To-Do-List

1.0

Generated by Doxygen 1.14.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 TaskManager Namespace Reference

Functions

- bool createTask (const Task &task)

  Creates Task in a database.
- std::vector< Task > getTasksForTeam (uint32_t teamId)

  Returns all Team's Task.
- std::vector< Task > getTasksForUser (uint32_t userId)

  Returns all User's Task.
- Task getTask (uint64_t id)

  Gets Task from a database identifying it by ID and returns as Task's object.
- bool updateTask (const Task &task)

  Updates Task in a database.
- bool deleteTask (const Task &task)

  Deletes Task from a database.

5.1.1 Function Documentation

5.1.1.1 createTask()

```
bool TaskManager::createTask (
                const Task & task)
```

Creates Task in a database.

Parameters

	task	Task to create in a database

Returns

True if success, false if there was any error

5.1.1.2 deleteTask()

```
bool TaskManager::deleteTask (
                const Task & task)
```

Deletes Task from a database.

Parameters

	task	Task to delete

Returns

True if success, false if Task doesn't exist or there was any error

5.1.1.3 getTask()

Task TaskManager::getTask (
uint64_t id)

Gets Task from a database identifying it by ID and returns as Task's object.

Parameters

	id	Task's ID

Returns

Task's object, if Task->id is 0, then Task doesn't exist, or there was an error

5.1.1.4 getTasksForTeam()

std::vector< Task > TaskManager::getTasksForTeam (
uint32_t teamId)

Returns all Team's Task.

Parameters

	teamId	Team's ID

Returns

Vector of Team's Tasks

5.1.1.5 getTasksForUser()

std::vector< Task > TaskManager::getTasksForUser (
uint32_t userId)

Returns all User's Task.

Parameters

	userId	User's ID

Returns

Vector of User's Tasks

5.1.1.6 updateTask()

```
bool TaskManager::updateTask (
                    const Task & task)
```

Updates Task in a database.

Parameters

	task	Task to update

Returns

True if success, false if there was any error

5.2 TeamManager Namespace Reference

Functions

- bool createTeam (const Team &team)
  Creates Team in a database.
- Team getTeam (const std::string &name)
  Gets Team from a database identifying it by name and returns as Team's object.
- Team getTeam (uint32_t id)
  Gets Team from a database identifying it by ID and returns as Team's object.
- bool updateTeam (const Team &team)
  Updates Team in a database.
- bool deleteTeam (const Team &team)
  Deletes Team from a database.
- bool deleteTeam (uint32_t id)
  Deletes Team using his ID from a database.
- std::vector< Team > getAllTeams ()
  Returns all Teams from a database.
- std::vector< Team > getTeamsForUser (uint32_t userId)
  Returns all Teams that the User belongs to from a database.
- Team getTeamForUser (uint32_t userId)
  Returns first User's Team from a database.

5.2.1 Function Documentation

5.2.1.1 createTeam()

```
bool TeamManager::createTeam (
                    const Team & team)
```

Creates Team in a database.

Creates Team in database.

Parameters

	team	Team to create

Returns

True if success, false if there was any error

5.2.1.2 deleteTeam() [1/2]

```
bool TeamManager::deleteTeam (
                const Team & team)
```

Deletes Team from a database.

Parameters

	team	Team to delete

Returns

True if success, false if Team doesn't exist or there was any error

5.2.1.3 deleteTeam() [2/2]

```
bool TeamManager::deleteTeam (
                uint32_t id)
```

Deletes Team using his ID from a database.

Parameters

	id	ID of a Team to delete

Returns

True if success, false if Team doesn't exist or there was any error

5.2.1.4 getAllTeams()

```
std::vector< Team > TeamManager::getAllTeams ()
```

Returns all Teams from a database.

Returns

Vector of all Teams from a database

5.2.1.5 getTeam() [1/2]

```
Team TeamManager::getTeam (
                const std::string & name)
```

Gets Team from a database identifying it by name and returns as Team's object.

Parameters

	name	Team's name

Returns

Team's object, if Team->id equals 0, then Team doesn't exist, or there was an error

5.2.1.6 getTeam() [2/2]

Team TeamManager::getTeam (
uint32_t id)

Gets Team from a database identifying it by ID and returns as Team's object.

Parameters

	id	Team's ID

Returns

Team's object, if Team->id equals 0, then Team doesn't exist, or there was an error

5.2.1.7 getTeamForUser()

Team TeamManager::getTeamForUser (
uint32_t userId)

Returns first User's Team from a database.

