yyyy

C. Report No.3 Software Requirement Specification

1. User Requirement Specification

1.1. House-member requirements

- All Information of home-members (face, phone number, password etc.) will be stored in database. Therefor, house-members can use most of system's functions.

- Unlock facial recognition: When house-members stand in front of the camera, his/her face will be captured, and then the image will be compared with existed images in database. After processing, the door lock will be opened the face is existed in database (home-member's is detected).

- Unlock by keypad: If there are some reasons make home-member can not use facial recognition, system provide a keypad for them to open by typing a passcode. To unlock by keypad, firstly home-member must select "Mở khóa bằng nhận diện khuôn mặt" function. For more convenient, these functions will be shown in LCD screen so user can select it easily. Secondly, User has to enter his/her phone number which registered before, and then the system also compares that number with database. An unlock passcode will be generated and sent to user's phone if the phone number is existed in database and at the same time, the LCD screen will show a passcode textbox for user to enter that received passcode. Finally the door lock will be opened if passcode is correct. The passcode is generated just valid for one hour to ensure about security.

- Receive passcode SMS: This is a sub-function in unlock by keypad function. System will send a random passcode to users and they must used to open the door lock by typing the keypad correctly.

1.2. House-owner requirements

- House-owner is one of house-members with the highest permission. House-owner can use all system functions and he/she can manage member's information in database. Their useable functions are:

- Unlock by facial recognition

- Unlock by keypad

- Receive passcode SMS

- Manage user: home-owner can manage information of member in your house. Phone number is one of member’s information which will be received message from system in case members can not use facial recognition.

- Manage face: home-owner can training a new face for system or block in some cases or remove this face in database of system.

- Show facial record: if users use facial recognition their face will be stored in database that can help house-owner track who access into the system.

- Received warning SMS: If other people not in member's list try to open the door lock, system will log down and sent a SMS to house-owner. Camera also captures that person's photo and saved to visited log.

1.3. Guest requirement

- A guest can interact with the system through LCD screen and keypad. So that they are able to use some function below:

- Unlock using keypad: like the house-member, guest can switch to "Unlock using keypad" by pressing a functional key. But after that, guest has to input phone number of house-members and the passcode will be generated and sent to house-members. Then if house-member allows guest to enter the house, he/she will show guest the unlock passcode which he/she received from the system.

2. System Requirement Specification

2.1 External Interface Requirement

2.1.1. User interface

- User interface use Graphical User Interface must be simple, clear and easy to use. Each screen will have the instruction to guide user how to use it.

