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: ///<

Greetings from Poland

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

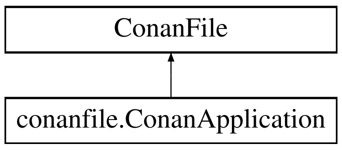
Namespace for functions which adjust printing of texts

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 Member Function Documentation

7.1.1.1 generate()

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

7.1.1.2 layout()

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

7.1.1.3 requirements()

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

7.1.2 Member Data Documentation

7.1.2.1 generators

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

7.1.2.2 package_type

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

7.1.2.3 settings

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

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

7.2.2 Constructor & Destructor Documentation

7.2.2.1 DBmanager()

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

7.2.2.2 ∼DBmanager()

```
DBmanager::∼DBmanager ()
```

7.2.3 Member Function Documentation

7.2.3.1 connectTagToPhrase()

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

7.2.3.2 deleteAdmin()

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

7.2.3.3 deleteFavourite()

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

7.2.3.4 deletePhrase()

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

7.2.3.5 deleteTag()

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

7.2.3.6 deleteUser()

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

7.2.3.7 getAdmins()

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

7.2.3.8 getFavourites()

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

7.2.3.9 getPhrase()

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

7.2.3.10 getPhrases()

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

7.2.3.11 getPhraseWithTag()

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

7.2.3.12 getTags()

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

7.2.3.13 getUsers()

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

7.2.3.14 insertAdmin()

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

7.2.3.15 insertFavourite()

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

7.2.3.16 insertPhrase()

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

7.2.3.17 insertTag()

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

7.2.3.18 insertUser()

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

7.2.3.19 loginUser()

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

7.2.3.20 updateUserPassword()

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

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

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

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

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

7.4.2 Constructor & Destructor Documentation

7.4.2.1 FiniteStateMachine()

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

Default constructor.

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

7.4.3.2 GetCurrentState() `[1/2]`

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

Function for returning current state

Returns

current state

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

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

7.4.3.5 OnUpdate()

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

Function called to change state

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	

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

7.4.4 Member Data Documentation

7.4.4.1 mCurrentState

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

pointer to current state

7.4.4.2 mStates

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

map of states

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

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

7.5.3.2 GetPrompt()

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

Function which gets new prompt to set directly from cin stream

Sets prompt from cin stream

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

7.5.3.4 RetValues()

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

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

Returns

prompt string

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

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

7.6.2 Member Function Documentation

7.6.2.1 connectTagToPhrase()

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

7.6.2.2 createAdminTable()

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

7.6.2.3 createPhraseTable()

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

7.6.2.4 createPhraseTagTable()

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

7.6.2.5 createTagTable()

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

7.6.2.6 createUserTable()

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

7.6.2.7 deleteAdmin()

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

7.6.2.8 deleteFavourite()

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

7.6.2.9 deletePhrase()

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

7.6.2.10 deleteTag()

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

7.6.2.11 deleteUser()

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

7.6.2.12 getAdmins()

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

7.6.2.13 getFavourites()

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

7.6.2.14 getPhrase()

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

7.6.2.15 getPhrases()

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

7.6.2.16 getPhrasesWithTag()

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

7.6.2.17 getTags()

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

7.6.2.18 getUsers()

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

7.6.2.19 insertAdmin()

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

7.6.2.20 insertFavourite()

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

7.6.2.21 insertPhrase()

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

7.6.2.22 insertTag()

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

7.6.2.23 insertUser()

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

7.6.2.24 loginUser()

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

7.6.2.25 updateUserPass()

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

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

7.7.2 Member Function Documentation

7.7.2.1 AskQuestion()

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

returns answers

7.7.2.2 ChangeJsonToString()

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

changes provided question/answer to string from json format

7.7.2.3 ChangingSpecialChar()

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

color special chars

7.7.2.4 checkTagQuestionList()

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

checks for list of questions based on tags

7.7.2.5 FillTabel()

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

fill table of questions for list tags state

7.7.2.6 GetAnswer()

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

gets answers

7.7.2.7 GetQuestionFromID()

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

gets quesiton from provided id

7.7.2.8 GetQuestionId()

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

returns id of question

7.7.2.9 getQuestionList()

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

returnS vector of questions

7.7.2.10 GetTitle()

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

returns title of question

7.7.2.11 LookForByTags()

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

7.7.2.12 RemoveHtmlTags()

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

removes html code

7.7.2.13 ReturnNiceCode()

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

utilize tabs and space to code format

7.7.2.14 SetQuestion()

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

sets question

7.7.2.15 SetQuestionByTags()

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

sets question with usage of tags

7.7.2.16 SetQuestionId()

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

sets question by provided id

7.7.3 Member Data Documentation

7.7.3.1 bestAnswer

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

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

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.

7.8.2.2 ∼State()

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

Virtual destructor.

7.8.3 Member Function Documentation

7.8.3.1 getID()

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

The ID of the state.

7.8.3.2 GetName()

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

The name of the state.

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.

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.

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.

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

7.8.4.2 mID

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

id of the state

7.8.4.3 mName

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

name of the state

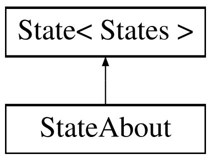
The documentation for this class was generated from the following file:

- 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

7.9.2 Constructor & Destructor Documentation

7.9.2.1 StateAbout()

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

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 >.

7.9.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.9.3.3 OnUpdate()

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

Implements virtual OnUpdate, in which provides app description

Reimplemented from State< States >.

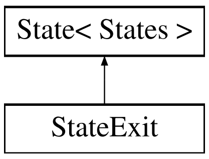
The documentation for this class was generated from the following files:

- 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

7.10.2 Constructor & Destructor Documentation

7.10.2.1 StateExit()

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

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 >.

7.10.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

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 >.

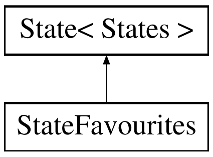
The documentation for this class was generated from the following files:

- 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

7.11.2 Constructor & Destructor Documentation

7.11.2.1 StateFavourites()

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

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 >.

7.11.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

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 >.