Returns

First User's Team

5.2.1.8 getTeamsForUser()

std::vector< Team > TeamManager::getTeamsForUser (
uint32_t userId)

Returns all Teams that the User belongs to from a database.

Returns

Vector of all Teams that the User belongs to

5.2.1.9 updateTeam()

bool TeamManager::updateTeam (
const Team & team)

Updates Team in a database.

Parameters

	team	Team to update

Returns

True if success, false if there was any error

5.3 Ui Namespace Reference

5.4 UserManager Namespace Reference

Functions

- bool createUser (const User &user)

  Creates User in a database.
- User getUser (const std::string &username)

  Gets User from a database identifying him by username and returns as User's object.
- User getUser (uint32_t id)

  Gets User from a database identifying him by username and returns as User's object.
- bool updateUser (const User &user)

  Updates User in a database.
- bool deleteUser (const User &user)

  Deletes User from a database.
- bool deleteUser (uint32_t id)

  Deletes User using his ID from a database.
- std::vector< User > getAllUsers ()

  Returns all Users from a database.

5.4.1 Function Documentation

5.4.1.1 createUser()

```
bool UserManager::createUser (
                const User & user)
```

Creates User in a database.

Parameters

	user	User to create in a database

Returns

True if success, false if there was any error

5.4.1.2 deleteUser() [1/2]

```
bool UserManager::deleteUser (
                const User & user)
```

Deletes User from a database.

Parameters

	user	User to delete

Returns

True if success, false if User doesn't exist or there was any error

5.4.1.3 deleteUser() [2/2]

```
bool UserManager::deleteUser (
                    uint32_t id)
```

Deletes User using his ID from a database.

Parameters

	id	ID of a User to delete

Returns

True if success, false if User doesn't exist or there was any error

5.4.1.4 getAllUsers()

```
std::vector< User > UserManager::getAllUsers ()
```

Returns all Users from a database.

Returns

Vector of all Users from a database

5.4.1.5 getUser() [1/2]

```
User UserManager::getUser (
                    const std::string & username)
```

Gets User from a database identifying him by username and returns as User's object.

Parameters

	username	User's name

Returns

User's object, if User->id equals 0, then User doesn't exist, or there was an error

Parameters

	username	User's name

Returns

User's object, if User->id is 0, then User doesn't exist or there was an error

5.4.1.6 getUser() [2/2]

```
User UserManager::getUser (
                    uint32_t id)
```

Gets User from a database identifying him by username and returns as User's object.

Gets User from a database identifying him by ID and returns as User's object.

Parameters

	username	User's name

Returns

User's object, if User->id is 0, then User doesn't exist, or there was an error

Parameters

	id	User's ID

Returns

User's object, if User->id is 0, then User doesn't exist or there was an error

5.4.1.7 updateUser()

```
bool UserManager::updateUser (
                    const User & user)
```

Updates User in a database.

Parameters

	user	User to update

Returns

True if success, false if there was any error

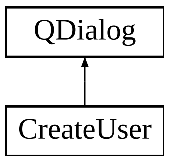
Chapter 6

Class Documentation

6.1 CreateUser Class Reference

#include <createuser.h>

Inheritance diagram for CreateUser:



Public Member Functions

- CreateUser (QWidget ∗parent=nullptr)
- ∼CreateUser ()

6.1.1 Constructor & Destructor Documentation

6.1.1.1 CreateUser()

```
CreateUser::CreateUser (
                QWidget ∗ parent = nullptr)
```

6.1.1.2 ∼CreateUser()

```
CreateUser::∼CreateUser ()
```

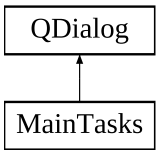
The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.cpp

6.2 MainTasks Class Reference

#include <maintasks.h>

Inheritance diagram for MainTasks:



Public Member Functions

- MainTasks (QWidget ∗parent=nullptr)
- ∼MainTasks ()

Protected Member Functions

- void resizeEvent (QResizeEvent ∗event) override
- void showEvent (QShowEvent ∗event) override

6.2.1 Constructor & Destructor Documentation

6.2.1.1 MainTasks()

MainTasks::MainTasks (
QWidget ∗ parent = nullptr) [explicit]

6.2.1.2 ∼MainTasks()

MainTasks::∼MainTasks ()

6.2.2 Member Function Documentation

6.2.2.1 resizeEvent()

void MainTasks::resizeEvent (
QResizeEvent ∗ event) [override], [protected]

6.2.2.2 showEvent()

void MainTasks::showEvent (
QShowEvent ∗ event) [override], [protected]

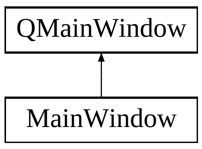
The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.cpp

