

StackScraper

v1.0.0

Generated by Doxygen 1.11.0

1 README	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 AboutTexts Namespace Reference	11
6.1.1 Detailed Description	11
6.2 cmd Namespace Reference	11
6.2.1 Detailed Description	11
6.3 conanfile Namespace Reference	11
6.4 FavouriteTexts Namespace Reference	11
6.4.1 Detailed Description	12
6.5 HistoryTexts Namespace Reference	12
6.5.1 Detailed Description	12
6.6 IdleTexts Namespace Reference	12
6.6.1 Detailed Description	12
6.7 ListState Namespace Reference	12
6.7.1 Detailed Description	12
6.8 LoginTexts Namespace Reference	12
6.8.1 Detailed Description	12
6.9 Manual Namespace Reference	13
6.9.1 Detailed Description	13
6.10 MenuTexts Namespace Reference	13
6.10.1 Detailed Description	13
6.11 PromptTexts Namespace Reference	13
6.11.1 Detailed Description	13
6.12 RegisterTexts Namespace Reference	13
6.12.1 Detailed Description	13
6.13 ResultTexts Namespace Reference	13
6.13.1 Detailed Description	14
6.14 Syntax Namespace Reference	14
6.14.1 Detailed Description	14
6.15 TagsTexts Namespace Reference	14
6.15.1 Detailed Description	14

6.16 TextColors Namespace Reference	14
6.16.1 Detailed Description	14
6.17 TextFunctions Namespace Reference	14
6.17.1 Detailed Description	14
7 Class Documentation	15
7.1 conanfile.ConanApplication Class Reference	15
7.1.1 Member Function Documentation	15
7.1.1.1 generate()	15
7.1.1.2 layout()	15
7.1.1.3 requirements()	16
7.1.2 Member Data Documentation	16
7.1.2.1 generators	16
7.1.2.2 package_type	16
7.1.2.3 settings	16
7.2 DBmanager Class Reference	16
7.2.1 Detailed Description	17
7.2.2 Constructor & Destructor Documentation	17
7.2.2.1 DBmanager()	17
7.2.2.2 ~DBmanager()	17
7.2.3 Member Function Documentation	17
7.2.3.1 connectTagToPhrase()	17
7.2.3.2 deleteAdmin()	17
7.2.3.3 deleteFavourite()	17
7.2.3.4 deletePhrase()	17
7.2.3.5 deleteTag()	17
7.2.3.6 deleteUser()	18
7.2.3.7 getAdmins()	18
7.2.3.8 getFavourites()	18
7.2.3.9 getPhrase()	18
7.2.3.10 getPhrases()	18
7.2.3.11 getPhraseWithTag()	18
7.2.3.12 getTags()	18
7.2.3.13 getUsers()	18
7.2.3.14 insertAdmin()	18
7.2.3.15 insertFavourite()	18
7.2.3.16 insertPhrase()	19
7.2.3.17 insertTag()	19
7.2.3.18 insertUser()	19
7.2.3.19 loginUser()	19
7.2.3.20 updateUserPassword()	19
7.3 Engine Class Reference	19

7.3.1 Detailed Description	20
7.3.2 Constructor & Destructor Documentation	20
7.3.2.1 Engine()	20
7.3.3 Member Function Documentation	20
7.3.3.1 Run()	20
7.4 FiniteStateMachine< T > Class Template Reference	20
7.4.1 Detailed Description	21
7.4.2 Constructor & Destructor Documentation	21
7.4.2.1 FiniteStateMachine()	21
7.4.3 Member Function Documentation	21
7.4.3.1 Add()	21
7.4.3.2 GetCurrentState() [1/2]	22
7.4.3.3 GetCurrentState() [2/2]	22
7.4.3.4 GetState()	22
7.4.3.5 OnUpdate()	23
7.4.3.6 SetCurrentState() [1/2]	23
7.4.3.7 SetCurrentState() [2/2]	23
7.4.4 Member Data Documentation	23
7.4.4.1 mCurrentState	23
7.4.4.2 mStates	23
7.5 PromptSingleton Class Reference	24
7.5.1 Detailed Description	24
7.5.2 Constructor & Destructor Documentation	24
7.5.2.1 PromptSingleton()	24
7.5.3 Member Function Documentation	24
7.5.3.1 GetInstance()	24
7.5.3.2 GetPrompt()	25
7.5.3.3 GetPromptAuto()	25
7.5.3.4 RetValues()	25
7.5.3.5 SetValues()	25
7.6 QueryHelper Class Reference	26
7.6.1 Detailed Description	26
7.6.2 Member Function Documentation	26
7.6.2.1 connectTagToPhrase()	26
7.6.2.2 createAdminTable()	26
7.6.2.3 createPhraseTable()	27
7.6.2.4 createPhraseTagTable()	27
7.6.2.5 createTagTable()	27
7.6.2.6 createUserTable()	27
7.6.2.7 deleteAdmin()	27
7.6.2.8 deleteFavourite()	27
7.6.2.9 deletePhrase()	27

7.6.2.10 deleteTag()	27
7.6.2.11 deleteUser()	27
7.6.2.12 getAdmins()	28
7.6.2.13 getFavourites()	28
7.6.2.14 getPhrase()	28
7.6.2.15 getPhrases()	28
7.6.2.16 getPhrasesWithTag()	28
7.6.2.17 getTags()	28
7.6.2.18 getUsers()	28
7.6.2.19 insertAdmin()	28
7.6.2.20 insertFavourite()	28
7.6.2.21 insertPhrase()	29
7.6.2.22 insertTag()	29
7.6.2.23 insertUser()	29
7.6.2.24 loginUser()	29
7.6.2.25 updateUserPass()	29
7.7 StackManager Class Reference	29
7.7.1 Detailed Description	30
7.7.2 Member Function Documentation	31
7.7.2.1 AskQuestion()	31
7.7.2.2 ChangeJsonToString()	31
7.7.2.3 ChangingSpecialChar()	31
7.7.2.4 checkTagQuestionList()	31
7.7.2.5 FillTabel()	31
7.7.2.6 GetAnswer()	31
7.7.2.7 GetQuestionFromID()	32
7.7.2.8 GetQuestionId()	32
7.7.2.9 getQuestionList()	32
7.7.2.10 GetTitle()	32
7.7.2.11 LookForByTags()	32
7.7.2.12 RemoveHtmlTags()	32
7.7.2.13 ReturnNiceCode()	32
7.7.2.14 SetQuestion()	33
7.7.2.15 SetQuestionByTags()	33
7.7.2.16 SetQuestionId()	33
7.7.3 Member Data Documentation	33
7.7.3.1 bestAnswer	33
7.8 State< T > Class Template Reference	33
7.8.1 Detailed Description	34
7.8.2 Constructor & Destructor Documentation	34
7.8.2.1 State()	34
7.8.2.2 ~State()	34