2.1.1.1. Main Menu

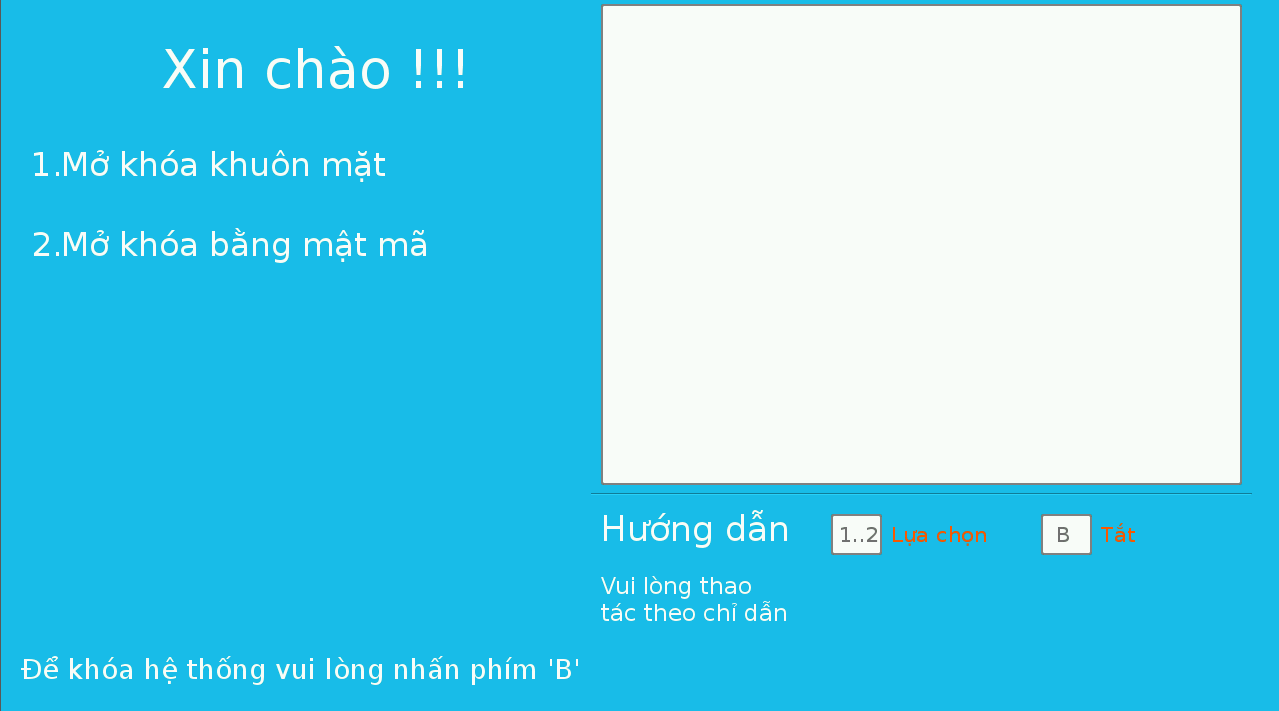


Figure 3. “Main menu” screen

2.1.1.2. System management

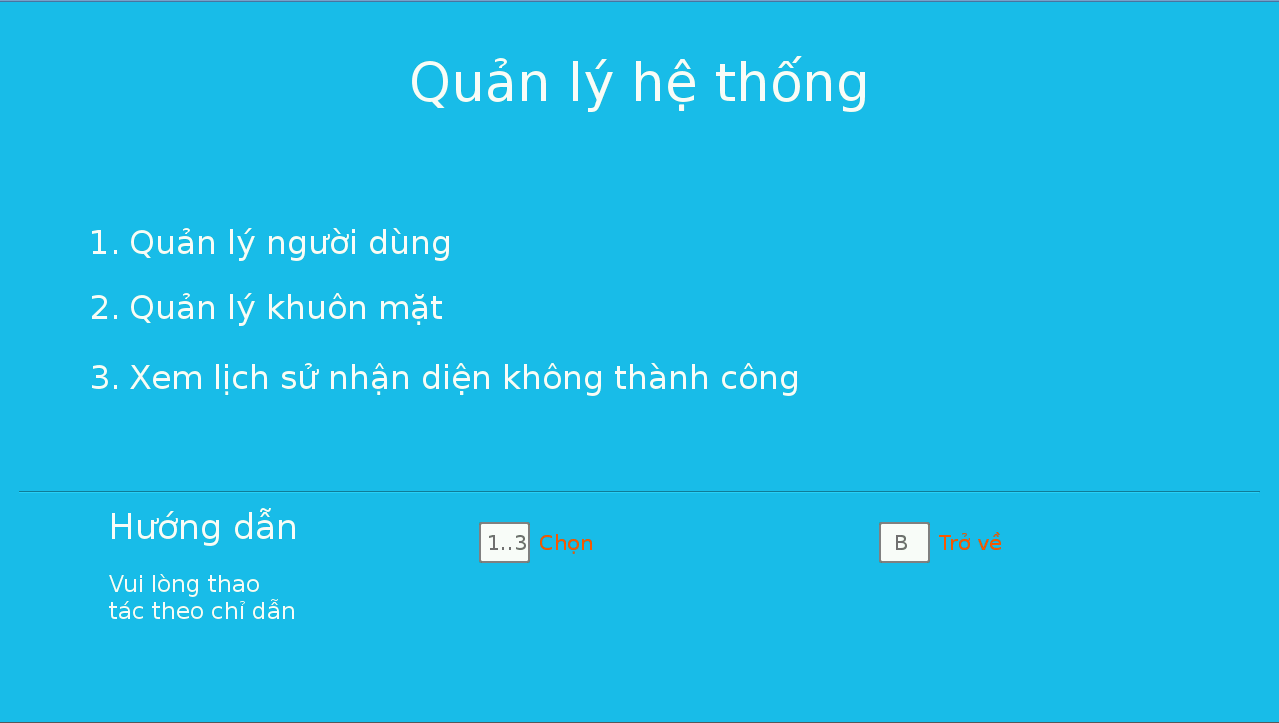


Figure 4. “System management” screen

2.1.1.3. User management

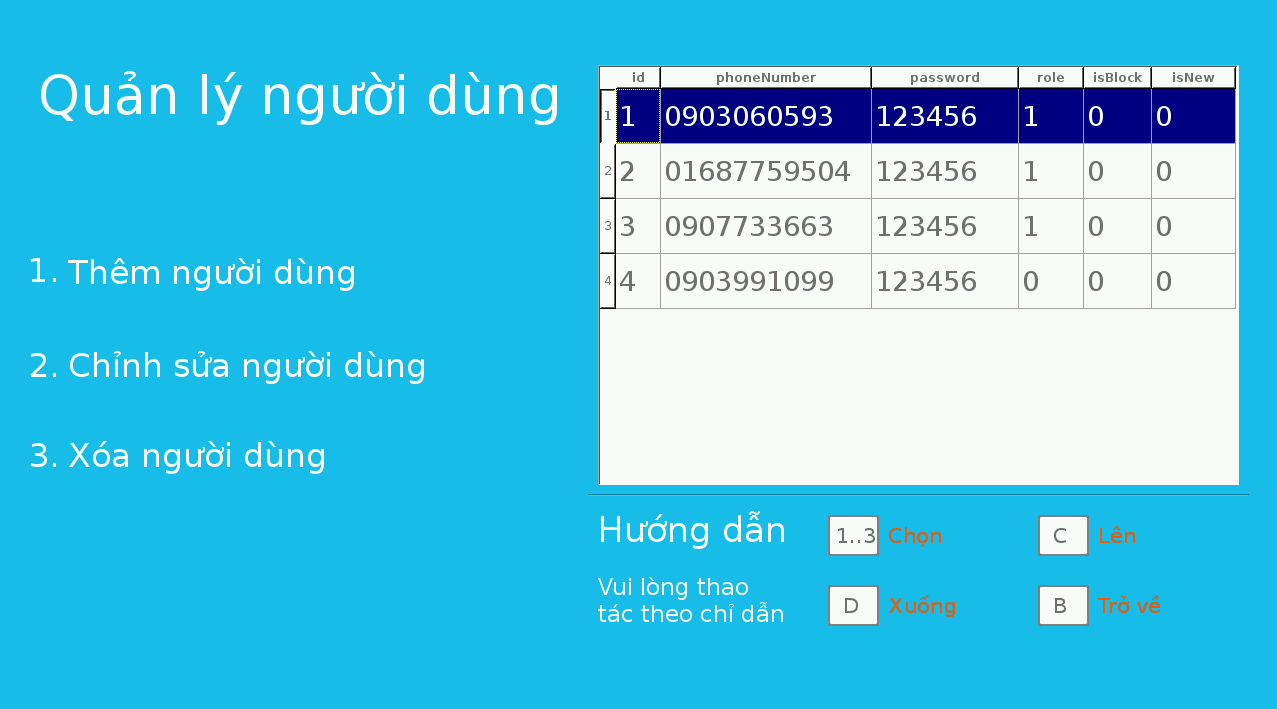


Figure 5. “User management” screen

2.1.1.4 Add user

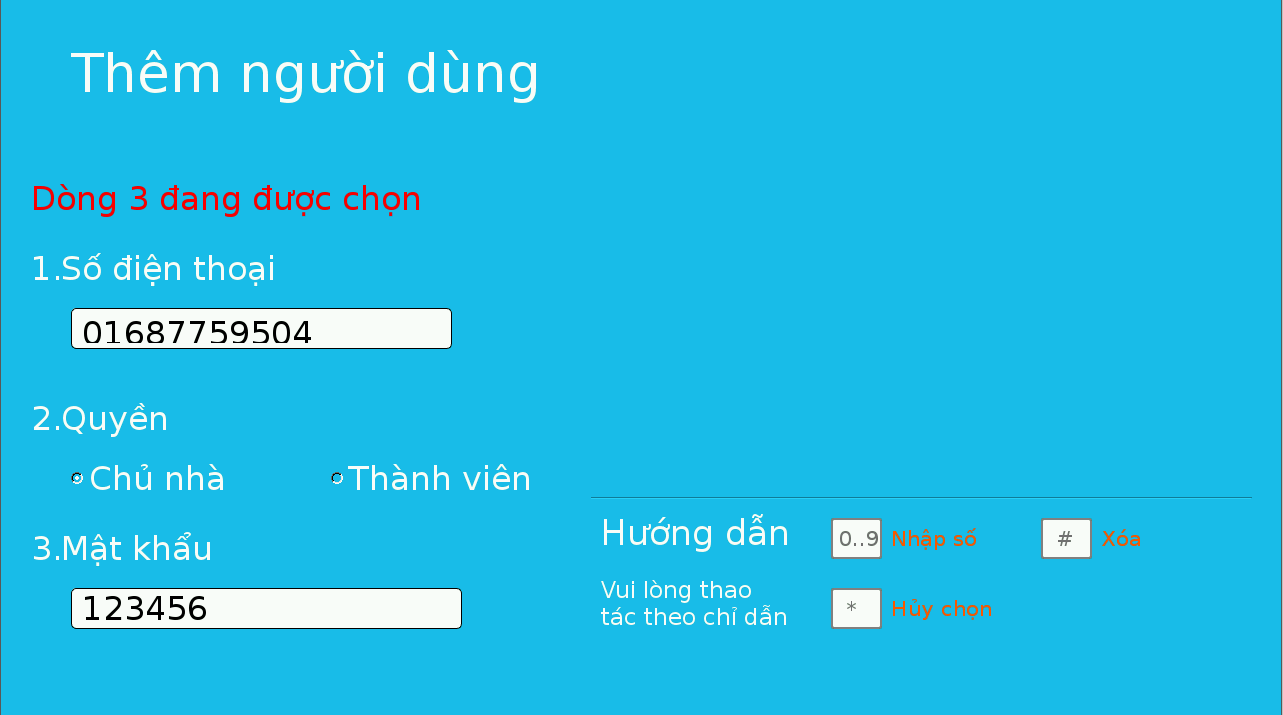


Figure 6. “Add user” screen

2.1.1.5. Modify User



Figure 7. “Modify user” screen

2.1.1.6. Manage face

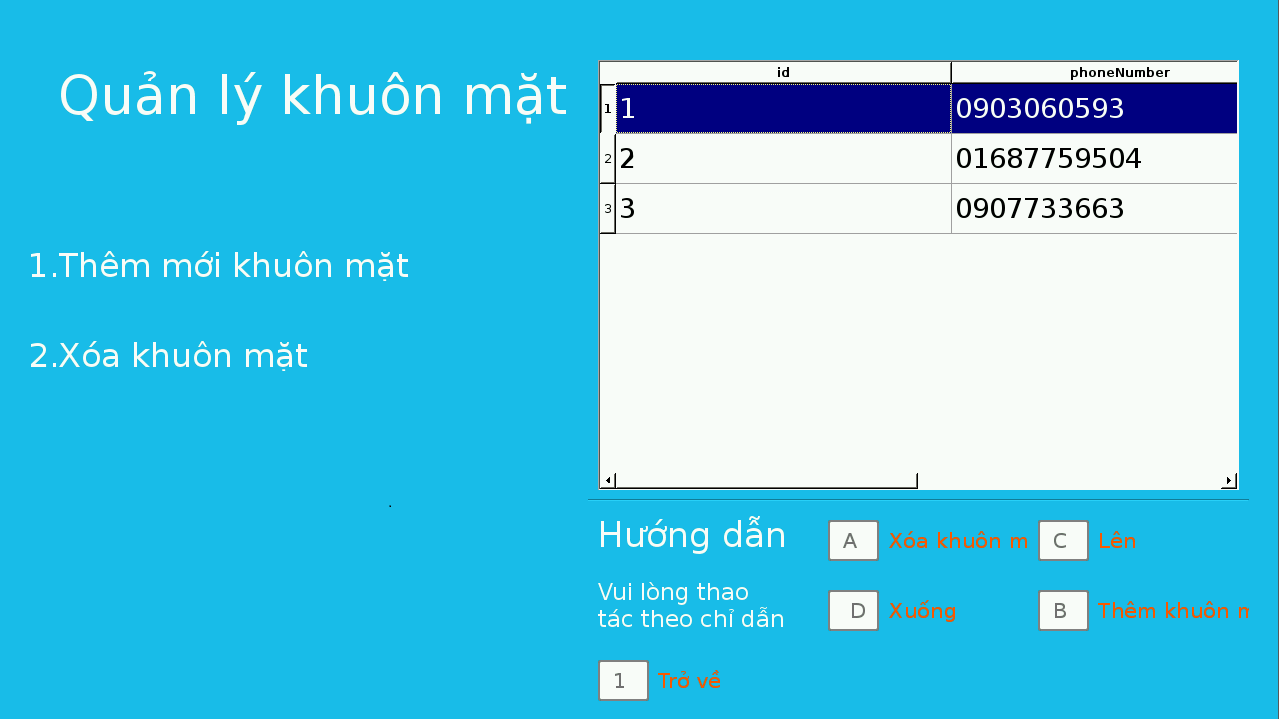


Figure 8. “Face management” screen

2.1.1.7. Add face

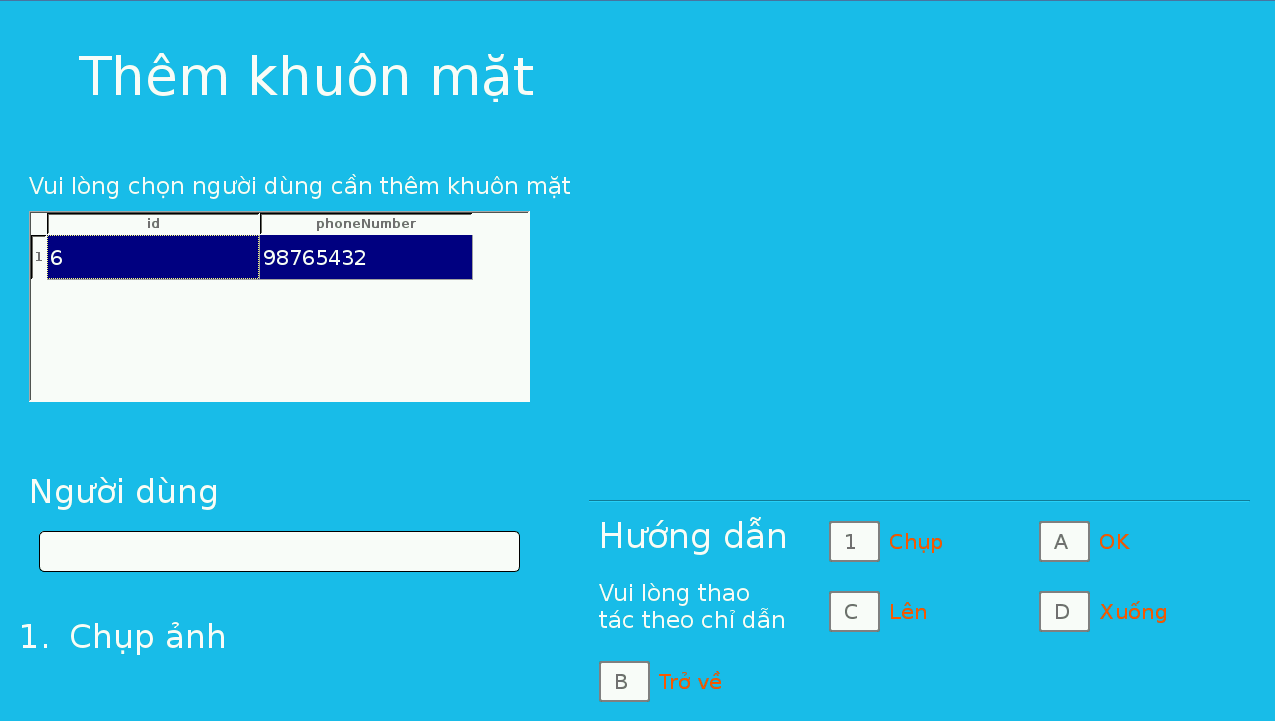


Figure 9. “Add face” screen

2.1.1.8. Show log face

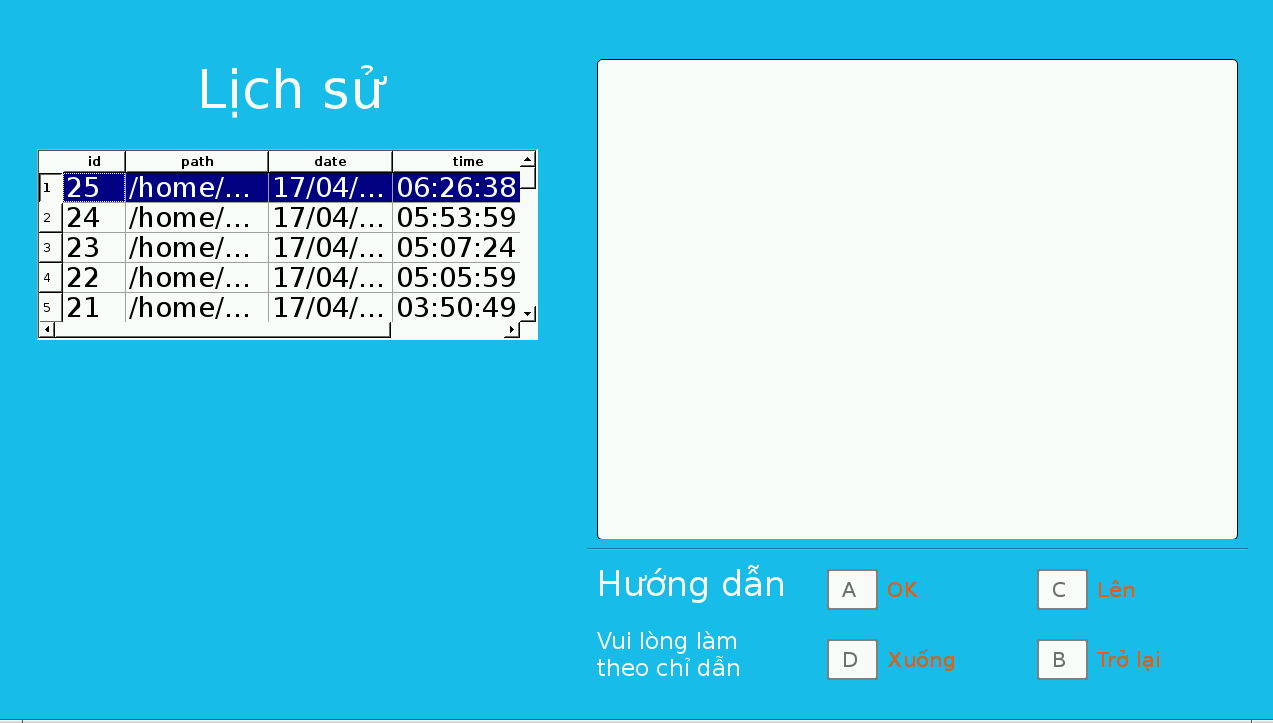


Figure 10. “Show log face” screen

2.1.1.9. Keypad mode

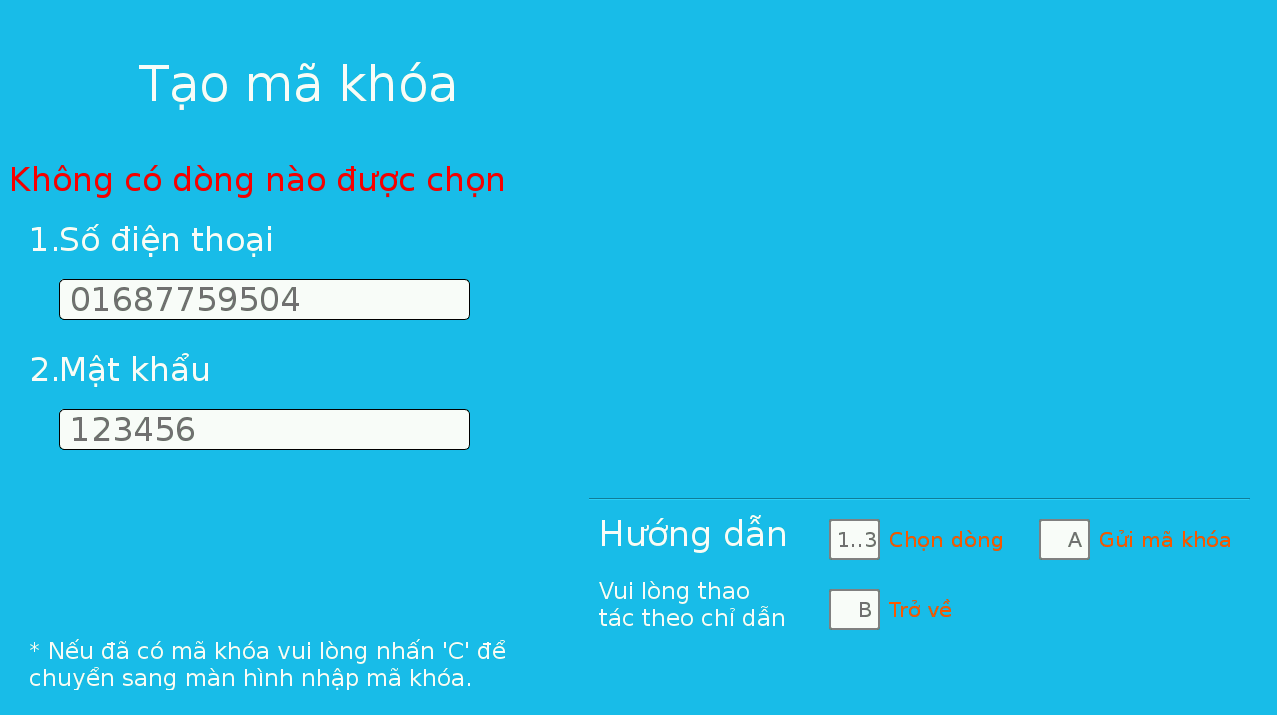


Figure 11. “Keypad mode” screen

2.1.1.10 Enter passcode

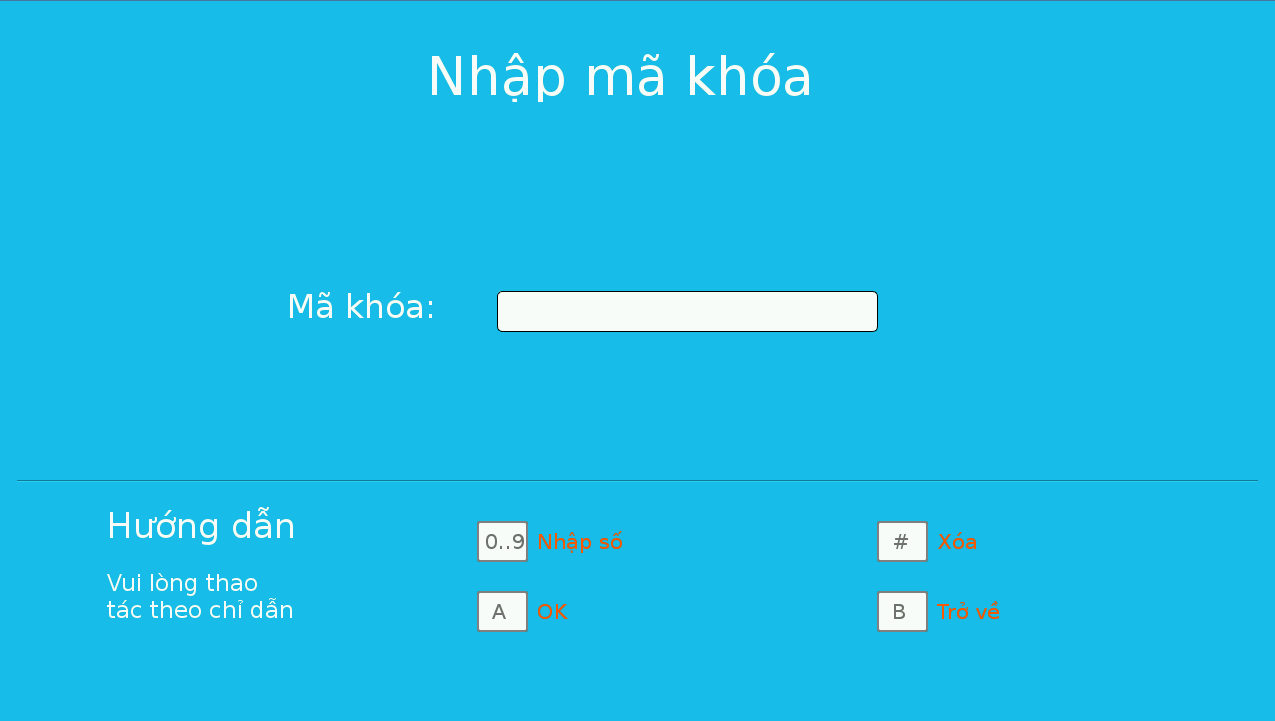


Figure 12. “Enter passcode” screen

2.1.1.11. Setting screen



Figure 13. “Setting” screen

2.1.1.12. Alert screen

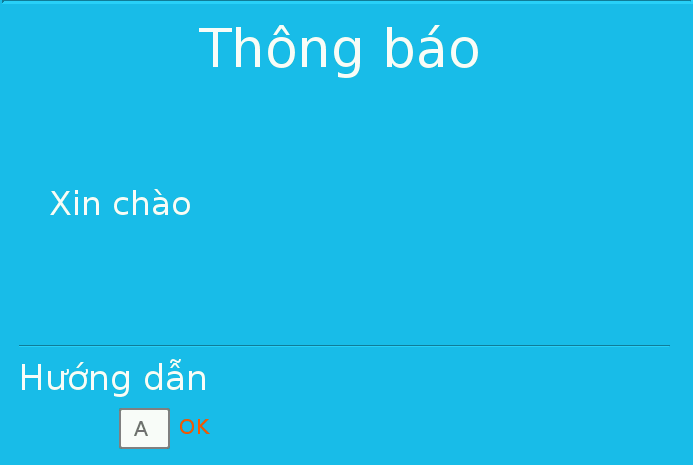


Figure 14. “Alert” screen

2.1.1.13. Confirm screen

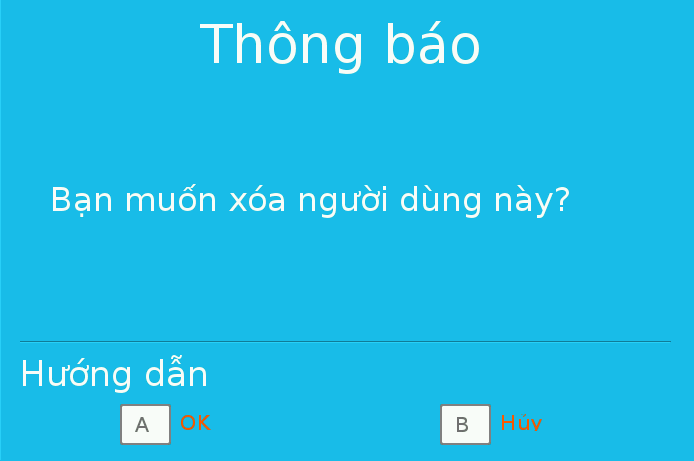


Figure 15. “Confirm” screen

2.1.2. Hardware interface

- Raspberry Pi B2 with SDRAM 1GB, Quad-core, 900MHz ARM Cortex-A7 chip

- Number keypad have 0-9 number, ‘\* #’ special characters and ‘A B C D’ characters.

- Camera module of Raspberry Kit.

- Cabinet Door Electric Lock Assembly Solenoid DC12V 0.6A Square bevel latch

2.1.3. Software Interface

- QT 4 Development Tools Version v4.8.2 and Q make version 2.01a

2.2 System Overview Use Case

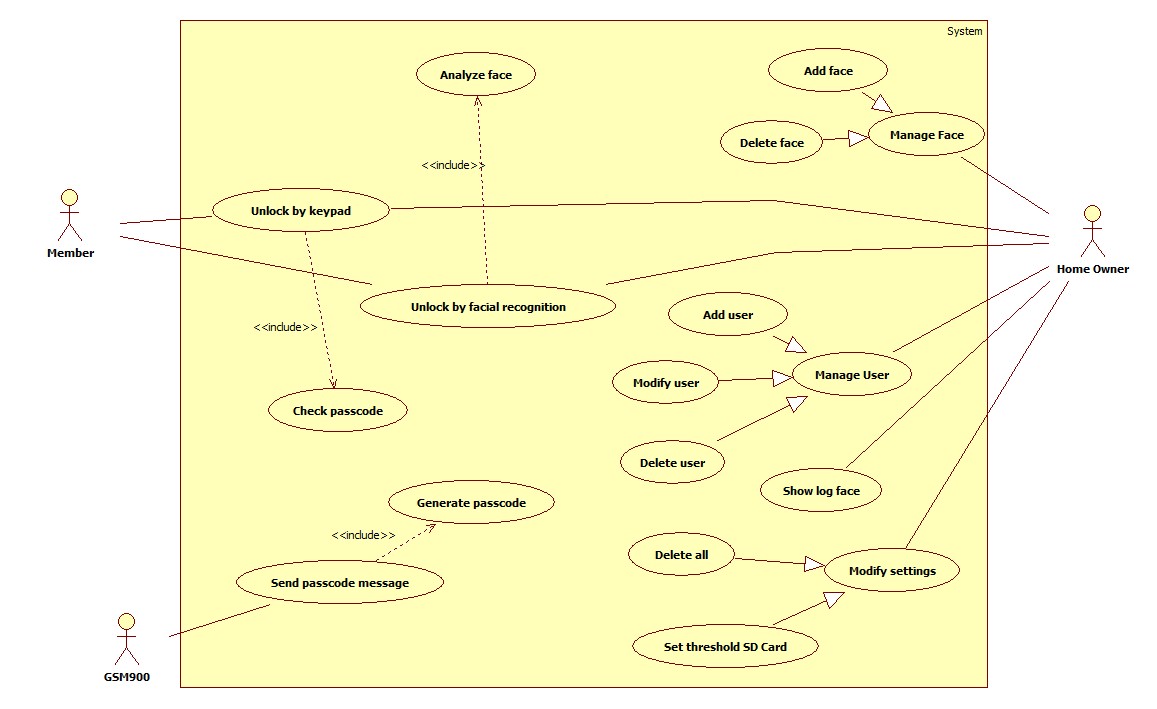


Figure 16. System overview use case

2.3. List of Use Case

2.3.1. <Member>,<Home owner> Overview Use Case

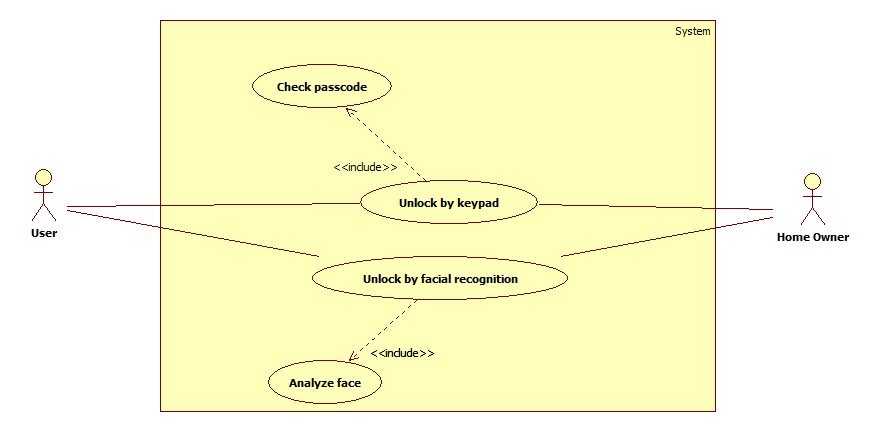


Figure 17 . <Member>,<Home owner> Overview use case

2.3.1.1. Unlock by keypad

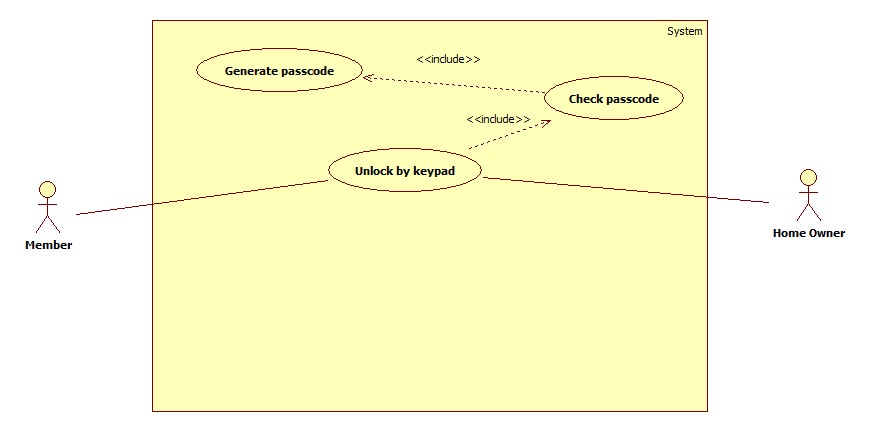


Figure 18. “Unlock by keypad” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC01** | | | |
| **Use Case No.** | **UUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Unlock by keypad | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Member  - Home owner  **Summary:**  - In case user can not unlock by facial recognition, user will use unlock by keypad mod to unlock. After switch mode, user will enter the passcode which is received by registered phone number.  **Goal:**  - User will unlock successfully.  **Triggers:**  - Press “2” button on keypad to open unlock keypad screen.  **Preconditions:**  - User entered “Unlock by keypad” screen.  **Post Conditions:**  - Success: User opens lock successfully. Lock will be opened.  - Fail: User can not unlock. Dialog will show message base on exception.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “2” button on keypad  [Alternative Scenario] | LCD will show screen with contains the following information  - Phone number: label  - Password: label | | 2 | Press “1” button on keypad | Phone number will be selected for user entered his phone number | | 3 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 4 | Press “2” button on keypad | Password will be selected for user entered password | | 5 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 6 | Press “A” button on keypad  [ Exception 1] | System will generate passcode and open “Enter passcode” screen. | | 7 | Press “0..9” button on keypad | Entered number of passcode | | 8 | Press “A” button on keypad  [ Exception 2]  [ Exception 3] | System check passcode and lock will be opened. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “C” button on keypad | Open “Enter passcode” screen. | | 2 | Press “0..9” button on keypad | Entered number of passcode | | 3 | Press “A” button on keypad | System check passcode and lock will be opened. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User entered phone number which is not registered by home owner or invalid password | Dialog will show “Số điện thoại chưa được đăng kí hoặc mật khẩu không đúng” message. | | 2 | Press invalid passcode | Dialog will show “Mã khóa không đúng.” message. | | 3 | Press expired passcode | Dialog will show “Mã khóa đã hết hạn.” message. |   **Relationships:**  - Have <<include>> relationship with “Check passcode” use case.  **Business Rules:**  - Users will use unlock by keypad in case they are guests or they do not have permission to unlock by facial recognition.  - Users must have a phone number which is registered by home owner to receive the passcode.  - User will enter the passcode which is received to unlock.  - The expired time of the passcode is one hour after this is generated by system. And the passcode is one time passcode. | | | |

Table 9. ”Unlock by keypad” specification.

2.3.1.2. Unlock by facial recognition

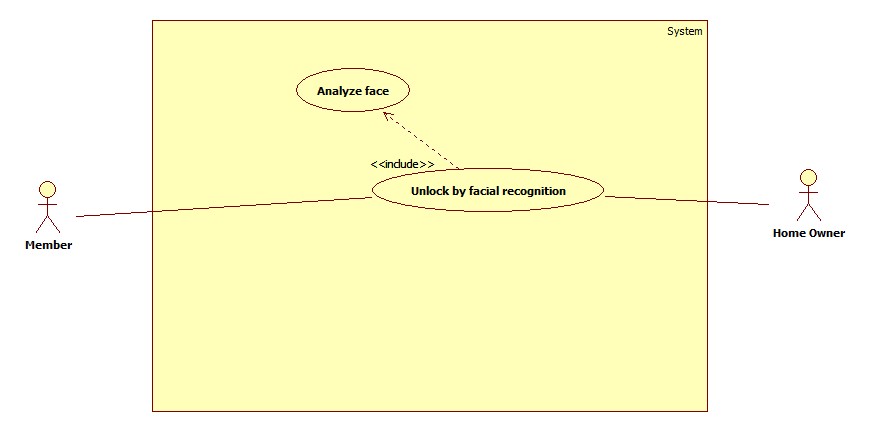


Figure 19. “Unlock by facial recognition” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC02** | | | |
| **Use Case No.** | **UUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Unlock by facial recognition | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Member  - Home owner  **Summary:**  - User will be opened lock by using face. User will stand in front of camera for system recognize, if system recognize successfully, lock will be opened.  **Goal:**  - User will unlock successfully.  **Triggers:**  - Press “1” button on keypad in “Menu” screen  **Preconditions:**  - User use mode unlock by facial recognition.  **Post Conditions:**  - Success: Lock will be opened.  - Fail: User can not unlock. Dialog will show message base on exception.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “1” button on keypad | Face of user will be shown in screen | | 2 | Stand in front of camera about 3-5 seconds and look at the camera.  [ Exception 1]  [ Exception 2] | Lock will be opened if system recognize successfully. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | System can not recognize face | Dialog will show “Mở khóa không thành công” message. | | 2 | User is blocked | Dialog will show “Người dùng đã bị khóa” message. |   **Relationships:**  - Have <<include>> relationship with “Analyze face” use case.  **Business Rules:**  - Users will press “1” to active system and camera then stand front of the camera and check in LCD the position of their face and adjust for system easy recognition. | | | |

Table 10.”Unlock by facial recognition” specification.

2.3.2. <Home Owner> Overview Use Case

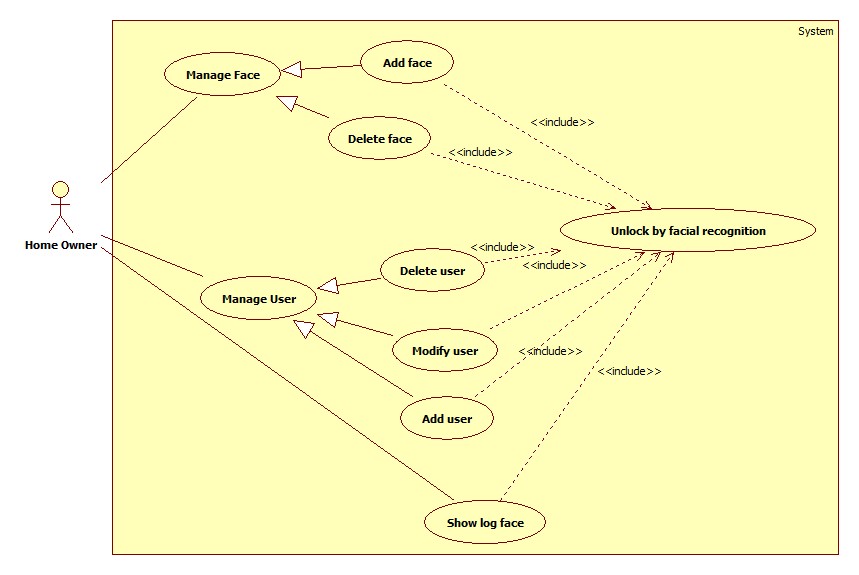


Figure 20. <Home owner> Overview use case

2.3.2.1. Add face

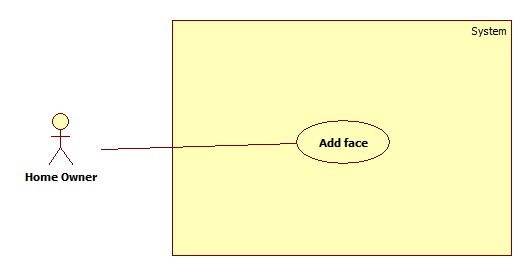


Figure 21. “Add face” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC01** | | | |
| **Use Case No.** | **HUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Add face | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will add list of face using when system recognize face. System will base on list of face to recognize and unlock if the user has face in database.  **Goal:**  - New face will be added into database.  **Triggers:**  - Press “1” button on keypad in “Face management” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  - Go to “Face management” screen  **Post Conditions:**  - Success: Add new face in database successfully. Message will be shown in LCD screen  “Thêm khuôn mặt thành công”  - Fail: User can not add new face in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “C” button to up and “D” button to down on keypad.  [ Exception 1] | Name of member will be changed base on user scroll the list | | 2 | Press “A” button on keypad | Select user will be added face | | 3 | Stand in front of camera then press “1” button on keypad. | System will capture new face and save it to database. After capture 10 pictures, dialog will show “Thêm khuôn mặt mới thành công.” message. | | 4 | Press “A” button to back “Face management” screen. | “Face management” screen will be shown. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | User press “1” button on keypad | Dialog will show “Vui lòng chọn người dùng cần thêm khuôn mặt” message. |   **Relationships:**  - N/A  **Business Rules:**  **-**This face will belong user which is added so before add new face you need to add new user, this user will has attribute to let system know that user has not have face. Then system will show the name of users who have not face in table.  - 10 pictures will be captured then executed after save into database. | | | |

Table 11.”Add face” specification.

2.3.2.2. Delete face

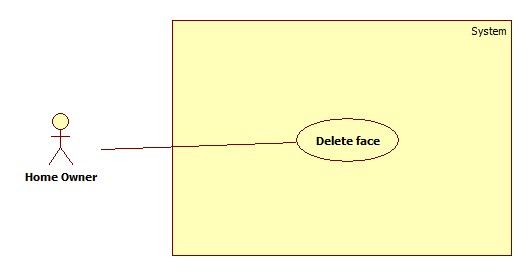


Figure 22. “Delete face” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC02** | | | |
| **Use Case No.** | **HUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Delete face | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will delete face in database when user wants to add new face into database. System will base on list of face to recognize and unlock if the user has face in database.  **Goal:**  - New face will be added into database.  **Triggers:**  - Press “2” button on keypad in “System management” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  **Post Conditions:**  - Success: Delete face in database successfully. Message will be shown in LCD screen “Xóa khuôn mặt thành công”.  - Fail: User can not delete face in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “C” button to up and “D” button to down on keypad. | Name of member will be changed base on user scroll the list | | 2 | Press “2” button on keypad  [Exception 1] | Confirm dialog will show”Bạn muốn xóa khuôn mặt này?” message. | | 3 | Press “A” button on keypad | Face will be deleted in database |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | Press “B” button on keypad | Face will not be deleted in database |   **Relationships:**  **-** N/A  **Business Rules:**  **-** Table will list all users who have existed face in database. After user delete face, user can use add face to add new face. If user does not have any face in database this user will not appear in table. | | | |

Table 12.”Delete face” specification.

2.3.2.3. Add user

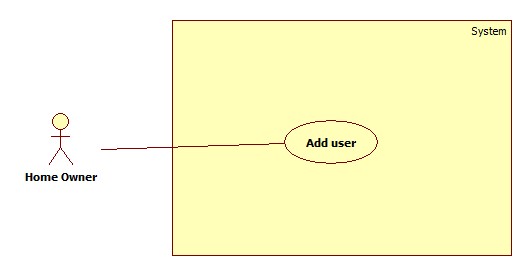


Figure 23. “Add user” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC03** | | | |
| **Use Case No.** | **HUC03** | **Use Case Version** | 2.0 |
| **Use Case Name** | Add user | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will add new user into database that means new user can unlock by using facial recognition or unlock by keypad.  **Goal:**  - New user will be added into database.  **Triggers:**  - Press “1” button on keypad in “User management” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  - Go to “User management” screen.  **Post Conditions:**  - Success: Add new user in database successfully. Message will be shown in LCD screen “Thêm người dùng thành công”  - Fail: New user can not be added in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “1” button on keypad  [ Exception 1]  [ Exception 2] | Phone number will be selected for user entered his phone number | | 2 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 3 | Press “2” button on keypad  [ Exception 2] | Role will be selected for user choose role | | 4 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 5 | Press “3” button on keypad  [ Exception 1]  [ Exception 2] | Password will be selected for user entered password | | 6 | Press “\*” button on keypad | System will be exchange no lie is selected to wait next action. | | 7 | Press “A” button on keypad | New user will be added into database |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | Press character | Dialog will show “Vui lòng nhập số” message. | | 2 | Press “A” button | Dialog will show “Vui long điền đầy đủ thông tin” message. |   **Relationships:**  **-** N/A  **Business Rules:**  **-** User has home-owner or member role when add user.  - After add new user, user can add new face. | | | |

Table 13.”Add user” specification.