6.3 MainWindow Class Reference

#include <mainwindow.h>

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget ∗parent=nullptr)
- ∼MainWindow ()

Static Public Attributes

- static User currentUser

6.3.1 Constructor & Destructor Documentation

6.3.1.1 MainWindow()

```
MainWindow::MainWindow (
                    QWidget ∗ parent = nullptr)
```

6.3.1.2 ∼MainWindow()

```
MainWindow::∼MainWindow ()
```

6.3.2 Member Data Documentation

6.3.2.1 currentUser

User MainWindow::currentUser [static]

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.cpp

6.4 Task Class Reference

#include <task.h>

Public Member Functions

- Task ()
- Task (uint32_t id)
- Task (uint32_t id, const std::string &name, const std::string &description, uint8_t priority, uint32↩ _t teamId, uint32_t userId, TaskStatus status, time_t deadline)
- Task (const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId, uint32_t userId, TaskStatus status, time_t deadline)
- void setId (uint64_t id)
- void setName (const std::string &name)
- void setDescription (const std::string &description)
- void setPriority (uint8_t priority)
- void setTeamId (uint32_t teamId)
- void setUserId (uint32_t userId)
- void setStatus (int status)
- void setStatus (TaskStatus status)
- void setDeadline (time_t deadline)
- uint64_t getId () const
- std::string getName () const
- std::string getDescription () const
- uint8_t getPriority () const
- uint32_t getTeamId () const
- uint32_t getUserId () const
- int getStatusAsInt () const
- TaskStatus getStatus () const
- time_t getDeadline () const

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Task() [1/4]

```
Task::Task ()
```

6.4.1.2 Task() [2/4]

```
Task::Task (
        uint32_t id)
```

6.4.1.3 Task() [3/4]

```
Task::Task (
        uint32_t id,
        const std::string & name,
        const std::string & description,
        uint8_t priority,
        uint32_t teamId,
        uint32_t userId,
        TaskStatus status,
        time_t deadline)
```

6.4.1.4 Task() [4/4]

```
Task::Task (
        const std::string & name,
        const std::string & description,
        uint8_t priority,
        uint32_t teamId,
        uint32_t userId,
        TaskStatus status,
        time_t deadline)
```

6.4.2 Member Function Documentation

6.4.2.1 getDeadline()

time_t Task::getDeadline () const

6.4.2.2 getDescription()

std::string Task::getDescription () const

6.4.2.3 getId()

uint64_t Task::getId () const

6.4.2.4 getName()

std::string Task::getName () const

6.4.2.5 getPriority()

uint8_t Task::getPriority () const

6.4.2.6 getStatus()

TaskStatus Task::getStatus () const

6.4.2.7 getStatusAsInt()

int Task::getStatusAsInt () const

6.4.2.8 getTeamId()

uint32_t Task::getTeamId () const

6.4.2.9 getUserId()

uint32_t Task::getUserId () const

6.4.2.10 setDeadline()

void Task::setDeadline (
time_t deadline)

6.4.2.11 setDescription()

void Task::setDescription (
const std::string & description)

6.4.2.12 setId()

void Task::setId (
uint64_t id)

6.4.2.13 setName()

void Task::setName (
const std::string & name)

6.4.2.14 setPriority()

void Task::setPriority (
uint8_t priority)

6.4.2.15 setStatus() [1/2]

void Task::setStatus (
int status)

6.4.2.16 setStatus() [2/2]

void Task::setStatus (
TaskStatus status)

6.4.2.17 setTeamId()

void Task::setTeamId (
uint32_t teamId)

6.4.2.18 setUserId()

```
void Task::setUserId (
            uint32_t userId)
```

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/task.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/task.cpp

6.5 Team Class Reference

#include <team.h>

Public Member Functions

- Team ()
- Team (uint32_t id, const std::string &name, std::string &password, const std::vector< uint32_t > &members)
- Team (uint32_t id, const std::string &name, const std::string &password, const std::vector< uint32_t > &members)
- Team (uint32_t id, const std::string &name, std::string &password)
- Team (uint32_t id, const std::string &name, const std::string &password)
- Team (const std::string &name, std::string &password, const std::vector< uint32_t > &members)
- Team (const std::string &name, const std::string &password, const std::vector< uint32_t > &members)
- void setId (uint32_t id)
- void setName (const std::string &name)
- void setPassword (const std::string &password)
- void setPassword (std::string &password)
- void setMembers (const std::vector< uint32_t > &members)
- uint32_t getId () const
- std::string getName () const
- std::string getPassword () const
- std::vector< uint32_t > getMembers () const
- std::vector< User > getMembersAsUsers () const
- bool containsUser (uint32_t userid) const
- bool containsUser (const User &user) const
- void addMember (const User &user)
- void addMember (uint32_t userid)
- void removeMember (const User &user)
- void removeMember (uint32_t userid)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 Team() [1/7]

```
Team::Team ()
```