7.8.3 Member Function Documentation	35
7.8.3.1 getID()	35
7.8.3.2 GetName()	35
7.8.3.3 OnEnter()	35
7.8.3.4 OnExit()	35
7.8.3.5 OnUpdate()	35
7.8.4 Member Data Documentation	36
7.8.4.1 mFsm	36
7.8.4.2 mID	36
7.8.4.3 mName	36
7.9 StateAbout Class Reference	36
7.9.1 Detailed Description	37
7.9.2 Constructor & Destructor Documentation	37
7.9.2.1 StateAbout()	37
7.9.3 Member Function Documentation	37
7.9.3.1 OnEnter()	37
7.9.3.2 OnExit()	38
7.9.3.3 OnUpdate()	38
7.10 StateExit Class Reference	38
7.10.1 Detailed Description	39
7.10.2 Constructor & Destructor Documentation	39
7.10.2.1 StateExit()	39
7.10.3 Member Function Documentation	39
7.10.3.1 OnEnter()	39
7.10.3.2 OnExit()	39
7.10.3.3 OnUpdate()	39
7.11 StateFavourites Class Reference	40
7.11.1 Detailed Description	40
7.11.2 Constructor & Destructor Documentation	41
7.11.2.1 StateFavourites()	41
7.11.3 Member Function Documentation	41
7.11.3.1 OnEnter()	41
7.11.3.2 OnExit()	41
7.11.3.3 OnUpdate()	41
7.12 StateHistory Class Reference	41
7.12.1 Detailed Description	42
7.12.2 Constructor & Destructor Documentation	42
7.12.2.1 StateHistory()	42
7.12.3 Member Function Documentation	42
7.12.3.1 OnEnter()	42
7.12.3.2 OnExit()	43
7.12.3.3 OnUpdate()	43

7.13 StateIdle Class Reference	43
7.13.1 Detailed Description	44
7.13.2 Constructor & Destructor Documentation	44
7.13.2.1 StateIdle()	44
7.13.3 Member Function Documentation	44
7.13.3.1 OnEnter()	44
7.13.3.2 OnExit()	45
7.13.3.3 OnUpdate()	45
7.14 StateListTags Class Reference	45
7.14.1 Detailed Description	46
7.14.2 Constructor & Destructor Documentation	46
7.14.2.1 StateListTags()	46
7.14.3 Member Function Documentation	46
7.14.3.1 ChoosingTitle()	46
7.14.3.2 ManageList()	46
7.14.3.3 OnEnter()	47
7.14.3.4 OnExit()	47
7.14.3.5 OnUpdate()	47
7.15 StateLogin Class Reference	47
7.15.1 Detailed Description	48
7.15.2 Constructor & Destructor Documentation	48
7.15.2.1 StateLogin()	48
7.15.3 Member Function Documentation	48
7.15.3.1 OnEnter()	48
7.15.3.2 OnExit()	49
7.15.3.3 OnUpdate()	49
7.16 StateMenu Class Reference	49
7.16.1 Detailed Description	50
7.16.2 Constructor & Destructor Documentation	50
7.16.2.1 StateMenu()	50
7.16.3 Member Function Documentation	50
7.16.3.1 OnEnter()	50
7.16.3.2 OnExit()	50
7.16.3.3 OnUpdate()	50
7.17 StatePrompt Class Reference	51
7.17.1 Detailed Description	51
7.17.2 Constructor & Destructor Documentation	52
7.17.2.1 StatePrompt()	52
7.17.3 Member Function Documentation	52
7.17.3.1 OnEnter()	52
7.17.3.2 OnExit()	52
7.17.3.3 OnUpdate()	52

7.18 StateRegister Class Reference	52
7.18.1 Detailed Description	53
7.18.2 Constructor & Destructor Documentation	53
7.18.2.1 StateRegister()	53
7.18.3 Member Function Documentation	53
7.18.3.1 OnEnter()	53
7.18.3.2 OnExit()	54
7.18.3.3 OnUpdate()	54
7.19 StateResult Class Reference	54
7.19.1 Detailed Description	55
7.19.2 Constructor & Destructor Documentation	55
7.19.2.1 StateResult()	55
7.19.3 Member Function Documentation	55
7.19.3.1 OnEnter()	55
7.19.3.2 OnExit()	55
7.19.3.3 OnUpdate()	55
7.19.3.4 QuestionManage()	56
7.20 StateResultTags Class Reference	56
7.20.1 Detailed Description	57
7.20.2 Constructor & Destructor Documentation	57
7.20.2.1 StateResultTags()	57
7.20.3 Member Function Documentation	57
7.20.3.1 OnEnter()	57
7.20.3.2 OnExit()	57
7.20.3.3 OnUpdate()	57
7.20.3.4 QuestionManage()	58
7.21 StateTags Class Reference	58
7.21.1 Detailed Description	59
7.21.2 Constructor & Destructor Documentation	59
7.21.2.1 StateTags()	59
7.21.3 Member Function Documentation	59
7.21.3.1 OnEnter()	59
7.21.3.2 OnExit()	59
7.21.3.3 OnUpdate()	59
7.22 SyntaxHighlighting Class Reference	59
7.22.1 Detailed Description	60
7.22.2 Constructor & Destructor Documentation	60
7.22.2.1 SyntaxHighlighting()	60
7.22.3 Member Function Documentation	60
7.22.3.1 ColorBracket()	60
7.22.3.2 ColorChar()	60
7.22.3.3 Hightlighting()	61