2.3.2.4. Modify user

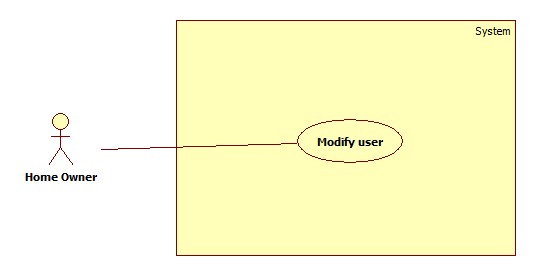


Figure 24. “Modify user” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC04** | | | |
| **Use Case No.** | **HUC04** | **Use Case Version** | 2.0 |
| **Use Case Name** | Modify user | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will update user’s information includes phone number, password, status of user into database.  **Goal:**  - New information of user will be updated into database.  **Triggers:**  - Press “2” button on keypad in “User management” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  - Go to “User management” screen, select user to update by press “C” button to up and “D” button to down.  **Post Conditions:**  - Success: Update user in database successfully. Message will be shown in LCD screen “Cập nhật người dùng thành công”.  - Fail: New user can not be added in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “1” button on keypad  [ Exception 1]  [ Exception 2] | Phone number will be selected for user entered his phone number | | 2 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 3 | Press “2” button on keypad  [ Exception 2] | Role will be selected for user choose role | | 4 | Press “\*” button on keypad | System will be exchange no line is selected to wait next action. | | 5 | Press “3” button on keypad  [ Exception 1]  [ Exception 2] | Password will be selected for user entered password | | 6 | Press “\*” button on keypad | System will be exchange no lie is selected to wait next action. | | 7 | Press “A” button on keypad | Update new information of user into database. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | Press character | Dialog will show “Vui lòng nhập số” message. | | 2 | Press “A” button | Dialog will show “Vui long điền đầy đủ thông tin” message. |   **Relationships:**  **-** N/A  **Business Rules:**  **-** User who is blocked can not unlock by using facial recognition or unlock by keypad | | | |

Table 14.”Update user” specification.

2.3.2.5. Delete user

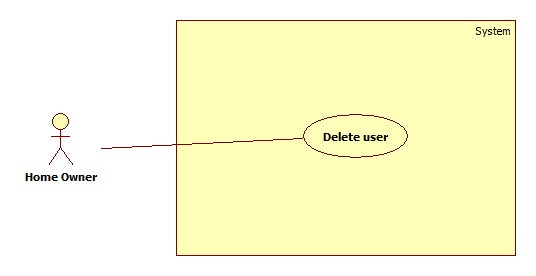


Figure 25. “Delete user” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC05** | | | |
| **Use Case No.** | **HUC05** | **Use Case Version** | 2.0 |
| **Use Case Name** | Delete user | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will delete face in database when user wants to add new face into database. System will base on list of face to recognize and unlock if the user has face in database.  **Goal:**  - New face will be added into database.  **Triggers:**  - Press “3” button on keypad in “User management” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  - Go to “User management” screen, select user to delete by press “C” button to up and “D” button to down.  **Post Conditions:**  - Success: Delete user in database successfully. Message will be shown in LCD screen. “Xóa người dùng thành công”.  - Fail: User can not delete user in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 3 | Press “A” button on keypad | User will be deleted in database |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “C” button to up and “D” button to down on keypad. | Name of member will be changed base on user scroll the list | | 2 | Press “3” button on keypad | Confirm dialog will show”Bạn muốn xóa người dùng?” message. | | 3 | Press “B” button on keypad | Face will be deleted in database |   **Relationships:**  **-** N/A  **Business Rules:**  **-** Table will list all users who have existed face in database. After user delete face, user can use add face to add new face. If user do not have any face in database this user will not appear in table. | | | |

Table 15.”Delete user” specification.

2.3.2.6. Show log face

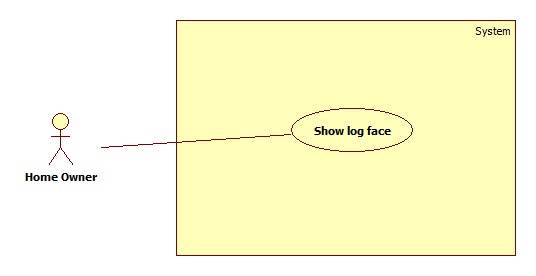


Figure 26. “Show log face” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC06** | | | |
| **Use Case No.** | **HUC06** | **Use Case Version** | 2.0 |
| **Use Case Name** | Show log face | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner can view history of face, user who used to use facial recognition mode.  **Goal:**  - Show image of who used to use facial recognition mode.  **Triggers:**  - Press “A” button on keypad  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  - Go to “Show Log Face” screen, then select face want to see by press “C” button to up and “D” button to down.  **Post Conditions:**  - Success: Show face in view  **Main Success Scenario:**  **-** N/A  **Alternative Scenario:**  **-** N/A  **Exceptions:**  - N/A  **Relationships:**  **-** N/A  **Business Rules:**  **-** Face in system will be shown with nearly date. This log face can be removed if the storage of SD card is full. | | | |

Table 16.”Show log face” specification.

2.3.2.7. Delete all information

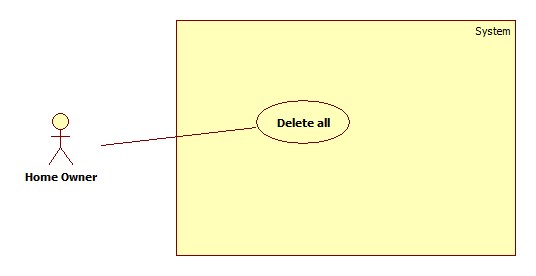


Figure 27. “Delete all information” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC07** | | | |
| **Use Case No.** | **HUC07** | **Use Case Version** | 2.0 |
| **Use Case Name** | Delete all information | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 21/05/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner will delete all data in database in case he/she wants to renew system.  **Goal:**  - All data will be deleted.  **Triggers:**  - Press “1” button on keypad in “Modifying setting” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  **Post Conditions:**  - Success: Delete all data in database succesfully. Message will be shown in LCD screen “Xóa thông tin thành công”.  - Fail: User can not delete data in database. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “A” button on keypad | Confirm dialog will show”Bạn muốn xóa tất cả dữ liệu?” message. | | 3 | Press “A” button on keypad | Data will be deleted in database |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | Press “B” button on keypad | Data will not be deleted in database |   **Relationships:**  **-** N/A  **Business Rules:**  **-** All information of system include information of user, information of face, information of log face will be deleted. The system is renew after this user case | | | |

Table 17.”Delete all information” specification.

2.3.2.8. Set threshold for SD card

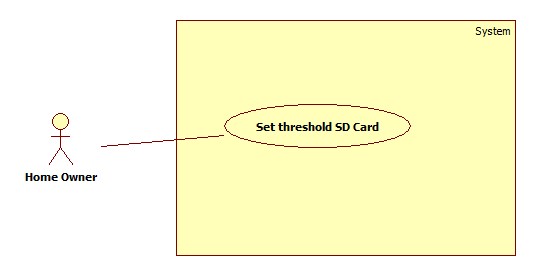


Figure 28. “Set threshold SD Card” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC08** | | | |
| **Use Case No.** | **HUC08** | **Use Case Version** | 2.0 |
| **Use Case Name** | Set threshold SD Card | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 21/05/2015 | **Priority** | High |
| **Actor:**  - Home owner  **Summary:**  - Home owner can set threshold for system when log face data will be deleted. If size which is used to store log face is increase and equal with threshold user will be notified to delete all log faces in database.  **Goal:**  - Log faces will be deleted when the used size of SD card is caught the threshold.  **Triggers:**  - Press “2” button on keypad in “Modifying setting” screen  **Preconditions:**  - Using facing recognition to unlock with “Home owner” role.  **Post Conditions:**  - Success: Set threshold successfully. Message will be shown in LCD screen “Cài đặt thành công”.  - Fail: User can not set threshold. Warning message will be shown in LCD screen “Bị lỗi trong quá trình thực thi. Vui lòng thực hiện lại sau”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “1” button on keypad  Press “2” button on keypad  Press “3” button on keypad | “3GB” will be selected.  “6GB” will be selected.  “8GB” will be selected | | 3 | Press “A” button on keypad | Setting will be saved in configuration file. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | Press “B” button on keypad | Setting will not be saved in configuration file. |   **Relationships:**  **-** N/A  **Business Rules:**  **-** The ability to store data of SD card is limited. In case too much log faces are saved can me the storage full. This use case can help use set threshold which they want all log faces will be deleted if the used size is caught the threshold. | | | |

Table 18.”Set threshold of SD Card” specification.

2.3.3. <System> Overview Use Case

2.3.3.1. Analyze face

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SUC01** | | | |
| **Use Case No.** | **SUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Analyze face | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - System  **Summary:**  - System will analyze the face of user at real time and compare it with other faces in database to determine that which user can unlock and can not.  **Goal:**  - Provide result about the face of user at real time that does he/she has permission to unlock.  **Triggers:**  **-** Press “1” button on keypad in “Menu” screen.  **Preconditions:**  -N/A  **Post Conditions:**  - Success: return result that means lock can open or not.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Press “1” button on keypad | Camera is active to capture face. | | 2 | Stand in front of camera from 3 to 5 seconds for system analyze the face  [ Exception 1] | System will return result for lock system |   **Alternative Scenario:**  **-** N/A  **Exceptions:**  **-** N/A  **Relationships:**  **-** Have <<include>> relationship with “Unlock facial recognition” use case  **Business Rules:**  - Analyze face can be executed a lots of process base on face recognition algorithm. Face of user will be detected by the system then his face can be prepare with data in database to check this face is existed in database. | | | |

Table 19.”Analyze face” specification.

2.3.3.2. Generate passcode

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SUC02** | | | |
| **Use Case No.** | **SUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Generate passcode | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - System  **Summary:**  - If user use unlock by keypad, system will generate passcode which is available in one hour and send it to the phone number.  **Goal:**  - System will generate passcode to send to phone number which is entered by user.  **Triggers:**  - Press “A” button after fill in all required information in “Keypad mode” screen.  **Preconditions:**  - N/A  **Post Conditions:**  - Success: Passcode will be generated successfully and send to phone number through SMS.  - Fail: Passcode can not generate.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Generate random passcode | Passcode will be saved | | 2 | Send passcode to phone number |  |   **Alternative Scenario:**  - N/A  **Exceptions:**  **-** N/A  **Relationships:**  **-** Have <<include>> relationship with “Check passcode” use case  **Business Rules:**  - Passcode will be generated random by the system. This passcode will be saved and valid in one hour after generated time. | | | |

Table 20.“Generate passcode” specification.

2.3.3.3. Check passcode

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SUC03** | | | |
| **Use Case No.** | **SUC03** | **Use Case Version** | 2.0 |
| **Use Case Name** | Check passcode | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - System  **Summary:**  - If user use unlock by keypad, system will check passcode which user entered with passcode that system generate and send to the phone number. If the passcode is valid the lock will be opened.  **Goal:**  - User unlock successfully after entered valid passcode.  **Triggers:**  - Press “A” button to check passcode in “Enter passcode” screen.  **Preconditions:**  - User uses mode unlock by keypad.  **Post Conditions:**  - Success: User can unlock.  - Fail: User can not unlock. LCD will show warning message.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Check passcode with system | - Valid passcode: unlock  - Invalid passcode: dialog will show “Mã khóa không đúng.” message.  - Expired passcode: dialog will show “Mã khóa đã hết hạn.” message. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**  **-** N/A  **Relationships:**  **-** Have <<include>> relationship with “Unlock by keypad” use case  **Business Rules:**  - Passcode is valid for one hour after it was generated by the system. After one hour passcode will expired and user must generate again to use “Unlock by keypad” feature. | | | |

Table 21. “Check passcode” specification.

2.3.4. <GSM 900> Overview Use Case



Figure 29.<GSM 900> Overview use case

2.3.4.1. Send passcode

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – GUC01** | | | |
| **Use Case No.** | **GUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Send passcode | | |
| **Author** | Nguyen Kieu Hanh Ha | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - GSM SIM 900  **Summary:**  - GSM SIM 900 will send passcode to phone number which is entered after passcode has generated.  **Goal:**  - Passcode will be sent to phone number which is entered.  **Triggers:**  - Press “A” button on keypad in “Keypad Mode” screen after fill in required information.  **Preconditions:**  -N/A  **Post Conditions:**  - Success: return result that means lock can open or not.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Generate passcode | Save passcode | | 2 | Send passcode |  |   **Alternative Scenario:**  **-** N/A  **Exceptions:**  - N/A  **Relationships:**  **-** Have <<include>> relationship with “Generate passcode” use case  **Business Rules:**  - N/A | | | |

Table 22.”Send passcode” specification.

3. Software System Attribute

3.1. Reliability

- The system use biometric method to recognition so the ability to be entered invalid is very minor.

- The system can work precision according to user’s behavior.

3.2. Availability

- System is related security of home so the system can be active 24/7. In case, the power is blackout system will be use backup battery which can supply power up to 8 hours. When the power comes back, backup batter will be charged.

3.3. Security

- System can cover the most of cases to protect your hour out of attack of thieves.

3.4. Maintainability

- When one of component parts is broken, it is easy to fix the problem by changing a new one.

3.5. Performance

- System can recognize face in the range from 3 to 5 seconds.

3.6. Usability

- System provides user friendly GUI with guideline.

-System provides alternative case to unlock in case facial recognition is not working.

4. Conceptual Diagram

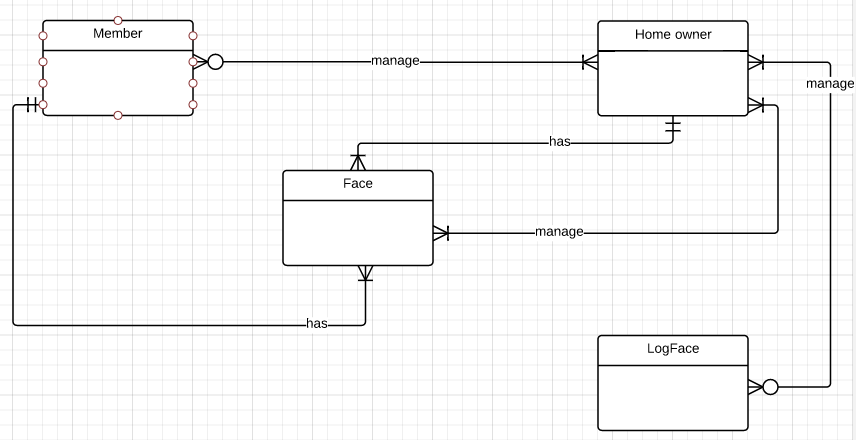


Figure 30.Conceptual diagram

|  |  |
| --- | --- |
| **Entity Data dictionary** | |
| **Entity Name** | **Description** |
| Member | - Member can is managed by one or more home owners  - Member just has one and only one face |
| Home owner | - Home owner manages zero or more members  - Home owner has one or more phone numbers  - Home owner just has one and only one face  - Home owner manages one or more faces  - Home owner manages zero or more log face. |
| Face | - Face is managed by one or more home owners  - Face just belongs one and only member.  - Face just belongs one and only home owner. |
| Log Face | - Log face is managed by one or more home owners. |

Table 23. Entity data dictionary

D. Report No.4 Software Design Specification

1. Design overview

* This document describes the technical and user interface design of SLFROR System. It includes the architectural design, the detailed design of common functions and business functions.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describe the connection and integration of the system.
* Section 4: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.
* Section 5: describe screens design.
* Section 6: describe a fully attributed ERD.
* Section 7: describe algorithms.

2. System Architectural Design

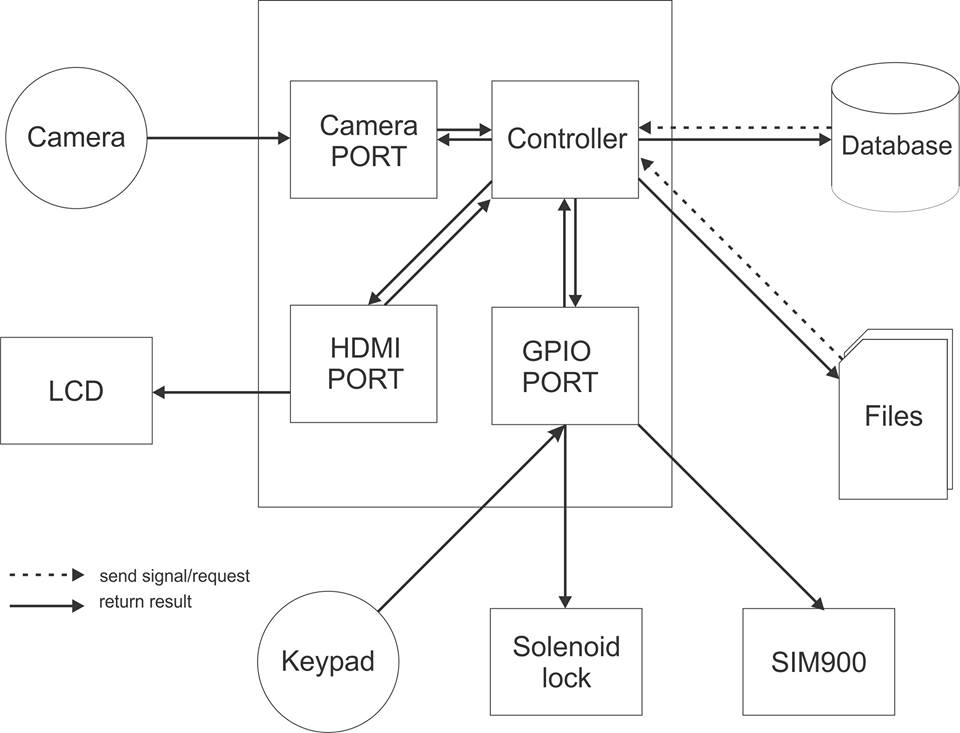


Figure 31. SLFROR system overview architecture

2.1 External Interface Requirement

2.1.1. User interface

- User interface use Graphical User Interface must be simple, clear and easy to use. We have one form to show face, user information and list of function that allow user choosing from number keypad.

2.1.2. Hardware interface

- Raspberry Pi B2 Kit with 1GB SDRAM, Quad-core, 900MHz ARM Cortex-A7 chip

- Number keypad have 0-9 number, ‘+ - / \* #’ special character.

- Logitech HD resolution Camera.

- Cabinet Door Electric Lock Assembly Solenoid DC12V 0.6A Square bevel latch

2.1.3. Software Interface

- QT 4 Development Tools Version v4.8.2 and Q make version 2.01a

3. Component diagram

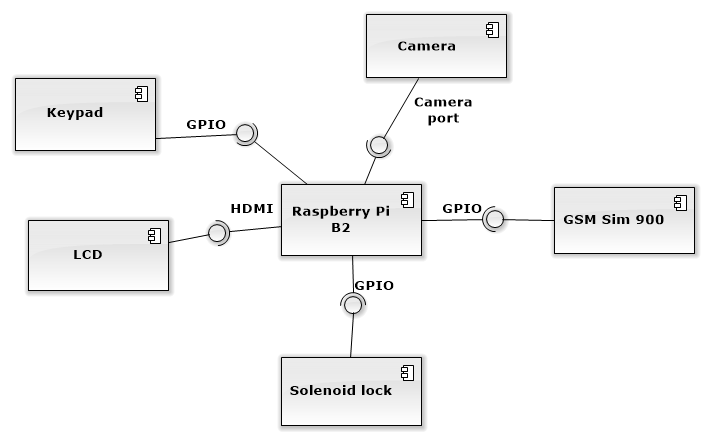


Figure 32. Component diagram

4. Detail description component

4.1. Hardware layer

4.1.1. Raspberry Pi B2



Figure 33. Raspberry Pi B2 Kit

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

|  |  |
| --- | --- |
| CPU | Quad-core, 900MHz ARM Cortex-A7 chip |
| RAM | 1GB |
| USB | 4 port |
| Ethernet port |  |
| HDMI port |  |
| Audio 3.5mm |  |
| Micro-SD |  |
| 40-GPIO pins |  |
| CSI, DSI connector |  |

Table 24. Raspberry Pi B2 specification

4.1.2. GSM Sim 900



Figure 34. GSM Sim 900

Module GSM SIM900A is be able to send SMS, call, receive call, GPRS... like a cellphone with small size. It controlled by AT command so we can easily to use, it also uses male-header connector 100mil.

|  |  |  |
| --- | --- | --- |
| VCC | 3.7V - 4.2V (so if we use 5V power, we must connect to diod  1N4007 in serial with VCC port and a 2000uF capacitor in parallel with VCC) | |
| Standby intensity | 10 mA | |
| Working intensity | 100 mA - 2A | |
| Size | 2.5 cm x 3.1 cm | |
| 7 male-header | VCC | 3.7 – 4.2 input power |
| Microphone | Input microphone |
| Headphone | Input headphone |
| TX | Send data (UART) |
| RX | Receive data (UART) |
| GND | Ground |
| PWR | Startup/Shutdonw/Reset |

Table 25. GSM Sim 900 specification

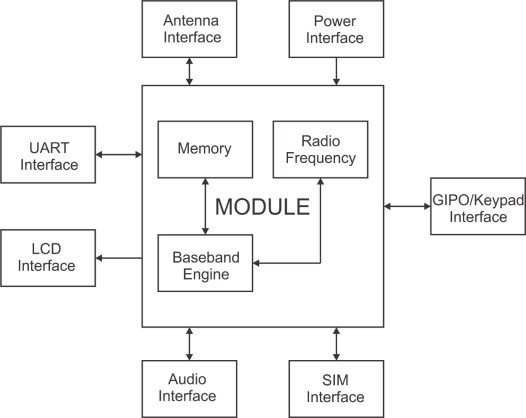


Figure 35. Function diagram of GSM 900

The following figure shows a functional diagram of the SIM900A and illustrates the mainly functional part:

- The GSM baseband engine

- Flash and SRAM

- The GSM radio frequency part

- The antenna interface

- The Other interfaces

*Connect to Raspberry:*

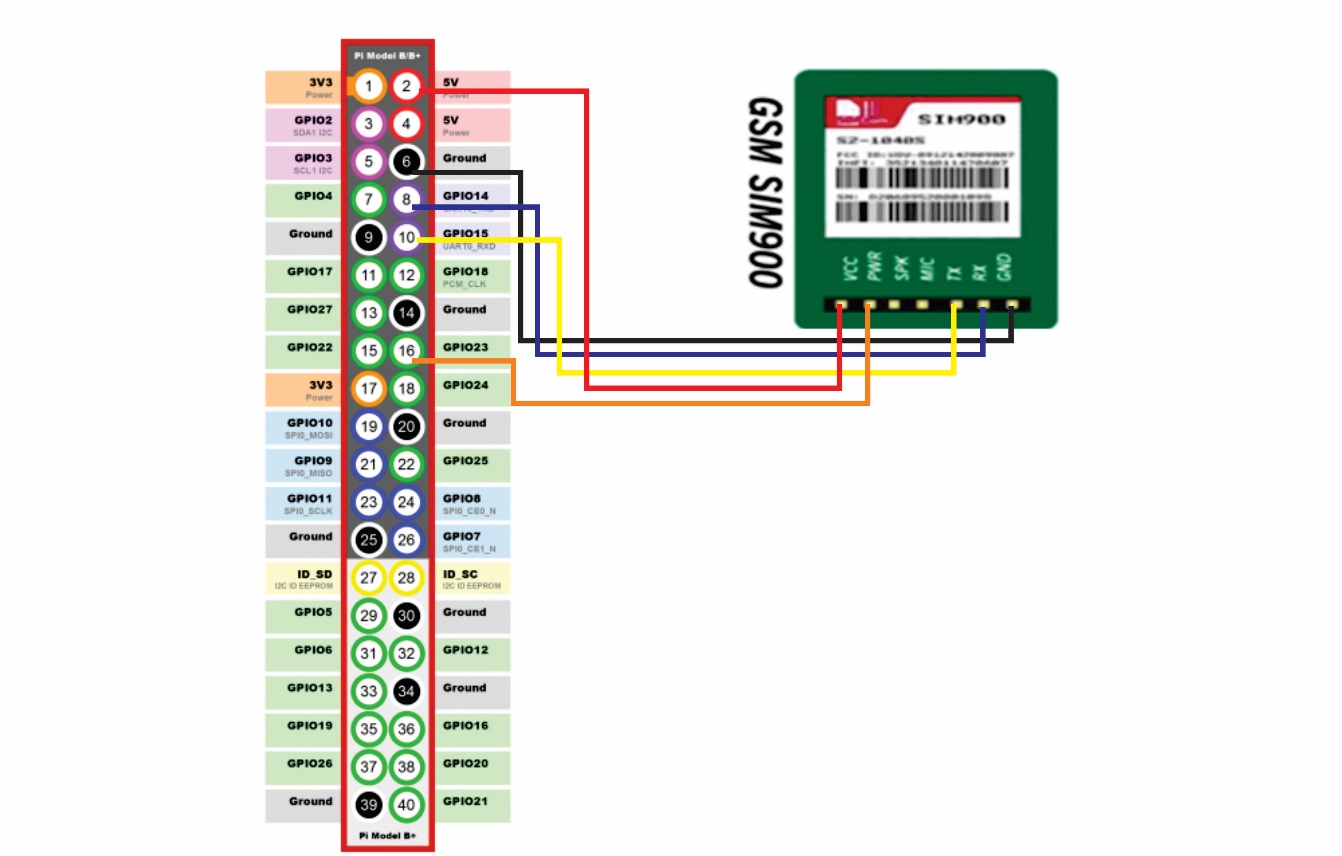


Figure 36. GSM Sim 900 Connect to Raspberry

|  |  |
| --- | --- |
| **GSM SIM900** | **RASPBERRY** |
| VVC | GPIO PIN 2 |
| PWR | GPIO PIN 16 |
| RX | GPIO PIN 8 |
| TX | GPIO PIN 10 |
| GND | GPIO PIN 6 |

