Agenda 拓展功能

拓展功能有：

1.更友好的 **Terminal UI;**

2.异常处理以及错误信息返回**;**

3.表单字段验证**;**

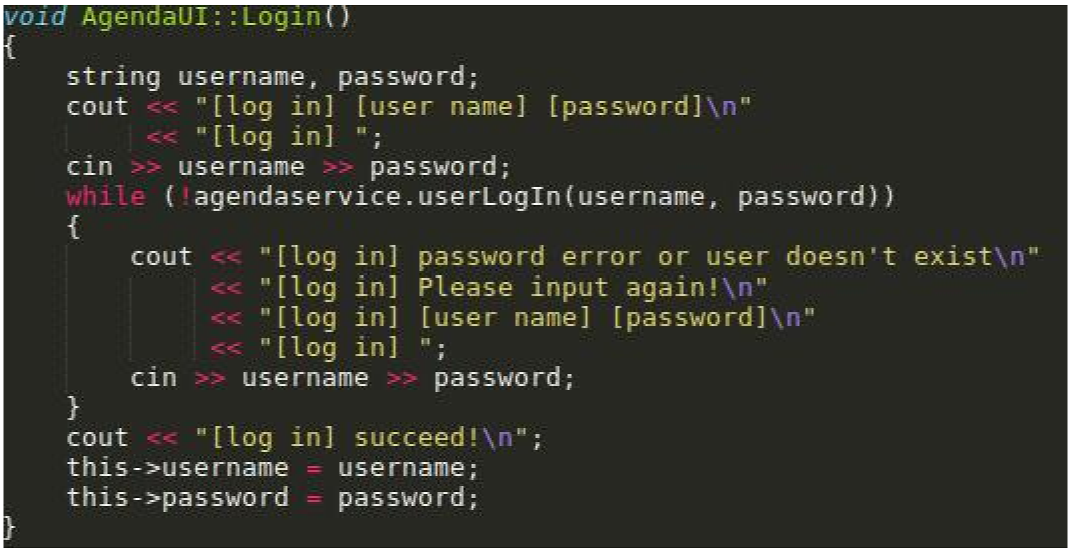
4.密码加密**;**

一. 更友好的 Terminal UI

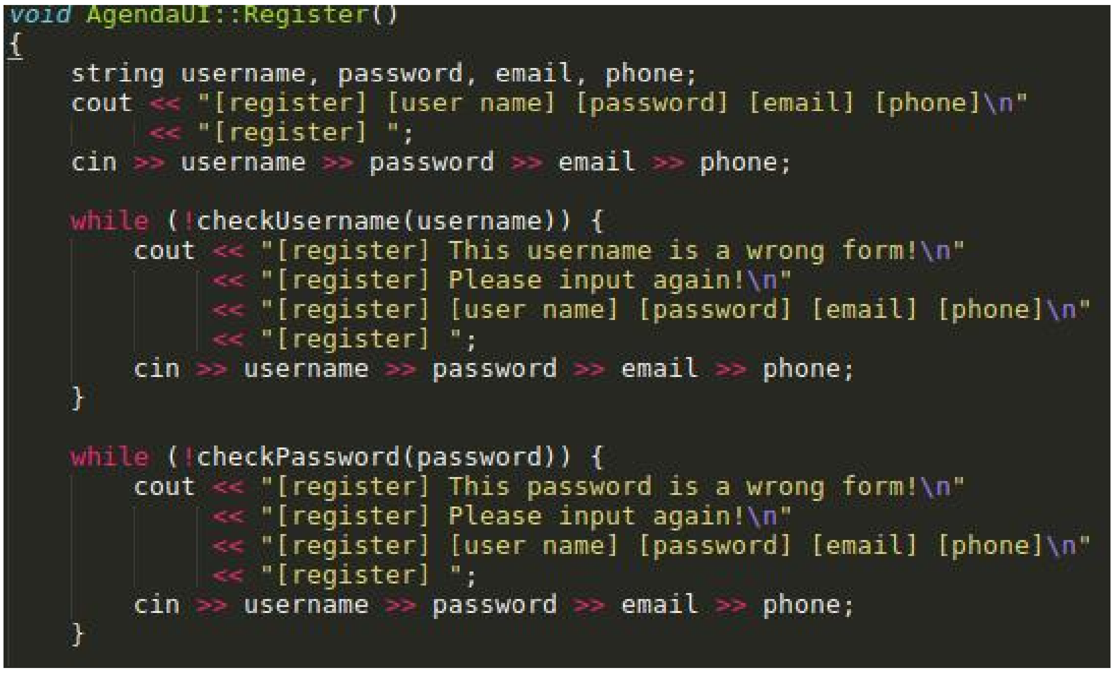
1.1 介绍：

当用户输入错误信息如错误的日期，错误的标题，错误的用户名、密码、邮箱或电话等时，提供给用户重新输入的机会。

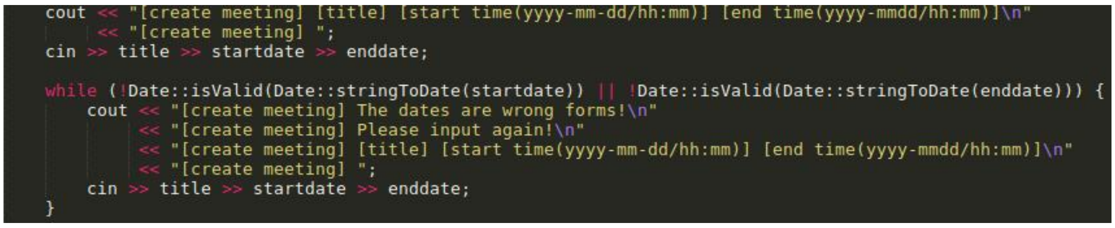
1.2 关键代码：



```
void AgendaUI::Login()
{
    string username, password;
    cout << "[log in] [user name] [password]\n"
         << "[log in] ";
    cin >> username >> password;
    while (!agendaservice.userLogIn(username, password))
    {
        cout << "[log in] password error or user doesn't exist\n"
             << "[log in] Please input again!\n"
             << "[log in] [user name] [password]\n"
             << "[log in] ";
        cin >> username >> password;
    }
    cout << "[log in] succeed!\n";
    this->username = username;
    this->password = password;
}
```