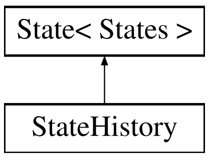
The documentation for this class was generated from the following files:

- 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

7.12.2 Constructor & Destructor Documentation

7.12.2.1 StateHistory()

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

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 >.

7.12.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.12.3.3 OnUpdate()

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

Provides information about history of searching in Prompt State

Reimplemented from State< States >.

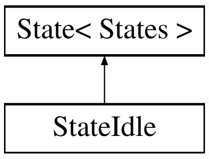
The documentation for this class was generated from the following files:

- 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

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

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 >.

7.13.3.2 OnExit()

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

overriding OnExit virtual function

Function which executes after on update

Reimplemented from State< States >.

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 >.

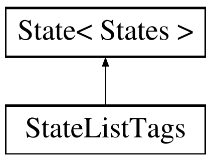
The documentation for this class was generated from the following files:

- 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

7.14.2 Constructor & Destructor Documentation

7.14.2.1 StateListTags()

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

7.14.3 Member Function Documentation

7.14.3.1 ChoosingTitle()

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

7.14.3.2 ManageList()

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

7.14.3.3 OnEnter()

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

Virtual function to describe behaviour of state on enter time

Reimplemented from State< States >.

7.14.3.4 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.14.3.5 OnUpdate()

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

Prints the list of the questions, fills the canva

Reimplemented from State< States >.

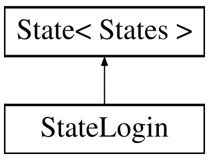
The documentation for this class was generated from the following files:

- 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

7.15.2 Constructor & Destructor Documentation

7.15.2.1 StateLogin()

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

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 >.

7.15.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.15.3.3 OnUpdate()

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

Controls credentials by refering to dbmanager instance

Reimplemented from State< States >.

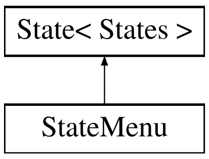
The documentation for this class was generated from the following files:

- 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

7.16.2 Constructor & Destructor Documentation

7.16.2.1 StateMenu()

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

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 >.

7.16.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.16.3.3 OnUpdate()

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

Implements transitions between main states

Reimplemented from State< States >.

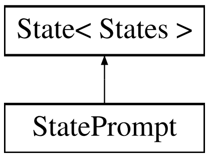
The documentation for this class was generated from the following files:

- 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

7.17.2 Constructor & Destructor Documentation

7.17.2.1 StatePrompt()

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

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 >.

7.17.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.17.3.3 OnUpdate()

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

Takes desired prompt from user

Reimplemented from State< States >.

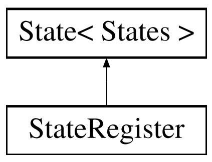
The documentation for this class was generated from the following files:

- 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

7.18.2 Constructor & Destructor Documentation

7.18.2.1 StateRegister()

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

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 >.

7.18.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.18.3.3 OnUpdate()

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

Check if credentials are valid and pass them to database

Reimplemented from State< States >.

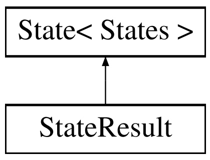
The documentation for this class was generated from the following files:

- 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

7.19.2 Constructor & Destructor Documentation

7.19.2.1 StateResult()

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

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 >.

7.19.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

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 >.

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

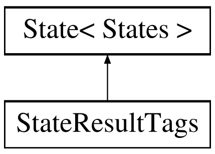
The documentation for this class was generated from the following files:

- 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

7.20.2 Constructor & Destructor Documentation

7.20.2.1 StateResultTags()

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

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 >.

7.20.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

7.20.3.3 OnUpdate()

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

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

Reimplemented from State< States >.

7.20.3.4 QuestionManage()

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

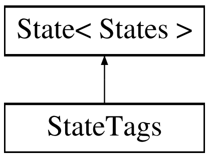
The documentation for this class was generated from the following files:

- 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

7.21.2 Constructor & Destructor Documentation

7.21.2.1 StateTags()

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

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 >.

7.21.3.2 OnExit()

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

Virtual function to describe behaviour of state on exit time

Reimplemented from State< States >.

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 >.

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

7.22.2 Constructor & Destructor Documentation

7.22.2.1 SyntaxHighlighting()

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

default constructor

7.22.3 Member Function Documentation

7.22.3.1 ColorBracket()

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

provides brackets coloring

7.22.3.2 ColorChar()

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

coloring special chars

7.22.3.3 Hightlighting()

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

highlights code

7.22.3.4 RecognizeSyntax()

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

finding `and` marks in questions and answers

7.22.3.5 RemoveTags()

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

deleting html tags

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

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

7.23.3 Member Function Documentation

7.23.3.1 GetID()

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

Returns

id of the tag

7.23.3.2 GetTitle()

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

Returns

title of the question

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 Engine.cpp File Reference

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

8.3 Engine.hpp File Reference

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

Classes

- class Engine

8.4 Engine.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP

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


class Engine {
    std::unique_ptr<FiniteStateMachine<States» fsm = nullptr;
public:
    Engine();
    void Run();
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
```

8.5 FSM/State.hpp File Reference

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

Classes

- class State< T >

8.6 State.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP

#include "StateMachine.hpp"

#include <string>
#include <utility>
template <typename T>
class FiniteStateMachine;

template <typename T>
class State
{
public:
    inline T getID()
    {
        return mID;
    }
    inline const std::string& GetName() const
    {
        return mName;
    }
    explicit State(FiniteStateMachine<T>& fsm, T id,
                   std::string name = "default")
            : mName(name)
            , mID(id)
            , mFsm(fsm)
    {
    }
    virtual ~State() {}
    virtual void OnEnter()
    {
    }
    virtual void OnExit()
    {
    }
    virtual void OnUpdate()
    {
    }
protected:
    std::string mName;
    T mID;
    FiniteStateMachine<T>& mFsm;
};

#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
```

8.7 FSM/StateMachine.hpp File Reference

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

Classes

- class FiniteStateMachine< T >

8.8 StateMachine.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP

#include "State.hpp"