6.5.1.2 Team() [2/7]

```
Team::Team (
        uint32_t id,
        const std::string & name,
        std::string & password,
        const std::vector< uint32_t > & members)
```

6.5.1.3 Team() [3/7]

```
Team::Team (
        uint32_t id,
        const std::string & name,
        const std::string & password,
        const std::vector< uint32_t > & members)
```

6.5.1.4 Team() [4/7]

```
Team::Team (
        uint32_t id,
        const std::string & name,
        std::string & password)
```

6.5.1.5 Team() [5/7]

```
Team::Team (
        uint32_t id,
        const std::string & name,
        const std::string & password)
```

6.5.1.6 Team() [6/7]

```
Team::Team (
        const std::string & name,
        std::string & password,
        const std::vector< uint32_t > & members)
```

6.5.1.7 Team() [7/7]

```
Team::Team (
        const std::string & name,
        const std::string & password,
        const std::vector< uint32_t > & members)
```

6.5.2 Member Function Documentation

6.5.2.1 addMember() [1/2]

```
void Team::addMember (
        const User & user)
```

6.5.2.2 addMember() [2/2]

void Team::addMember (
    uint32_t userid)

6.5.2.3 containsUser() [1/2]

bool Team::containsUser (
    const User & user) const

6.5.2.4 containsUser() [2/2]

bool Team::containsUser (
    uint32_t userid) const

6.5.2.5 getId()

uint32_t Team::getId () const

6.5.2.6 getMembers()

std::vector< uint32_t > Team::getMembers () const

6.5.2.7 getMembersAsUsers()

std::vector< User > Team::getMembersAsUsers () const

6.5.2.8 getName()

std::string Team::getName () const

6.5.2.9 getPassword()

std::string Team::getPassword () const

6.5.2.10 removeMember() [1/2]

void Team::removeMember (
    const User & user)

6.5.2.11 removeMember() [2/2]

void Team::removeMember (
    uint32_t userid)

6.5.2.12 setId()

```
void Team::setId (
        uint32_t id)
```

6.5.2.13 setMembers()

```
void Team::setMembers (
        const std::vector< uint32_t > & members)
```

6.5.2.14 setName()

```
void Team::setName (
        const std::string & name)
```

6.5.2.15 setPassword() [1/2]

```
void Team::setPassword (
        const std::string & password)
```

6.5.2.16 setPassword() [2/2]

```
void Team::setPassword (
        std::string & password)
```

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/team.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/team.cpp

6.6 User Class Reference

#include <user.h>

Public Member Functions

- User ()
- User (uint32_t id, const std::string &username, const std::string &password)
- User (uint32_t id, const std::string &username, const std::string &password, const QDate &creationDate)
- User (uint32_t id, const std::string &username, std::string &password)
- User (uint32_t id, const std::string &username, std::string &password, const QDate &creationDate)
- User (const std::string &username, const std::string &password)
- User (const std::string &username, const std::string &password, const QDate &creationDate)
- User (const std::string &username, std::string &password)
- User (const std::string &username, std::string &password, const QDate &creationDate)
- User (const std::string &username)
- void setId (uint32_t id)
- void setUsername (const std::string &username)
- void setPassword (const std::string &password)
- void setPassword (std::string &password)
- void setCreationDate (const QDate &creationDate)
- uint32_t getId () const
- std::string getUsername () const
- std::string getPassword () const
- QDate getCreationDate () const

6.6.1 Constructor & Destructor Documentation

6.6.1.1 User() [1/10]

```
User::User ()
```

6.6.1.2 User() [2/10]

```
User::User (
                uint32_t id,
                const std::string & username,
                const std::string & password)
```

6.6.1.3 User() [3/10]

```
User::User (
                uint32_t id,
                const std::string & username,
                const std::string & password,
                const QDate & creationDate)
```

6.6.1.4 User() [4/10]

```
User::User (
                uint32_t id,
                const std::string & username,
                std::string & password)
```

6.6.1.5 User() [5/10]

```
User::User (
                uint32_t id,
                const std::string & username,
                std::string & password,
                const QDate & creationDate)
```

6.6.1.6 User() [6/10]

```
User::User (
                const std::string & username,
                const std::string & password)
```

