**第三次实训工作报告**

软件2班 杨璐玮

**本周工作小结：**

根据实际需要完善了用户、团队和事件的结构体成员，在此基础上

实现了团队管理类以及团队和用户、事件之间的关联逻辑。

**完善TEAM和EVENT结构体**

struct TEAM

{

int team\_id;

QString team\_passwd;

QString team\_name;

int leader\_id;

int team\_members\_nums;

int team\_events\_nums;

//存放团队成员信息的文件

QString team\_members\_filename;

//存放团队事件信息的文件

QString team\_events\_filename;

QVector<USER> team\_members;

QVector<EVENT> team\_events;

TEAM():team\_members\_nums(0),team\_events\_nums(0),leader\_id(0),team\_id(0),team\_name(""),team\_passwd(""),team\_members\_filename(""),team\_events\_filename(""),team\_members(),team\_events()

{

}

TEAM(QString team\_name,QString team\_passwd,int USR\_ID\_NOW)

{

//把创始人作为第一个成员存入团队的人员信息中

team\_members.push\_back(users[USR\_ID\_NOW- USER\_ID\_FORE]);

}

};

struct EVENT

{

int event\_id;

QString event\_name;

QString event\_date;

QString event\_starttime;

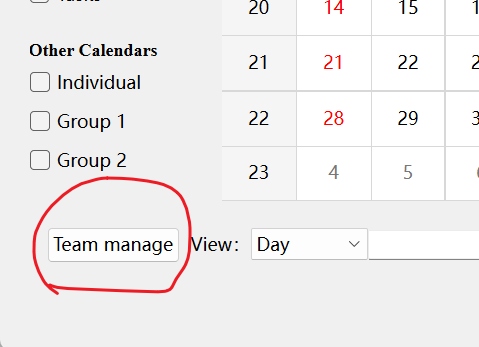
QString event\_endtime;

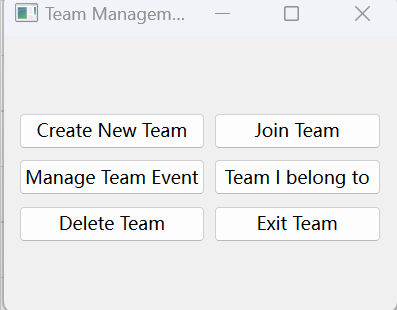
QString event\_content;

int urgentdegree;

};

**在日历主视图中增加了团队管理的入口：**



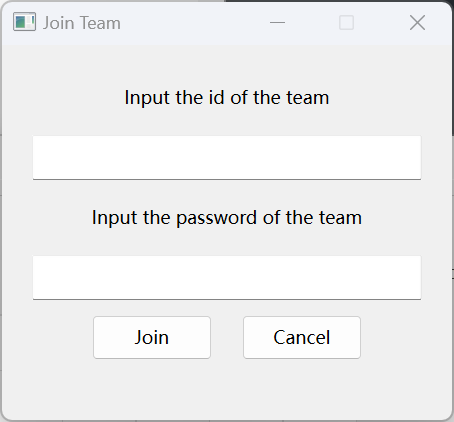
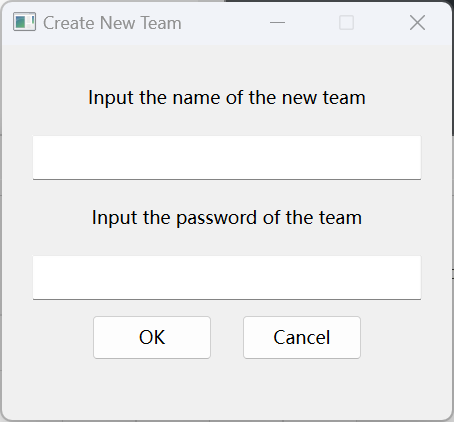


计划在团队管理中实现两种使用者身份的对应功能：团队创建者和团队成员。

作为团队创建者，应该有创建团队、管理团队日程和解散团队三大基本功能。

而作为团队成员，应该有加入团队、查看所属团队、和推出团队的基本功能。下面进行创建和加入团队的业务逻辑完善。

使用team\_id来唯一标识特定的团队，并且在创建和加入的时候需要设置/输入正确密码。



**下面介绍团队管理具体功能的实现：**

团队管理的模块主要构成为team\_mng.h/cpp两个文件

首先定义团队结构体数组用于存放所有的团队，以及一个维护团队数量的变量team\_nums;

//所有团队的集合

TEAM teams[MAX\_TEAM\_NUM];

unsigned int team\_nums;

extern USER users[MAX\_USRS\_NUM];

extern int USR\_ID\_NOW;

QVector<TEAM> myteam\_create=users[USR\_ID\_NOW- USER\_ID\_FORE].teams\_create;

QVector<TEAM> myteam\_belong=users[USR\_ID\_NOW- USER\_ID\_FORE].teams\_belong;