#include <memory>
#include <map>
#include <string>
#include <cassert>
#include <utility>

template <typename T>
class State;

template <typename T>
class FiniteStateMachine
{
protected:
    std::map<T, std::unique_ptr<State<T>>> mStates;
    State<T>* mCurrentState;
public:
    FiniteStateMachine()
            : mCurrentState(nullptr)
    {}
    template <class S>
    State<T>& Add(T id)
    {
        static_assert(not std::is_same<State<T>, S>());
        mStates[id] = std::make_unique<S>(*this);
        return *mStates[id];
    }
    State<T>& GetState(T stateID)
    {
        return *mStates[stateID];
    }
    State<T>& GetCurrentState()
    {
        return *mCurrentState;
    }

    const State<T>& GetCurrentState() const
    {
        return *mCurrentState;
    }

    void SetCurrentState(T stateID)
    {
        State<T>* state = &GetState(stateID);
        SetCurrentState(state);
    }
    void OnUpdate()
    {
        if (mCurrentState != nullptr)
        {
            mCurrentState->OnUpdate();
        }
    }
protected:
    void SetCurrentState(State<T>* state)
    {
        if (mCurrentState == state)
        {
            return;
        }
        if (mCurrentState != nullptr)
        {
            mCurrentState->OnExit();
        }
        mCurrentState = state;
        if (mCurrentState != nullptr)
        {
            mCurrentState->OnEnter();
        }
    }
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
```

8.9 Globals.hpp File Reference

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

Namespaces

- namespace cmd

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

8.10 Globals.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 24.04.2024.
//
```

```

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP

#include <windows.h>

namespace cmd
{
    static HANDLE hOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    static HANDLE hInput = GetStdHandle(STD_INPUT_HANDLE);
}

#endif //INC_2024__TAB_DSA__8_BRODZIAK_GLOBALS_HPP
```

8.11 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

8.11.1 Typedef Documentation

8.11.1.1 sqlite3_callback

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

8.11.2 Variable Documentation

8.11.2.1 receivedData

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

8.12 Logic/Database/DBmanager.hpp File Reference

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

Classes

- class DBmanager

8.13 DBmanager.hpp

Go to the documentation of this file.

```
//
// Created by lucja on 11.05.2024.
//

#ifndef DBMANAGER_HPP
#define DBMANAGER_HPP

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

class DBmanager {
    static std::string nickName;
    static int id;

    sqlite3 *db;
    char *zErrMsg;
    int rc;
    const char* data = "Callback function called";

    int openDatabase();
    int createDatabase();
    int closeDatabase();

    int createUserTable();
    int createAdminTable();
    int createPhraseTable();
    int createTagTable();
    int createPhraseTagTable();
public:
    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();
};



#endif //DBMANAGER_HPP
```

8.14 Logic/Database/QueryHelper.cpp File Reference

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

8.15 Logic/Database/QueryHelper.hpp File Reference

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

Classes

- class QueryHelper

8.16 QueryHelper.hpp

Go to the documentation of this file.

```
//
// Created by lucja on 21.05.2024.
//

#ifndef QUERYHELPER_HPP
#define QUERYHELPER_HPP

#include<string>
#include <format>

class QueryHelper {
public:
     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();
};


#endif //QUERYHELPER_HPP
```

8.17 Logic/PromptSingleton.cpp File Reference

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

Functions

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

8.17.1 Function Documentation

8.17.1.1 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

8.18 Logic/PromptSingleton.hpp File Reference

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

Classes

- class PromptSingleton

8.19 PromptSingleton.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 23.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP

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

class PromptSingleton{
private:
    std::string prompt;
    static PromptSingleton* instancePtr;
    PromptSingleton()= default;
public:
    PromptSingleton(const PromptSingleton& obj)
    = delete;
    static PromptSingleton* GetInstance();
    void SetValues(std::string& val);
    std::string RetValues(){ return prompt; }
    void GetPrompt();
    void GetPromptAuto(std::vector<std::string> dict);
};





#endif //INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
```

8.20 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"
```

8.21 Logic/StackApi/StackManager.hpp File Reference

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

Classes

- class StackManager

8.22 StackManager.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 10.05.2024.
//

#ifndef STACKMANAGER_HPP
#define STACKMANAGER_HPP

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

class StackManager {
    //URL TO SEARCH
    //https://api.stackexchange.com/2.3/search?order=desc&sort=activity&intitle=CPP&site=stackoverflow

    //value which store answear id "accepted_answer_id": 63548573,
    //URL TO FIND ANSWEAR
//https://api.stackexchange.com/2.3/answers/63548573?order=desc&sort=activity&site=stackoverflow&filter=withbody

    std::string space = "%20";
    std::string baseInput = "https://api.stackexchange.com/";
    std::string apiVesion = "2.3/";
    std::string questionInput = "";
    std::string finalInput = "";
    std::string answerInput = "";
    std::string stringQuestionID = "";
    int questionID = 0;
    std::string title;
    static std::vector<TagsList> questionsList;



public:


    std::string bestAnswer[3] = {"","",""};
    void AskQuestion(std::string & question);
    void SetQuestion(std::string newInput);
    void SetQuestionByTags(std::string newInput);
    void GetAnswer(std::string res);
    void ChangeJsonToString(std::string&);
    void SetQuestionId(std::string);
    void FillTabel(std::string input);
    void RemoveHtmlTags(std::string& input);
    void ReturnNiceCode(std::string& input);
    void ChangingSpecialChar(std::string &input,std::string inChar, std::string outChar);
    void LookForByTags(std::string& input);
    void checkTagQuestionList(std::string& tagInput);
    static std::vector<TagsList> getQuestionList();
    std::string GetTitle();
    std::string GetQuestionId();
    void GetQuestionFromID(std::string id);
};