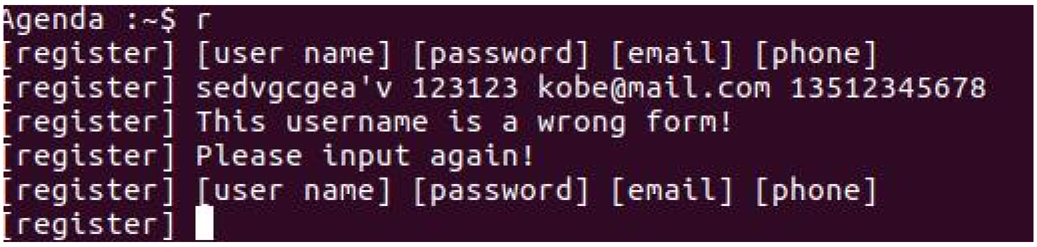
```
void AgendaUI::Register()
{
    string username, password, email, phone;
    cout << "[register] [user name] [password] [email] [phone]\n"
         << "[register] ";
    cin >> username >> password >> email >> phone;

    while (!checkUsername(username)) {
        cout << "[register] This username is a wrong form!\n"
             << "[register] Please input again!\n"
             << "[register] [user name] [password] [email] [phone]\n"
             << "[register] ";
        cin >> username >> password >> email >> phone;
    }

    while (!checkPassword(password)) {
        cout << "[register] This password is a wrong form!\n"
             << "[register] Please input again!\n"
             << "[register] [user name] [password] [email] [phone]\n"
             << "[register] ";
        cin >> username >> password >> email >> phone;
    }
```



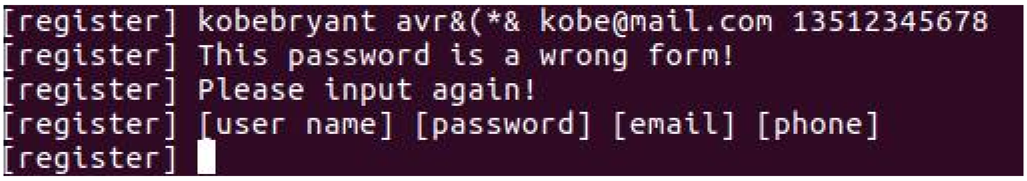
```
    cout << "[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]\n"
         << "[create meeting] ";
    cin >> title >> startdate >> enddate;

    while (!Date::isValid(Date::stringToDate(startdate)) || !Date::isValid(Date::stringToDate(enddate))) {
        cout << "[create meeting] The dates are wrong forms!\n"
             << "[create meeting] Please input again!\n"
             << "[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]\n"
             << "[create meeting] ";
        cin >> title >> startdate >> enddate;
    }
```

1.3 测试样例：

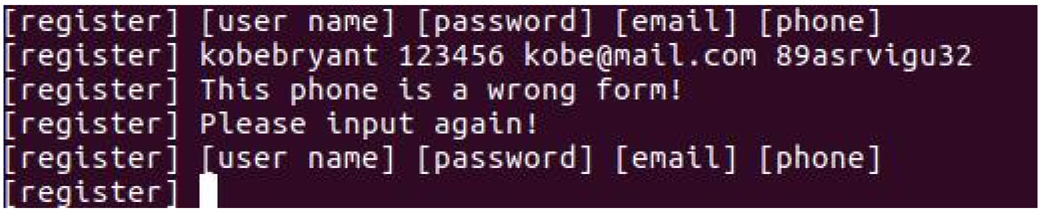
1.3.1 当用户名、密码、邮箱或电话格式错误时，可提示错误并重新输入：



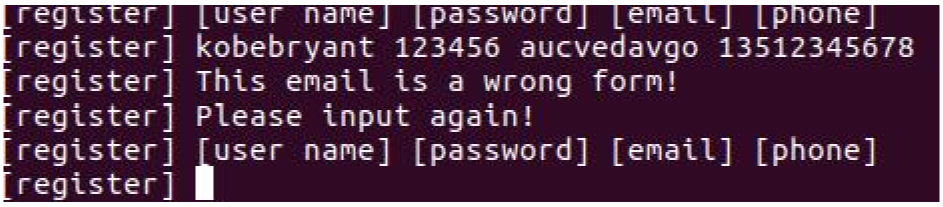
```
Agenda :~$ r
[register] [user name] [password] [email] [phone]
[register] sedvgcgea'v 123123 kobe@mail.com 13512345678
[register] This username is a wrong form!
[register] Please input again!
[register] [user name] [password] [email] [phone]
[register] 
```



```
[register] kobebryant avr&(*& kobe@mail.com 13512345678
[register] This password is a wrong form!
[register] Please input again!
[register] [user name] [password] [email] [phone]
[register] 
```

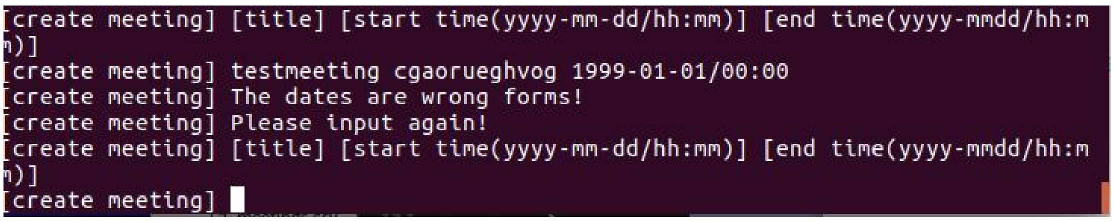


```
[register] [user name] [password] [email] [phone]
[register] kobebryant 123456 kobe@mail.com 89asrvigu32
[register] This phone is a wrong form!
[register] Please input again!
[register] [user name] [password] [email] [phone]
[register] 
```



```
[register] [user name] [password] [email] [phone]
[register] kobebryant 123456 aucvedavgo 13512345678
[register] This email is a wrong form!
[register] Please input again!
[register] [user name] [password] [email] [phone]
[register] 
```