7.22.3.4 RecognizeSyntax()	61
7.22.3.5 RemoveTags()	61
7.23 TagsList Class Reference	61
7.23.1 Detailed Description	61
7.23.2 Constructor & Destructor Documentation	61
7.23.2.1 TagsList()	61
7.23.3 Member Function Documentation	62
7.23.3.1 GetID()	62
7.23.3.2 GetTitle()	62
8 File Documentation	63
8.1 conanfile.py File Reference	63
8.2 Engine.cpp File Reference	63
8.3 Engine.hpp File Reference	63
8.4 Engine.hpp	64
8.5 FSM/State.hpp File Reference	64
8.6 State.hpp	64
8.7 FSM/StateMachine.hpp File Reference	65
8.8 StateMachine.hpp	65
8.9 Globals.hpp File Reference	66
8.10 Globals.hpp	66
8.11 Logic/Database/DBmanager.cpp File Reference	67
8.11.1 Typedef Documentation	67
8.11.1.1 sqlite3_callback	67
8.11.2 Variable Documentation	67
8.11.2.1 receivedData	67
8.12 Logic/Database/DBmanager.hpp File Reference	67
8.13 DBmanager.hpp	68
8.14 Logic/Database/QueryHelper.cpp File Reference	68
8.15 Logic/Database/QueryHelper.hpp File Reference	69
8.16 QueryHelper.hpp	69
8.17 Logic/PromptSingleton.cpp File Reference	69
8.17.1 Function Documentation	70
8.17.1.1 GetMatch()	70
8.18 Logic/PromptSingleton.hpp File Reference	70
8.19 PromptSingleton.hpp	70
8.20 Logic/StackApi/StackManager.cpp File Reference	71
8.21 Logic/StackApi/StackManager.hpp File Reference	71
8.22 StackManager.hpp	71
8.23 Logic/StackApi/Syntax.hpp File Reference	72
8.24 Syntax.hpp	72
8.25 Logic/StackApi/SyntaxHighlighting.cpp File Reference	75

8.26 Logic/StackApi/SyntaxHighlighting.hpp File Reference	75
8.27 SyntaxHighlighting.hpp	75
8.28 Logic/TagList/TagsList.cpp File Reference	76
8.29 Logic/TagList/TagsList.hpp File Reference	76
8.30 TagsList.hpp	76
8.31 Logic/TextFormatter.hpp File Reference	76
8.32 TextFormatter.hpp	77
8.33 main.cpp File Reference	78
8.33.1 Function Documentation	78
8.33.1.1 main()	78
8.33.1.2 PrintHelp()	78
8.34 README.md File Reference	79
8.35 States/StateAbout.cpp File Reference	79
8.36 States/StateAbout.hpp File Reference	79
8.37 StateAbout.hpp	79
8.38 States/StateExit.cpp File Reference	80
8.39 States/StateExit.hpp File Reference	80
8.40 StateExit.hpp	80
8.41 States/StateFavourites.cpp File Reference	81
8.42 States/StateFavourites.hpp File Reference	81
8.43 StateFavourites.hpp	81
8.44 States/StateHistory.cpp File Reference	82
8.45 States/StateHistory.hpp File Reference	82
8.46 StateHistory.hpp	82
8.47 States/StateIdle.cpp File Reference	83
8.48 States/StateIdle.hpp File Reference	83
8.49 StateIdle.hpp	83
8.50 States/StateListTags.cpp File Reference	84
8.51 States/StateListTags.hpp File Reference	84
8.52 StateListTags.hpp	84
8.53 States/StateLogin.cpp File Reference	85
8.54 States/StateLogin.hpp File Reference	85
8.55 StateLogin.hpp	85
8.56 States/StateMenu.cpp File Reference	85
8.57 States/StateMenu.hpp File Reference	86
8.58 StateMenu.hpp	86
8.59 States/StatePrompt.cpp File Reference	86
8.60 States/StatePrompt.hpp File Reference	87
8.61 StatePrompt.hpp	87
8.62 States/StateRegister.cpp File Reference	87
8.63 States/StateRegister.hpp File Reference	88
8.64 StateRegister.hpp	88

8.65 States/StateResult.cpp File Reference	88
8.66 States/StateResult.hpp File Reference	89
8.67 StateResult.hpp	89
8.68 States/StateResultTags.cpp File Reference	89
8.69 States/StateResultTags.hpp File Reference	90
8.70 StateResultTags.hpp	90
8.71 States/StatesConf.hpp File Reference	90
8.71.1 Enumeration Type Documentation	91
8.71.1.1 States	91
8.72 StatesConf.hpp	91
8.73 States/StatesWrapper.hpp File Reference	91
8.74 StatesWrapper.hpp	92
8.75 States/StateTags.cpp File Reference	92
8.76 States/StateTags.hpp File Reference	92
8.77 StateTags.hpp	93
8.78 Texts/AllTexts.hpp File Reference	93
8.79 AllTexts.hpp	94
Index	99

Chapter 1

README

Purpose: Have you find yourself distracted with some nonsense 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: Doxygen: 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:

AboutTexts	Namespace for About state	11
cmd	CMD - Namespace responsible for holding globals connected to shell application	11
conanfile	11
FavouriteTexts	Namespace for Favourites state	11
HistoryTexts	Namespace for History state	12
IdleTexts	Namespace for Idle state	12
ListState	Namespace for List state	12
LoginTexts	Namespace for Login state	12
Manual	Namespace for manual	13
MenuTexts	Namespace for Menu state	13
PromptTexts	Namespace for Prompt state	13
RegisterTexts	Namespace for Register state	13
ResultTexts	Namespace for Result state	13
Syntax	14
TagsTexts	Namespace for Tags state	14
TextColors	Viable colors of the text	14
TextFunctions	14

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

DBmanager	16
Engine	19
FiniteStateMachine< T >	20
FiniteStateMachine< States >	20
PromptSingleton	24
QueryHelper	26
StackManager	29
State< T >	33
State< States >	33
StateAbout	36
StateExit	38
StateFavourites	40
StateHistory	41
StateIdle	43
StateListTags	45
StateLogin	47
StateMenu	49
StatePrompt	51
StateRegister	52
StateResult	54
StateResultTags	56
StateTags	58
SyntaxHighlighting	59
TagsList	61
ConanFile	
conanfile.ConanApplication	15

