HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

REPORT PROJECT 1

Building an Event Management Application

TRẦN THỊ TƯỜNG VÂN
van.ttt225590@sis.hust.edu.vn

PHẠM MINH TIẾN
tien.pm2255555@sis.hust.edu.vn

Major: Cyber Security

Supervisor: PhD Tạ Trung Kiên ______________

Signature

Department: Computer Engineering

School: The School of Information and Communications Technology

HANOI, 06/2024

ACKNOWLEDGMENTS

Sincere thank you, Mr. Ta Trung Kien, for your diligent attention in helping us complete Project 1. Your dedication and expertise played a pivotal role in our success, and we are truly grateful for the opportunity to have learned from you. We are particularly thankful for your patience in explaining complex concepts and ensuring that we were always on the right track. Your willingness to answer our questions, provide us with relevant resources, and encourage us to persevere through challenges was instrumental in our progress. Furthermore, we recognize the importance of self-learning and independent research in project completion. The process of actively seeking out knowledge related to Project 1 has equipped us with valuable skills and experiences that will undoubtedly serve us well in our future endeavors. Once again, we extend our heartfelt appreciation for your exceptional mentorship. Your guidance has been invaluable, and we hope to continue learning from you in future projects.

ABSTRACT

This report presents the development and implementation of an Event Management Application designed to streamline the process of organizing and managing events. The application aims to provide a user-friendly interface for creating, scheduling, and coordinating various types of events. Additionally, the report discusses the technology stack used for the application, the design and architecture considerations, as well as the challenges faced during the development process. Overall, the Event Management Application serves as a friendly tool for simplifying event planning and execution, helping to improve effectiveness in managing events.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Motivation

Due to the increased frequency of events, developing an effective event management system is necessary. This application will help the managers create an event, invite someone to their event so that the process of preparation for the event will faster and more effective.

1.2 Application Requirements

The application includes the following functions: register a new account, log in to your account to use, and create an event with time, location, event type (Public or Private), description, Send event invitations to friends, accept invitation to join the event, send a request to participate in a certain Public event, grant requests to participate in the event

1.3 Report Layout

The primary content is in the next chapters. Chapter 2 discusses theoretical foundations, focusing on the Java programming language, databases, and cryptography. Chapter 3 is on method and design, which covers creating diagrams such use case diagrams, class diagrams, sequence diagrams, and databases. The fourth chapter discusses how we structure our code into packages, how we build UI, and how we secure database information. Chapter 5 covers the testing process, followed by Chapter 6 on usage instructions, and finally Chapter 7 on conclusion and future development.

1.4 Contribution

We all strive to complete our work. Tran Thi Tuong Van drew the diagrams and coded the controller and the UI, while Pham Minh Tien drew the database, set up the database, coded the JDBC interaction, encryption, and report writing.

CHAPTER 2. THEORETICAL BASIS

This chapter will explain the technologies used in the project.

2.1 Java Programming Language

During the preparation for this project, we were instructed to use the Java language to complete the topic that we are assigned. Java is a cross-platform programming language that offers extensive libraries, multithreading support, high reliability, and integrated security features.

2.2 Database

In order to store system's information, we used MySQL data management system. MySQL support many types of connection such as Local Conenction, Remote Connection, TCP/IP Protocol Connection and Socket Protocol Connection. Since the program is required to be usable by multiple users sharing the same LAN network, we use TCP/IP protocol connection to connect the database and the program.

2.3 Security Coding

Secure coding practice is a very importance part of this CyberSecurity project. This includes input validation, authentication and authorization, data protection, secure coding pactices and security testing.

We employ encryption techniques to protect sensitive data. More precise, we use Advanced Encryption Standard(AES) for sensitive data in user and private event and SHA-256 for one-way password encryption.

For security testing, we also make use of Jenkins and SonarQube, which are code analysis tools, to help clean our code and reduce the bugs as well.

CHAPTER 3. METHOD

3.1 Diagram design

3.1.1 UseCase diagram

The features of the application are listed in the Usecase diagram. The application allows users to perform the following basic functions. Firstly, it is to register a new account. Secondly, that is to log in to the created account. Next is to create a new event, from which users can invite other users to join and accept invitations from other users. Finally, it is registering to participate in public events from other users as well as accepting registration requests from other users.

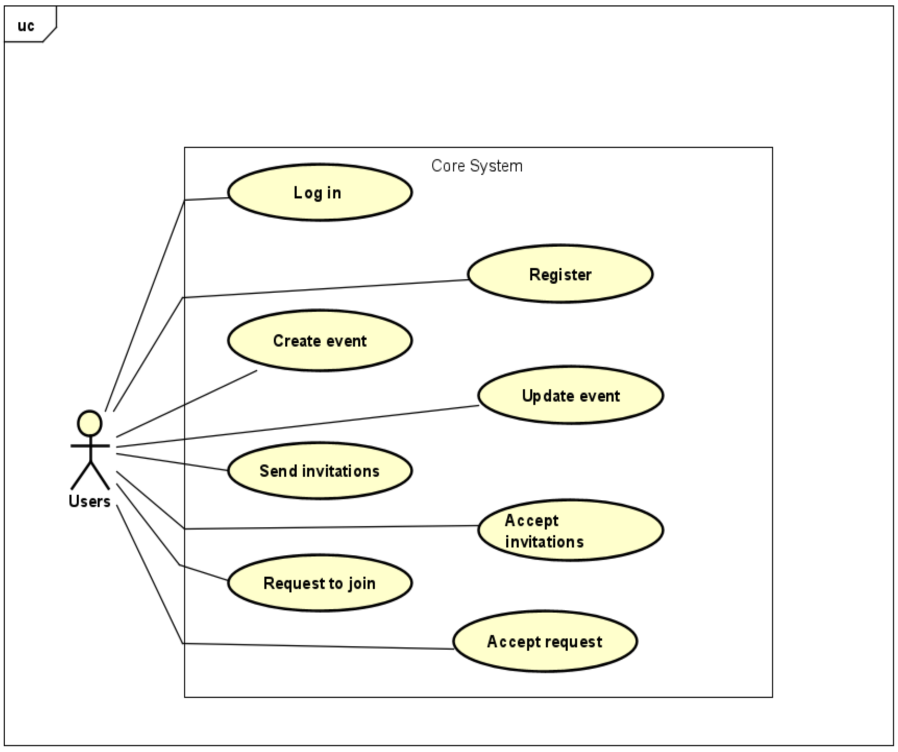


Figure 3.1: Usecase Diagram

3.1.2 Class diagram

a, Main class diagram

The main package has three classes: User, Event, and EventList, which are the main objects of the program. The User class and Event class have attributes that follow the requirements of the topic. The EventList class contains a list of event used for store data which will be used in controller classes.

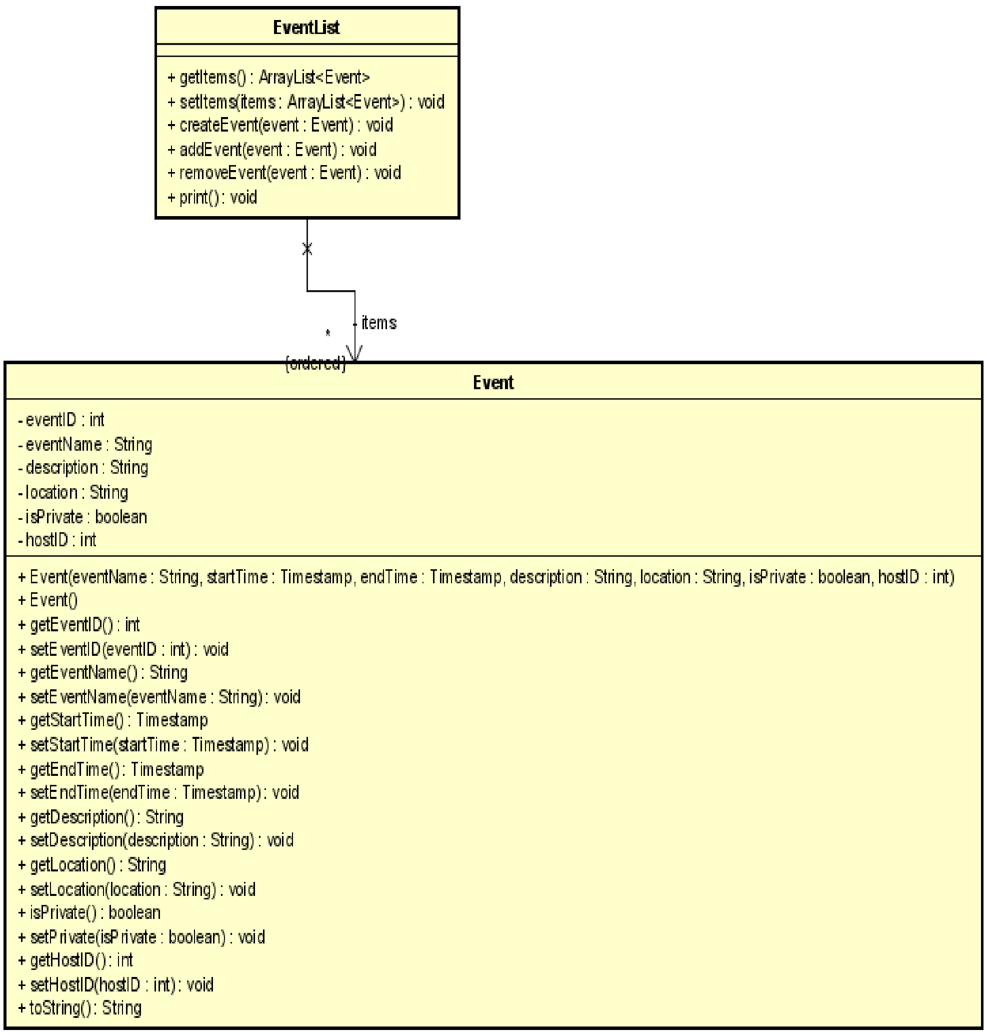

EventList
+ getItems() : ArrayList<Event>
+ setItems(items : ArrayList<Event>) : void
+ createEvent(event : Event) : void
+ addEvent(event : Event) : void
+ removeEvent(event : Event) : void
+ print() : void
- items
*
{ordered}
Event
- eventID : int
- eventName : String
- description : String
- location : String
- isPrivate : boolean
- hostID : int
+ Event(eventName : String, startTime : Timestamp, endTime : Timestamp, description : String, location : String, isPrivate : boolean, hostID : int)
+ Event()
+ getEventID() : int
+ setEventID(eventID : int) : void
+ getEventName() : String
+ setEventName(eventName : String) : void
+ getStartTime() : Timestamp
+ setStartTime(startTime : Timestamp) : void
+ getEndTime() : Timestamp
+ setEndTime(endTime : Timestamp) : void
+ getDescription() : String
+ setDescription(description : String) : void
+ getLocation() : String
+ setLocation(location : String) : void
+ isPrivate() : boolean
+ setPrivate(isPrivate : boolean) : void
+ getHostID() : int
+ setHostID(hostID : int) : void
+ toString() : String

Figure 3.2: Object class diagram-Event class

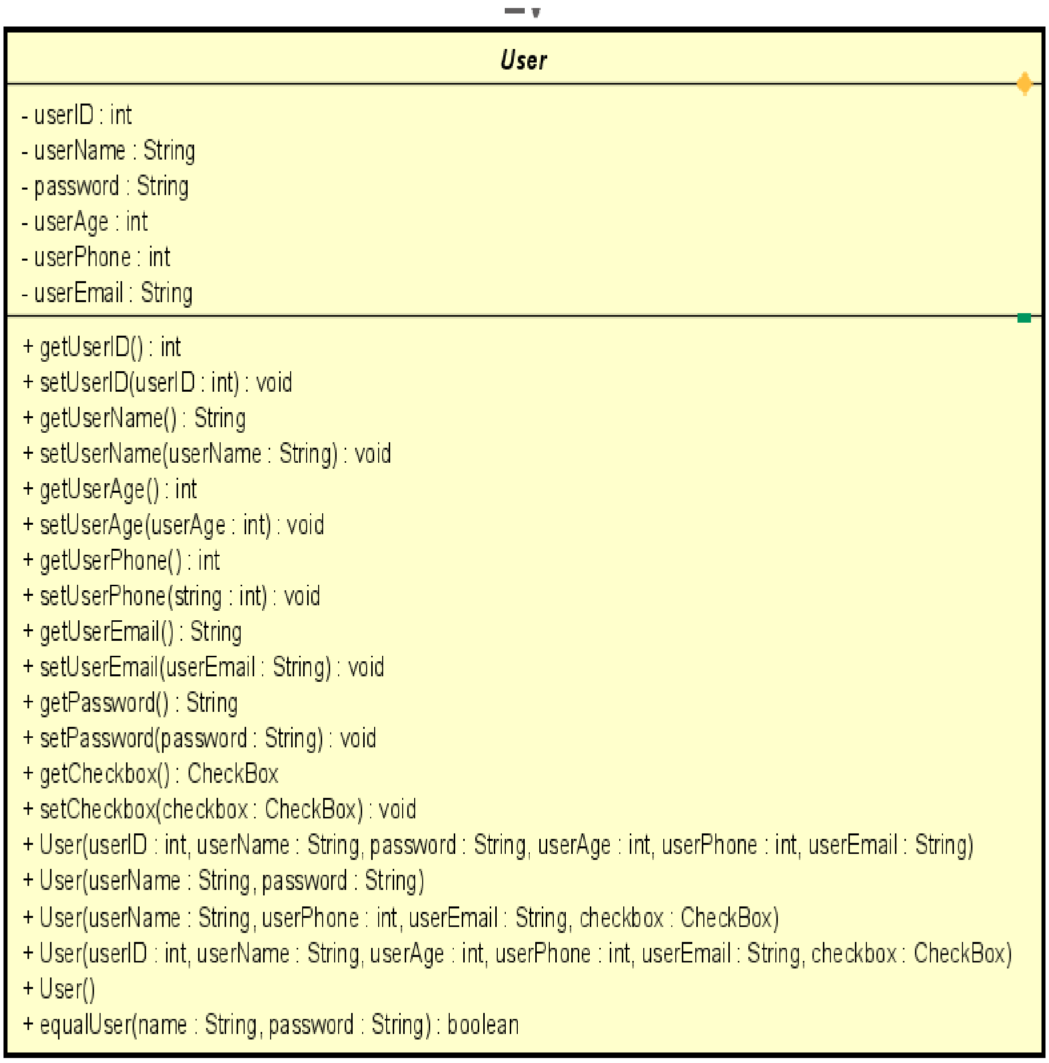

User
- userID : int
- userName : String
- password : String
- userAge : int
- userPhone : int
- userEmail : String
+ getUserID() : int
+ setUserID(userID : int) : void
+ getUserName() : String
+ setUserName(userName : String) : void
+ getUserAge() : int
+ setUserAge(userAge : int) : void
+ getUserPhone() : int
+ setUserPhone(string : int) : void
+ getUserEmail() : String
+ setUserEmail(userEmail : String) : void
+ getPassword() : String
+ setPassword(password : String) : void
+ getCheckbox() : CheckBox
+ setCheckbox(checkbox : CheckBox) : void
+ User(userID : int, userName : String, password : String, userAge : int, userPhone : int, userEmail : String)
+ User(userName : String, password : String)
+ User(userName : String, userPhone : int, userEmail : String, checkbox : CheckBox)
+ User(userID : int, userName : String, userAge : int, userPhone : int, userEmail : String, checkbox : CheckBox)
+ User()
+ equalUser(name : String, password : String) : boolean

Figure 3.3: Object class diagram-User class

b, LogIn class diagram

The login package contain two class: LogIn and LogInController. LogIn class is an inheritance of User and has a new attribute is CheckBox which will be used for management function. LogIn class connect to the database then store user's data into a LogIn. On the other hand, the LogInController class has two function are btnLogninPressed() and btnSignupPressed(). The btnLoginPressed() compare input data with stored data, if the input data is correct, screen will change to Home screen, else return the inccorrect messages. The btnSignuPressed() change the screen to SignUp screen.