6.6.1.7 User() [7/10]

```
User::User (
                const std::string & username,
                const std::string & password,
                const QDate & creationDate)
```

6.6.1.8 User() [8/10]

```
User::User (
        const std::string & username,
        std::string & password)
```

6.6.1.9 User() [9/10]

```
User::User (
        const std::string & username,
        std::string & password,
        const QDate & creationDate)
```

6.6.1.10 User() [10/10]

```
User::User (
        const std::string & username)  [explicit]
```

6.6.2 Member Function Documentation

6.6.2.1 getCreationDate()

```
QDate User::getCreationDate () const
```

6.6.2.2 getId()

```
uint32_t User::getId () const
```

6.6.2.3 getPassword()

```
std::string User::getPassword () const
```

6.6.2.4 getUsername()

```
std::string User::getUsername () const
```

6.6.2.5 setCreationDate()

```
void User::setCreationDate (
        const QDate & creationDate)
```

6.6.2.6 setId()

```
void User::setId (
        uint32_t id)
```

6.6.2.7 setPassword() [1/2]

```
void User::setPassword (
                    const std::string & password)
```

6.6.2.8 setPassword() [2/2]

```
void User::setPassword (
                    std::string & password)
```

6.6.2.9 setUsername()

```
void User::setUsername (
                    const std::string & username)
```

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/user.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/user.cpp

Chapter 7

File Documentation

7.1 C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.cpp File Reference

#include "createuser.h"
#include "ui_createuser.h"

7.2 C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.h File Reference

#include "user.h"
#include "usermanager.h"

Classes

- class CreateUser

Namespaces

- namespace Ui

7.3 createuser.h

Go to the documentation of this file.

```
#ifndef CREATEUSER_H
#define CREATEUSER_H

#include "user.h"
#include "usermanager.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class CreateUser;
}
QT_END_NAMESPACE

class CreateUser : public QDialog
{
    Q_OBJECT

public:
    CreateUser(QWidget *parent = nullptr);
    ~CreateUser();

private slots:

    // Event handlers

    // Click handlers

    void on_createNewAccountButton_clicked();

private:
    Ui::CreateUser *ui;
    bool validateInput();

    // Deprecated
    // Consider using UserManager::createUser()
    bool createUserInDatabase(const User& user);
};

#endif // CREATEUSER_H
```

7.4 C:/Users/Krzysztof/CLionProjects/To-Do-List/main.cpp File Reference

#include "mainwindow.h"
#include <QApplication>
#include <QIcon>
#include <QFile>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>

Functions

- int main (int argc, char ∗argv[])

7.4.1 Function Documentation

7.4.1.1 main()

```
int main (
        int argc,
        char ∗ argv[])
```

7.5 C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.cpp File Reference

#include ”maintasks.h”
#include ”ui_maintasks.h”
#include ”mainwindow.h”
#include ”taskmanager.h”
#include <QListWidgetItem>
#include <QSqlQuery>
#include <QSqlError>
#include ”teammanager.h”
#include <QStringListModel>
#include <QInputDialog>

7.6 C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.h File Reference

#include <QListWidgetItem>
#include <QDialog>
#include <QTimer>
#include <QTime>
#include <QSoundEffect>
#include <QDateTime>
#include <algorithm>
#include <QBrush>

Classes

- class MainTasks

Namespaces

- namespace Ui

7.7 maintasks.h

Go to the documentation of this file.

```
#ifndef MAINTASKS_H
#define MAINTASKS_H
#include <QListWidgetItem>
#include <QDialog>
#include <QTimer>
#include <QTime>
#include <QSoundEffect>
#include <QDateTime>
#include <algorithm>
#include <QBrush>

namespace Ui {
class MainTasks;
}

