Stack Scraper

Technical Documentation

Section:
Aleksander Sykulski
Jakub Kromołowski
Jakub Zając
Konrad Kobielus
Michał Brodziak
INF SII sem. VI

1. System Assumptions

The main objective of the project is to create a database-based information system – a console application for browsing existing questions on Stack Overflow. The application aims to speed up the search for topics by automatically opening the found threads. An additional enhancement is the ability to browse topics from favorite categories and save the user's history. For user convenience, the found conversations are organized in a clear way, and code blocks are highlighted according to common standards.

2. Non-functional requirements

The application provides an intuitive and easy-to-use text interface. The displayed threads should be clearly organized, and the included code snippets should be appropriately highlighted. Additionally, to enhance functionality, the history of browsed threads is continuously saved. The user can also save favorite threads and topic categories. Using an Intel i3 8100 processor, 8GB of RAM, and the Windows 10 operating system, the application should generate responses in less than 5 seconds.

3. Functional requirements

The purpose of the application is to display threads posted on the Stack Overflow platform. The application is written in C++ and then compiled using the cross-platform tool CMake. For better library management, we use the Conan tool. The SQLite3 library is used to connect to the database. We use cpr library for getting JSON from URL. Then nlohmann_json library is responsible for reading JSON text.

4. Use case list

1) Creating account

Description: Creating a new user account to use an application.
Actor: Unlogged user
Preconditions: User is signed out or does not have an account.
Triggers: User creates an account.

2) Logging

Description: Sign in user to the application.
Actor: Unlogged user
Preconditions: User is signed out and has an account.
Triggers: User logs in.

3) Looking for problem by phrase or tag

Description: User after logging in or creating account searches a problem by phrase or tag.
Actor: Logged user
Preconditions: User is logged in.
Triggers: User checks the code for his problem.

4) History reviewing

Description: User checks history of searched problems.
Actor: Logged user

Preconditions: User is logged in and searched for at least a single problem.
Triggers: User checks listed history of searched phrases.

5) Adding to the favorite

Description: User adds favorite phrase.
Actor: Logged user
Preconditions: User is signed in, looked for at least one phrase.
Triggers: User adds favorite phrases or tags to the list.

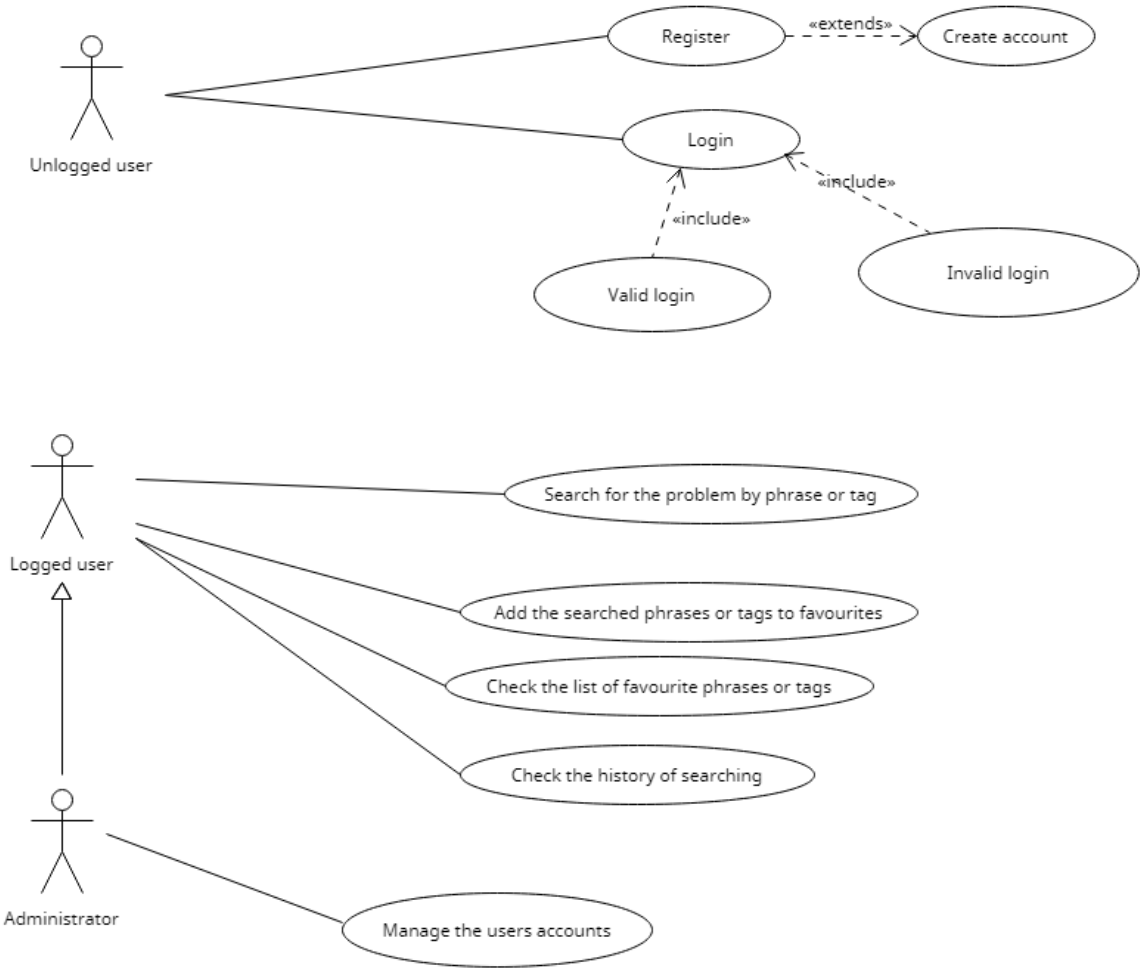
6) Reviewing favorite phrases or tags

Description: Reviewing favorite phrases or tags which one is added to favorite section.
Actor: Logged user
Preconditions: User is signed in and added at least single phrase or tag to favorite.
Triggers: User checks favorite phrases or tags.

7) Management of account

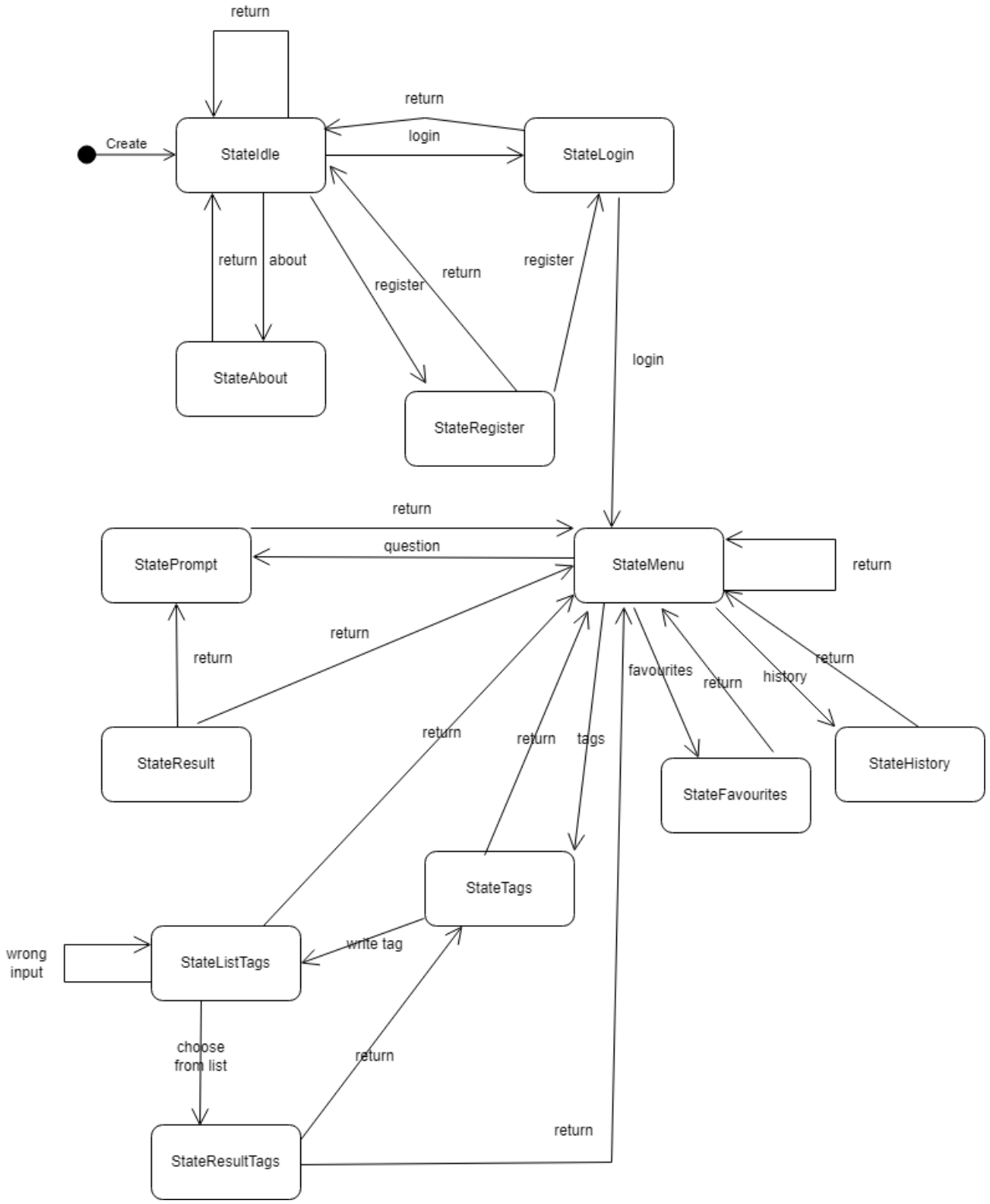
Description: User manages users accounts.
Actor: Administrator
Preconditions: User is logged as administrator.
Triggers: User creates, edits or deletes user accounts.

5. Use case diagram



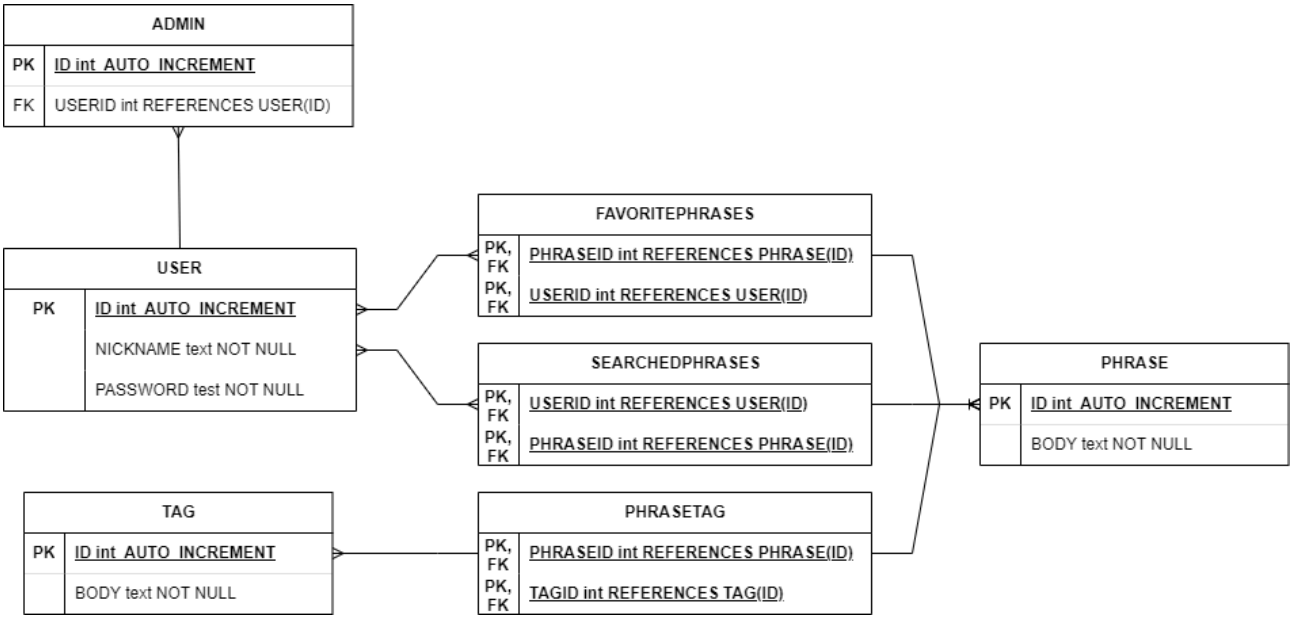
Img 5.1 Use case diagram

6. State diagram



Img 6.1 State diagram

7. Logical data mode



Img 7.1 Logical data model

8. Internal specification

The internal specification was generated with doxygen and is attached at the end of this document.

9. External specification

To run the application, the user opens the .exe file, which launches a new terminal. The application commands are entered via the keyboard with the appropriate parameters. Upon starting the application, the user is prompted to log in to retrieve their browsing history and favorite results. If the user encounters any difficulties with the program, they can access a full description of the functions by entering the command `help`.

10. Conclusions

The project taught us how to work as a team on a shared application. Clear communication and the division of individual tasks were essential for this purpose. Additionally, we learned techniques for using a database in the application to store grouped data. Such solutions allow for better organization and broader processing of the data. During the development of the application, the big challenge was connecting the libraries for database management. We resolved this issue by using the Conan tool and the SQLite3 library.

StackScraper

v1.0.0

Generated by Doxygen 1.11.0

Chapter 1

README

Purpose: Have you find yourself distracted with some nonesense errors while developing your new-fresh TO-DO List? Now you don't have to exit focus mode to type error in google, instead you can paste it in StackScraper. StackScraper is console based application developed in C++ and CMake. With StackScraper you can paste error, and get instant respond with question related to your problem from StackOverflow. Question on itself won't help you much, that is why we also include answers ;)

Installation: Currently working on 1st version of app

Development: For development of app, you need C++ compiler with CMake (VSC with MinGW or other compiler and CMake addon or CLion). Also, as we use Conan package manager you need Python 3.6 (or newer) and installed Conan. Follow https://www.jetbrains.com/help/clion/conan-plugin.html for CLion setup with Conan. Rest of libs is downloaded runtime with CMake itself.

Coding conventions: functions - PascalCase rest - camelCase

Comments: Doxxygen: multiline comment:

/**

- **Returns**
- */

singleline comment: ///

inline comment: ///<

Recent documentation: https://michalshy.github.io/StackScraperDocumentation/

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 AboutTexts Namespace Reference

namespace for About state

6.1.1 Detailed Description

namespace for About state

6.2 cmd Namespace Reference

CMD - Namespace responsible for holding globals connected to shell application.

6.2.1 Detailed Description

CMD - Namespace responsible for holding globals connected to shell application.

File that holds all globals of the program

6.3 conanfile Namespace Reference

Classes

- class ConanApplication

6.4 FavouriteTexts Namespace Reference

namespace for Favourites state

6.4.1 Detailed Description

namespace for Favourites state

6.5 HistoryTexts Namespace Reference

namespace for History state

6.5.1 Detailed Description

namespace for History state

6.6 IdleTexts Namespace Reference

namespace for Idle state

6.6.1 Detailed Description

namespace for Idle state

File which contains namespaces of strings to use by all states of the program Every state has its own namespace

6.7 ListState Namespace Reference

namespace for List state

6.7.1 Detailed Description

namespace for List state

6.8 LoginTexts Namespace Reference

namespace for Login state

6.8.1 Detailed Description

namespace for Login state

6.9 Manual Namespace Reference

namespace for manual

6.9.1 Detailed Description

namespace for manual

6.10 MenuTexts Namespace Reference

namespace for Menu state

6.10.1 Detailed Description

namespace for Menu state

6.11 PromptTexts Namespace Reference

namespace for Prompt state

6.11.1 Detailed Description

namespace for Prompt state

6.12 RegisterTexts Namespace Reference

namespace for Register state

6.12.1 Detailed Description

namespace for Register state

6.13 ResultTexts Namespace Reference

namespace for Result state

6.13.1 Detailed Description

namespace for Result state

6.14 Syntax Namespace Reference

6.14.1 Detailed Description

Contains syntax keywords for all supported programming languages

6.15 TagsTexts Namespace Reference

namespace for Tags state

6.15.1 Detailed Description

namespace for Tags state

6.16 TextColors Namespace Reference

Viable colors of the text.

6.16.1 Detailed Description

Viable colors of the text.

File which contains all text based operations and variables of the application

6.17 TextFunctions Namespace Reference

6.17.1 Detailed Description

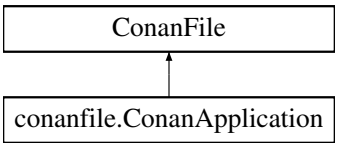
Namespace for functions which adjust printing of texts

Chapter 7

Class Documentation

7.1 conanfile.ConanApplication Class Reference

Inheritance diagram for conanfile.ConanApplication:



Public Member Functions

- layout (self)
- generate (self)
- requirements (self)

Static Public Attributes

- str package_type = "application"
- str settings = "os", "compiler", "build_type", "arch"
- str generators = "CMakeDeps"

7.1.1 Detailed Description

Definition at line 10 of file conanfile.py.

7.1.2 Member Function Documentation

7.1.2.1 generate()

```
conanfile.ConanApplication.generate (
              self)
```

Definition at line 19 of file conanfile.py.

7.1.2.2 layout()

```
conanfile.ConanApplication.layout (
              self)
```

Definition at line 16 of file conanfile.py.

7.1.2.3 requirements()

```
conanfile.ConanApplication.requirements (
              self)
```

Definition at line 24 of file conanfile.py.

7.1.3 Member Data Documentation

7.1.3.1 generators

```
str conanfile.ConanApplication.generators = "CMakeDeps"  [static]
```

Definition at line 13 of file conanfile.py.

7.1.3.2 package_type

```
str conanfile.ConanApplication.package_type = "application"  [static]
```

Definition at line 11 of file conanfile.py.

7.1.3.3 settings

```
str conanfile.ConanApplication.settings = "os", "compiler", "build_type", "arch"  [static]
```

Definition at line 12 of file conanfile.py.

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

- conanfile.py

7.2 DBmanager Class Reference

```
#include <DBmanager.hpp>
```

Public Member Functions

- bool insertUser (std::string &nickname, std::string &password)
- std::vector< std::pair< std::string, std::string > > getUsers ()
- bool updateUserPassword (int id, std::string &password)
- bool deleteUser (int id)
- bool loginUser (std::string &log, std::string &pass)
- bool insertAdmin (int Id)
- std::vector< std::pair< std::string, std::string > > getAdmins ()
- bool deleteAdmin (int adminId)
- bool insertPhrase (std::string &body, std::string &response)
- std::vector< std::pair< std::string, std::string > > getPhrases ()
- std::vector< std::pair< std::string, std::string > > getPhrase (int phraseId)
- bool deletePhrase (int id)
- bool insertTag (std::string &body)
- std::vector< std::pair< std::string, std::string > > getTags ()
- bool deleteTag (int id)
- bool insertFavourite (int phraseId)
- std::vector< std::pair< std::string, std::string > > getFavourites ()
- bool deleteFavourite (int favId)
- bool connectTagToPhrase (int phraseId, int tagId)
- std::vector< std::pair< std::string, std::string > > getPhraseWithTag ()
- DBmanager ()
- ~DBmanager ()

7.2.1 Detailed Description

Controls usage of database in the application Most of the functions are self-explanatory

Definition at line 17 of file DBmanager.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 DBmanager()

```
DBmanager::DBmanager ()
```

Definition at line 409 of file DBmanager.cpp.

7.2.2.2 ~DBmanager()

```
DBmanager::~DBmanager ()
```

Definition at line 421 of file DBmanager.cpp.

7.2.3 Member Function Documentation

7.2.3.1 connectTagToPhrase()

```
bool DBmanager::connectTagToPhrase (
            int phraseId,
            int tagId)
```

Definition at line 364 of file DBmanager.cpp.

7.2.3.2 deleteAdmin()

```
bool DBmanager::deleteAdmin (
            int adminId)
```

Definition at line 220 of file DBmanager.cpp.

7.2.3.3 deleteFavourite()

```
bool DBmanager::deleteFavourite (
            int favId)
```

Definition at line 348 of file DBmanager.cpp.

7.2.3.4 deletePhrase()

```
bool DBmanager::deletePhrase (
            int id)
```

Definition at line 273 of file DBmanager.cpp.

7.2.3.5 deleteTag()

```
bool DBmanager::deleteTag (
            int id)
```

Definition at line 310 of file DBmanager.cpp.

7.2.3.6 deleteUser()

```
bool DBmanager::deleteUser (
            int id)
```

Definition at line 180 of file DBmanager.cpp.

7.2.3.7 getAdmins()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getAdmins ()
```

Definition at line 207 of file DBmanager.cpp.

7.2.3.8 getFavourites()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getFavourites ()
```

Definition at line 335 of file DBmanager.cpp.

7.2.3.9 getPhrase()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getPhrase (
            int phraseId)
```

Definition at line 260 of file DBmanager.cpp.

7.2.3.10 getPhrases()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getPhrases ()
```

Definition at line 247 of file DBmanager.cpp.

7.2.3.11 getPhraseWithTag()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getPhraseWithTag ()
```

Definition at line 376 of file DBmanager.cpp.

7.2.3.12 getTags()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getTags ()
```

Definition at line 297 of file DBmanager.cpp.

7.2.3.13 getUsers()

```
std::vector< std::pair< std::string, std::string > > DBmanager::getUsers ()
```

Definition at line 151 of file DBmanager.cpp.

7.2.3.14 insertAdmin()

```
bool DBmanager::insertAdmin (
            int Id)
```

Definition at line 193 of file DBmanager.cpp.

7.2.3.15 insertFavourite()

```
bool DBmanager::insertFavourite (
            int phraseId)
```

Definition at line 322 of file DBmanager.cpp.

7.2.3.16 insertPhrase()

```
bool DBmanager::insertPhrase (
            std::string & body,
            std::string & response)
```

Definition at line 232 of file DBmanager.cpp.

7.2.3.17 insertTag()

```
bool DBmanager::insertTag (
            std::string & body)
```

Definition at line 285 of file DBmanager.cpp.

7.2.3.18 insertUser()

```
bool DBmanager::insertUser (
            std::string & nickname,
            std::string & password)
```

Definition at line 138 of file DBmanager.cpp.

7.2.3.19 loginUser()

```
bool DBmanager::loginUser (
            std::string & log,
            std::string & pass)
```

Definition at line 389 of file DBmanager.cpp.

7.2.3.20 updateUserPassword()

```
bool DBmanager::updateUserPassword (
            int id,
            std::string & password)
```

Definition at line 165 of file DBmanager.cpp.