1.3.2 当输入时间格式错误时，可提示错误并重新输入：

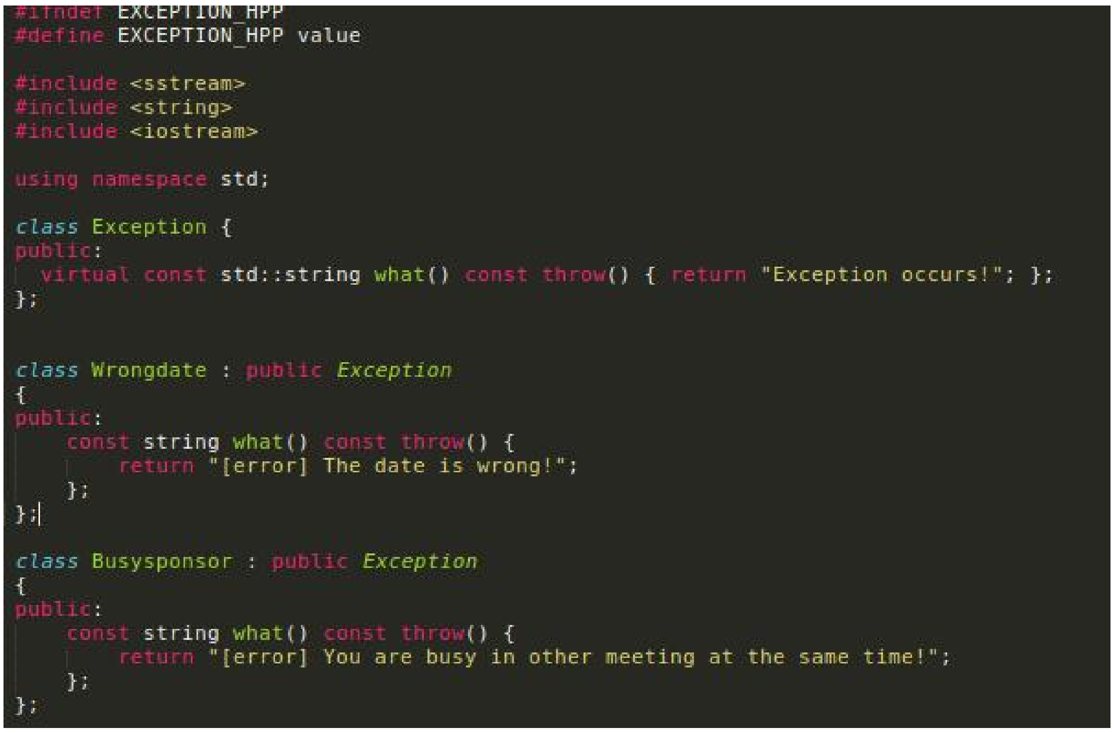

```
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:m
m)]
[create meeting] testmeeting cgaorueghvog 1999-01-01/00:00
[create meeting] The dates are wrong forms!
[create meeting] Please input again!
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:m
m)]
[create meeting] 
```