#endif //STACKMANAGER_HPP
```

8.23 Logic/StackApi/Syntax.hpp File Reference

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

Namespaces

- namespace Syntax

8.24 Syntax.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 23.05.2024.
//
#include <vector>
#include <string>

#ifndef SYNTAX_HPP
#define SYNTAX_HPP

namespace Syntax {
 static std::vector<std::string> basicSyntax = {
  "for", "while", "do",
  "if", "else", "int",
  "string", "::", "std",
  "double", "float", "bool",
  "main", "switch", "case",
  "char", "cin", "getline",
  "cout", "return", "long",
  "short", "cerr", "«",
  "»", "include", "using",
  "NULL", "nullptr", "class",
  "void", "private", "public",
  "*", "&", "\"",
  "=", "const", "static",
  "delete", "new", "break",
  "continue", "protected", "enum",
  "typedef", "try",
  "catch", "throw", "template",
  "operator", "this", "friend",
  "volatile", "extern", "struct",
  "sizeof", "finally", "AND",
  "OR", "&&", "||",
  "false", "true",
  //PYTHONE
  "False", "None", "True",
  "and", "as", "assert",
  "def", "del", "await",
  "async", "elif", "except",
  "global", "from", "import",
  "in", "is", "lambda",
  "not", "!", "raise",
  "with", "pass", "yield",
  //C KEYWORD
  "auto", "default", "inline",
  "signed", "malloc", "printf",
  "free",
  //JAVA KEYWORD
  "abstract", "boolen", "implements",
  "interface", "native", "package",
  "super",
  //PHP KEYWORD
  "array", "clone", "declare",
  "echo", "elseif", "foreach",
  "empty", "endfor", "endif",
  "endforeach", "endswitch", "isset",
  "unset", "var", "use",
  "xor",
  //JS KEYWORD
  "let", "function", "export",
  //HTML TAGS
  "div", "<", ">",
  "area", "blockquote", "body",
  "html", "head", "button",
  "dl", "dt", "h1",
  "h2", "h3", "h4",
  "h5", "h6", "nav",
  "script", "strong", "style",
  "td", "table", "sup",
  "ul", "ol", "li",
  "p", "b", "s",
  "i", "br", "td",
  "a", "img", "tr",
  //others
  "print", "namespace", "__name__",
  "__main__", "__init__",
  //css
  "display", "position", "top",
  "float", "clear", "both",
  "width", "height", "min-height",
  "min-width", "margin", "padding",
  "color", "font", "text-align",
  "text-decoration", "letter-spacing", "border",
  "transform", "transition", "flex",
  "flex-align", "flex-directory", "flex-wrap",
  "justift-content", "grid", "grid-template-columns",
  "grid-templeta-rows", "cursor", "pointer",
  ":hover", ":focus", "visted",
  "margin-left", "margin-right", "margin-top",
  "margin-bottom", "left", "right",
  "bottom", "overflow", "hidden",
  "background-color", "background", "opactity",
  "absolute", "fixed", "style",
  "span", "input", "placeholder",
  "#ifndef", "define", "regex",
    "println"

 };
 static std::vector<std::string> keyWord = {
  //CPP KEYWORD
  "\033[0;32mfor\033[0m", "\033[0;32mwhile\033[0m", "\033[0;32mdo\033[0m",
  "\033[0;34mif\033[0m", "\033[0;34melse\033[0m", "\033[0;33mint\033[0m",
  "\033[0;33mstring\033[0m", "\033[0;31m::\033[0m", "\033[0;35mstd\033[0m",
  "\033[0;33mdouble\033[0m", "\033[0;33mfloat\033[0m", "\033[0;33mbool\033[0m",
  "\033[0;34mmain\033[0m", "\033[0;36mswitch\033[0m", "\033[0;33mcase\033[0m",
  "\033[0;33mchar\033[0m", "\033[0;31mcin\033[0m", "\033[0;31mgetline\033[0m",
  "\033[0;31mcout\033[0m", "\033[0;31mreturn\033[0m", "\033[0;33mlong\033[0m",
  "\033[0;33mshort\033[0m", "\033[0;31mcerr\033[0m", "\033[0;32m«\033[0m",
  "\033[0;32m»\033[0m", "\033[0;33minclude\033[0m", "\033[0;32musing\033[0m",
  "\033[0;32mNULL\033[0m", "\033[0;32mnullptr\033[0m", "\033[0;33mclass\033[0m",
  "\033[0;33mvoid\033[0m", "\033[0;31mprivate\033[0m", "\033[0;32mpublic\033[0m",
  "\033[0;34m*\033[0m", "\033[0;34m&\033[0m", "\033[0;32m\"\033[0m",
  "\033[0;33m=\033[0m", "\033[0;35mconst\033[0m", "\033[0;35mstatic\033[0m",
  "\033[0;31mdelete\033[0m", "\033[0;36mnew\033[0m", "\033[0;31mbreak\033[0m",
  "\033[0;33mcontinue\033[0m", "\033[0;33mprotected\033[0m", "\033[0;33menum\033[0m",
  "\033[0;32mtypedef\033[0m", "\033[0;36mtry\033[0m",
  "\033[0;36mcatch\033[0m", "\033[0;31mthrow\033[0m", "\033[0;34mtemplate\033[0m",
  "\033[0;33moperator\033[0m", "\033[0;32mthis\033[0m", "\033[0;35mfriend\033[0m",
  "\033[0;33mvolatile\033[0m", "\033[0;33mextern\033[0m", "\033[0;33mstruct\033[0m",
  "\033[0;33msizeof\033[0m", "\033[0;33mfinally\033[0m", "\033[0;32mAND\033[0m",
  "\033[0;32mOR\033[0m", "\033[0;32m&&\033[0m", "\033[0;32m||\033[0m",
  "\033[0;32mfalse\033[0m", "\033[0;32mtrue\033[0m",
  //PYTHONE KEYWORD
  "\033[0;31mfalse\033[0m", "\033[0;33mNone\033[0m", "\033[0;32mtrue\033[0m",
  "\033[0;32mand\033[0m", "\033[0;33mAs\033[0m", "\033[0;33mAssert\033[0m",