```

```
class MainTasks : public QDialog
{
    Q_OBJECT

public:
    explicit MainTasks(QWidget *parent = nullptr);
    ~MainTasks();

private slots:

    // Event handlers

    // Click handlers

    void on_startPomodoroButton_clicked();

    void on_pomodoroButton_clicked();

    void on_shortBreakButton_clicked();

    void on_longBreakButton_clicked();

    void on_addTaskButton_clicked();

    void on_cancelNewTaskButton_clicked();

    void on_updatePasswordButton_clicked();

    void on_removeAccountButton_clicked();

    void on_confirmTaskAddButton_clicked();

    void on_taskListDisplay_itemDoubleClicked(QListWidgetItem *item);

    void on_createTeamButton_clicked();

    void on_addMembersButton_clicked();

    void on_crateTeamCancelButton_clicked();

    void on_addMemberCancelButton_clicked();

    void on_leaveJoinTeamButton_clicked();

    void on_createTeamConfirmButton_clicked();

    void on_allTeamsComboBox_currentIndexChanged(int index);

    void on_addMemberConfimButton_clicked();

    void on_sortTasksComboBox_currentIndexChanged(int index);

    // Others

    void updateDisplay();

    void setDisplay(int time);

    void setTimer(int time);

    void refreshTaskList();

    void updateProfileStats();

    void moveAddTaskButton();

private:
    Ui::MainTasks *ui;

    //Timer
    QTimer *pomodoroTimer;
    int remainingTime;
    int startingTime;
    bool isRunning = false;
    QSoundEffect *timerEndSound;

    //Add task button
    QPushButton *addTaskButton;

    void loadAllTeamsToComboBox();

    // Task Sorting
    enum TaskSortCriteria { // «< Updated enum
        SortByDueDateAsc,
        SortByDueDateDesc,
        SortByNameAsc,
        SortByNameDesc
    };
    TaskSortCriteria currentTaskSortCriteria;

protected:
    void resizeEvent(QResizeEvent *event) override;
    void showEvent(QShowEvent *event) override;
};

#endif // MAINTASKS_H
```

7.8 C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.cpp File Reference

#include ”mainwindow.h”
#include ”./ui_mainwindow.h”
#include ”maintasks.h”
#include <QPixmap>
#include ”createuser.h”

7.9 C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.h File Reference

#include <QMainWindow>
#include ”maintasks.h”
#include ”createuser.h”
#include ”user.h”

Classes

- class MainWindow

Namespaces

- namespace Ui

7.10 mainwindow.h

Go to the documentation of this file.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "maintasks.h"
#include "createuser.h"
#include "user.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
```

```
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    static User currentUser; // Static member to store current logged user

private slots:

    // Event handlers

    // Click handlers

    void on_loginButton_clicked();
    void on_registerButton_clicked();

private:
    Ui::MainWindow *ui;
    MainTasks *taskWindow;
    CreateUser *createUserWindow;

    bool authenticateUser(const QString& username, const QString& password);

    User getCurrentUser() const;


};

#endif // MAINWINDOW_H
```

7.11 C:/Users/Krzysztof/CLionProjects/To-Do-List/task.cpp File Reference

#include "task.h"

7.12 C:/Users/Krzysztof/CLionProjects/To-Do-List/task.h File Reference

#include <cstdint>
#include <string>
#include "taskstatus.h"

Classes

- class Task

7.13 task.h

Go to the documentation of this file.

```
#ifndef TASK_H
#define TASK_H

#include <cstdint>
#include <string>

#include "taskstatus.h"

class Task
```

```
{
private:
    uint64_t id;
    std::string name;
    std::string description;
    uint8_t priority;
    uint32_t teamId; // If teamId is 0, that means it's a Task specified only for one User
    uint32_t userId; // If userId is 0, that means it's Task specified for Team
    TaskStatus status;
    time_t deadline; // If 0 - Task with unlimited time.
public:
    // Constructors
    Task();
    Task(uint32_t id);
    Task(uint32_t id, const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId, uint32_t
userId, TaskStatus status, time_t deadline);
    Task(const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId, uint32_t userId,
TaskStatus status, time_t deadline);

    // Setters
    void setId(uint64_t id);
    void setName(const std::string &name);
    void setDescription(const std::string &description);
    void setPriority(uint8_t priority);
    void setTeamId(uint32_t teamId);
    void setUserId(uint32_t userId);
    void setStatus(int status);
    void setStatus(TaskStatus status);
    void setDeadline(time_t deadline);

    // Getters
    uint64_t getId() const;
    std::string getName() const;
    std::string getDescription() const;
    uint8_t getPriority() const;
    uint32_t getTeamId() const;
    uint32_t getUserId() const;
    int getStatusAsInt() const;
    TaskStatus getStatus() const;
    time_t getDeadline() const;

};

#endif // TASK_H
```

7.14 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskmanager.cpp File Reference

#include "taskmanager.h"
#include <QSqlQuery>
#include <QSqlError>
#include <QMessageBox>
#include <QDateTime>
#include <vector>

7.15 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskmanager.h File Reference

#include "task.h"
#include <vector>

Namespaces

- namespace TaskManager

Functions

- bool TaskManager::createTask (const Task &task)

    Creates Task in a database.
- std::vector< Task > TaskManager::getTasksForTeam (uint32_t teamId)

    Returns all Team's Task.
- std::vector< Task > TaskManager::getTasksForUser (uint32_t userId)

    Returns all User's Task.
- Task TaskManager::getTask (uint64_t id)

    Gets Task from a database identifying it by ID and returns as Task's object.
- bool TaskManager::updateTask (const Task &task)

    Updates Task in a database.
- bool TaskManager::deleteTask (const Task &task)

    Deletes Task from a database.

7.16 taskmanager.h

Go to the documentation of this file.