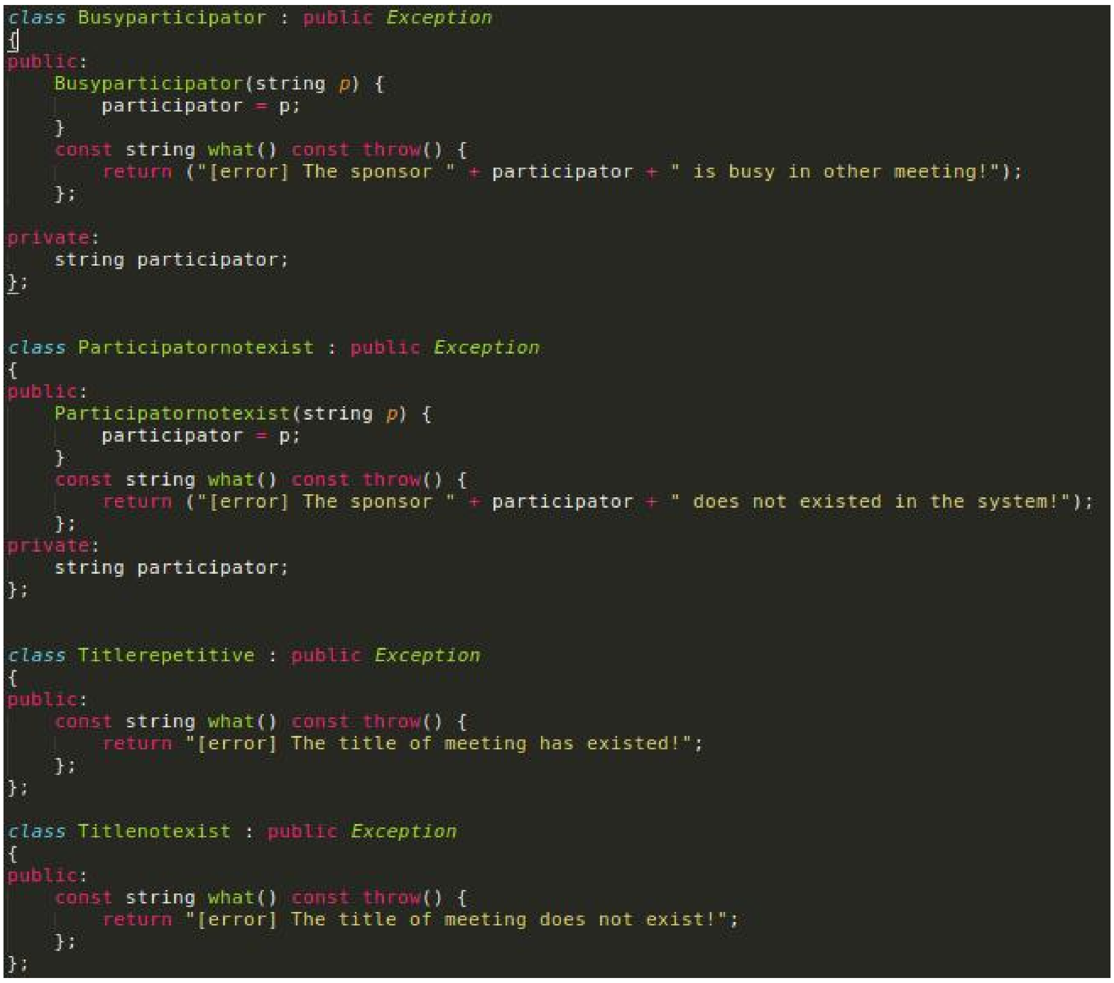
二. 异常处理以及错误信息返回

2.1 介绍：

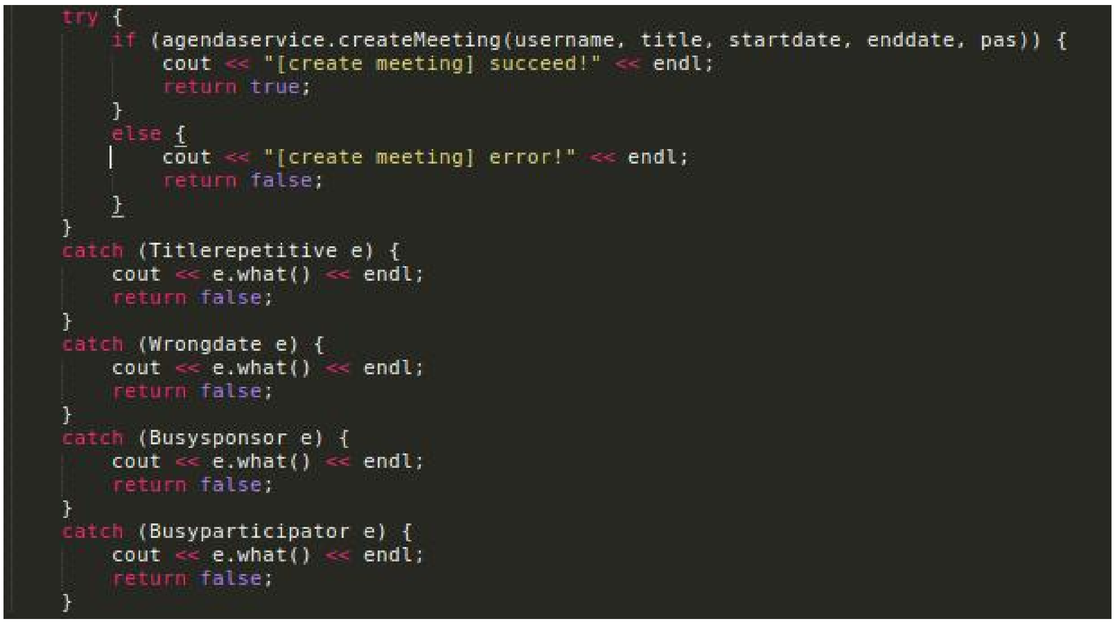
当输入错误信息或乱码时，程序能找到错误原因并返回错误信息，以及处理异常错误，在 UI 层采用 try catch 机制捕获从 Agenda 桌 Service 中抛出的异常，再处理反馈给用户，并创建异常类。

2.2 关键代码：

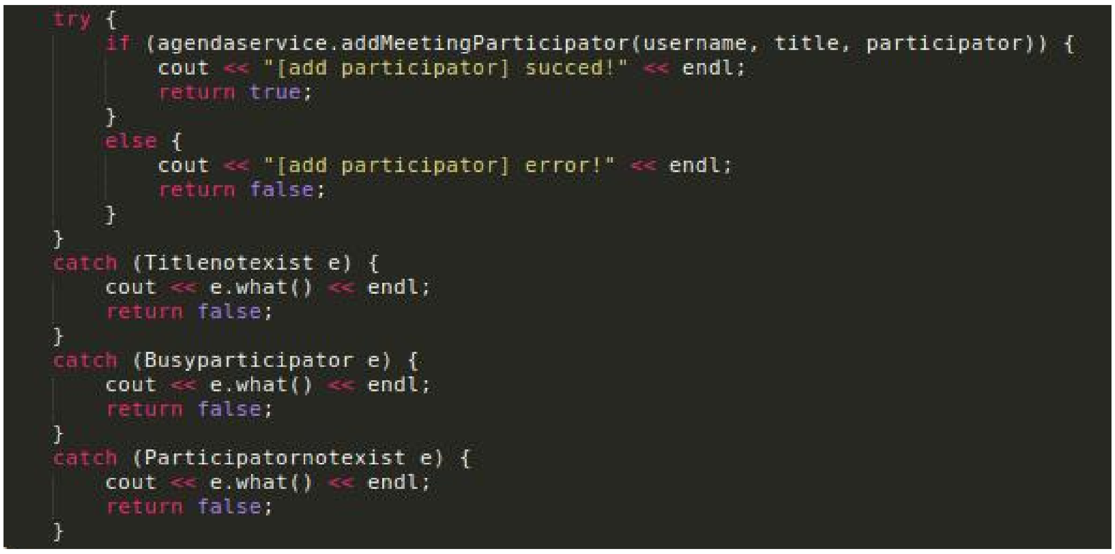

```cpp
#ifndef EXCEPTION_HPP
#define EXCEPTION_HPP value

#include <sstream>
#include <string>
#include <iostream>

using namespace std;

class Exception {
public:
  virtual const std::string what() const throw() { return "Exception occurs!"; };
};


class Wrongdate : public Exception
{
public:
    const string what() const throw() {
        return "[error] The date is wrong!";
    };
};

class Busysponsor : public Exception
{
public:
    const string what() const throw() {
        return "[error] You are busy in other meeting at the same time!";
    };
};
```



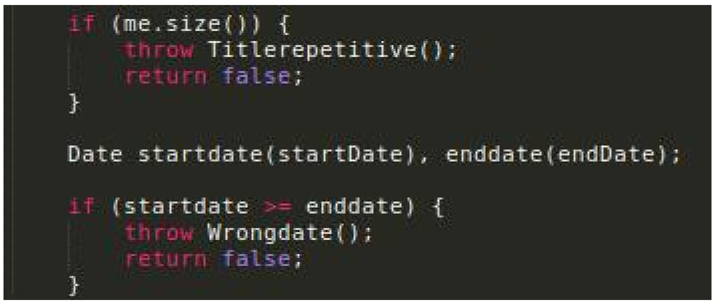
```cpp
class Busyparticipator : public Exception
{
public:
    Busyparticipator(string p) {
        participator = p;
    }
    const string what() const throw() {
        return ("[error] The sponsor " + participator + " is busy in other meeting!");
    };

private:
    string participator;
};


class Participatornotexist : public Exception
{
public:
    Participatornotexist(string p) {
        participator = p;
    }
    const string what() const throw() {
        return ("[error] The sponsor " + participator + " does not existed in the system!");
    };
private:
    string participator;
};


class Titlerepetitive : public Exception
{
public:
    const string what() const throw() {
        return "[error] The title of meeting has existed!";
    };
};

class Titlenotexist : public Exception
{
public:
    const string what() const throw() {
        return "[error] The title of meeting does not exist!";
    };
};
```