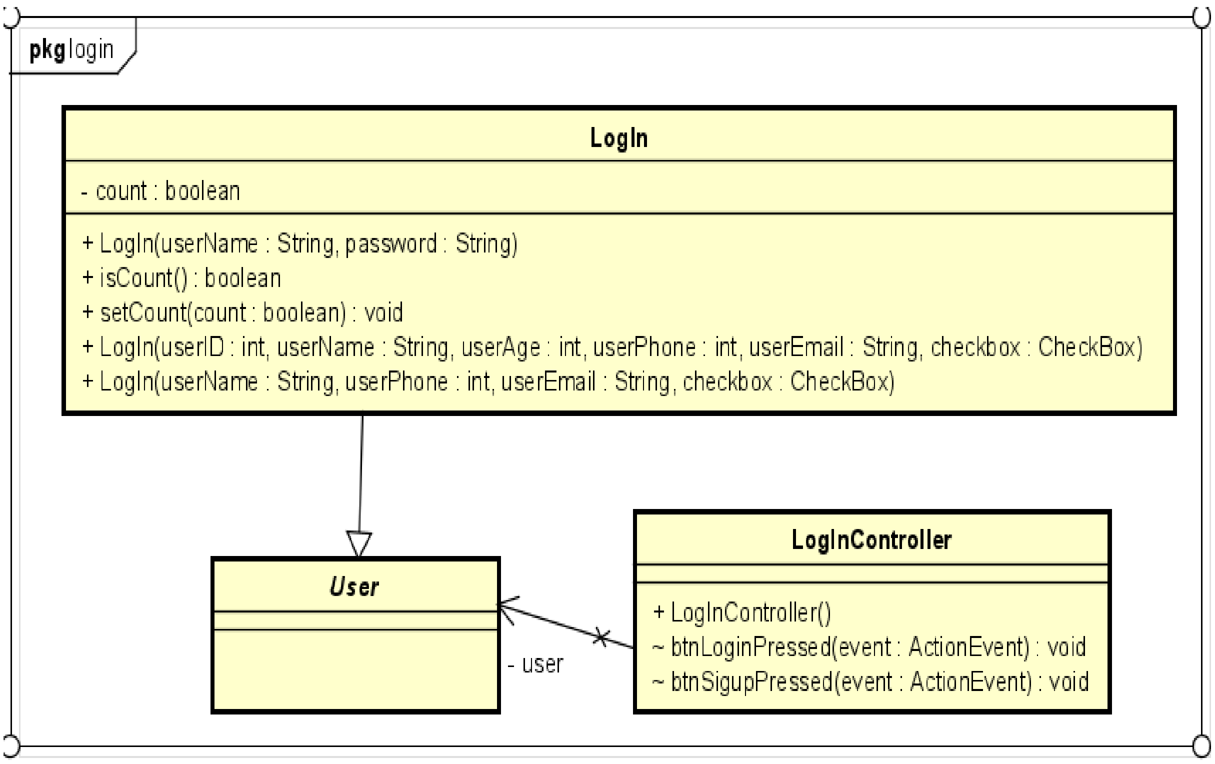

pkg login
LogIn
- count : boolean
+ LogIn(userName : String, password : String)
+ isCount() : boolean
+ setCount(count : boolean) : void
+ LogIn(userID : int, userName : String, userAge : int, userPhone : int, userEmail : String, checkbox : CheckBox)
+ LogIn(userName : String, userPhone : int, userEmail : String, checkbox : CheckBox)
User
- user
LogInController
+ LogInController()
~ btnLoginPressed(event : ActionEvent) : void
~ btnSigupPressed(event : ActionEvent) : void

Figure 3.4: LogIn class diagram

c, Home class diagram

The Hom package only has one class that is HomeController class. The HomeController includes imgNewEventClicked() connectes to dashboard package, btnUserEventClicked() connects to user package, btnManagerEventClicked() connects to manager package, imgMyAccountClicked() connects to account package and the btnCreateEventClicked() connects to create package. All the function of the HomeController connect to other package throw User attribute, which contain information about user and query the appropriate data.

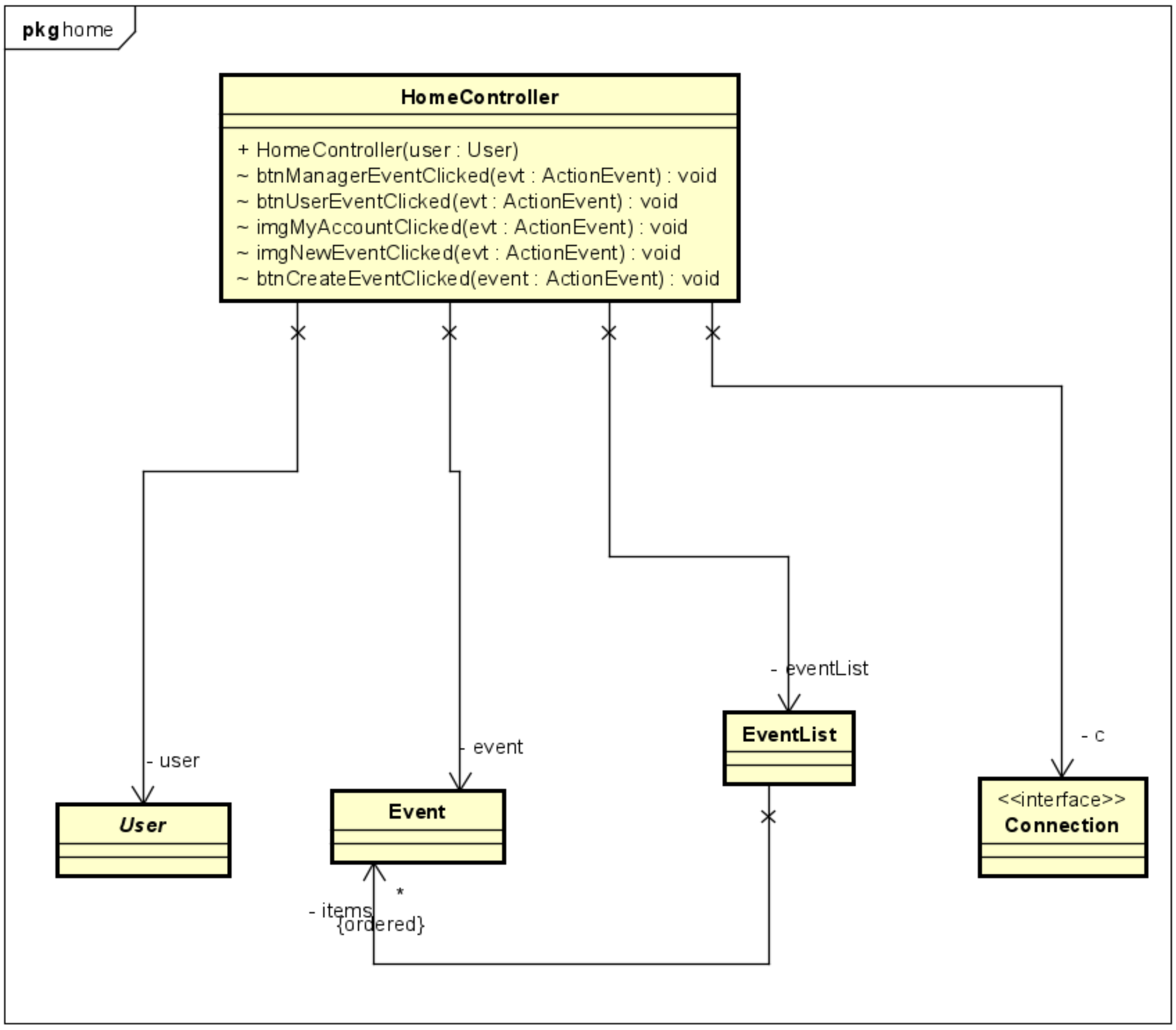

pkg home
HomeController
+ HomeController(user : User)
~ btnManagerEventClicked(evt : ActionEvent) : void
~ btnUserEventClicked(evt : ActionEvent) : void
~ imgMyAccountClicked(evt : ActionEvent) : void
~ imgNewEventClicked(evt : ActionEvent) : void
~ btnCreateEventClicked(event : ActionEvent) : void
- user
User
- event
Event
- eventList
EventList
- c
<<interface>>
Connection
- items
*
{ordered}

Figure 3.5: Home class diagram

d, Dashboard class diagram

The dashboard package is connected to Homcontroller throw EventItemController(User user) function. This contain list of event that is public event or private invited event. The invited events have ‘Accept‘ button, when user click this button, the btnAcceptPressed() function will be start and database will be updated. The other events which are not invited have ‘Request‘ button, when user click this button, the btnRequestPressed() function will be start and database will be uodate. For public event, when event are requested, user will be able to participate immediately without waiting for approval.

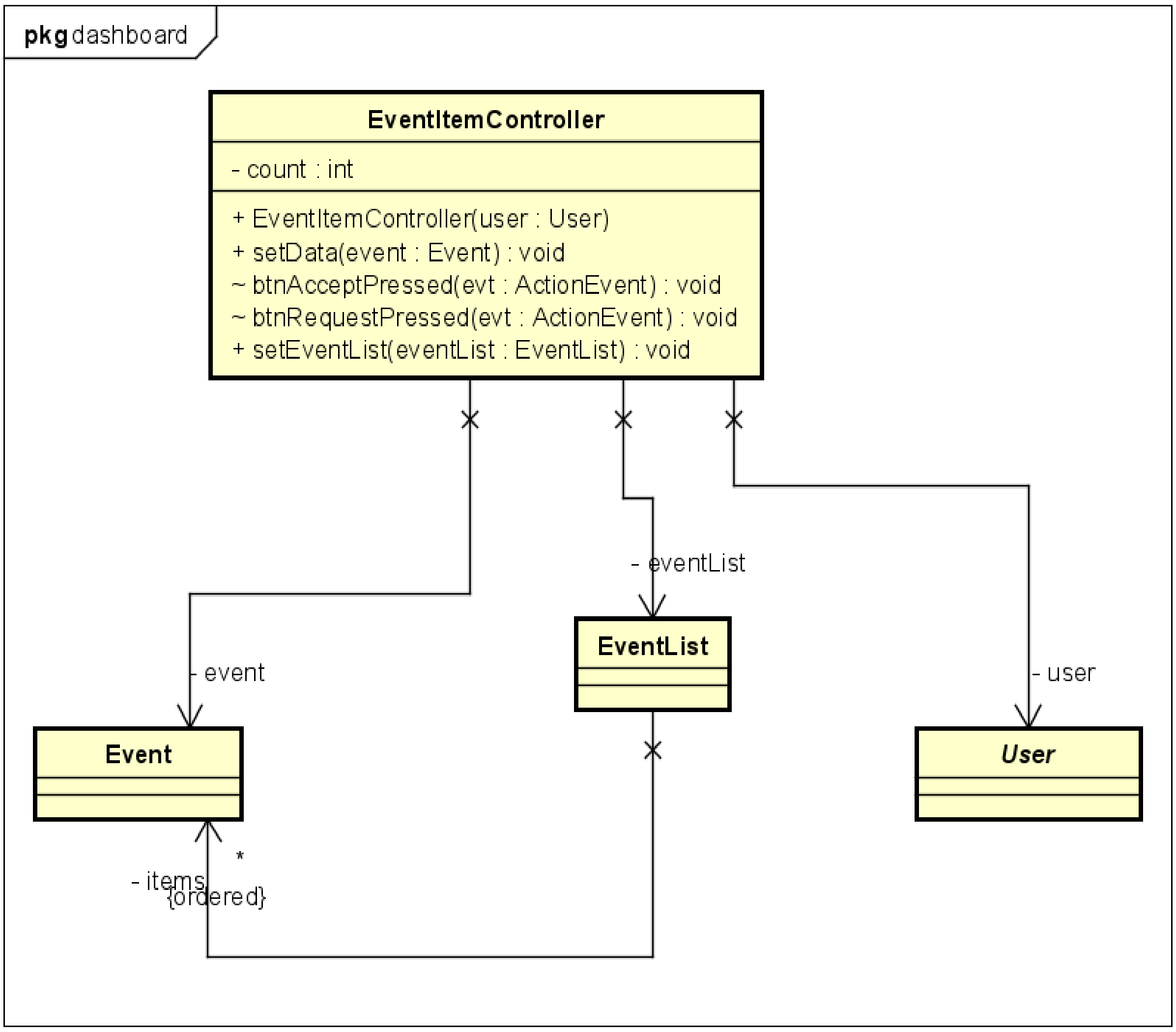

pkg dashboard
EventItemController
- count : int
+ EventItemController(user : User)
+ setData(event : Event) : void
~ btnAcceptPressed(evt : ActionEvent) : void
~ btnRequestPressed(evt : ActionEvent) : void
+ setEventList(eventList : EventList) : void
- eventList
EventList
- event
Event
- user
User
- items
*
{ordered}

Figure 3.6: Dashboard class diagram

e, UserEvent class diagram

The user package is connected to Homcontroller throw UserItemController(User user) function. This contain list of event which are joined and exits in user_event table in database. Each event has two button: ‘Remove‘ button and ‘Invite Friend‘ button The ‘Remove‘ button will delete the event from user_event table and update other tables. The ‘Invite Friend‘ request to enter the fiend’s name, if friend’s name exist, the invitation will be sent, else return error message.

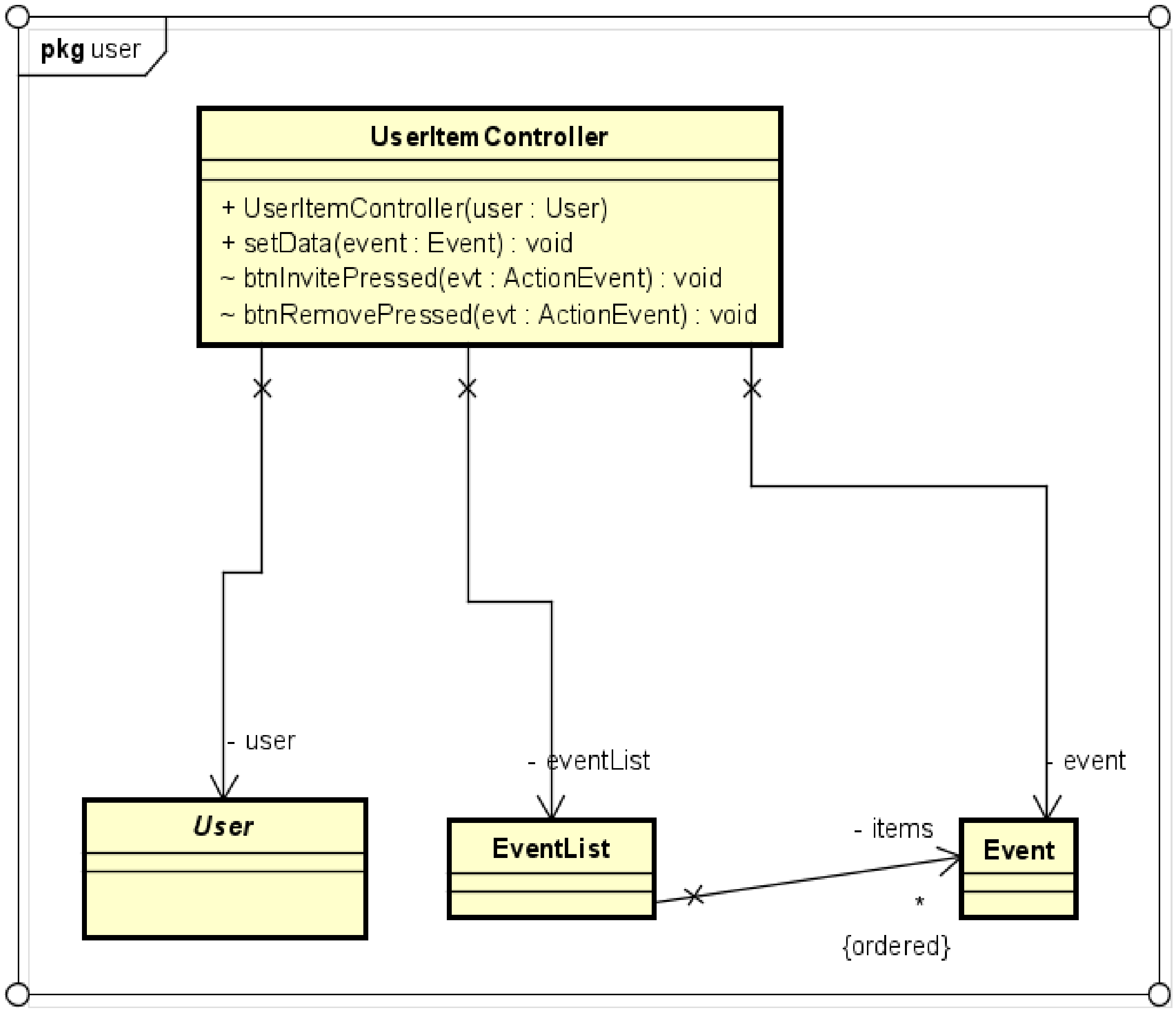

pkg user
UserItem Controller
+ UserItemController(user : User)
+ setData(event : Event) : void
~ btnInvitePressed(evt : ActionEvent) : void
~ btnRemovePressed(evt : ActionEvent) : void
- user
- eventList
- event
User
EventList
- items
Event
*
{ordered}

Figure 3.7: UserEvent class diagram

f, ManagerEvent class diagram

The manager package has two class is EventItemController and InfoController and is connected to Homcontroller throw EventItemController(User user) function. The EventItemController contains a list of event which is hosted by user, each event has a list of participants. In the EventItem scence, there are three button includes: ‘Edit‘ button, ‘Update‘ button and ‘Invite‘ button.The first,‘Invite‘ button is the same as ‘Invite Friend‘ button in the UserEventController class. The second, ‘Update’ button is used to update user whose is approved or not to participate in the event. The last, ’Edit’ button is connect to InfoController that is change the event’s information and the changed field will be updated to the database.