Chapter 4

Class Index

4.1 Class List

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

conanfile.ConanApplication	15
DBmanager	16
Engine	19
FiniteStateMachine< T >	20
PromptSingleton	24
QueryHelper	26
StackManager	29
State< T >	33
StateAbout	36
StateExit	38
StateFavourites	40
StateHistory	41
StateIdle	43
StateListTags	45
StateLogin	47
StateMenu	49
StatePrompt	51
StateRegister	52
StateResult	54
StateResultTags	56
StateTags	58
SyntaxHighlighting	59
TagsList	61

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

conanfile.py	63
Engine.cpp	63
Engine.hpp	63
Globals.hpp	66
main.cpp	78
FSM/State.hpp	64
FSM/StateMachine.hpp	65
Logic/PromptSingleton.cpp	69
Logic/PromptSingleton.hpp	70
Logic/TextFormatter.hpp	76
Logic/Database/DBmanager.cpp	67
Logic/Database/DBmanager.hpp	67
Logic/Database/QueryHelper.cpp	68
Logic/Database/QueryHelper.hpp	69
Logic/StackApi/StackManager.cpp	71
Logic/StackApi/StackManager.hpp	71
Logic/StackApi/Syntax.hpp	72
Logic/StackApi/SyntaxHighlighting.cpp	75
Logic/StackApi/SyntaxHighlighting.hpp	75
Logic/TagList/TagsList.cpp	76
Logic/TagList/TagsList.hpp	76
States/StateAbout.cpp	79
States/StateAbout.hpp	79
States/StateExit.cpp	80
States/StateExit.hpp	80
States/StateFavourites.cpp	81
States/StateFavourites.hpp	81
States/StateHistory.cpp	82
States/StateHistory.hpp	82
States/StateIdle.cpp	83
States/StateIdle.hpp	83
States/StateListTags.cpp	84
States/StateListTags.hpp	84
States/StateLogin.cpp	85
States/StateLogin.hpp	85

States/StateMenu.cpp	85
States/StateMenu.hpp	86
States/StatePrompt.cpp	86
States/StatePrompt.hpp	87
States/StateRegister.cpp	87
States/StateRegister.hpp	88
States/StateResult.cpp	88
States/StateResult.hpp	89
States/StateResultTags.cpp	89
States/StateResultTags.hpp	90
States/StatesConf.hpp	90
States/StatesWrapper.hpp	91
States/StateTags.cpp	92
States/StateTags.hpp	92
Texts/AllTexts.hpp	93

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

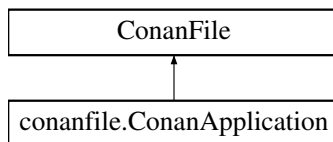
Namespace for functions which adjust printing of texts

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 phrasId)
- 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 phrasId)
- std::vector< std::pair< std::string, std::string > > [getFavourites](#) ()
- bool [deleteFavourite](#) (int favId)
- bool [connectTagToPhrase](#) (int phrasId, 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

<i>id</i>	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

<i>stateID</i>	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

<i>state</i>	
--------------	--

7.4.3.7 SetCurrentState() [2/2]

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

Function for changing state as desired

Parameters

<i>stateID</i>	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

<i>obj</i>	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

<i>dict</i>	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

<i>val</i>	value to be set
------------	-----------------

Function to change current value of prompt

Parameters

<i>val</i>	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 [FillTable](#) (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

<i>T</i>	template class which State is created for
----------	---

Predefinition of [State](#) class

Template Parameters

<i>T</i>	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

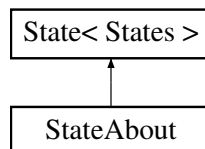
The documentation for this class was generated from the following file:

- 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 >](#).

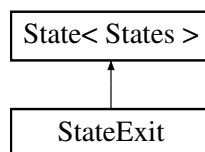
The documentation for this class was generated from the following files:

- [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 >](#).

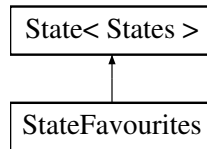
The documentation for this class was generated from the following files:

- [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 >](#).

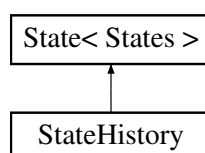
The documentation for this class was generated from the following files:

- [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 >](#).

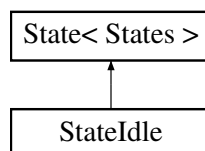
The documentation for this class was generated from the following files:

- [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

<i>fsm</i>	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 >](#).

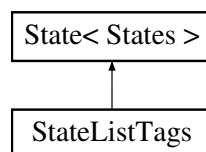
The documentation for this class was generated from the following files:

- [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 >](#).

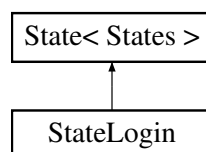
The documentation for this class was generated from the following files:

- [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 referring to dbmanager instance

Reimplemented from [State< States >](#).

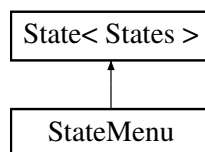
The documentation for this class was generated from the following files:

- [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 >](#).

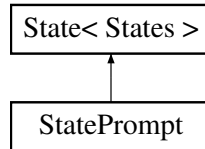
The documentation for this class was generated from the following files:

- [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](#) >.

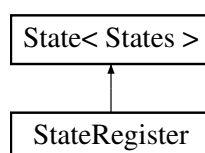
The documentation for this class was generated from the following files:

- [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 >](#).