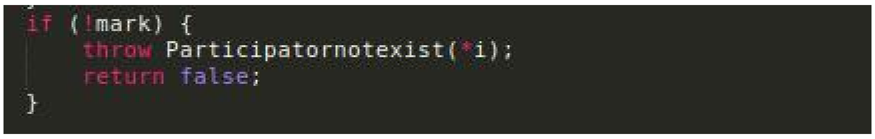
```cpp
    try {
        if (agendaservice.createMeeting(username, title, startdate, enddate, pas)) {
            cout << "[create meeting] succeed!" << endl;
            return true;
        }
        else {
            cout << "[create meeting] error!" << endl;
            return false;
        }
    }
    catch (Titlerepetitive e) {
        cout << e.what() << endl;
        return false;
    }
    catch (Wrongdate e) {
        cout << e.what() << endl;
        return false;
    }
    catch (Busysponsor e) {
        cout << e.what() << endl;
        return false;
    }
    catch (Busyparticipator e) {
        cout << e.what() << endl;
        return false;
    }
```



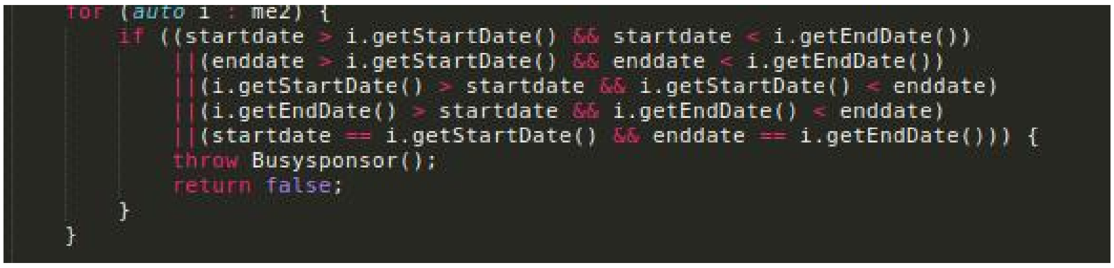
```
    try {
        if (agendaservice.addMeetingParticipator(username, title, participator)) {
            cout << "[add participator] succed!" << endl;
            return true;
        }
        else {
            cout << "[add participator] error!" << endl;
            return false;
        }
    }
    catch (Titlenotexist e) {
        cout << e.what() << endl;
        return false;
    }
    catch (Busyparticipator e) {
        cout << e.what() << endl;
        return false;
    }
    catch (Participatornotexist e) {
        cout << e.what() << endl;
        return false;
    }
```



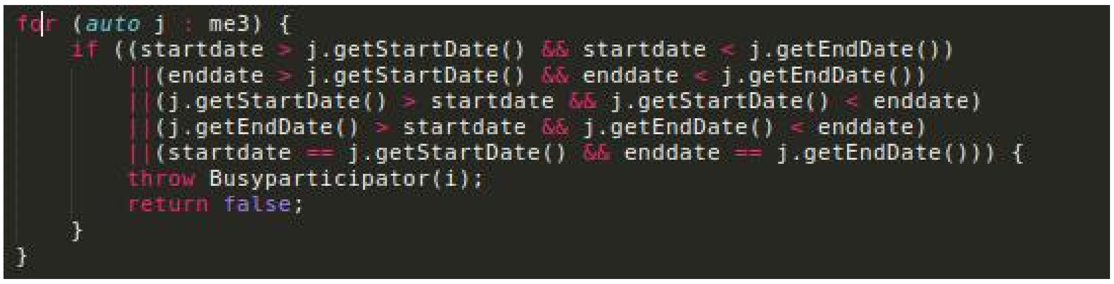
```
    if (me.size()) {
        throw Titlerepetitive();
        return false;
    }

    Date startdate(startDate), enddate(endDate);

    if (startdate >= enddate) {
        throw Wrongdate();
        return false;
    }
```



```
 if (!mark) {
     throw Participatornotexist(*i);
     return false;
 }
```



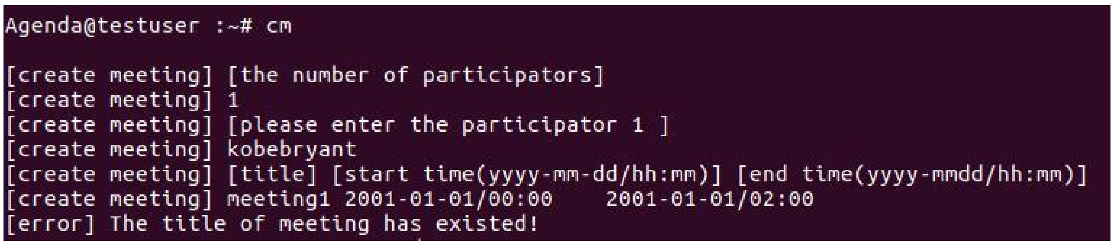
```
    for (auto i : me2) {
        if ((startdate > i.getStartDate() && startdate < i.getEndDate())
            ||(enddate > i.getStartDate() && enddate < i.getEndDate())
            ||(i.getStartDate() > startdate && i.getStartDate() < enddate)
            ||(i.getEndDate() > startdate && i.getEndDate() < enddate)
            ||(startdate == i.getStartDate() && enddate == i.getEndDate())) {
            throw Busysponsor();
            return false;
        }
    }
```



```
for (auto j : me3) {
    if ((startdate > j.getStartDate() && startdate < j.getEndDate())
        ||(enddate > j.getStartDate() && enddate < j.getEndDate())
        ||(j.getStartDate() > startdate && j.getStartDate() < enddate)
        ||(j.getEndDate() > startdate && j.getEndDate() < enddate)
        ||(startdate == j.getStartDate() && enddate == j.getEndDate())) {
        throw Busyparticipator(i);
        return false;
    }
}
```