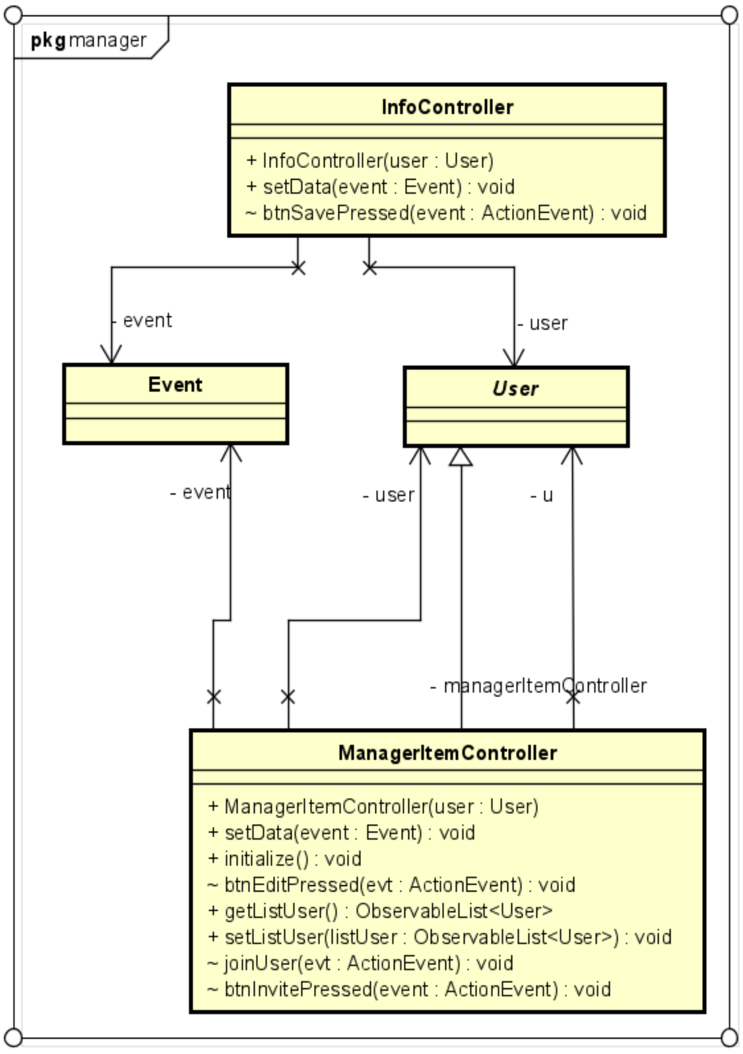

pkg manager
InfoController
+ InfoController(user : User)
+ setData(event : Event) : void
~ btnSavePressed(event : ActionEvent) : void
- event
- user
Event
User
- event
- user
- u
- managerItemController
ManagerItemController
+ ManagerItemController(user : User)
+ setData(event : Event) : void
+ initialize() : void
~ btnEditPressed(evt : ActionEvent) : void
+ getListUser() : ObservableList<User>
+ setListUser(listUser : ObservableList<User>) : void
~ joinUser(evt : ActionEvent) : void
~ btnInvitePressed(event : ActionEvent) : void

Figure 3.8: ManagerEvent class diagram

g, Account class diagram

The account package is connected to Homcontroller throw AccountController(User user) function. This function will display user information on the screen. The AccountController class has ‘Edit’ button , when user clicks this button the change field apprear. User can change thier information such as Emai, PhoneNumber, Age and then submit, the changed data will be updated in database.

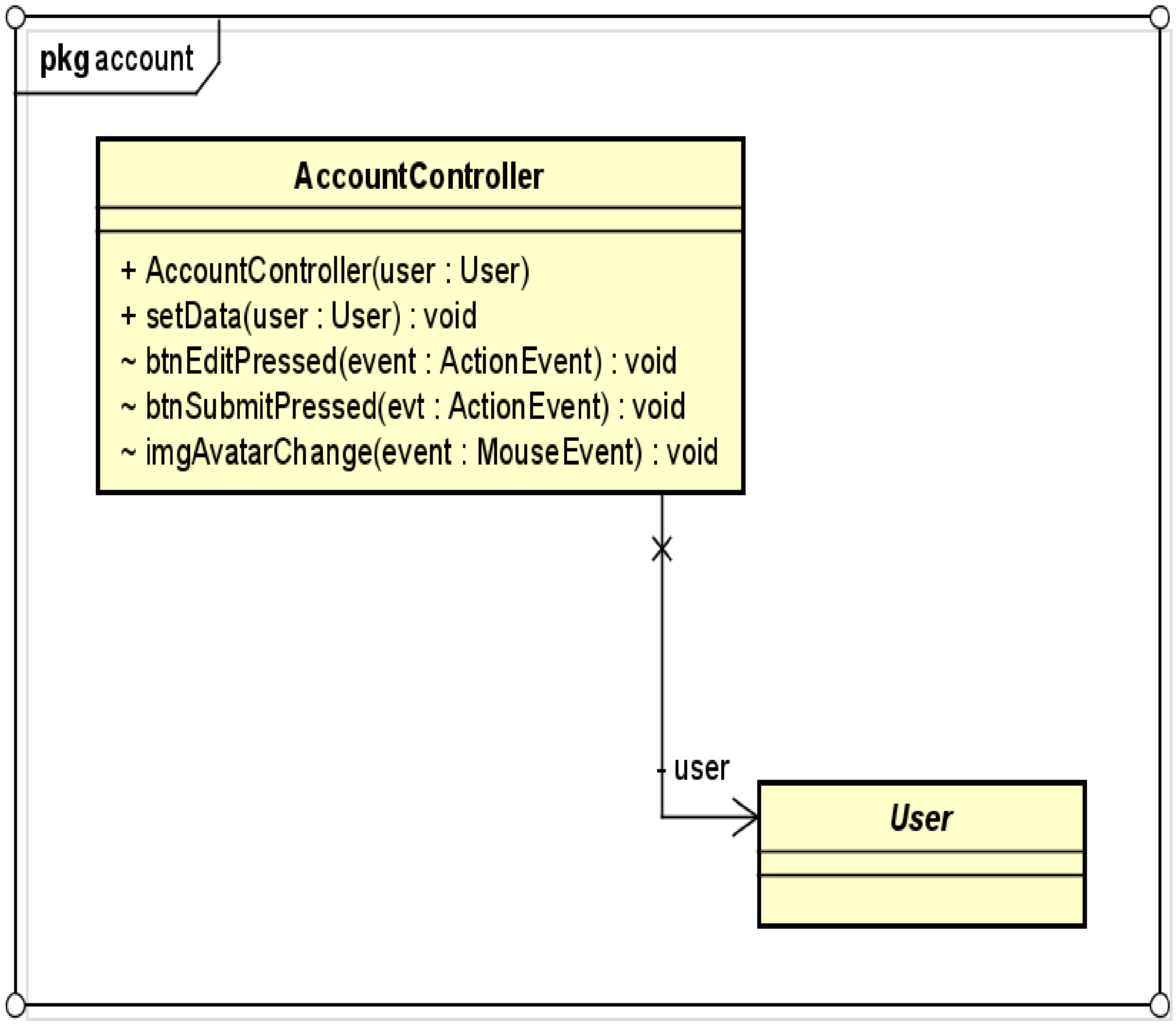

pkg account
AccountController
+ AccountController(user : User)
+ setData(user : User) : void
~ btnEditPressed(event : ActionEvent) : void
~ btnSubmitPressed(evt : ActionEvent) : void
~ imgAvatarChange(event : MouseEvent) : void
- user
User

Figure 3.9: Account class diagram

h, CreateEvent class diagram

The create package is connected to Homcontroller throw CreateController(User user) function. User has to fill in all information to create a new event. New event will be stored in database and the user is the host of the event.

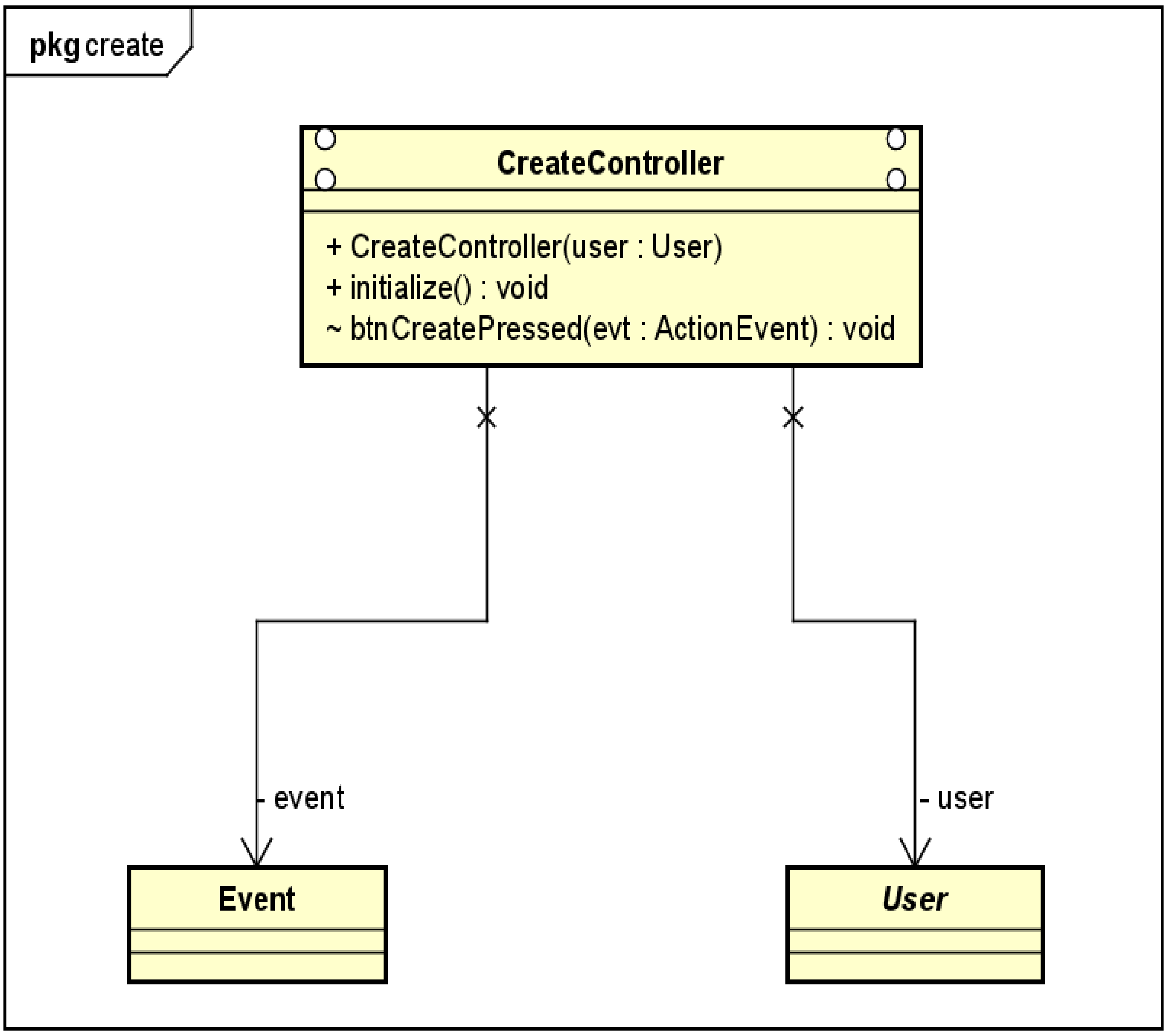

pkg create
CreateController
+ CreateController(user : User)
+ initialize() : void
~ btnCreatePressed(evt : ActionEvent) : void
- event
- user
Event
User

Figure 3.10: CreateEvent class diagram

3.1.3 Sequence diagram

This is the sequence diagram of the appication.

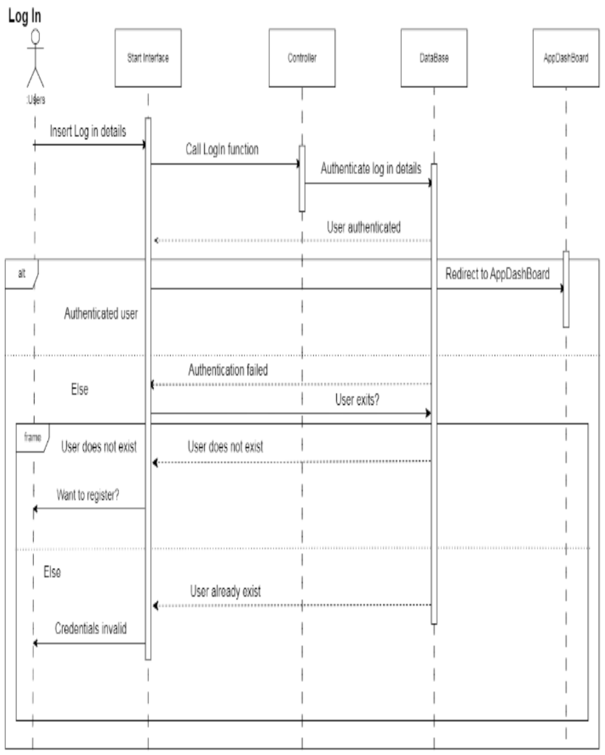


Figure 3.11: Login sequence diagram

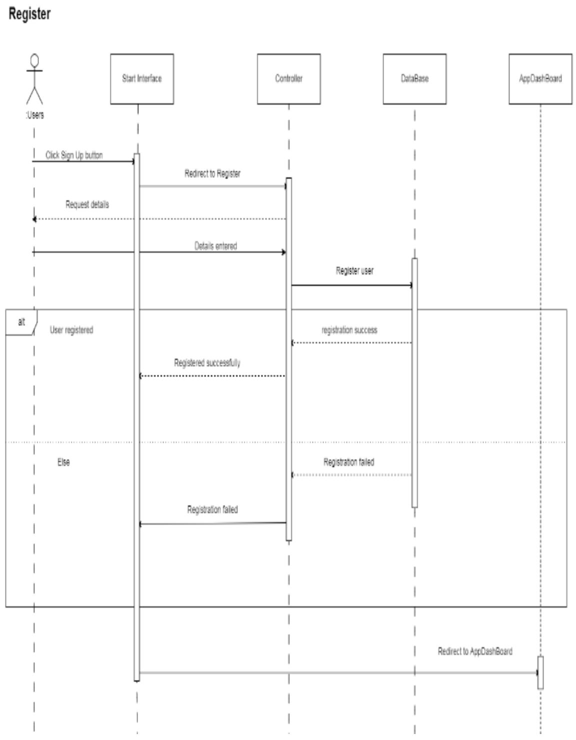


Figure 3.12: Register sequence diagram

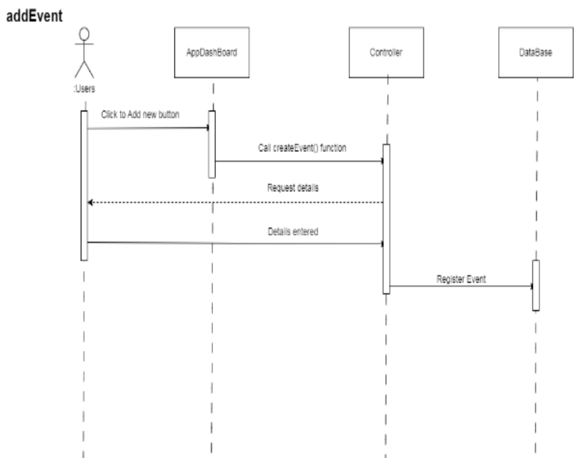


Figure 3.13: Create event sequence diagram

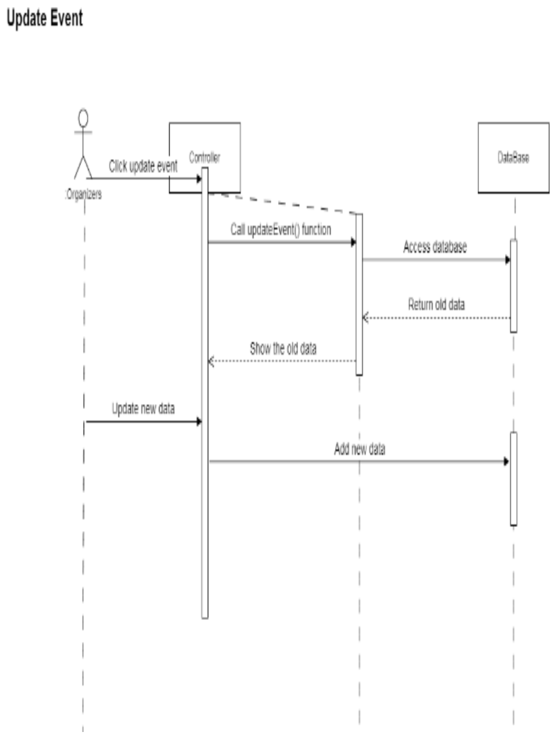


Figure 3.14: Update event sequence diagram

3.2 Database design

The database contains six tables with two main table Event, User. The User table stores name, age, phone, email which are encrypted in Java program. Only password use hash function instead. The Event table stores host-id, private-event, description, start-time, end-time, location, link, min number attender, max number attender, and number of organizers. If private-event = 1, the description to number of organizers are encrypted. Next, the user-event table is created for listing the event that user join, following the host-event is created for listing the event that user own. Request-user is the list of users request to join the event and invited-user is the list of users are invited.