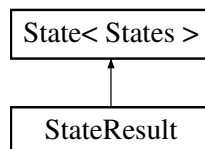
The documentation for this class was generated from the following files:

- [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

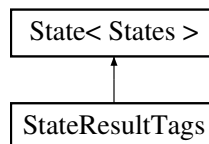
The documentation for this class was generated from the following files:

- [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 ()
```

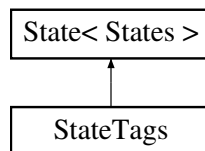
The documentation for this class was generated from the following files:

- 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 Highlighting()

```
std::string SyntaxHighlighting::Highlighting (
    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

<code>_id</code>	id of the question
<code>_title</code>	title of the question

7.23.3 Member Function Documentation

7.23.3.1 GetID()

```
int TagList::GetID ()
```

Returns

id of the tag

7.23.3.2 GetTitle()

```
std::string TagList::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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
00007
00008 #include "States/StatesWrapper.hpp"
00009 #include "FSM/StateMachine.hpp"
00010 #include "Logic/PromptSingleton.hpp"
00011
00012
00017 class Engine {
00018     std::unique_ptr<FiniteStateMachine<States>> fsm = nullptr;
00019 public:
00020     Engine();
00021     void Run();
00022 };
00023
00024
00025 #endif //INC_2024__TAB_DSA__8_BRODZIAK_ENGINE_HPP
```

8.5 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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATE_HPP
00007
00011 #include "StateMachine.hpp"
00012
00013 #include <string>
00014 #include <utility>
00015 template <typename T>
00016 class FiniteStateMachine;
00017
00022 template <typename T>
00023 class State
00024 {
00025 public:
00027     inline T getID()
00028     {
00029         return mID;
00030     }
00032     inline const std::string& GetName() const
00033     {
00034         return mName;
00035     }
00037     explicit State(FiniteStateMachine<T>& fsm, T id,
00038                   std::string name = "default")
00039         : mName(name)
```

```

00040         , mID(id)
00041         , mFsm(fsm)
00042     {
00043     }
00044     virtual ~State() {}
00045     virtual void OnEnter()
00046     {
00047     }
00048     virtual void OnExit()
00049     {
00050     }
00051     virtual void OnUpdate()
00052     {
00053     }
00054     protected:
00055     std::string mName;
00056     T mID;
00057     FiniteStateMachine<T>& mFsm;
00058 };
00059 #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.](#)

```

00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEMACHINE_HPP
00007
00008 #include "State.hpp"
00009
00010 #include <memory>
00011 #include <map>
00012 #include <string>
00013 #include <cassert>
00014 #include <utility>
00015
00016 template <typename T>
00017 class State;
00018
00019 template <typename T>
00020 class FiniteStateMachine
00021 {
00022     protected:
00023         std::map<T, std::unique_ptr<State<T>>> mStates;
00024         State<T>* mCurrentState;
00025     public:
00026         FiniteStateMachine()
00027             : mCurrentState(nullptr)
00028         {}
00029         template <class S>

```

```

00045     State<T>& Add(T id)
00046     {
00047         static_assert(not std::is_same<State<T>, S>());
00048         mStates[id] = std::make_unique<S>(*this);
00049         return *mStates[id];
00050     }
00051     State<T>& GetState(T stateID)
00052     {
00053         return *mStates[stateID];
00054     }
00055     State<T>& GetCurrentState()
00056     {
00057         return *mCurrentState;
00058     }
00059     const State<T>& GetCurrentState() const
00060     {
00061         return *mCurrentState;
00062     }
00063     void SetCurrentState(T stateID)
00064     {
00065         State<T>* state = &GetState(stateID);
00066         SetCurrentState(state);
00067     }
00068     void OnUpdate()
00069     {
00070         if (mCurrentState != nullptr)
00071         {
00072             mCurrentState->OnUpdate();
00073         }
00074     }
00075 protected:
00076     void SetCurrentState(State<T>* state)
00077     {
00078         if (mCurrentState == state)
00079         {
00080             return;
00081         }
00082         if (mCurrentState != nullptr)
00083         {
00084             mCurrentState->OnExit();
00085         }
00086         mCurrentState = state;
00087         if (mCurrentState != nullptr)
00088         {
00089             mCurrentState->OnEnter();
00090         }
00091     }
00092 };
00093 #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.](#)

```

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

```

```

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

```

8.11 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.](#)

```

00001 //
00002 // Created by lucja on 11.05.2024.
00003 //
00004
00005 #ifndef DBMANAGER_HPP
00006 #define DBMANAGER_HPP
00007
00008 #include<string>
00009 #include<vector>
00010 #include <sqlite3.h>
00011 #include "QueryHelper.hpp"
00012
00013 class DBmanager {
00014     static std::string nickName;
00015     static int id;
00016
00017     sqlite3 *db;
00018     char *zErrMsg;
00019     int rc;
00020     const char* data = "Callback function called";
00021
00022     int openDatabase();
00023     int createDatabase();
00024     int closeDatabase();
00025
00026     int createUserTable();
00027     int createAdminTable();
00028     int createPhraseTable();
00029     int createTagTable();
00030     int createPhraseTagTable();
00031 public:
00032     bool insertUser(std::string& nickname, std::string& password);
00033     std::vector<std::pair<std::string, std::string>> getUsers();
00034     bool updateUserPassword(int id, std::string& password);
00035     bool deleteUser(int id);
00036
00037     bool loginUser(std::string& log, std::string& pass);
00038
00039     bool insertAdmin(int Id);
00040     std::vector<std::pair<std::string, std::string>> getAdmins();
00041     bool deleteAdmin(int adminId);
00042
00043     bool insertPhrase(std::string &body, std::string &response);
00044     std::vector<std::pair<std::string, std::string>> getPhrases();
00045     std::vector<std::pair<std::string, std::string>> getPhrase(int phraseId);
00046     bool deletePhrase(int id);
00047
00048     bool insertTag(std::string& body);
00049     std::vector<std::pair<std::string, std::string>> getTags();
00050     bool deleteTag(int id);
00051
00052     bool insertFavourite(int phraseId);
00053     std::vector<std::pair<std::string, std::string>> getFavourites();
00054     bool deleteFavourite(int favId);
00055
00056     bool connectTagToPhrase(int phraseId, int tagId);
00057     std::vector<std::pair<std::string, std::string>> getPhraseWithTag();
00058
00059     DBmanager();
00060     ~DBmanager();
00061 };
00062
00063 #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.](#)

```
00001 //
00002 // Created by lucja on 21.05.2024.
00003 //
00004
00005 #ifndef QUERYHELPER_HPP
00006 #define QUERYHELPER_HPP
00007
00008 #include<string>
00009 #include <format>
00010
00015 class QueryHelper {
00016 public:
00017     static std::string createUserTable();
00018     static std::string createAdminTable();
00019     static std::string createPhraseTable();
00020     static std::string createTagTable();
00021     static std::string createPhraseTagTable();
00022
00023     static std::string insertUser(std::string nick, std::string pass);
00024     static std::string getUsers();
00025     static std::string deleteUser(int id);
00026     static std::string updateUserPass(int id, std::string pass);
00027
00028     static std::string insertAdmin(int userId);
00029     static std::string getAdmins();
00030     static std::string deleteAdmin(int adminId);
00031
00032     static std::string loginUser(std::string &log, std::string &pass);
00033
00034     static std::string insertPhrase(int &id, std::string &body, std::string &response);
00035     static std::string getPhrases();
00036     static std::string getPhrase(int phraseId);
00037     static std::string deletePhrase(int id);
00038
00039     static std::string insertTag(std::string body);
00040     static std::string getTags();
00041     static std::string deleteTag(int id);
00042
00043     static std::string insertFavourite(int phraseId);
00044     static std::string getFavourites(int userId);
00045     static std::string deleteFavourite(int phraseId);
00046
00047     static std::string connectTagToPhrase(int phraseId, int tagId);
00048     static std::string getPhrasesWithTag();
00049 };
00050
00051
00052 #endif //QUERYHELPER_HPP
```