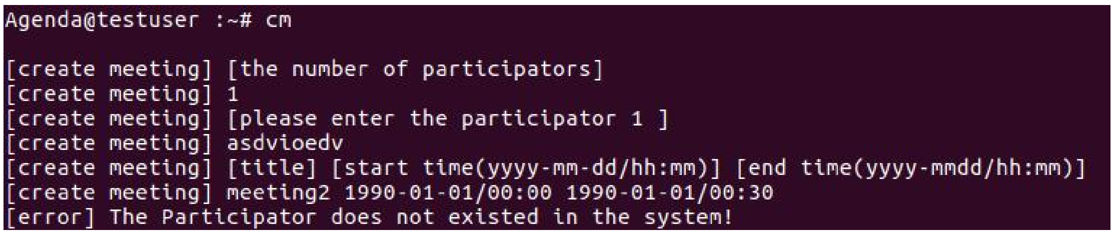
2.3 测试样例：

2.3.1 当创建会议时标题已存在：



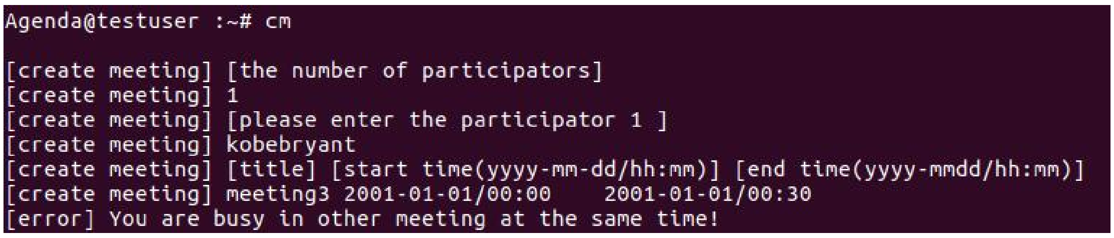
```
Agenda@testuser :~# cm

[create meeting] [the number of participators]
[create meeting] 1
[create meeting] [please enter the participator 1 ]
[create meeting] kobebryant
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]
[create meeting] meeting1 2001-01-01/00:00    2001-01-01/02:00
[error] The title of meeting has existed!
```

2.3.2 当创建会议时添加参与者不在系统内：



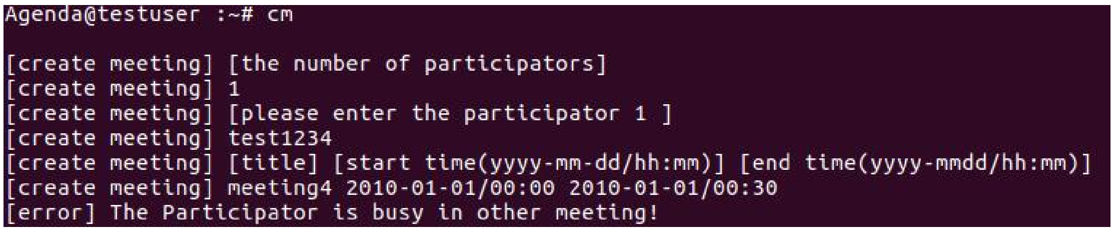
```
Agenda@testuser :~# cm

[create meeting] [the number of participators]
[create meeting] 1
[create meeting] [please enter the participator 1 ]
[create meeting] asdvioedv
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]
[create meeting] meeting2 1990-01-01/00:00 1990-01-01/00:30
[error] The Participator does not existed in the system!
```

2.3.3 当创建会议时会议时间与发起者的其他会议时间冲突：



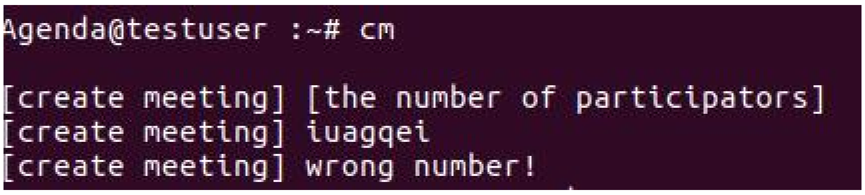
```
Agenda@testuser :~# cm

[create meeting] [the number of participators]
[create meeting] 1
[create meeting] [please enter the participator 1 ]
[create meeting] kobebryant
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]
[create meeting] meeting3 2001-01-01/00:00    2001-01-01/00:30
[error] You are busy in other meeting at the same time!
```

2.3.4 当创建会议时会议时间与参与者的其他会议时间冲突：



```
Agenda@testuser :~# cm

[create meeting] [the number of participators]
[create meeting] 1
[create meeting] [please enter the participator 1 ]
[create meeting] test1234
[create meeting] [title] [start time(yyyy-mm-dd/hh:mm)] [end time(yyyy-mmdd/hh:mm)]
[create meeting] meeting4 2010-01-01/00:00 2010-01-01/00:30
[error] The Participator is busy in other meeting!
```