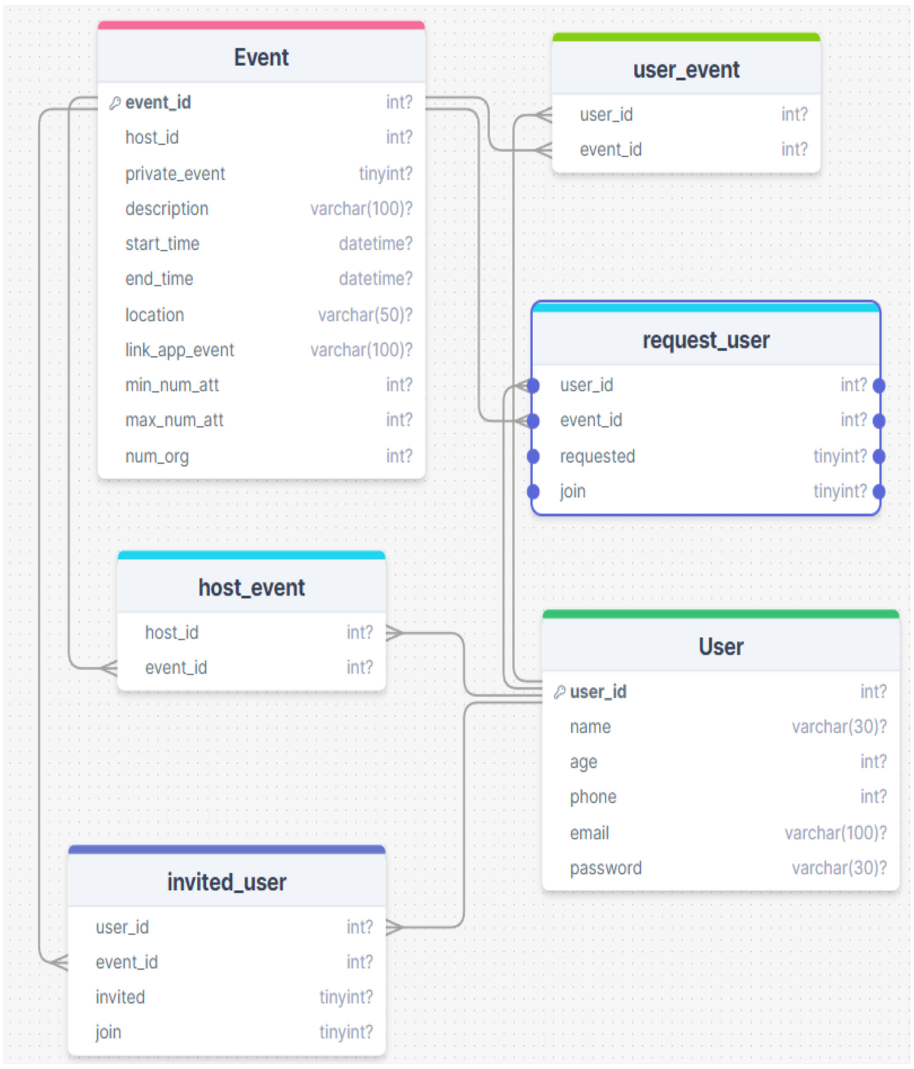


Figure 3.15: Database diagram

CHAPTER 4. DESIGN ANALYSIS

4.1 Packages analysis

4.1.1 *database* package

First of all, in our database package, the *dbConnection.java* class creates the new user and help user take the information in database. The *ConnectionUtil.java* class help us manage the user table and user-event table in database, for example the requestUser method or invitedUser method....

4.1.2 *main* package

The *main* package contains 3 java files and icons for UI in sub package name *icon*. The *Event* class and *EventList* class help us initialize Event variable, add and remove Event variable in ArrayList items and lastly, the *User* class helps us take the information of user and most importantly helps us check the SHA-256 password when user login.

4.1.3 *encryption* package

This is the package where we store our cryptography method. We are using AES encryption for the private information like user information or private event, and SHA-256 to store the password user.

4.1.4 *client* package

This is the package contains the controller and UI of our app. We create sub package for each work: account, create event, register, login... Each sub package contain their controller and their UI interaction.

4.2 UI analysis

We coding our UI by using JavaFX, help us building the UI faster with screen Builder app. We also change the color button in the left screen when you click the button.

5.1 Functional testing

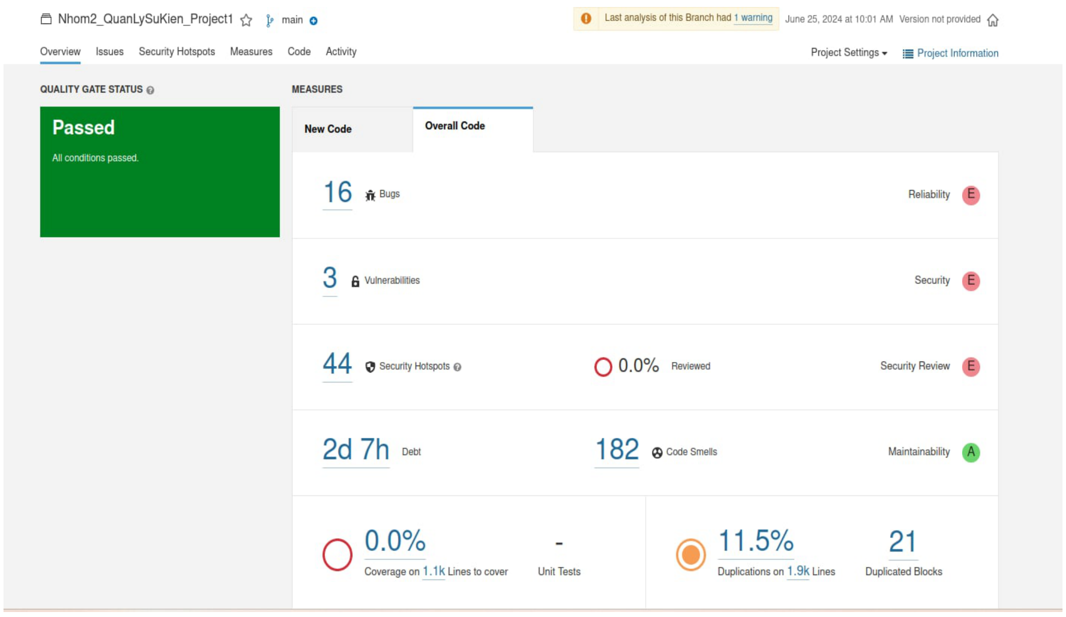
The functionalities include login, sign-up, creating new events, sending invitations, accepting invitations, sending requests, and accepting requests have been testing while coding to ensure the application runs correctly.

5.2 Stability testing

Stability testing shows that the application can operate smoothly and consistently under different settings, ensuring a seamless user experience. However, there may have some interruption because of database server delay.

5.3 Testing with Sonarqube

After testing with Sonarqube, we found that our application has 16 bugs, 3 vulnerabilities, and 44 security hotspots.


Nhom2_QuanLySuKien_Project1 main

Last analysis of this Branch had 1 warning June 25, 2024 at 10:01 AM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code Overall Code

16 Bugs Reliability E

3 Vulnerabilities Security E

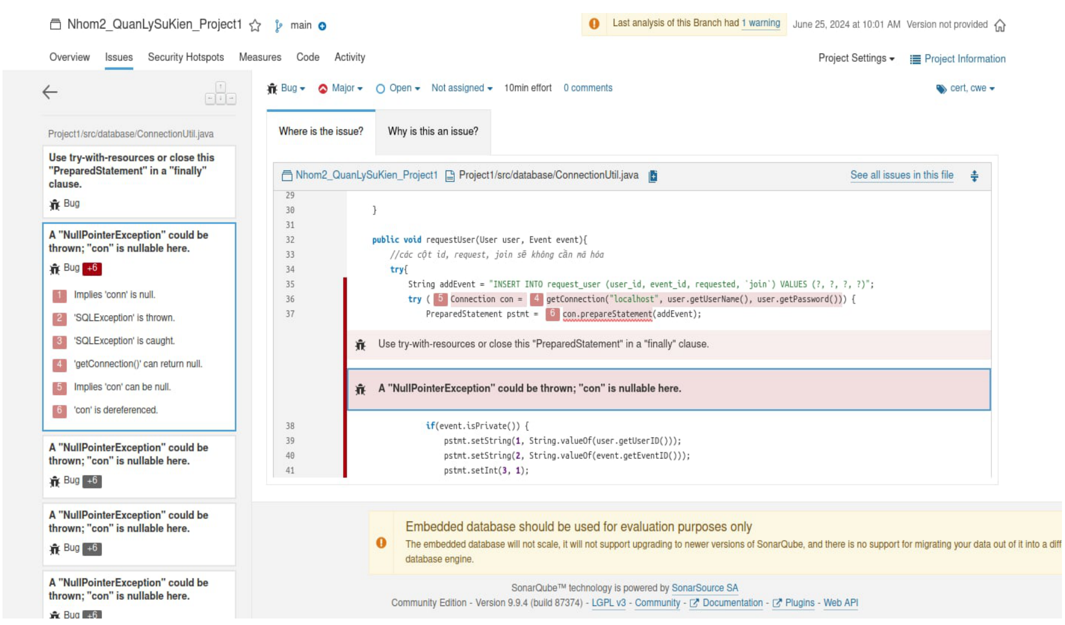
44 Security Hotspots 0.0% Reviewed Security Review E

2d 7h Debt 182 Code Smells Maintainability A

0.0% Coverage on 1.1k Lines to cover - Unit Tests 11.5% Duplications on 1.9k Lines 21 Duplicated Blocks

Figure 5.1: Testing with Sonarqube

We evaluate the errors happen mostly because the con connection can be null. We can fix that by adding try/catch null exception.



Project1/src/database/ConnectionUtil.java

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Bug

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

1 Implies 'conn' is null.
2 'SQLException' is thrown.
3 'SQLException' is caught.
4 'getConnection()' can return null.
5 Implies 'con' can be null.
6 'con' is dereferenced.

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

Nhom2_QuanLySuKien_Project1 Project1/src/database/ConnectionUtil.java

```
    }

    public void requestUser(User user, Event event){
        //các cột id, request, join sẽ không cần mã hóa
        try{
            String addEvent = "INSERT INTO request_user (user_id, event_id, requested, `join`) VALUES (?, ?, ?, ?)";
            try ( 5 Connection con = 4 getConnection("localhost", user.getUserName(), user.getPassword())) {
                PreparedStatement pstmt = 6 con.prepareStatement(addEvent);
```

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.

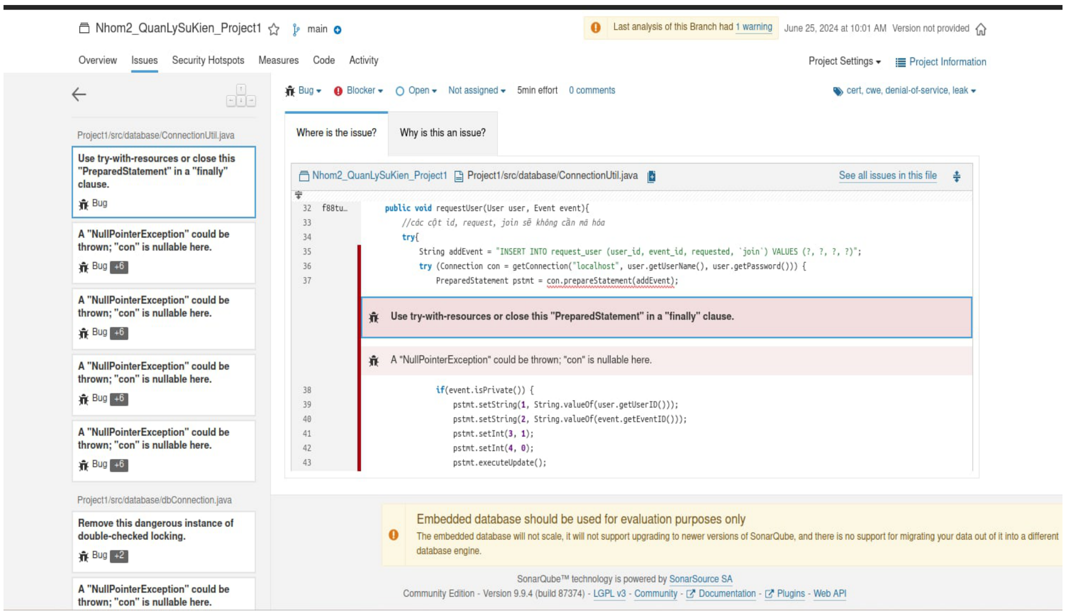
A "NullPointerException" could be thrown; "con" is nullable here.

```
                if(event.isPrivate()) {
                    pstmt.setString(1, String.valueOf(user.getUserID()));
                    pstmt.setString(2, String.valueOf(event.getEventID()));
                    pstmt.setInt(3, 1);
```

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a diff
database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9.4 (build 87374) - LGPL v3 - Community - Documentation - Plugins - Web API

Figure 5.2: Testing with Sonarqube



Project1/src/database/ConnectionUtil.java

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Bug

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

Project1/src/database/dbConnection.java

Remove this dangerous instance of double-checked locking.
Bug +2

A "NullPointerException" could be thrown; "con" is nullable here.

Nhom2_QuanLySuKien_Project1 Project1/src/database/ConnectionUtil.java

```
f88tu…  public void requestUser(User user, Event event){
            //các cột id, request, join sẽ không cần mã hóa
            try{
                String addEvent = "INSERT INTO request_user (user_id, event_id, requested, `join`) VALUES (?, ?, ?, ?)";
                try (Connection con = getConnection("localhost", user.getUserName(), user.getPassword())) {
                    PreparedStatement pstmt = con.prepareStatement(addEvent);
```

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.

A "NullPointerException" could be thrown; "con" is nullable here.