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

- Logic/Database/DBmanager.hpp
- Logic/Database/DBmanager.cpp

7.3 Engine Class Reference

```
#include <Engine.hpp>
```

Public Member Functions

- Engine ()
  Default constructor.
- void Run ()
  Function which executes start of state machine.

7.3.1 Detailed Description

Engine of the program Responsible for handling init of StateMachine

Definition at line 17 of file Engine.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Engine()

```
Engine::Engine ()
```

Default constructor.

Function which inits state machine and all of it's states Sets first state to IDLE

Definition at line 18 of file Engine.cpp.

7.3.3 Member Function Documentation

7.3.3.1 Run()

```
void Engine::Run ()
```

Function which executes start of state machine.

Function which executes first onUpdate of state

Definition at line 10 of file Engine.cpp.

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

- Engine.hpp
- Engine.cpp

7.4 FiniteStateMachine< T > Class Template Reference

```
#include <StateMachine.hpp>
```

Public Member Functions

- FiniteStateMachine ()
  Default constructor.
- template<class S >
  State< T > & Add (T id)
- State< T > & GetState (T stateID)
- State< T > & GetCurrentState ()
- const State< T > & GetCurrentState () const
- void SetCurrentState (T stateID)
- void OnUpdate ()

Protected Member Functions

- void SetCurrentState (State< T > *state)

Protected Attributes

- std::map< T, std::unique_ptr< State< T > > > mStates
  map of states
- State< T > * mCurrentState
  pointer to current state

7.4.1 Detailed Description

template<typename T>
class FiniteStateMachine< T >

Pre-definition of FSM class

Template class of state machine

Template Parameters

T	template class for which state machine is created

Definition at line 28 of file StateMachine.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 FiniteStateMachine()

```
template<typename T >
FiniteStateMachine< T >::FiniteStateMachine ()  [inline]
```

Default constructor.

Definition at line 35 of file StateMachine.hpp.

7.4.3 Member Function Documentation

7.4.3.1 Add()

```
template<typename T >
template<class S >
State< T > & FiniteStateMachine< T >::Add (
            T id)  [inline]
```

Function which adds new state to the map

Template Parameters

S	template class, should be chosen accordingly to whole state machine

Parameters

id	id of the freshly added state

Returns

pointer on the new state

Definition at line 45 of file StateMachine.hpp.

7.4.3.2 GetCurrentState() [1/2]

```
template<typename T >
State< T > & FiniteStateMachine< T >::GetCurrentState ()  [inline]
```

Function for returning current state

Returns

current state

Definition at line 64 of file StateMachine.hpp.

7.4.3.3 GetCurrentState() [2/2]

```
template<typename T >
const State< T > & FiniteStateMachine< T >::GetCurrentState () const  [inline]
```

Function for returning current state

Returns

current state

Definition at line 73 of file StateMachine.hpp.

7.4.3.4 GetState()

```
template<typename T >
State< T > & FiniteStateMachine< T >::GetState (
            T stateID)  [inline]
```

Function for returning interesting state

Parameters

stateID	identification of desired state

Returns

desired state

Definition at line 56 of file StateMachine.hpp.

7.4.3.5 OnUpdate()

```
template<typename T >
void FiniteStateMachine< T >::OnUpdate ()  [inline]
```

Function called to change state

Definition at line 90 of file StateMachine.hpp.

7.4.3.6 SetCurrentState() [1/2]

```
template<typename T >
void FiniteStateMachine< T >::SetCurrentState (
            State< T > * state)  [inline], [protected]
```

Protected function which is used by public setCurrentState to change state for existing instead of creating multiple instances of same state

Parameters

state	

Definition at line 103 of file StateMachine.hpp.

7.4.3.7 SetCurrentState() [2/2]

```
template<typename T >
void FiniteStateMachine< T >::SetCurrentState (
            T stateID)  [inline]
```

Function for changing state as desired

Parameters

stateID	id of desired state

Definition at line 82 of file StateMachine.hpp.

7.4.4 Member Data Documentation

7.4.4.1 mCurrentState

```
template<typename T >
State<T>* FiniteStateMachine< T >::mCurrentState  [protected]
```

pointer to current state

Definition at line 32 of file StateMachine.hpp.

7.4.4.2 mStates

```
template<typename T >
std::map<T, std::unique_ptr<State<T> > > FiniteStateMachine< T >::mStates  [protected]
```

map of states

Definition at line 31 of file StateMachine.hpp.

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

- FSM/State.hpp
- FSM/StateMachine.hpp

7.5 PromptSingleton Class Reference

```
#include <PromptSingleton.hpp>
```

Public Member Functions

- PromptSingleton (const PromptSingleton &obj)=delete
- void SetValues (std::string &val)
- std::string RetValues ()
- void GetPrompt ()
- void GetPromptAuto (std::vector< std::string > dict)

Static Public Member Functions

- static PromptSingleton * GetInstance ()

7.5.1 Detailed Description

Singleton class which contains prompt typed by user, so application can control what is actual prompt

Definition at line 15 of file PromptSingleton.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 PromptSingleton()

```
PromptSingleton::PromptSingleton (
            const PromptSingleton & obj)  [delete]
```

Parameters

obj	copy constructor deleted

7.5.3 Member Function Documentation

7.5.3.1 GetInstance()

```
PromptSingleton * PromptSingleton::GetInstance ()  [static]
```

Function to return instance of the singleton

Returns

Returns instance of singleton

Returns

instance of singleton

Definition at line 49 of file PromptSingleton.cpp.

7.5.3.2 GetPrompt()

```
void PromptSingleton::GetPrompt ()
```

Function which gets new prompt to set directly from cin stream

Sets prompt from cin stream

Definition at line 64 of file PromptSingleton.cpp.

7.5.3.3 GetPromptAuto()

```
void PromptSingleton::GetPromptAuto (
            std::vector< std::string > dict)
```

Function which gets new prompt to set directly from cin stream but also consist autocomplete

Parameters

dict	dictionary used to check for autocompletion

Definition at line 72 of file PromptSingleton.cpp.

7.5.3.4 RetValues()

```
std::string PromptSingleton::RetValues ()  [inline]
```

Function which returns prompt of the singleton (not an instance)

Returns

prompt string

Definition at line 37 of file PromptSingleton.hpp.

7.5.3.5 SetValues()

```
void PromptSingleton::SetValues (
            std::string & val)
```

Function to change value of the prompt

Parameters

val	value to be set

Function to change current value of prompt

Parameters

val	value to be set

Definition at line 41 of file PromptSingleton.cpp.

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

- Logic/PromptSingleton.hpp
- Logic/PromptSingleton.cpp
- main.cpp

7.6 QueryHelper Class Reference

`#include <QueryHelper.hpp>`

Static Public Member Functions

- static std::string createUserTable ()
- static std::string createAdminTable ()
- static std::string createPhraseTable ()
- static std::string createTagTable ()
- static std::string createPhraseTagTable ()
- static std::string insertUser (std::string nick, std::string pass)
- static std::string getUsers ()
- static std::string deleteUser (int id)
- static std::string updateUserPass (int id, std::string pass)
- static std::string insertAdmin (int userId)
- static std::string getAdmins ()
- static std::string deleteAdmin (int adminId)
- static std::string loginUser (std::string &log, std::string &pass)
- static std::string insertPhrase (int &id, std::string &body, std::string &response)
- static std::string getPhrases ()
- static std::string getPhrase (int phraseId)
- static std::string deletePhrase (int id)
- static std::string insertTag (std::string body)
- static std::string getTags ()
- static std::string deleteTag (int id)
- static std::string insertFavourite (int phraseId)
- static std::string getFavourites (int userId)
- static std::string deleteFavourite (int phraseId)
- static std::string connectTagToPhrase (int phraseId, int tagId)
- static std::string getPhrasesWithTag ()

7.6.1 Detailed Description

Helper class which contains queries All queries are self-explanatory

Definition at line 15 of file QueryHelper.hpp.

7.6.2 Member Function Documentation

7.6.2.1 connectTagToPhrase()

```
std::string QueryHelper::connectTagToPhrase (
            int phraseId,
            int tagId)  [static]
```

Definition at line 125 of file QueryHelper.cpp.

7.6.2.2 createAdminTable()

```
std::string QueryHelper::createAdminTable ()  [static]
```

Definition at line 15 of file QueryHelper.cpp.

7.6.2.3 createPhraseTable()

```
std::string QueryHelper::createPhraseTable ()  [static]
```

Definition at line 23 of file QueryHelper.cpp.

7.6.2.4 createPhraseTagTable()

```
std::string QueryHelper::createPhraseTagTable ()  [static]
```

Definition at line 42 of file QueryHelper.cpp.

7.6.2.5 createTagTable()

```
std::string QueryHelper::createTagTable ()  [static]
```

Definition at line 34 of file QueryHelper.cpp.

7.6.2.6 createUserTable()

```
std::string QueryHelper::createUserTable ()  [static]
```

Definition at line 7 of file QueryHelper.cpp.

7.6.2.7 deleteAdmin()

```
std::string QueryHelper::deleteAdmin (
            int adminId)  [static]
```

Definition at line 77 of file QueryHelper.cpp.

7.6.2.8 deleteFavourite()

```
std::string QueryHelper::deleteFavourite (
            int phraseId)  [static]
```

Definition at line 121 of file QueryHelper.cpp.

7.6.2.9 deletePhrase()

```
std::string QueryHelper::deletePhrase (
            int id)  [static]
```

Definition at line 97 of file QueryHelper.cpp.

7.6.2.10 deleteTag()

```
std::string QueryHelper::deleteTag (
            int id)  [static]
```

Definition at line 109 of file QueryHelper.cpp.

7.6.2.11 deleteUser()

```
std::string QueryHelper::deleteUser (
            int id)  [static]
```

Definition at line 60 of file QueryHelper.cpp.

7.6.2.12 getAdmins()

```
std::string QueryHelper::getAdmins ()  [static]
```

Definition at line 73 of file QueryHelper.cpp.

7.6.2.13 getFavourites()

```
std::string QueryHelper::getFavourites (
            int userId)  [static]
```

Definition at line 117 of file QueryHelper.cpp.

7.6.2.14 getPhrase()

```
std::string QueryHelper::getPhrase (
            int phraseId)  [static]
```

Definition at line 93 of file QueryHelper.cpp.

7.6.2.15 getPhrases()

```
std::string QueryHelper::getPhrases ()  [static]
```

Definition at line 89 of file QueryHelper.cpp.

7.6.2.16 getPhrasesWithTag()

```
std::string QueryHelper::getPhrasesWithTag ()  [static]
```

Definition at line 129 of file QueryHelper.cpp.

7.6.2.17 getTags()

```
std::string QueryHelper::getTags ()  [static]
```

Definition at line 105 of file QueryHelper.cpp.

7.6.2.18 getUsers()

```
std::string QueryHelper::getUsers ()  [static]
```

Definition at line 56 of file QueryHelper.cpp.

7.6.2.19 insertAdmin()

```
std::string QueryHelper::insertAdmin (
            int userId)  [static]
```

Definition at line 69 of file QueryHelper.cpp.

7.6.2.20 insertFavourite()

```
std::string QueryHelper::insertFavourite (
            int phraseId)  [static]
```

Definition at line 113 of file QueryHelper.cpp.

7.6.2.21 insertPhrase()

```
std::string QueryHelper::insertPhrase (
            int & id,
            std::string & body,
            std::string & response)  [static]
```

Definition at line 85 of file QueryHelper.cpp.

7.6.2.22 insertTag()

```
std::string QueryHelper::insertTag (
            std::string body)  [static]
```

Definition at line 101 of file QueryHelper.cpp.

7.6.2.23 insertUser()

```
std::string QueryHelper::insertUser (
            std::string nick,
            std::string pass)  [static]
```

Definition at line 52 of file QueryHelper.cpp.

7.6.2.24 loginUser()

```
std::string QueryHelper::loginUser (
            std::string & log,
            std::string & pass)  [static]
```

Definition at line 81 of file QueryHelper.cpp.

7.6.2.25 updateUserPass()

```
std::string QueryHelper::updateUserPass (
            int id,
            std::string pass)  [static]
```

Definition at line 64 of file QueryHelper.cpp.

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

- Logic/Database/QueryHelper.hpp
- Logic/Database/QueryHelper.cpp

7.7 StackManager Class Reference

```
#include <StackManager.hpp>
```

Public Member Functions

- void AskQuestion (std::string &question)

  returns answers
- void SetQuestion (std::string newInput)

  sets question
- void SetQuestionByTags (std::string newInput)

  sets question with usage of tags
- void GetAnswer (std::string res)

  gets answers
- void ChangeJsonToString (std::string &)

  changes provided question/answer to string from json format
- void SetQuestionId (std::string)

  sets question by provided id
- void FillTabel (std::string input)

  fill table of questions for list tags state
- void RemoveHtmlTags (std::string &input)

  removes html code
- void ReturnNiceCode (std::string &input)

  utilize tabs and space to code format
- void ChangingSpecialChar (std::string &input, std::string inChar, std::string outChar)

  color special chars
- void LookForByTags (std::string &input)
- void checkTagQuestionList (std::string &tagInput)

  checks for list of questions based on tags
- std::string GetTitle ()

  returns title of question
- std::string GetQuestionId ()

  returns id of question
- void GetQuestionFromID (std::string id)

  gets quesiton from provided id

Static Public Member Functions

- static std::vector< TagsList > getQuestionList ()

  returnS vector of questions

Public Attributes

- std::string bestAnswer [3] = {"","",""}

7.7.1 Detailed Description

Class utilizing stackoverflow api

Definition at line 17 of file StackManager.hpp.

7.7.2 Member Function Documentation

7.7.2.1 AskQuestion()

```
void StackManager::AskQuestion (
            std::string & question)
```

returns answers

Definition at line 13 of file StackManager.cpp.

7.7.2.2 ChangeJsonToString()

```
void StackManager::ChangeJsonToString (
            std::string & input)
```

changes provided question/answer to string from json format

Definition at line 36 of file StackManager.cpp.

7.7.2.3 ChangingSpecialChar()

```
void StackManager::ChangingSpecialChar (
            std::string & input,
            std::string inChar,
            std::string outChar)
```

color special chars

Definition at line 96 of file StackManager.cpp.

7.7.2.4 checkTagQuestionList()

```
void StackManager::checkTagQuestionList (
            std::string & tagInput)
```

checks for list of questions based on tags

Definition at line 114 of file StackManager.cpp.

7.7.2.5 FillTabel()

```
void StackManager::FillTabel (
            std::string input)
```

fill table of questions for list tags state

Definition at line 72 of file StackManager.cpp.

7.7.2.6 GetAnswer()

```
void StackManager::GetAnswer (
            std::string res)
```

gets answers

Definition at line 28 of file StackManager.cpp.

7.7.2.7 GetQuestionFromID()

```
void StackManager::GetQuestionFromID (
            std::string id)
```

gets quesiton from provided id

Definition at line 153 of file StackManager.cpp.

7.7.2.8 GetQuestionId()

```
std::string StackManager::GetQuestionId ()
```

returns id of question

Definition at line 148 of file StackManager.cpp.

7.7.2.9 getQuestionList()

```
std::vector< TagsList > StackManager::getQuestionList ()  [static]
```

returnS vector of questions

Definition at line 140 of file StackManager.cpp.

7.7.2.10 GetTitle()

```
std::string StackManager::GetTitle ()
```

returns title of question

Definition at line 144 of file StackManager.cpp.

7.7.2.11 LookForByTags()

```
void StackManager::LookForByTags (
            std::string & input)
```

Definition at line 110 of file StackManager.cpp.

7.7.2.12 RemoveHtmlTags()

```
void StackManager::RemoveHtmlTags (
            std::string & input)
```

removes html code

Definition at line 92 of file StackManager.cpp.

7.7.2.13 ReturnNiceCode()

```
void StackManager::ReturnNiceCode (
            std::string & input)
```

utilize tabs and space to code format

Definition at line 103 of file StackManager.cpp.

7.7.2.14 SetQuestion()

```
void StackManager::SetQuestion (
            std::string newInput)
```

sets question

Definition at line 18 of file StackManager.cpp.

7.7.2.15 SetQuestionByTags()

```
void StackManager::SetQuestionByTags (
            std::string newInput)
```

sets question with usage of tags

Definition at line 22 of file StackManager.cpp.

7.7.2.16 SetQuestionId()

```
void StackManager::SetQuestionId (
            std::string input)
```

sets question by provided id

Definition at line 53 of file StackManager.cpp.

7.7.3 Member Data Documentation

7.7.3.1 bestAnswer

```
std::string StackManager::bestAnswer[3] = {"","",""}
```

Definition at line 41 of file StackManager.hpp.

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

- Logic/StackApi/StackManager.hpp
- Logic/StackApi/StackManager.cpp

7.8 State< T > Class Template Reference

```
#include <State.hpp>
```

Public Member Functions

- T getID ()
  The ID of the state.
- const std::string & GetName () const
  The name of the state.
- State (FiniteStateMachine< T > &fsm, T id, std::string name="default")
  Default constructor.
- virtual ~State ()
  Virtual destructor.
- virtual void OnEnter ()
- virtual void OnExit ()
- virtual void OnUpdate ()

Protected Attributes

- std::string mName
  name of the state
- T mID
  id of the state
- FiniteStateMachine< T > & mFsm
  state machine that state is created for

7.8.1 Detailed Description

template<typename T>
class State< T >

Template class of one state of FSM

Template Parameters

T	template class which State is created for

Predefinition of State class

Template Parameters

T	template class for which state is created

Definition at line 23 of file State.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 State()

```
template<typename T >
State< T >::State (
            FiniteStateMachine< T > & fsm,
            T id,
            std::string name = "default")  [inline], [explicit]
```

Default constructor.

Definition at line 37 of file State.hpp.

7.8.2.2 ~State()

```
template<typename T >
virtual State< T >::~State ()  [inline], [virtual]
```

Virtual destructor.

Definition at line 45 of file State.hpp.

7.8.3 Member Function Documentation

7.8.3.1 getID()

```
template<typename T >
T State< T >::getID ()  [inline]
```

The ID of the state.

Definition at line 27 of file State.hpp.

7.8.3.2 GetName()

```
template<typename T >
const std::string & State< T >::GetName () const  [inline]
```

The name of the state.

Definition at line 32 of file State.hpp.

7.8.3.3 OnEnter()

```
template<typename T >
virtual void State< T >::OnEnter ()  [inline], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented in StateAbout, StateExit, StateFavourites, StateHistory, StateIdle, StateListTags, StateLogin, StateMenu, StatePrompt, StateRegister, StateResult, StateResultTags, and StateTags.

Definition at line 49 of file State.hpp.

7.8.3.4 OnExit()

```
template<typename T >
virtual void State< T >::OnExit ()  [inline], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented in StateAbout, StateExit, StateFavourites, StateHistory, StateIdle, StateListTags, StateLogin, StateMenu, StatePrompt, StateRegister, StateResult, StateResultTags, and StateTags.

Definition at line 55 of file State.hpp.

7.8.3.5 OnUpdate()

```
template<typename T >
virtual void State< T >::OnUpdate ()  [inline], [virtual]
```

Virtual function to describe behaviour of state on update time on update - in the middle of the state flow

Reimplemented in StateAbout, StateExit, StateFavourites, StateHistory, StateIdle, StateListTags, StateLogin, StateMenu, StatePrompt, StateRegister, StateResult, StateResultTags, and StateTags.

Definition at line 62 of file State.hpp.

7.8.4 Member Data Documentation

7.8.4.1 mFsm

```
template<typename T >
FiniteStateMachine<T>& State< T >::mFsm  [protected]
```

state machine that state is created for

Definition at line 68 of file State.hpp.

7.8.4.2 mID

```
template<typename T >
T State< T >::mID  [protected]
```

id of the state

Definition at line 67 of file State.hpp.

7.8.4.3 mName

```
template<typename T >
std::string State< T >::mName  [protected]
```

name of the state

Definition at line 66 of file State.hpp.

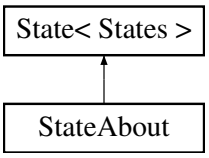
The documentation for this class was generated from the following file:

- FSM/State.hpp

7.9 StateAbout Class Reference

```
#include <StateAbout.hpp>
```

Inheritance diagram for StateAbout:



Public Member Functions

- StateAbout (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName
  name of the state
- States mID
  id of the state
- FiniteStateMachine< States > & mFsm
  state machine that state is created for

7.9.1 Detailed Description

Provides information about application

Definition at line 21 of file StateAbout.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 StateAbout()

```
StateAbout::StateAbout (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 27 of file StateAbout.hpp.

7.9.3 Member Function Documentation

7.9.3.1 OnEnter()

```
void StateAbout::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 11 of file StateAbout.cpp.

7.9.3.2 OnExit()

```
void StateAbout::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 44 of file StateAbout.cpp.

7.9.3.3 OnUpdate()

```
void StateAbout::OnUpdate ()  [override], [virtual]
```

Implements virtual OnUpdate, in which provides app description

Reimplemented from State< States >.

Definition at line 20 of file StateAbout.cpp.

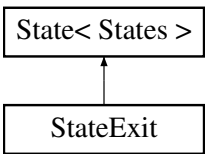
The documentation for this class was generated from the following files:

- States/StateAbout.hpp
- States/StateAbout.cpp

7.10 StateExit Class Reference

```
#include <StateExit.hpp>
```

Inheritance diagram for StateExit:



Public Member Functions

- StateExit (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName
  name of the state
- States mID
  id of the state
- FiniteStateMachine< States > & mFsm
  state machine that state is created for

7.10.1 Detailed Description

Exit State of the application TBD

Definition at line 20 of file StateExit.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 StateExit()

```
StateExit::StateExit (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 23 of file StateExit.hpp.

7.10.3 Member Function Documentation

7.10.3.1 OnEnter()

```
void StateExit::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 15 of file StateExit.cpp.

7.10.3.2 OnExit()

```
void StateExit::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 7 of file StateExit.cpp.

7.10.3.3 OnUpdate()

```
void StateExit::OnUpdate ()  [override], [virtual]
```

Virtual function to describe behaviour of state on update time on update - in the middle of the state flow

Reimplemented from State< States >.

Definition at line 11 of file StateExit.cpp.

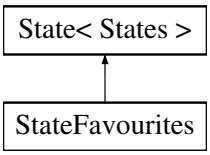
The documentation for this class was generated from the following files:

- States/StateExit.hpp
- States/StateExit.cpp

7.11 StateFavourites Class Reference

```
#include <StateFavourites.hpp>
```

Inheritance diagram for StateFavourites:



Public Member Functions

- StateFavourites (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.11.1 Detailed Description

Provides information about questions which user added to favourites

Definition at line 20 of file StateFavourites.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 StateFavourites()

```
StateFavourites::StateFavourites (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 34 of file StateFavourites.hpp.

7.11.3 Member Function Documentation

7.11.3.1 OnEnter()

```
void StateFavourites::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 111 of file StateFavourites.cpp.