2.3.5 当需要输入数字时输入了乱码：



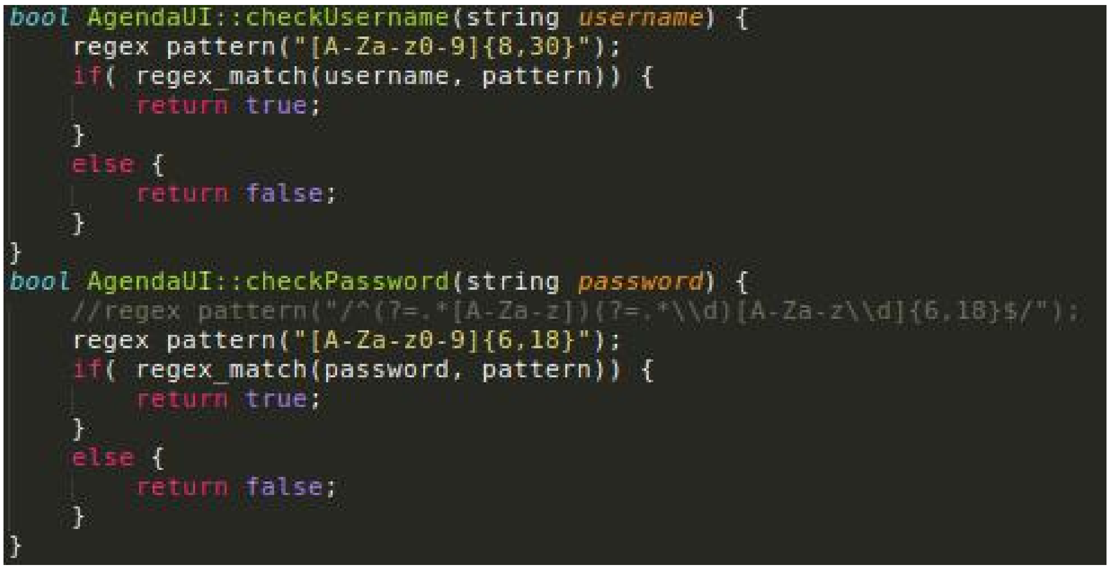
```
Agenda@testuser :~# cm

[create meeting] [the number of participators]
[create meeting] iuagqei
[create meeting] wrong number!
```

三. 表单字段验证

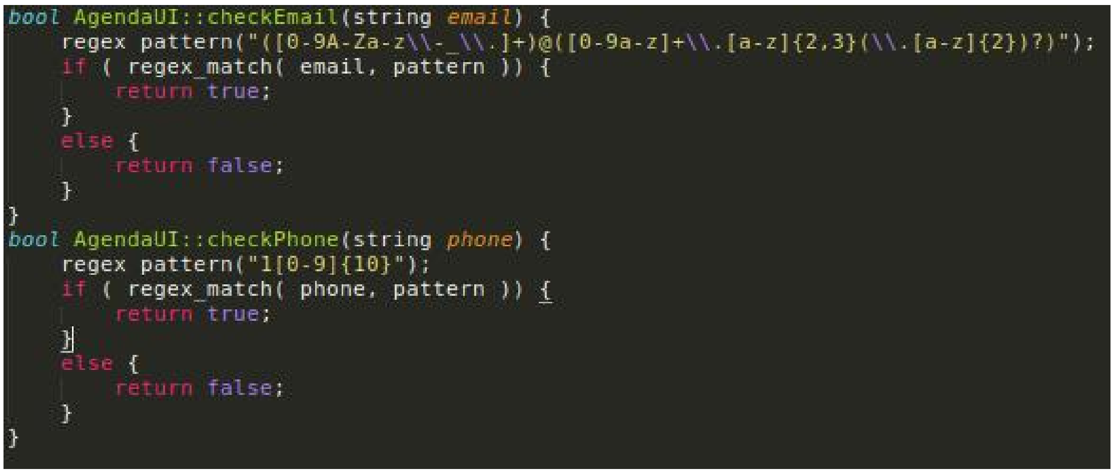
3.1 介绍：

利用正则表达式，对用户的用户名、密码、邮箱及电话进行字段验证：其中用户名为 8～30 个字母或数字组成,密码为 6～18 个字母或数字组成,邮箱包含字母、数字、@符号和小数点，电话为 1 开头的 13 个数字组成。

3.2 关键代码：



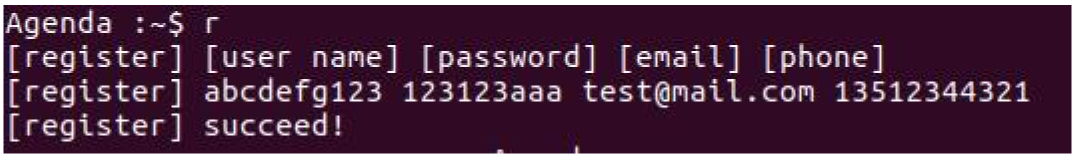
```
bool AgendaUI::checkUsername(string username) {
    regex pattern("[A-Za-z0-9]{8,30}");
    if( regex_match(username, pattern)) {
        return true;
    }
    else {
        return false;
    }
}
bool AgendaUI::checkPassword(string password) {
    //regex pattern("/^(?=.*[A-Za-z])(?=.*\\d)[A-Za-z\\d]{6,18}$/");
    regex pattern("[A-Za-z0-9]{6,18}");
    if( regex_match(password, pattern)) {
        return true;
    }
    else {
        return false;
    }
}
```



```cpp
bool AgendaUI::checkEmail(string email) {
    regex pattern("([0-9A-Za-z\\-_\\.]+)@([0-9a-z]+\\.[a-z]{2,3}(\\.[a-z]{2})?)");
    if ( regex_match( email, pattern )) {
        return true;
    }
    else {
        return false;
    }
}
bool AgendaUI::checkPhone(string phone) {
    regex pattern("1[0-9]{10}");
    if ( regex_match( phone, pattern )) {
        return true;
    }
    else {
        return false;
    }
}
```

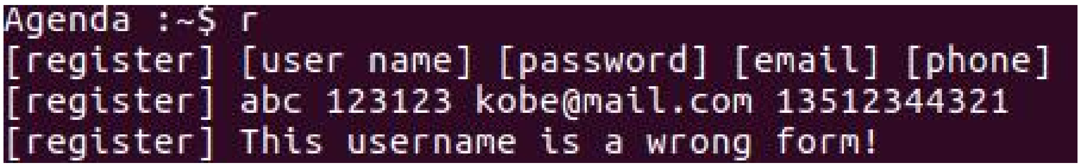
3.3 测试样例

3.3.1 正确的输入格式：



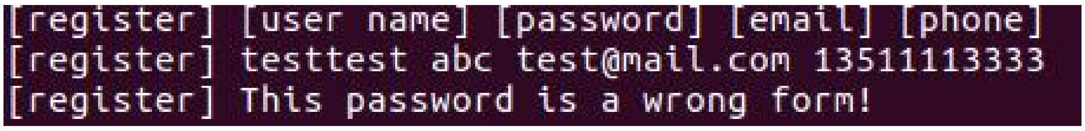
```
Agenda :~$ r
[register] [user name] [password] [email] [phone]
[register] abcdefg123 123123aaa test@mail.com 13512344321
[register] succeed!
```

