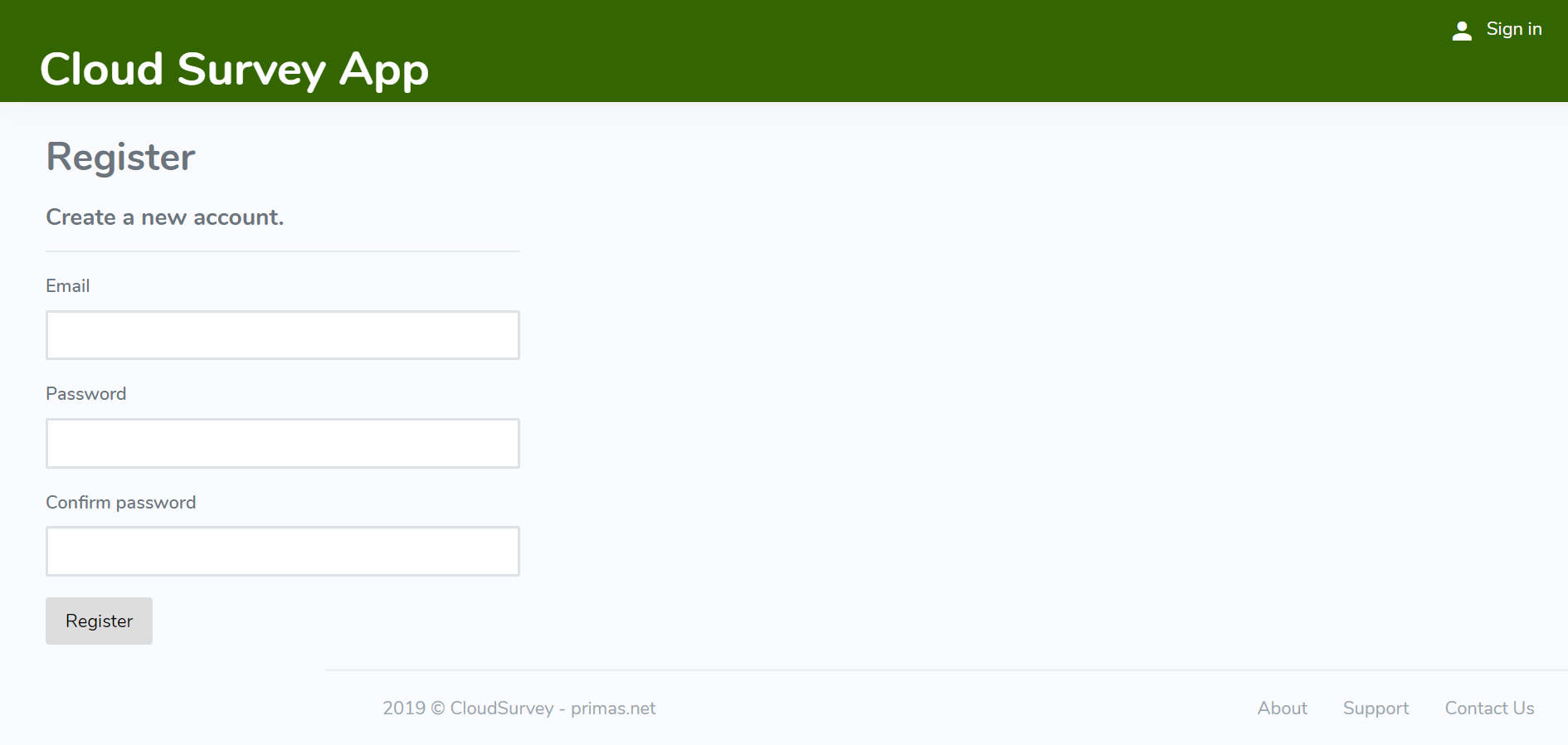


Image 8. Main page

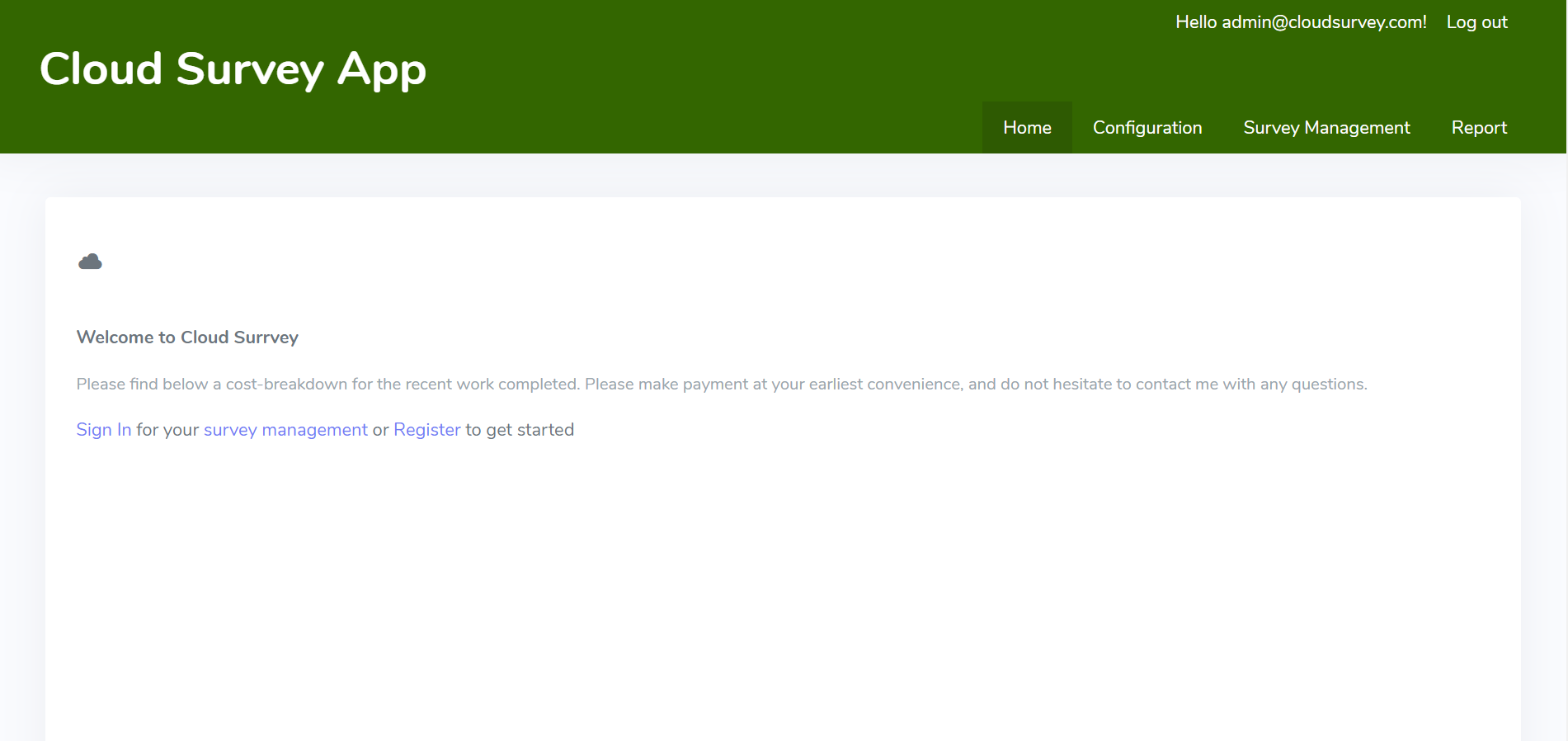


Image 9. Configuration page & Image 10. Domain configuration

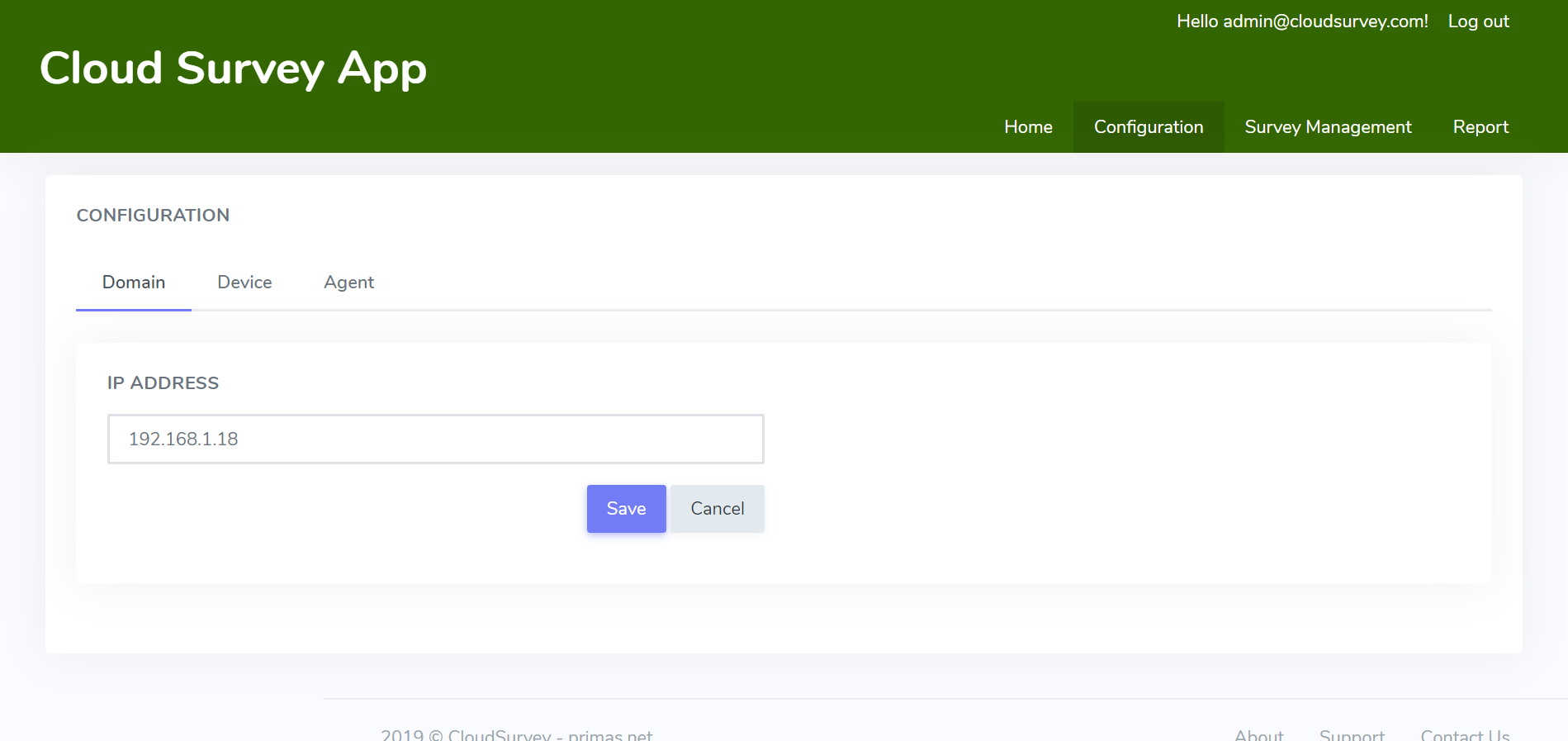


Image 11. Device configuration

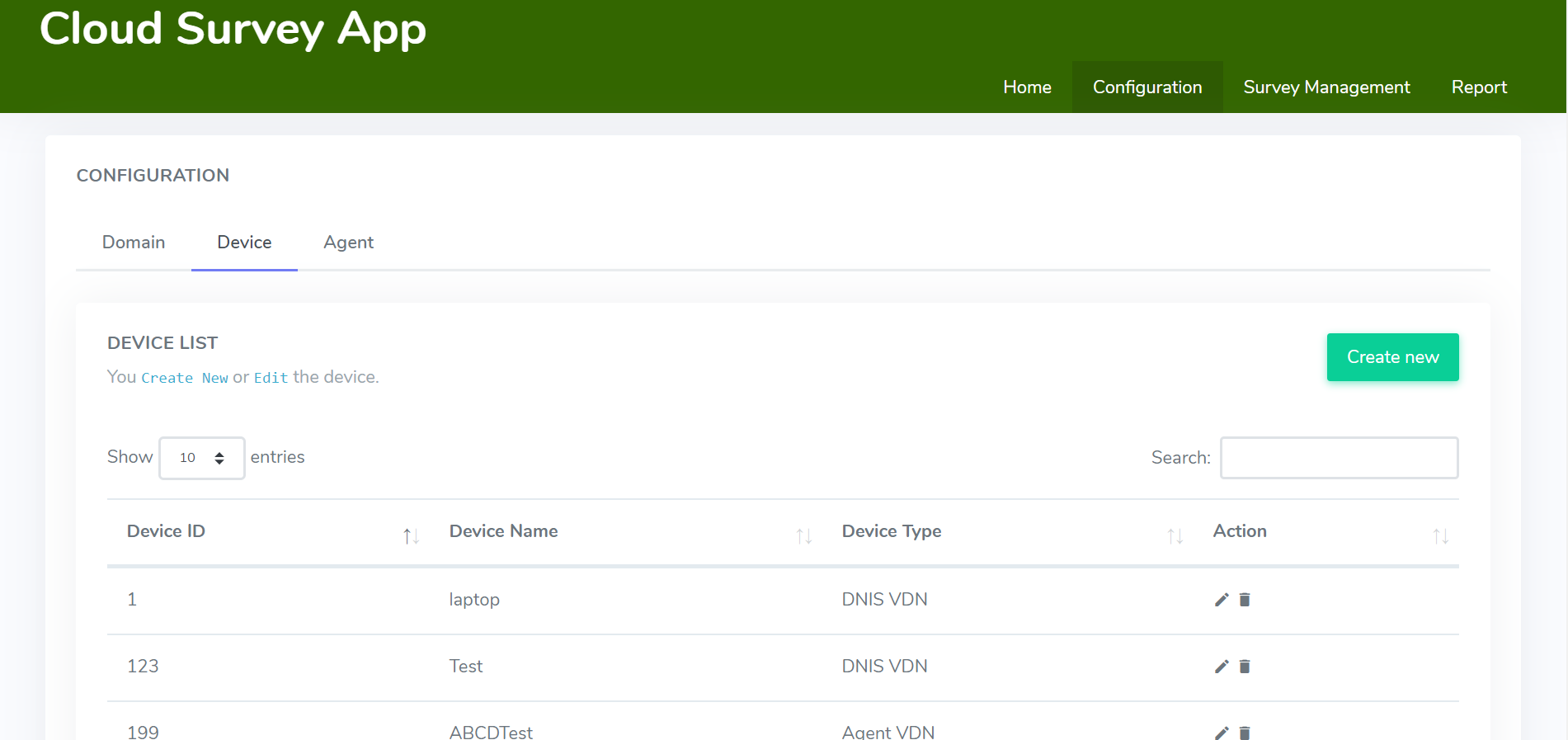


Image 12. Add device

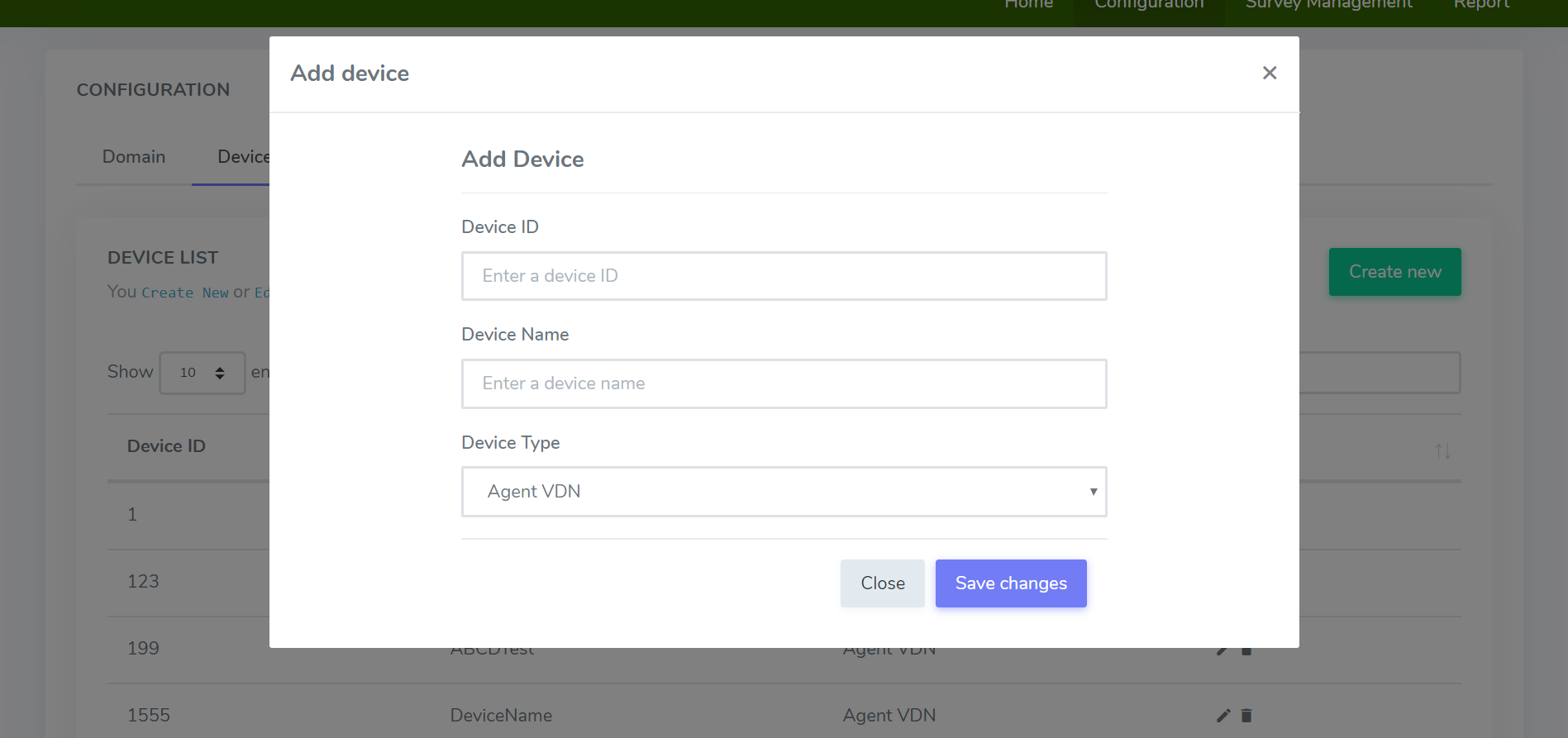


Image 13. Edit device

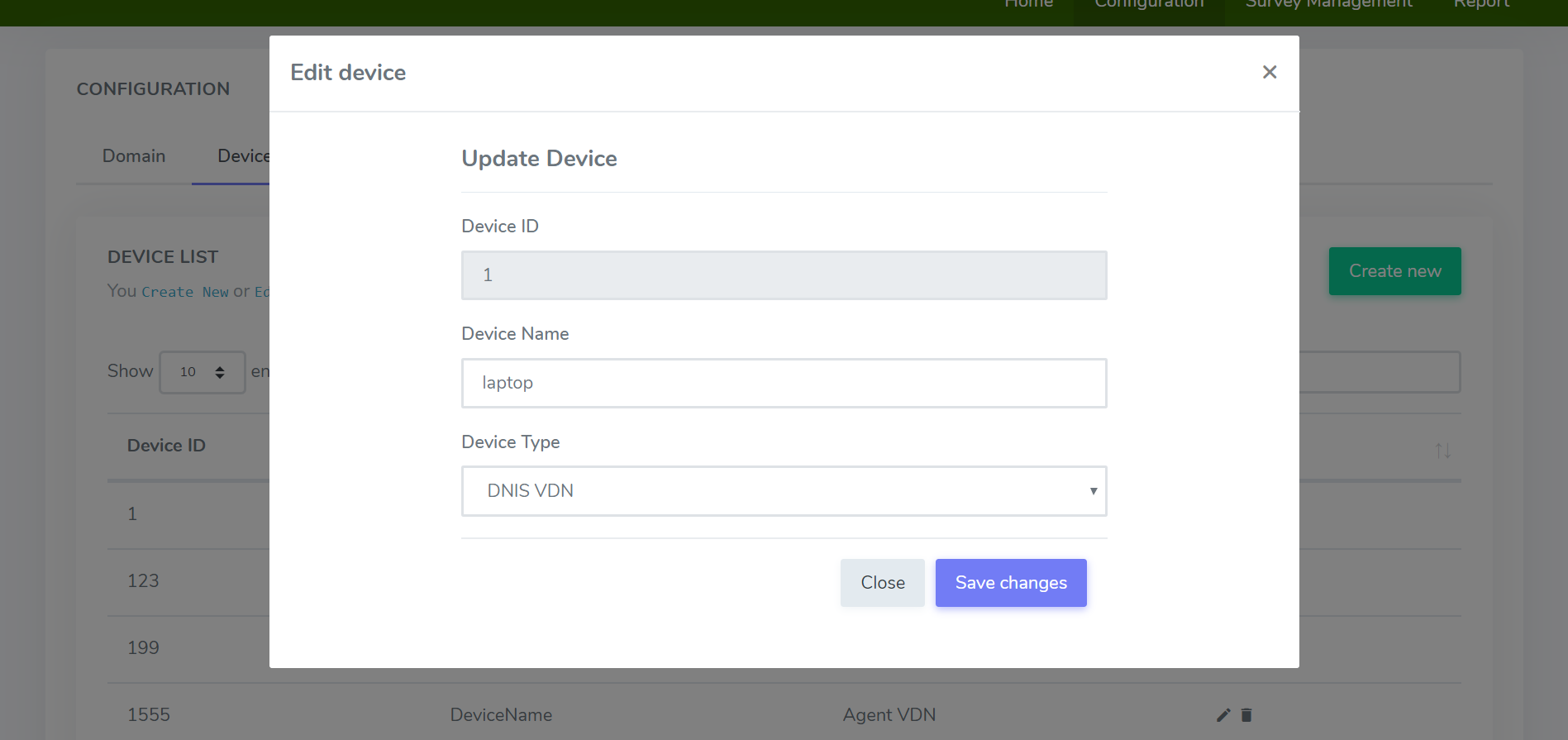


Image 14. Agent configuration

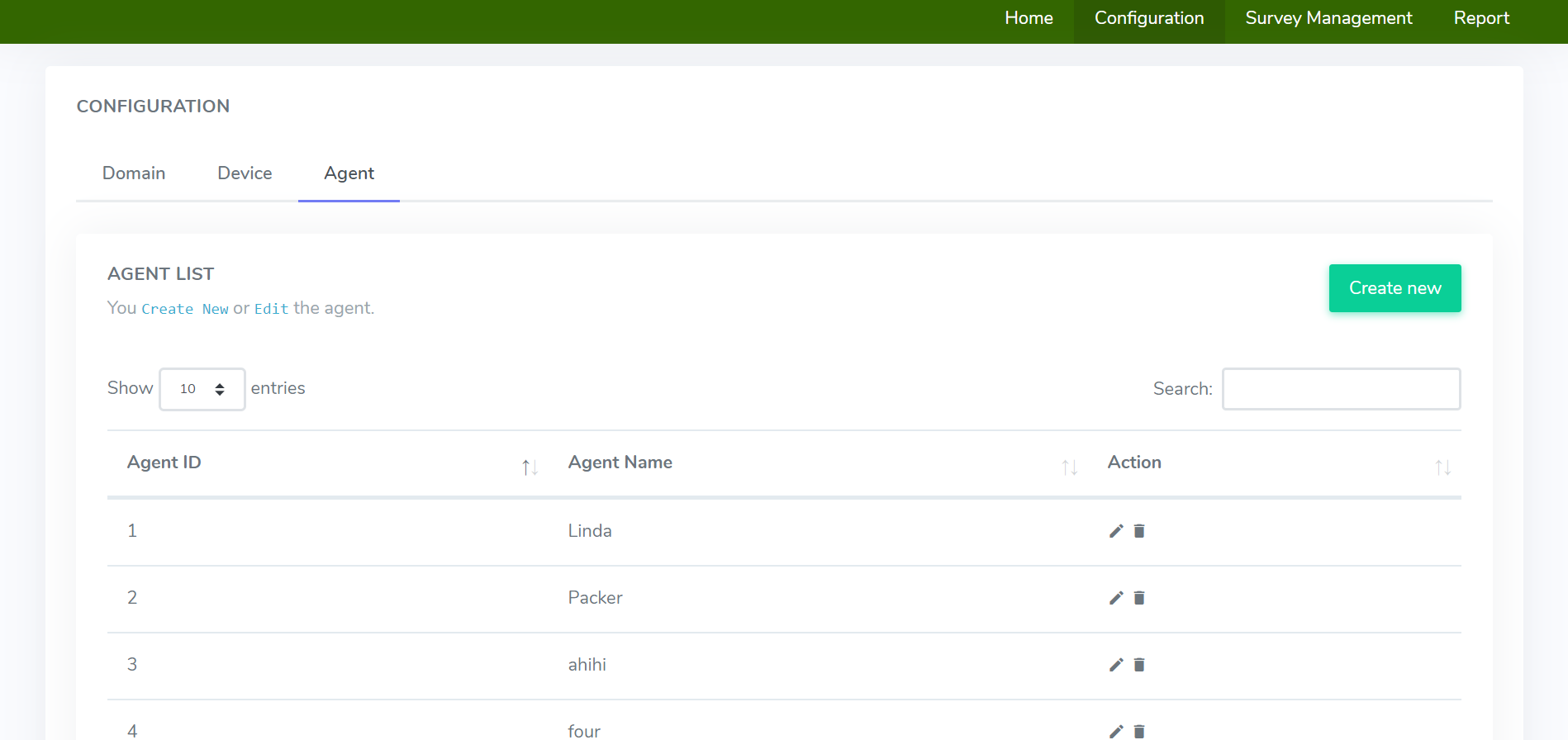


Image 15. Add agent

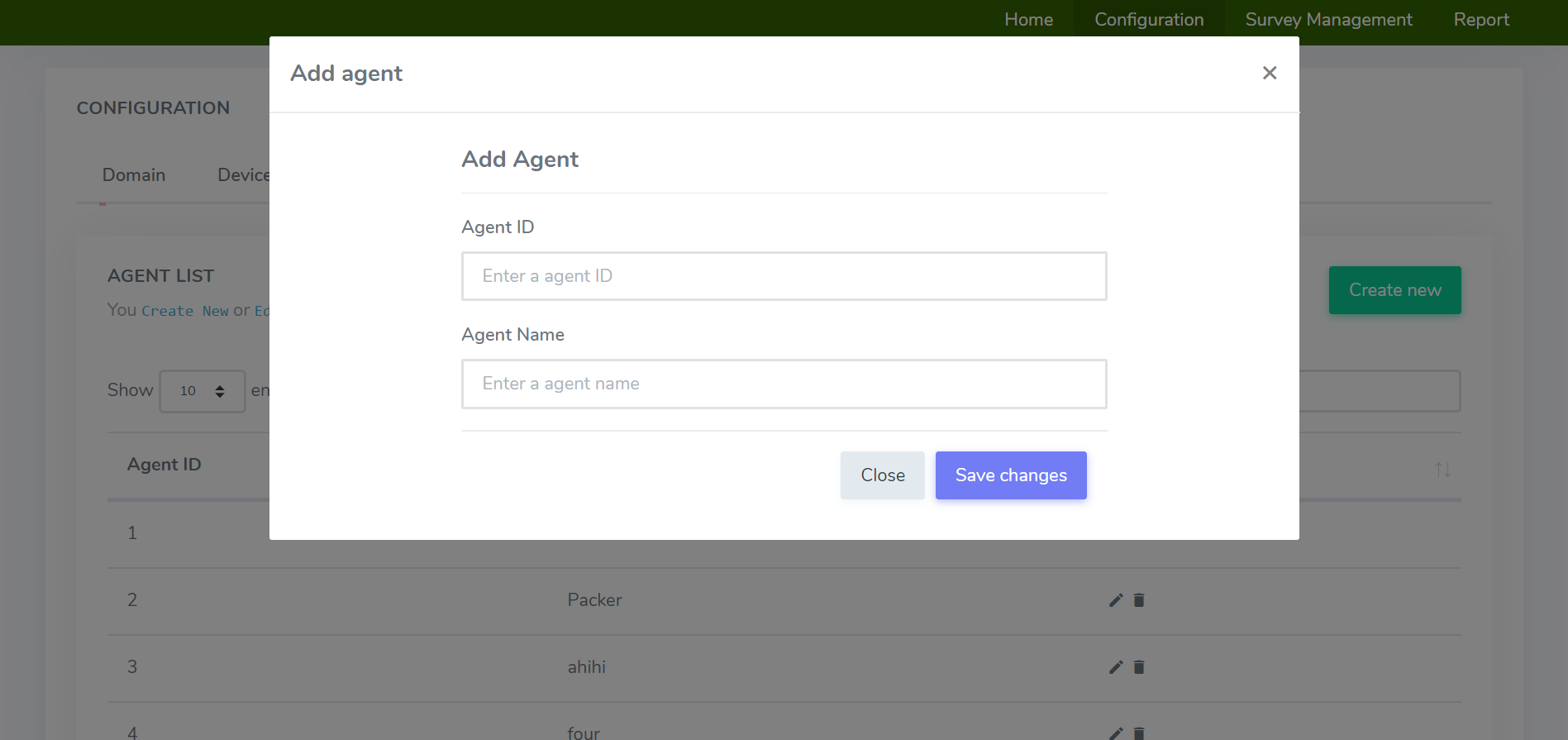