7.11.3.2 OnExit()

```
void StateFavourites::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 12 of file StateFavourites.cpp.

7.11.3.3 OnUpdate()

```
void StateFavourites::OnUpdate ()  [override], [virtual]
```

Provides favourites question of user by overriding virtual OnUpdate from State

Reimplemented from State< States >.

Definition at line 19 of file StateFavourites.cpp.

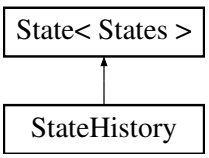
The documentation for this class was generated from the following files:

- States/StateFavourites.hpp
- States/StateFavourites.cpp

7.12 StateHistory Class Reference

```
#include <StateHistory.hpp>
```

Inheritance diagram for StateHistory:



Public Member Functions

- StateHistory (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.12.1 Detailed Description

State which contains history of our searching

Definition at line 20 of file StateHistory.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 StateHistory()

```
StateHistory::StateHistory (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 31 of file StateHistory.hpp.

7.12.3 Member Function Documentation

7.12.3.1 OnEnter()

```
void StateHistory::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 95 of file StateHistory.cpp.

7.12.3.2 OnExit()

```
void StateHistory::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 9 of file StateHistory.cpp.

7.12.3.3 OnUpdate()

```
void StateHistory::OnUpdate ()  [override], [virtual]
```

Provides information about history of searching in Prompt State

Reimplemented from State< States >.

Definition at line 16 of file StateHistory.cpp.

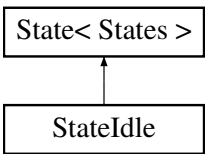
The documentation for this class was generated from the following files:

- States/StateHistory.hpp
- States/StateHistory.cpp

7.13 StateIdle Class Reference

```
#include <StateIdle.hpp>
```

Inheritance diagram for StateIdle:



Public Member Functions

- StateIdle (FiniteStateMachine< States > &fsm)
- void OnEnter () override

  overriding OnEnter virtual function
- void OnUpdate () override

  overriding OnUpdate virtual function
- void OnExit () override

  overriding OnExit virtual function

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.13.1 Detailed Description

State from which program launches - default state Description of this state will be wider, since lots of fields are similar in most of states

Definition at line 20 of file StateIdle.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 StateIdle()

```
StateIdle::StateIdle (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Parameters

fsm	explicit constructor with declared FSM to be used and the name of state

Definition at line 28 of file StateIdle.hpp.

7.13.3 Member Function Documentation

7.13.3.1 OnEnter()

```
void StateIdle::OnEnter ()  [override], [virtual]
```

overriding OnEnter virtual function

Function which executes on enter to state

Reimplemented from State< States >.

Definition at line 13 of file StateIdle.cpp.

7.13.3.2 OnExit()

```
void StateIdle::OnExit ()  [override], [virtual]
```

overriding OnExit virtual function

Function which executes after on update

Reimplemented from State< States >.

Definition at line 53 of file StateIdle.cpp.

7.13.3.3 OnUpdate()

```
void StateIdle::OnUpdate ()  [override], [virtual]
```

overriding OnUpdate virtual function

Function which executes after on enter Provides title and options to choose (login/register)

Reimplemented from State< States >.

Definition at line 24 of file StateIdle.cpp.

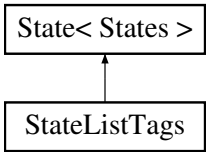
The documentation for this class was generated from the following files:

- States/StateIdle.hpp
- States/StateIdle.cpp

7.14 StateListTags Class Reference

```
#include <StateListTags.hpp>
```

Inheritance diagram for StateListTags:



Public Member Functions

- StateListTags (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override
- void ManageList ()
- bool ChoosingTitle (std::string in)

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.14.1 Detailed Description

State which provides list of questions found by tags provided in Tags state

Definition at line 21 of file StateListTags.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 StateListTags()

```
StateListTags::StateListTags (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 31 of file StateListTags.hpp.

7.14.3 Member Function Documentation

7.14.3.1 ChoosingTitle()

```
bool StateListTags::ChoosingTitle (
            std::string in)
```

Definition at line 85 of file StateListTags.cpp.

7.14.3.2 ManageList()

```
void StateListTags::ManageList ()
```

Definition at line 65 of file StateListTags.cpp.

7.14.3.3 OnEnter()

```
void StateListTags::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 10 of file StateListTags.cpp.

7.14.3.4 OnExit()

```
void StateListTags::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 61 of file StateListTags.cpp.

7.14.3.5 OnUpdate()

```
void StateListTags::OnUpdate ()  [override], [virtual]
```

Prints the list of the questions, fills the canva

Reimplemented from State< States >.

Definition at line 19 of file StateListTags.cpp.

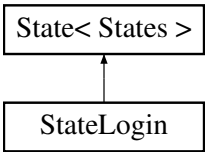
The documentation for this class was generated from the following files:

- States/StateListTags.hpp
- States/StateListTags.cpp

7.15 StateLogin Class Reference

```
#include <StateLogin.hpp>
```

Inheritance diagram for StateLogin:



Public Member Functions

- StateLogin (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.15.1 Detailed Description

State which controls login authorization

Definition at line 18 of file StateLogin.hpp.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 StateLogin()

```
StateLogin::StateLogin (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 23 of file StateLogin.hpp.

7.15.3 Member Function Documentation

7.15.3.1 OnEnter()

```
void StateLogin::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 12 of file StateLogin.cpp.

7.15.3.2 OnExit()

```
void StateLogin::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 51 of file StateLogin.cpp.

7.15.3.3 OnUpdate()

```
void StateLogin::OnUpdate ()  [override], [virtual]
```

Controls credentials by refering to dbmanager instance

Reimplemented from State< States >.

Definition at line 21 of file StateLogin.cpp.

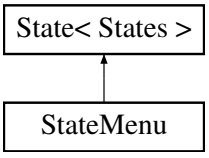
The documentation for this class was generated from the following files:

- States/StateLogin.hpp
- States/StateLogin.cpp

7.16 StateMenu Class Reference

```
#include <StateMenu.hpp>
```

Inheritance diagram for StateMenu:



Public Member Functions

- StateMenu (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.16.1 Detailed Description

State which provides main Menu of the app Contains all of the most important options - question, tags, favourites

Definition at line 21 of file StateMenu.hpp.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 StateMenu()

```
StateMenu::StateMenu (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 31 of file StateMenu.hpp.

7.16.3 Member Function Documentation

7.16.3.1 OnEnter()

```
void StateMenu::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 10 of file StateMenu.cpp.

7.16.3.2 OnExit()

```
void StateMenu::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 53 of file StateMenu.cpp.

7.16.3.3 OnUpdate()

```
void StateMenu::OnUpdate ()  [override], [virtual]
```

Implements transitions between main states

Reimplemented from State< States >.

Definition at line 19 of file StateMenu.cpp.

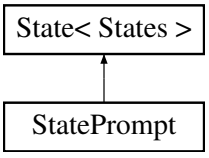
The documentation for this class was generated from the following files:

- States/StateMenu.hpp
- States/StateMenu.cpp

7.17 StatePrompt Class Reference

```
#include <StatePrompt.hpp>
```

Inheritance diagram for StatePrompt:



Public Member Functions

- StatePrompt (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.17.1 Detailed Description

Takes keywords from user, which will be used later in the stack scraping

Definition at line 20 of file StatePrompt.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 StatePrompt()

```
StatePrompt::StatePrompt (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 27 of file StatePrompt.hpp.

7.17.3 Member Function Documentation

7.17.3.1 OnEnter()

```
void StatePrompt::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 13 of file StatePrompt.cpp.

7.17.3.2 OnExit()

```
void StatePrompt::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 51 of file StatePrompt.cpp.

7.17.3.3 OnUpdate()

```
void StatePrompt::OnUpdate ()  [override], [virtual]
```

Takes desired prompt from user

Reimplemented from State< States >.

Definition at line 22 of file StatePrompt.cpp.

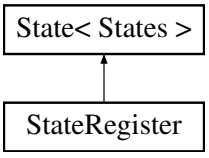
The documentation for this class was generated from the following files:

- States/StatePrompt.hpp
- States/StatePrompt.cpp

7.18 StateRegister Class Reference

```
#include <StateRegister.hpp>
```

Inheritance diagram for StateRegister:



Public Member Functions

- StateRegister (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.18.1 Detailed Description

Controls registration process

Definition at line 19 of file StateRegister.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 StateRegister()

```
StateRegister::StateRegister (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 25 of file StateRegister.hpp.

7.18.3 Member Function Documentation

7.18.3.1 OnEnter()

```
void StateRegister::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 11 of file StateRegister.cpp.

7.18.3.2 OnExit()

```
void StateRegister::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 57 of file StateRegister.cpp.

7.18.3.3 OnUpdate()

```
void StateRegister::OnUpdate ()  [override], [virtual]
```

Check if credentials are valid and pass them to database

Reimplemented from State< States >.

Definition at line 20 of file StateRegister.cpp.

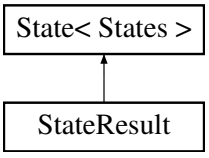
The documentation for this class was generated from the following files:

- States/StateRegister.hpp
- States/StateRegister.cpp

7.19 StateResult Class Reference

```
#include <StateResult.hpp>
```

Inheritance diagram for StateResult:



Public Member Functions

- StateResult (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override
- void QuestionManage ()

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.19.1 Detailed Description

Most important class of the program Searches for questions and answers on stackoverflow

Definition at line 23 of file StateResult.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 StateResult()

```
StateResult::StateResult (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 34 of file StateResult.hpp.

7.19.3 Member Function Documentation

7.19.3.1 OnEnter()

```
void StateResult::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 10 of file StateResult.cpp.

7.19.3.2 OnExit()

```
void StateResult::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 38 of file StateResult.cpp.

7.19.3.3 OnUpdate()

```
void StateResult::OnUpdate ()  [override], [virtual]
```

Virtual function to describe behaviour of state on update time on update - in the middle of the state flow prepare canva

find questions, answers and print them

Reimplemented from State< States >.

Definition at line 17 of file StateResult.cpp.

7.19.3.4 QuestionManage()

```
void StateResult::QuestionManage ()
```

Provides stack scraping and formatting parse json to string

remove html tags

return string with spaces, tabs and with some attributes changed

color syntax

Do the same as above, but for answers

Definition at line 46 of file StateResult.cpp.

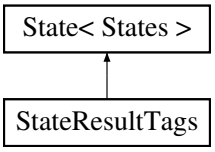
The documentation for this class was generated from the following files:

- States/StateResult.hpp
- States/StateResult.cpp

7.20 StateResultTags Class Reference

`#include <StateResultTags.hpp>`

Inheritance diagram for StateResultTags:



Public Member Functions

- StateResultTags (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override
- void QuestionManage ()

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.20.1 Detailed Description

Prints result question and answers chosen in StateListTags by user

Definition at line 17 of file StateResultTags.hpp.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 StateResultTags()

```
StateResultTags::StateResultTags (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 28 of file StateResultTags.hpp.

7.20.3 Member Function Documentation

7.20.3.1 OnEnter()

```
void StateResultTags::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 10 of file StateResultTags.cpp.

7.20.3.2 OnExit()

```
void StateResultTags::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 41 of file StateResultTags.cpp.

7.20.3.3 OnUpdate()

```
void StateResultTags::OnUpdate ()  [override], [virtual]
```

Prints question with answer(s) based on specific question id

Reimplemented from State< States >.

Definition at line 20 of file StateResultTags.cpp.

7.20.3.4 QuestionManage()

```
void StateResultTags::QuestionManage ()
```

Definition at line 45 of file StateResultTags.cpp.

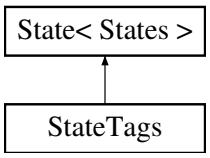
The documentation for this class was generated from the following files:

- States/StateResultTags.hpp
- States/StateResultTags.cpp

7.21 StateTags Class Reference

`#include <StateTags.hpp>`

Inheritance diagram for StateTags:



Public Member Functions

- StateTags (FiniteStateMachine< States > &fsm)
- void OnEnter () override
- void OnUpdate () override
- void OnExit () override

Public Member Functions inherited from State< States >

- States getID ()

  The ID of the state.
- const std::string & GetName () const

  The name of the state.
- State (FiniteStateMachine< States > &fsm, States id, std::string name="default")

  Default constructor.
- virtual ~State ()

  Virtual destructor.

Additional Inherited Members

Protected Attributes inherited from State< States >

- std::string mName

  name of the state
- States mID

  id of the state
- FiniteStateMachine< States > & mFsm

  state machine that state is created for

7.21.1 Detailed Description

Provides searching by tags

Definition at line 21 of file StateTags.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 StateTags()

```
StateTags::StateTags (
            FiniteStateMachine< States > & fsm)  [inline], [explicit]
```

Definition at line 25 of file StateTags.hpp.

7.21.3 Member Function Documentation

7.21.3.1 OnEnter()

```
void StateTags::OnEnter ()  [override], [virtual]
```

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

Definition at line 9 of file StateTags.cpp.

7.21.3.2 OnExit()

```
void StateTags::OnExit ()  [override], [virtual]
```

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

Definition at line 42 of file StateTags.cpp.

7.21.3.3 OnUpdate()

```
void StateTags::OnUpdate ()  [override], [virtual]
```

Takes tags from user, which will be later used to search

Reimplemented from State< States >.

Definition at line 22 of file StateTags.cpp.

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

- States/StateTags.hpp
- States/StateTags.cpp

7.22 SyntaxHighlighting Class Reference

```
#include <SyntaxHighlighting.hpp>
```

Public Member Functions

- SyntaxHighlighting ()
  default constructor
- void RecognizeSyntax (std::string &in)
  finding `and` marks in questions and answers
- std::string Hightlighting (std::string &in)
  highlights code
- void RemoveTags (std::string &input, std::string tag, std::string out, int pos)
  deleting html tags
- void ColorChar (std::string &input, std::string tag, std::string out)
  coloring special chars
- void ColorBracket (std::string &in)
  provides brackets coloring

7.22.1 Detailed Description

Takes control of highlighting of syntax

Definition at line 16 of file SyntaxHighlighting.hpp.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 SyntaxHighlighting()

```
SyntaxHighlighting::SyntaxHighlighting ()
```

default constructor

Definition at line 12 of file SyntaxHighlighting.cpp.

7.22.3 Member Function Documentation

7.22.3.1 ColorBracket()

```
void SyntaxHighlighting::ColorBracket (
            std::string & in)
```

provides brackets coloring

Definition at line 89 of file SyntaxHighlighting.cpp.

7.22.3.2 ColorChar()

```
void SyntaxHighlighting::ColorChar (
            std::string & input,
            std::string tag,
            std::string out)
```

coloring special chars

Definition at line 77 of file SyntaxHighlighting.cpp.

7.22.3.3 Hightlighting()

```
std::string SyntaxHighlighting::Hightlighting (
            std::string & in)
```

highlights code

Definition at line 54 of file SyntaxHighlighting.cpp.

7.22.3.4 RecognizeSyntax()

```
void SyntaxHighlighting::RecognizeSyntax (
            std::string & in)
```

finding `and` marks in questions and answers

Definition at line 23 of file SyntaxHighlighting.cpp.

7.22.3.5 RemoveTags()

```
void SyntaxHighlighting::RemoveTags (
            std::string & input,
            std::string tag,
            std::string out,
            int pos)
```

deleting html tags

Definition at line 70 of file SyntaxHighlighting.cpp.

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

- Logic/StackApi/SyntaxHighlighting.hpp
- Logic/StackApi/SyntaxHighlighting.cpp

7.23 TagsList Class Reference

`#include <TagsList.hpp>`

Public Member Functions

- int GetID ()
- std::string GetTitle ()
- TagsList (int _id, std::string &_title)

7.23.1 Detailed Description

Class which contains functions to list of questions creation

Definition at line 14 of file TagsList.hpp.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 TagsList()

```
TagsList::TagsList (
            int _id,
            std::string & _title)
```

constructor

Parameters

_id	id of the question
_title	title of the question

Definition at line 28 of file TagsList.cpp.

7.23.3 Member Function Documentation

7.23.3.1 GetID()

```
int TagsList::GetID ()
```

Returns

id of the tag

Definition at line 13 of file TagsList.cpp.

7.23.3.2 GetTitle()

```
std::string TagsList::GetTitle ()
```

Returns

title of the question

Definition at line 20 of file TagsList.cpp.

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

- Logic/TagList/TagsList.hpp
- Logic/TagList/TagsList.cpp

Chapter 8

File Documentation

8.1 conanfile.py File Reference

Classes

- class conanfile.ConanApplication

Namespaces

- namespace conanfile

8.2 conanfile.py

Go to the documentation of this file.

```
00001 # This file is managed by Conan, contents will be overwritten.
00002 # To keep your changes, remove these comment lines, but the plugin won't be able to modify your
      requirements
00003
00004 from conan import ConanFile
00005 from conan.tools.cmake import cmake_layout, CMakeToolchain
00006
00007 #  Default generated file by conan package manager
00008 #  Provides all information about actions on packages
00009
00010 class ConanApplication(ConanFile):
00011     package_type = "application"
00012     settings = "os", "compiler", "build_type", "arch"
00013     generators = "CMakeDeps"
00014
00015
00016     def layout(self):
00017         cmake_layout(self)
00018
00019     def generate(self):
00020         tc = CMakeToolchain(self)
00021         tc.user_presets_path = False
00022         tc.generate()
00023
00024     def requirements(self):
00025         requirements = self.conan_data.get('requirements', [])
00026         for requirement in requirements:
00027             self.requires(requirement)
```

8.3 Engine.cpp File Reference

```
#include "Engine.hpp"
```

8.4 Engine.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "Engine.hpp"
00006
00010 void Engine::Run() {
00011     fsm->OnUpdate();
00012 }
00013
00018 Engine::Engine() {
00019     fsm = std::make_unique<FiniteStateMachine<States»();
00020     State<States>& idleState = fsm->Add<StateIdle>(States::IDLE);
00021     State<States>& loginState = fsm->Add<StateLogin>(States::LOGIN);
00022     State<States>& exitState = fsm->Add<StateExit>(States::EXIT);
00023     State<States>& favouritesState = fsm->Add<StateFavourites>(States::FAVOURITES);
00024     State<States>& historyState = fsm->Add<StateHistory>(States::HISTORY);
00025     State<States>& menuState = fsm->Add<StateMenu>(States::MENU);
00026     State<States>& promptState = fsm->Add<StatePrompt>(States::PROMPT);
00027     State<States>& registerState = fsm->Add<StateRegister>(States::REGISTER);
00028     State<States>& resultState = fsm->Add<StateResult>(States::RESULT);
00029     State<States>& tagsState = fsm->Add<StateTags>(States::TAGS);
00030     State<States>& aboutState = fsm->Add<StateAbout>(States::ABOUT);
00031     State<States>& listTagsState = fsm->Add<StateListTags>(States::LISTTAGS);
00032     State<States>& resultTagsState = fsm->Add<StateResultTags>(States::RESULTTAGS);
00033
00034     fsm->SetCurrentState(States::IDLE);
00035
00036 }
```

8.5 Engine.hpp File Reference

```
#include "States/StatesWrapper.hpp"
#include "FSM/StateMachine.hpp"
#include "Logic/PromptSingleton.hpp"
```

Classes

- class Engine

8.6 Engine.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
00007
00008 #include "States/StatesWrapper.hpp"
00009 #include "FSM/StateMachine.hpp"
```

```
00010 #include "Logic/PromptSingleton.hpp"
00011
00012
00017 class Engine {
00018     std::unique_ptr<FiniteStateMachine<States» fsm = nullptr;
00019 public:
00020     Engine();
00021     void Run();
00022 };
00023
00024
00025 #endif //INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
```

8.7 FSM/State.hpp File Reference

```
#include "StateMachine.hpp"
#include <string>
#include <utility>
```

Classes

- class State< T >

8.8 State.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
00007
00011 #include "StateMachine.hpp"
00012
00013 #include <string>
00014 #include <utility>
00015 template <typename T>
00016 class FiniteStateMachine;
00017
00022 template <typename T>
00023 class State
00024 {
00025 public:
00027     inline T getID()
00028     {
00029         return mID;
00030     }
00032     inline const std::string& GetName() const
00033     {
00034         return mName;
00035     }
00037     explicit State(FiniteStateMachine<T>& fsm, T id,
00038                    std::string name = "default")
00039             : mName(name)
00040             , mID(id)
00041             , mFsm(fsm)
00042     {
00043     }
00045     virtual ~State() {}
00049     virtual void OnEnter()
00050     {
00051     }
00055     virtual void OnExit()
00056     {
00057     }
00062     virtual void OnUpdate()
00063     {
00064     }
```

```
00065 protected:
00066     std::string mName;
00067     T mID;
00068     FiniteStateMachine<T>& mFsm;
00069 };
00070
00071 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
```

8.9 FSM/StateMachine.hpp File Reference

```
#include "State.hpp"
#include <memory>
#include <map>
#include <string>
#include <cassert>
#include <utility>
```

Classes

- class FiniteStateMachine< T >

8.10 StateMachine.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
00007
00008 #include "State.hpp"
00009
00010 #include <memory>
00011 #include <map>
00012 #include <string>
00013 #include <cassert>
00014 #include <utility>
00015
00020 template <typename T>
00021 class State;
00022
00027 template <typename T>
00028 class FiniteStateMachine
00029 {
00030 protected:
00031     std::map<T, std::unique_ptr<State<T>>> mStates;
00032     State<T>* mCurrentState;
00033 public:
00035     FiniteStateMachine()
00036             : mCurrentState(nullptr)
00037     {}
00044     template <class S>
00045     State<T>& Add(T id)
00046     {
00047         static_assert(not std::is_same<State<T>, S>());
00048         mStates[id] = std::make_unique<S>(*this);
00049         return *mStates[id];
00050     }
00056     State<T>& GetState(T stateID)
00057     {
00058         return *mStates[stateID];
00059     }
00064     State<T>& GetCurrentState()
00065     {
00066         return *mCurrentState;
00067     }
```

```
00068
00073     const State<T>& GetCurrentState() const
00074     {
00075         return *mCurrentState;
00076     }
00077
00082     void SetCurrentState(T stateID)
00083     {
00084         State<T>* state = &GetState(stateID);
00085         SetCurrentState(state);
00086     }
00090     void OnUpdate()
00091     {
00092         if (mCurrentState != nullptr)
00093         {
00094             mCurrentState->OnUpdate();
00095         }
00096     }
00097 protected:
00103     void SetCurrentState(State<T>* state)
00104     {
00105         if (mCurrentState == state)
00106         {
00107             return;
00108         }
00109         if (mCurrentState != nullptr)
00110         {
00111             mCurrentState->OnExit();
00112         }
00113         mCurrentState = state;
00114         if (mCurrentState != nullptr)
00115         {
00116             mCurrentState->OnEnter();
00117         }
00118     }
00119 };
00120
00121
00122 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
```

8.11 Globals.hpp File Reference

```
#include <windows.h>
```

Namespaces

- namespace cmd

  CMD - Namespace responsible for holding globals connected to shell application.