8.17 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

<i>text</i>	text to be matched
<i>dict</i>	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.](#)

```
00001 //
00002 // Created by Michin on 23.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP
00007
00008 #include <string>
00009 #include <iostream>
00010 #include <vector>
00011
00015 class PromptSingleton{
00016 private:
00017     std::string prompt;
00018     static PromptSingleton* instancePtr;
00019     PromptSingleton()= default;
00020 public:
```

```

00021     PromptSingleton(const PromptSingleton& obj)
00022     = delete;
00027     static PromptSingleton* GetInstance();
00032     void SetValues(std::string& val);
00037     std::string RetValues(){ return prompt; }
00041     void GetPrompt();
00046     void GetPromptAuto(std::vector<std::string> dict);
00047 };
00048
00049
00050
00051
00052
00053 #endif //INC_2024__TAB_DSA__8_BRODZIAK_PROMPTSINGLETON_HPP

```

8.20 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.](#)

```

00001 //
00002 // Created by jakub on 10.05.2024.
00003 //
00004
00005 #ifndef STACKMANAGER_HPP
00006 #define STACKMANAGER_HPP
00007
00008 #include <algorithm>
00009 #include <iostream>
00010 #include <regex>
00011 #include "nlohmann/adl_serializer.hpp"
00012 #include "../TagList/TagsList.hpp"
00013
00017 class StackManager {
00018     //URL TO SEARCH
00019     //https://api.stackexchange.com/2.3/search?order=desc&sort=activity&intitle=CPP&site=stackoverflow
00020
00021     //value which store answer id "accepted_answer_id": 63548573,
00022     //URL TO FIND ANSWER

```

```

00023 //https://api.stackexchange.com/2.3/answers/63548573?order=desc&sort=activity&site=stackoverflow&filter=withbody
00024
00025     std::string space = "%20";
00026     std::string baseInput = "https://api.stackexchange.com/";
00027     std::string apiVesion = "2.3/";
00028     std::string questionInput = "";
00029     std::string finalInput = "";
00030     std::string answerInput = "";
00031     std::string stringQuestionID = "";
00032     int questionID = 0;
00033     std::string title;
00034     static std::vector<TagsList> questionsList;
00035
00036
00037
00038 public:
00039
00040
00041     std::string bestAnswer[3] = {"", "", ""};
00042     void AskQuestion(std::string & question);
00043     void SetQuestion(std::string newInput);
00044     void SetQuestionByTags(std::string newInput);
00045     void GetAnswer(std::string res);
00046     void ChangeJsonToString(std::string&);
00047     void SetQuestionId(std::string);
00048     void FillTabel(std::string input);
00049     void RemoveHtmlTags(std::string& input);
00050     void ReturnNiceCode(std::string& input);
00051     void ChangingSpecialChar(std::string &input, std::string inChar, std::string outChar);
00052     void LookForByTags(std::string& input);
00053     void checkTagQuestionList(std::string& tagInput);
00054     static std::vector<TagsList> getQuestionList();
00055     std::string GetTitle();
00056     std::string GetQuestionId();
00057     void GetQuestionFromID(std::string id);
00058 };
00059
00060
00061
00062 #endif //STACKMANAGER_HPP

```

8.23 Logic/StackApi/Syntax.hpp File Reference

```

#include <vector>
#include <string>

```

Namespaces

- namespace [Syntax](#)

8.24 Syntax.hpp

[Go to the documentation of this file.](#)

```

00001 //
00002 // Created by jakub on 23.05.2024.
00003 //
00004 #include <vector>
00005 #include <string>
00006
00007 #ifndef SYNTAX_HPP
00008 #define SYNTAX_HPP
00009
00013 namespace Syntax {
00014     static std::vector<std::string> basicSyntax = {
00015         "for", "while", "do",
00016         "if", "else", "int",
00017         "string", ":", "std",
00018         "double", "float", "bool",