3.3.2 错误的用户名输入格式：



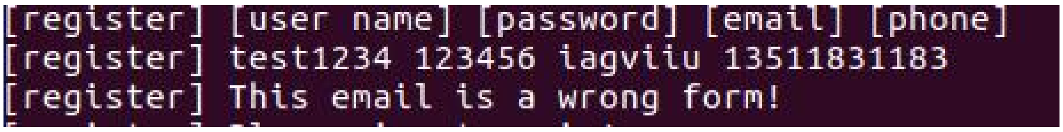
```
Agenda :~$ r
[register] [user name] [password] [email] [phone]
[register] abc 123123 kobe@mail.com 13512344321
[register] This username is a wrong form!
```

3.3.3 错误的密码输入格式：



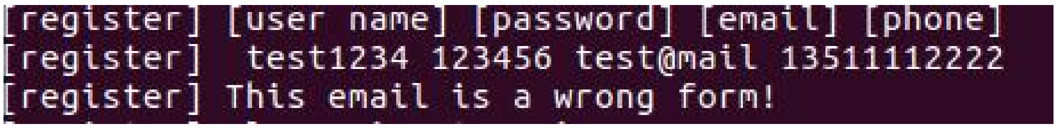
```
[register] [user name] [password] [email] [phone]
[register] testtest abc test@mail.com 13511113333
[register] This password is a wrong form!
```

3.3.4 错误的邮箱输入格式 1：



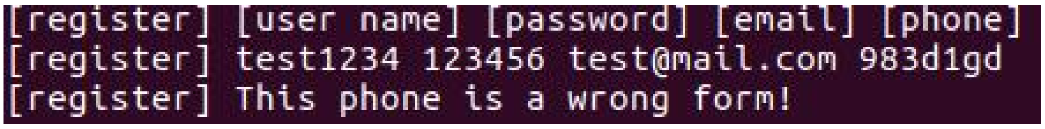
```
[register] [user name] [password] [email] [phone]
[register] test1234 123456 iagviiu 13511831183
[register] This email is a wrong form!
```

3.3.5 错误的邮箱输入格式 2：



```
[register] [user name] [password] [email] [phone]
[register]  test1234 123456 test@mail 13511112222
[register] This email is a wrong form!
```

3.3.6 错误的电话输入格式：



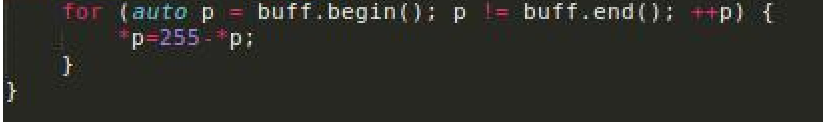
```
[register] [user name] [password] [email] [phone]
[register] test1234 123456 test@mail.com 983d1gd
[register] This phone is a wrong form!
```

四．密码加密

4.1 介绍

利用简单加密算法将密码加密，只存储密文，防止数据文件被窃取时，用户密码泄露。

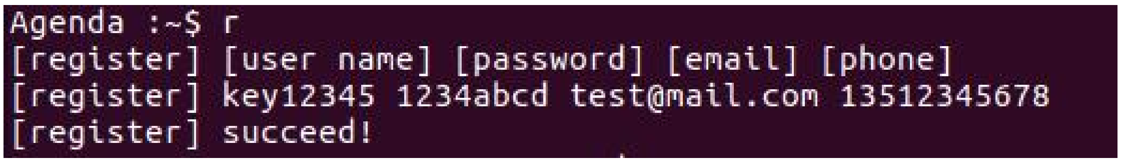
4.2 关键代码



```
void encode(string &buff){
    for (auto p = buff.begin(); p != buff.end(); ++p) {
        *p=255-*p;
    }
}
```

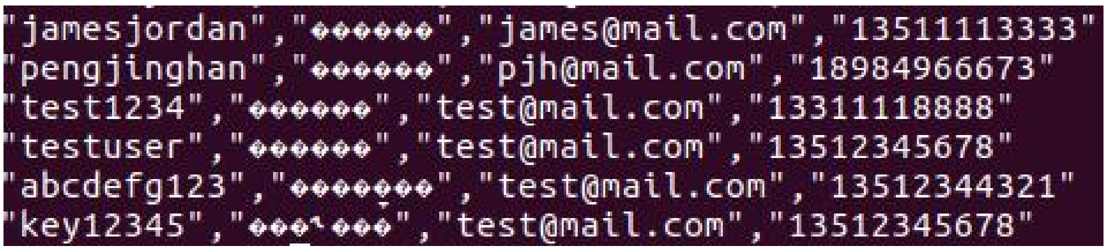
4.3 测试样例

4.3.1 注册时的密码设置为 key12345：



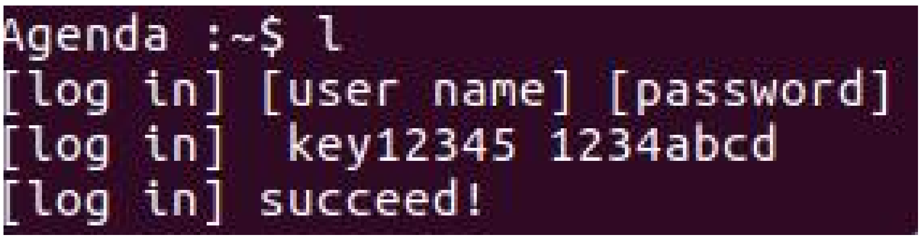
```
Agenda :~$ r
[register] [user name] [password] [email] [phone]
[register] key12345 1234abcd test@mail.com 13512345678
[register] succeed!
```

4.3.2 打开数据文件时的密码为乱码：



```
"jamesjordan","������","james@mail.com","13511113333"
"pengjinghan","������","pjh@mail.com","18984966673"
"test1234","������","test@mail.com","13311118888"
"testuser","������","test@mail.com","13512345678"
"abcdefg123","�������","test@mail.com","13512344321"
"key12345","���`���","test@mail.com","13512345678"
```

4.3.3 重新登录时能已注册的密码成功登录：



```
Agenda :~$ l
[log in] [user name] [password]
[log in]  key12345 1234abcd
[log in] succeed!
```