8.12 Globals.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP
00007
00008 #include <windows.h>
00009
00015 namespace cmd
00016 {
00017     static HANDLE hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
00018     static HANDLE hInput = GetStdHandle(STD_INPUT_HANDLE);
00019 }
00020
00021 #endif //INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP
```

8.13 Logic/Database/DBmanager.cpp File Reference

```
#include "DBmanager.hpp"
#include <string>
#include <fstream>
#include <iostream>
```

Typedefs

- typedef int(* sqlite3_callback) (void *, int, char **, char **)

Variables

- std::vector< std::pair< std::string, std::string > > receivedData

8.13.1 Typedef Documentation

8.13.1.1 sqlite3_callback

```
typedef int(* sqlite3_callback) (void *, int, char **, char **)
```

Definition at line 28 of file DBmanager.cpp.

8.13.2 Variable Documentation

8.13.2.1 receivedData

```
std::vector<std::pair<std::string,std::string> > receivedData
```

Definition at line 13 of file DBmanager.cpp.

8.14 DBmanager.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by lucja on 11.05.2024.
00003 //
00004
00005 #include "DBmanager.hpp"
00006 #include <string>
00007 #include <fstream>
00008 #include <iostream>
00009
00010 std::string DBmanager::nickName = "admin";
00011 int DBmanager::id = 1;
00012
00013 std::vector<std::pair<std::string,std::string» receivedData;
00014
00015
00016 static int callback(void *NotUsed, int argc, char **argv, char **azColName) {
00017
00018     for(int i = 0; i<argc; i++) {
00019         std::string key = azColName[i];
```

```
00020         std::string value = argv[i] ? argv[i] : "NULL";
00021         std::pair<std::string,std::string> pair = {key,value};
00022         receivedData.push_back(pair);
00023     }
00024
00025     return 0;
00026 }
00027
00028 typedef int (*sqlite3_callback)(
00029    void*,    /* Data provided in the 4th argument of sqlite3_exec() */
00030    int,      /* The number of columns in row */
00031    char**,   /* An array of strings representing fields in the row */
00032    char**    /* An array of strings representing column names */
00033 );
00034
00035
00036
00037 int DBmanager::openDatabase()
00038 {
00039     rc = sqlite3_open("ss.db", &db);
00040     return rc? 0:1;
00041 }
00042
00043 int DBmanager::createDatabase()
00044 {
00045     this->createUserTable();
00046     this->createAdminTable();
00047     this->createPhraseTable();
00048     this->createTagTable();
00049     this->createPhraseTagTable();
00050
00051     std::string admin = "admin";
00052     this->insertUser(admin,admin);
00053     this->insertAdmin(1);
00054     return 1;
00055 }
00056
00057 int DBmanager::closeDatabase()
00058 {
00059     sqlite3_close(db);
00060     return 1;
00061 }
00062
00063 int DBmanager::createUserTable()
00064 {
00065     /* Create SQL statement */
00066     const std::string sql = QueryHelper::createUserTable();
00067
00068     /* Execute SQL statement */
00069     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00070
00071     if( rc != SQLITE_OK ){
00072         return 0;
00073     } else {
00074         return 1;
00075     }
00076 }
00077
00078 int DBmanager::createAdminTable() {
00079
00080     /* Create SQL statement */
00081     const std::string sql = QueryHelper::createAdminTable();
00082
00083     /* Execute SQL statement */
00084     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00085
00086     if( rc != SQLITE_OK ){
00087         return 0;
00088     } else {
00089         return 1;
00090     }
00091
00092 }
00093
00094 int DBmanager::createPhraseTable() {
00095     /* Create SQL statement */
00096     const std::string sql = QueryHelper::createPhraseTable();
00097
00098     /* Execute SQL statement */
00099     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00100
00101     if( rc != SQLITE_OK ){
00102         return 0;
00103     } else {
00104         return 1;
00105     }
00106 }
```

```
00107
00108 int DBmanager::createTagTable() {
00109     /* Create SQL statement */
00110     const std::string sql = QueryHelper::createTagTable();
00111
00112     /* Execute SQL statement */
00113     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00114
00115     if( rc != SQLITE_OK ){
00116         return 0;
00117     } else {
00118         return 1;
00119     }
00120
00121 }
00122
00123 int DBmanager::createPhraseTagTable()
00124 {
00125     /* Create SQL statement */
00126      const std::string sql = QueryHelper::createPhraseTagTable();
00127
00128     /* Execute SQL statement */
00129     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00130
00131     if( rc != SQLITE_OK ){
00132         return 0;
00133     } else {
00134         return 1;
00135     }
00136 }
00137
00138  bool DBmanager::insertUser(std::string& nickname, std::string& password)
00139 {
00140     const char* sql = QueryHelper::insertUser(nickname,password).c_str();
00141
00142     rc = sqlite3_exec(db, sql, callback, nullptr, &zErrMsg);
00143
00144     if( rc != SQLITE_OK ){
00145         return false;
00146     } else {
00147         return true;
00148     }
00149 }
00150
00151  std::vector<std::pair<std::string,std::string» DBmanager::getUsers()
00152 {
00153     receivedData = {};
00154     const char* sql = QueryHelper::getUsers().c_str();
00155
00156     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00157
00158     if( rc != SQLITE_OK ) {
00159         return {{"SQL ERROR",zErrMsg}};
00160     } else {
00161         return receivedData;
00162     }
00163 }
00164
00165 bool DBmanager::updateUserPassword(int id, std::string &password)
00166 {
00167     const char* sql = QueryHelper::updateUserPass(id,password).c_str();
00168
00169 /* Execute SQL statement */
00170     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00171
00172     if( rc != SQLITE_OK ) {
00173     return false;
00174     } else {
00175      return true;
00176 }
00177 }
00178
00179
00180 bool DBmanager::deleteUser(int id)
00181 {
00182     const char* sql = QueryHelper::deleteUser(id).c_str();
00183
00184     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00185
00186     if( rc != SQLITE_OK ) {
00187         return false;
00188     } else {
00189         return true;
00190     }
00191 }
00192
00193 bool DBmanager::insertAdmin(int Id) {
```

```
00194     const char* sql = QueryHelper::insertAdmin(Id).c_str();
00195
00196     rc = sqlite3_exec(db, sql, callback, nullptr, &zErrMsg);
00197
00198     if( rc != SQLITE_OK ){
00199         return false;
00200     } else {
00201         return true;
00202     }
00203
00204
00205 }
00206
00207 std::vector<std::pair<std::string, std::string>> DBmanager::getAdmins() {
00208     receivedData = {};
00209     const char* sql = QueryHelper::getAdmins().c_str();
00210
00211     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00212
00213     if( rc != SQLITE_OK ) {
00214         return {{"SQL ERROR",zErrMsg}};
00215     } else {
00216         return receivedData;
00217     }
00218 }
00219
00220 bool DBmanager::deleteAdmin(int adminId) {
00221     const char* sql = QueryHelper::deleteAdmin(adminId).c_str();
00222
00223     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00224
00225     if( rc != SQLITE_OK ) {
00226         return false;
00227     } else {
00228         return true;
00229     }
00230 }
00231
00232 bool DBmanager::insertPhrase(std::string &body, std::string &response) {
00233
00234     const std::string sql = QueryHelper::insertPhrase(id,body,response);
00235
00236     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00237
00238     if( rc != SQLITE_OK ){
00239         return false;
00240     } else {
00241         return true;
00242     }
00243
00244
00245 }
00246
00247 std::vector<std::pair<std::string, std::string>> DBmanager::getPhrases() {
00248     receivedData = {};
00249     const char* sql = QueryHelper::getPhrases().c_str();
00250
00251     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00252
00253     if( rc != SQLITE_OK ) {
00254         return {{"SQL ERROR",zErrMsg}};
00255     } else {
00256         return receivedData;
00257     }
00258 }
00259
00260 std::vector<std::pair<std::string, std::string>> DBmanager::getPhrase(int phraseId) {
00261     receivedData = {};
00262     const char* sql = QueryHelper::getPhrase(phraseId).c_str();
00263
00264     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00265
00266     if( rc != SQLITE_OK ) {
00267         return {{"SQL ERROR",zErrMsg}};
00268     } else {
00269         return receivedData;
00270     }
00271 }
00272
00273 bool DBmanager::deletePhrase(int id) {
00274     const char* sql = QueryHelper::deletePhrase(id).c_str();
00275
00276     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00277
00278     if( rc != SQLITE_OK ) {
00279         return false;
00280     } else {
```

```
00281         return true;
00282     }
00283 }
00284
00285 bool DBmanager::insertTag(std::string& body) {
00286     const std::string sql = QueryHelper::insertTag(body);
00287
00288     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00289
00290     if( rc != SQLITE_OK ){
00291         return false;
00292     } else {
00293         return true;
00294     }
00295 }
00296
00297 std::vector<std::pair<std::string, std::string» DBmanager::getTags() {
00298     receivedData = {};
00299     const char* sql = QueryHelper::getTags().c_str();
00300
00301     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00302
00303     if( rc != SQLITE_OK ) {
00304         return {{"SQL ERROR",zErrMsg}};
00305     } else {
00306         return receivedData;
00307     }
00308 }
00309
00310 bool DBmanager::deleteTag(int id) {
00311     const char* sql = QueryHelper::deleteTag(id).c_str();
00312
00313     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00314
00315     if( rc != SQLITE_OK ) {
00316         return false;
00317     } else {
00318         return true;
00319     }
00320 }
00321
00322 bool DBmanager::insertFavourite(int phraseId) {
00323     const std::string sql = QueryHelper::insertFavourite(phraseId);
00324
00325     rc = sqlite3_exec(db, sql.c_str(), callback, nullptr, &zErrMsg);
00326
00327     if( rc != SQLITE_OK ){
00328         return false;
00329     } else {
00330         return true;
00331     }
00332
00333 }
00334
00335 std::vector<std::pair<std::string, std::string» DBmanager::getFavourites() {
00336     receivedData = {};
00337     const char* sql = QueryHelper::getFavourites(id).c_str();
00338
00339     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00340
00341     if( rc != SQLITE_OK ) {
00342         return {{"SQL ERROR",zErrMsg}};
00343     } else {
00344         return receivedData;
00345     }
00346 }
00347
00348 bool DBmanager::deleteFavourite(int phraseId) {
00349     if(this->getPhrase(phraseId).empty()) {
00350         return false;
00351     }
00352
00353     const char* sql = QueryHelper::deleteFavourite(phraseId).c_str();
00354
00355     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00356
00357     if( rc != SQLITE_OK ) {
00358         return false;
00359     } else {
00360         return true;
00361     }
00362 }
00363
00364 bool DBmanager::connectTagToPhrase(int phraseId, int tagId) {
00365     const char* sql = QueryHelper::connectTagToPhrase(phraseId,tagId).c_str();
00366
00367     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
```

```
00368
00369     if( rc != SQLITE_OK ) {
00370         return false;
00371     } else {
00372         return true;
00373     }
00374 }
00375
00376 std::vector<std::pair<std::string, std::string» DBmanager::getPhraseWithTag() {
00377     receivedData = {};
00378     const std::string sql = QueryHelper::getPhrasesWithTag();
00379
00380     rc = sqlite3_exec(db, sql.c_str(), callback, (void*)data, &zErrMsg);
00381
00382     if( rc != SQLITE_OK ) {
00383         return {{"SQL ERROR",zErrMsg}};
00384     } else {
00385         return receivedData;
00386     }
00387 }
00388
00389 bool DBmanager::loginUser(std::string &log, std::string &pass) {
00390
00391     const char* sql = QueryHelper::loginUser(log,pass).c_str();
00392
00393     receivedData = {};
00394
00395     rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
00396
00397
00398     if( rc != SQLITE_OK ) {
00399         return false;
00400     } else {
00401         if(receivedData.empty()) {
00402             return false;
00403         }
00404         return true;
00405     }
00406
00407 }
00408
00409 DBmanager::DBmanager()
00410 {
00411     std::ifstream f("ss.db");
00412     if(f.good()) {
00413         this->openDatabase();
00414     }
00415     else {
00416         this->openDatabase();
00417         this->createDatabase();
00418     }
00419 }
00420
00421 DBmanager::~DBmanager()
00422 {
00423     this->closeDatabase();
00424 }
```

8.15 Logic/Database/DBmanager.hpp File Reference

```
#include <string>
#include <vector>
#include <sqlite3.h>
#include "QueryHelper.hpp"
```

Classes

- class DBmanager

8.16 DBmanager.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by lucja on 11.05.2024.
00003 //
00004
00005 #ifndef DBMANAGER_HPP
00006 #define DBMANAGER_HPP
00007
00008 #include<string>
00009 #include<vector>
00010 #include <sqlite3.h>
00011 #include "QueryHelper.hpp"
00012
00017 class DBmanager {
00018     static std::string nickName;
00019     static int id;
00020
00021     sqlite3 *db;
00022     char *zErrMsg;
00023     int rc;
00024     const char* data = "Callback function called";
00025
00026     int openDatabase();
00027     int createDatabase();
00028     int closeDatabase();
00029
00030     int createUserTable();
00031     int createAdminTable();
00032     int createPhraseTable();
00033     int createTagTable();
00034     int createPhraseTagTable();
00035 public:
00036     bool insertUser(std::string& nickname, std::string& password);
00037     std::vector<std::pair<std::string,std::string» getUsers();
00038     bool updateUserPassword(int id,std::string& password);
00039     bool deleteUser(int id);
00040
00041     bool loginUser(std::string& log, std::string& pass);
00042
00043     bool insertAdmin(int Id);
00044     std::vector<std::pair<std::string,std::string» getAdmins();
00045     bool deleteAdmin(int adminId);
00046
00047     bool insertPhrase(std::string &body, std::string &response);
00048     std::vector<std::pair<std::string,std::string»  getPhrases();
00049     std::vector<std::pair<std::string,std::string»  getPhrase(int phraseId);
00050     bool deletePhrase(int id);
00051
00052     bool insertTag(std::string& body);
00053     std::vector<std::pair<std::string,std::string» getTags();
00054     bool deleteTag(int id);
00055
00056     bool insertFavourite(int phraseId);
00057     std::vector<std::pair<std::string,std::string» getFavourites();
00058     bool deleteFavourite(int favId);
00059
00060     bool connectTagToPhrase(int phraseId,int tagId);
00061     std::vector<std::pair<std::string,std::string» getPhraseWithTag();
00062
00063     DBmanager();
00064     ~DBmanager();
00065 };
00066
00067
00068
00069 #endif //DBMANAGER_HPP
```

8.17 Logic/Database/QueryHelper.cpp File Reference

```
#include "QueryHelper.hpp"
```

8.18 QueryHelper.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by lucja on 21.05.2024.
00003 //
00004
00005 #include "QueryHelper.hpp"
00006
00007 std::string QueryHelper::createUserTable() {
00008     return "CREATE TABLE USER("  \
00009        " ID INTEGER PRIMARY KEY AUTOINCREMENT," \
00010        "NICKNAME           TEXT    NOT NULL," \
00011        "PASSWORD           TEXT    NOT NULL);" \
00012     ;
00013 }
00014
00015 std::string QueryHelper::createAdminTable() {
00016     return "CREATE TABLE ADMIN("  \
00017        "ID INTEGER PRIMARY KEY AUTOINCREMENT," \
00018        "USERID INTEGER," \
00019        "FOREIGN KEY (USERID) REFERENCES USER(ID));" \
00020     ;
00021 }
00022
00023 std::string QueryHelper::createPhraseTable() {
00024     return "CREATE TABLE PHRASE("  \
00025        "ID INTEGER PRIMARY KEY AUTOINCREMENT," \
00026        "USERID INTEGER," \
00027        "BODY TEXT NOT NULL," \
00028        "RESPONSE TEXT NOT NULL,"\
00029        "FAVOURITE INTEGER DEFAULT 0,"\
00030        "FOREIGN KEY (USERID) REFERENCES USER(ID));" \
00031     ;
00032 }
00033
00034 std::string QueryHelper::createTagTable() {
00035     return "CREATE TABLE TAG("  \
00036        "ID INTEGER PRIMARY KEY AUTOINCREMENT," \
00037        "BODY TEXT NOT NULL" \
00038        ");" \
00039     ;
00040 }
00041
00042 std::string QueryHelper::createPhraseTagTable() {
00043     return "CREATE TABLE PHRASETAG("  \
00044        "PHRASEID INTEGER," \
00045        "TAGID INTEGER," \
00046        "PRIMARY KEY (PHRASEID, TAGID)," \
00047        "FOREIGN KEY (PHRASEID) REFERENCES PHRASE(ID)," \
00048        "FOREIGN KEY (TAGID) REFERENCES TAG(ID));" \
00049     ;
00050 }
00051
00052 std::string QueryHelper::insertUser(std::string nick, std::string pass) {
00053     return std::format("INSERT INTO USER (NICKNAME,PASSWORD) VALUES ('{}', '{}' );",nick,pass );
00054 }
00055
00056 std::string QueryHelper::getUsers() {
00057     return "SELECT * from USER;";
00058 }
00059
00060 std::string QueryHelper::deleteUser(int id) {
00061     return std::format("DELETE from USER where ID={}; ",id);
00062 }
00063
00064 std::string QueryHelper::updateUserPass(int id, std::string pass)
00065 {
00066     return std::format("UPDATE USER set PASSWORD = '{}' where ID={}; ",pass,id );
00067 }
00068
00069 std::string QueryHelper::insertAdmin(int userId) {
00070     return std::format("INSERT INTO ADMIN (USERID) VALUES ({});",userId );
00071 }
00072
00073 std::string QueryHelper::getAdmins() {
00074     return "SELECT * from ADMIN";
00075 }
00076
00077 std::string QueryHelper::deleteAdmin(int adminId) {
00078     return std::format("DELETE from ADMIN where ID={}; ",adminId);
00079 }
00080
00081 std::string QueryHelper::loginUser(std::string &log, std::string &pass) {
00082     return std::format("SELECT * from USER where NICKNAME='{}' AND PASSWORD='{}' ;",log,pass);
```

```
00083 }
00084
00085 std::string QueryHelper::insertPhrase(int &id, std::string &body, std::string &response) {
00086     return std::format("INSERT INTO PHRASE (USERID,BODY,RESPONSE) VALUES ({},
      '{}','{}');",id,body,response );
00087 }
00088
00089 std::string QueryHelper::getPhrases() {
00090     return "SELECT * from PHRASE";
00091 }
00092
00093 std::string QueryHelper::getPhrase(int phraseId) {
00094     return std::format("SELECT * from PHRASE WHERE ID={}",phraseId);
00095 }
00096
00097 std::string QueryHelper::deletePhrase(int id) {
00098     return std::format("DELETE from PHRASE where ID={}; ",id);
00099 }
00100
00101 std::string QueryHelper::insertTag(std::string body) {
00102    return std::format( "INSERT INTO TAG (BODY) VALUES ('{}');",body);
00103 }
00104
00105 std::string QueryHelper::getTags() {
00106     return "SELECT * from TAG";
00107 }
00108
00109 std::string QueryHelper::deleteTag(int id) {
00110     return std::format("DELETE from TAG where ID={}; ",id);
00111 }
00112
00113 std::string QueryHelper::insertFavourite(int phraseId) {
00114     return std::format("UPDATE PHRASE set FAVOURITE=1 where ID={}; ",phraseId );
00115 }
00116
00117 std::string QueryHelper::getFavourites(int userId) {
00118     return std::format("SELECT * FROM PHRASE WHERE FAVOURITE=1 AND USERID={};",userId);
00119 }
00120
00121 std::string QueryHelper::deleteFavourite(int phraseId) {
00122     return std::format("UPDATE PHRASE set FAVOURITE=0 where ID={}; ",phraseId );
00123 }
00124
00125 std::string QueryHelper::connectTagToPhrase(int phraseId, int tagId) {
00126     return std::format( "INSERT INTO PHRASETAG (PHRASEID,TAGID) VALUES ({},{});",phraseId,tagId);
00127 }
00128
00129 std::string QueryHelper::getPhrasesWithTag() {
00130     return "SELECT p.Id AS PhraseId,p.Body AS PhraseBody,p.Response AS PhraseResponse, t.Body AS
      TagBody FROM PHRASE p JOIN "\
00131     "PHRASETAG pt ON p.id = pt.PhraseId JOIN Tag t ON pt.TagId = t.ID";
00132 }
```

8.19 Logic/Database/QueryHelper.hpp File Reference

```
#include <string>
#include <format>
```

Classes

- class QueryHelper

8.20 QueryHelper.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by lucja on 21.05.2024.
00003 //
00004
```

```
00005 #ifndef QUERYHELPER_HPP
00006 #define QUERYHELPER_HPP
00007
00008 #include<string>
00009 #include <format>
00010
00015 class QueryHelper {
00016 public:
00017      static std::string createUserTable();
00018      static std::string createAdminTable();
00019      static std::string createPhraseTable();
00020      static std::string createTagTable();
00021      static std::string createPhraseTagTable();
00022
00023      static std::string insertUser(std::string nick, std::string pass);
00024      static std::string getUsers();
00025      static std::string deleteUser(int id);
00026      static std::string updateUserPass(int id, std::string pass);
00027
00028      static std::string insertAdmin(int userId);
00029      static std::string getAdmins();
00030      static std::string deleteAdmin(int adminId);
00031
00032      static std::string loginUser(std::string &log, std::string &pass);
00033
00034      static std::string insertPhrase(int &id,std::string &body, std::string &response);
00035      static std::string getPhrases();
00036      static std::string getPhrase(int phraseId);
00037      static std::string deletePhrase(int id);
00038
00039      static std::string insertTag(std::string body);
00040      static std::string getTags();
00041      static std::string deleteTag(int id);
00042
00043      static std::string insertFavourite(int phraseId);
00044      static std::string getFavourites(int userId);
00045      static std::string deleteFavourite(int phraseId);
00046
00047      static std::string connectTagToPhrase(int phraseId,int tagId);
00048      static std::string getPhrasesWithTag();
00049 };
00050
00051
00052 #endif //QUERYHELPER_HPP
```

8.21 Logic/PromptSingleton.cpp File Reference

```
#include "PromptSingleton.hpp"
#include <utility>
```

Functions

- std::string GetMatch (std::string &text, std::vector< std::string > dict)

8.21.1 Function Documentation

8.21.1.1 GetMatch()