  "\033[0;33mdef\033[0m", "\033[0;31mdel\033[0m", "\033[0;35mawit\033[0m",
  "\033[0;35masync\033[0m", "\033[0;34melif\033[0m", "\033[0;31mexcept\033[0m",
  "\033[0;36mglobal\033[0m", "\033[0;35mfrom\033[0m", "\033[0;35mimport\033[0m",
  "\033[0;35min\033[0m", "\033[0;35mis\033[0m", "\033[0;36mlambdal\033[0m",
  "\033[0;31mnot\033[0m", "\033[0;31m!\033[0m", "\033[0;36mraise\033[0m",
  "\033[0;36mwith\033[0m", "\033[0;35mpass\033[0m", "\033[0;36myield\033[0m",
  //C KEYWORD
  "\033[0;33mauto\033[0m", "\033[0;34mdefault\033[0m", "\033[0;34minline\033[0m",
  "\033[0;33msigned\033[0m", "\033[0;31mmalloc\033[0m", "\033[0;31mprintf\033[0m",
  "\033[0;32mfree\033[0m",
  //JAVA KEYWORD
  "\033[0;33mabstract\033[0m", "\033[0;33mboolen\033[0m", "\033[0;36mimplements\033[0m",
  "\033[0;33mnative\033[0m", "\033[0;35mnative\033[0m", "\033[0;35mpackage\033[0m",
  "\033[0;32msuper\033[0m",
  //PHP KEYWORD
  "\033[0;33marray\033[0m", "\033[0;35mclone\033[0m", "\033[0;35mdeclare\033[0m",
  "\033[0;32mecho\033[0m", "\033[0;34melseif\033[0m", "\033[0;32mforeach\033[0m",
  "\033[0;36mempty\033[0m", "\033[0;32mendfor\033[0m", "\033[0;34mendif\033[0m",
  "\033[0;32mendforeach\033[0m", "\033[0;36mendswitch\033[0m", "\033[0;33misset\033[0m",
  "\033[0;33munset\033[0m", "\033[0;36mvar\033[0m", "\033[0;31muse\033[0m",
  "\033[0;33mxor\033[0m",
  //JS KEYWORD
  "\033[0;36mlet\033[0m", "\033[0;32mfunction\033[0m", "\033[0;32mexport\033[0m",
  //HTML TAGS
  "\033[0;36mdiv\033[0m", "\033[0;32m<\033[0m", "\033[0;32m>\033[0m",
  "\033[0;35marea\033[0m", "\033[0;36mblockquote\033[0m", "\033[0;31mbody\033[0m",
  "\033[0;31mhtml\033[0m", "\033[0;31mhead\033[0m", "\033[0;32mbutton\033[0m",
  "\033[0;32mdl\033[0m", "\033[0;32mdt\033[0m", "\033[0;34mh1\033[0m",
  "\033[0;34mh2\033[0m", "\033[0;34mh3\033[0m", "\033[0;34mh4\033[0m",
  "\033[0;34mh5\033[0m", "\033[0;34mh6\033[0m", "\033[0;32mnav\033[0m",
  "\033[0;33mscript\033[0m", "\033[0;31mstrong\033[0m", "\033[0;32mstyle\033[0m",
  "\033[0;32mtd\033[0m", "\033[0;32mtable\033[0m", "\033[0;32msup\033[0m",
  "\033[0;33mul\033[0m", "\033[0;33mol\033[0m", "\033[0;33mli\033[0m",
  "\033[0;33mp\033[0m", "\033[0;31mb\033[0m", "\033[0;33ms\033[0m",
  "\033[0;35mi\033[0m", "\033[0;35mbr\033[0m", "\033[0;31mtd\033[0m",
  "\033[0;34ma\033[0m", "\033[0;32mimg\033[0m", "\033[0;31mtr\033[0m",
  //others
  "\033[0;31mprint\033[0m", "\033[0;32mnamespace\033[0m", "\033[0;35m__name__\033[0m",
  "\033[0;35m__main__\033[0m", "\033[0;35m__init__\033[0m",
  //css
  "\033[0;31mdisplay\033[0m", "\033[0;31mposition\033[0m", "\033[0;31mtop\033[0m",
  "\033[0;31mfloat\033[0m", "\033[0;31mfloat\033[0m", "\033[0;32mdboth\033[0m",
  "\033[0;31mwidth\033[0m", "\033[0;31mheight\033[0m", "\033[0;31mmin-height\033[0m",
  "\033[0;31mmin-width\033[0m", "\033[0;31mmargin\033[0m", "\033[0;31mpadding\033[0m",
  "\033[0;31mcolor\033[0m", "\033[0;31mfont\033[0m", "\033[0;31mtext-align\033[0m",
  "\033[0;31mtext-decoration\033[0m", "\033[0;31mletter-spacing\033[0m", "\033[0;31mborder\033[0m",
  "\033[0;31mtransform\033[0m", "\033[0;31mtransition\033[0m", "\033[0;31mflex\033[0m",
  "\033[0;31mflex-align\033[0m", "\033[0;31mflex-directory\033[0m", "\033[0;31mflex-wrap\033[0m",
  "\033[0;31mjustify-content\033[0m", "\033[0;31mgrid\033[0m",
"\033[0;31mgrid-template-columns\033[0m",
  "\033[0;31mgrid-template-rows\033[0m", "\033[0;31mcursor\033[0m", "\033[0;32mpointer\033[0m",
  "\033[0;35mhover\033[0m", "\033[0;35mfocus\033[0m", "\033[0;35mvistied\033[0m",
  "\033[0;31mmargin-left\033[0m", "\033[0;31mmargin-right\033[0m", "\033[0;31mmargin-top\033[0m",
  "\033[0;31mmargin-bottom\033[0m", "\033[0;31mleft\033[0m", "\033[0;31mright\033[0m",
  "\033[0;31mbottom\033[0m", "\033[0;31moverflow\033[0m", "\033[0;32mhidden\033[0m",
  "\033[0;31mbackground-color\033[0m", "\033[0;31mbackground\033[0m", "\033[0;31mopacity\033[0m",
  "\033[0;32mabsolute\033[0m", "\033[0;32mfixed\033[0m", "\033[0;32mstyle\033[0m",
  "\033[0;31mspan\033[0m", "\033[0;33minput\033[0m", "\033[0;32mplaceholder\033[0m",
  "\033[0;35mifndef\033[0m", "\033[0;34mdefine\033[0m", "\033[0;32mregex\033[0m",
  "\033[0;36mprintln\033[0m"
 };
 static std::vector<std::string> specialCharacter = {
  "<", ">", "\"",
  "\'", "*", "&",
  "|", "$", ":",
  "->", "#", "%"
 };
 static std::vector<std::string> colorSpecialCharacter = {
  "\033[0;32m<\033[0m", "\033[0;32m>\033[0m", "\033[0;32m\"\033[0m",
  "\033[0;31m\'\033[0m", "\033[0;34m*\033[0m", "\033[0;34m&\033[0m",
  "\033[0;33m|\033[0m", "\033[0;34m$\033[0m", "\033[0;32m:\033[0m",
  "\033[0;32m->\033[0m", "\033[0;33m#\033[0m", "\033[0;33m%\033[0m"
 };
}
#endif //SYNTAX_HPP
```