Image 16. Edit agent

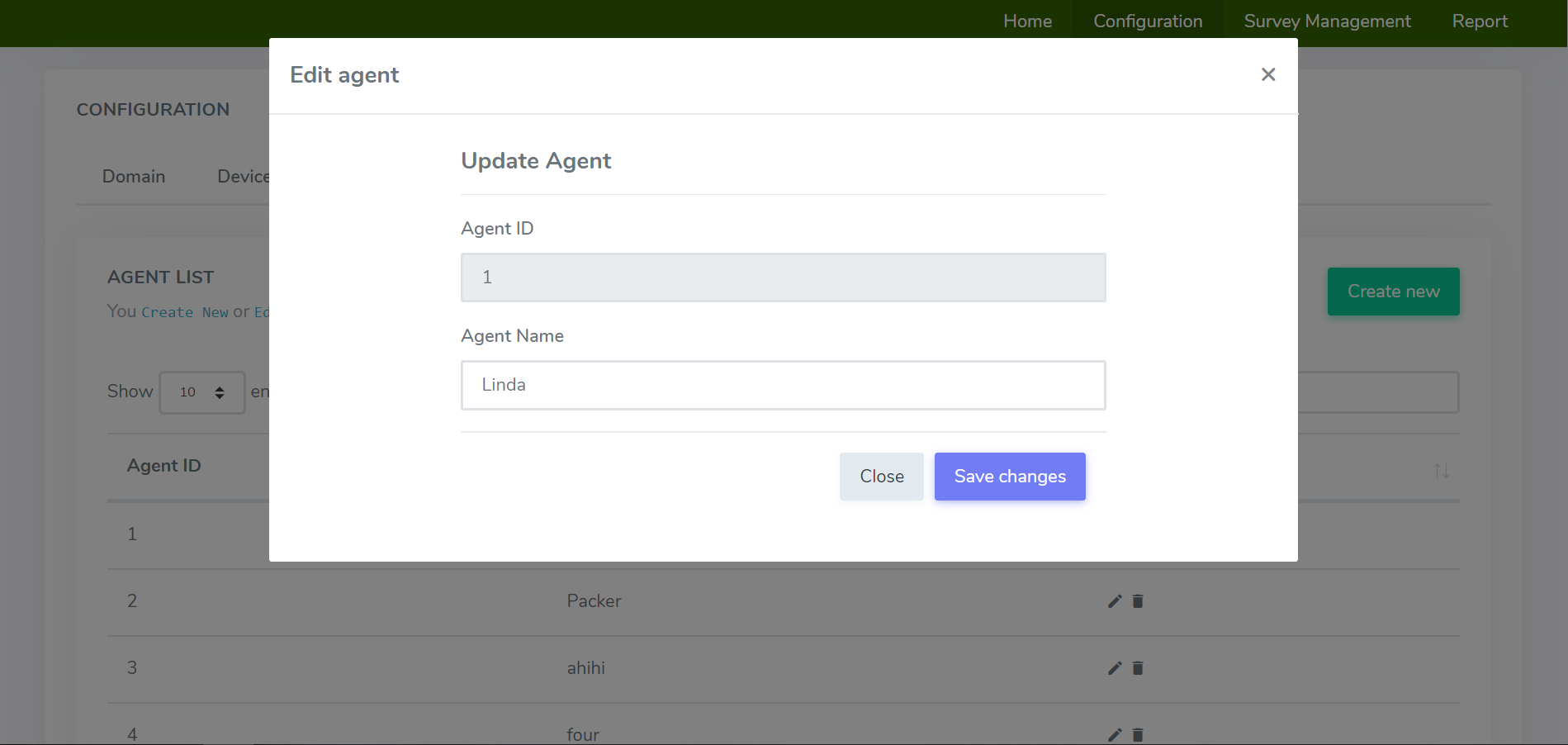


Image 17. Survey management page

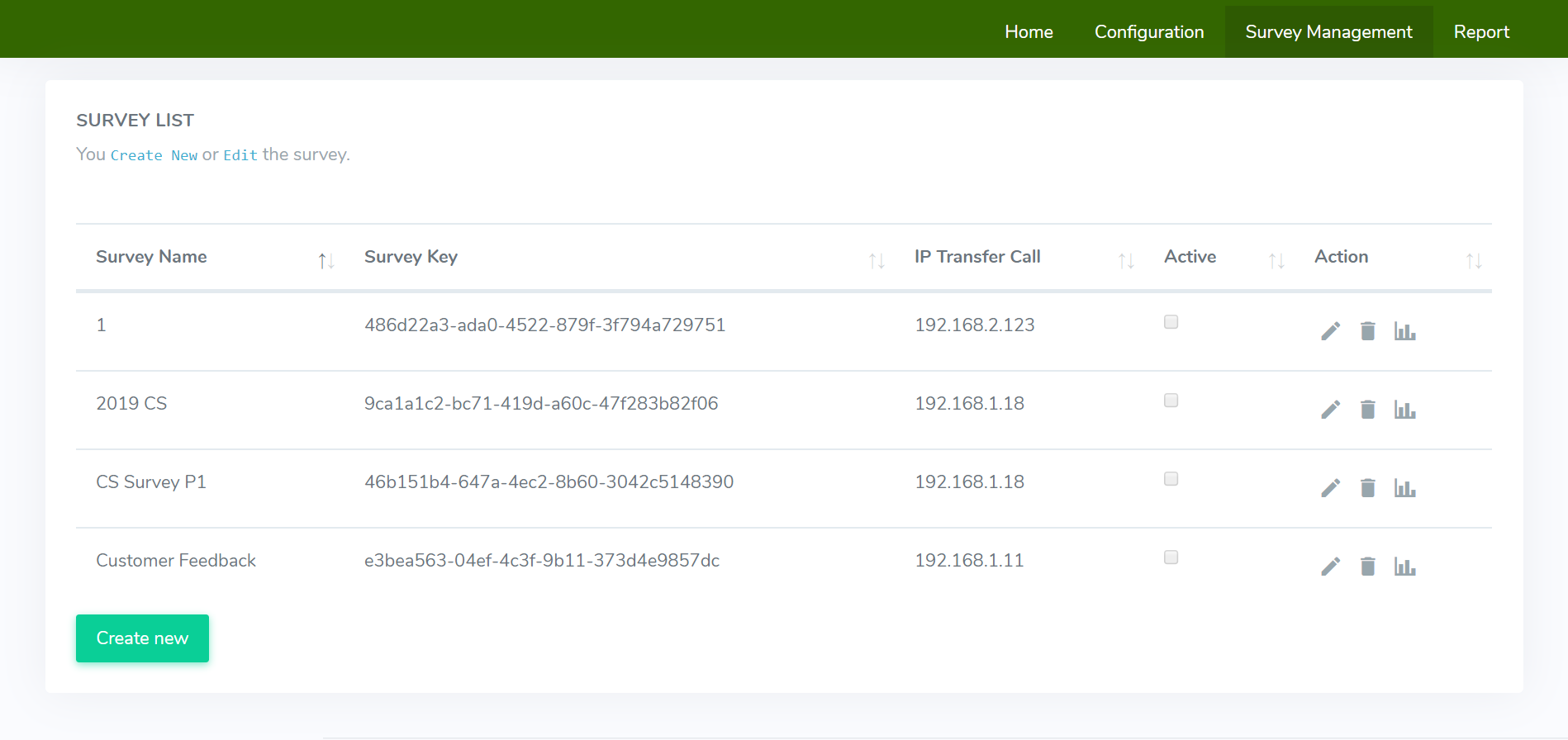


Image 18. Add survey

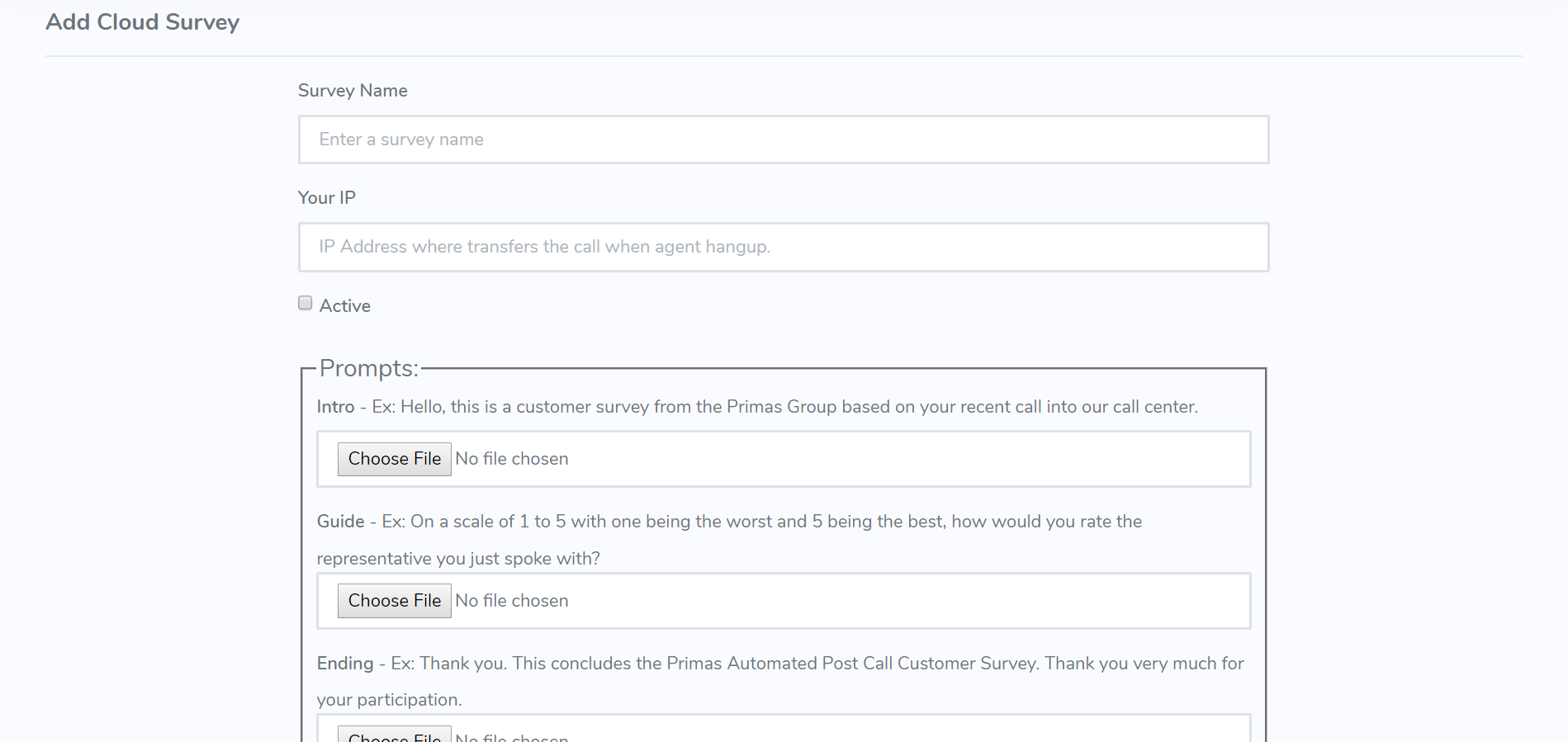


Image 19. Edit survey

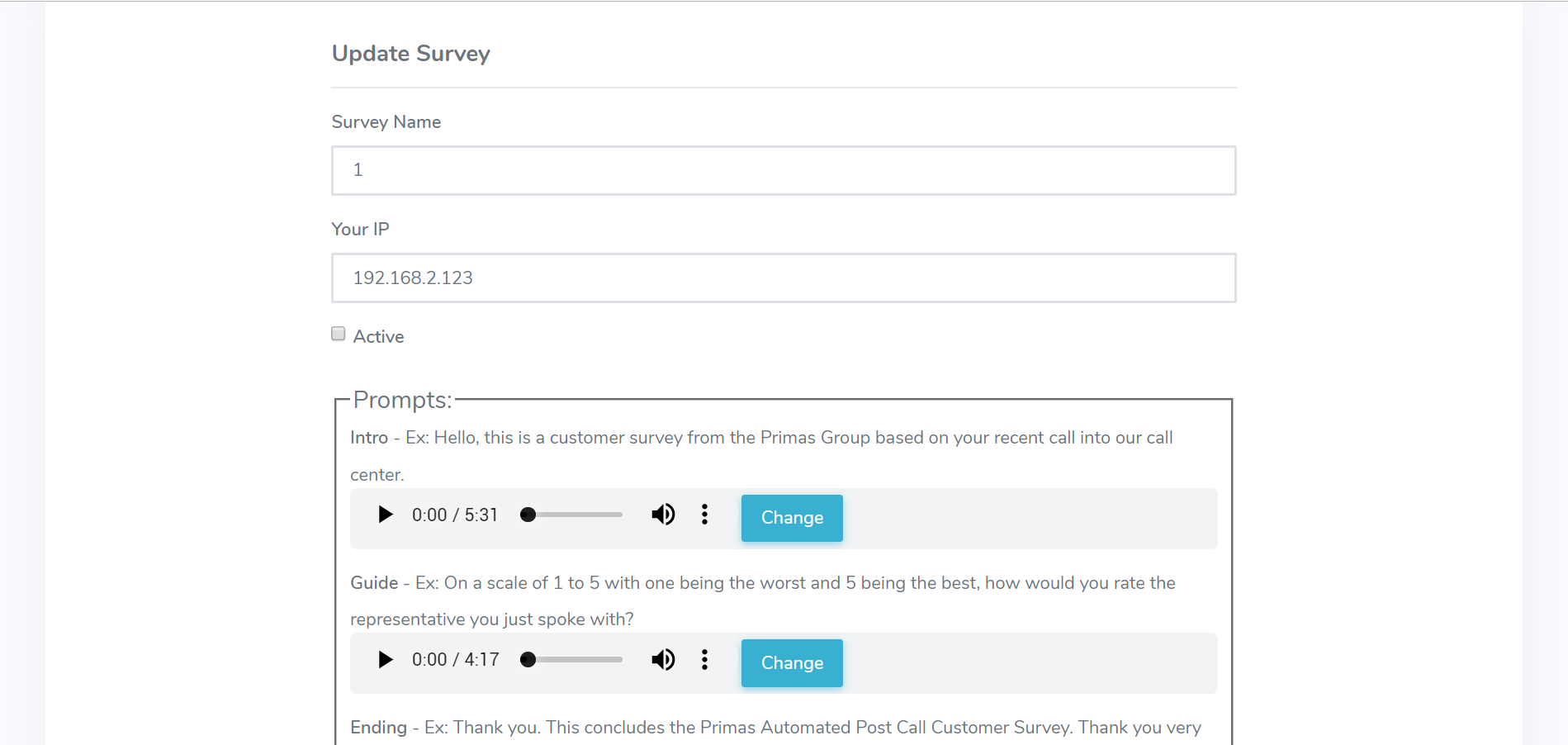
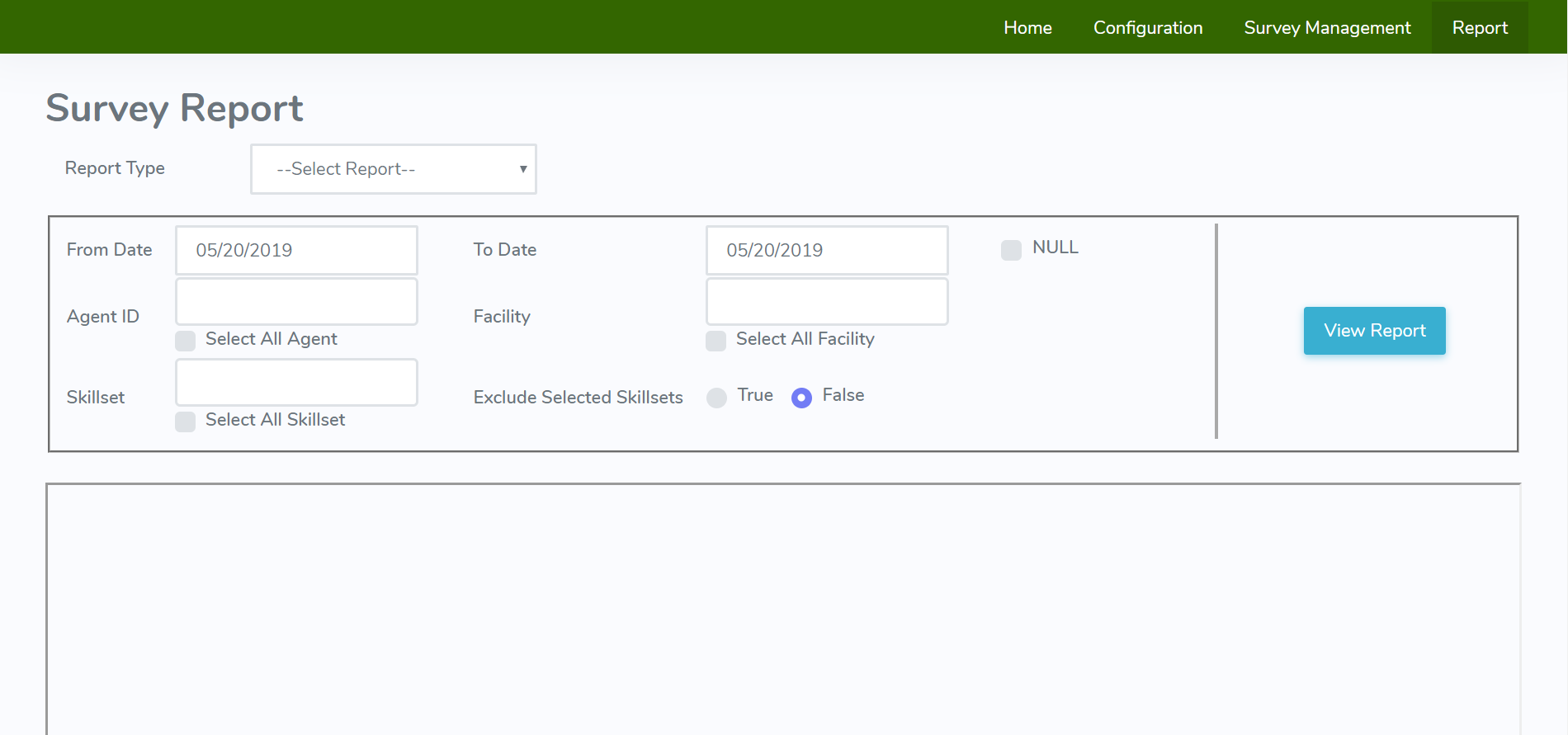


Image 20. Report page



## **2.4. Class diagram**

Image 21. Class diagram

**List of classes are used in the project**

Table 22. List of classes

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Class Name | Responsible | Purpose |