```
std::string GetMatch (
            std::string & text,
            std::vector< std::string > dict)
```

Local function implemented to check for matches with dictionary in getPromptAuto

Parameters

text	text to be matched
dict	dict to search from

Returns

Definition at line 15 of file PromptSingleton.cpp.

8.22 PromptSingleton.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #include "PromptSingleton.hpp"
00006
00007 #include <utility>
00008
00015 std::string GetMatch(std::string &text, std::vector<std::string> dict) {
00016     bool nEqual = false;
00017     int len = dict.size();
00018     for (int i = 0; i < len; i++) {
00019         int size = (text.size()>dict[i].size() ? dict[i].size() : text.size());
00020         nEqual = false;
00021         for(int j = 0; j < size; j++)
00022         {
00023             if(text[j] != dict[i][j])
00024             {
00025                 nEqual = true;
00026                 break;
00027             }
00028         }
00029         if(!nEqual)
00030         {
00031             return dict[i];
00032         }
00033     }
00034     return text;
00035 }
00036
00041 void PromptSingleton::SetValues(std::string &val) {
00042     this->prompt = val;
00043 }
00044
00049 PromptSingleton *PromptSingleton::GetInstance() {
00050     if (instancePtr == nullptr)
00051     {
00052         instancePtr = new PromptSingleton();
00053         return instancePtr;
00054     }
00055     else
00056     {
00057         return instancePtr;
00058     }
00059 }
00060
00064 void PromptSingleton::GetPrompt() {
00065     std::getline(std::cin, prompt);
00066 }
00067
00072 void PromptSingleton::GetPromptAuto(std::vector<std::string> dict) {
00073     std::getline(std::cin, prompt);
00074     std::string t1 = this->RetValues();
00075     std::string temp = GetMatch(t1, std::move(dict));
00076     this->SetValues(temp);
00077 }
```

8.23 Logic/PromptSingleton.hpp File Reference

```
#include <string>
#include <iostream>
#include <vector>
```

Classes

- class PromptSingleton

8.24 PromptSingleton.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
00007
00008 #include <string>
00009 #include <iostream>
00010 #include <vector>
00011
00015 class PromptSingleton{
00016 private:
00017     std::string prompt;
00018     static PromptSingleton* instancePtr;
00019     PromptSingleton()= default;
00020 public:
00021     PromptSingleton(const PromptSingleton& obj)
00022     = delete;
00027     static PromptSingleton* GetInstance();
00032     void SetValues(std::string& val);
00037     std::string RetValues(){ return prompt; }
00041     void GetPrompt();
00046     void GetPromptAuto(std::vector<std::string> dict);
00047 };
00048
00049
00050
00051
00052
00053 #endif //INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
```

8.25 Logic/StackApi/StackManager.cpp File Reference

```
#include "StackManager.hpp"
#include <iostream>
#include "cpr/cpr.h"
#include "nlohmann/json.hpp"
#include <string>
#include "../TagList/TagsList.hpp"
```

8.26 StackManager.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 10.05.2024.
00003 //
00004
00005 #include "StackManager.hpp"
00006 #include <iostream>
00007 #include "cpr/cpr.h"
00008 #include "nlohmann/json.hpp"
00009 #include <string>
00010 #include "../TagList/TagsList.hpp"
00011
00012 //SEARCH:
      https://api.stackexchange.com/2.3/search/advanced?order=desc&sort=relevance&q=how%20to%20declare%20array%20of%20string%2
00013 void StackManager::AskQuestion(std::string & question) {
00014     cpr::Response r = cpr::Get(cpr::Url{finalInput});
00015     question = r.text;
00016     //return  finalInput;
00017 }
00018 void StackManager::SetQuestion(std::string newInput) {
00019     questionInput = regex_replace(newInput, std::regex(" "), space);
00020     finalInput =
      baseInput+apiVesion+"search/advanced?order=desc&sort=relevance&q="+questionInput+"&site=stackoverflow&filter=withbody";
00021 }
00022 void StackManager::SetQuestionByTags(std::string newInput) {
00023     questionInput = regex_replace(newInput, std::regex(" "), space);
00024     // finalInput =
      baseInput+apiVesion+"search?pagesize=1&order=desc&sort=votes&intitle=="+questionInput+"&site=stackoverflow&filter=withbo
00025     finalInput =
      baseInput+apiVesion+"search/advanced?order=desc&sort=activity&tagged="+questionInput+"&site=stackoverflow";
00026
00027 }
00028 void StackManager::GetAnswer(std::string res) {
00029     SetQuestionId(res);
00030     stringQuestionID=std::to_string(questionID);
00031     //answerInput = std::to_string(temp);
00032     answerInput =
      baseInput+apiVesion+"questions/"+stringQuestionID+"/answers?pagesize=3&order=desc&sort=votes&site=stackoverflow&filter=w
00033     FillTabel(answerInput);
00034 }
00035
00036 void StackManager::ChangeJsonToString(std::string & input) {
00037
00038     nlohmann::json data = nlohmann::json::parse(input);
00039     if (data.contains("items") && data["items"].is_array()) {
00040         nlohmann::json item = data["items"][0];
00041         if (item.contains("body")) {
00042             std::string body = item["body"];
00043             input = body;
00044         } else {
00045             input = "Can not find similar problem";
00046         }
00047     } else {
00048         input = "Can not find similar problem";
00049     }
00050
00051 }
00052
00053 void StackManager::SetQuestionId(std::string input) {
00054
00055     nlohmann::json data = nlohmann::json::parse(input);
00056
00057     if (data.contains("items") && data["items"].is_array()) {
00058         nlohmann::json item = data["items"][0];
00059         if (item.contains("question_id")) {
00060             questionID = item["question_id"];
00061             title = item["title"];
00062         } else {
00063             title = "Not found";
00064             questionID = 0;
00065         }
00066     } else {
00067         title = "Not found";
00068         questionID = 0;
00069     }
00070 }
00071
00072 void StackManager::FillTabel(std::string input) {
00073     cpr::Response r = cpr::Get(cpr::Url{input});
00074     std::string jsonText = r.text;
00075     nlohmann::json data = nlohmann::json::parse(jsonText);
00076     if (data.contains("items") && data["items"].is_array()) {
00077         // Iteruj przez elementy w "items" (maksymalnie 3 pierwsze)
```

```
00078         for (int i = 0; i < std::min(3, (int)data["items"].size()); ++i) {
00079             nlohmann::json item = data["items"][i];
00080             if (item.contains("body")) {
00081                 std::string bodyContent = item["body"].get<std::string>();
00082                 bestAnswer[i] = bodyContent;
00083             } else {
00084                 std::cout « "Answer not found " « std::endl;
00085                 //bestAnswer[i] = "Brak zawartości";
00086             }
00087         }
00088     } else {
00089         std::cout « "" « std::endl;
00090     }
00091 }
00092 void StackManager::RemoveHtmlTags(std::string& input) {
00093     std::regex htmlTagRegex(R"(<(?!\/?code)[^>]*>)");
00094     input = std::regex_replace(input, htmlTagRegex, "");
00095 }
00096 void StackManager::ChangingSpecialChar(std::string &input,std::string inChar, std::string outChar) {
00097     int pos = input.find(inChar);
00098     while (pos != std::string::npos) {
00099         input.replace(pos, inChar.length(), outChar);
00100         pos = input.find(inChar, pos + 1);
00101     }
00102 }
00103 void StackManager::ReturnNiceCode(std::string& input) {
00104     ChangingSpecialChar(input, "&lt;", "<");
00105     ChangingSpecialChar(input, "&gt;", ">");
00106     ChangingSpecialChar(input, "&quot;", "\"");
00107     ChangingSpecialChar(input, "&amp;", "&");
00108 }
00109 //https://api.stackexchange.com/questions?site=stackoverflow&tagged=c++;java;&filter=withbody
00110 void StackManager::LookForByTags(std::string &input) {
00111     questionInput = regex_replace(input, std::regex(" "), ";");
00112     finalInput =
      baseInput+apiVesion+"questions?site=stackoverflow&tagged="+questionInput+"&filter=withbody";
00113 }
00114 void StackManager::checkTagQuestionList(std::string &tagInput) {
00115     nlohmann::json data = nlohmann::json::parse(tagInput);
00116
00117     questionsList.clear();
00118     for(int i = 0; i < 20; i++)
00119     {
00120     if (data.contains("items") && data["items"].is_array()) {
00121         nlohmann::json item = data["items"][i];
00122         if (item.contains("question_id")) {
00123             questionID = item["question_id"];
00124             title = item["title"];
00125
00126             TagsList tl = TagsList(questionID,title);
00127             questionsList.push_back(tl);
00128
00129         } else {
00130             title = "Not found";
00131             questionID = 0;
00132         }
00133     } else {
00134         title = "Not found";
00135         questionID = 0;
00136     }
00137     }
00138 }
00139
00140 std::vector<TagsList> StackManager::getQuestionList(){
00141     return questionsList;
00142 }
00143
00144 std::string StackManager::GetTitle() {
00145     return this->title;
00146 }
00147
00148 std::string StackManager::GetQuestionId() {
00149     return std::to_string(this->questionID);
00150 }
00151
00152
      //https://api.stackexchange.com/2.3/questions/75339138?order=desc&sort=activity&site=stackoverflow&filter=withbody
00153 void StackManager::GetQuestionFromID(std::string id) {
00154     finalInput = baseInput + apiVesion + "questions/" + id +
      "?order=desc&sort=activity&site=stackoverflow&filter=withbody";
00155 }
00156
00157 std::vector<TagsList> StackManager::questionsList;
00158
```

8.27 Logic/StackApi/StackManager.hpp File Reference

```
#include <algorithm>
#include <iostream>
#include <regex>
#include "nlohmann/adl_serializer.hpp"
#include "../TagList/TagsList.hpp"
```

Classes

- class StackManager

8.28 StackManager.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 10.05.2024.
00003 //
00004
00005 #ifndef STACKMANAGER_HPP
00006 #define STACKMANAGER_HPP
00007
00008 #include <algorithm>
00009 #include <iostream>
00010 #include <regex>
00011 #include "nlohmann/adl_serializer.hpp"
00012 #include "../TagList/TagsList.hpp"
00013
00017 class StackManager {
00018     //URL TO SEARCH
00019     //https://api.stackexchange.com/2.3/search?order=desc&sort=activity&intitle=CPP&site=stackoverflow
00020
00021     //value which store answear id "accepted_answer_id": 63548573,
00022     //URL TO FIND ANSWEAR
00023
      //https://api.stackexchange.com/2.3/answers/63548573?order=desc&sort=activity&site=stackoverflow&filter=withbody
00024
00025     std::string space = "%20";
00026     std::string baseInput = "https://api.stackexchange.com/";
00027     std::string apiVesion = "2.3/";
00028     std::string questionInput = "";
00029     std::string finalInput = "";
00030     std::string answerInput = "";
00031     std::string stringQuestionID = "";
00032     int questionID = 0;
00033     std::string title;
00034     static std::vector<TagsList> questionsList;
00035
00036
00037
00038 public:
00039
00040
00041     std::string bestAnswer[3] = {"","",""};
00042     void AskQuestion(std::string & question);
00043     void SetQuestion(std::string newInput);
00044     void SetQuestionByTags(std::string newInput);
00045     void GetAnswer(std::string res);
00046     void ChangeJsonToString(std::string&);
00047     void SetQuestionId(std::string);
00048     void FillTabel(std::string input);
00049     void RemoveHtmlTags(std::string& input);
00050     void ReturnNiceCode(std::string& input);
00051     void ChangingSpecialChar(std::string &input,std::string inChar, std::string outChar);
00052     void LookForByTags(std::string& input);
00053     void checkTagQuestionList(std::string& tagInput);
00054     static std::vector<TagsList> getQuestionList();
00055     std::string GetTitle();
00056     std::string GetQuestionId();
00057     void GetQuestionFromID(std::string id);
00058 };
00059
00060
00061
00062 #endif //STACKMANAGER_HPP
```

8.29 Logic/StackApi/Syntax.hpp File Reference

```
#include <vector>
#include <string>
```

Namespaces

- namespace Syntax

8.30 Syntax.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 23.05.2024.
00003 //
00004 #include <vector>
00005 #include <string>
00006
00007 #ifndef SYNTAX_HPP
00008 #define SYNTAX_HPP
00009
00013 namespace Syntax {
00014  static std::vector<std::string> basicSyntax = {
00015   "for", "while", "do",
00016   "if", "else", "int",
00017   "string", "::", "std",
00018   "double", "float", "bool",
00019   "main", "switch", "case",
00020   "char", "cin", "getline",
00021   "cout", "return", "long",
00022   "short", "cerr", "«",
00023   "»", "include", "using",
00024   "NULL", "nullptr", "class",
00025   "void", "private", "public",
00026   "*", "&", "\"",
00027   "=", "const", "static",
00028   "delete", "new", "break",
00029   "continue", "protected", "enum",
00030   "typedef", "try",
00031   "catch", "throw", "template",
00032   "operator", "this", "friend",
00033   "volatile", "extern", "struct",
00034   "sizeof", "finally", "AND",
00035   "OR", "&&", "||",
00036   "false", "true",
00037   //PYTHONE
00038   "False", "None", "True",
00039   "and", "as", "assert",
00040   "def", "del", "await",
00041   "async", "elif", "except",
00042   "global", "from", "import",
00043   "in", "is", "lambda",
00044   "not", "!", "raise",
00045   "with", "pass", "yield",
00046   //C KEYWORD
00047   "auto", "default", "inline",
00048   "signed", "malloc", "printf",
00049   "free",
00050   //JAVA KEYWORD
00051   "abstract", "boolen", "implements",
00052   "interface", "native", "package",
00053   "super",
00054   //PHP KEYWORD
00055   "array", "clone", "declare",
00056   "echo", "elseif", "foreach",
00057   "empty", "endfor", "endif",
00058   "endforeach", "endswitch", "isset",
00059   "unset", "var", "use",
00060   "xor",
00061   //JS KEYWORD
00062   "let", "function", "export",
00063   //HTML TAGS
00064   "div", "<", ">",
```

```
00065   "area", "blockquote", "body",
00066   "html", "head", "button",
00067   "dl", "dt", "h1",
00068   "h2", "h3", "h4",
00069   "h5", "h6", "nav",
00070   "script", "strong", "style",
00071   "td", "table", "sup",
00072   "ul", "ol", "li",
00073   "p", "b", "s",
00074   "i", "br", "td",
00075   "a", "img", "tr",
00076   //others
00077   "print", "namespace", "__name__",
00078   "__main__", "__init__",
00079   //css
00080   "display", "position", "top",
00081   "float", "clear", "both",
00082   "width", "height", "min-height",
00083   "min-width", "margin", "padding",
00084   "color", "font", "text-align",
00085   "text-decoration", "letter-spacing", "border",
00086   "transform", "transition", "flex",
00087   "flex-align", "flex-directory", "flex-wrap",
00088   "justift-content", "grid", "grid-template-columns",
00089   "grid-templeta-rows", "cursor", "pointer",
00090   ":hover", ":focus", "visted",
00091   "margin-left", "margin-right", "margin-top",
00092   "margin-bottom", "left", "right",
00093   "bottom", "overflow", "hidden",
00094   "background-color", "background", "opactity",
00095   "absolute", "fixed", "style",
00096   "span", "input", "placeholder",
00097   "#ifndef", "define", "regex",
00098     "println"
00099
00100 };
00101 static std::vector<std::string> keyWord = {
00102   //CPP KEYWORD
00103   "\033[0;32mfor\033[0m", "\033[0;32mwhile\033[0m", "\033[0;32mdo\033[0m",
00104   "\033[0;34mif\033[0m", "\033[0;34melse\033[0m", "\033[0;33mint\033[0m",
00105   "\033[0;33mstring\033[0m", "\033[0;31m::\033[0m", "\033[0;35mstd\033[0m",
00106   "\033[0;33mdouble\033[0m", "\033[0;33mfloat\033[0m", "\033[0;33mbool\033[0m",
00107   "\033[0;34mmain\033[0m", "\033[0;36mswitch\033[0m", "\033[0;33mcase\033[0m",
00108   "\033[0;33mchar\033[0m", "\033[0;31mcin\033[0m", "\033[0;31mgetline\033[0m",
00109   "\033[0;31mcout\033[0m", "\033[0;31mreturn\033[0m", "\033[0;33mlong\033[0m",
00110   "\033[0;33mshort\033[0m", "\033[0;31mcerr\033[0m", "\033[0;32m«\033[0m",
00111   "\033[0;32m»\033[0m", "\033[0;33minclude\033[0m", "\033[0;32musing\033[0m",
00112   "\033[0;32mNULL\033[0m", "\033[0;32mnullptr\033[0m", "\033[0;33mclass\033[0m",
00113   "\033[0;33mvoid\033[0m", "\033[0;31mprivate\033[0m", "\033[0;32mpublic\033[0m",
00114   "\033[0;34m*\033[0m", "\033[0;34m&\033[0m", "\033[0;32m\"\033[0m",
00115   "\033[0;33m=\033[0m", "\033[0;35mconst\033[0m", "\033[0;35mstatic\033[0m",
00116   "\033[0;31mdelete\033[0m", "\033[0;36mnew\033[0m", "\033[0;31mbreak\033[0m",
00117   "\033[0;33mcontinue\033[0m", "\033[0;33mprotected\033[0m", "\033[0;33menum\033[0m",
00118   "\033[0;32mtypedef\033[0m", "\033[0;36mtry\033[0m",
00119   "\033[0;36mcatch\033[0m", "\033[0;31mthrow\033[0m", "\033[0;34mtemplate\033[0m",
00120   "\033[0;33moperator\033[0m", "\033[0;32mthis\033[0m", "\033[0;35mfriend\033[0m",
00121   "\033[0;33mvolatile\033[0m", "\033[0;33mextern\033[0m", "\033[0;33mstruct\033[0m",
00122   "\033[0;33msizeof\033[0m", "\033[0;33mfinally\033[0m", "\033[0;32mAND\033[0m",
00123   "\033[0;32mOR\033[0m", "\033[0;32m&&\033[0m", "\033[0;32m||\033[0m",
00124   "\033[0;32mfalse\033[0m", "\033[0;32mtrue\033[0m",
00125   //PYTHONE KEYWORD
00126   "\033[0;31mfalse\033[0m", "\033[0;33mNone\033[0m", "\033[0;32mtrue\033[0m",
00127   "\033[0;32mand\033[0m", "\033[0;33mAs\033[0m", "\033[0;33mAssert\033[0m",
00128   "\033[0;33mdef\033[0m", "\033[0;31mdel\033[0m", "\033[0;35mawit\033[0m",
00129   "\033[0;35masync\033[0m", "\033[0;34melif\033[0m", "\033[0;31mexcept\033[0m",
00130   "\033[0;36mglobal\033[0m", "\033[0;35mfrom\033[0m", "\033[0;35mimport\033[0m",
00131   "\033[0;35min\033[0m", "\033[0;35mis\033[0m", "\033[0;36mlambdal\033[0m",
00132   "\033[0;31mnot\033[0m", "\033[0;31m!\033[0m", "\033[0;36mraise\033[0m",
00133   "\033[0;36mwith\033[0m", "\033[0;35mpass\033[0m", "\033[0;36myield\033[0m",
00134   //C KEYWORD
00135   "\033[0;33mauto\033[0m", "\033[0;34mdefault\033[0m", "\033[0;34minline\033[0m",
00136   "\033[0;33msigned\033[0m", "\033[0;31mmalloc\033[0m", "\033[0;31mprintf\033[0m",
00137   "\033[0;32mfree\033[0m",
00138   //JAVA KEYWORD
00139   "\033[0;33mabstract\033[0m", "\033[0;33mboolen\033[0m", "\033[0;36mimplements\033[0m",
00140   "\033[0;33mnative\033[0m", "\033[0;35mnative\033[0m", "\033[0;35mpackage\033[0m",
00141   "\033[0;32msuper\033[0m",
00142   //PHP KEYWORD
00143   "\033[0;33marray\033[0m", "\033[0;35mclone\033[0m", "\033[0;35mdeclare\033[0m",
00144   "\033[0;32mecho\033[0m", "\033[0;34melseif\033[0m", "\033[0;32mforeach\033[0m",
00145   "\033[0;36mempty\033[0m", "\033[0;32mendfor\033[0m", "\033[0;34mendif\033[0m",
00146   "\033[0;32mendforeach\033[0m", "\033[0;36mendswitch\033[0m", "\033[0;33misset\033[0m",
00147   "\033[0;33munset\033[0m", "\033[0;36mvar\033[0m", "\033[0;31muse\033[0m",
00148   "\033[0;33mxor\033[0m",
00149   //JS KEYWORD
00150   "\033[0;36mlet\033[0m", "\033[0;32mfunction\033[0m", "\033[0;32mexport\033[0m",
00151   //HTML TAGS
```

```
00152   "\033[0;36mdiv\033[0m", "\033[0;32m<\033[0m", "\033[0;32m>\033[0m",
00153   "\033[0;35marea\033[0m", "\033[0;36mblockquote\033[0m", "\033[0;31mbody\033[0m",
00154   "\033[0;31mhtml\033[0m", "\033[0;31mhead\033[0m", "\033[0;32mbutton\033[0m",
00155   "\033[0;32mdl\033[0m", "\033[0;32mdt\033[0m", "\033[0;34mh1\033[0m",
00156   "\033[0;34mh2\033[0m", "\033[0;34mh3\033[0m", "\033[0;34mh4\033[0m",
00157   "\033[0;34mh5\033[0m", "\033[0;34mh6\033[0m", "\033[0;32mnav\033[0m",
00158   "\033[0;33mscript\033[0m", "\033[0;31mstrong\033[0m", "\033[0;32mstyle\033[0m",
00159   "\033[0;32mtd\033[0m", "\033[0;32mtable\033[0m", "\033[0;32msup\033[0m",
00160   "\033[0;33mul\033[0m", "\033[0;33mol\033[0m", "\033[0;33mli\033[0m",
00161   "\033[0;33mp\033[0m", "\033[0;31mb\033[0m", "\033[0;33ms\033[0m",
00162   "\033[0;35mi\033[0m", "\033[0;35mbr\033[0m", "\033[0;31mtd\033[0m",
00163   "\033[0;34ma\033[0m", "\033[0;32mimg\033[0m", "\033[0;31mtr\033[0m",
00164   //others
00165   "\033[0;31mprint\033[0m", "\033[0;32mnamespace\033[0m", "\033[0;35m__name__\033[0m",
00166   "\033[0;35m__main__\033[0m", "\033[0;35m__init__\033[0m",
00167   //css
00168   "\033[0;31mdisplay\033[0m", "\033[0;31mposition\033[0m", "\033[0;31mtop\033[0m",
00169   "\033[0;31mfloat\033[0m", "\033[0;31mfloat\033[0m", "\033[0;32mdboth\033[0m",
00170   "\033[0;31mwidth\033[0m", "\033[0;31mheight\033[0m", "\033[0;31mmin-height\033[0m",
00171   "\033[0;31mmin-width\033[0m", "\033[0;31mmargin\033[0m", "\033[0;31mpadding\033[0m",
00172   "\033[0;31mcolor\033[0m", "\033[0;31mfont\033[0m", "\033[0;31mtext-align\033[0m",
00173   "\033[0;31mtext-decoration\033[0m", "\033[0;31mletter-spacing\033[0m", "\033[0;31mborder\033[0m",
00174   "\033[0;31mtransform\033[0m", "\033[0;31mtransition\033[0m", "\033[0;31mflex\033[0m",
00175   "\033[0;31mflex-align\033[0m", "\033[0;31mflex-directory\033[0m", "\033[0;31mflex-wrap\033[0m",
00176   "\033[0;31mjustify-content\033[0m", "\033[0;31mgrid\033[0m",
     "\033[0;31mgrid-template-columns\033[0m",
00177   "\033[0;31mgrid-template-rows\033[0m", "\033[0;31mcursor\033[0m", "\033[0;32mpointer\033[0m",
00178   "\033[0;35mhover\033[0m", "\033[0;35mfocus\033[0m", "\033[0;35mvistied\033[0m",
00179   "\033[0;31mmargin-left\033[0m", "\033[0;31mmargin-right\033[0m", "\033[0;31mmargin-top\033[0m",
00180   "\033[0;31mmargin-bottom\033[0m", "\033[0;31mleft\033[0m", "\033[0;31mright\033[0m",
00181   "\033[0;31mbottom\033[0m", "\033[0;31moverflow\033[0m", "\033[0;32mhidden\033[0m",
00182   "\033[0;31mbackground-color\033[0m", "\033[0;31mbackground\033[0m", "\033[0;31mopacity\033[0m",
00183   "\033[0;32mabsolute\033[0m", "\033[0;32mfixed\033[0m", "\033[0;32mstyle\033[0m",
00184   "\033[0;31mspan\033[0m", "\033[0;33minput\033[0m", "\033[0;32mplaceholder\033[0m",
00185   "\033[0;35mifndef\033[0m", "\033[0;34mdefine\033[0m", "\033[0;32mregex\033[0m",
00186   "\033[0;36mprintln\033[0m"
00187  };
00188  static std::vector<std::string> specialCharacter = {
00189   "<", ">", "\"",
00190   "\'", "*", "&",
00191   "|", "$", ":",
00192   "->", "#", "%"
00193  };
00194  static std::vector<std::string> colorSpecialCharacter = {
00195   "\033[0;32m<\033[0m", "\033[0;32m>\033[0m", "\033[0;32m\"\033[0m",
00196   "\033[0;31m\'\033[0m", "\033[0;34m*\033[0m", "\033[0;34m&\033[0m",
00197   "\033[0;33m|\033[0m", "\033[0;34m$\033[0m", "\033[0;32m:\033[0m",
00198   "\033[0;32m->\033[0m", "\033[0;33m#\033[0m", "\033[0;33m%\033[0m"
00199  };
00200 }
00201 #endif //SYNTAX_HPP
```