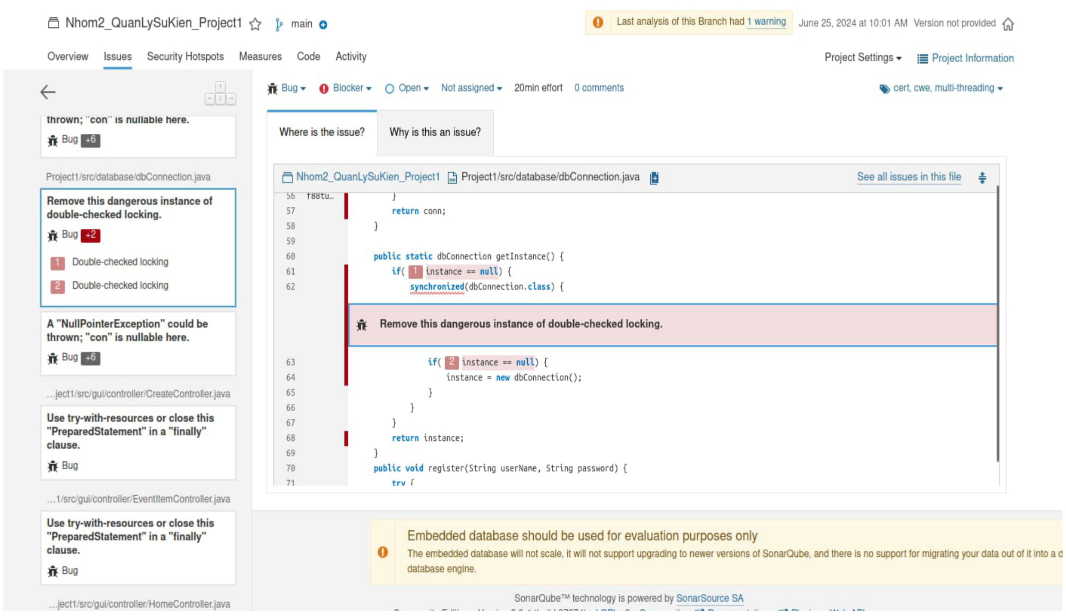
```
                    if(event.isPrivate()) {
                        pstmt.setString(1, String.valueOf(user.getUserID()));
                        pstmt.setString(2, String.valueOf(event.getEventID()));
                        pstmt.setInt(3, 1);
                        pstmt.setInt(4, 0);
                        pstmt.executeUpdate();
```

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9.4 (build 87374) - LGPL v3 - Community - Documentation - Plugins - Web API

Figure 5.3: Testing with Sonarqube

There is the double-checked locking error, this may not work as expected and can lead to potential bugs. We don't have the solution for this problem yet.



Last analysis of this Branch had 1 warning June 25, 2024 at 10:01 AM Version not provided

thrown; "con" is nullable here.
Bug +6

Project1/src/database/dbConnection.java

Remove this dangerous instance of double-checked locking.
Bug +2
1 Double-checked locking
2 Double-checked locking

A "NullPointerException" could be thrown; "con" is nullable here.
Bug +6

...ject1/src/gui/controller/CreateController.java

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Bug

...1/src/gui/controller/EventItemController.java

Use try-with-resources or close this "PreparedStatement" in a "finally" clause.
Bug

...ject1/src/gui/controller/HomeController.java

Nhom2_QuanLySuKien_Project1 Project1/src/database/dbConnection.java

```
        }
        return conn;
    }

    public static dbConnection getInstance() {
        if( 1 instance == null) {
            synchronized(dbConnection.class) {
```

Remove this dangerous instance of double-checked locking.

```
                if( 2 instance == null) {
                    instance = new dbConnection();
                }
            }
        }
        return instance;
    }
    public void register(String userName, String password) {
        try {
```

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a d database engine.

SonarQube™ technology is powered by SonarSource SA

Figure 5.4: Testing with Sonarqube

5.4 Security Problem

We are aware that there are a big problem in encryption package, the hacker can easily take the key of AES function to decrypt the sensitive information in database. This can be handle by adding the firewall anđ middle-ware between the user to database connection.

We also know that the SHA-256 is easily being hacked because normal SHA-256 is a fast function. We can switch to use bcrypt to handle that.

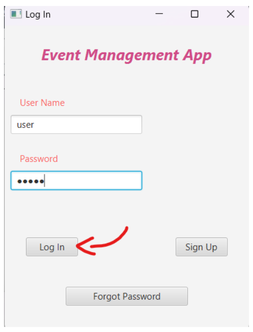
6.1 Instruction for using an app

6.1.1 Set up application

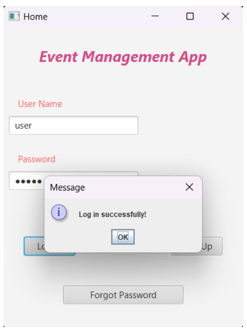
For installation the application, you can download the app from Project1 folder on the main branch of Github `https://github.com/ecsecsec/Project1_2023.2/tree/main/Project1`.

6.1.2 Login/ Sign up

Firstly, please run the test.java, then the login screen will pop up. If you password is wrong the screen will give you error message.

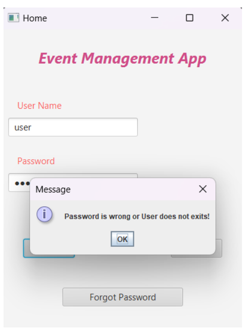


(a) Log In Screen



Log in successfully!

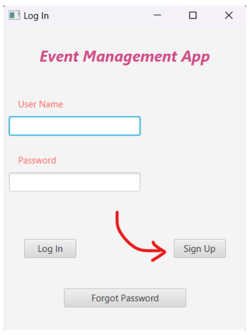
(b) Log In Successfully



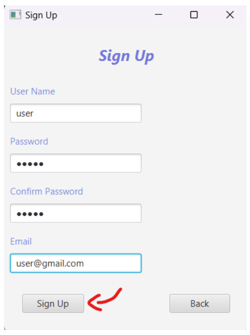
Password is wrong or User does not exits!

(c) Wrong password

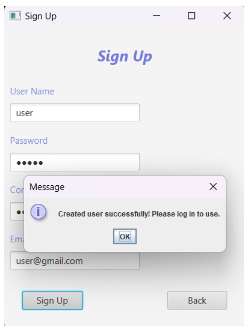
To sign up, press the button sign up, then the new screen will pop up for you to fill your information, if the username coincide the same, the sign up screen will notice that.



(a) Sign up Button

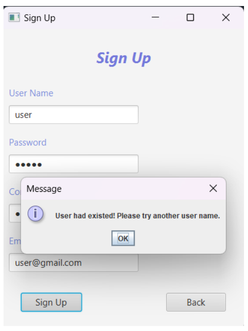


(b) Sign up Screen



Created user successfully! Please log in to use.

(c) Success

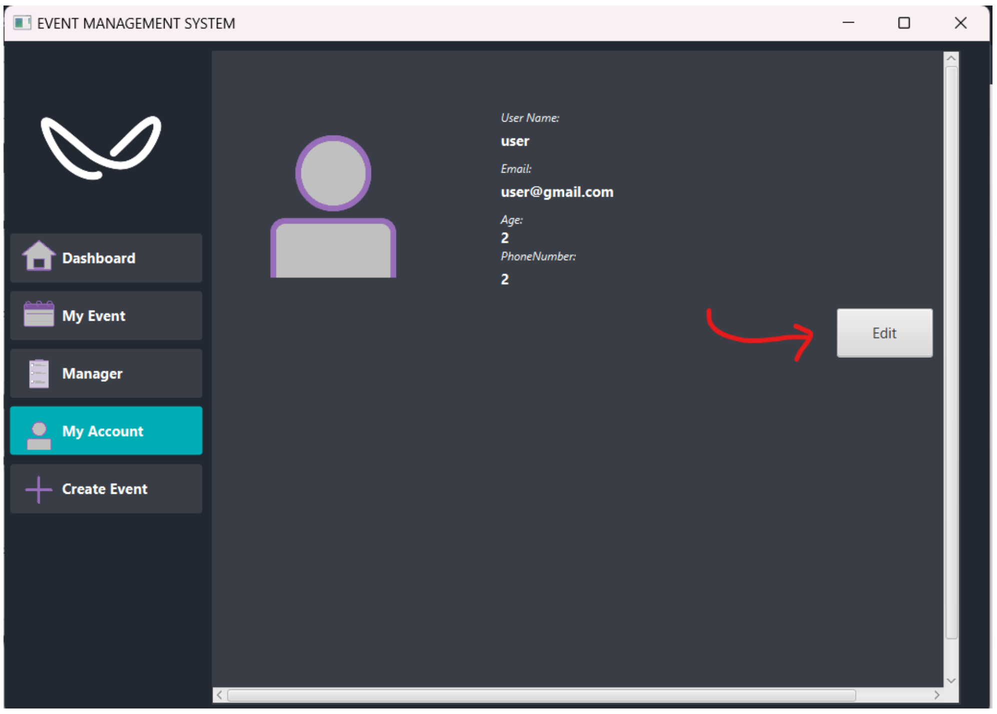


User had existed! Please try another user name.

(d) Duplicate User

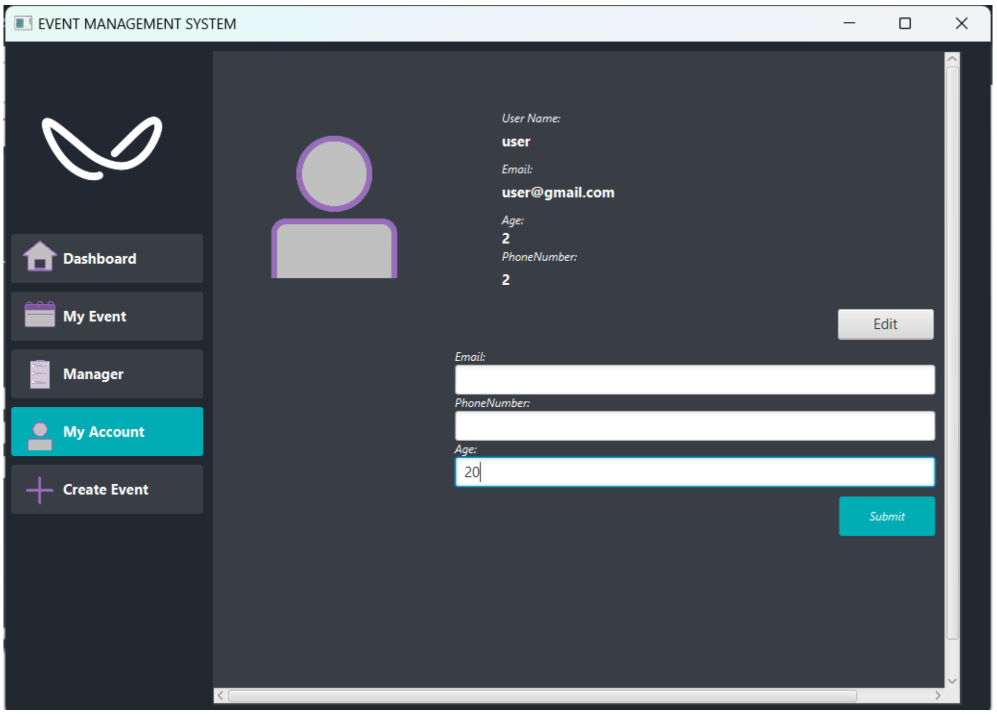
6.1.3 Edit User

If you want to edit your personal data, click the edit button in the *My Account* section (in the left edge below manager)



User Name:
user
Email:
user@gmail.com
Age:
2
PhoneNumber:
2

Figure 6.3: Edit Button



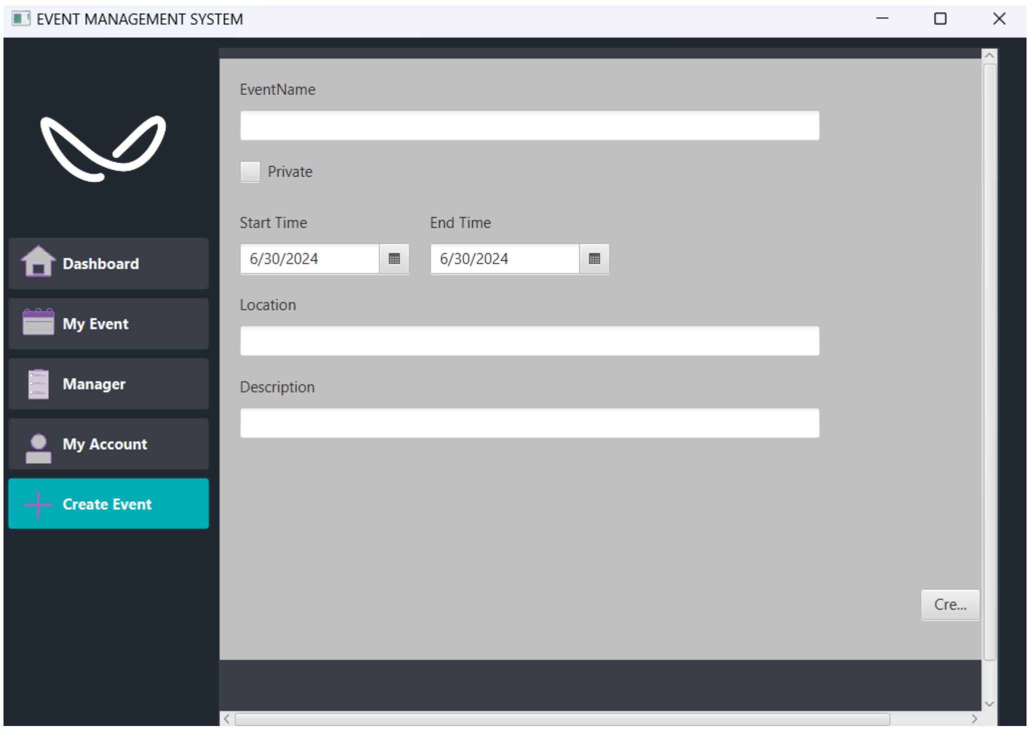
User Name:
user
Email:
user@gmail.com
Age:
2
PhoneNumber:
2

Email:
PhoneNumber:
Age:
20

Figure 6.4: Edit Data

6.1.4 Create Event

To create an event, press the button *Create Event* below *My Account*, the screen will pop up for you to fill the data. Remember to fill out all of the info.