Table 26. GSM Sim 900 Connect to Raspberry

4.1.3. Solenoid lock



Figure 37. Solenoid lock

Solenoids are basically electromagnets: they are made of a big coil of copper wire with an armature (a slug of metal) in the middle. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull from one end. Normally the lock is active so you can't open the door because the solenoid slug is in the way. It does not use any power in this state. When 9-12VDC is applied, the slug pulls in so it doesn't stick out anymore and the door can be opened. 12VDC (you can use 9-12 DC volts, but lower voltage results in weaker/slower operation)

|  |  |
| --- | --- |
| VDC | 9V – 12V |
| Working intensity | 5mA – 650mA |
| Designed for 1-10 seconds long activation time |  |
| Max Dimensions | 41.85mm / 1.64" x 53.57mm / 2.1" x 27.59mm / 11.08" |
| Dimensions | 23.57mm / 0.92" x 67.47mm / 2.65" x 27.59mm / 11.08" |
| Wire length | 222.25mm / 8.75" |
| Weight | 147.71g |

Table 27. Solenoid lock specification

Connect to Raspberry:

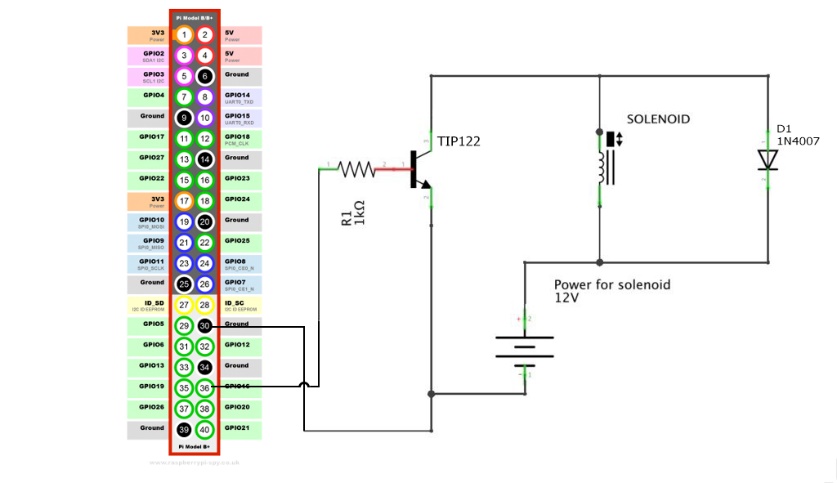


Figure 38. Solenoid lock connect to Raspberry

|  |  |
| --- | --- |
| **SOLENOID LOCK** | **RASPBERRY** |
| TIP122 – BASE | GPIO PIN 36 |
| GND | GPIO PIN 30 |

Table 28. Solenoid lock connect to Raspberry

4.2. Software layer

Raspbian is an unofficial port of Debian Wheezy armhf with compilation settings adjusted to produce optimized "hard float" code that will run on the Raspberry Pi. This provides significantly faster performance for applications that make heavy use of floating point arithmetic operations. All other applications will also gain some performance through the use of advanced instructions of the ARMv6 CPU in Raspberry Pi.

Although Raspbian is primarily the efforts of Mike Thompson (mpthompson) and Peter Green (plugwash), it has also benefited greatly from the enthusiastic support of Raspberry Pi community members who wish to get the maximum performance from their device.

5. Detail diagram

5.1. Class diagram



Figure 39. “Overview class diagram”

5.2. Class diagram explanation

|  |  |
| --- | --- |
| **Class dictionary: Describe Class** | |
| **Class name** | **Description** |
| AddFace | Describe all information and methods relative to add face screen |
| AddUser | Describe all information and methods relative to add user screen |
| ModifyUser | Describe all information and methods relative to modify user screen |
| Dialog | Describe all information and methods relative to show dialog |
| EnterPasscode | Describe all information and methods relative to enter passcode screen |
| Gsm | Describe all information and methods relative to configure GSM900 and send message |
| MainWindow | Describe all information and methods relative to System management screen |
| MyDatabase | Describe all information and methods relative to connect database |
| KeypadMode | Describe all information and methods relative to Keypad mode screen |
| ManageUser | Describe all information and methods relative to Manage User screen |
| TrainFace | Describe all information and methods relative to training face |
| MainMenu | Describe all information and methods relative to Menu screen |
| User | Describe all information and methods relative to user |
| LogFile | Describe all information and methods relative to log file |
| ThreadKeypad | Describe all information and methods relative to detect input from keypad |
| ShowHistory | Describe all information and methods relative to show history face screen |
| ProcessImage | Describe all information and methods relative to training face |
| ThreadOpenDoor | Describe all information and methods relative to thread open door |

Table 29. Class diagram explanation

5.2.1. AddFace

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Length | Int | Private | Length of list of faces. |
| Id | Int | Private | Store face id which is selected in table |
| Username | QString | Private | Store username which is selected |
| Screenname | QString | Private | Store name of screen |
| threadKeypadAddFace | ThreadKeypad | Public | Receive entered key from keypad |
| threadTrainFace | TrainFace | Public | Capture image and face detect |
| threadProcessImage | Processmage | Public | Face recorgnition and save training face data |

Table 30. Attribute of Add face class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendUserName | Void | Private | Send user name to TrainFace |
| SendResetThreadSignal | Void | Private | Reset all variable at class TrainFace and ProcessImage |
| SendSignalFinishKeypad | Void | Private | Send signal to ThreadKeypad class to finish this thread |
| SendBackActiveSignal | Void | Private | Re-active root Windows |
| capture | Void | Public | Save image path captured to database |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressUp | Void | Public | Move selected line to above line |
| pressDown | Void | Public | Move selected line to bottom line |
| pressOK | Void | Public | Accept selected user |
| doNothing | Void | Public | Show message cause user entered invalid button |
| pressBack | Void | Public | Back to last screen |
| addFace | Void | Public | Add new face to database |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart Keypad Thread |
| onTakeImage | Void | Public | Receive and show captured image to windows (real-time) |
| onLinkImage | Void | Public | Receive Links of training images |

Table 31. Method of Add face class

5.2.2. AddUser

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Length | Int | Private | Length of list of faces. |
| Id | Int | Private | Store face id which is selected in table |
| screenName | QString | Private | Store name of screen |
| threadKeypadAddFace | ThreadKeypad | Public | Receive entered key from keypad |

Table 32. Attribute of Add user class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SendSignalFinishKeypad | Void | Private | Stop Keypad record thread at Windows |
| SendBackActiveSignal | Void | Private | Re-active root Windows |
| addUser | Void | Public | Add user into database |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressUp | Void | Public | Move selected line to above line |
| pressDown | Void | Public | Move selected line to bottom line |
| pressOK | Void | Public | Execute add user process |
| doNothing | Void | Public | Show message cause user entered invalid button |
| validateUser | Bool | Public | Validate information of user |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad thread |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 33. Method of Add user class

5.2.3. ModifyUser

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| objUser | User | Private | Store all information of one user |
| Line | Int | Private | Store line which is selected by user |
| ScreenName | QString | Private | Store name of screen |
| threadKeypadModifyUser | ThreadKeypad | Public | Read keypad input throught GPIOs pin and send to window to handle |

Table 34. Attribute of Modify user class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SendSignalFinishKeypad | Void | Private | Stop keypad record thread at windows |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| updateUser | Void | Public | Update user’s information into database |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| validateUserInfo | Bool | Public | Validate user’s information |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 35. Method of Modify user class

5.2.4. Dialog

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Screen | QString | Private | Store which screen call this dialog |
| Content | QString | Private | Content of alert |
| Action | QString | Private | Kind of action |
| Id | QString | Public | Store identify of user or face |
| isConfirm | Bool |  | Flag to check this dialog is confirm or just alert |
| screenName | QString |  | Store screen name of this screen |
| threadKeypadDialog | ThreadKeypad |  | Read keypad input throught gpios pin and send to window to handle |

Table 36. Attribute of Dialog class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at windows |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| RestartThread | Void | Public | Send signal to restart keypad thread at recent window from dialog |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressOK | Void | Public | Execute process |
| setDialog | Void | Public | Set variable at dialog |
| onReadKeypad | Void | Public | Handle char entered from keypad |

Table 37. Method of Dialog class

5.2.5. EnterPasscode

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| screenName | QString | Private | Store name of screen |
| threadKeypadEnterpasscode | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |

Table 38. Attribute of EnterPasscode class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SendSignalFinishKeypad | Void | Private | Stop keypad record thread at windows |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| checkPasscode | Void | Public | Check available passcode to unlock the door |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| checkTime | Bool | Public | Check time is invalid or not |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 39. Method of EnterPasscode class

5.2.6. Gsm

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getUart\_filestream | Int | Public |  |
| setUart\_filestram | Void | Public |  |
| Init | Void | Public |  |
| sendPasscode | Void | Public |  |
| sendWarning | Void | Public |  |
| closeGsm | Void | Public |  |

Table 40. Method of Gsm class

5.2.7. MainWindow

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| screenName | QString | Private | Store name of screen |
| threadKeypadMainWindow | ThreadKeypad | Public | Read keypad input throught gpios pin and send to window to handle |

Table 41. Attribute of MainWindow class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 42. Method of MainWindow class

5.2.8. MyDatabase

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| **myDB** | QSqlDatabase | Private | Object of SQL database |

Table 43. Attribute of MyDatabase class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| connectionOpen | Bool | Public | Open connection |
| connectionClose | Void | Public | Close connection |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 44. Method of MyDatabase class

5.2.9. KeypadMode

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Line | QString | Private | Store line which is selected by user |
| screenName | QString | Private | Store name of screen |
| threadKeypadKeypadMode | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |

Table 45. Attribute of KeypadMode class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| addPhone | Void | Public | Validate phone number and generate passcode |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| validateInfo | Bool | Public | Validate information of user |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 46. Method of KeypadMode class

5.2.10. ManageUser

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Length | Int | Private | Length of list of faces. |
| ScreenName | QString | Private | Store name of screen |
| threadKeypadManageUser | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |

Table 47. Attribute of ManageUser class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| updateTable | Void | Public | Update table after modify or delete, add new data |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressUp | Void | Public | Move selected line to above line |
| pressDown | Void | Public | Move selected line to bottom line |
| doNothing | Void | Public | Show message cause user entered invalid button |
| pressBack | Void | Public | Back to last screen |
| addUser | Void | Public | Open add user screen |
| modifyUser | Void | Public | Open modify user screen |
| deleteUser | Void | Public | Delete user into database |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public |  |

Table 49. Method of ManageUser class

5.2.11. TrainFace

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| finishThread | Bool | Private | Signal to finish face record thread |

Table 50. Attribute of TrainFace class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| TakeImage | Void | Private | Send image to recent windows to show on real-time |
| SendImageUrls | Void | Private | Send Image paths to save to sqlite at recent window |
| FaceNumber | Void | Private | Send user id that record to mainmenu, mainmenu will check it and decide open the door or not |
| SendFaceRecord | Void | Private | Send face Mat and Rect to ProcessImage class |
| SendFlagToProcess | Void | Private | Send a signal to process, to start training face exist on database |
| SendUserName | Void | Private | Send user name to ProcessImage class |
| SendResetThreadSignal | Void | Private | Send signal to reset variable at ProcessImage class |
| SendBackImageRecord | Void | Private | Send Image to mainmenu, save at “face history” |
| onReceiveFlagTrainData | Void | Public | Receive signal from MainMenu to train data when start application |
| onReceiveUserName | Void | Public | Receive User Name from AddFace class |
| onReceiveTrainUrls | Void | Public | Receive Image paths from ProcessImage class |
| onReturnIdentityStatus | Void | Public | Receive status identity from ProcessImage class |
| onReceiveResetThreadSignal | Void | Public | Receive reset variable from recent window |
| onReceiveSingalFinishThread | Void | Public | Receive finish signal from recent window |
| run | Void | Public | Star thread. Record and process image from camera |

Table 51. Method of TrainFace class

5.2.12. MainMenu

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| isAdmin | Bool | Private | Flag to check role of user |
| isCapturing | Bool | Private | Function Face Recorgnition is running or not |
| User | QString | Private | Store user which is unlock successfully |
| ScreenName | QString | Private | Store name of screen |
| threadKeypadMainMenu | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |
| threadProcessImage | Processmage | Public | Processing image receive from TrainData class |
| threadIdentify | TrainFace | Public | Capture image from camera and Detect face |

Table 52. Attribute of MainMennu class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendTrainData | Void | Private | Set value of attribute |
| SendResetThreadSignal | Void | Private | Send signal to reset variable at ProcessImage and TrainFace classes |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| sendFinishCapture | Void | Private | Send signal to finish Face Capture |
| setStatusInstrucion | Void | Public | Set show/hide of instruction button |
| setContentInstructon | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| onFaceNumber | Void | Public | Receive Face User ID from TrainFace, then check them |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onTakeImage | Void | Public | Receive Image from TrainFace and show it in label of window |
| addFacialRecord | Void | Public | Save image to show at “Face History” function |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 53. Method of MainMennu class

5.2.13. User

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Int | Private | Identify id of user |
| phoneNumber | QString | Private | User’s phone number |
| Role | Bool | Private | User’s role |
| Password | QString | Private | User’s password |
| isBlock | Bool | Private | Status of user |
| isNew | Bool | Private | Flag to check user can add or remove face |

Table 54. Attribute of User class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set value of attribute |

Table 55. Method of User class

5.2.14. LogFile

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| writeFile | Void | Public | Write action into log file |

Table 56. Method of LogFile class

5.2.15. ThreadKeypad

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| closeSignal | Bool | Private | Length of list of faces. |
| bStop | Bool | Public | Store face id which is selected in table |

Table 57. Attribute of ThreadKeypad class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| ReadKeypad | Void | Private | Read GPIOs PIN to specify what key entered |
| Run | Void | Public | Start record keypad input |
| onReceiveSignalFinishKeypad | Void | Public | Finish recording |

Table 58. Method of ThreadKeypad class

5.2.16. ShowHistory

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Length | Int | Private | Length of list of faces. |
| screenName | QString | Private | Store name of this screen |
| threadShowHistory | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |

Table 59. Attribute of ShowHistory class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| setStatusInstrucion | Void | Public | Set show/hide of instruction button |
| setContentInstructon | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressUp | Void | Public | Move selected line to above line |
| pressDown | Void | Public | Move selected line to bottom line |
| pressOK | Void | Public | Show this face |
| doNothing | Void | Public | Show message cause user entered invalid button |
| pressBack | Void | Public | Back to last screen |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public | Re-active windows when receive signal from child window |

Table 60. Method of ShowHistory class

5.2.17. ProcessImage

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| finishThread | Bool | Private | Length of list of faces. |

Table 61. Attribute of ProcessImage class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| ReturnTrainUrls | Void | Private | Send Train Images path to TrainFace class after save images |
| ReturnIdentityStatus | Void | Private | Set value of attribute |
| onReceiveFaceRecord | Void | Public | Receive image from TrainFace and add to queue |
| onReceiveTrainData | Void | Public | Start train face data exist on database, stand for face recognition |
| onReceiveUserName | Void | Public | Receive user name to train new faces |
| onReceiveResetThreadSignal | Void | Public | Reset variables at this class |
| onReceiveSignalFinishThread | Void | Public | Finish this thread |
| addTrainData | Void | Public | Train face data |
| Run | Void | Public | Start this thread, process all image received from TrainFace |

Table 62. Method of ProcessImage class

5.2.18. ThreadOpenDoor

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Run | Void | Public | Set value of attribute |

Table 63. Method of ThreadOpenDoor class

5.2.19. Setting

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| ScreenName | QString | Private | Store name of screen |
| threadSetting | ThreadKeypad | Public | Read keypad input throught GPIOs PIN and send to window to handle |

Table 64. Attribute of Setting class

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SendSignalFinishKeypad | Void | Private | Stop keypad recording thread at recent window |
| SendBackActiveSignal | Void | Private | Send signal to re-active root windows |
| deleteAll | Void | Public | Delete all records in all tables and data |
| setStatusInstruction | Void | Public | Set show/hide of instruction button |
| setContentInstruction | Void | Public | Set hot key of instruction |
| setContent | Void | Public | Set content of instruction |
| pressOK | Void | Public | Update new threshold of SD card storage |
| doNothing | Void | Public | Show message cause user entered invalid button |
| pressBack | Void | Public | Back to last screen |
| onReadKeypad | Void | Public | Handle char entered from keypad |
| onRestartThread | Void | Public | Restart keypad reading thread (that stopped when open new window) |
| onReceiveActiveSignal | Void | Public |  |

Table 65. Method of Setting class

5.2.20. Ultilities

Method

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| getSize | Int | Public | Get size of SD card storage |
| checkSize | Bool | Public | Check current size of SD card storage and threshold which user is set. |
| deleteFile | Void | Public | Delete file with specific path |
| deleteFolder | Void | Public | Delete all file in folder with specific path |