8.31 Logic/StackApi/SyntaxHighlighting.cpp File Reference

```
#include "SyntaxHighlighting.hpp"
#include <string>
#include "nlohmann/json.hpp"
#include <regex>
#include "Syntax.hpp"
#include <sstream>
```

8.32 SyntaxHighlighting.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 20.05.2024.
00003 //
00004
00005 #include "SyntaxHighlighting.hpp"
00006 #include <string>
00007 #include "nlohmann/json.hpp"
```

```
00008 #include <regex>
00009 #include "Syntax.hpp"
00010 #include <sstream>
00011
00012 SyntaxHighlighting::SyntaxHighlighting() {
00013     for(int i = 0; i < Syntax::basicSyntax.size(); i++)
00014     {
00015         std::regex wordRegex("\\b" +
00016                              std::regex_replace(Syntax::basicSyntax[i],
      std::regex(R"([-[\]{}()*+?.,\^$|#\s:])"),
00017                                                 R"(\$&<>)") + "\\b");
00018
00019         regexes.push_back(wordRegex);
00020     }
00021 }
00022
00023 void SyntaxHighlighting::RecognizeSyntax(std::string &in) {
00024     std::string sentence;
00025     int codePos = 0;
00026     std::string result;
00027     bool terminate = false;
00028     while (!terminate) {
00029         sentence = in.substr(codePos, in.find("<code>") - codePos);
00030         if (in.find("<code>") != std::string::npos) {
00031             codePos = in.find("<code>");
00032             RemoveTags(in, "<code>", "", codePos);
00033             result += sentence;
00034             sentence = in.substr(codePos, in.find("</code>") - codePos);
00035             if (in.find("</code>") != std::string::npos) {
00036                 codePos = in.find("</code>");
00037                 RemoveTags(in, "</code>", "", codePos);
00038                 result += Hightlighting(sentence);
00039             } else {
00040                 terminate = true;
00041             }
00042         } else {
00043             terminate = true;
00044         }
00045
00046         if (terminate && (codePos != in.length())) {
00047             sentence = in.substr(codePos, in.length() - codePos);
00048             result += sentence;
00049         }
00050     }
00051     in = result;
00052 }
00053
00054 std::string SyntaxHighlighting::Hightlighting(std::string &in) {
00055
00056
00057
00058     for (size_t i = 0; i < regexes.size(); i++) {
00059         in = std::regex_replace(in, regexes[i], Syntax::keyWord[i]);
00060     }
00061     for (int i = 0; i < Syntax::specialCharacter.size(); i++) {
00062         ColorChar(in, Syntax::specialCharacter[i], Syntax::colorSpecialCharacter[i]);
00063     }
00064
00065     ColorBracket(in);
00066
00067     return in;
00068 }
00069
00070 void SyntaxHighlighting::RemoveTags(std::string &input, std::string tag, std::string out, int pos) {
00071     pos = input.find(tag, pos + out.length());
00072     if (pos != std::string::npos) {
00073         input.replace(pos, tag.length(), out);
00074     }
00075 }
00076
00077 void SyntaxHighlighting::ColorChar(std::string &input, std::string tag, std::string out) {
00078
00079     int pos = 0;
00080     pos = input.find(tag, pos + out.length());
00081     while (pos != std::string::npos) {
00082         input.replace(pos, tag.length(), out);
00083         pos = input.find(tag, pos + out.length());
00084
00085         //out=="" ? pos = input.find(tag, pos + 1) : pos = input.find(tag, pos + out.length());
00086
00087     }
00088 }
00089 void SyntaxHighlighting::ColorBracket(std::string &in) {
00090     int numberOfBracket=0;
00091     int tempNumberOfBracket=0;
00092     bool gtSeven = false;
00093
```

```
00094     for(int i=0;i<in.length();i++) {
00095         char letter = in[i];
00096         if(letter == '(') {
00097             numberOfBracket++;
00098             if(numberOfBracket>7) {
00099                 gtSeven = true;
00100                 tempNumberOfBracket++;
00101             } else {
00102                 gtSeven = false;
00103             }
00104             std::string newBracket = "\033[0;3"
00105                                      + std::to_string(gtSeven ? tempNumberOfBracket : numberOfBracket)
00106                                      + "m(\033[0m";
00107             in.replace(i, 1, newBracket);
00108             i += newBracket.length() - 1;
00109 
00110         }else if(letter == ')') {
00111             std::string newBracket = "\033[0;3"
00112                                      + std::to_string(numberOfBracket>7 ? tempNumberOfBracket :
      numberOfBracket)
00113                                      + "m)\033[0m";
00114             in.replace(i, 1, newBracket);
00115             i += newBracket.length() - 1;
00116             numberOfBracket--;
00117             if(numberOfBracket>7) tempNumberOfBracket--;
00118 
00119         }
00120 
00121         else if(letter == '{') {
00122             numberOfBracket++;
00123             if(numberOfBracket>7) {
00124                 gtSeven = true;
00125                 tempNumberOfBracket++;
00126             } else {
00127                 gtSeven = false;
00128             }
00129             std::string newBracket = "\033[0;3"
00130                                      + std::to_string(gtSeven ? tempNumberOfBracket : numberOfBracket)
00131                                      + "m{\033[0m";
00132             in.replace(i, 1, newBracket);
00133             i += newBracket.length() - 1;
00134 
00135         }else if(letter == '}') {
00136             std::string newBracket = "\033[0;3"
00137                                      + std::to_string(numberOfBracket>7 ? tempNumberOfBracket :
      numberOfBracket)
00138                                      + "m}\033[0m";
00139             in.replace(i, 1, newBracket);
00140             i += newBracket.length() - 1;
00141             numberOfBracket--;
00142             if(numberOfBracket>7) tempNumberOfBracket--;
00143 
00144         }
00145     }
00146 }
```

8.33 Logic/StackApi/SyntaxHighlighting.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <regex>
```

Classes

- class SyntaxHighlighting

8.34 SyntaxHighlighting.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 20.05.2024.
00003 //
00004
00005 #ifndef SYNTAXHIGHLIGHTING_HPP
00006 #define SYNTAXHIGHLIGHTING_HPP
00007
00008 #include <iostream>
00009 #include <string>
00010 #include <vector>
00011 #include <regex>
00012
00016 class SyntaxHighlighting {
00017         std::vector<std::regex> regexes;
00018 public:
00019         SyntaxHighlighting();
00020         void RecognizeSyntax(std::string& in);
00021         std::string Hightlighting(std::string &in);
00022         void RemoveTags(std::string &input,std::string tag, std::string out, int pos);
00023         void ColorChar(std::string &input,std::string tag, std::string out);
00024         void ColorBracket(std::string &in);
00025 };
00026
00027
00028
00029 #endif //SYNTAXHIGHLIGHTING_HPP
```

8.35 Logic/TagList/TagsList.cpp File Reference

```
#include "TagsList.hpp"
#include <iostream>
#include <string>
```

8.36 TagsList.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 29.05.2024.
00003 //
00004
00005 #include "TagsList.hpp"
00006
00007 #include <iostream>
00008 #include <string>
00009
00013 int TagsList::GetID(){
00014     return id;
00015 }
00020 std::string TagsList::GetTitle(){
00021     return title;
00022 }
00028 TagsList::TagsList(int _id, std::string& _title){
00029     id=_id;
00030     title=_title;
00031 }
```

8.37 Logic/TagList/TagsList.hpp File Reference

```
#include <iostream>
#include <string>
```

Classes

- class TagsList

8.38 TagsList.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 29.05.2024.
00003 //
00004
00005 #ifndef TAGSLIST_HPP
00006 #define TAGSLIST_HPP
00007
00008 #include <iostream>
00009 #include <string>
00010
00014 class TagsList {
00015     int id;
00016     std::string title;
00017 public:
00018     int GetID();
00019     std::string GetTitle();
00020     TagsList(int _id, std::string& _title);
00021 };
00022
00023
00024
00025 #endif //TAGSLIST_HPP
```

8.39 Logic/TextFormatter.hpp File Reference

```
#include <windows.h>
#include <iostream>
#include <thread>
#include <iomanip>
#include "../Globals.hpp"
#include <algorithm>
#include <cctype>
```

Namespaces

- namespace TextColors

  Viable colors of the text.
- namespace TextFunctions

8.40 TextFormatter.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
00007
00008 #include <windows.h>
00009 #include <iostream>
```

```
00010 #include <thread>
00011 #include <iomanip>
00012 #include "../Globals.hpp"
00013 #include <algorithm>
00014 #include <cctype>
00015
00021 namespace TextColors
00022 {
00023     static int BLUE = 1;
00024     static int GREEN = 2;
00025     static int LIGHTBLUE = 3;
00026     static int RED = 4;
00027     static int PURPLE = 5;
00028     static int YELLOW = 6;
00029     static int WHITE = 7;
00030     static int GREY = 8;
00031     static int BLUEBERRY = 9;
00032     static int LIGHTGREEN = 10;
00033     static int CYAN = 11;
00034     static int ROSE = 12;
00035     static int PINK = 13;
00036     static int BEIGE = 14;
00037 }
00038
00042 namespace TextFunctions{
00043
00048     static void changeTextColor(int color)
00049     {
00050         SetConsoleTextAttribute(cmd::hOutput, color);
00051     }
00052
00058     static void typeWriteMessage(std::string& s, int time)
00059     {
00060         for (const auto c : s) {
00061             std::cout « c « std::flush;
00062             std::this_thread::sleep_for(std::chrono::milliseconds(time));
00063         }
00064         printf("\n");
00065     }
00066
00071     static void print(std::string& message)
00072     {
00073         std::cout«message«std::endl;
00074     }
00075
00082     static bool setCursor(short x, short y)
00083     {
00084         return SetConsoleCursorPosition(cmd::hOutput, {x, y});
00085     }
00086     static COORD GetConsoleCursorPosition(HANDLE hConsoleOutput)
00087     {
00088         CONSOLE_SCREEN_BUFFER_INFO cbsi;
00089         if (GetConsoleScreenBufferInfo(hConsoleOutput, &cbsi))
00090         {
00091             return cbsi.dwCursorPosition;
00092         }
00093         else
00094         {
00095             // The function failed. Call GetLastError() for details.
00096             COORD invalid = { 0, 0 };
00097             return invalid;
00098         }
00099     }
00105     static std::string toLower(std::string data) {
00106
00107          std::transform(data.begin(), data.end(), data.begin(),
00108     [](unsigned char c){ return std::tolower(c); });
00109
00110         return data;
00111     }
00112
00113
00114 }
00115
00116
00117
00118 #endif //INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
```

8.41 main.cpp File Reference

```
#include <fstream>
#include "Engine.hpp"
```

```
#include "Texts/AllTexts.hpp"
#include "Logic/TextFormatter.hpp"
```

Functions

- void PrintHelp (char *argv)
- int main (int argc, char *argv[])

8.41.1 Function Documentation

8.41.1.1 main()

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

Main class of the program Creates Engine and start main lop

Returns

0 if everything executes fine

< Start engine

Definition at line 41 of file main.cpp.

8.41.1.2 PrintHelp()

```
void PrintHelp (
            char * argv)
```

Creating help file and printing it

Parameters

argv	name of parameter, it has to be –help or -h so it executes function

Definition at line 18 of file main.cpp.

8.42 main.cpp

Go to the documentation of this file.

```
00001 //Right now project has implemented FSM but it need
00002 //mechanism to handle functions from specific Model classes
00003
00004 #include <fstream>
00005 #include "Engine.hpp"
00006 #include "Texts/AllTexts.hpp"
00007 #include "Logic/TextFormatter.hpp"
00012 PromptSingleton* PromptSingleton::instancePtr = nullptr;
00013
00018 void PrintHelp(char *argv)
00019 {
00020     std::string p = argv;
00021     if(p == "--help" || p == "-h")
00022     {
00023         std::ofstream outfile ("HELP.txt");
00024         outfile « Manual::manual « std::endl;
00025         outfile.close();
00026
00027         std::ifstream f("HELP.txt");
00028         if (f.is_open())
00029             std::cout«f.rdbuf();
00030         TextFunctions::print(Manual::help);
00031         TextFunctions::print(Manual::exit);
00032         std::cin.get();
00033     }
00034 }
00035
00041 int main(int argc, char *argv[]) {
00042
00043     if(argc == 2)
00044         PrintHelp(argv[1]);
00045
00046     Engine eng = Engine();
00047     bool finish = false;
00048     while(!finish)
00049     {
00050         eng.Run();
00051     }
00052     return 0;
00053 }
```

8.43 README.md File Reference

8.44 States/StateAbout.cpp File Reference

```
#include "StateAbout.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.45 StateAbout.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 01.05.2024.
00003 //
00004
00005 #include "StateAbout.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008
00009
00010
00011 void StateAbout::OnEnter() {
00012     State::OnEnter();
```

```
00013     system("cls");
00014     TextFunctions::print(AboutTexts::title);
00015 }
00016
00020 void StateAbout::OnUpdate() {
00021     State::OnUpdate();
00022     TextFunctions::setCursor(42, 10);
00023     TextFunctions::typeWriteMessage(AboutTexts::aboutText, 1);
00024     TextFunctions::changeTextColor(TextColors::PINK);
00025     TextFunctions::setCursor(42 + AboutTexts::aboutText.length(), 10);
00026     TextFunctions::typeWriteMessage(AboutTexts::appText, 1);
00027     TextFunctions::changeTextColor(TextColors::BEIGE);
00028     TextFunctions::setCursor(32, 12);
00029     TextFunctions::typeWriteMessage(AboutTexts::description, 1);
00030     TextFunctions::setCursor(32, 14);
00031     TextFunctions::typeWriteMessage(AboutTexts::returnText, 1);
00032     TextFunctions::setCursor(32, 16);
00033     prompt->GetPromptAuto(dict);
00034     if (TextFunctions::toLower(prompt->RetValues()) == "return")
00035     {
00036         mFsm.SetCurrentState(States::IDLE);
00037     }
00038     else
00039     {
00040         OnEnter();
00041     }
00042 }
00043
00044 void StateAbout::OnExit() {
00045     State::OnExit();
00046 }
```

8.46 States/StateAbout.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class StateAbout

8.47 StateAbout.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 01.05.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
00007
00008
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012 #include "../Logic/PromptSingleton.hpp"
00013
00014 #include <iostream>
00015 #include <string>
00016 #include <vector>
00017
00021 class StateAbout : public State<States> {
```

```
00022     PromptSingleton* prompt = PromptSingleton::GetInstance();
00023     std::vector<std::string> dict = {
00024             "return"
00025     };
00026 public:
00027     explicit StateAbout(FiniteStateMachine<States>& fsm)
00028             : State<States>(fsm, States::ABOUT, "ABOUT"){}
00029
00030     void OnEnter() override;
00031     void OnUpdate() override;
00032     void OnExit() override;
00033 };
00034
00035
00036 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
```

8.48 States/StateExit.cpp File Reference

```
#include "StateExit.hpp"
```

8.49 StateExit.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateExit.hpp"
00006
00007 void StateExit::OnExit() {
00008     State::OnExit();
00009 }
00010
00011 void StateExit::OnUpdate() {
00012     State::OnUpdate();
00013 }
00014
00015 void StateExit::OnEnter() {
00016     State::OnEnter();
00017 }
```

8.50 States/StateExit.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include <iostream>
#include <string>
```

Classes

- class StateExit

8.51 StateExit.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/PromptSingleton.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015
00020 class StateExit : public State<States> {
00021     PromptSingleton* prompt = PromptSingleton::GetInstance();
00022 public:
00023     explicit StateExit(FiniteStateMachine<States>& fsm)
00024     : State<States>(fsm, States::EXIT, "EXIT"){}
00025
00026     void OnEnter() override;
00027     void OnUpdate() override;
00028     void OnExit() override;
00029 };
00030
00031
00032 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
```

8.52 States/StateFavourites.cpp File Reference

```
#include "StateFavourites.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.53 StateFavourites.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateFavourites.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008
00009
00010
00011
00012 void StateFavourites::OnExit() {
00013     State::OnExit();
00014 }
00015
00019 void StateFavourites::OnUpdate() {
00020     State::OnUpdate();
00021     this->trimmedData = {};
00022     this->indexes ={};
00023     this->ManageData();
00024     TextFunctions::setCursor(32, 10);
00025     TextFunctions::typeWriteMessage(FavouriteTexts::returnText, 1);
00026     TextFunctions::setCursor(4, 12);
00027     TextFunctions::changeTextColor(TextColors::PINK);
00028     TextFunctions::typeWriteMessage(FavouriteTexts::favTheme, 1);
00029     TextFunctions::changeTextColor(TextColors::BEIGE);
00030     TextFunctions::setCursor(4 + FavouriteTexts::favTheme.length() + 1, 13);
00031
00032     TextFunctions::setCursor(0, 14);
```

```
00033     for(int i=0;i<trimmedData.size();i++) {
00034         if(i%2==0){    TextFunctions::changeTextColor(TextColors::WHITE);}
00035         else {    TextFunctions::changeTextColor(TextColors::BEIGE);}
00036         TextFunctions::print(trimmedData[i]);
00037     }
00038     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00039     prompt->GetPromptAuto(dict);
00040     TextFunctions::changeTextColor(TextColors::BEIGE);
00041
00042     int phraseId = CheckFav(prompt->RetValues());
00043
00044     if(phraseId>0) {
00045         int index = indexes[phraseId-1];
00046         if(db.deleteFavourite(index)) TextFunctions::print(FavouriteTexts::successText);
00047         prompt->GetPrompt();
00048         if(prompt->RetValues() == "return") {
00049             mFsm.SetCurrentState(States::MENU);
00050         }
00051         else {
00052             OnEnter();
00053         }
00054     }
00055     else if (TextFunctions::toLower(prompt->RetValues()) == "return")
00056     {
00057         mFsm.SetCurrentState(States::MENU);
00058     }
00059     else
00060     {
00061         OnEnter();
00062     }
00063 }
00064
00065 void StateFavourites::ManageData() {
00066     DBmanager db = DBmanager();
00067
00068     const std::vector<std::pair<std::string,std::string» data = db.getFavourites();
00069     std::string tmp;
00070     int iter = 1;
00071     for(int i = 0;i<data.size();i++) {
00072         if(data[i].first == "ID") {
00073             tmp = std::to_string(iter) + ". ";
00074             iter++;
00075             indexes.push_back(std::stoi(data[i].second));
00076         }
00077         if(data[i].first == "BODY") {
00078             tmp+=  data[i].second;
00079             if(tmp[tmp.size()-1]!='\n') {
00080                 tmp+='\n';
00081             }
00082             trimmedData.push_back(tmp);
00083         }
00084     }
00085 }
00086
00087 int StateFavourites::CheckFav(std::string prompt) {
00088     std::string num;
00089     if(prompt[0]=='d' && prompt.size()>1) {
00090         for(int i =1;i<prompt.size();i++) {
00091             if(isdigit(prompt[i])) {
00092                 num+=prompt[i];
00093             }
00094             else {
00095                 return -1;
00096             }
00097         }
00098     }
00099     else {
00100         return -1;
00101     }
00102     int ret = std::stoi(num);
00103
00104     if(ret > trimmedData.size()) {
00105         return -1;
00106     }
00107
00108     return  ret;
00109 }
00110
00111 void StateFavourites::OnEnter() {
00112     State::OnEnter();
00113     system("cls");
00114     TextFunctions::print(FavouriteTexts::title);
00115 }
```