8.25 Logic/StackApi/SyntaxHighlighting.cpp File Reference

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

8.26 Logic/StackApi/SyntaxHighlighting.hpp File Reference

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

Classes

- class SyntaxHighlighting

8.27 SyntaxHighlighting.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 20.05.2024.
//

#ifndef SYNTAXHIGHLIGHTING_HPP
#define SYNTAXHIGHLIGHTING_HPP

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

class SyntaxHighlighting {
        std::vector<std::regex> regexes;
public:
        SyntaxHighlighting();
        void RecognizeSyntax(std::string& in);
        std::string Hightlighting(std::string &in);
        void RemoveTags(std::string &input,std::string tag, std::string out, int pos);
        void ColorChar(std::string &input,std::string tag, std::string out);
        void ColorBracket(std::string &in);
};



#endif //SYNTAXHIGHLIGHTING_HPP
```

8.28 Logic/TagList/TagsList.cpp File Reference

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

8.29 Logic/TagList/TagsList.hpp File Reference

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

Classes

- class TagsList

8.30 TagsList.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 29.05.2024.
//

#ifndef TAGSLIST_HPP
#define TAGSLIST_HPP

#include <iostream>
#include <string>

class TagsList {
    int id;
    std::string title;
public:
    int GetID();
    std::string GetTitle();
    TagsList(int _id, std::string& _title);
};



#endif //TAGSLIST_HPP
```

8.31 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

8.32 TextFormatter.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 24.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP

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

namespace TextColors
{
    static int BLUE = 1;
    static int GREEN = 2;
    static int LIGHTBLUE = 3;
    static int RED = 4;
    static int PURPLE = 5;
    static int YELLOW = 6;
    static int WHITE = 7;
    static int GREY = 8;
    static int BLUEBERRY = 9;
    static int LIGHTGREEN = 10;
    static int CYAN = 11;
    static int ROSE = 12;
    static int PINK = 13;
    static int BEIGE = 14;
}

namespace TextFunctions{

    static void changeTextColor(int color)
    {
        SetConsoleTextAttribute(cmd::hOutput, color);
    }

    static void typeWriteMessage(std::string& s, int time)
    {
        for (const auto c : s) {
            std::cout « c « std::flush;
            std::this_thread::sleep_for(std::chrono::milliseconds(time));
        }
        printf("\n");
    }

    static void print(std::string& message)
    {
        std::cout«message«std::endl;
    }

    static bool setCursor(short x, short y)
    {
        return SetConsoleCursorPosition(cmd::hOutput, {x, y});
    }
    static COORD GetConsoleCursorPosition(HANDLE hConsoleOutput)
    {
        CONSOLE_SCREEN_BUFFER_INFO cbsi;
        if (GetConsoleScreenBufferInfo(hConsoleOutput, &cbsi))
        {
            return cbsi.dwCursorPosition;
        }
        else
        {
            // The function failed. Call GetLastError() for details.
            COORD invalid = { 0, 0 };
            return invalid;
        }
    }
    static std::string toLower(std::string data) {

         std::transform(data.begin(), data.end(), data.begin(),
    [](unsigned char c){ return std::tolower(c); });

        return data;
    }


}



#endif //INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
```

8.33 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[ ])

8.33.1 Function Documentation

8.33.1.1 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

8.33.1.2 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

8.34 README.md File Reference

8.35 States/StateAbout.cpp File Reference

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

8.36 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

8.37 StateAbout.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 01.05.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP


#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"

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

class StateAbout : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
            "return"
    };
public:
    explicit StateAbout(FiniteStateMachine<States>& fsm)
            : State<States>(fsm, States::ABOUT, "ABOUT"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
```

8.38 States/StateExit.cpp File Reference

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

8.39 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

8.40 StateExit.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"

#include <iostream>
#include <string>

class StateExit : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
public:
    explicit StateExit(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::EXIT, "EXIT"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
```

8.41 States/StateFavourites.cpp File Reference

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

8.42 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

8.43 StateFavourites.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "..//Logic/Database/DBmanager.hpp"

#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"

class StateFavourites : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
            "return"
    };

    DBmanager db;
    std::vector<int> indexes;
    std::vector<std::pair<std::string,std::string» data;
    std::vector<std::string> trimmedData;

    void ManageData();
    int CheckFav(std::string);
public:
    explicit StateFavourites(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::FAVOURITES, "FAVOURITES"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
```

8.44 States/StateHistory.cpp File Reference

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

8.45 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

8.46 StateHistory.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "..//Logic/Database/DBmanager.hpp"

#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"

class StateHistory : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
            "return"
    };
    DBmanager db;
    std::vector<std::string> trimmedData;

    void ManageData();
    int CheckFav(std::string);
public:
    explicit StateHistory(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::HISTORY, "HISTORY"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
```

8.47 States/StateIdle.cpp File Reference

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

8.48 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

8.49 StateIdle.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP

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


class StateIdle : public State<States>{
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
            "login",
            "register",
            "about"
    };
public:
    explicit StateIdle(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::IDLE, "IDLE"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP

```

8.50 States/StateListTags.cpp File Reference

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

8.51 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

8.52 StateListTags.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 28.05.2024.
//

#ifndef STATELISTTAGS_HPP
#define STATELISTTAGS_HPP
#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"



class StateListTags: public State<States>  {
    std::string question;
    std::vector<TagsList> questionsList;
    StackManager sm = StackManager();
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
        "return"
};

public:
    explicit StateListTags(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::LISTTAGS, "LISTTAGS"){}
    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
    void ManageList();
    bool ChoosingTitle(std::string in);
};