Table 66. Method of Ultilities class

5.3. Flowchart diagram

5.3.1. Unlock by facial recognition

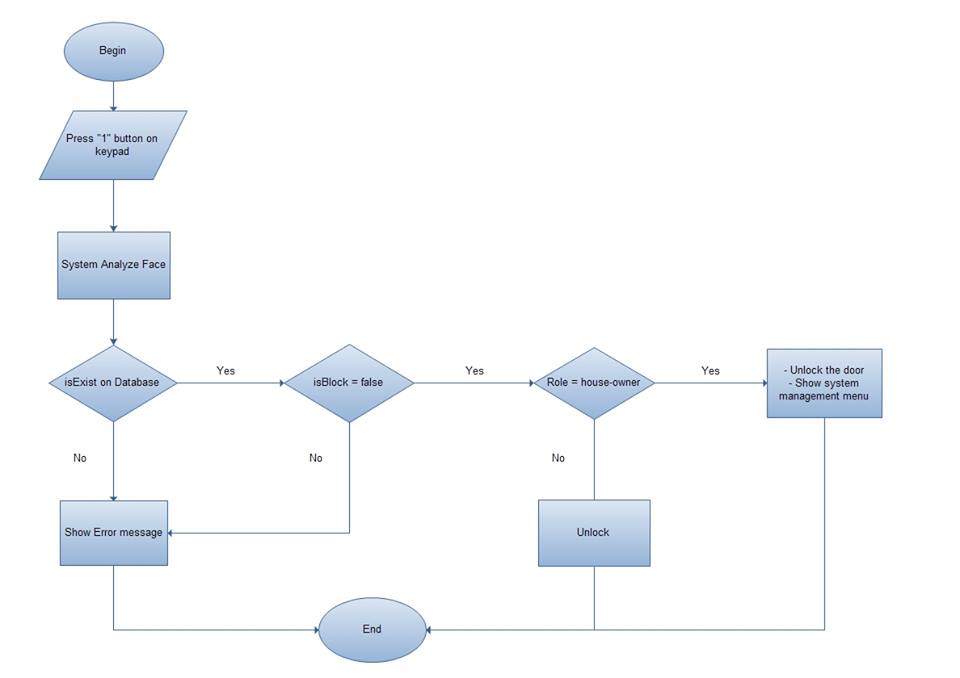


Figure 40. “Unlock by facial recognition” flow diagram

5.3.2. Unlock by keypad mode

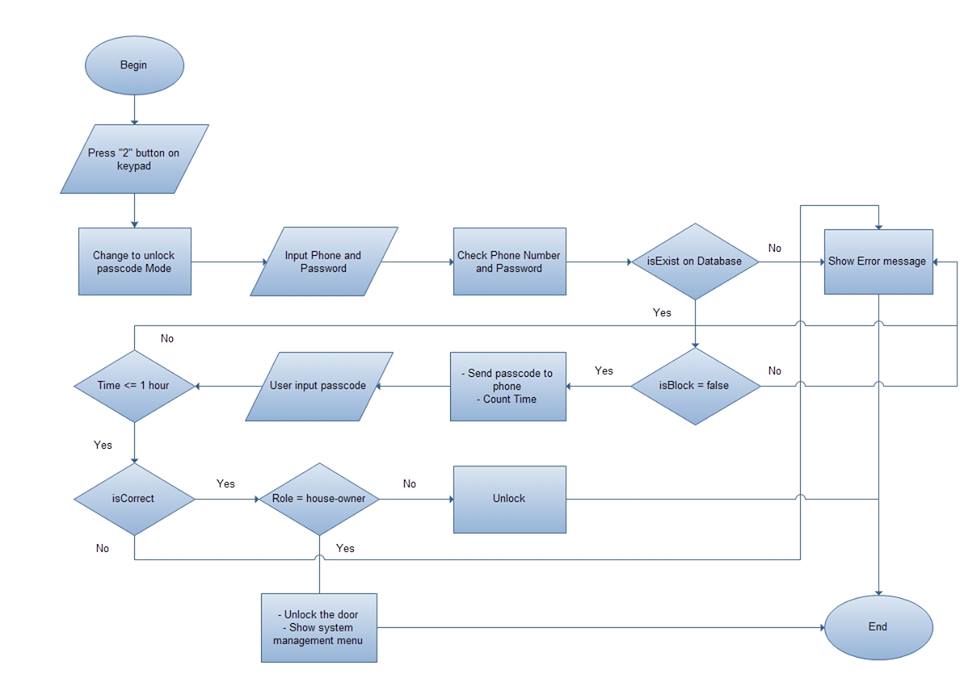


Figure 41. “Unlock by facial recognition” flow diagram

5.3.3. Analyze face

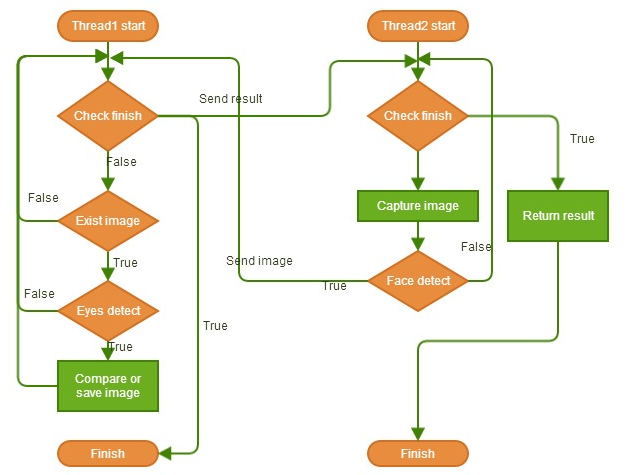


Figure 42. “Analyze face” flow chart

5.4. Sequence diagram

5.4.1. Unlock by facial recognition

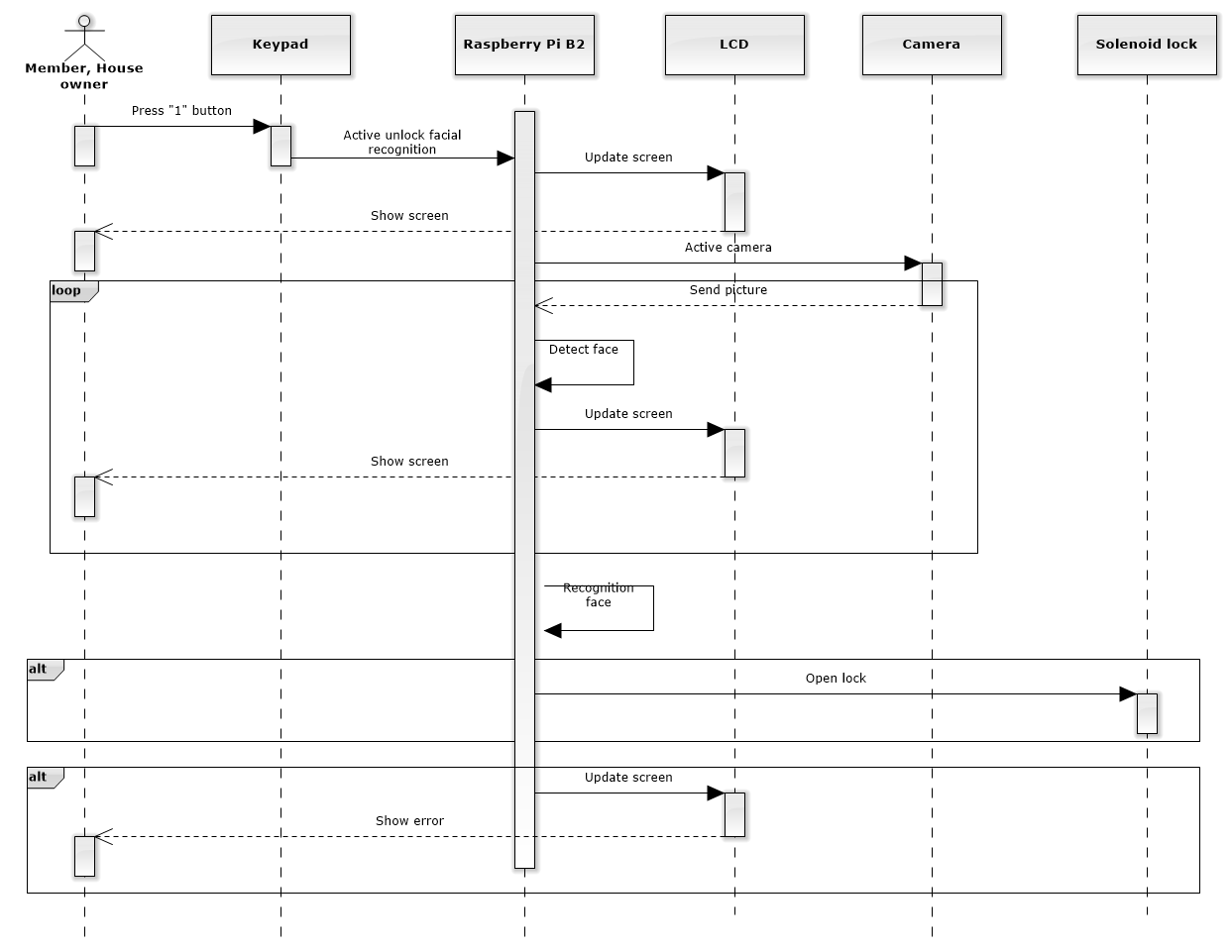


Figure 43. “Unlock by facial recognition” sequence diagram

5.4.2. Unlock by keypad

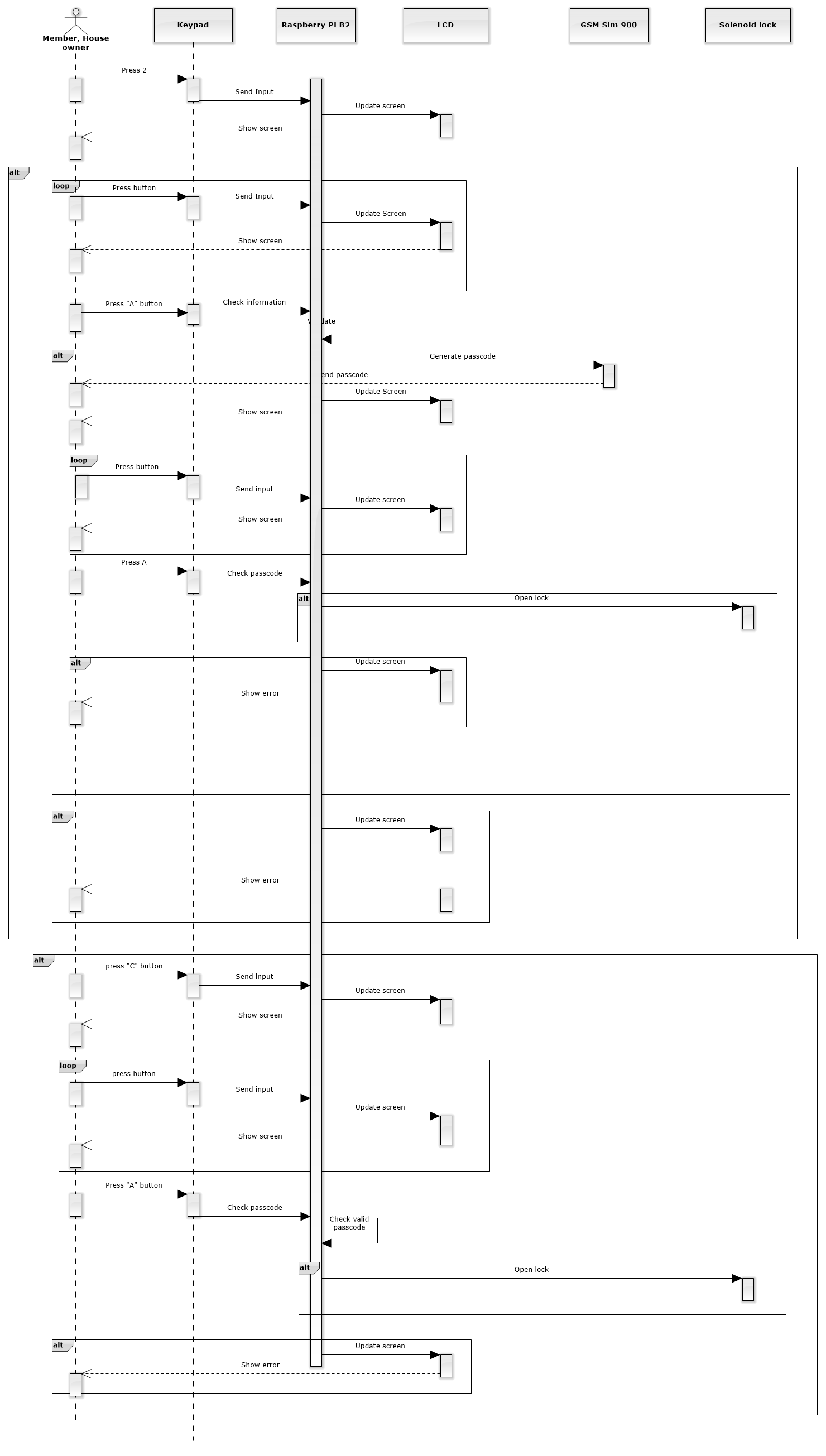


Figure 44. “Unlock by keypad mode” sequence diagram

6. User interface design

6.1. Main Menu



Figure 45. “Main menu” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Menu |  | Yes | No | Menu | N/A | N/A |
| 2 | Image |  | Yes | No | Image | N/A | N/A |
| 3 | Instruction |  | Yes | No | Label | N/A | N/A |
| 4 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 67. Field of “Main menu” screen

6.2. System management



Figure 46. “System management” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Menu |  | Yes | No | Menu | N/A | N/A |
| 2 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 68. Field of “System management” screen

6.3. User management

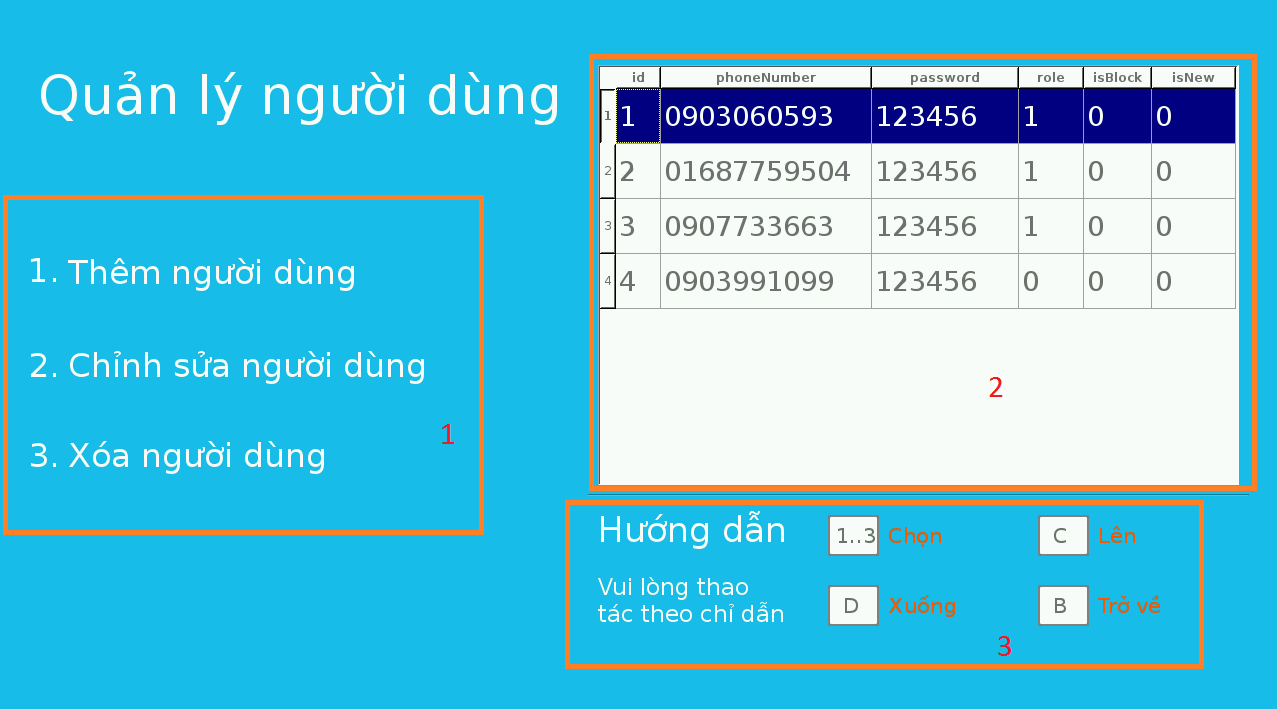


Figure 47. “User management” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Menu |  | Yes | No | Menu | N/A | N/A |
| 2 | Table |  | Yes | No | Table | N/A | N/A |
| 3 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 69. Field of “User management” screen

6.4 Add user

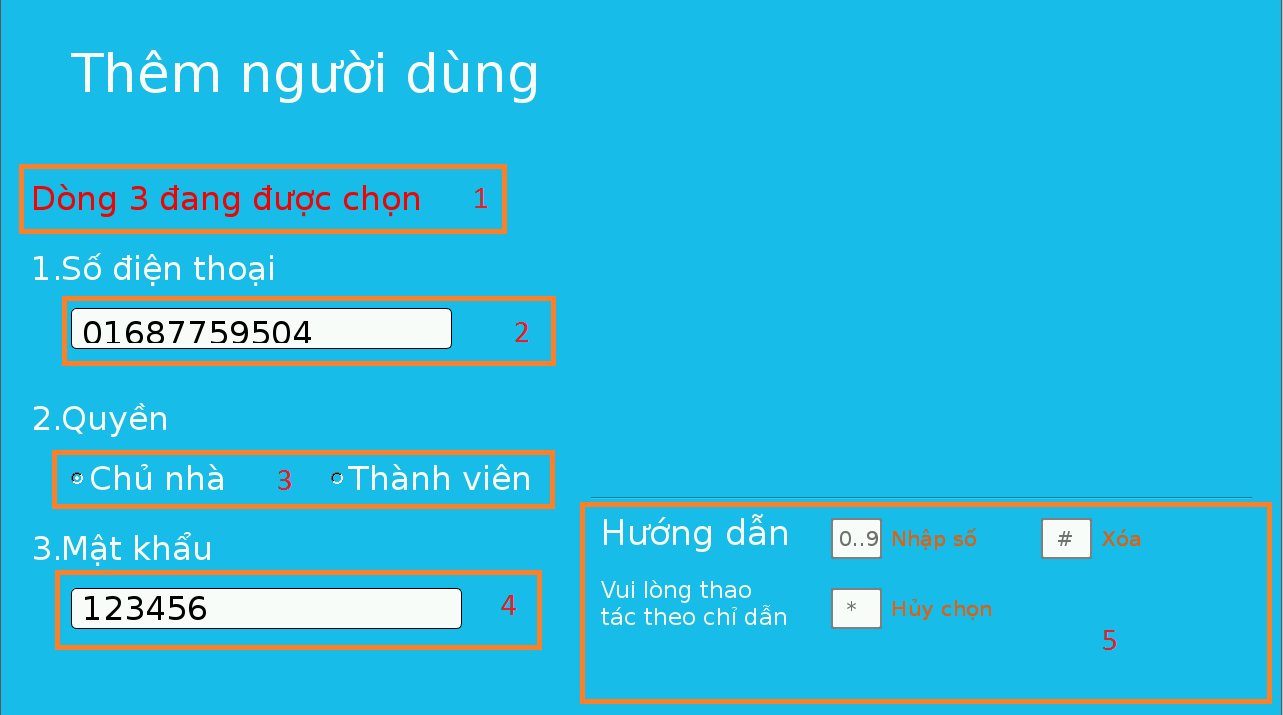


Figure 48. “Add user” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Status |  | Yes | No | Label | N/A | N/A |
| 2 | Phone Number |  | No | Yes | Text Field | QString | N/A |
| 3 | Role |  | No | Yes | Radio button | N/A | N/A |
| 4 | Password |  | No | Yes | Text Field | QString | N/A |
| 5 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 70. Field of “Add User” screen

6.5. Modify User



Figure 49. “Modify user” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Status |  | Yes | No | Label | N/A | N/A |
| 2 | Phone Number |  | No | Yes | Text Field | QString | N/A |
| 3 | Role |  | No | Yes | Radio button | N/A | N/A |
| 4 | Password |  | No | Yes | Text Field | QString | N/A |
| 5 | Status |  | No | No | Checkbox | N/A | N/A |
| 5 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 71. Field of “Modify user” screen

6.6. Manage face

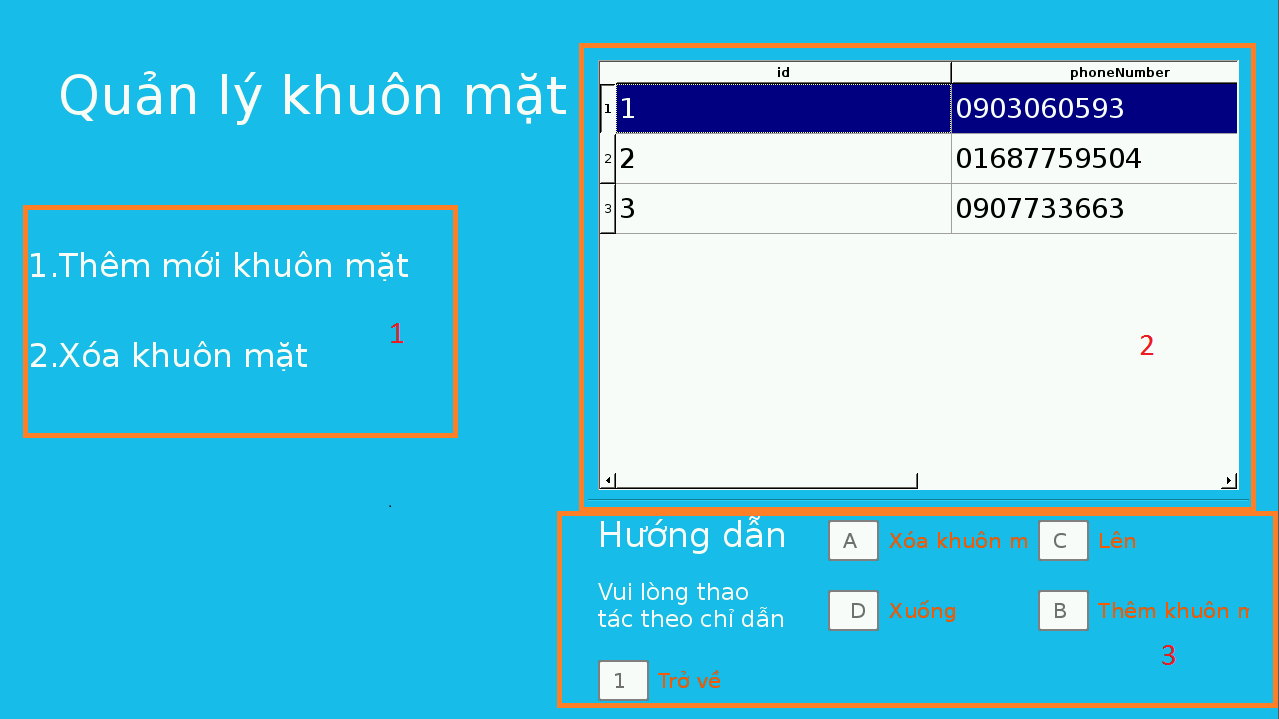


Figure 50. “Face management” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Menu |  | Yes | No | Label | N/A | N/A |
| 2 | Table |  | Yes | No | Table | N/A | N/A |
| 5 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 72. Field of “Face management” screen

6.7. Add face



Figure 51. “Add face” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Table |  | Yes | Yes | Table | N/A | N/A |
| 2 | Phone number |  | No | Yes | Text Field | QString | N/A |
| 3 | Menu |  | Yes | No | Menu | N/A | N/A |
| 4 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 73. Field of “Add face” screen

6.8. Show log face

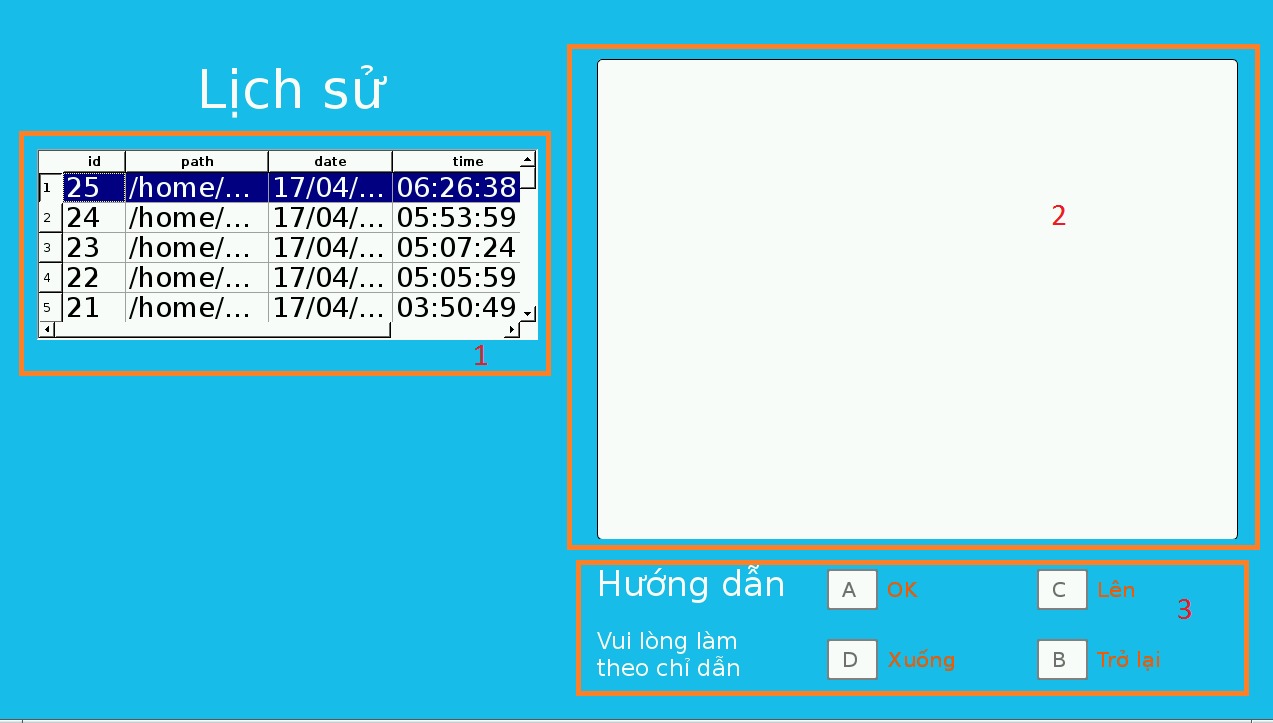


Figure 52. “Show log face” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Show Log Face Table |  | Yes | No | Table | N/A | N/A |
| 2 | Image |  | No | Yes | QImage | N/A | N/A |
| 3 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 74. Field of “Show log face” screen

6.9. Keypad mode



Figure 53. “Keypad mode” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Status |  | Yes | No | Label | N/A | N/A |
| 2 | Phone Number |  | No | Yes | Text Field | QString | N/A |
| 3 | Password |  | No | Yes | Text Field | QString | N/A |
| 4 | Instruction |  | Yes | No | Label | N/A | N/A |
| 5 | Alternative case |  | Yes | No | Label | N/A | N/A |

Table 75. Field of “Keypad mode” screen

6.10. Enter passcode

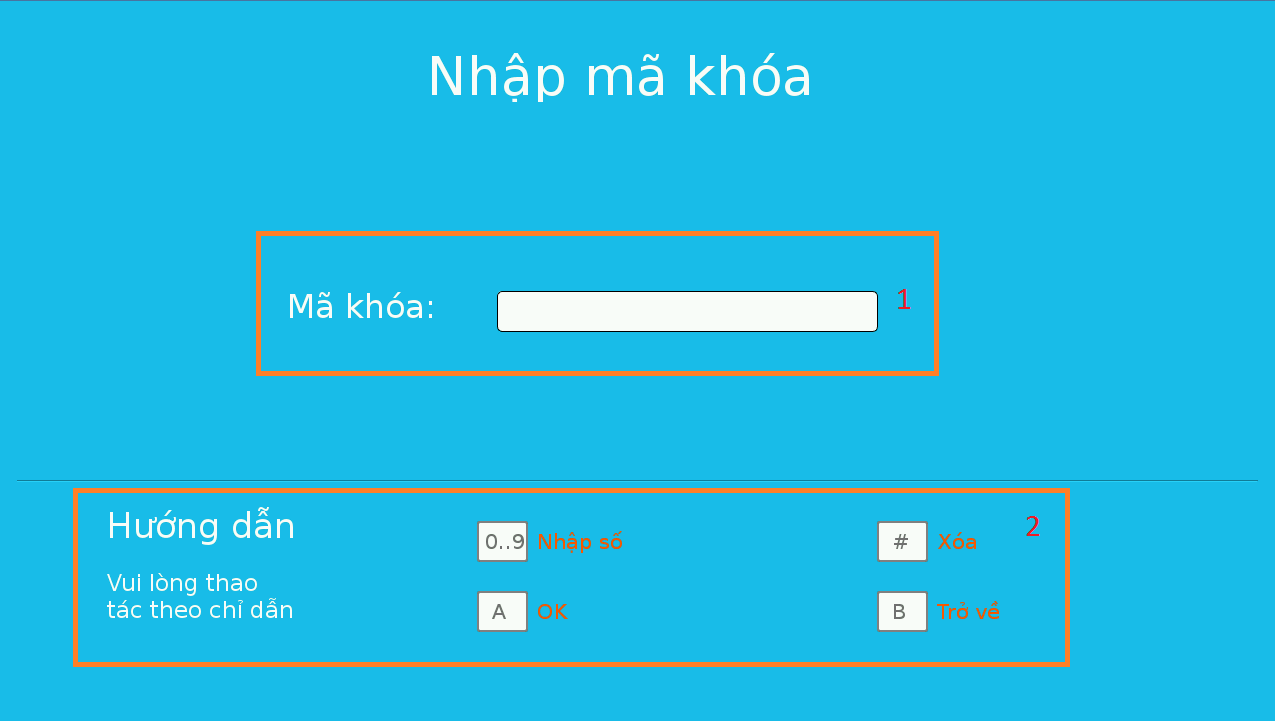


Figure 54. “Enter passcode” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Passcode |  | No | Yes | Text Field | N/A | N/A |
| 2 | Instruction |  | Yes | No | Label | N/A | N/A |