8.54 States/StateFavourites.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "..//Logic/Database/DBmanager.hpp"
#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
```

Classes

- class StateFavourites

8.55 StateFavourites.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "..//Logic/Database/DBmanager.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016
00020 class StateFavourites : public State<States> {
00021     PromptSingleton* prompt = PromptSingleton::GetInstance();
00022     std::vector<std::string> dict = {
00023             "return"
00024     };
00025
00026     DBmanager db;
00027     std::vector<int> indexes;
00028     std::vector<std::pair<std::string,std::string» data;
00029     std::vector<std::string> trimmedData;
00030
00031     void ManageData();
00032     int CheckFav(std::string);
00033 public:
00034     explicit StateFavourites(FiniteStateMachine<States>& fsm)
00035     : State<States>(fsm, States::FAVOURITES, "FAVOURITES"){}
00036
00037     void OnEnter() override;
00038     void OnUpdate() override;
00039     void OnExit() override;
00040 };
00041
00042
00043 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
```

8.56 States/StateHistory.cpp File Reference

```
#include "StateHistory.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.57 StateHistory.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateHistory.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008
00009 void StateHistory::OnExit() {
00010     State::OnExit();
00011 }
00012
00016 void StateHistory::OnUpdate() {
00017     this->trimmedData = {};
00018     this->ManageData();
00019     State::OnUpdate();
00020     TextFunctions::setCursor(32, 10);
00021     TextFunctions::typeWriteMessage(HistoryTexts::returnText, 1);
00022     TextFunctions::setCursor(4, 12);
00023     TextFunctions::typeWriteMessage(HistoryTexts::historyTheme, 1);
00024     TextFunctions::setCursor(0, 14);
00025
00026     for(int i=0;i<trimmedData.size();i++) {
00027         if(i%2==0){    TextFunctions::changeTextColor(TextColors::WHITE);}
00028         else {    TextFunctions::changeTextColor(TextColors::BEIGE);}
00029         TextFunctions::print(trimmedData[i]);
00030     }
00031     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00032     prompt->GetPromptAuto(dict);
00033     TextFunctions::changeTextColor(TextColors::BEIGE);
00034
00035     int phraseId = CheckFav(prompt->RetValues());
00036
00037     if(phraseId>0 && phraseId <= trimmedData.size()) {
00038         db.insertFavourite(phraseId);
00039         TextFunctions::print(HistoryTexts::successText);
00040         prompt->GetPrompt();
00041         if(prompt->RetValues() == "return") {
00042             mFsm.SetCurrentState(States::MENU);
00043         }
00044         else {
00045             OnEnter();
00046         }
00047     }
00048     else if( TextFunctions::toLower(prompt->RetValues()) == "return")
00049     {
00050         mFsm.SetCurrentState(States::MENU);
00051     }
00052     else {
00053         system("cls");
00054         TextFunctions::print(HistoryTexts::title);
00055     }
00056 }
00057
00058 void StateHistory::ManageData() {
00059     DBmanager db = DBmanager();
00060
00061     const std::vector<std::pair<std::string,std::string» data = db.getPhrases();
00062     std::string tmp;
00063     for(int i = 0;i<data.size();i++) {
00064         if(data[i].first == "ID") {
00065             tmp = data[i].second + ". ";
00066         }
00067         if(data[i].first == "BODY") {
00068             tmp+=  data[i].second;
00069             if(tmp[tmp.size()-1]!='\n') {
00070                 tmp+='\n';
00071             }
00072             this->trimmedData.push_back(tmp);
00073         }
00074     }
00075 }
00076
00077 int StateHistory::CheckFav(std::string prompt) {
00078     std::string num;
00079     if(prompt[0]=='f' && prompt.size()>1) {
00080         for(int i =1;i<prompt.size();i++) {
00081             if(isdigit(prompt[i])) {
00082                 num+=prompt[i];
00083             }
00084             else {
00085                 return -1;
```

```
00086             }
00087         }
00088     }
00089     else {
00090         return -1;
00091     }
00092     return std::stoi(num);
00093 }
00094
00095 void StateHistory::OnEnter() {
00096     State::OnEnter();
00097     TextFunctions::changeTextColor(TextColors::BEIGE);
00098     system("cls");
00099     TextFunctions::print(HistoryTexts::title);
00100 }
```

8.58 States/StateHistory.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "..//Logic/Database/DBmanager.hpp"
#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
```

Classes

- class StateHistory

8.59 StateHistory.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "..//Logic/Database/DBmanager.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016
00020 class StateHistory : public State<States> {
00021     PromptSingleton* prompt = PromptSingleton::GetInstance();
00022     std::vector<std::string> dict = {
00023             "return"
00024     };
00025     DBmanager db;
00026     std::vector<std::string> trimmedData;
00027
00028     void ManageData();
00029     int CheckFav(std::string);
00030 public:
00031     explicit StateHistory(FiniteStateMachine<States>& fsm)
00032     : State<States>(fsm, States::HISTORY, "HISTORY"){}
00033
00034     void OnEnter() override;
00035     void OnUpdate() override;
00036     void OnExit() override;
00037 };
00038
00039
00040 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
```

8.60 States/StateIdle.cpp File Reference

```
#include "StateIdle.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.61 StateIdle.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateIdle.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008
00009
00013 void StateIdle::OnEnter(){
00014     State<States>::OnEnter();
00015     system("cls");
00016     TextFunctions::changeTextColor(TextColors::BEIGE);
00017     TextFunctions::print(IdleTexts::title);
00018 }
00019
00024 void StateIdle::OnUpdate() {
00025
00026     State<States>::OnUpdate();
00027     TextFunctions::setCursor(40, 9);
00028     TextFunctions::typeWriteMessage(IdleTexts::helloIns, 1);
00029     TextFunctions::setCursor(40, 11);
00030     prompt->GetPromptAuto(dict);
00031     if(prompt->RetValues() == "login")
00032     {
00033         mFsm.SetCurrentState(States::LOGIN);
00034     }
00035     else if(prompt->RetValues() == "register")
00036     {
00037         mFsm.SetCurrentState(States::REGISTER);
00038     }
00039     else if(prompt->RetValues() == "about")
00040     {
00041         mFsm.SetCurrentState(States::ABOUT);
00042     }
00043     else
00044     {
00045         OnEnter();
00046         mFsm.SetCurrentState(States::IDLE);
00047     }
00048 }
00049
00053 void StateIdle::OnExit() {
00054     State<States>::OnExit();
00055 }
```

8.62 States/StateIdle.hpp File Reference

```
#include <iostream>
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include <vector>
```

Classes

- class StateIdle

8.63 StateIdle.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00007
00008 #include <iostream>
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012 #include "../Logic/PromptSingleton.hpp"
00013 #include <vector>
00014
00015
00020 class StateIdle : public State<States>{
00021     PromptSingleton* prompt = PromptSingleton::GetInstance();
00022     std::vector<std::string> dict = {
00023             "login",
00024             "register",
00025             "about"
00026     };
00027 public:
00028     explicit StateIdle(FiniteStateMachine<States>& fsm)
00030     : State<States>(fsm, States::IDLE, "IDLE"){}
00031
00032     void OnEnter() override;
00033     void OnUpdate() override;
00034     void OnExit() override;
00035 };
00036
00037
00038 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00039
```

8.64 States/StateListTags.cpp File Reference

```
#include "StateListTags.hpp"
#include "../Logic/TextFormatter.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Globals.hpp"
```

8.65 StateListTags.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 28.05.2024.
00003 //
00004
00005 #include "StateListTags.hpp"
00006 #include "../Logic/TextFormatter.hpp"
00007 #include "../Texts/AllTexts.hpp"
00008 #include "../Globals.hpp"
00009
00010 void StateListTags::OnEnter() {
00011     State::OnEnter();
00012     system("cls");
00013     TextFunctions::print(ListState::title);
```

```
00014 }
00015
00019 void StateListTags::OnUpdate() {
00020     State::OnUpdate();
00021
00022     TextFunctions::setCursor(10, 10);
00023     TextFunctions::print(ListState::tagText);
00024
00025     TextFunctions::changeTextColor(TextColors::WHITE);
00026
00027     ManageList();
00028
00029     bool changeState= false;
00030     while(!changeState) {
00031         prompt->GetPromptAuto(dict);
00032
00033         if(TextFunctions::toLower(prompt->RetValues()) == "return") {
00034             changeState=true;
00035             mFsm.SetCurrentState(States::MENU);
00036             changeState=true;
00037
00038         }else if(!ChoosingTitle(prompt->RetValues())) {
00039             mFsm.SetCurrentState(States::LISTTAGS);
00040             std::string error = "\033[0;31mYou have to choose from the list\033[0m";
00041             COORD position = TextFunctions::GetConsoleCursorPosition(cmd::hOutput);
00042             TextFunctions::print(error);
00043
00044             TextFunctions::setCursor(position.X,position.Y-1);
00045
00046             std::cout « "\x1b[2K";
00047             TextFunctions::changeTextColor(TextColors::YELLOW);
00048             std::cout « "Prompt: ";
00049             TextFunctions::changeTextColor(TextColors::WHITE);
00050             changeState=true;
00051
00052         }
00053         else if (ChoosingTitle(prompt->RetValues()))
00054         {
00055             mFsm.SetCurrentState(States::RESULTTAGS);
00056             changeState=true;
00057         }
00058     }
00059 }
00060
00061 void StateListTags::OnExit() {
00062     State::OnExit();
00063 }
00064
00065 void StateListTags::ManageList() {
00066     sm.SetQuestionByTags(prompt->RetValues());
00067     sm.AskQuestion(question);
00068     std::string jSonTemp = question;
00069     sm.checkTagQuestionList(question);
00070     questionsList = StackManager::getQuestionList();
00071     std::string temp;
00072     if(!questionsList.empty()) {
00073         for(int i = 0; i < questionsList.size(); i++) {
00074             temp = std::to_string(i+1) + ". " + questionsList[i].GetTitle();
00075             sm.ReturnNiceCode(temp);
00076             TextFunctions::print( temp);
00077         }
00078     }
00079     TextFunctions::changeTextColor(TextColors::YELLOW);
00080     std::cout « std::endl « "Prompt: ";
00081     TextFunctions::changeTextColor(TextColors::WHITE);
00082
00083 }
00084
00085 bool StateListTags::ChoosingTitle(std::string in) {
00086
00087     for (int i=0; i<in.length(); i++){
00088         if (!isdigit(in[i])){
00089             return false;
00090         }
00091     }
00092     int questionIdx = std::stoi(in);
00093
00094     if(questionIdx>0 && questionIdx<21) {
00095         return true;
00096     }
00097     return false;
00098
00099 }
```

8.66 States/StateListTags.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include "../Logic/StackApi/StackManager.hpp"
#include "../Logic/TagList/TagsList.hpp"
```

Classes

- class StateListTags

8.67 StateListTags.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 28.05.2024.
00003 //
00004
00005 #ifndef STATELISTTAGS_HPP
00006 #define STATELISTTAGS_HPP
00007 #include "StatesConf.hpp"
00008 #include "../FSM/StateMachine.hpp"
00009 #include "../FSM/State.hpp"
00010 #include "../Logic/PromptSingleton.hpp"
00011 #include "../Logic/StackApi/StackManager.hpp"
00012
00013 #include "../Logic/TagList/TagsList.hpp"
00014
00015
00016
00021 class StateListTags: public State<States>  {
00022     std::string question;
00023     std::vector<TagsList> questionsList;
00024     StackManager sm = StackManager();
00025     PromptSingleton* prompt = PromptSingleton::GetInstance();
00026     std::vector<std::string> dict = {
00027         "return"
00028 };
00029
00030 public:
00031     explicit StateListTags(FiniteStateMachine<States>& fsm)
00032     : State<States>(fsm, States::LISTTAGS, "LISTTAGS"){}
00033     void OnEnter() override;
00034     void OnUpdate() override;
00035     void OnExit() override;
00036     void ManageList();
00037     bool ChoosingTitle(std::string in);
00038 };
00039
00040
00041
00042 #endif //STATELISTTAGS_HPP
```

8.68 States/StateLogin.cpp File Reference

```
#include "StateLogin.hpp"
#include "../Logic/Database/DBmanager.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.69 StateLogin.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateLogin.hpp"
00006
00007 #include "../Logic/Database/DBmanager.hpp"
00008 #include "../Texts/AllTexts.hpp"
00009 #include "../Logic/TextFormatter.hpp"
00010 #include "..//Logic/Database/DBmanager.hpp"
00011
00012 void StateLogin::OnEnter() {
00013     State::OnEnter();
00014     system("cls");
00015     TextFunctions::print(LoginTexts::title);
00016 }
00017
00021 void StateLogin::OnUpdate() {
00022     State::OnUpdate();
00023     TextFunctions::setCursor(32, 10);
00024     TextFunctions::typeWriteMessage(LoginTexts::credentials, 1);
00025     TextFunctions::setCursor(32, 12);
00026     TextFunctions::print(LoginTexts::login);
00027     TextFunctions::setCursor(39, 12);
00028     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00029     prompt->GetPrompt();
00030     TextFunctions::changeTextColor(TextColors::BEIGE);
00031     log = prompt->RetValues();
00032     TextFunctions::setCursor(32, 14);
00033     TextFunctions::print(LoginTexts::password);
00034     TextFunctions::setCursor(42, 14);
00035     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00036     prompt->GetPrompt();
00037     TextFunctions::changeTextColor(TextColors::BEIGE);
00038     pass = prompt->RetValues();
00039     DBmanager db = DBmanager();
00040     if(db.loginUser(log,pass))
00041     {
00042         mFsm.SetCurrentState(States::MENU);
00043
00044     }
00045     else
00046     {
00047         mFsm.SetCurrentState(States::IDLE);
00048     }
00049 }
00050
00051 void StateLogin::OnExit() {
00052     State::OnExit();
00053 }
```

8.70 States/StateLogin.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include <iostream>
#include <string>
```

Classes

- class StateLogin

8.71 StateLogin.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/PromptSingleton.hpp"
00012 #include <iostream>
00013 #include <string>
00014
00018 class StateLogin : public State<States>{
00019     PromptSingleton* prompt = PromptSingleton::GetInstance();
00020     std::string log;
00021     std::string pass;
00022 public:
00023     explicit StateLogin(FiniteStateMachine<States>& fsm)
00024     : State<States>(fsm, States::LOGIN, "LOGIN"){}
00025
00026     void OnEnter() override;
00027     void OnUpdate() override;
00028     void OnExit() override;
00029 };
00030
00031
00032 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP
```

8.72 States/StateMenu.cpp File Reference

```
#include "StateMenu.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.73 StateMenu.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateMenu.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008 using namespace TextFunctions;
00009
00010 void StateMenu::OnEnter() {
00011     State::OnEnter();
00012     system("cls");
00013     print(MenuTexts::title);
00014 }
00015
00019 void StateMenu::OnUpdate() {
00020     State::OnUpdate();
00021     TextFunctions::setCursor(32, 10);
00022     typeWriteMessage(MenuTexts::helloText, 1);
00023     TextFunctions::setCursor(32+ MenuTexts::helloText.length(), 10);
00024     TextFunctions::changeTextColor(TextColors::PINK);
00025     TextFunctions::print(MenuTexts::favText);
00026     TextFunctions::setCursor(32, 12);
00027     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00028     prompt->GetPromptAuto(dict);
00029     TextFunctions::changeTextColor(TextColors::BEIGE);
00030     if(prompt->RetValues() == "question")
00031     {
```

```
00032         mFsm.SetCurrentState(States::PROMPT);
00033     }
00034     else if(prompt->RetValues() == "history")
00035     {
00036         mFsm.SetCurrentState(States::HISTORY);
00037     }
00038     else if(prompt->RetValues() == "tags")
00039     {
00040         mFsm.SetCurrentState(States::TAGS);
00041     }
00042     else if (prompt->RetValues() == "favourites")
00043     {
00044         mFsm.SetCurrentState(States::FAVOURITES);
00045     }
00046     else
00047     {
00048         OnEnter();
00049         mFsm.SetCurrentState(States::MENU);
00050     }
00051 }
00052
00053 void StateMenu::OnExit() {
00054     State::OnExit();
00055 }
```

8.74 States/StateMenu.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class StateMenu

8.75 StateMenu.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/PromptSingleton.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include <vector>
00016
00021 class StateMenu : public State<States> {
00022     PromptSingleton* prompt = PromptSingleton::GetInstance();
00023     std::vector<std::string> dict = {
00024             "question",
00025             "history",
00026             "tags",
00027             "favourites"
00028     };
00029
```

```
00030 public:
00031     explicit StateMenu(FiniteStateMachine<States>& fsm)
00032     : State<States>(fsm, States::MENU, "MENU"){}
00033
00034     void OnEnter() override;
00035     void OnUpdate() override;
00036     void OnExit() override;
00037 };
00038
00039
00040 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
```

8.76 States/StatePrompt.cpp File Reference

```
#include "StatePrompt.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
#include <nlohmann/json.hpp>
#include <string>
```

8.77 StatePrompt.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StatePrompt.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008 #include <nlohmann/json.hpp>
00009 #include <string>
00010
00011
00012
00013 void StatePrompt::OnEnter() {
00014     State::OnEnter();
00015     system("cls");
00016     TextFunctions::print(PromptTexts::title);
00017 }
00018
00022 void StatePrompt::OnUpdate() {
00023     State::OnUpdate();
00024     TextFunctions::setCursor(10, 10);
00025     TextFunctions::typeWriteMessage(PromptTexts::promptText, 30);
00026     prompt->GetPromptAuto(dict);
00027     if(prompt->RetValues() == "return")
00028         mFsm.SetCurrentState(States::MENU);
00029     else
00030         mFsm.SetCurrentState(States::RESULT);
00031
00032
00033     // TextFunctions::typeWriteMessage(tescik, 30);
00034     // std::cout « "Answer 2:" « std::endl;
00035     //
00036     // tescik = sm.bestAnswer[1];
00037     // TextFunctions::typeWriteMessage(tescik, 30);
00038     // std::cout « "Answer 3:" « std::endl;
00039     //
00040     // tescik = sm.bestAnswer[2];
00041     // TextFunctions::typeWriteMessage(tescik, 30);
00042
00043     // std::string answer = sm.getAnswer();
00044     // TextFunctions::typeWriteMessage(answer, 30);
00045     //std::string answer = sm.getAnswer();
00046     //answer = sm.changeJsonToString(answer);
00047     //int answer = sm.getAnswerID(JsonQuestion);
00048
00049 }
00050
00051 void StatePrompt::OnExit() {
00052     State::OnExit();
00053 }
```

8.78 States/StatePrompt.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
#include "../Logic/StackApi/StackManager.hpp"
```

Classes

- class StatePrompt

8.79 StatePrompt.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011
00012 #include <iostream>
00013 #include <string>
00014 #include "../Logic/PromptSingleton.hpp"
00015 #include "../Logic/StackApi/StackManager.hpp"
00016
00020 class StatePrompt : public State<States> {
00021     PromptSingleton* prompt = PromptSingleton::GetInstance();
00022     StackManager sm = StackManager();
00023     std::vector<std::string> dict = {
00024             "return"
00025     };
00026 public:
00027     explicit StatePrompt(FiniteStateMachine<States>& fsm)
00028     : State<States>(fsm, States::PROMPT, "PROMPT"){}
00029
00030     void OnEnter() override;
00031     void OnUpdate() override;
00032     void OnExit() override;
00033 };
00034
00035
00036 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP
```

8.80 States/StateRegister.cpp File Reference

```
#include "StateRegister.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
#include "..//Logic/Database/DBmanager.hpp"
```

8.81 StateRegister.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #include "StateRegister.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008 #include "..//Logic/Database/DBmanager.hpp"
00009 using namespace TextFunctions;
00010
00011 void StateRegister::OnEnter() {
00012     State::OnEnter();
00013     system("cls");
00014     print(RegisterTexts::title);
00015 }
00016
00020 void StateRegister::OnUpdate() {
00021     State::OnUpdate();
00022     TextFunctions::setCursor(32, 10);
00023     typeWriteMessage(RegisterTexts::credentials, 50);
00024     TextFunctions::setCursor(32, 12);
00025     typeWriteMessage(RegisterTexts::login, 0);
00026     TextFunctions::setCursor(39, 12);
00027     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00028     prompt->GetPrompt();
00029     TextFunctions::changeTextColor(TextColors::BEIGE);
00030     log = prompt->RetValues();
00031     TextFunctions::setCursor(32, 14);
00032     typeWriteMessage(RegisterTexts::password,0);
00033     TextFunctions::setCursor(42, 14);
00034     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00035     prompt->GetPrompt();
00036     TextFunctions::changeTextColor(TextColors::BEIGE);
00037     pass = prompt->RetValues();
00038     TextFunctions::setCursor(32, 16);
00039     typeWriteMessage(RegisterTexts::email,0);
00040     TextFunctions::setCursor(39, 16);
00041     TextFunctions::changeTextColor(TextColors::LIGHTGREEN);
00042     prompt->GetPrompt();
00043     TextFunctions::changeTextColor(TextColors::BEIGE);
00044
00045     email = prompt->RetValues();
00046     DBmanager db = DBmanager();
00047     if(db.insertUser(log,pass))
00048     {
00049         mFsm.SetCurrentState(States::LOGIN);
00050     }
00051     else
00052     {
00053         mFsm.SetCurrentState(States::IDLE);
00054     }
00055 }
00056
00057 void StateRegister::OnExit() {
00058     State::OnExit();
00059 }
```

8.82 States/StateRegister.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
```

Classes

- class StateRegister

8.83 StateRegister.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011
00012 #include <iostream>
00013 #include <string>
00014 #include "../Logic/PromptSingleton.hpp"
00015
00019 class StateRegister : public State<States> {
00020     PromptSingleton* prompt = PromptSingleton::GetInstance();
00021     std::string log;
00022     std::string pass;
00023     std::string email;
00024 public:
00025     explicit StateRegister(FiniteStateMachine<States>& fsm)
00026     : State<States>(fsm, States::REGISTER, "REGISTER"){}
00027
00028     void OnEnter() override;
00029     void OnUpdate() override;
00030     void OnExit() override;
00031 };
00032
00033
00034 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP
```

8.84 States/StateResult.cpp File Reference