#endif //STATELISTTAGS_HPP
```

8.53 States/StateLogin.cpp File Reference

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

8.54 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

8.55 StateLogin.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP

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

class StateLogin : public State<States>{
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::string log;
    std::string pass;
public:
    explicit StateLogin(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::LOGIN, "LOGIN"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATELOGIN_HPP
```

8.56 States/StateMenu.cpp File Reference

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

8.57 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

8.58 StateMenu.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"
#include "../Logic/PromptSingleton.hpp"

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

class StateMenu : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::vector<std::string> dict = {
            "question",
            "history",
            "tags",
            "favourites"
    };

public:
    explicit StateMenu(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::MENU, "MENU"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
```

8.59 States/StatePrompt.cpp File Reference

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

8.60 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

8.61 StatePrompt.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"

#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"
#include "../Logic/StackApi/StackManager.hpp"

class StatePrompt : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    StackManager sm = StackManager();
    std::vector<std::string> dict = {
            "return"
    };
public:
    explicit StatePrompt(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::PROMPT, "PROMPT"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEPROMPT_HPP
```

8.62 States/StateRegister.cpp File Reference

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

8.63 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

8.64 StateRegister.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP

#include "StatesConf.hpp"
#include "../FSM/StateMachine.hpp"
#include "../FSM/State.hpp"

#include <iostream>
#include <string>
#include "../Logic/PromptSingleton.hpp"

class StateRegister : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::string log;
    std::string pass;
    std::string email;
public:
    explicit StateRegister(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::REGISTER, "REGISTER"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEREGISTER_HPP
```

8.65 States/StateResult.cpp File Reference

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

8.66 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

8.67 StateResult.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 24.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP


#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"

class StateResult : public State<States> {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::string question;
    std::string answer;
    StackManager sm = StackManager();
    SyntaxHighlighting sh = SyntaxHighlighting();
    std::vector<std::string> dict = {
            "question",
            "return"
    };
public:
    explicit StateResult(FiniteStateMachine<States>& fsm)
            : State<States>(fsm, States::RESULT, "RESULT"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;

    void QuestionManage();
};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
```

8.68 States/StateResultTags.cpp File Reference

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

8.69 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

8.70 StateResultTags.hpp

Go to the documentation of this file.

```
//
// Created by jakub on 28.05.2024.
//

#ifndef STATERESULTTAGS_HPP
#define STATERESULTTAGS_HPP
#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"

class StateResultTags : public State<States>  {
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::string question;
    std::string answer;
    StackManager sm = StackManager();
    SyntaxHighlighting sh = SyntaxHighlighting();
    std::vector<std::string> dict = {
        "tags",
        "return"
};
public:
    explicit StateResultTags(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::RESULTTAGS, "RESULTTAGS"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;

    void QuestionManage();

};



#endif //STATERESULTTAGS_HPP
```

8.71 States/StatesConf.hpp File Reference

Enumerations

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

8.71.1 Enumeration Type Documentation

8.71.1.1 States

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

File which provides Enumeration of all States possible

Enumerator

IDLE	
LOGIN	
REGISTER	
MENU	
PROMPT	
FAVOURITES	
TAGS	
HISTORY	
EXIT	
RESULT	
ABOUT	
RESULTTAGS	
LISTTAGS	

8.72 StatesConf.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 21.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP

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

#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATESCONF_HPP
```

8.73 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"
```

8.74 StatesWrapper.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 23.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP

#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"

#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATESWRAPPER_HPP
```

8.75 States/StateTags.cpp File Reference

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

8.76 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

8.77 StateTags.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 23.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP

#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"

class StateTags : public State<States>{
    PromptSingleton* prompt = PromptSingleton::GetInstance();
    std::string tags;
public:
    explicit StateTags(FiniteStateMachine<States>& fsm)
    : State<States>(fsm, States::TAGS, "TAGS"){}

    void OnEnter() override;
    void OnUpdate() override;
    void OnExit() override;

};


#endif //INC_2024__TAB_DSA__8_BRODZIAK_STATETAGS_HPP
```

8.78 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

8.79 AllTexts.hpp

Go to the documentation of this file.

```
//
// Created by Michin on 23.04.2024.
//

#ifndef INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
#define INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP

#include <string>

namespace IdleTexts
{
    static std::string title =   " ____    __                   __      ____
\n"
                          "/\\  _`\\ /\\ \\__               /\\ \\    /\\  _`\\
\n"
                          "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _
__   __    _____     __   _ __  \n"
                          " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
/'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\\n"
                          "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\
\\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
                          "    \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
\\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
                          "     \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/
\\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
                          "
\\ \\_\\              \n"
                          "
\\/_/             ";

    static std::string helloIns = "Type login/register or about";
}

namespace LoginTexts
{
    static std::string title =   " ____    __                   __      ____
\n"
                          "/\\  _`\\ /\\ \\__               /\\ \\    /\\  _`\\
\n"
                          "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _
__   __    _____     __   _ __  \n"
                          " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
/'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\\n"
                          "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\
\\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
                          "    \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
\\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
                          "     \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/
\\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
                          "
\\ \\_\\              \n"
                          "
\\/_/             ";

    static std::string credentials = "We will ask you about credentials right now, okay?";
    static std::string login = "login:";
    static std::string password = "password:";
}