| 1 | iDataAccess  (Interface) | Pham Xuan Khiem | Declares 2 main methods are: ExcutequeryDataset() and UpdateDataTable() which will be used many times by other classes that inherited this interface. |
| 2 | SQLServer | Pham Xuan Khiem | - Declares SqlConnection, SqlDataAdapter and SqlCommand.  - This class is inherited the interface, redefine those methods from interface for using. |
| 3 | MySQL | Pham Xuan Khiem | - Declares MySqlConnection, MySqlDataAdapter and MySqlCommand.  - This class is inherited the interface, redefine those methods from interface for using. |
| 4 | ACCESS | Pham Xuan Khiem | - Declares OleDBConnection, OleDBDataAdapter and OleDBCommand.  - This class is inherited the interface, redefine those methods from interface for using. |
| 5 | PostgreSQL | Pham Xuan Khiem | -Declares NpgsqlConnection, NpgsqlDataAdapter and NpgsqlCommand.  - This class is inherited the interface, redefine those methods from interface for using. |
| 6 | BLL | Tran Tan Quy | This class will have have a static instace of interface, it will also create a new instance of class by checking app.config to see which is the current dbms. |
| 7 | ProductsBL | Tran Tan Quy | This class will inherit the BLL class to get the created instance and start working on it. |
| 8 | LoginBL | Tran Tan Quy | This one is pretty much like ProductsBL class but use for checking login. |
| 9 | frmLogin | Tran Tan Quy | Display interfaces components for user to login. |
| 10 | frmProducts | Tran Tan Quy | Display data, allow user to interact with it. |

**Methods in iDataAccess**

Table 23. Methods in iDataAccess

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p);  Input: strSQL (query), ct and p (parameters).  Output: a data set  Pseudo code: none. | It will return as a data set depends on input queries. It also will be used and be called many times by other classes. | iDataAccess.cs,  Line 13. | Pham Xuan Khiem |
| 2 | int UpdateDataSet(DataSet ds);  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | It will save any changes that happened on the data set. It also will be used and be called many times by other classes. | iDataAccess.cs,  Line 14. | Pham Xuan Khiem |