EventName

Private

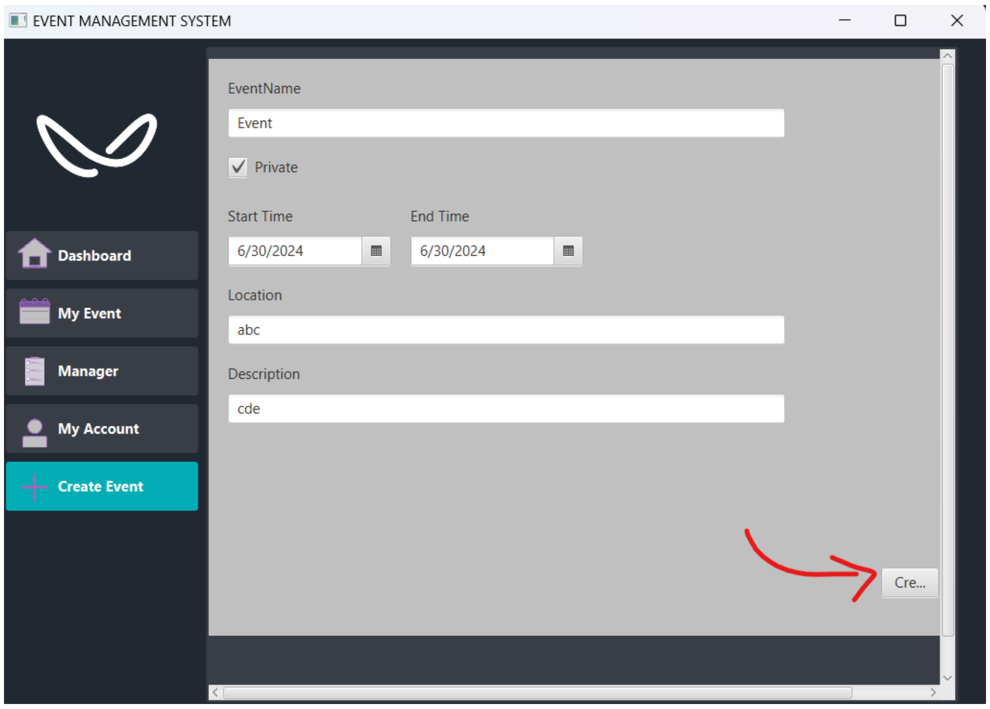
Start Time 6/30/2024

End Time 6/30/2024

Location

Description

Figure 6.5: Create an Event



EventName: Event

✓ Private

Start Time 6/30/2024

End Time 6/30/2024

Location: abc

Description: cde

Figure 6.6: Make an Event action

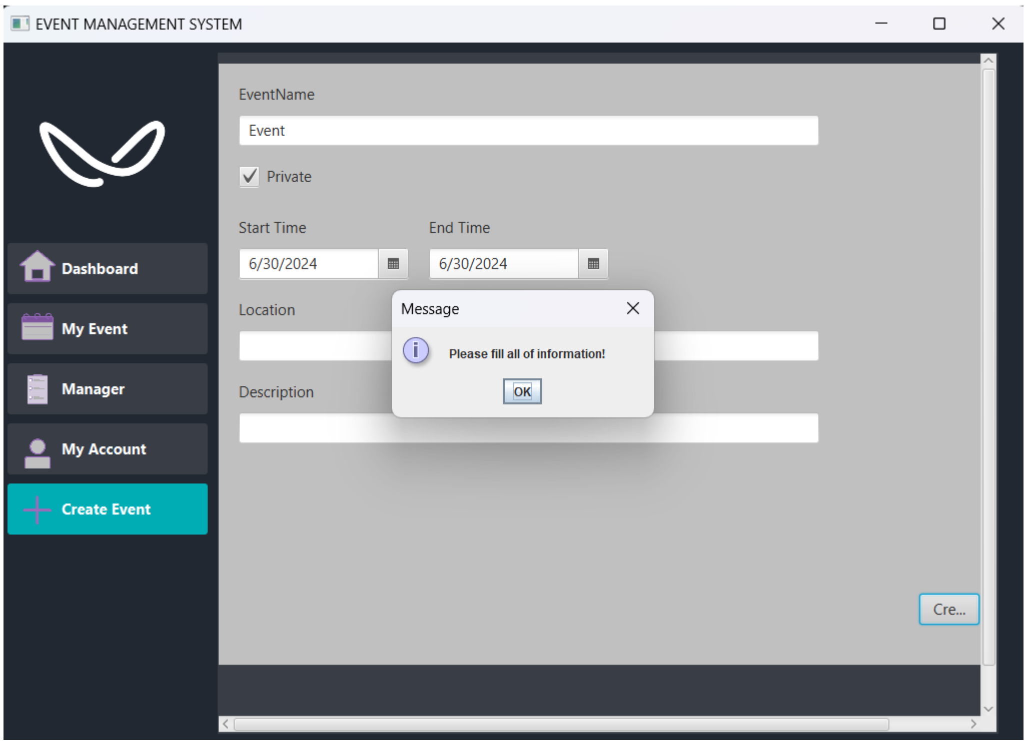


Figure 6.7: Too little information

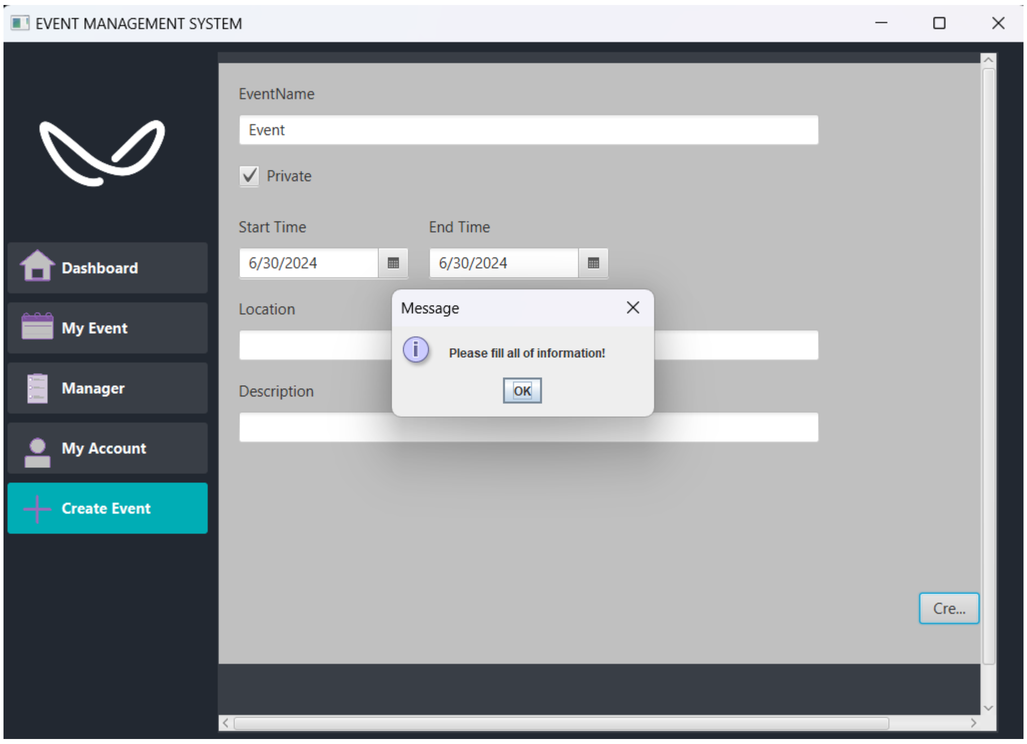
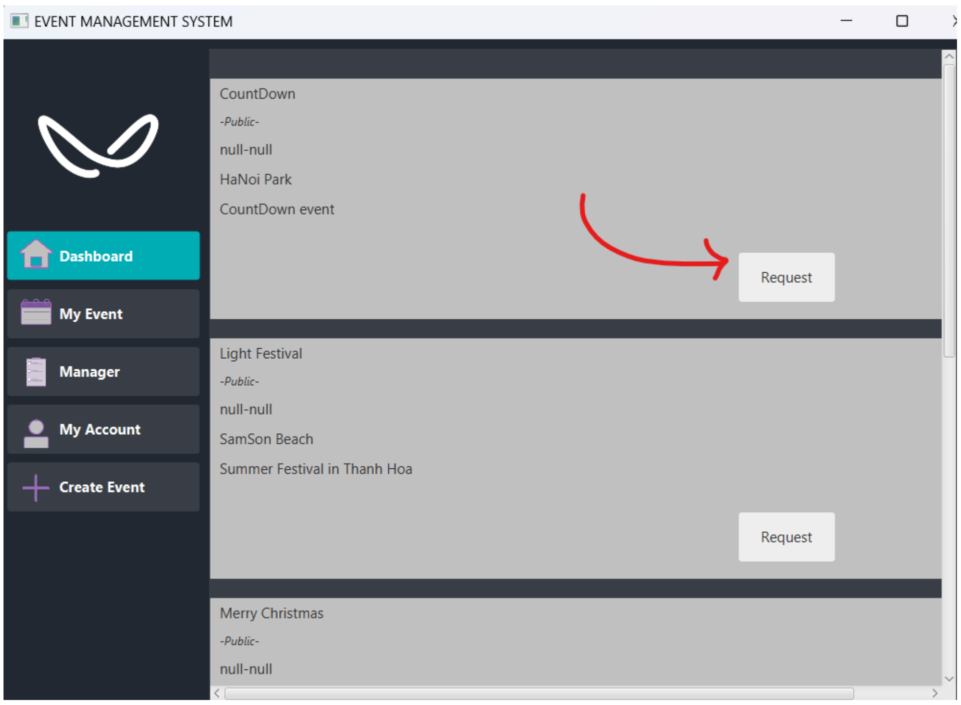


Figure 6.8: Create an Event Successfully

6.1.5 Request, Invite and Remove Event

You can request to join only the Public Event in *Dashboard*. It will automatically add this event in *My Event* section.



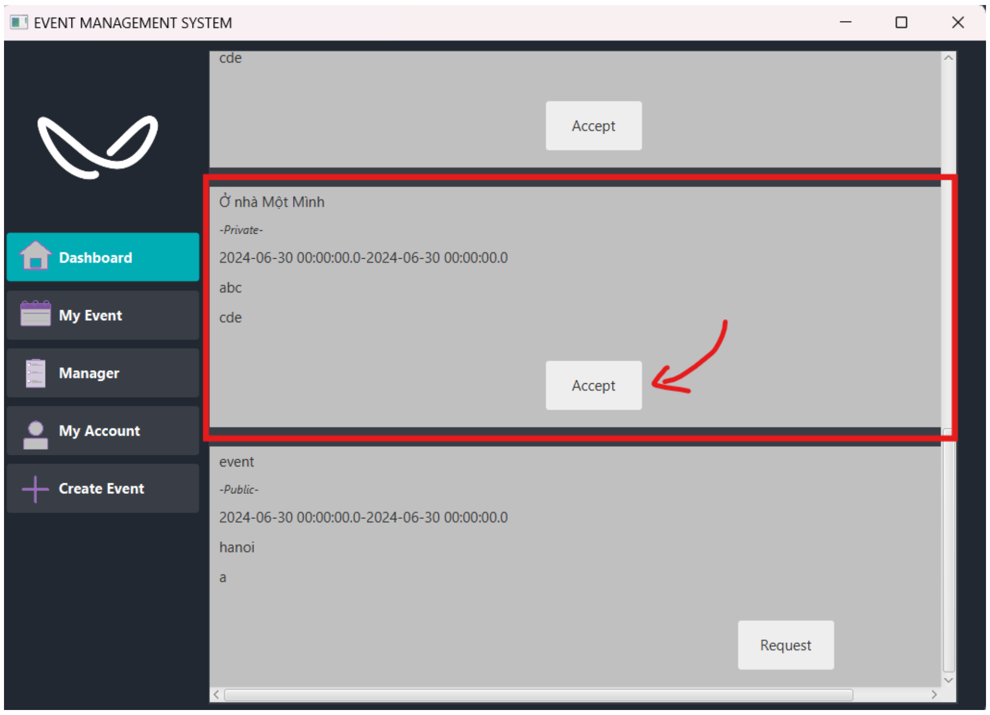
CountDown
-Public-
null-null
HaNoi Park
CountDown event

Light Festival
-Public-
null-null
SamSon Beach
Summer Festival in Thanh Hoa

Merry Christmas
-Public-
null-null

Figure 6.9: Dashboard

If you are invited by your friend or the event host, the event will pop up the Accept button instead of the Request button. You can have an invitation in both Public Event and Private Event.

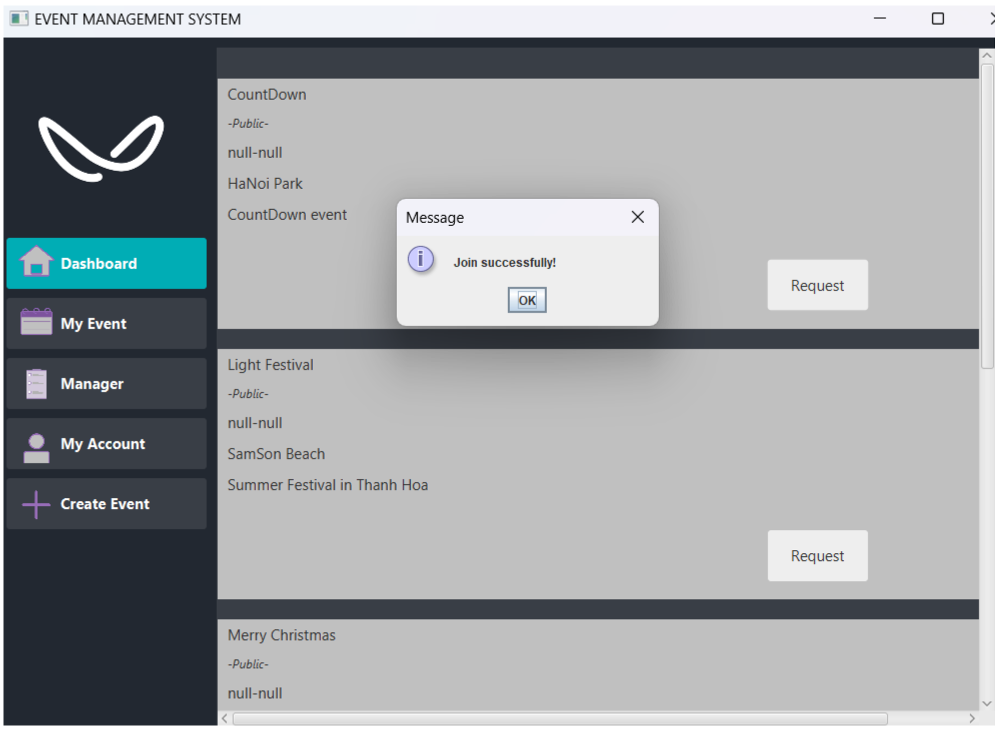


cde

Ở nhà Một Mình
-Private-
2024-06-30 00:00:00.0-2024-06-30 00:00:00.0
abc
cde

event
-Public-
2024-06-30 00:00:00.0-2024-06-30 00:00:00.0
hanoi
a

Figure 6.10: Accept Button



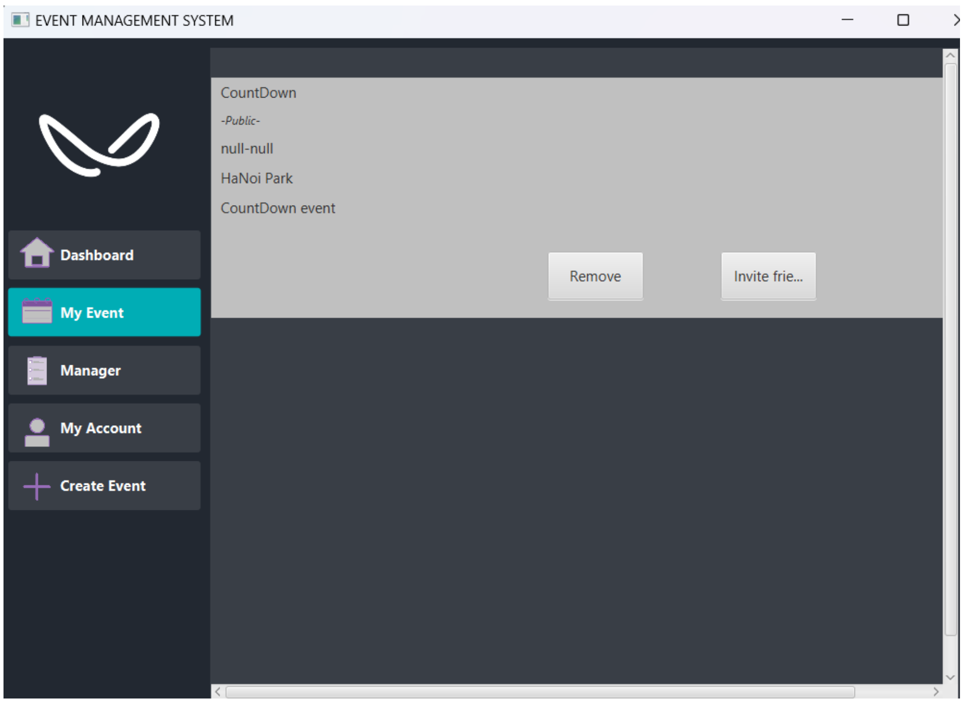
CountDown
-Public-
null-null
HaNoi Park
CountDown event

Join successfully!