```
#include "StateResult.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
#include "../Logic/Database/DBmanager.hpp"
```

8.85 StateResult.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
00004
00005 #include "StateResult.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008 #include "../Logic/Database/DBmanager.hpp"
00009
00010 void StateResult::OnEnter() {
00011     State::OnEnter();
00012     TextFunctions::changeTextColor(TextColors::BEIGE);
00013     system("cls");
00014     TextFunctions::print(ResultTexts::title);
00015 }
00016
00017 void StateResult::OnUpdate() {
00019     State::OnUpdate();
00020     TextFunctions::setCursor(32, 10);
00021     TextFunctions::typeWriteMessage(ResultTexts::questionText, 30);
00022     TextFunctions::setCursor(50, 10);
00023     TextFunctions::changeTextColor(TextColors::BEIGE);
00024     question = prompt->RetValues();
00025     TextFunctions::print(question);
00027     QuestionManage();
```

```
00028     prompt->GetPromptAuto(dict);
00029     if(prompt->RetValues() == "return")
00030     {
00031         mFsm.SetCurrentState(States::MENU);
00032     }
00033     else {
00034         mFsm.SetCurrentState(States::PROMPT);
00035     }
00036 }
00037
00038 void StateResult::OnExit() {
00039     State::OnExit();
00040     TextFunctions::changeTextColor(TextColors::BEIGE);
00041 }
00042
00046 void StateResult::QuestionManage() {
00047     sm.SetQuestion(prompt->RetValues());
00048     sm.AskQuestion(question);
00049     std::string jSonTemp = question;
00051     sm.ChangeJsonToString(question);
00052     TextFunctions::print(ResultTexts::questionText);
00053     TextFunctions::changeTextColor(TextColors::WHITE);
00055     sm.RemoveHtmlTags(question);
00057     sm.ReturnNiceCode(question);
00059     sh.RecognizeSyntax(question);
00060     TextFunctions::print(question);
00061     sm.GetAnswer(jSonTemp);
00062     DBmanager db = DBmanager();
00063     std::string body = sm.GetTitle();
00064     std::string response = sm.GetQuestionId();
00065     if(body != "Not found")db.insertPhrase(body,response);
00069     for (int i = 0; i < 3; i++) {
00070         if (sm.bestAnswer[i] != "") {
00071             std::cout << "Answer: " << i + 1 << std::endl;
00072             std::string ans = sm.bestAnswer[i];
00073             sm.RemoveHtmlTags(ans);
00074             sm.ReturnNiceCode(ans);
00075             sh.RecognizeSyntax(ans);
00076             TextFunctions::print(ans);
00077             std::cout << std::endl;
00078         }
00079         sm.bestAnswer[i] = "";
00080     }
00081 }
```

8.86 States/StateResult.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
#include "../Logic/StackApi/StackManager.hpp"
#include "../Logic/StackApi/SyntaxHighlighting.hpp"
```

Classes

- class StateResult

8.87 StateResult.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
```

```
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
00007
00008
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016 #include "../Logic/StackApi/StackManager.hpp"
00017 #include "../Logic/StackApi/SyntaxHighlighting.hpp"
00018
00023 class StateResult : public State<States> {
00024     PromptSingleton* prompt = PromptSingleton::GetInstance();
00025     std::string question;
00026     std::string answer;
00027     StackManager sm = StackManager();
00028     SyntaxHighlighting sh = SyntaxHighlighting();
00029     std::vector<std::string> dict = {
00030             "question",
00031             "return"
00032     };
00033 public:
00034     explicit StateResult(FiniteStateMachine<States>& fsm)
00035             : State<States>(fsm, States::RESULT, "RESULT"){}
00036
00037     void OnEnter() override;
00038     void OnUpdate() override;
00039     void OnExit() override;
00040
00041     void QuestionManage();
00042 };
00043
00044
00045 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
```

8.88 States/StateResultTags.cpp File Reference

```
#include "StateResultTags.hpp"
#include "../Logic/TextFormatter.hpp"
#include "../Texts/AllTexts.hpp"
#include "..//Logic/Database/DBmanager.hpp"
```

8.89 StateResultTags.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 28.05.2024.
00003 //
00004
00005 #include "StateResultTags.hpp"
00006 #include "../Logic/TextFormatter.hpp"
00007 #include "../Texts/AllTexts.hpp"
00008 #include "..//Logic/Database/DBmanager.hpp"
00009
00010 void StateResultTags::OnEnter() {
00011     State::OnEnter();
00012     TextFunctions::changeTextColor(TextColors::BEIGE);
00013     system("cls");
00014     TextFunctions::print(ResultTexts::title);
00015 }
00016
00020 void StateResultTags::OnUpdate() {
00021     State::OnUpdate();
00022     TextFunctions::setCursor(32, 10);
00023     TextFunctions::typeWriteMessage(ResultTexts::questionText, 30);
00024     TextFunctions::setCursor(50, 10);
00025     TextFunctions::changeTextColor(TextColors::BEIGE);
00026     question = prompt->RetValues();
```

```
00027     int index = std::stoi(prompt->RetValues())-1;
00028     std::string title = StackManager::getQuestionList()[index].GetTitle();
00029     TextFunctions::print(title);
00030     QuestionManage();
00031     prompt->GetPromptAuto(dict);
00032     if(prompt->RetValues() == "return")
00033     {
00034         mFsm.SetCurrentState(States::MENU);
00035     }
00036     else {
00037         mFsm.SetCurrentState(States::TAGS);
00038     }
00039 }
00040
00041 void StateResultTags::OnExit() {
00042     State<States>::OnExit();
00043 }
00044
00045 void StateResultTags::QuestionManage() {
00046
00047
00048     int index = std::stoi(prompt->RetValues())-1;
00049     std::string problem = std::to_string(StackManager::getQuestionList()[index].GetID());
00050     sm.GetQuestionFromID(problem);
00051     sm.AskQuestion(question);
00052     std::string jSonTemp = question;
00053     sm.ChangeJsonToString(question);
00054     TextFunctions::print(ResultTexts::questionText);
00055     TextFunctions::changeTextColor(TextColors::WHITE);
00056     sm.RemoveHtmlTags(question);
00057     sm.ReturnNiceCode(question);
00058     sh.RecognizeSyntax(question);
00059     TextFunctions::print(question);
00060     sm.GetAnswer(jSonTemp);
00061     DBmanager db = DBmanager();
00062     std::string body = sm.GetTitle();
00063     std::string response = sm.GetQuestionId();
00064     if(body != "Not found")db.insertPhrase(body,response);
00065     for (int i = 0; i < 3; i++) {
00066         if (sm.bestAnswer[i] != "") {
00067             std::cout << "Answer: " << i + 1 << std::endl;
00068             std::string ans = sm.bestAnswer[i];
00069             sm.RemoveHtmlTags(ans);
00070             sm.ReturnNiceCode(ans);
00071             sh.RecognizeSyntax(ans);
00072             TextFunctions::print(ans);
00073             std::cout << std::endl;
00074         }
00075
00076         sm.bestAnswer[i] = "";
00077     }
00078 }
```

8.90 States/StateResultTags.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"
#include "../Logic/StackApi/StackManager.hpp"
#include "../Logic/StackApi/SyntaxHighlighting.hpp"
```

Classes

- class StateResultTags

8.91 StateResultTags.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by jakub on 28.05.2024.
00003 //
00004
00005 #ifndef STATERESULTTAGS_HPP
00006 #define STATERESULTTAGS_HPP
00007 #include "StatesConf.hpp"
00008 #include "../FSM/StateMachine.hpp"
00009 #include "../FSM/State.hpp"
00010 #include "../Logic/PromptSingleton.hpp"
00011 #include "../Logic/StackApi/StackManager.hpp"
00012 #include "../Logic/StackApi/SyntaxHighlighting.hpp"
00013
00017 class StateResultTags : public State<States>  {
00018     PromptSingleton* prompt = PromptSingleton::GetInstance();
00019     std::string question;
00020     std::string answer;
00021     StackManager sm = StackManager();
00022     SyntaxHighlighting sh = SyntaxHighlighting();
00023     std::vector<std::string> dict = {
00024         "tags",
00025         "return"
00026 };
00027 public:
00028     explicit StateResultTags(FiniteStateMachine<States>& fsm)
00029     : State<States>(fsm, States::RESULTTAGS, "RESULTTAGS"){}
00030
00031     void OnEnter() override;
00032     void OnUpdate() override;
00033     void OnExit() override;
00034
00035     void QuestionManage();
00036
00037 };
00038
00039
00040
00041 #endif //STATERESULTTAGS_HPP
```

8.92 States/StatesConf.hpp File Reference

Enumerations

- enum class States {
  IDLE , LOGIN , REGISTER , MENU ,
  PROMPT , FAVOURITES , TAGS , HISTORY ,
  EXIT , RESULT , ABOUT , RESULTTAGS ,
  LISTTAGS }

8.92.1 Enumeration Type Documentation

8.92.1.1 States

```
enum class States  [strong]
```

File which provides Enumeration of all States possible

Enumerator

IDLE	
LOGIN	
REGISTER	

Enumerator

MENU	
PROMPT	
FAVOURITES	
TAGS	
HISTORY	
EXIT	
RESULT	
ABOUT	
RESULTTAGS	
LISTTAGS	

Definition at line 12 of file StatesConf.hpp.

8.93 StatesConf.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP
00007
00012 enum class States
00013 {
00014     IDLE,
00015     LOGIN,
00016     REGISTER,
00017     MENU,
00018     PROMPT,
00019     FAVOURITES,
00020     TAGS,
00021     HISTORY,
00022     EXIT,
00023     RESULT,
00024     ABOUT,
00025     RESULTTAGS,
00026     LISTTAGS
00027 };
00028
00029 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP
```

8.94 States/StatesWrapper.hpp File Reference

```
#include "StateExit.hpp"
#include "StateLogin.hpp"
#include "StateRegister.hpp"
#include "StateFavourites.hpp"
#include "StateHistory.hpp"
#include "StateMenu.hpp"
#include "StatePrompt.hpp"
#include "StateIdle.hpp"
#include "StateResult.hpp"
#include "StateTags.hpp"
#include "StateAbout.hpp"
#include "StateListTags.hpp"
#include "StateResultTags.hpp"
```

8.95 StatesWrapper.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP
00007
00012 #include "StateExit.hpp"
00013 #include "StateLogin.hpp"
00014 #include "StateRegister.hpp"
00015 #include "StateFavourites.hpp"
00016 #include "StateHistory.hpp"
00017 #include "StateMenu.hpp"
00018 #include "StatePrompt.hpp"
00019 #include "StateIdle.hpp"
00020 #include "StateResult.hpp"
00021 #include "StateTags.hpp"
00022 #include "StateAbout.hpp"
00023 #include "StateListTags.hpp"
00024 #include "StateResultTags.hpp"
00025
00026 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP
```

8.96 States/StateTags.cpp File Reference

```
#include "StateTags.hpp"
#include "../Texts/AllTexts.hpp"
#include "../Logic/TextFormatter.hpp"
```

8.97 StateTags.cpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #include "StateTags.hpp"
00006 #include "../Texts/AllTexts.hpp"
00007 #include "../Logic/TextFormatter.hpp"
00008
00009 void StateTags::OnEnter() {
00010     State::OnEnter();
00011     system("cls");
00012     TextFunctions::print(TagsTexts::title);
00013     std::vector<std::string> _tags;// = DBmanager::getTags();
00014     for (auto tag: _tags) {
00015         tags += tag;
00016     }
00017 }
00018
00022 void StateTags::OnUpdate() {
00023     State::OnUpdate();
00024     TextFunctions::setCursor(32, 10);
00025     TextFunctions::typeWriteMessage(TagsTexts::returnText, 30);
00026     TextFunctions::setCursor(4, 12);
00027     TextFunctions::typeWriteMessage(TagsTexts::tagText, 30);
00028     TextFunctions::setCursor(4 + TagsTexts::tagText.length() +1, 13);
00029     TextFunctions::print(tags);
00030    // TextFunctions::setCursor(4 + TagsTexts::tagText.length() +1, 13+DBmanager::getTags().size());
00031     prompt->GetPrompt();
00032     if (TextFunctions::toLower(prompt->RetValues()) == "return")
00033     {
00034         mFsm.SetCurrentState(States::MENU);
00035     }
00036     else
00037     {
00038         mFsm.SetCurrentState(States::LISTTAGS);
```

```
00039     }
00040 }
00041
00042 void StateTags::OnExit() {
00043     State::OnExit();
00044 }
```

8.98 States/StateTags.hpp File Reference

```
#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "..//Logic/Database/DBmanager.hpp"
#include <iostream>
#include <string>
#include <vector>
#include "../Logic/PromptSingleton.hpp"
```

Classes

- class StateTags

8.99 StateTags.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "..//Logic/Database/DBmanager.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include<vector>
00016 #include "../Logic/PromptSingleton.hpp"
00017
00021 class StateTags : public State<States>{
00022     PromptSingleton* prompt = PromptSingleton::GetInstance();
00023     std::string tags;
00024 public:
00025     explicit StateTags(FiniteStateMachine<States>& fsm)
00026     : State<States>(fsm, States::TAGS, "TAGS"){}
00027
00028     void OnEnter() override;
00029     void OnUpdate() override;
00030     void OnExit() override;
00031
00032 };
00033
00034
00035 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP
```

8.100 Texts/AllTexts.hpp File Reference

```
#include <string>
```

Namespaces

- namespace IdleTexts

  namespace for Idle state
- namespace LoginTexts

  namespace for Login state
- namespace RegisterTexts

  namespace for Register state
- namespace MenuTexts

  namespace for Menu state
- namespace PromptTexts

  namespace for Prompt state
- namespace ResultTexts

  namespace for Result state
- namespace HistoryTexts

  namespace for History state
- namespace FavouriteTexts

  namespace for Favourites state
- namespace AboutTexts

  namespace for About state
- namespace TagsTexts

  namespace for Tags state
- namespace ListState

  namespace for List state
- namespace Manual

  namespace for manual

8.101 AllTexts.hpp

Go to the documentation of this file.

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
00007
00008 #include <string>
00009
00016 namespace IdleTexts
00017 {
00018     static std::string title =   " ____     __                    __      ____
    \n"
00019                            "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
    \n"
00020                            "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _
__    __     _____      __   _ __   \n"
00021                            " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
/'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\\n"
00022                            "   /\\ \\L\\ \\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\
\\__/\\ \\ \\/\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00023                            "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
\\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00024                            "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/
\\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
00025                            "
\\ \\_\\               \n"
00026                            "
\\/_/             ";
00027
00028     static std::string helloIns = "Type login/register or about";
00029 }
00030
```

```
00032 namespace LoginTexts
00033 {
00034     static std::string title =   " ____    __                    __      ____
      \n"
00035                                  "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00036                                  "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _
      __    __     _____      __   _ __  \n"
00037                                  " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
      /'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\n"
00038                                  "   /\\ \\L\\ \\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\
      \\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00039                                  "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
      \\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00040                                  "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/
      \\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
00041                                  "
      \\ \\_\\               \n"
00042                                  "
      \\/_/             ";
00043
00044     static std::string credentials = "We will ask you about credentials right now, okay?";
00045     static std::string login = "login:";
00046     static std::string password = "password:";
00047 }
00048
00050 namespace RegisterTexts
00051 {
00052     static std::string title =   " ____    __                    __      ____
      \n"
00053                                  "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00054                                  "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\
      ___   _ __    __     _____      __   _ __  \n"
00055                                  " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
      /'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\n"
00056                                  "   /\\ \\L\\ \\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\
      \\/\\ \\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00057                                  "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
      \\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00058                                  "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/
      \\/_/ \\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
00059                                  "
      \\ \\_\\               \n"
00060                                  "
      \\/_/             ";
00061
00062     static std::string credentials = "We will ask you about credentials right now, okay?";
00063     static std::string login = "login:";
00064     static std::string password = "password:";
00065     static std::string email="email:";
00066 }
00067
00069 namespace MenuTexts
00070 {
00071     static std::string title =   " ____    __                    __      ____
      \n"
00072                                  "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00073                                  "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\
      ___   _ __    __     _____      __   _ __  \n"
00074                                  " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
      /'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\n"
00075                                  "   /\\ \\L\\ \\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\
      \\/\\ \\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00076                                  "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
      \\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00077                                  "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/
      \\/_/ \\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
00078                                  "
      \\ \\_\\               \n"
00079                                  "
      \\/_/             ";
00080     static std::string helloText = "Hi! Choose question/history/tags/";
00081     static std::string favText = "favourites";
00082 }
00083
00085 namespace PromptTexts
00086 {
00087     static std::string title = " ____    __                    __      ____
      \n"
00088         "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00089         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
      __   _ __  \n"
00090         " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
      '__`\\  /'__`\\/\\`'__\\n"
```

```
00091         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
    \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00092         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
    \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00093         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
    \\/____/ \\/_/ \n"
00094         "                                                                    \\ \\_\\
    \n"
00095         "                                                                     \\/_/
    ";
00096     static std::string promptText = "Write a question or type return to go back";
00097 }
00098
00100 namespace ResultTexts
00101 {
00102     static std::string title = " ____    __                  __      ____
    \n"
00103         "/\\  _`\\ /\\ \\__               /\\ \\    /\\  _`\\
    \n"
00104         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
    __   _ __  \n"
00105         " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
    '__`\\  /'__`\\/\\`'__\\\n"
00106         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
    \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00107         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
    \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00108         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
    \\/____/ \\/_/ \n"
00109         "                                                                    \\ \\_\\
    \n"
00110         "                                                                     \\/_/
    ";
00111     static std::string answerSubtitle = "Most upvoted answered question: ";
00112     static std::string answer = "Example answered question: ";
00113     static std::string proceed = "Press enter to get back to menu";
00114     static std::string questionText = "Your question is: ";
00115     static std::string firstAnswer = "Answer 1: ";
00116 }
00117
00119 namespace HistoryTexts
00120 {
00121     static std::string title = " ____    __                  __      ____
    \n"
00122       "/\\  _`\\ /\\ \\__               /\\ \\    /\\  _`\\
    \n"
00123       "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
    __   _ __  \n"
00124       " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
    '__`\\  /'__`\\/\\`'__\\\n"
00125       "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
    \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00126       "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
    \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00127       "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
    \\/____/ \\/_/ \n"
00128       "                                                                    \\ \\_\\
    \n"
00129       "                                                                     \\/_/
    ";
00130     static std::string historyTheme = "Your recent questions:";
00131     static std::string returnText = "Type return to get back to menu or type f$ where $ = question
    index";
00132     static std::string successText = "Question added sucessfully";
00133 }
00134
00136 namespace FavouriteTexts
00137 {
00138     static std::string title = " ____    __                  __      ____
    \n"
00139         "/\\  _`\\ /\\ \\__               /\\ \\    /\\  _`\\
    \n"
00140         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
    __   _ __  \n"
00141         " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
    '__`\\  /'__`\\/\\`'__\\\n"
00142         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
    \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00143         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
    \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00144         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
    \\/____/ \\/_/ \n"
00145         "                                                                    \\ \\_\\
    \n"
00146         "                                                                     \\/_/
    ";
00147     static std::string favTheme = "Your favourite questions:";
```

```
00148     static std::string returnText = "Type return to get back to menu or type d$ where $ = question
      index";
00149     static std::string successText = "Question deleted from favourites sucessfully";
00150 }
00151
00153 namespace AboutTexts
00154 {
00155     static std::string title = " ____    __                    __      ____
      \n"
00156         "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00157         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
      __   _ __  \n"
00158         " \\/_\\__ \\\\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
      '__`\\  /'__`\\/\\`'__\\\n"
00159         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
      \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00160         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
      \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00161         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
      \\/____/ \\/_/ \n"
00162         "                                                                    \\ \\_\\
      \n"
00163         "                                                                     \\/_/
      ";
00164     static std::string aboutText = "About ";
00165     static std::string appText = "StackScraper";
00166     static std::string description = "Super CLI app to give answers about various problems!";
00167     static std::string returnText = "Type return to get back to menu";
00168 }
00169
00171 namespace TagsTexts
00172 {
00173     static std::string title = " ____    __                    __      ____
      \n"
00174         "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00175         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
      __   _ __  \n"
00176         " \\/_\\__ \\\\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
      '__`\\  /'__`\\/\\`'__\\\n"
00177         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
      \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00178         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
      \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00179         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
      \\/____/ \\/_/ \n"
00180         "                                                                    \\ \\_\\
      \n"
00181         "                                                                     \\/_/
      ";
00182     static std::string tagText = "Tags: ";
00183     static std::string returnText = "Type return to get back to menu";
00184 }
00185
00187 namespace ListState
00188 {
00189     static std::string title = " ____    __                    __      ____
      \n"
00190         "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
      \n"
00191         "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
      __   _ __  \n"
00192         " \\/_\\__ \\\\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
      '__`\\  /'__`\\/\\`'__\\\n"
00193         "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
      \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
00194         "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
      \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
00195         "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
      \\/____/ \\/_/ \n"
00196         "                                                                    \\ \\_\\
      \n"
00197         "                                                                     \\/_/
      ";
00198     static std::string tagText = "List of questions consisting of inputted tags: ";
00199     static std::string returnText = "Type return to get back to menu";
00200 }
00201
00203 namespace Manual
00204 {
00205     static std::string manual = "(01010011 01110100 01100001 01100011 01101011 01010011 01100011
      01110010 01100001 01110000 01100101 01110010)\n"
00206                                 "Manual of StackScraper \n"
00207                                 "\n"
00208                                 "Provided states and commands which can be used in them\n"
00209                                 "Idle:\n"
```

```
00210                                " - login - go to login\n"
00211                                " - register - go to register\n"
00212                                "in both of login and register app will ask about credentials.\n"
00213                                "Login approval will result in transfering to main menu\n"
00214                                "Register approval will result in transfering to login\n"
00215                                "\n"
00216                                "Main menu:\n"
00217                                " - question - ask question on stack overflos\n"
00218                                " - tags - search for questions by tags\n"
00219                                " - history - check your history\n"
00220                                " - favourites - check your favourites questions\n"
00221                                "\n"
00222                                "Question:\n"
00223                                "prompt your question to ask and go to result,\n"
00224                                "type return to go back to menu\n"
00225                                "\n"
00226                                "Tags:\n"
00227                                "prompt tags to check for questions and move to list of them,\n"
00228                                "type return to go back to menu\n"
00229                                "\n"
00230                                "List of questions:\n"
00231                                "choose question from list by prompting number,\n"
00232                                "type return to go back to menu\n"
00233                                "\n"
00234                                "History:\n"
00235                                "lists your recent questions,\n"
00236                                "type return to go back,\n"
00237                                "type number of question to move to this question,\n"
00238                                "type f$ where $ is number of question to add it to favourites\n"
00239                                "\n"
00240                                "Favourites:\n"
00241                                "lists your favourites questions,\n"
00242                                "type return to go back,\n"
00243                                "type number of question to move to this question,\n"
00244                                "type d$ where $ is number of question to delete it from favourites\n"
00245                                "\n"
00246                                "Result:\n"
00247                                "you can move to result from different states,\n"
00248                                "from question by prompting question,\n"
00249                                "from list of tags, history or favourites by choosing number from
      list\n"
00250                                "it always gives back question with max 3 top rated answers\n"
00251                                "and possibility to type return to go back to previous state\n"
00252                                "\n"
00253                                "MOST OF THE STATES HAVE AUTOCOMPLETE SO YOU CAN TYPE\n"
00254                                "R INSTEAD OF RETURN TO EXECUTE DESIRED COMMAND";
00255
00256     static std::string exit = "\nPress any key + enter to exit\n";
00257     static std::string help = "\nYou can also double check commands in the new file HELP.txt which
      just got created"
00258                               "in the exe directory\n";
00259 }
00260
00261
00262 #endif //INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
```

Index