**Methods in SQLServer**

Table 24. Methods in SQLServer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p);  Input: strSQL (query), ct and p (parameters).  Output: a data set  Pseudo code: none. | It will return as a data set depends on input queries. This is inherited from the interface. | SQLServer.cs,  Line 28. | Pham Xuan Khiem |
| 2 | int UpdateDataSet(DataSet ds);  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | It will save any changes that happened on the data set. This is inherited from the interface. | SQLServer.cs,  Line 38. | Pham Xuan Khiem |
| 3 | SQLServer()  Input: none.  Output: none.  Pseudo code: none. | Initialize class’s components. | SQLServer.cs,  Line 18. | Pham Xuan Khiem |

**Methods in MySQL**

Table 25. Methods in MySQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p);  Input: strSQL (query), ct and p (parameters).  Output: a data set  Pseudo code: none. | It will return as a data set depends on input queries. This is inherited from the interface. | MySQL.cs,  Line 32. | Pham Xuan Khiem |
| 2 | int UpdateDataSet(DataSet ds);  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | It will save any changes that happened on the data set. This is inherited from the interface. | MySQL.cs,  Line 42. | Pham Xuan Khiem |
| 3 | MySQL()  Input: none.  Output: none.  Pseudo code: none. | Initialize class’s components. | MySQL.cs,  Line 23. | Pham Xuan Khiem |

**Methods in ACCESS**

Table 26. Methods in ACCESS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p);  Input: strSQL (query), ct and p (parameters).  Output: a data set  Pseudo code: none. | It will return as a data set depends on input queries. This is inherited from the interface. | ACCESS.cs,  Line 30. | Pham Xuan Khiem |
| 2 | int UpdateDataSet(DataSet ds);  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | It will save any changes that happened on the data set. This is inherited from the interface. | ACCESS.cs,  Line 40. | Pham Xuan Khiem |
| 3 | ACCESS()  Input: none.  Output: none.  Pseudo code: none. | Initialize class’s components. | ACCESS.cs,  Line 20. | Pham Xuan Khiem |

**Methods in PostgreSQL**

Table 27. Methods in PostgreSQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p);  Input: strSQL (query), ct and p (parameters).  Output: a data set  Pseudo code: none. | It will return as a data set depends on input queries. This is inherited from the interface. | PostgreSQL.cs,  Line 28. | Pham Xuan Khiem |
| 2 | int UpdateDataSet(DataSet ds);  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | It will save any changes that happened on the data set. This is inherited from the interface. | PostgreSQL.cs,  Line 38. | Pham Xuan Khiem |
| 3 | PostgreSQL()  Input: none.  Output: none.  Pseudo code: none. | Initialize class’s components. | PostgreSQL.cs,  Line 19. | Pham Xuan Khiem |

**Methods in BLL**

Table 28. Methods in BLL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | BLL()  Input: none.  Output: instance of the interface.  Pseudo code: none. | It will return an instance of interface bases on which dbms that user chose. | BLL.cs,  Line 16. | Tran Tan Quy |
| 2 | GetConfig()  Input: none.  Output: chosen dbms.  Pseudo code: none. | It will get the value from app.config to know which dbms that user has chosen. | BLL.cs,  Line 29. | Tran Tan Quy |

**Methods in ProductsBL**

Table 29. Methods in ProductsBL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | DataSet GetProducts()  Input: none.  Output: data set from a table in specific database.  Pseudo code: none. | It will get data from a specific table in specific database. | ProductsBL.cs,  Line 13. | Tran Tan Quy |
| 2 | int UpdateDataSet(DataSet ds)  Input: a data set.  Output: from -1 to 1 depends action’s status (successful or failed).  Pseudo code: none. | This will call a returned instance from BLL class and its method to update modified data to database. | ProductsBL.cs,  Line 33. | Tran Tan Quy |
| 3 | int AutoIncreasePID(DataTable a)  Input: a data table.  Output: product ID.  Pseudo code: none. | This method will count a datable has how many rows and return with number of rows + 1 so that the product ID won’t be duplicate. | ProductsBL.cs,  Line 37. | Tran Tan Quy |

**Methods in LoginBL**

Table 30. Methods in LoginBL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | bool CheckLogin(string username, string password)  Input: username & password.  Output: true or false.  Pseudo code: none. | Check if users type their username and password correctly. | LoginBL.cs,  Line 13. | Tran Tan Quy |

**Methods in frmLogin**

Table 31. Methods in frmLogin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | txtPassword\_KeyPress(object sender, KeyPressEventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | - After typing a password, user can press enter to login. If username and password are correct, this event will let user access the main form otherwise it will let user know that their account is incorrect.  - This event will call a CheckLogin method from LoginBL class. | frmLogin.cs,  Line 141. | Tran Tan Quy |
| 2 | btnSignIn\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Performs a same action as event txtPassword\_KeyPress but instead of pressing ‘Enter’, user need to click on ‘Sign in’ button. | frmLogin.cs,  Line 172. | Tran Tan Quy |

**Methods in frmProducts**

Table 32. Methods in frmProducts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | frmProducts()  Input: none.  Output: none.  Pseudo code: none. | Initialize class’s components. | frmProducts.cs,  Line 24. | Tran Tan Quy |
| 2 | LoadData()  Input: none.  Output: none.  Pseudo code: none. | - Create columns on data grid view.  - Load data to data grid view and binding it to columns. | frmProducts.cs,  Line 30. | Tran Tan Quy |
| 3 | DatagridViewTextBoxColumn createTextboxColumn(string name, string text)  Input: name, text.  Output: a column.  Pseudo code: none. | Create a column on DatagridView. | frmProducts.cs,  Line 71. | Tran Tan Quy |
| 4 | frmProducts\_Load(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Load data to data table. | frmProducts.cs,  Line 80. | Tran Tan Quy |
| 5 | btnUpdate\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Update any changes on data table to database. | frmProducts.cs,  Line 131. | Tran Tan Quy |
| 6 | btnInsert\_Click(object sender, EventArgs e) | Allow user to insert row(s) on data grid view. | frmProducts.cs,  Line 172. | Tran Tan Quy |
| 7 | btnCancel\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Reject every change on data grid view. | frmProducts.cs,  Line 204. | Tran Tan Quy |
| 8 | btnEdit\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Allow user to edit row(s) on data grid view. | frmProducts.cs,  Line 232. | Tran Tan Quy |
| 9 | btnDelete\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Allow user to delete row(s) on data grid view. | frmProducts.cs,  Line 263. | Tran Tan Quy |
| 10 | btnYesDelete\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Confirm delete a row on data grid view. | frmProducts.cs,  Line 273. | Tran Tan Quy |
| 11 | btnNoDelete\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Cancel delete action. | frmProducts.cs,  Line 285. | Tran Tan Quy |
| 12 | btnOK\_Click(object sender, EventArgs e)  Input: sender, e.  Output: none.  Pseudo code: none. | Confirm insert and edit actions, | frmProducts.cs,  Line 314. | Tran Tan Quy |

## **2.5. Data sources**

Data sources are from available databases in these database management systems:

+ SQL Server DBMS.

+ MySQL DBMS.

+ MS Access DBMS.

+ PostgreSQL DBMS.

## **2.6. Configuration**

Database management systems such as: SQL Server, MySQL, MS Access and PostgreSQL must be installed and work properly in user’s computer.

# ***3. Project implementation***

## **3.1. Enviroment**

- Integrated Development Environment: Visual Studio 2017.

- Programming language: C#.

- NET Framework: Microsoft .NET Framework v4.0.

- Database management systems: SQL Server, MySQL Server, MS Access and PostgreSQL Server.