需要注意的是，该日历项目中的所有业务操作都需要用户在登录状态下完成，因为所有的操作都会背记录到对应的文件中。因此用户登录之后获得的用户id是需要在全局维护的。这里使用extern引用当前操作用户的id和所有用户的结构体数组。便于后续的文件读写操作。

在开始创建团队之前，和用户以及日程文件读取类似地，使用一个团队结构体初始化函数来实现数据的留存和读写，初始化团队结构体数组。

void init\_teams\_structarr(TEAM\* teams, int size)

{

QFile file("TEAMS.txt");

if (file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QTextStream stream(&file);

int index = 0;

while (!stream.atEnd() && index < size)

{

QString line = stream.readLine();

QStringList values = line.split(',');

// 读取并保存到结构体数组

if (values.size() >= 8) {

teams[index].team\_id = values[0].toInt();

teams[index].team\_name = values[1];

teams[index].team\_passwd = values[2];

teams[index].leader\_id = values[3].toInt();

teams[index].team\_members\_nums = values[4].toInt();

teams[index].team\_events\_nums = values[5].toInt();

teams[index].team\_members\_filename = values[6];

teams[index].team\_events\_filename = values[7];

index++;

}

}

team\_nums = index;

file.close();

qDebug() << "团队结构体数组初始化成功！";

}

else {

qDebug() << "无法打开文件！";

}

}

执行团队创建业务的主要函数是void team\_mng::create\_new\_team()

{

/\*

省略界面设计的代码

\*/

//创建一个新的团队

myteam\_create.push\_back(TEAM(lineEdit->text(), lineEdit\_2->text(), USR\_ID\_NOW));

//分配团队id

new\_team\_id = TEAM\_ID\_FORE+team\_nums;

//将新团队的信息写入所有团队数组中该团队的对应信息中

teams[team\_nums].team\_id = new\_team\_id;

teams[team\_nums].team\_name = lineEdit->text();

teams[team\_nums].team\_passwd = lineEdit\_2->text();

teams[team\_nums].leader\_id= USR\_ID\_NOW;

teams[team\_nums].team\_members\_nums = 1;

teams[team\_nums].team\_members.push\_back(users[USR\_ID\_NOW-USER\_ID\_FORE]);

teams[team\_nums].team\_events\_nums = 0;

//为每个团队的成员信息专门创建文件并写入

QString filename = QString::number(new\_team\_id) + "\_team members.txt";

QFile file(filename);

if (!file.open(QIODevice::WriteOnly | QIODevice::Text | QIODevice::Append))

return;

QTextStream out(&file);

out << teams[team\_nums].team\_members\_nums << ",";

//循环写入所有成员的id

for (int i = 0; i < teams[team\_nums].team\_members\_nums; i++)

{

out << teams[team\_nums].team\_members[i].usr\_id << ",";

}

//为每个团队的事件信息专门创建文件并写入

QString filename\_2 = QString::number(new\_team\_id) + "\_team events.txt";

QFile file\_2(filename\_2);

if (!file\_2.open(QIODevice::WriteOnly | QIODevice::Text | QIODevice::Append))

return;

QTextStream out\_2(&file\_2);

out\_2 << teams[team\_nums].team\_events\_nums << " ,";

//循环写入所有事件的id

for (int i = 0; i < teams[team\_nums].team\_events\_nums; i++)

{

out\_2 << teams[team\_nums].team\_events[i].event\_id << ",";

}

//将新团队的信息写入用户的团队列表

users[USR\_ID\_NOW - USER\_ID\_FORE].teams\_create.push\_back(teams[team\_nums]);

//将新团队的信息写入TEAMS文件和创建自己的文件

QFile team\_file("TEAMS.txt");

if (!team\_file.open(QIODevice::WriteOnly | QIODevice::Text | QIODevice::Append))

return;

QTextStream team\_file\_out(&team\_file);

team\_file\_out << teams[team\_nums].team\_id <<","<< teams[team\_nums].team\_name << "," << teams[team\_nums].team\_passwd << "," << USR\_ID\_NOW << "," << 1 << "," << 0<<","<< filename <<","<< filename\_2;

team\_file\_out << "\n";

team\_file.close();

}

当进行团队创建时，如果创建成功，会自动根据已有团队总数来分配团队的id并在点击确定按钮之后进行提示。已有的团队总数则是通过读取TEAMS.txt文件中团队记录的条数获得的。

void team\_mng::confirm\_creat()

{

//弹出窗口提示创建团队成功并且反馈团队id

QMessageBox::information(nullptr, "Successfully create a team!", "The id of your team is: " + QString::number(new\_team\_id));

}

需要注意的是，为了提高程序的可读性和可维护性，在配置文件”config.h”中使用宏定义将一些常用的常量事先规定好：

例如：

#define MAX\_USRS\_NUM 100

#define MAX\_USRS\_EVENT 100

#define MAX\_TEAM\_NUM 100

//用户id的前缀

#define USER\_ID\_FORE 1001

//团队id的前缀

#define TEAM\_ID\_FORE 2001