Table 76. Field of “Enter passcode” screen

6.11. Setting

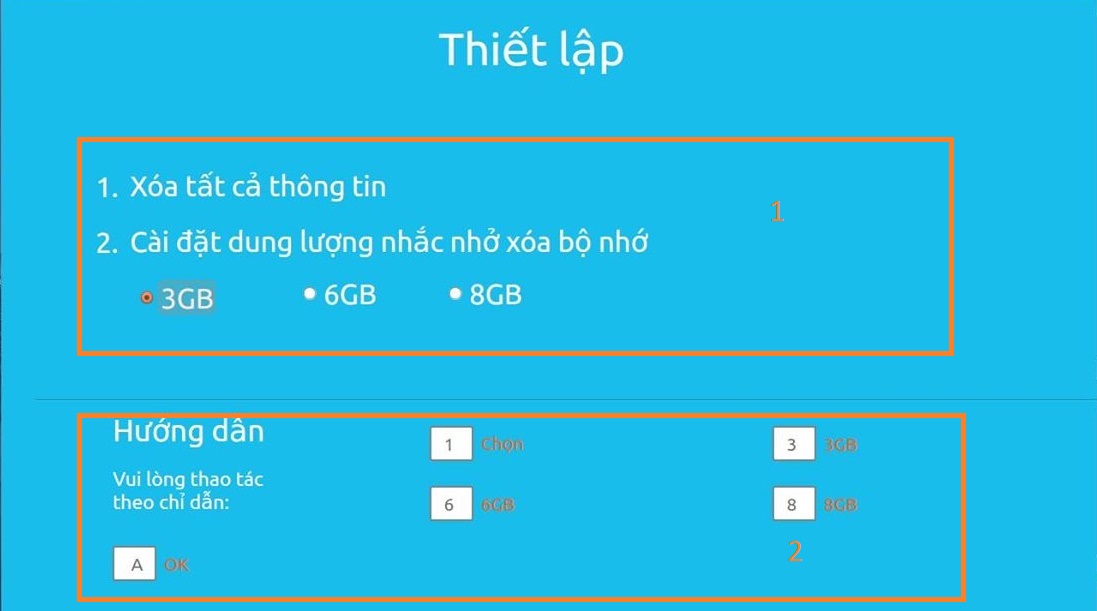


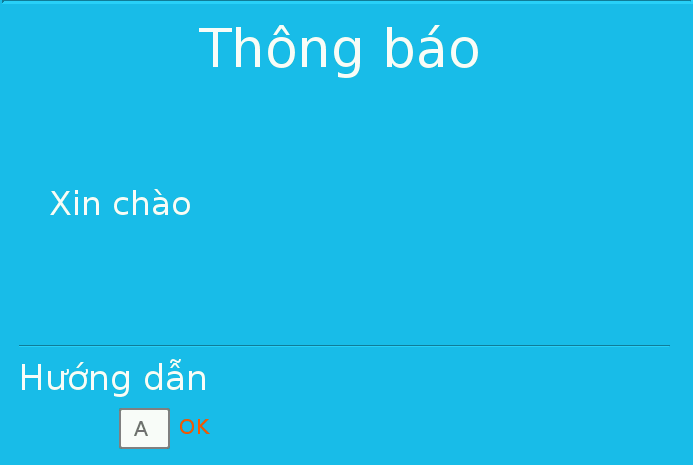
Figure 55. “Setting” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Menu |  | Yes | Yes | Label | N/A | N/A |
| 2 | Instruction |  | Yes | Yes | Label | N/A | N/A |

Table 77. Field of “Setting” screen

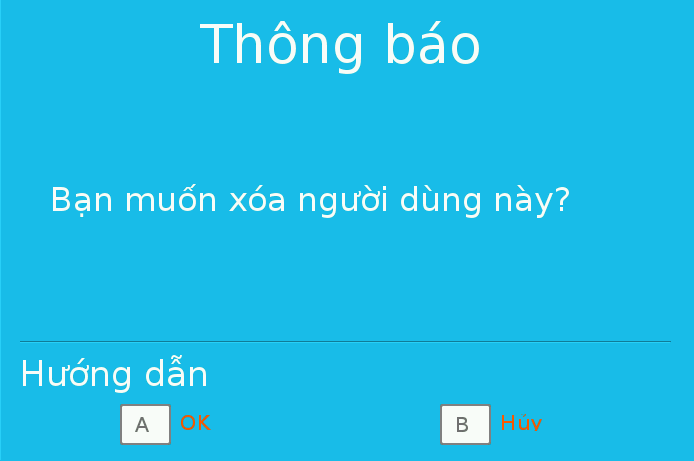
6.12. Dialog



3

1

Figure 56. “Alert” screen



3

2

1

1

Figure 57. “Confirm” screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Title |  | Yes | Yes | Label | N/A | N/A |
| 2 | Content |  | Yes | Yes | Label | N/A | N/A |
| 3 | Instruction |  | Yes | Yes | Label | N/A | N/A |

Table 78. Field of “Dialog” screen

7. Database design

7.1. Logical diagrams



Figure 58. Logical diagram

7.2. Data dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Member | Describe all members profiles in the system |
| Home owner | Describe all home owners profiles in the system |
| Face | Describe all faces in the system. |
| Log Face | Describe all face is logged in the system. |

Table 79. Data dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Member | id {PK} | Unique identifier of member, auto increment. | Integer | No |
| phoneNumber | Member’s phone number | Text | No |
| password | Member’s password | Integer | No |
| role | Member’s role | Boolean | No |
| isBlock | Flag for block user | Boolean | No |
| isDelete | Flag for delete user | Boolean | No |
| isNew | Flag for add face user | Boolean | No |
| Home owner | id {PK} | Unique identifier of member, auto increment. | Integer | No |
| phoneNumber | Member’s phone number | Text | No |
| password | Member’s password | Integer | No |
| role | Member’s role | Boolean | No |
| isBlock | Flag for block user | Boolean | No |
| isDelete | Flag for delete user | Boolean | No |
| isNew | Flag for add face user | Boolean | No |
| Face | id {PK} | Unique identifier of face, auto increment. | Integer | No |
| Face\_image | Path of user’s images. | Text | No |
| userID {FK} | Identifier of user. Foreign key | Integer | No |
| isDelete | Flag for delete user | Boolean | No |
| Log Face | id{PK} | Unique identifier of Log face. | Integer | No |
| path | Path of log face image | Text | No |
| date | Date save user’s face | Text | No |
| time | Time save user’s face | Text | No |

Table 73. Entity dictionaries detail

8. Algorithm

OpenCV (Open Source Computer Vision) is a library started by Intel in 1999. It focus on realtime image processing. OpenCV have FaceRecognizer class for face recognition. At version 2.4.10, we have 3 algorithms:

* Eigenfaces
* Fisherfaces
* Local Binary Pattents Histogram (LBPH)

And we choose LBPH algorithm to use on this project, because it is less dependent on brighness of image.

First, OpenCV will record a video from camera, then capture 1 frame using “read” function. read(Mat &image) return frame matrix form.

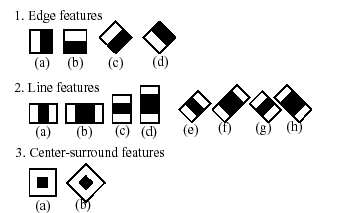
Convert matrix image to gray image, using **cvtColor** function, from RGB matrix to gray matrix:



→

Next, we must detect faces on captured frame base haar-like features, provided by OpenCV: **detectMultiScare** function of CascadeClassifier class.

Haar-like features sample:



Example:

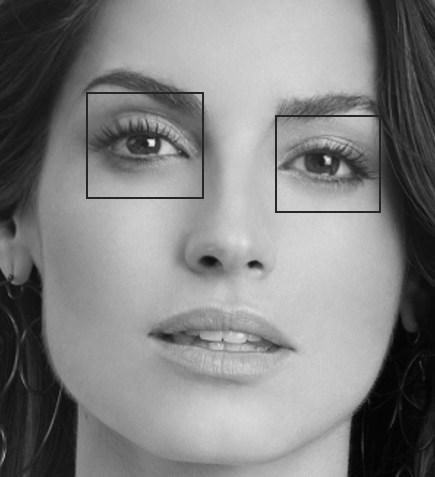




detectMultiScale return a vector of Rectanges (with 4 numbers: x, y, width, height) that specified position of face. And then, cut image limited by Rect:



detectMultiscale again, base haar-like too, but use other haar-like sample (other cascade file loaded) to detect eyes:



Continue, rotate image matrix if face is not balance.

Calculation angle to rotate:

**angle = atan2(y, x) \* 180 / CV\_PI**

**getRotationMatrix2D** function calculates an affine matrix of 2D rotation

The function calculates the following matrix:



Where,

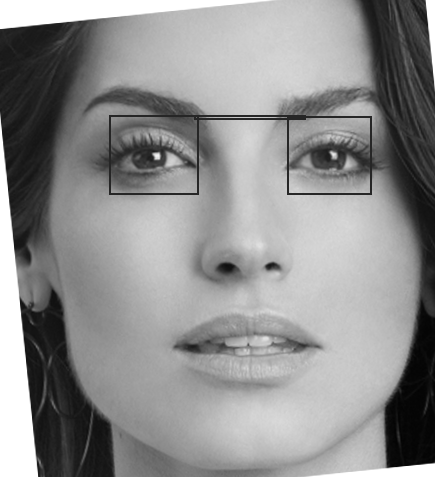


Applies an affine transformation to an image, use **warpAffine** function

The function warpAffine transforms the source image using the specified matrix:



Result:



After rotate matrix image, detect eyes one more time to ensure about correctly.

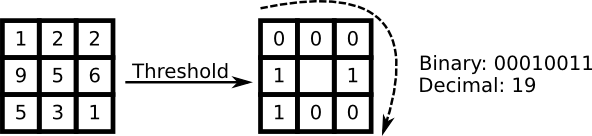
Then cut small image. The core image use to train faces and recognition:



Local Binary Pattents Histogram (LBPH):

We cannot sure all image take with the same light. 1 at night, 1 at mid-day, or afternoon, morning, .etc. of 1 person. But LBPH can handle that.

The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You’ll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you’ll end up with 2^8 possible combinations, called Local Binary Patterns or sometimes referred to as LBP codes. The first LBP operator described in literature actually used a fixed 3 x 3 neighborhood just like this:



**Algorithmic Description:**

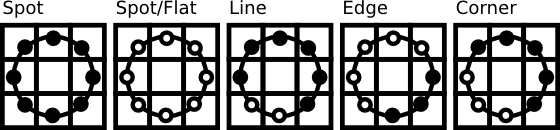
A more formal description of the LBP operator can be given as:



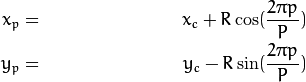
, with (x\_c, y\_c) as central pixel with intensity i\_c; and i\_n being the intensity of the the neighbor pixel. s is the sign function defined as:



This description enables you to capture very fine grained details in images. In fact the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. The idea is to align an abritrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:



For a given Point (x\_c,y\_c) the position of the neighbor (x\_p,y\_p), p \in P can be calculated by:

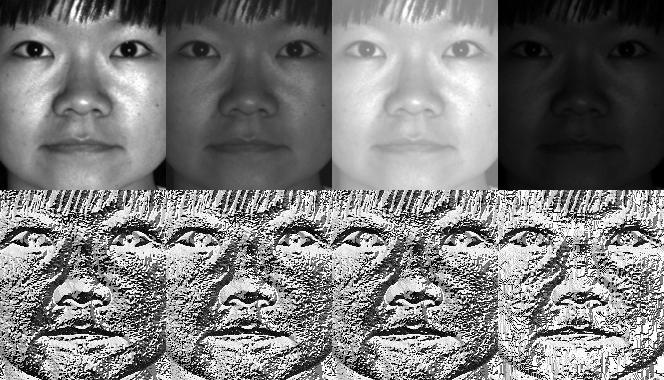


Where R is the radius of the circle and P is the number of sample points.

The operator is an extension to the original LBP codes, so it’s sometimes called Extended LBP (also referred to as Circular LBP). If a point’s coordinate on the circle doesn’t correspond to image coordinates, the point get’s interpolated. Computer science has a bunch of clever interpolation schemes; the OpenCV implementation does a bilinear interpolation:



By definition the LBP operator is robust against monotonic gray scale transformations. We can easily verify this by looking at the LBP image of an artificially modified image:



E. Report No.5 Implement & Testing

1. Introduction

1.1. System overview

- This project is about to build a Smart lock bases on Raspberry kit which has the ability to detect user's facial to unlock or input the keypad a unlock passcode which has been sent from system by SMS.

- This is an independence system run on Raspberry board and it will to control all system's activities like: capture facial, send SMS message, and control the solenoid lock.

1.2. Test approach

**- White box test**: Developers self-test on code in which function they developed (code inspector, unit test).

**- Black box test**: Test all functions independently to make sure they work correctly. Then, we execute integration test to test the system's function when integrated. Finally, we do the system test to retest all system.

2. Database relationship diagram

* 1. Physical Diagram

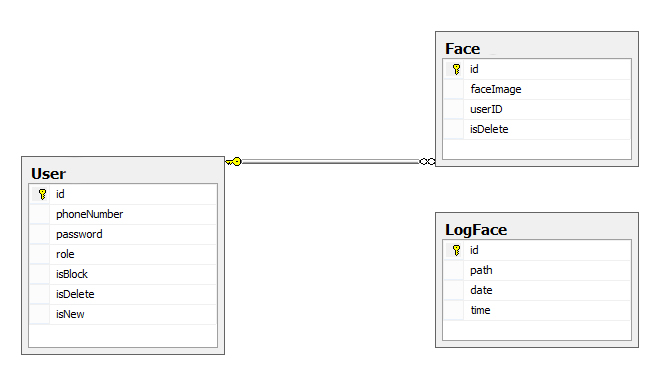


Figure 59. Physical diagram

2.2. Data dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| User | Describe all users profiles in the system |
| Face | Describe all faces in the system. |
| Log Face | Describe all face is logged in the system. |

Table 80. Data dictionary of physical diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| User | id {PK} | Unique identifier of member, auto increment. | Integer | No |
| phoneNumber | Member’s phone number | Text | No |
| password | Member’s password | Integer | No |
| role | Member’s role | Boolean | No |
| isBlock | Flag for block user | Boolean | No |
| isDelete | Flag for delete user | Boolean | No |
| isNew | Flag for add face user | Boolean | No |
| Face | id {PK} | Unique identifier of face, auto increment. | Integer | No |
| Face\_image | Path of user’s images. | Text | No |
| userID {FK} | Identifier of user. Foreign key | Integer | No |
| isDelete | Flag for delete user | Boolean | No |
| Log Face | id{PK} | Unique identifier of Log face. | Integer | No |
| path | Path of log face image | Text | No |
| date | Date save user’s face | Text | No |
| time | Time save user’s face | Text | No |

Table 81. Data dictionaries detail of physical diagram

3. Performance Measures

3.1. Clustering Performance

Clustering is performed by running K Mean Algorithm which has complexity of : O(n \* k \* I \* d)

* n : number of points
* k : number of cluster
* I : number of iteration
* d : number of attributes (3)

Clustering take almost the time of process that we can ignore the time needed to load data from database, digitalize data.

The speed of clustering will vary and increase dramatically when n increase. The purpose of this project is not about optimizing K-Mean Algorithm so it is accepted to let the process run till it completes. Moreover, the clustering is designed to run by staff, wait time is acceptable.

4. Test plan

We have two main test phases: Function test (Integration test) and System test.

For Integration test, we use the functional test. This kind of test shows us the working of each function right or wrong. We focus on the result of the function in different cases.

The System test check all the activities the Smart lock after setup such as: what the system do when it is turned on and if user do somethings, what will response. We create test case to check each function’s activity. Then, we observe and record the result.

4.1. Features to be tested

- Functional Testing:

+ Integration testing: Test all function on Raspberry Application and each external module

+ System testing: Test all function for Raspberry Application – external modules

- UI Testing: Test UI on Raspberry Application

4.2. Features not to be tested

- Hardware module: Raspberry Pi B2.

4.3. Testing tools and environment

- Testing tools:

+ A raspberry board with connect to PC/Laptop running Linux OS

+ QT Creattor

+ Minicom on Raspberry

- Environment:

+ Some where with good light condition

4.4. Test past/ fail criteria

For system testing, the criteria are:

* 90% of the test cases must pass.
* 100% of test cases about hardware module must pass.
* All test cases dealing with critical functionality must pass.
* All medium and high severity defects must be fixed.
* Test coverage must be at least 90%.

5. System Testing Test Case

5.1. Test on real device

5.1.1. Component testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **TEST CASE DESCRIPTION** | **PRE-CONDITION** | **TEST CASE PROCEDURE** | **EXPECTED OUTPUT** | **Result** | **Note** |
| CPN01 | Send message to GSM SIM900 | \_ GSM SIM900 have sim card \_ GSM SIM 900 is turned on \_ Raspberry is turn on GSM SIM900 connect to Raspberry through UART port | \_Turn on terminal on Raspberry \_Using command "minicom -b 115200 -o -D /dev/ttyAMA0" \_type in "AT+CPIN?" then press ENTER | \_Recognize devce \_GSM SIM900 respone  "READY" in terminal | Pass |  |
| CPN02 | Send sms message using GSM SIM900 | \_ GSM SIM900 have sim card \_ Simcard is available to send sms \_ GSM SIM 900 is turned on \_ Raspberry is turn on GSM SIM900 connect to Raspberry through UART port | \_Turn on terminal on Raspberry \_Use command "minicom -b 115200 -o -D /dev/ttyAMA0" to open minicom \_type in "AT+CMGS="+84945246345"" then press ENTER \_type in "test sms" then press Ctrl+Z | \_GSM SIM900 respone  "OK" in terminal \_Recieved SMS "test sms" on mobile | Pass |  |
| CPN03 | Change Solenoid Lock's status to unlock | \_Solenoid lock connect to 12V power \_Solenoid lock connect to Raspberry using GPIO port \_In raspberry have test.c file which have testing unlock code - Raspberry have bcm2835 library | \_Turn on terminal on Raspberry \_type in"sudo gcc test test.c -lbcm2835" then ENTER \_ type in "sudo ./test.c" then ENTER | Solenoid lock changes status to unlock, after 5s, Its status changes back to lock | Pass |  |
| CPN04 | Receive picture from camera | \_Camera connect to Raspberry using Camera port. - Raspberry is alrealy installed camera driver - Connect to LCD | \_ Open terminal on raspberry \_ Type command "raspistill -o image.jpg" then press enter | \_ LCD show a camera windows and image which is recorded by camera | Pass |  |
| CPN05 | Receive signal from keypad | \_Keypad connect to Raspberry using GPIO port - Raspberry have bcm2835 library | \_Turn on terminal on Raspberry \_type in"sudo gcc test test.c -lbcm2835" then ENTER \_ type in "sudo ./testkepad.c" then ENTER - Press any button on keypad | Raspberry receive signal and show pressed key on terminal | Pass |  |
| CPN06 | Show screen on LCD | \_LCD connect to Raspberry using HDMI port | \_Turn on LCD  \_ Do something on Raspberry | See your activities on LCD screen. | Pass |  |

5.1.2. Integration test

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **TEST CASE DESCRIPTION** | **PRE-CONDITION** | **TEST CASE PROCEDURE** | **EXPECTED OUTPUT** | **Result** | **Note** |
| SLFROR01 | Open lock by keypad - correct phone - correct password - correct passcode  - enter passcode <= 1h | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = "0903060593" | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad then press A | \_ Password = "123456" the System change to input passcode line | Pass |  |
| \_ Wait a moment until when received Passcode SMS from system | \_System sent passcode SMS to "0903060593" and change into enter passcode screen | Pass |  |
| \_ Enter correct passcode using keypad | \_Sysem compare entered passcode successfully | Pass |  |
| \_ Press "A" button on keypad | \_Lock is opened | Pass |  |
| SLFROR02 | Open lock by keypad - incorrect phone - correct password - correct passcode | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter " 0945246345" using keypad | \_ Phone number = "0945246345" | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad then press A | \_ Password = "123456" then system show error message “Số điện thoại chưa đăng kí” | Pass |  |
| SLFROR03 | Open lock by keypad - correct phone - incorrect password - correct passcode | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = 0903060593 | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “111111” from keypad then press A | \_ System show error message “Password sai | Pass |  |
| SLFROR04 | Open lock by keypad - correct phone - correct password - incorrect passcode | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = "0903060593" | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad then press A | \_ Password = "123456" the System change to input passcode line | Pass |  |
| \_ Wait a moment until when received Passcode SMS from system | \_System sent passcode SMS to "0903060593" and change into enter passcode screen | Pass |  |
| \_ Enter incorrect passcode | \_Sysem compare entered passcode unsuccessfully and show error message “Passcode Sai | Pass |  |
| \_ Press "A" button on keypad | \_Lock is not opened | Pass |  |
| SLFROR06 | Open lock by keypad - correct phone - correct password - correct passcode - enter passcode > 1h | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = "0903060593" | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad then press A | \_ Password = "123456" the System change to input passcode line | Pass |  |
| \_ Wait a moment until when received Passcode SMS from system | \_System sent passcode SMS to "0903060593" and change into enter passcode screen | Pass |  |
| \_ Wait for over 1h then enter correct passcode using keypad | \_Sysem compare entered passcode unsuccessfully and show error message “Passcode hết hạn” | Pass |  |
| \_ Press "A" button on keypad | \_Lock is not opened | Pass |  |
| SLFROR07 | Open lock by keypad - correct phone - correct password - passcode not input | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = "0903060593" | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad then press A | \_ Password = "123456" the System change to input passcode line | Pass |  |
| \_ Press "A" button on keypad | \_Lock is not opened | Pass |  |
| SLFROR08 | Open lock by keypad - nothing input | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “A” button on keypad | \_ System show error message “Vui lòng điền thông tin” | Pass |  |
| SLFROR09 | Open lock by keypad - phone not input \_ corect password | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “\*” button on keypad | \_ Menu screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “2” button on keypad | \_ Select input password line | Pass |  |
| \_ Enter “123456” from keypad | \_ Password = "123456" the System change to input passcode line | Pass |  |
| \_ Press “A” button on keypad | \_ System show error message “Vui lòng điền thông tin” | Pass |  |
| SLFROR10 | Open lock by keypad - correct phone - password not input | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “2” button on keypad | \_ Menu keypad unlock screen appeared on LCD | Pass |  |
| \_ Press “1” button on keypad | \_ Change to unlock using keypad screen | Pass |  |
| \_ Press “2” button on keypad | \_ Change to enter phone number screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select input phone line | Pass |  |
| \_ Enter “0903060593” using keypad | \_ Phone number = "0903060593" | Pass |  |
| \_ Press “A” button on keypad | \_ System show error message “Vui lòng điền thông tin” | Pass |  |
| SLFROR11 | Open lock by facial - home member face | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “1” button on keypad | \_ Menu Facial unlock screen appeared on LCD | Pass |  |
| \_ User stand in front of the camera | \_ System compare user face with database successfully | Pass |  |
|  | \_ System show error message “Vui lòng điền thông tin” | Pass |  |
| SLFROR12 | Open lock by facial - Guess face | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “1” button on keypad | \_ Menu Facial unlock screen appeared on LCD | Pass |  |
| \_ Guess stand in front of the camera | \_ System compare user face with database unsuccessfully - system show error message “Not regconize” | Pass |  |
|  | \_ Lock is not opened | Pass |  |
| SLFROR13 | Open lock by facial - Using facial photo of home member | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port | \_ Press “1” button on keypad | \_ Menu Facial unlock screen appeared on LCD | Pass |  |
| \_Hold the home-member's facial photo in front of the camera | \_ System compare user face with database unsuccessfully - system show error message “Đây không phải là một khuôn mặt thật” | Fail |  |
|  | \_ Lock is not opened | Fail |  |
| SLFROR14 | User management | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “1” button on keypad | \_ Change to user management screen with user table in right side | Pass |  |
| SLFROR15 | User management - add phone | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “1” button on keypad | \_ Change to user management screen | Pass |  |
| \_ Press “1” button on keypad | \_ Change to add user screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select phone line | Pass |  |
| \_ Enter phone = “0945246345” using keypad | \_ Phone = 0945246345 | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select role line | Pass |  |
| \_ Choosing “Home owner” role using “1” or “2” button | \_ “Home owner” selected | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “3” button on keypad | \_ Select Password line | Pass |  |
| \_ Enter user's password = “123456” using number on keypad | \_ Password = “123456” | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “A” button on keypad | \_ User “0945246345” added,back to user management screen and new phone will be added with column isNew = 1 | Pass |  |
| SLFROR15 | User management - modify user | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “1” button on keypad | \_ Change to user management screen with user table in right side | Pass |  |
| \_ Select “dthk” user to modify using “D” and “C” button | \_ "dthk" in selected | Pass |  |
| \_ Press “2” button on keypad | \_ Change to “dthk” 's user modify screen | Pass |  |
| \_ Press “1” button on keypad | \_ Select phone line | Pass |  |
| \_ Enter phone = “0908070600” using keypad | \_ Phone = 0908070600 | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “2” button on keypad | \_ Select role line | Pass |  |
| \_ Choosing "Memer” role using “1” or “2” button | \_ “Member” selected | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “3” button on keypad | \_ Select Password line | Pass |  |
| \_ Enter user's password = “111111” using number on keypad | \_ Password = “111111” | Pass |  |
| \_ Press “\*” button on keypad | \_ Back to none select line | Pass |  |
| \_ Press “4” button on keypad | \_ Select block line | Pass |  |
| \_ Choosing “block” role using “1” or “2” button | \_ “Block” selected | Pass |  |
| \_ Press “A” button on keypad | \_ User modified successfully, system will back to user management screen and user's info will be changed | Pass |  |
| SLFROR16 | User management - remove user | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “1” button on keypad | \_ Change to user management screen with user table in right side | Pass |  |
| \_ Select "0908070600" line using “D” and “C” button | \_ "0908070600" in selected | Pass |  |
| \_ Press “3” button on keypad | \_ Confirm dialog will be shown | Pass |  |
| \_ Press “A” button on keypad | \_ User removed sucessfully, system will back to user management screen and user “0908070600” will be removed | Pass |  |
| SLFROR17 | Face management | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to facial management screen with user table in right side | Pass |  |
| SLFROR18 | Face management - add face | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “\*” button on keypad | \_ Menu screen appeared on LCD | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to facial management screen with user table in right side | Pass |  |
| \_ Press “2” button on keypad | \_ Change to add face screen | Pass |  |
| \_ Select user name = "0945246345" using “C” and “D” button and press “A” button to confirm selection | \_ User = "0945246345" | Pass |  |
| \_ Press “1” button on keypad | \_ User's face is captured successfully | Pass |  |
| SLFROR19 | Face management - remove face | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner | \_ Press “3” button on keypad | \_Change to System menu | Pass |  |
| \_ Press “2” button on keypad | \_ Change to management menu | Pass |  |
| \_ Select user name = "0945246345" using “C” and “D” button and press “A” button to confirm selection | \_ User = "0945246345" is selected | Pass |  |
| \_ Press “2” button on keypad | \_ Confirm dialog will be shown | Pass |  |
| Press “A” button to confirm selection | \_ User = "0945246345" facial data is deleted | Pass |  |
| SLFROR20 | Lock system management | Raspberry is connected to: - GSM SIM900 using UART PORT - Solenoid lock using GPIO PORT - KEYPAD using GPIO PORT - CAMERA using GPIO PORT \_ LCD using HDMI port \_Login as home-owner \_ User is on System management mode | \_ Press “B” button on keypad | \_ Back to main menu | Pass |  |
| \_ Press “B” button on keypad | \_ Turn of the system management | Pass |  |

F. Report No.6 Software User’s Manual

1. Installation Guide

1.1. Setting up environment

1.1.1. Hardware requirement

- Raspberry Pi B2 Kit (change from Pi B2) with Micro SD class 10 16GB

- LCD 7” with solution 1280x720 pixel

- Number Keypad 4x4

- Raspberry pi Camera module Rev 1.3

- GSM900 module for Raspberry Pi (with 1 working SIM)

- Locker solenoid

- Circuit board design by our team

1.1.2. Software requirement

- Raspbian environment for Raspberry Pi (16-02-2015 release date or newer)

- Opencv library version 2.4.10

- Qt Development tool and Qt Creator

1.2. Deployment

1.2.1. Prepare deployment package

Project zip package which will be deployed.