```
#ifndef TASKMANAGER_H
#define TASKMANAGER_H
#include "task.h"
#include <vector>

namespace TaskManager {

    bool createTask(const Task& task);

    std::vector<Task> getTasksForTeam(uint32_t teamId);

    std::vector<Task> getTasksForUser(uint32_t userId);

    Task getTask(uint64_t id);

    bool updateTask(const Task& task);

    bool deleteTask(const Task& task);

}

#endif // TASKMANAGER_H
```

7.17 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskstatus.cpp File Reference

#include "taskstatus.h"

Functions

- TaskStatus getTaskStatus (int status)
- int getTaskStatusInt (TaskStatus status)

7.17.1 Function Documentation

7.17.1.1 getTaskStatus()

TaskStatus getTaskStatus (
    int status)

7.17.1.2 getTaskStatusInt()

int getTaskStatusInt (
    TaskStatus status)

7.18 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskstatus.h File Reference

Enumerations

- enum TaskStatus { DONE , IN_PROGRESS , NOT_DONE }

Functions

- TaskStatus getTaskStatus (int status)
- int getTaskStatusInt (TaskStatus status)

7.18.1 Enumeration Type Documentation

7.18.1.1 TaskStatus

enum TaskStatus

Enumerator

	DONE
	IN_PROGRESS
	NOT_DONE

7.18.2 Function Documentation

7.18.2.1 getTaskStatus()

TaskStatus getTaskStatus (
    int status)

7.18.2.2 getTaskStatusInt()

```
int getTaskStatusInt (
            TaskStatus status)
```

7.19 taskstatus.h

Go to the documentation of this file.

```
#ifndef TASKSTATUS_H
#define TASKSTATUS_H

enum TaskStatus {
    DONE,
    IN_PROGRESS,
    NOT_DONE
};

TaskStatus getTaskStatus(int status);

int getTaskStatusInt(TaskStatus status);

#endif // TASKSTATUS_H
```

7.20 C:/Users/Krzysztof/CLionProjects/To-Do-List/team.cpp File Reference

#include ”team.h”

7.21 C:/Users/Krzysztof/CLionProjects/To-Do-List/team.h File Reference

#include <cstdint>
#include <string>
#include <vector>
#include <QCryptographicHash>
#include ”user.h”
#include ”usermanager.h”

Classes

- class Team

7.22 team.h

Go to the documentation of this file.

```
#ifndef TEAM_H
#define TEAM_H

#include <cstdint>
#include <string>
#include <vector>
#include <QCryptographicHash>

#include "user.h"
#include "usermanager.h"

class Team
{
private:
    uint32_t id;
    std::string name;
    std::string password;
    std::vector<uint32_t> members;
public:
    // Constructors
    Team();
    Team(uint32_t id, const std::string &name, std::string &password, const std::vector<uint32_t> &members);
    Team(uint32_t id, const std::string &name, const std::string &password, const std::vector<uint32_t> &members);
    Team(uint32_t id, const std::string &name, std::string &password);
    Team(uint32_t id, const std::string &name, const std::string &password);
    Team(const std::string &name, std::string &password, const std::vector<uint32_t> &members);
    Team(const std::string &name, const std::string &password, const std::vector<uint32_t> &members);

    // Setters
    void setId(uint32_t id);
    void setName(const std::string &name);
    void setPassword(const std::string &password);
    void setPassword(std::string &password);
    void setMembers(const std::vector<uint32_t> &members);

    // Getters
    uint32_t getId() const;
    std::string getName() const;
    std::string getPassword() const;
    std::vector<uint32_t> getMembers() const;
    std::vector<User> getMembersAsUsers() const;
    bool containsUser(uint32_t userid) const;
    bool containsUser(const User &user) const;

    // Adding members
    void addMember(const User &user);
    void addMember(uint32_t userid);
    void removeMember(const User &user);
    void removeMember(uint32_t userid);

};