```

```
namespace RegisterTexts
{
    static std::string title =   " ____    __                    __      ____
\n"
                                 "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
                                 "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\
___   _ __    __     _____      __    _ __   \n"
                                 " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
/'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\\n"
                                 "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\
\\/\\ \\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
                                 "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
\\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
                                 "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/
\\/_/ \\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
                                 "
\\ \\_\\               \n"
                                 "
\\/_/              ";

    static std::string credentials = "We will ask you about credentials right now, okay?";
    static std::string login = "login:";
    static std::string password = "password:";
    static std::string email="email:";
}

namespace MenuTexts
{
    static std::string title =   " ____    __                    __      ____
\n"
                                 "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
                                 "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\
___   _ __    __     _____      __    _ __   \n"
                                 " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\
/'___\\/\\`'__\\/'__`\\  /\\ '__`\\  /'__`\\/\\`'__\\\n"
                                 "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\
\\/\\ \\__/\\ \\ \\//\\ \\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
                                 "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\
\\____\\\\ \\_\\\\ \\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
                                 "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/
\\/_/ \\/__/\\/_/ \\ \\ \\/  \\/____/ \\/_/ \n"
                                 "
\\ \\_\\               \n"
                                 "
\\/_/              ";
    static std::string helloText = "Hi! Choose question/history/tags/";
    static std::string favText = "favourites";
}

namespace PromptTexts
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
__   _ __   \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                  \\ \\_\\
\n"
        "                                                                   \\/_/
";
    static std::string promptText = "Write a question or type return to go back";
}

namespace ResultTexts
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
__   _ __   \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                           \\ \\_\\
\n"
        "                                                                            \\/_/
";
    static std::string answerSubtitle = "Most upvoted answered question: ";
    static std::string answer = "Example answered question: ";
    static std::string proceed = "Press enter to get back to menu";
    static std::string questionText = "Your question is: ";
    static std::string firstAnswer = "Answer 1: ";
}

namespace HistoryTexts
{
    static std::string title = " ____    __                    __      ____
\n"
      "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
      "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
__   _ __  \n"
      " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
      "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
      "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
      "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
      "                                                                           \\ \\_\\
\n"
      "                                                                            \\/_/
";
    static std::string historyTheme = "Your recent questions:";
    static std::string returnText = "Type return to get back to menu or type f$ where $ = question
index";
    static std::string successText = "Question added sucessfully";
}

namespace FavouriteTexts
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
__   _ __  \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                           \\ \\_\\
\n"
        "                                                                            \\/_/
";
    static std::string favTheme = "Your favourite questions:";
    static std::string returnText = "Type return to get back to menu or type d$ where $ = question
index";
    static std::string successText = "Question deleted from favourites sucessfully";
}

namespace AboutTexts
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     _____
__   _ __  \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                           \\ \\_\\
\n"
        "                                                                            \\/_/
";
    static std::string aboutText = "About ";
    static std::string appText = "StackScraper";
    static std::string description = "Super CLI app to give answers about various problems!";
    static std::string returnText = "Type return to get back to menu";
}

namespace TagsTexts
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     ______
__   _ __  \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                          \\ \\_\\
\n"
        "                                                                           \\/_/
";
    static std::string tagText = "Tags: ";
    static std::string returnText = "Type return to get back to menu";
}

namespace ListState
{
    static std::string title = " ____    __                    __      ____
\n"
        "/\\  _`\\ /\\ \\__                /\\ \\    /\\  _`\\
\n"
        "\\ \\,\\L\\_\\ \\ ,_\\    __      ___\\ \\ \\/'\\\\ \\,\\L\\_\\    ___   _ __    __     ______
__   _ __  \n"
        " \\/_\\__ \\\\ \\ \\/  /'__`\\   /'___\\ \\ , < \\/_\\__ \\   /'___\\/\\`'__\\/'__`\\  /\\
'__`\\  /'__`\\/\\`'__\\\n"
        "   /\\ \\L\\ \\ \\ \\_/\\ \\L\\.\\_/\\ \\__/\\ \\ \\\\`\\ /\\ \\L\\ \\/\\ \\__/\\ \\ \\//\\
\\L\\.\\_\\ \\ \\L\\ \\/\\  __/\\ \\ \\/ \n"
        "   \\ `\\____\\ \\__\\ \\__/.\\_\\ \\____\\\\ \\_\\ \\_\\ `\\____\\ \\____\\\\ \\_\\\\
\\__/.\\_\\\\ \\ ,__/\\ \\____\\\\ \\_\\ \n"
        "    \\/_____/\\/__/\\/__/\\/_/\\/____/ \\/_/\\/_/\\/_____/\\/____/ \\/_/ \\/__/\\/_/ \\ \\ \\/
\\/____/ \\/_/ \n"
        "                                                                          \\ \\_\\
\n"
        "                                                                           \\/_/
";
    static std::string tagText = "List of questions consisting of inputted tags: ";
    static std::string returnText = "Type return to get back to menu";
}

namespace Manual
{
    static std::string manual = "(01010011 01110100 01100001 01100011 01101011 01010011 01100011
01110010 01100001 01110000 01100101 01110010)\n"
                                "Manual of StackScraper \n"
                                "\n"
                                "Provided states and commands which can be used in them\n"
                                "Idle:\n"
                                " - login - go to login\n"
                                " - register - go to register\n"
                                "in both of login and register app will ask about credentials.\n"
                                "Login approval will result in transfering to main menu\n"
                                "Register approval will result in transfering to login\n"
                                "\n"
                                "Main menu:\n"
                                " - question - ask question on stack overflos\n"
                                " - tags - search for questions by tags\n"
                                " - history - check your history\n"
                                " - favourites - check your favourites questions\n"
                                "\n"
                                "Question:\n"
                                "prompt your question to ask and go to result,\n"
                                "type return to go back to menu\n"
                                "\n"
                                "Tags:\n"
                                "prompt tags to check for questions and move to list of them,\n"
                                "type return to go back to menu\n"
                                "\n"
                                "List of questions:\n"
                                "choose question from list by prompting number,\n"
                                "type return to go back to menu\n"
                                "\n"
                                "History:\n"
                                "lists your recent questions,\n"
                                "type return to go back,\n"
                                "type number of question to move to this question,\n"
                                "type f$ where $ is number of question to add it to favourites\n"
                                "\n"
                                "Favourites:\n"
                                "lists your favourites questions,\n"
                                "type return to go back,\n"
                                "type number of question to move to this question,\n"
                                "type d$ where $ is number of question to delete it from favourites\n"
                                "\n"
                                "Result:\n"
                                "you can move to result from different states,\n"
                                "from question by prompting question,\n"
                                "from list of tags, history or favourites by choosing number from list\n"
                                "it always gives back question with max 3 top rated answers\n"
                                "and possibility to type return to go back to previous state\n"
                                "\n"
                                "MOST OF THE STATES HAVE AUTOCOMPLETE SO YOU CAN TYPE\n"
                                "R INSTEAD OF RETURN TO EXECUTE DESIRED COMMAND";

    static std::string exit = "\nPress any key + enter to exit\n";
    static std::string help = "\nYou can also double check commands in the new file HELP.txt which just got created"
                              "in the exe directory\n";
}


#endif //INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
```

Index