1.2.2. Configure Raspberry Pi B2 before deploy

Step 1: Install Raspbian environment for Raspberry Pi B2.

Step 2: We choose Raspbian because it is based on Debian, because we choose Ubuntu is programming environment on PC, and Ubuntu and Debian are closely related. It is favorable to build our project on Raspberry Pi B2. Download Raspbian from this link:

<http://downloads.raspberrypi.org/raspbian_latest>.

Step 3: After download unzip file, obtained \*.img file.

Step 4: Use Win32 Disk Imager to write a raw disk image to Micro SD device,can be downloaded here: <http://sourceforge.net/projects/win32diskimager/>

Step 5: Put Micro SD to PC, target to \*.img file and click “write”.

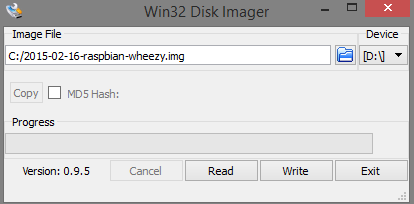


Figure 60. Write raw disk image to MicroSD

Step 6: Put Micro SD into Raspbery Pi B2, then turn on power. We must have 1 keyboard, screen connected to Raspberry VIA HDMI cable, and internet cable to install some requirement package.

Step 7: Configure raspbian on the first running



Figure 61. Configure raspbian

Step 8: Config options 1, 2, 5, then restart

1.2.3. Configure IP for Raspberr

Step 1: Open command line

Step 2: At command line, enter: sudo nano /etc/network/interfaces

Step 3: And replace text on file by this:

auto lo

iface lo inet loopback //default of Kit

iface eth0 inet static //config LAN network

address 192.168.1.230 //set static IP for raspbery

// if your moderm have defferent gateway, you must set other IP //following gateway

netmask 255.255.255.0

gateway 192.168.1.1 //set gateway

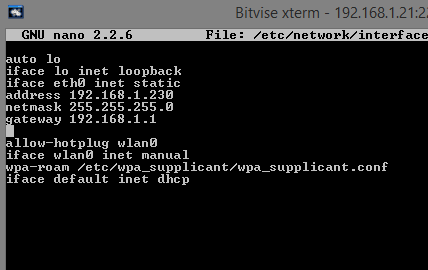
****

Figure 62. Configure IP network

Step 4: Press ctrl+X to save, then restart again. Now your raspberry pi have IP 192.168.1.230. You can set to other IP VIA modem IP and visible to remote Raspberry from your PC VIA Terminal. Install xrdp if you want to remote desktop. Xrdp uses the remote desktop protocol to present a GUI to the user: Sudo apt-get install xrdp

1.2.4. Install some needed packages

Step 1: Open command line

Step 2: At command line, enter:

- Update some needed available package : sudo apt-get update

- Update to newer version: sudo apt-get upgrade

- Install essential file: sudo apt-get install build-essential cmake libv4l-dev pkg-config libgtk2.0-dev libtiff4-dev libjasper-dev libavformat-dev libswscale-dev libavcodec-dev libjpeg-dev libpng-dev .

\*Note:

**build-essentials** is a reference for all the packages needed to compile a debian package.

**cmake** is cross-platform for managing the build process of software.

**libgtk2.0-dev** The default GUI backend for highgui on OSes other than Windows and MacOSX.

**libtiff4-dev** For reading and writing TIFF images.

**libjasper-dev** Library for the coding and manipulation of images (e.g. JPEG-2000 format).

**libavformat-dev** Library containing demuxers and muxers for audio/video container formats.

**libswscale-dev** Library containing video image scaling and colorspace/pixelformat conversion routines.

**libavcodec-dev** Library containing all the FFmpeg audio/video encoders and decoders.

**libjpeg-dev** For reading and writing JPEG images.

**libpng-dev** For reading and writing PNG images.

1.2.5. Install OpenCV library

We choose openCV 2.4.10 but not 3.0 because 3.0 versions still have many errors, and 2.4.10 is more stabilize.

Download openCV 2.4.10 from:

<http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip/download>

Or, using wget to download by:

Step 1: Open command line

Step 2: At command line, enter:

wget <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip>

Step 3: Unzip file: unzip opencv-2.4.10.zip

Step 4: Change directory to opencv folder: cd opencv-2.4.10

Step 5: Create new folder to build: mkdir release. Then entered this folder: cd release

Step 6: Use Cmake to generate makefile (is a special file, containing shell commands contain a list of rules)

cmake -D CMAKE\_BUILD\_TYPE=RELEASE –D CMAKE\_INSTALL\_PREFIX=/usr/local ..

Step 7: Run make to start compile: make

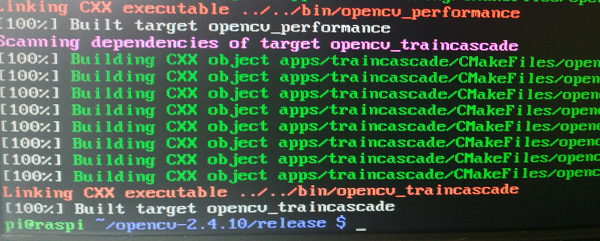


Figure 63. Start compile

Step 8: Install package openCV:

sudo make install

sudo ldconfig

\* “make” command may take several hours to build

1.2.6. Install QT

Step 1: Open command line.

Step 2: Install cross-platform C++ application framework can use to build Qt project: sudo apt\_get install qt4-dev-tools

Step 3: Install Qt programming: sudo apt-get install qtcreator

Step 4: Install standard terminal emulator for the X Window System: sudo apt-get install xterm

Step 5: Install subversion (version control system): sudo apt-get install subversion

1.2.7. Install bcm2835 to work with GPIOs and UART

Step 1: Open command line.

Step 2: In command line, enter:

Wget www.airspayce.com/mikem/bcm2835/bcm2835-1.42.tar.gz

tar zxvf bcm2835-1.xx.tar.gz

cd bcm2835-1.xx

./configure

make

sudo make check

sudo make install

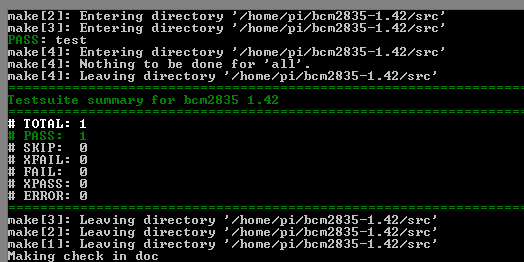


Figure 64. Install bcm2835

Step 1: Config UART will be available for another process to use it:

sudo nano /boot/cmdline.txt

Step 2: Delete parameters involving “ttyAMA0″ example “console=ttyAMA0,115200”. Save by press Ctrl+x

Step 3: Enter: sudo nano /etc/inittab

Step 4: Search for the serial port usage by using CTRL+W and typing: “ttyAMA0” comment that line content “ttyAMA0” by enter “#” at start of that line, then save.

Step 5: Restart Raspberry pi: sudo shutdown –r now

1.2.8. Set Raspberry will run GUI application when start without run desktop

Step 1: Edit file rc.local. this step can help Raspberry auto login and start GUI (startx):

Sudo nano /etc/rc.local

Step 2: Add this line at first

su -s /bin/bash -c startx pi&

“pi” is default user of raspberry pi. If you use other use, place it, then save file.

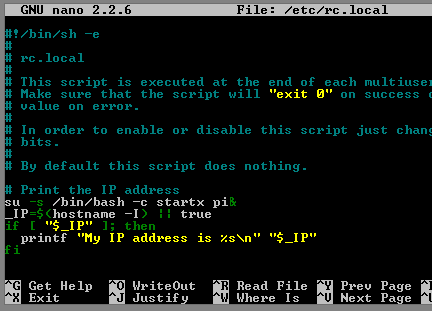


Figure 65. Modify rc.local

Step 3: run :sudo dpkg-reconfigure x11-common. And select “anybody” to config who are allowed to start X server (startx)

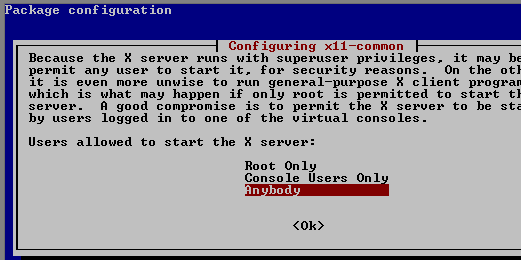


Figure 66. Config X Server

Step 4**:** Create file .xinitrc. This file contain any command you want to run, it will auto run when login with user have directory that file .xinitrc place on

sudo nano /home/pi/.xinitrc

With “pi” is your user, and enter command you want to run when start Raspberry. Besides, some command line can be added in to this file:

- Keep screen always turn on

xset s noblank

xset s off

xset -dpms

- Load the module bcm2835-v4l2 to work with Raspi Camera

sudo modprobe bcm2835-v4l2

#run main GUI application

sudo ./home/pi/DemoCapstone/DemoCapstone

Step 5: Then save. Now when we start raspberry, application will run without start desktop

1.2.9. Fix solution on LCD 7 inch

Application will not run full screen with 720x1280px if you do not configure

Step 1: Open command line, enter: Sudo nano /boot/config.txt

Step 2: Uncommen line “disable\_overscan=1 and make sure number is “1”. Save and reboot: sudo shutdown -r now

1.2.10. Deploy application on Raspberry Pi B2

Step 1: Copy .zip file of application to raspberry VIA scp command, bitvise ssh, or any program at directory “/home/pi”

Step 2: At raspberry, unzip file zip, open command line, enter:

unzip DemoCapstone.zip

cd DemoCapstone

Step 3: Build qt application.

- Create makefile by qt development tool: qmake DemoCapstone.pro;

- Build application: make

Step 4: Finally, reboot raspberry and that auto run application (with path to execution file (DemoCapstone file on DemoCapstone directory) you entered on .xinitrc file

1.2.11. Connection map between components

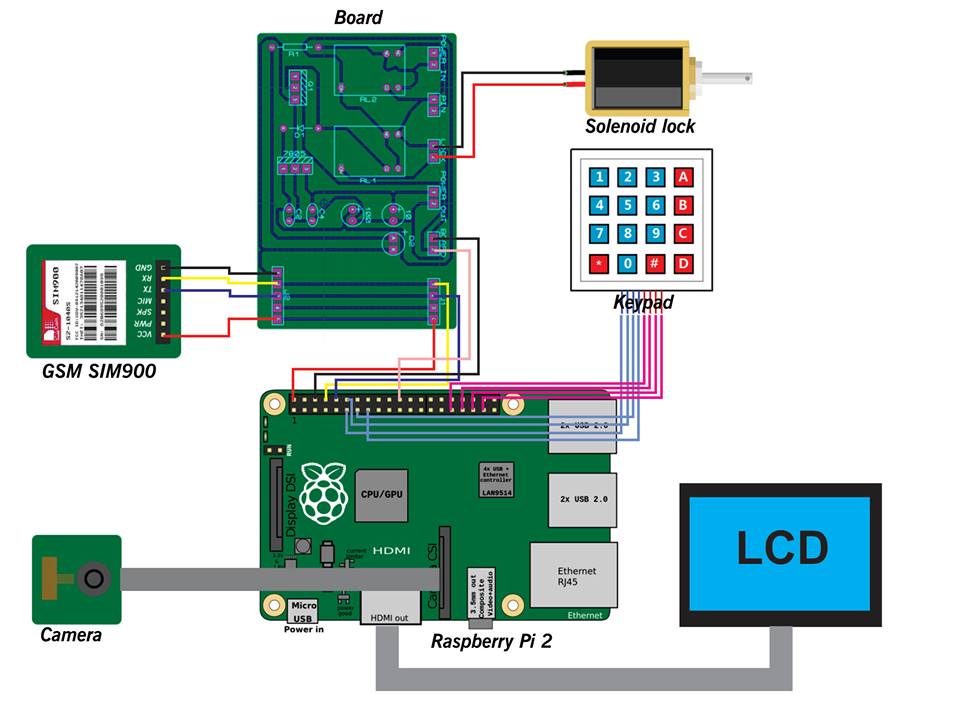


Figure 67. Connection map between all components

1. User Guide

2.1. Unlock by facial recognition

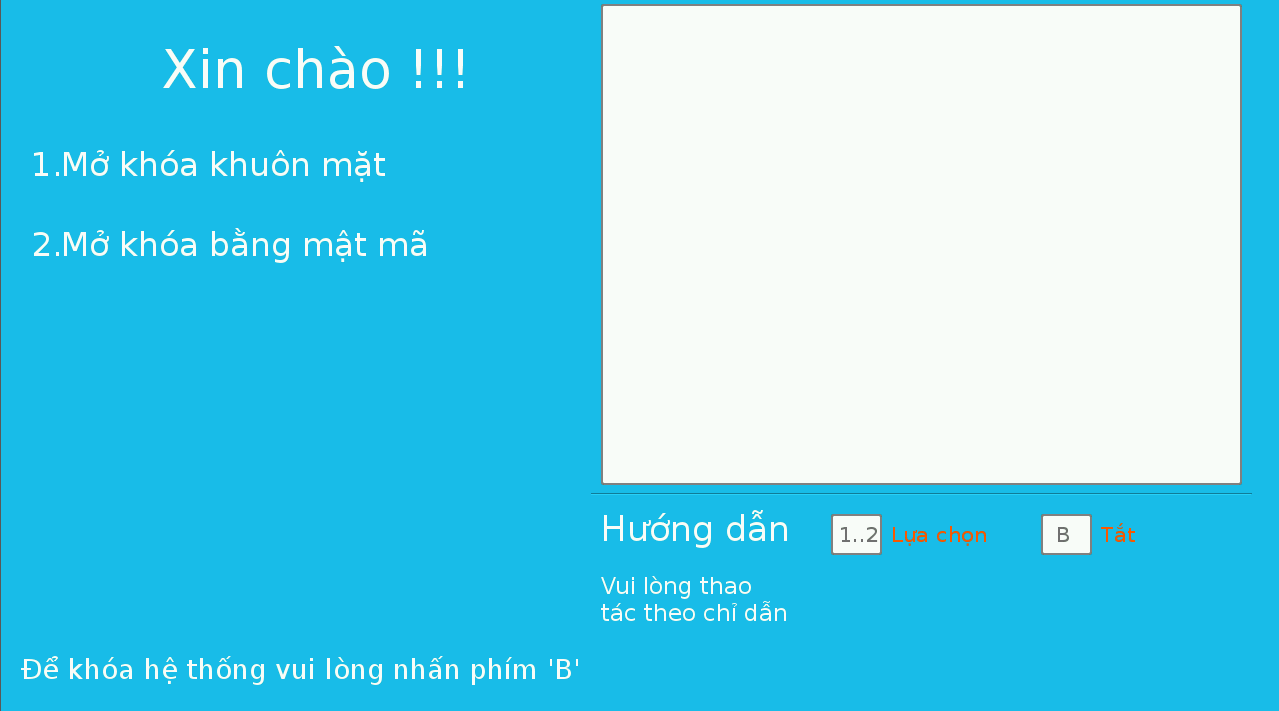


Figure 68. “Menu” screen

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In menu screen. Press “1” button to active unlock by facial recognition |
| 2 | View the screen to adjust your position and light until see system detects your face by red rectangle. |
| 3 | Stand from 3 to 5 seconds to system execute recognition process. |
| 4 | Lock will be open if system recognize successfully |

2.2. Unlock by keypad mode

To unlock by keypad mode we have 2 cases: one is user have not had passcode and the other case that user had the passcode.

Case 1: User has not had passcode, so follow step to generate passcode:

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In menu screen. Press “2” button to active unlock by keypad |
| 2 | “Unlock by keypad” screen will be show ( Figure 10) |
| 3 | Press “1” button to select phone number line. |
| 4 | Press number from 0 to 9 to fill in phone number. Can use “#” button to delete previous number. |
| 5 | Press “\*” button to cancel select |
| 6 | Press “2” button to select password line |
| 7 | Press number from “0” to “9” button to fill in phone number. Can use “#” button to delete previous number. |
| 8 | Press “\*” button to cancel select |
| 9 | Press “A” button to accept send passcode |
| 10 | “Enter passcode” screen will be shown after passcode generate and send successfully ( Figure 11) |
| 11 | Press number from “0” to “9” button to fill in passcode with number you receive |
| 12 | Press “A” button to unlock |
| 13 | Lock will be open if passcode is valid |

Case 2: In case, user has had passcode

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In menu screen. Press “2” button to active unlock by keypad |
| 2 | “Unlock by keypad” screen will be show ( Figure 10) |
| 3 | Press “C” button to open “Enter passcode” screen ( Figure 11) |
| 4 | Press number from “0” to “9” button to fill in passcode with number you receive |
| 5 | Press “A” button to unlock |
| 6 | Lock will be open if passcode is valid |

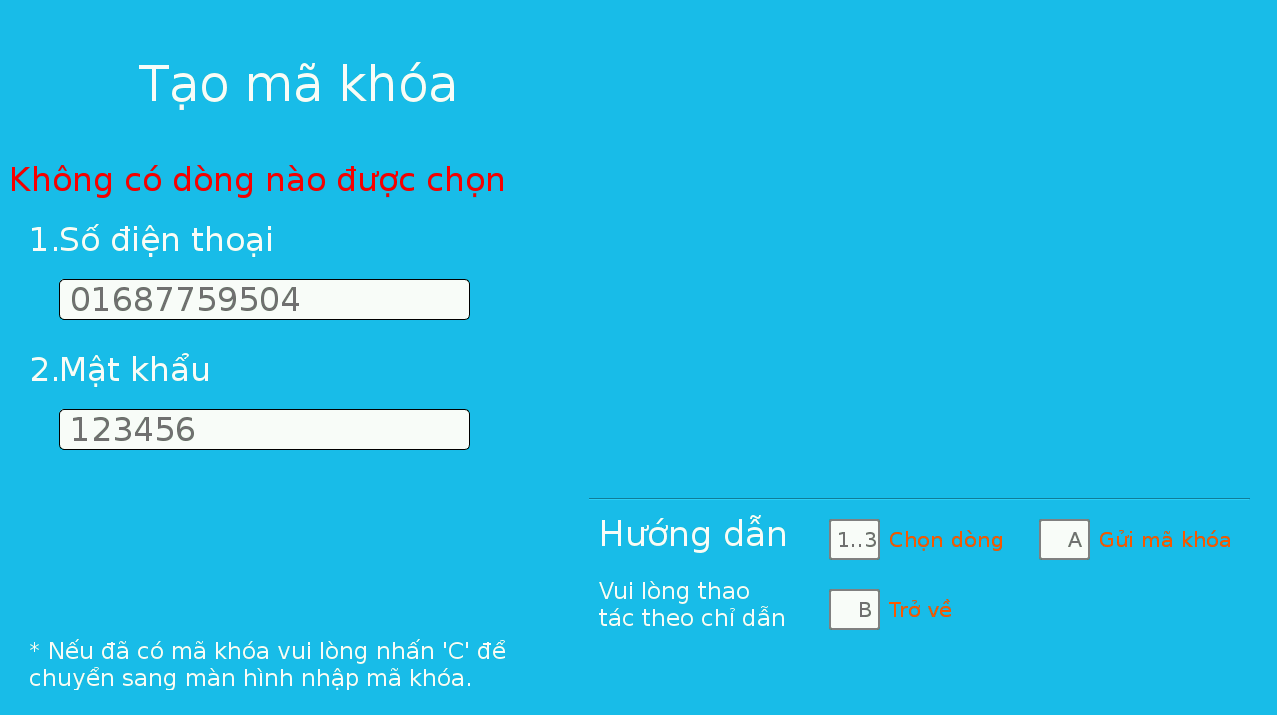


Figure 69. “Keypad mode” screen

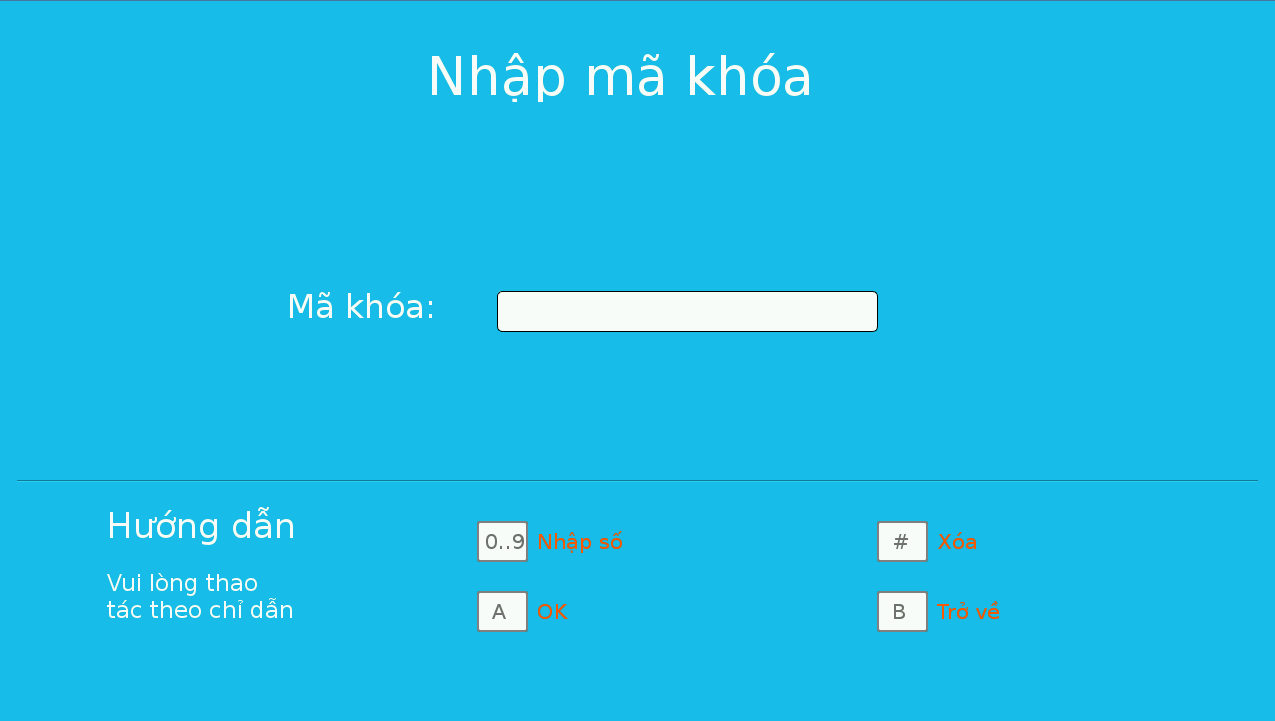


Figure 70. “Enter passcode” screen

2.3. User management

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Unlock by using facial recognition with home-owner role (Figure 12). |
| 2 | In menu screen. Press “3” button to go “System management” screen (Figure 13). |
| 3 | In “System management” screen, press “1” button to go “User management” screen (Figure 14). |