Light Festival
-Public-
null-null
SamSon Beach
Summer Festival in Thanh Hoa

Merry Christmas
-Public-
null-null

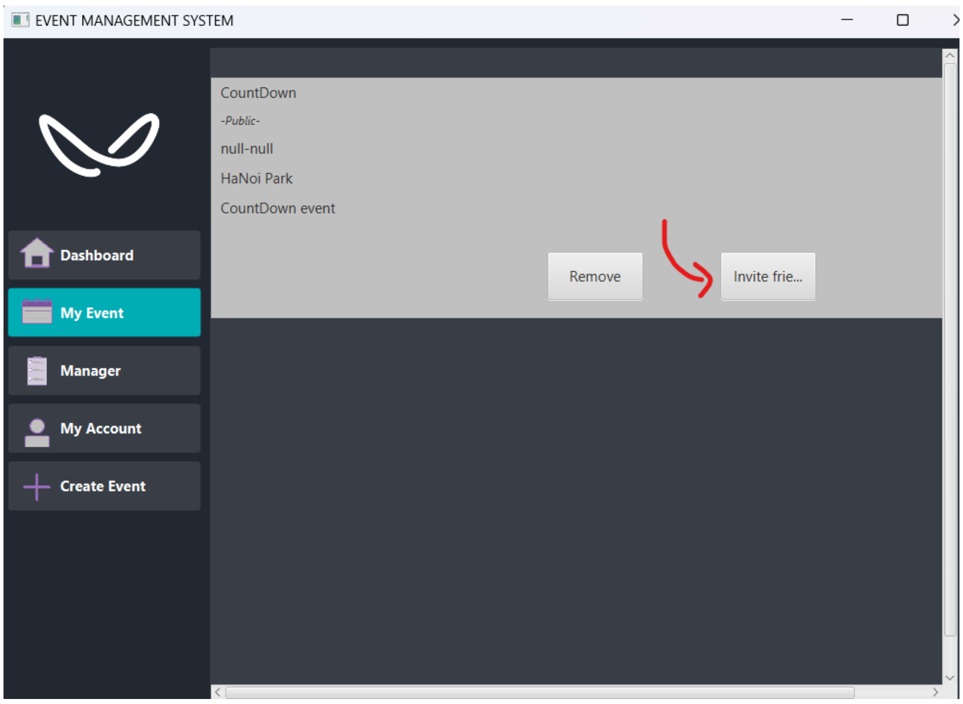
Figure 6.11: Join an Event Successfully



CountDown
-Public-
null-null
HaNoi Park
CountDown event

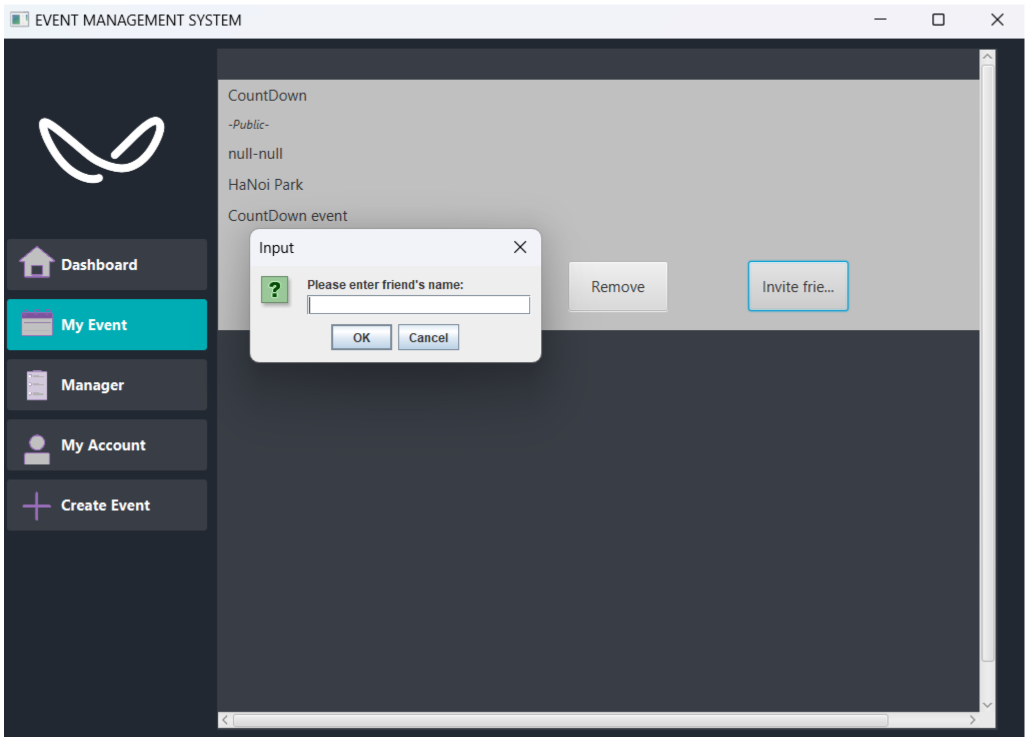
Figure 6.12: Event in My Event section

If you want invite your friend, press the Invite friend button. The input screen will pop up for you to enter your friend name. If this is a Private Event, your friend are added to request queue.



CountDown
-Public-
null-null
HaNoi Park
CountDown event

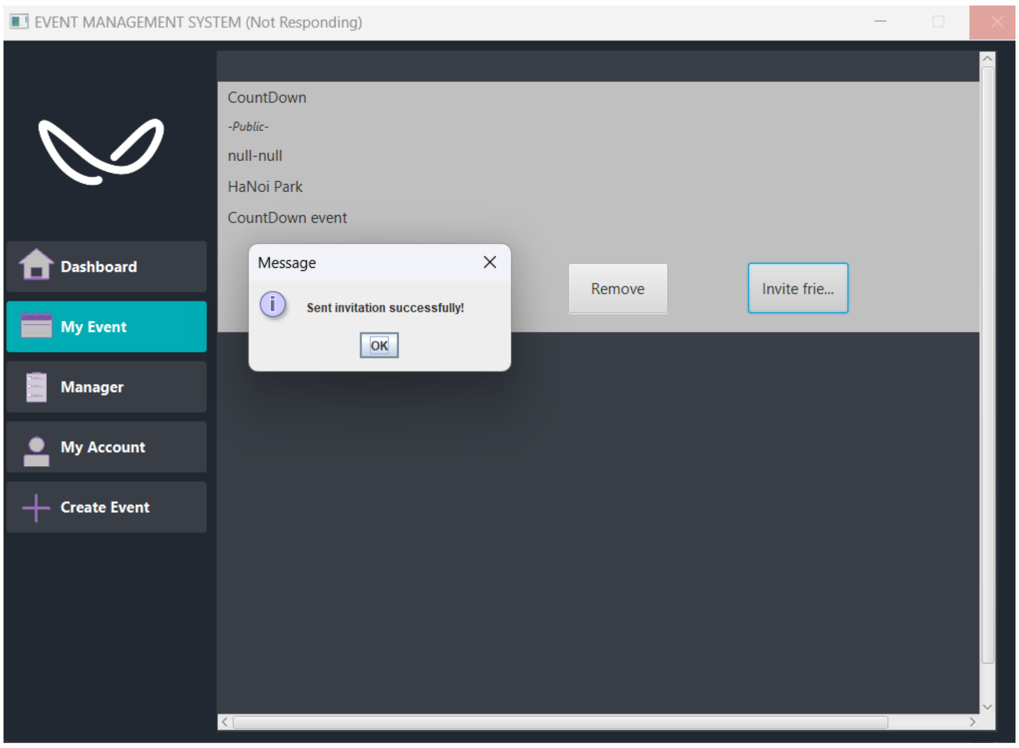
Figure 6.13: Invite Button



CountDown
-Public-
null-null
HaNoi Park
CountDown event

Please enter friend's name:

Figure 6.14: Enter your friend name

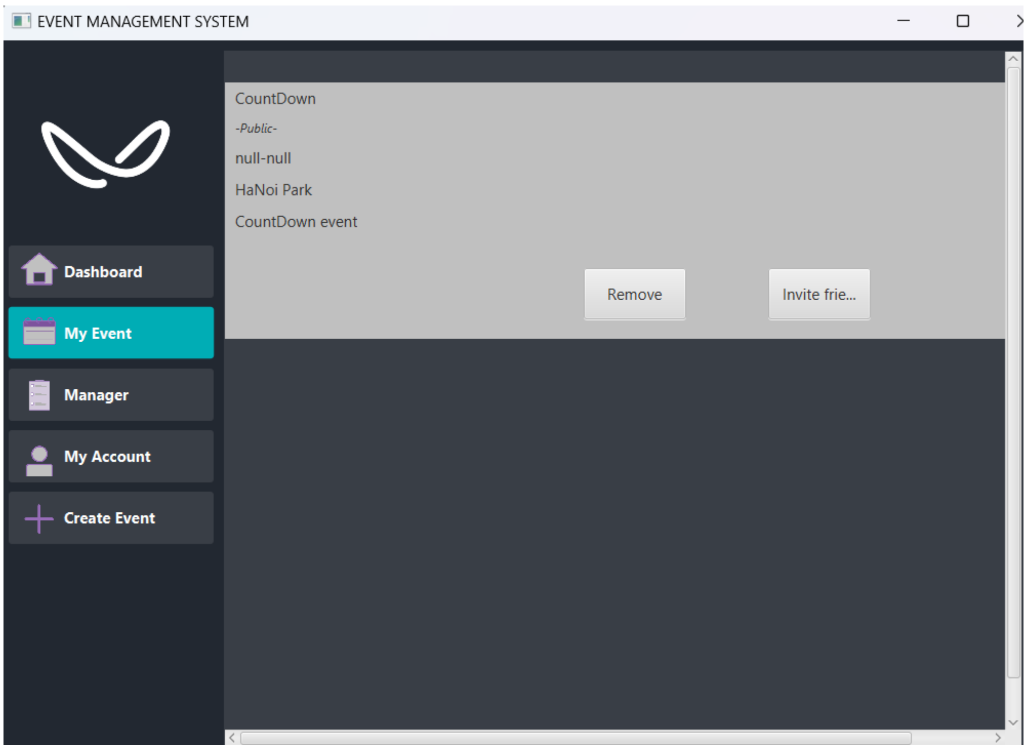


CountDown
-Public-
null-null
HaNoi Park
CountDown event

Sent invitation successfully!

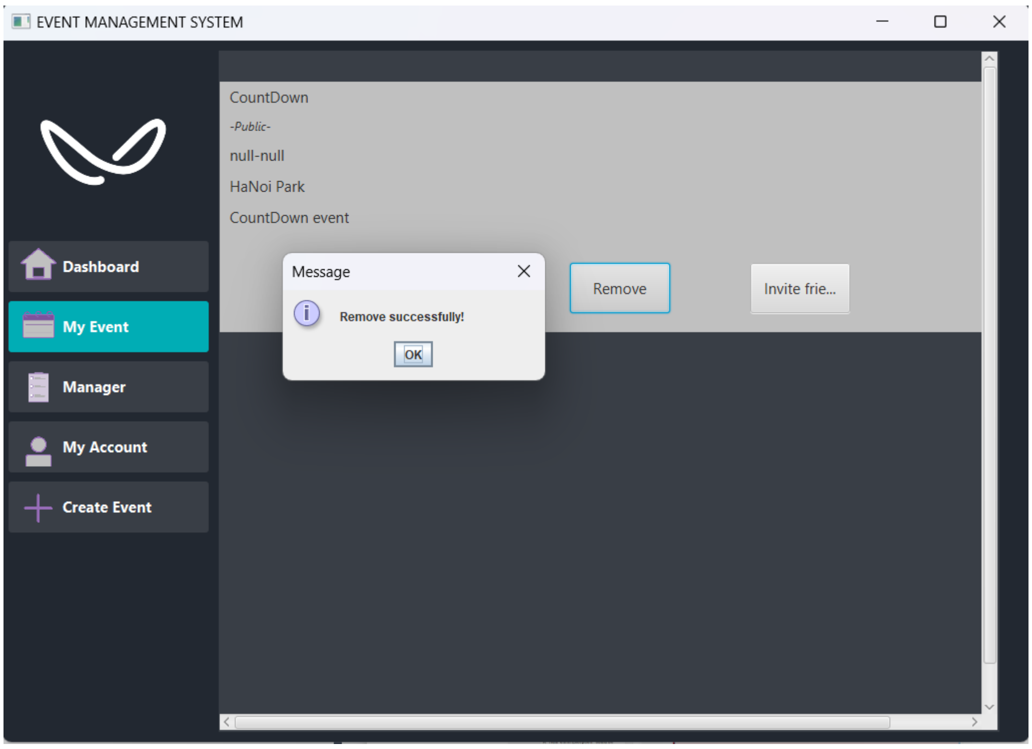
Figure 6.15: Sent Invitation Successfully

You can see that there is a remove button, click that to out your event.



CountDown
-Public-
null-null
HaNoi Park
CountDown event

Figure 6.16: Remove Button



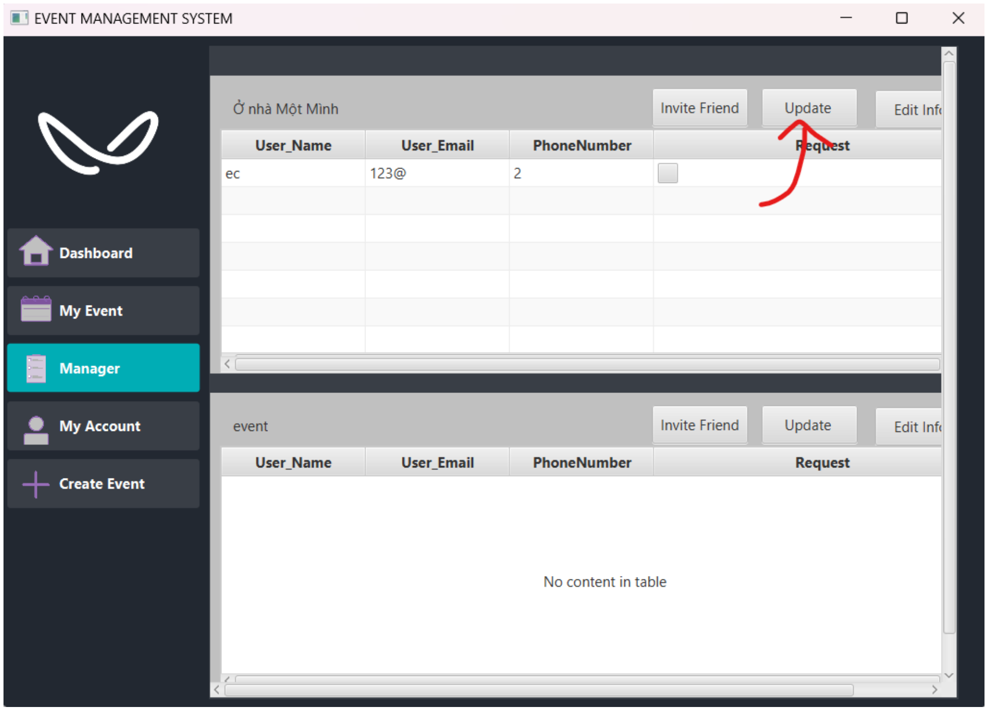
CountDown
-Public-
null-null
HaNoi Park
CountDown event

Message
Remove successfully!

Figure 6.17: Remove Successfully

6.1.6 Manager Function

Shows a list of events user hosts, each has a table containing the user's information (both participating and waiting to be approved), the update button is used to reload the list participants, check box select is approved. The invite button uses for invite other users. The edit button uses for fixing the event information.



Ở nhà Một Minh

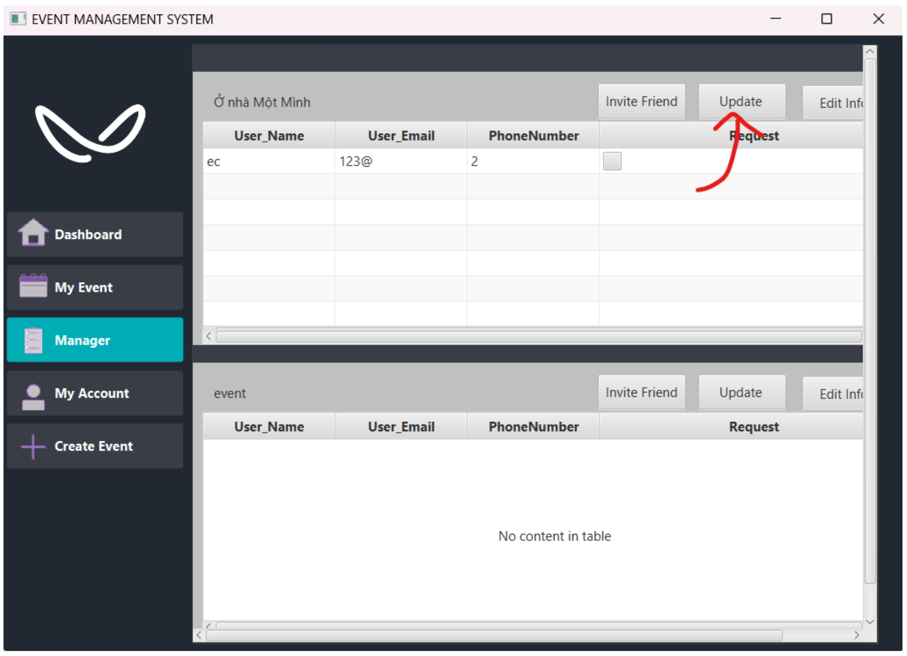
User_Name	User_Email	PhoneNumber	Request
ec	123@	2	☐

event

User_Name	User_Email	PhoneNumber	Request

No content in table

Figure 6.18: Before Update an Event



User_Name	User_Email	PhoneNumber	Request
ec	123@	2	☐

Figure 6.19: After Update an Event

Press the invite button to invite your friend as the host. The input screen will pop up to enter your friend name. The user will appear in the table after being invited successfully.