## **3.2. Work assignment**

Table 33. Work assignment

|  |  |  |
| --- | --- | --- |
| Task | Student’s name | Evaluate contribution |
| Design UI | Tran Tan Quy | 100% |
| Design Class Diagram | Pham Xuan Khiem | 100% |
| Coding | Tran Tan Quy  Pham Xuan Khiem | 100% |
| Testing and debugging | Tran Tan Quy  Pham Xuan Khiem | 100% |
| Writing report. | Tran Tan Quy  Pham Xuan Khiem | 80%-20% |

## **3.3. Work plan**

Table 34. Work plan

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BUILDING A MULTIPLE DATABASE MANAGEMENT SYSTEMS APPLICATION | | | | | | | | | | | |
| No. | Goal | Schedule | | | | | | | | Quy | Khiem |
| 1 | Understand requirement. | **o** | **o** |  |  |  |  |  |  | 🗸 | 🗸 |
| 2 | Describe project requirement. | **o** | **o** |  |  |  |  |  |  | 🗸 | 🗸 |
| 3 | Usecase diagram. |  | **o** | **o** |  |  |  |  |  |  | 🗸 |
| 4 | Design user interface (UI). |  | **o** | **o** |  |  |  |  |  | 🗸 |  |
| 5 | Understanding about interface and use it to design classes. |  |  | **o** | **o** | **o** |  |  |  | 🗸 | 🗸 |
| 6 | Design class diagram. |  |  |  |  | **o** | **o** |  |  |  | 🗸 |
| 7 | Create the project using interface with 3-layer architecture. |  |  | **o** | **o** | **o** | **o** | **o** |  | 🗸 | 🗸 |
| 8 | Actualizing classes base on interface. |  |  | **o** | **o** | **o** | **o** |  |  | 🗸 | 🗸 |
| 9 | Testing and debugging the project. |  |  |  |  |  | **o** | **o** |  | 🗸 | 🗸 |
| 10 | Writing a report. |  |  |  |  |  |  | **o** | **o** | 🗸 |  |
| Day | | 22/10/2018 | 29/10/2018 | 05/11/2018 | 12/11/2018 | 19/11/2018 | 26/11/2018 | 03/12/2018 | 10/12/2018 |  |  |
| Week | | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |  |  |
| Note | | **o** – Begin  **o** – Complete 50%  **o** – Complete 100% | | | | | | | | | |

# ***4. Manual instruction***

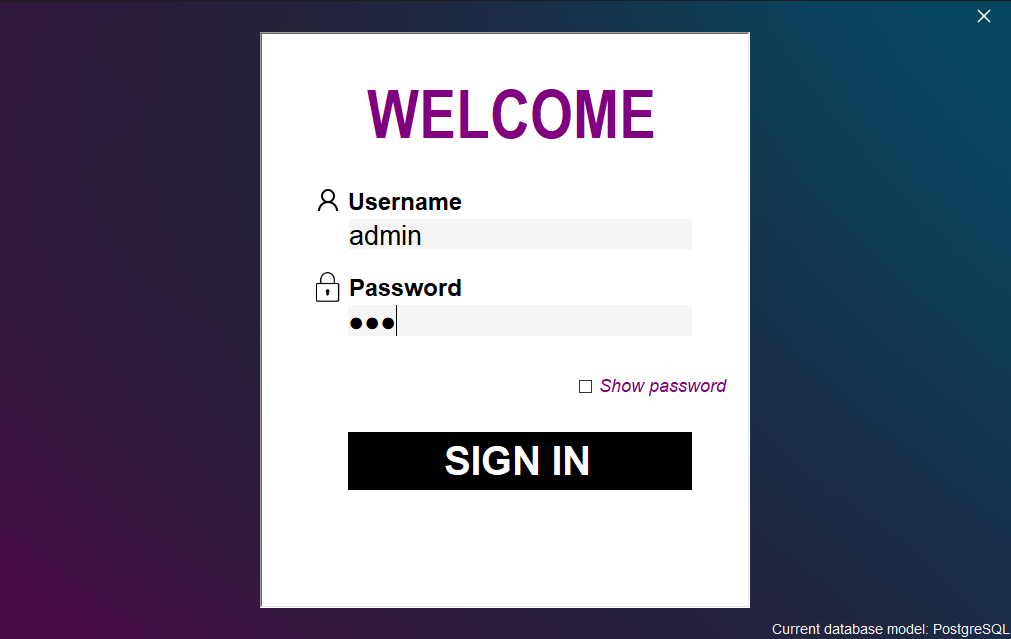


Image 22. Manual instruction

User have to type username and password (default username: admin & default password: 123) then press ‘Enter’ or click on ‘Sign in’ button to access the application. If everything is correct, it will forward user to the main form of the application.

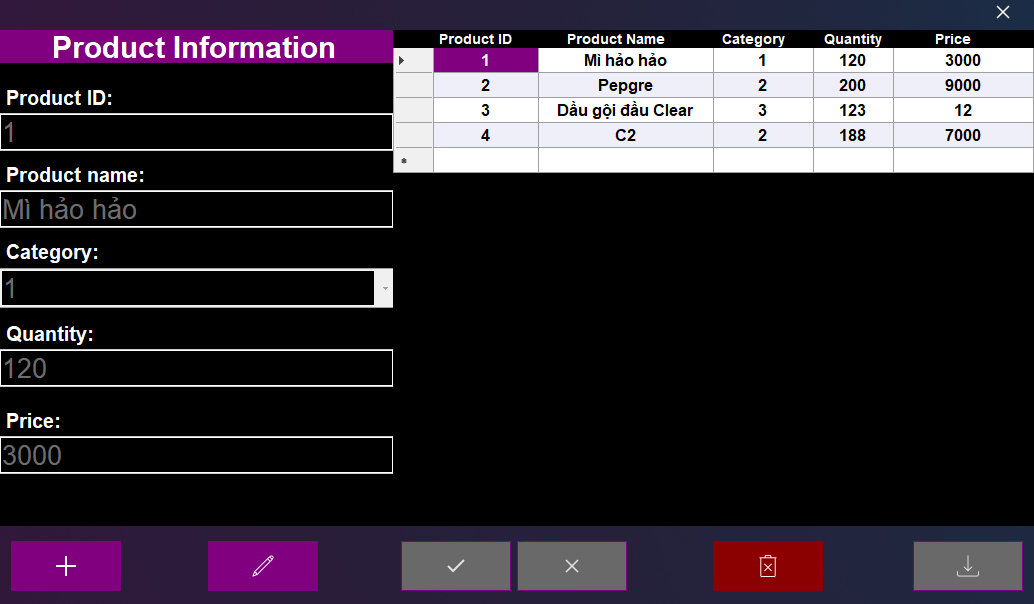


Image 23. Manual instruction main form

- To insert row(s) in data grid view, user need to click on  button, it will enable text boxes on the left and allow user to type data.

- To edit row(s) in data grid view, user need to click on  button, it will also enable text boxes on the left and allow user to modify data on a selected row.

- To delete row(s) in data grid view, user need to click on  button, it will allow user to delete a selected row.

- Whenever user perform those actions above, these , ,  buttons will be enabled. To apply all changes to data grid view, simply click  button. To reject all changes on data grid view, click  to perform the action. After applying changes to data grid view, if user want to update all the changes to database, simply click  button to perform the action.

# ***5. Conclusion***

## **5.1. Results**

- Project completion is about 95%.

- Successfully applied 3-layer architecture as main architecture for this project.

- Successfully applied interface in this project.

- Project can support at least 4 database management systems.

## **5.2. Difficulties**

- Hard to understand about how interface works.

- Couldn’t design project classes that meet requirements at first.

- Couldn’t update modified data to database.

## **5.3. Solutions**

- After spending hours searching for interface definition, how it works. We finally can understand it.

- With lecturer’s help. We are able to design classes that meet requirements.

- After some days of searching, with programming community’s help. Eventually, we can update everything to database.

## **5.4. Advantages**

- Being built with 3-layer architecture. This project is reusable, easy to maintain, improve.

- Nice looking, modern UI, friendly to users.

- Be able to connect to at least 4 database management systems.

- Classes and codes are appropriately designed.

## **5.5. Disadvantages**

- This project can only interact with 1 table at the time.

- Code might be not optimized.

- Only be able to do some simple actions such as: insert, delete, update and view data.

## **5.6. Developing ideas**

- Make this project be able to use more dbms.

- Redesign the project to allow it to interact with more tables.

- Apply client/server model for this project.

## **5.7. Source code**

<https://github.com/quytrandev/TheProject1>

# **REFERENCES**

[1] Asterisk, Get Started, <https://www.asterisk.org/get-started>, accessed 19th May, 2019.

[2] Asterisk Wiki, Beginning Asterisk, <https://wiki.asterisk.org/wiki/display/AST/Beginning+Asterisk>, accessed 19th May, 2019.

[3] StackOverflow, Creating an instance of an interface, <https://stackoverflow.com/questions/7122260/creating-an-instance-of-an-interface>, accessed 9th November, 2018.

[4] HowKteam, Interface in Object-oriented programming, <https://www.howkteam.vn/course/lap-trinh-oop-voi-c/interface-trong-lap-trinh-huong-doi-tuong-1396>, accessed 9th November, 2018.

[5] StackOverflow, <https://stackoverflow.com/questions/19629644/how-to-edit-a-row-in-the-datatable>, accessed 4th December, 2018.