Figure 71. “Enter passcode” screen



Figure 72. “System management” screen

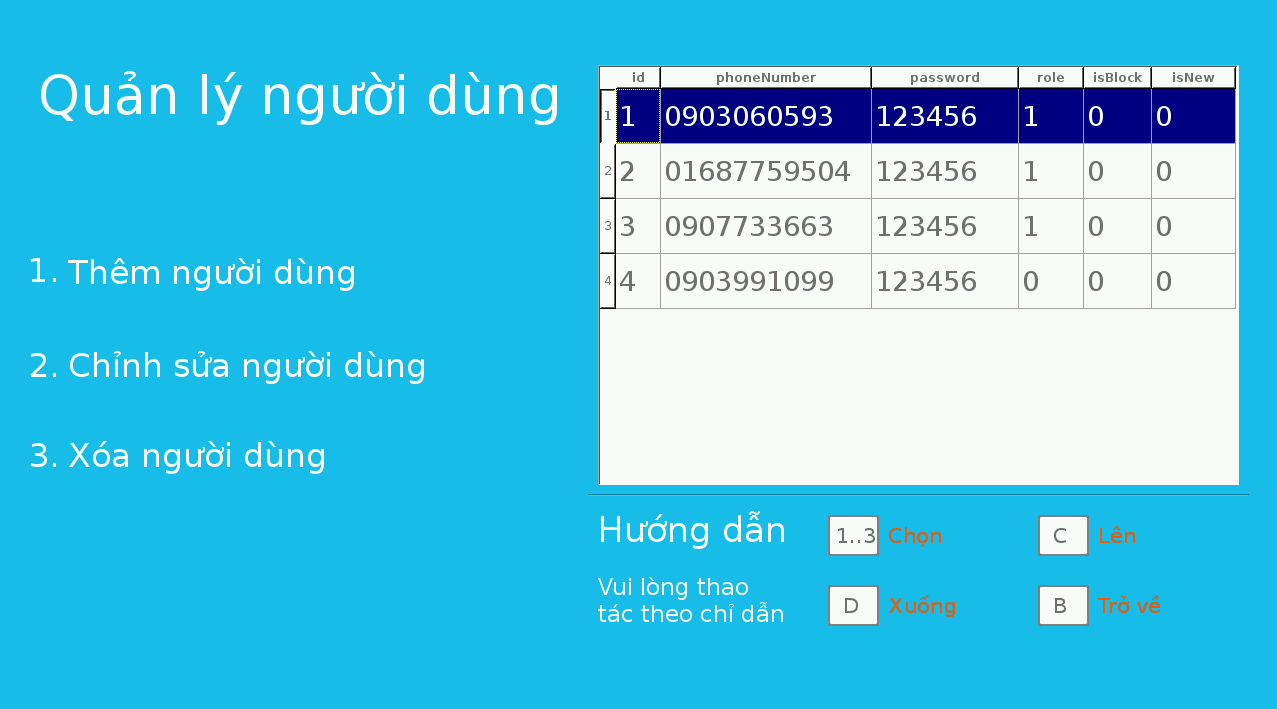


Figure 73. “User management” screen

2.4. Add new user

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In “User management” screen (Figure 14). Press “1” button to go “Add user” screen (Figure 15). |
| 1 | Press “1” button to select phone number line. |
| 2 | Press from “0” to “9” button to entered phone number |
| 3 | Press “\*” button to cancel selected line |
| 4 | Press “2” button to select role line |
| 5 | Press “1” button for “Chủ nhà” and press “2” for “Thành viên” |
| 6 | Press “\*” button to cancel selected line |
| 7 | Press “3” button to select password line |
| 8 | Press from “0” to “9” button to entered phone number |
| 9 | Press “\*” button to cancel selected line |
| 10 | Press “A” button to add user. |

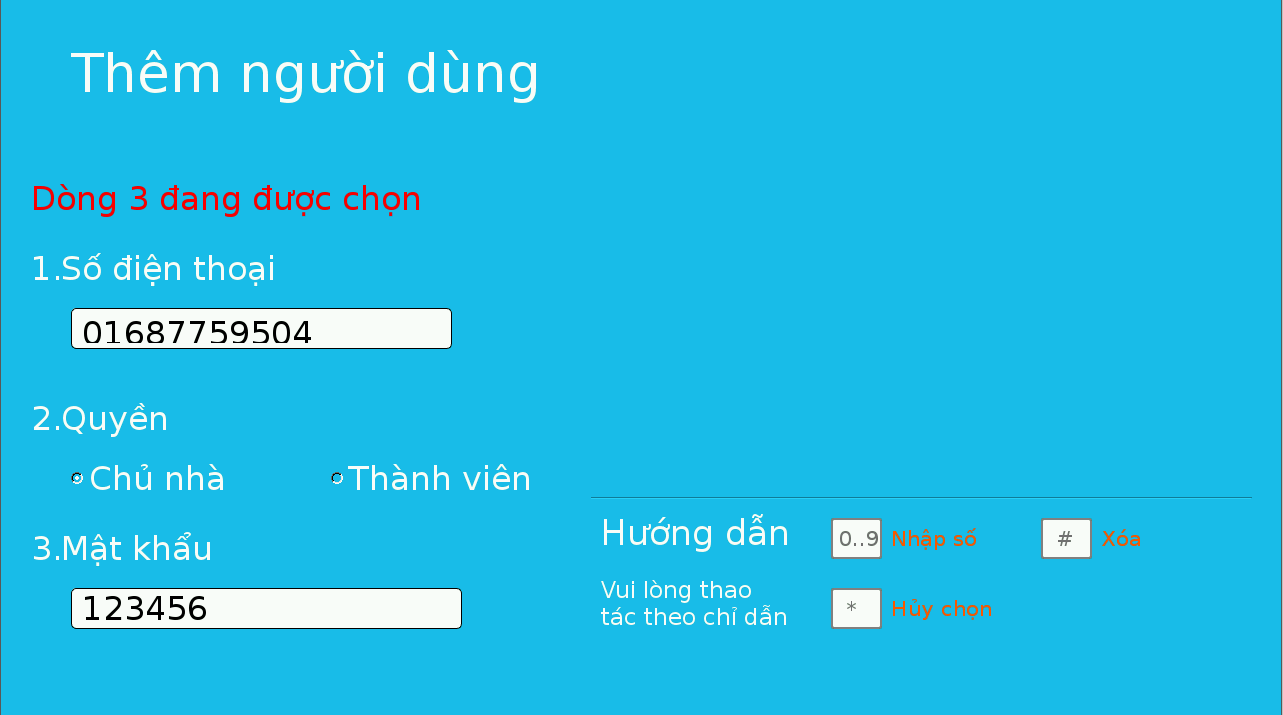


Figure 74. “Add user” screen

2.5. Update user

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In “User management” screen (Figure 14). Press “C” to up and “D” to down, after select user who will be modified. Press “2” button, “Modify user” screen will be shown (Figure 16). |
| 2 | Press “1” button to select phone number line |
| 3 | Press from “0” to “9” button to entered phone number |
| 4 | Press “\*” button to cancel selected line |
| 5 | Press “2” button to select role line |
| 6 | Press “1” button for “Chủ nhà” and press “2” for “Thành viên” |
| 7 | Press “\*” button to cancel selected line |
| 8 | Press “3” button to select password line |
| 9 | Press from “0” to “9” button to entered phone number |
| 10 | Press “\*” button to cancel selected line |
| 11 | Press “4” button to select status line |
| 12 | Press “1” button to check and press “2” button to uncheck. |
| 13 | Press “\*” button to cancel selected line |
| 14 | Press “A” to update user’s information. |



Figure 75. “Modify user” screen

2.6. Delete user

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In “User management” screen (Figure 14). Press “C” to up and “D” to down, after select user who will be deleted. |
| 1 | Press “3” button; confirm dialog screen will be shown (Figure 17). |
| 2 | Press “A” button to delete user |

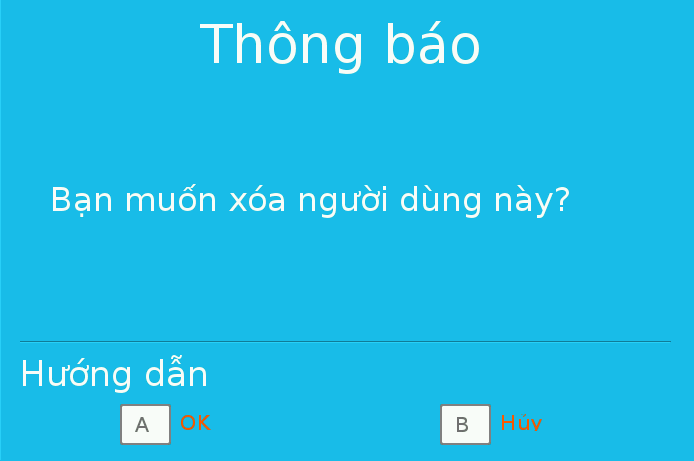


Figure 76. “Confirm Delete user” screen

2.7. Face management

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Unlock by using facial recognition with home-owner role (Figure 12). |
| 2 | In menu screen. Press “3” button to go “System management” screen (Figure 12). |
| 3 | In “System management” screen, press “2” button to go “Face management” screen (Figure 18). |

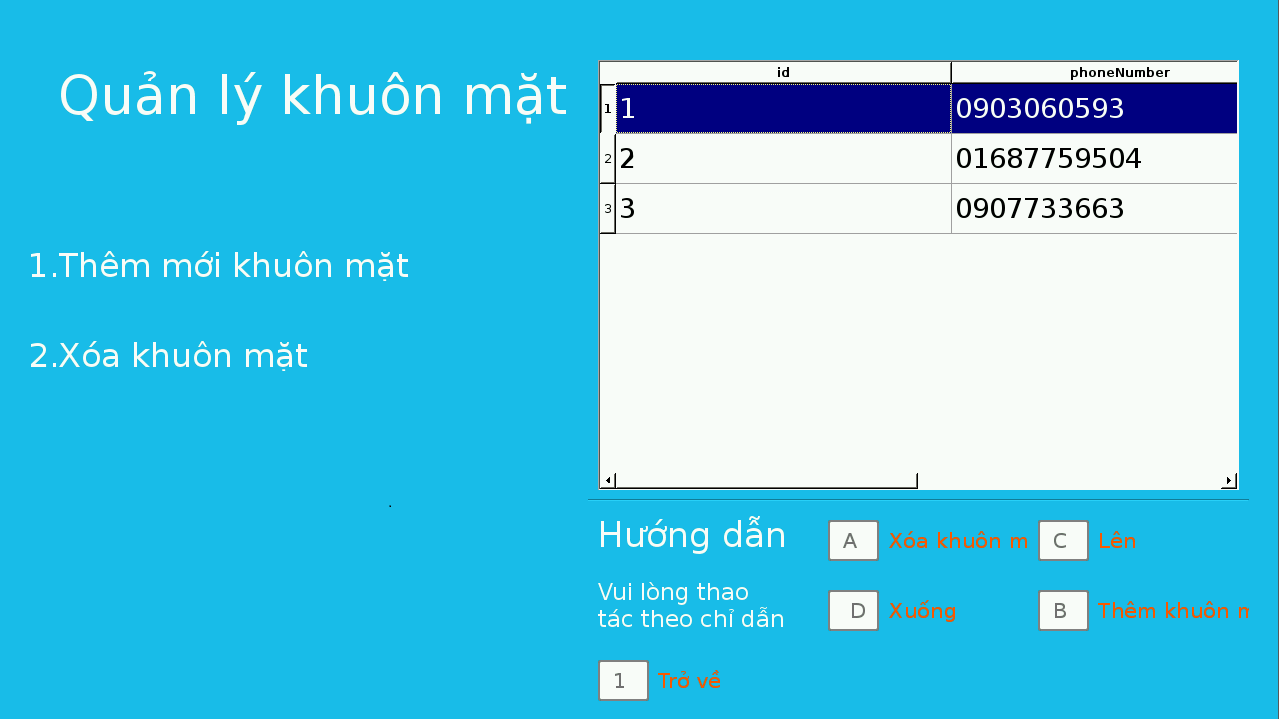


Figure 77. “Face management” screen

2.8. Add new face

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In “Face management” screen (Figure 18), press “1” button to go “Add face” screen (Figure 19) |
| 1 | Press “C” to up and “D” to down to select phone number which represent for user who need to add face. |
| 2 | Press “A” to select phone number. |
| 3 | Press “1” button to capture |
| 4 | Adjust position and light until system can detect face by red reactangle. |
| 5 | Stand for from 3 to 5 minutes until dialog show “Đã thêm thành công” |

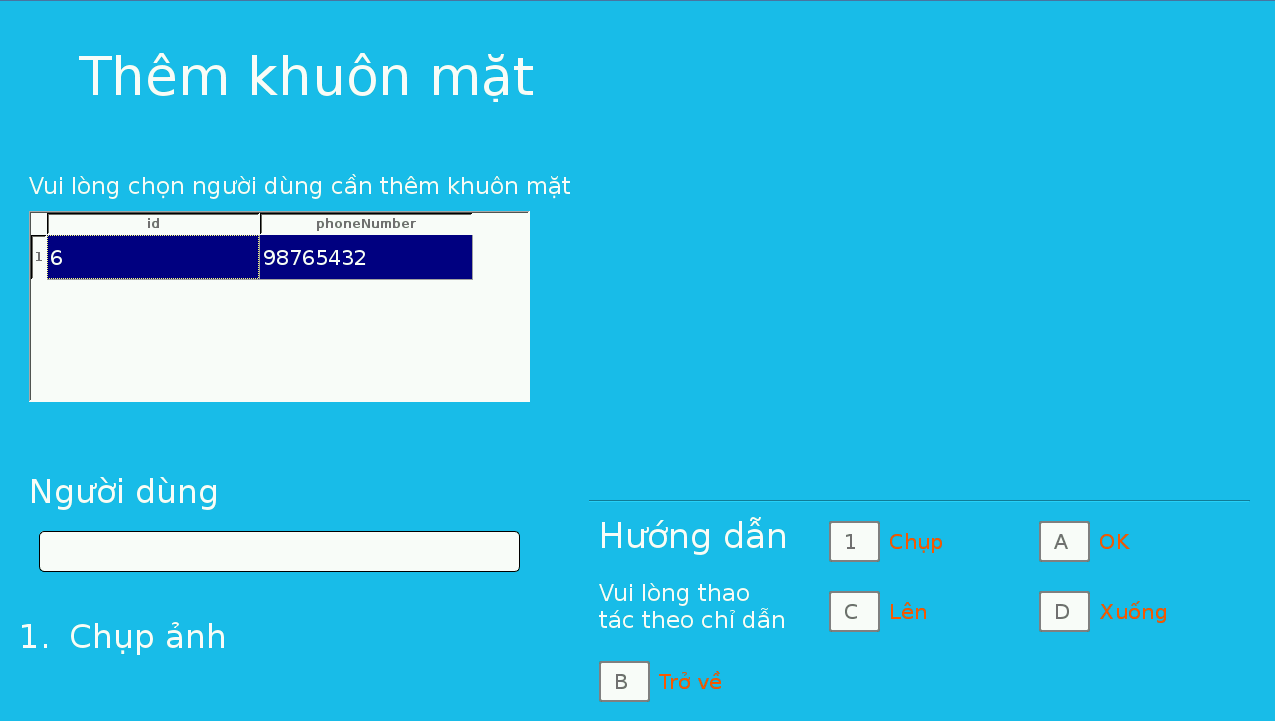


Figure 78. “Add face” screen

2.9. Delete face

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In “Face management” screen (Figure 18) |
| 2 | Press “C” to up and “D” to down to select phone number which represent for user who need to delete face. |
| 3 | Press “A” to delete face. |
| 4 | Confirm dialog will shown (Figure 20) |
| 5 | Adjust position and light until system can detect face by red reactangle. |
| 6 | Stand for from 3 to 5 minutes until dialog show “Đã thêm thành công” |

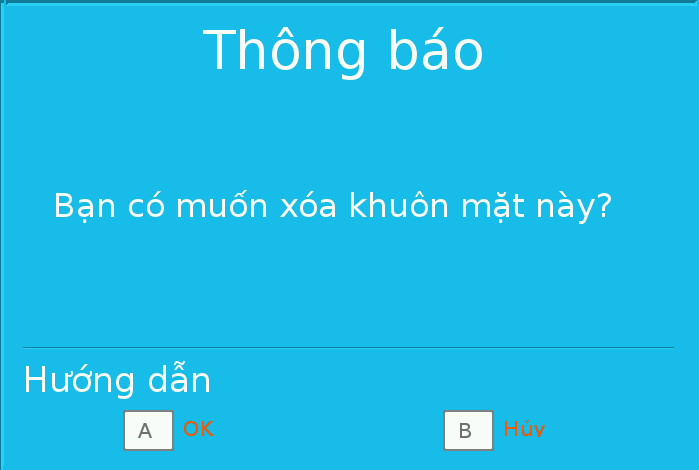


Figure 79. “Confirm delete face” screen

2.10. Show log face

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Unlock by using facial recognition with home-owner role (Figure 12). |
| 2 | In menu screen. Press “3” button to go “System management” screen (Figure 12). |
| 3 | In “System management” screen, press “3” button to go “Show log face” screen (Figure 21). |
| 4 | Press “C” to up and “D” to down to select face which wants to show. |
| 5 | Press “A” to show face. |

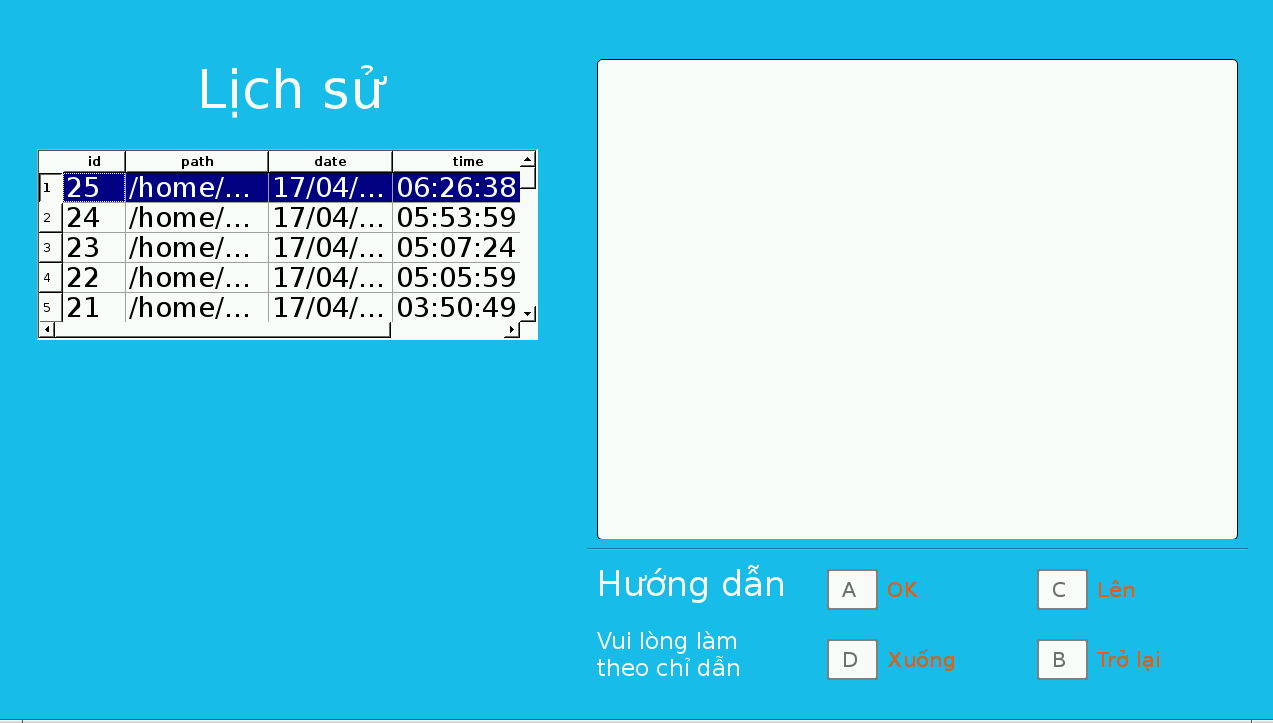


Figure 80. “Show log face” screen

2.10. Delete all information

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Unlock by using facial recognition with home-owner role (Figure 12). |
| 2 | In menu screen. Press “3” button to go “System management” screen (Figure 12). |
| 3 | In “System management” screen, press “4” button to go “Setting” screen (Figure 21). |
| 4 | Press “1” to delete all information. Confirm dialog will show (Figure) |
| 5 | Press “A” to delete all information. |



Figure 81. “Setting” screen



Figure 82. “Confirm delete all” screen

2.10. Set threshold for SD Card storage

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Unlock by using facial recognition with home-owner role (Figure 12). |
| 2 | In menu screen. Press “3” button to go “System management” screen (Figure 12). |
| 3 | In “System management” screen, press “4” button to go “Setting” screen (Figure 21). |
| 4 | Press “3” , “6”, “8” button to select storage. |
| 5 | Press “A” to set threshold for SD card storage. |

G. Appendix

1. Rasbian operating system:

http://www.raspbian.org/RaspbianAbout

2. Qt developer resource:

<http://www.qt.io/developers/>

3. Qt documentation:

http://doc.qt.io/

4. OpenCV library to detect object: <http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html>

5. OpenCV library to rect: http://docs.opencv.org/modules/contrib/doc/facerec/facerec\_tutorial.html

6. OpenCV library to transform image: <http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html>

7. Raspberry Pi Camera module document:

<http://www.ics.com/blog/raspberry-pi-camera-module#.VDqccvldX6U>

8. Graphic application without: <https://www.raspberrypi.org/forums/viewtopic.php?p=344408>

9. Compile Qt project:

<http://www.aboutlinux.info/2006/01/creating-and-compiling-qt-projects-on.html>

10. GPIO library for Raspberry:

<http://www.airspayce.com/mikem/bcm2835/>

11. How to setup Qt in Raspberry:

<http://innovationclub.vn/threads/cai-qt-tren-raspberry-pi-b.125/>

12. Multithread in QT:

<https://code.google.com/p/qt-opencv-multithreaded/wiki/Documentation>

13. Setup rasbian OS:

http://chiaseprojects.blogspot.com/2014/06/huong-dan-cai-at-he-ieu-hanh-raspbian.html

14. Test classification:

<http://forums.testervn.com/archive/index.php/t-1143.html>

15. Raspberry Pi B2 specification:

<http://www.pcworld.com/article/2886260/raspberry-pi-2-review-the-revolutionary-35-micro-pc-supercharged.html>

16. GSM Sim900 specification:

<http://hshop.vn/module-gsm-sim900?search=sim900>

17. Solenoid specification:

<http://www.adafruit.com/products/1512>

18. Development in/out with raspberry:

http://codientu.org/threads/10519/

19. UART raspberry guideline:

<http://www.raspberry-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart>

20. GPRS:

http://www.comminica.se/multitech/gsprs\_at.pdf