```

```

00019    "main", "switch", "case",
00020    "char", "cin", "getline",
00021    "cout", "return", "long",
00022    "short", "cerr", "«",
00023    "»", "include", "using",
00024    "NULL", "nullptr", "class",
00025    "void", "private", "public",
00026    "*", "&", "\\",
00027    "=", "const", "static",
00028    "delete", "new", "break",
00029    "continue", "protected", "enum",
00030    "typedef", "try",
00031    "catch", "throw", "template",
00032    "operator", "this", "friend",
00033    "volatile", "extern", "struct",
00034    "sizeof", "finally", "AND",
00035    "OR", "&&", "||",
00036    "false", "true",
00037    //PYTHON
00038    "False", "None", "True",
00039    "and", "as", "assert",
00040    "def", "del", "await",
00041    "async", "elif", "except",
00042    "global", "from", "import",
00043    "in", "is", "lambda",
00044    "not", "!", "raise",
00045    "with", "pass", "yield",
00046    //C KEYWORD
00047    "auto", "default", "inline",
00048    "signed", "malloc", "printf",
00049    "free",
00050    //JAVA KEYWORD
00051    "abstract", "boolean", "implements",
00052    "interface", "native", "package",
00053    "super",
00054    //PHP KEYWORD
00055    "array", "clone", "declare",
00056    "echo", "elseif", "foreach",
00057    "empty", "endfor", "endif",
00058    "endforeach", "endswitch", "isset",
00059    "unset", "var", "use",
00060    "xor",
00061    //JS KEYWORD
00062    "let", "function", "export",
00063    //HTML TAGS
00064    "div", "<", ">",
00065    "area", "blockquote", "body",
00066    "html", "head", "button",
00067    "dl", "dt", "h1",
00068    "h2", "h3", "h4",
00069    "h5", "h6", "nav",
00070    "script", "strong", "style",
00071    "td", "table", "sup",
00072    "ul", "ol", "li",
00073    "p", "b", "s",
00074    "i", "br", "td",
00075    "a", "img", "tr",
00076    //others
00077    "print", "namespace", "__name__",
00078    "__main__", "__init__",
00079    //css
00080    "display", "position", "top",
00081    "float", "clear", "both",
00082    "width", "height", "min-height",
00083    "min-width", "margin", "padding",
00084    "color", "font", "text-align",
00085    "text-decoration", "letter-spacing", "border",
00086    "transform", "transition", "flex",
00087    "flex-align", "flex-directory", "flex-wrap",
00088    "justift-content", "grid", "grid-template-columns",
00089    "grid-templeta-rows", "cursor", "pointer",
00090    ":hover", ":focus", "visted",
00091    "margin-left", "margin-right", "margin-top",
00092    "margin-bottom", "left", "right",
00093    "bottom", "overflow", "hidden",
00094    "background-color", "background", "opacity",
00095    "absolute", "fixed", "style",
00096    "span", "input", "placeholder",
00097    "#ifndef", "define", "regex",
00098    "println"
00099    };
00100    };
00101    static std::vector<std::string> keyWord = {
00102    //CPP KEYWORD
00103    "\033[0;32mfor\033[0m", "\033[0;32mwhile\033[0m", "\033[0;32mdo\033[0m",
00104    "\033[0;34mif\033[0m", "\033[0;34melse\033[0m", "\033[0;33mint\033[0m",
00105    "\033[0;33mstring\033[0m", "\033[0;31m::\033[0m", "\033[0;35mstd\033[0m",