#endif // TEAM_H
```

7.23 C:/Users/Krzysztof/CLionProjects/To-Do-List/teammanager.cpp File Reference

#include "teammanager.h"

7.24 C:/Users/Krzysztof/CLionProjects/To-Do-List/teammanager.h File Reference

#include "team.h"
#include <string>
#include <QMessageBox>

#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

Namespaces

- namespace TeamManager

Functions

- bool TeamManager::createTeam (const Team &team)

  Creates Team in a database.
- Team TeamManager::getTeam (const std::string &name)

  Gets Team from a database identifying it by name and returns as Team's object.
- Team TeamManager::getTeam (uint32_t id)

  Gets Team from a database identifying it by ID and returns as Team's object.
- bool TeamManager::updateTeam (const Team &team)

  Updates Team in a database.
- bool TeamManager::deleteTeam (const Team &team)

  Deletes Team from a database.
- bool TeamManager::deleteTeam (uint32_t id)

  Deletes Team using his ID from a database.
- std::vector< Team > TeamManager::getAllTeams ()

  Returns all Teams from a database.
- std::vector< Team > TeamManager::getTeamsForUser (uint32_t userId)

  Returns all Teams that the User belongs to from a database.
- Team TeamManager::getTeamForUser (uint32_t userId)

  Returns first User's Team from a database.

7.25 teammanager.h

Go to the documentation of this file.

```
#ifndef TEAMMANAGER_H
#define TEAMMANAGER_H
#include "team.h"

#include <string>
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

namespace TeamManager {

    bool createTeam(const Team& team);

    Team getTeam(const std::string& name);

    Team getTeam(uint32_t id);

    bool updateTeam(const Team& team);

    bool deleteTeam(const Team& team);

    bool deleteTeam(uint32_t id);


    std::vector<Team> getAllTeams();

    std::vector<Team> getTeamsForUser(uint32_t userId);

    Team getTeamForUser(uint32_t userId);



}

#endif // TEAMMANAGER_H
```

7.26 C:/Users/Krzysztof/CLionProjects/To-Do-List/user.cpp File Reference

#include ”user.h”

7.27 C:/Users/Krzysztof/CLionProjects/To-Do-List/user.h File Reference

#include <cstdint>
#include <string>
#include <QDate>
#include <QCryptographicHash>

Classes

- class User

7.28 user.h

Go to the documentation of this file.

```
#ifndef USER_H
#define USER_H

#include <cstdint>
#include <string>
#include <QDate>
#include <QCryptographicHash>

class User
{
private:
    uint32_t id;
    std::string username;
    std::string password;
    QDate creationDate;
public:
    // Constructors
    User();
    User(uint32_t id, const std::string& username, const std::string &password);
    User(uint32_t id, const std::string& username, const std::string &password, const QDate &creationDate);
    User(uint32_t id, const std::string& username, std::string &password);
    User(uint32_t id, const std::string& username, std::string &password, const QDate &creationDate);
    User(const std::string& username, const std::string &password);
    User(const std::string& username, const std::string &password, const QDate &creationDate);
    User(const std::string& username, std::string &password);
    User(const std::string& username, std::string &password, const QDate &creationDate);
    explicit User(const std::string& username);

    // Setters
    void setId(uint32_t id);
    void setUsername(const std::string &username);
    void setPassword(const std::string &password);
    void setPassword(std::string &password);
    void setCreationDate(const QDate &creationDate);

    // Getters
    uint32_t getId() const;
    std::string getUsername() const; // Make const
    std::string getPassword() const;
    QDate getCreationDate() const;
};

#endif // USER_H
```

7.29 C:/Users/Krzysztof/CLionProjects/To-Do-List/usermanager.cpp File Reference

#include ”usermanager.h”
#include <vector>

7.30 C:/Users/Krzysztof/CLionProjects/To-Do-List/usermanager.h File Reference

#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include ”user.h”

Namespaces

- namespace UserManager

Functions

- bool UserManager::createUser (const User &user)

  Creates User in a database.
- User UserManager::getUser (const std::string &username)

  Gets User from a database identifying him by username and returns as User's object.
- User UserManager::getUser (uint32_t id)

  Gets User from a database identifying him by username and returns as User's object.
- bool UserManager::updateUser (const User &user)

  Updates User in a database.
- bool UserManager::deleteUser (const User &user)

  Deletes User from a database.
- bool UserManager::deleteUser (uint32_t id)

  Deletes User using his ID from a database.
- std::vector< User > UserManager::getAllUsers ()

  Returns all Users from a database.

7.31 usermanager.h

Go to the documentation of this file.

```
#ifndef USERMANAGER_H
#define USERMANAGER_H
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

#include "user.h"

namespace UserManager {
    bool createUser(const User& user);

    User getUser(const std::string& username);

    User getUser(uint32_t id);

    bool updateUser(const User& user);

    bool deleteUser(const User& user);

    bool deleteUser(uint32_t id);

    std::vector<User> getAllUsers();


}

#endif // USERMANAGER_H
```