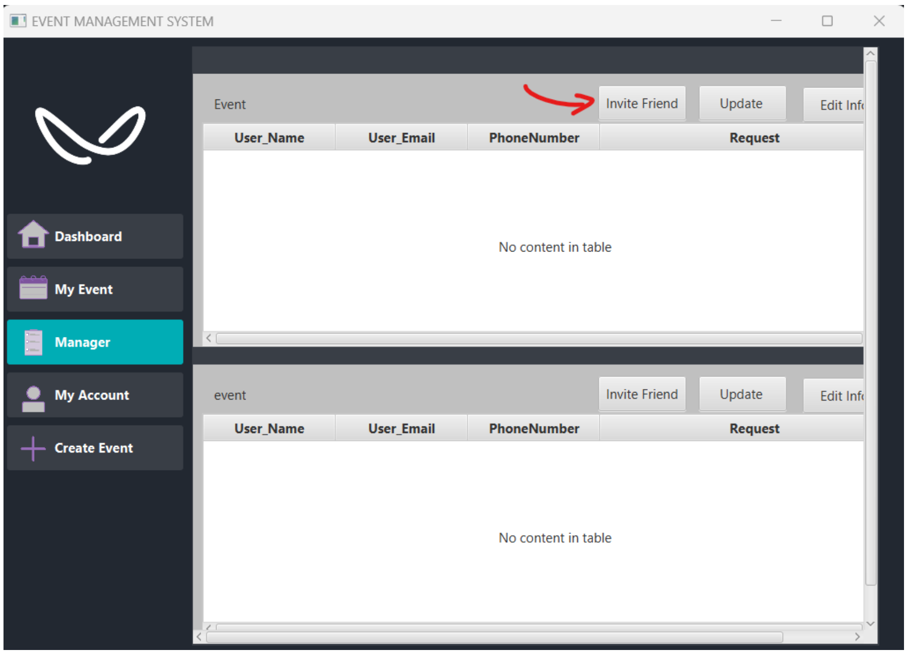


Figure 6.20: Invite Button

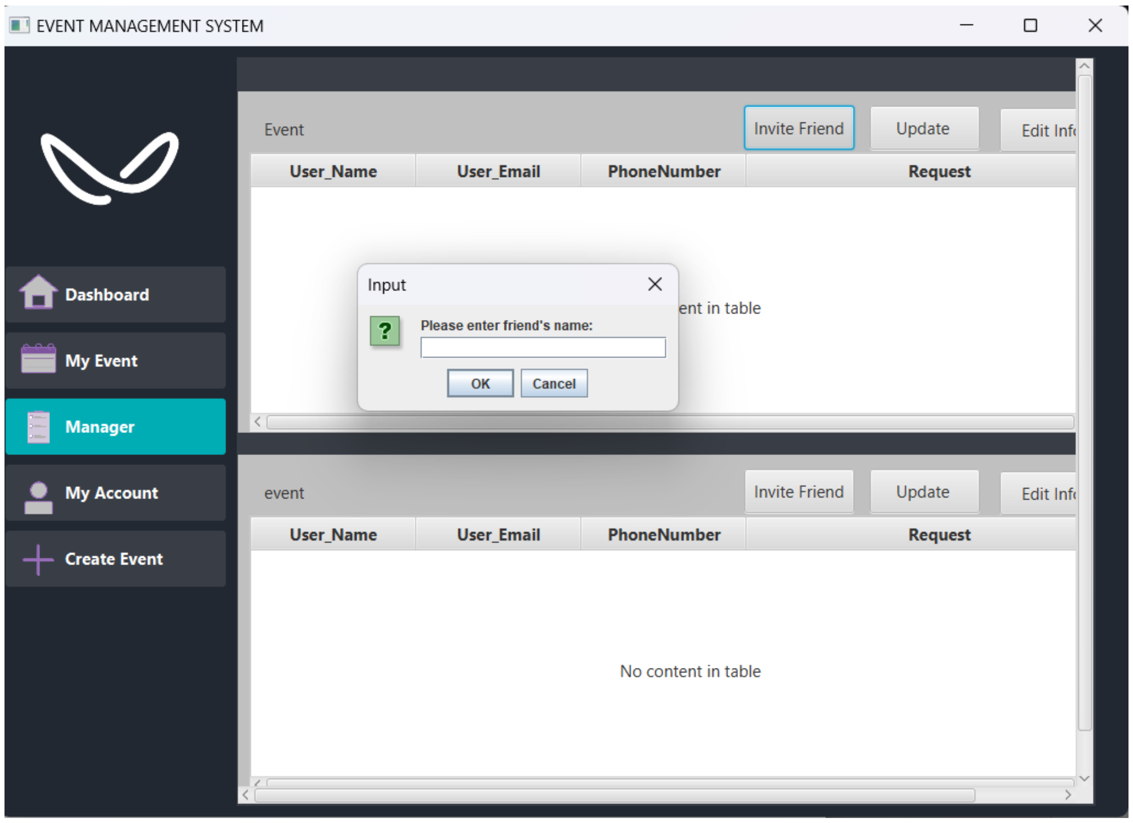


Figure 6.21: Enter User name

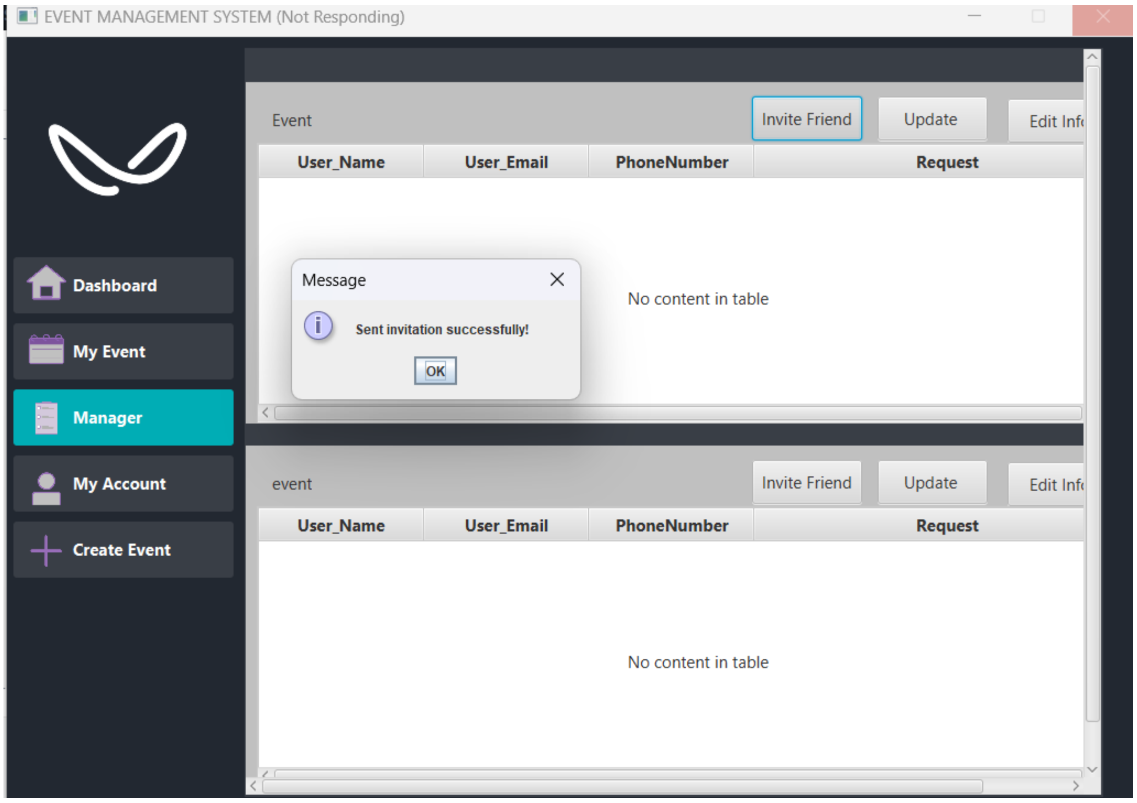
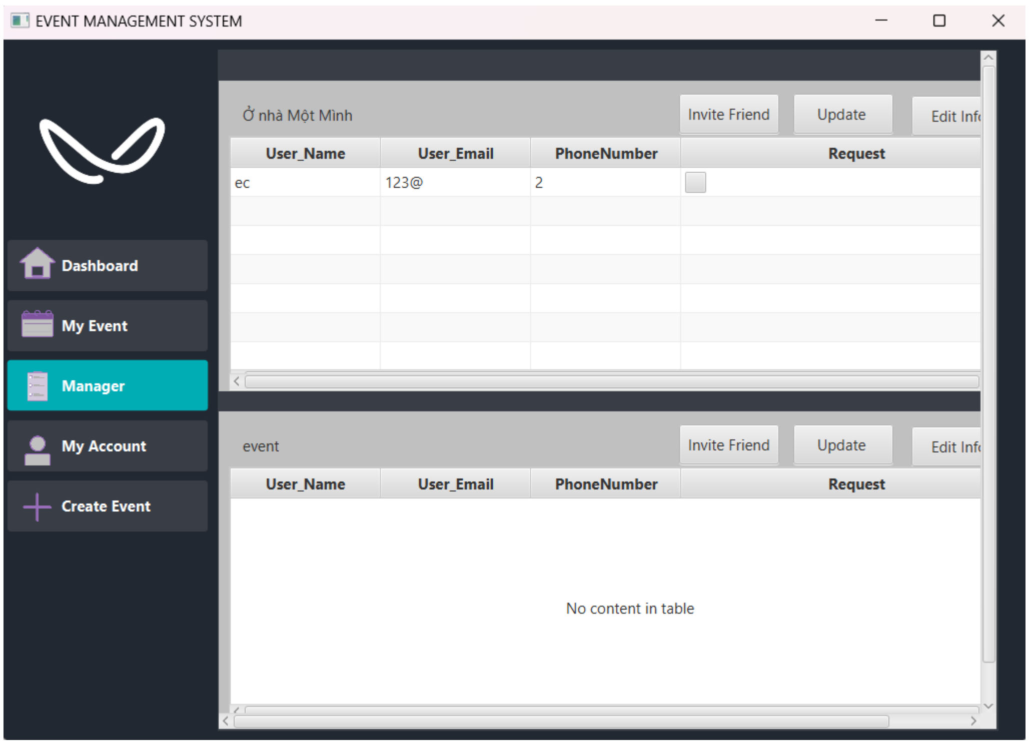


Figure 6.22: Sent Invitation Successfully



User_Name	User_Email	PhoneNumber	Request
ec	123@	2	☐

User_Name	User_Email	PhoneNumber	Request

Figure 6.23: List of Participant after invite

To edit the event data, click the edit button; an information screen will appear, where you can enter your new information.

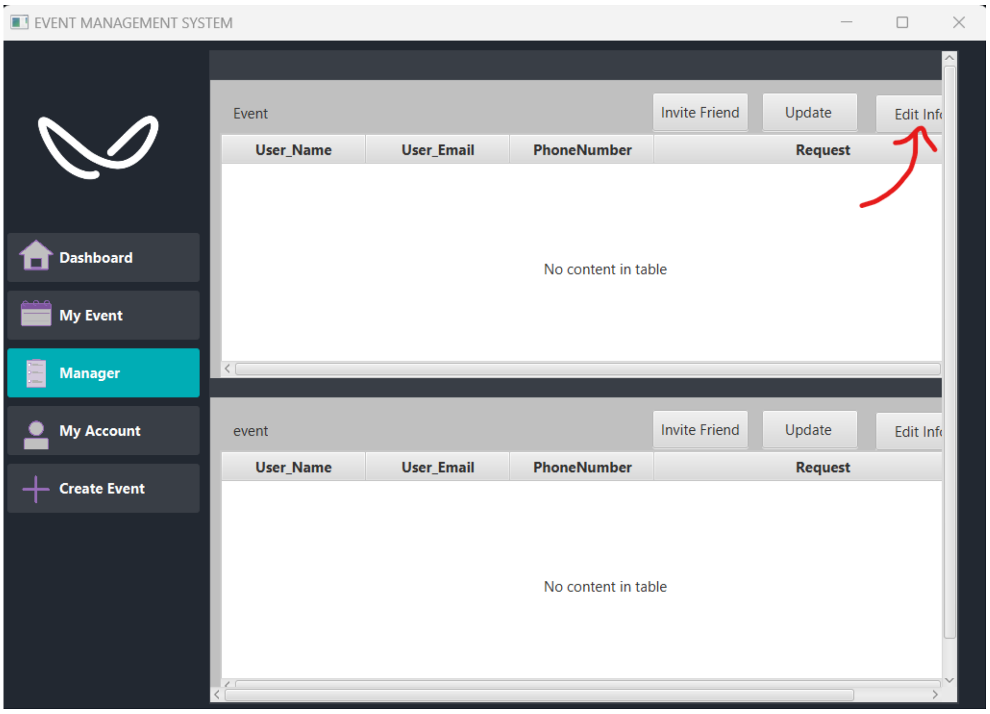
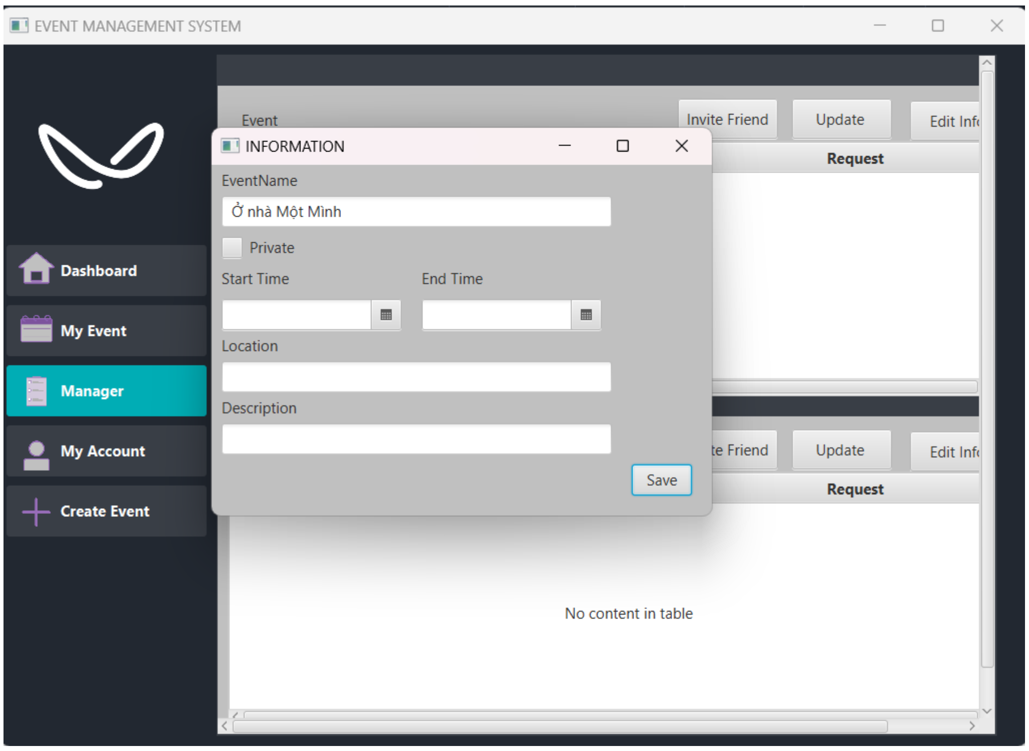


Figure 6.24: Edit Button



EventName: Ở nhà Một Mình

Private

Start Time

End Time

Location

Description

Save

Figure 6.25: INFORMATION Screen

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

The application have been build exactly like the project requirement. Despite of the error and a lack of secure, the application still run smoothly. This is the first time we build an application by ourselves, so we ran into a lot of problems. We encountered some problem in the process of connecting to database, fixing a secure code, fixing the interface, fixing errors during testing. After completing the project, we learned a lot of coding skill, cryptography, JDBC and utilizing Github.

7.2 Future work

In the future, we hopefully can fix the bug, add more function like job control for the organizers, add the firewall and the middle-ware server to control security for SHA-256 and AES encryption. Beside, the SHA-256 can be upgrade to newer hash function or add more layer for reinforcement security.

REFERENCE

[1] geeksforgeeks, *What is Data Encryption?*. Available: https://www.geeksforgeeks.org/what-is-data-encryption/?ref=lbp.

[2] geeksforgeeks, *Hash Functions in System Security*. Available: https://www.geeksforgeeks.org/hash-functions-system-security/?ref=lbp.

[3] Chua Hock-Chuan, *Java Programming Tutorial Programming Graphical User Interface (GUI)*. Available: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html.

[4] SonarQube, *the code quality tool for better code*. [Online]. Available: https://www.sonarsource.com/products/sonarqube/.

[5] Jenkins, [Online]. Available: https://www.jenkins.io/.