```

```

00106     "\\033[0;33mdouble\\033[0m", "\\033[0;33mfloat\\033[0m", "\\033[0;33mbool\\033[0m",
00107     "\\033[0;34mmain\\033[0m", "\\033[0;36mswitch\\033[0m", "\\033[0;33mcase\\033[0m",
00108     "\\033[0;33mchar\\033[0m", "\\033[0;31mcin\\033[0m", "\\033[0;31mgetline\\033[0m",
00109     "\\033[0;31mcout\\033[0m", "\\033[0;31mreturn\\033[0m", "\\033[0;33mlong\\033[0m",
00110     "\\033[0;33mshort\\033[0m", "\\033[0;31mcerr\\033[0m", "\\033[0;32m«\\033[0m",
00111     "\\033[0;32m»\\033[0m", "\\033[0;33minclude\\033[0m", "\\033[0;32musing\\033[0m",
00112     "\\033[0;32mNULL\\033[0m", "\\033[0;32mnullptr\\033[0m", "\\033[0;33mclass\\033[0m",
00113     "\\033[0;33mvoid\\033[0m", "\\033[0;31mprivate\\033[0m", "\\033[0;32mpublic\\033[0m",
00114     "\\033[0;34m*\\033[0m", "\\033[0;34m&\\033[0m", "\\033[0;32m\\033[0m",
00115     "\\033[0;33m=\\033[0m", "\\033[0;35mconst\\033[0m", "\\033[0;35mstatic\\033[0m",
00116     "\\033[0;31mdelete\\033[0m", "\\033[0;36mnew\\033[0m", "\\033[0;31mbreak\\033[0m",
00117     "\\033[0;33mcontinue\\033[0m", "\\033[0;33mprotected\\033[0m", "\\033[0;33mmenu\\033[0m",
00118     "\\033[0;32mtypedef\\033[0m", "\\033[0;36mtry\\033[0m",
00119     "\\033[0;36mcatch\\033[0m", "\\033[0;31mthrow\\033[0m", "\\033[0;34mtemplate\\033[0m",
00120     "\\033[0;33moperator\\033[0m", "\\033[0;32mthis\\033[0m", "\\033[0;35mfriend\\033[0m",
00121     "\\033[0;33mvolatile\\033[0m", "\\033[0;33mextern\\033[0m", "\\033[0;33mstruct\\033[0m",
00122     "\\033[0;33msizeof\\033[0m", "\\033[0;33mfinally\\033[0m", "\\033[0;32mAND\\033[0m",
00123     "\\033[0;32mOR\\033[0m", "\\033[0;32m&&\\033[0m", "\\033[0;32m|\\033[0m",
00124     "\\033[0;32mfalse\\033[0m", "\\033[0;32mtrue\\033[0m",
00125     //PYTHON KEYWORD
00126     "\\033[0;31mfalse\\033[0m", "\\033[0;33mNone\\033[0m", "\\033[0;32mtrue\\033[0m",
00127     "\\033[0;32mand\\033[0m", "\\033[0;33mAs\\033[0m", "\\033[0;33mAssert\\033[0m",
00128     "\\033[0;33mdef\\033[0m", "\\033[0;31mdel\\033[0m", "\\033[0;35mawit\\033[0m",
00129     "\\033[0;33masync\\033[0m", "\\033[0;34melif\\033[0m", "\\033[0;31mexcept\\033[0m",
00130     "\\033[0;36mglobal\\033[0m", "\\033[0;35mfrom\\033[0m", "\\033[0;35mimport\\033[0m",
00131     "\\033[0;35min\\033[0m", "\\033[0;35mis\\033[0m", "\\033[0;36mlambda\\033[0m",
00132     "\\033[0;31mnot\\033[0m", "\\033[0;31m!\\033[0m", "\\033[0;36mraise\\033[0m",
00133     "\\033[0;36mwith\\033[0m", "\\033[0;35mpass\\033[0m", "\\033[0;36myield\\033[0m",
00134     //C KEYWORD
00135     "\\033[0;33mauto\\033[0m", "\\033[0;34mdefault\\033[0m", "\\033[0;34minline\\033[0m",
00136     "\\033[0;33msigned\\033[0m", "\\033[0;31mmalloc\\033[0m", "\\033[0;31mprintf\\033[0m",
00137     "\\033[0;32mfree\\033[0m",
00138     //JAVA KEYWORD
00139     "\\033[0;33mabstract\\033[0m", "\\033[0;33mboolean\\033[0m", "\\033[0;36mimplements\\033[0m",
00140     "\\033[0;33mnative\\033[0m", "\\033[0;35mnative\\033[0m", "\\033[0;35mpackage\\033[0m",
00141     "\\033[0;32msuper\\033[0m",
00142     //PHP KEYWORD
00143     "\\033[0;33marray\\033[0m", "\\033[0;35mclone\\033[0m", "\\033[0;35mdeclare\\033[0m",
00144     "\\033[0;32mecho\\033[0m", "\\033[0;34melseif\\033[0m", "\\033[0;32mforeach\\033[0m",
00145     "\\033[0;36mempty\\033[0m", "\\033[0;32mendfor\\033[0m", "\\033[0;34mendif\\033[0m",
00146     "\\033[0;32mendforeach\\033[0m", "\\033[0;36mendswitch\\033[0m", "\\033[0;33mmisset\\033[0m",
00147     "\\033[0;33munset\\033[0m", "\\033[0;36mvar\\033[0m", "\\033[0;31muse\\033[0m",
00148     "\\033[0;33mxor\\033[0m",
00149     //JS KEYWORD
00150     "\\033[0;36mlet\\033[0m", "\\033[0;32mfunction\\033[0m", "\\033[0;32mexport\\033[0m",
00151     //HTML TAGS
00152     "\\033[0;36mdiv\\033[0m", "\\033[0;32m<\\033[0m", "\\033[0;32m>\\033[0m",
00153     "\\033[0;35marea\\033[0m", "\\033[0;36mblockquote\\033[0m", "\\033[0;31mbody\\033[0m",
00154     "\\033[0;31mhtml\\033[0m", "\\033[0;31mhead\\033[0m", "\\033[0;32mbutton\\033[0m",
00155     "\\033[0;32mdl\\033[0m", "\\033[0;32mdt\\033[0m", "\\033[0;34mh1\\033[0m",
00156     "\\033[0;34mh2\\033[0m", "\\033[0;34mh3\\033[0m", "\\033[0;34mh4\\033[0m",
00157     "\\033[0;34mh5\\033[0m", "\\033[0;34mh6\\033[0m", "\\033[0;32mnav\\033[0m",
00158     "\\033[0;33mscript\\033[0m", "\\033[0;31mstrong\\033[0m", "\\033[0;32mstyle\\033[0m",
00159     "\\033[0;32mtd\\033[0m", "\\033[0;32mtable\\033[0m", "\\033[0;32msup\\033[0m",
00160     "\\033[0;33mul\\033[0m", "\\033[0;33mol\\033[0m", "\\033[0;33mli\\033[0m",
00161     "\\033[0;33mp\\033[0m", "\\033[0;31mb\\033[0m", "\\033[0;33ms\\033[0m",
00162     "\\033[0;33mi\\033[0m", "\\033[0;35mbr\\033[0m", "\\033[0;31mtd\\033[0m",
00163     "\\033[0;34ma\\033[0m", "\\033[0;32mimg\\033[0m", "\\033[0;31mtr\\033[0m",
00164     //others
00165     "\\033[0;31mprint\\033[0m", "\\033[0;32mnamespace\\033[0m", "\\033[0;35m__name__\\033[0m",
00166     "\\033[0;35m__main__\\033[0m", "\\033[0;35m__init__\\033[0m",
00167     //css
00168     "\\033[0;31mdisplay\\033[0m", "\\033[0;31mposition\\033[0m", "\\033[0;31mtop\\033[0m",
00169     "\\033[0;31mfloat\\033[0m", "\\033[0;31mfloat\\033[0m", "\\033[0;32mboth\\033[0m",
00170     "\\033[0;31mwidth\\033[0m", "\\033[0;31mheight\\033[0m", "\\033[0;31mmin-height\\033[0m",
00171     "\\033[0;31mmin-width\\033[0m", "\\033[0;31mmargin\\033[0m", "\\033[0;31mpadding\\033[0m",
00172     "\\033[0;31mcolor\\033[0m", "\\033[0;31mfont\\033[0m", "\\033[0;31mtext-align\\033[0m",
00173     "\\033[0;31mtext-decoration\\033[0m", "\\033[0;31mletter-spacing\\033[0m", "\\033[0;31mborder\\033[0m",
00174     "\\033[0;31mtransform\\033[0m", "\\033[0;31mtransition\\033[0m", "\\033[0;31mflex\\033[0m",
00175     "\\033[0;31mflex-align\\033[0m", "\\033[0;31mflex-directory\\033[0m", "\\033[0;31mflex-wrap\\033[0m",
00176     "\\033[0;31mjustify-content\\033[0m", "\\033[0;31mgrid\\033[0m",
    "\\033[0;31mgrid-template-columns\\033[0m",
00177     "\\033[0;31mgrid-template-rows\\033[0m", "\\033[0;31mcursor\\033[0m", "\\033[0;32mpointer\\033[0m",
00178     "\\033[0;35mhover\\033[0m", "\\033[0;35mfocus\\033[0m", "\\033[0;35mvisted\\033[0m",
00179     "\\033[0;31mmargin-left\\033[0m", "\\033[0;31mmargin-right\\033[0m", "\\033[0;31mmargin-top\\033[0m",
00180     "\\033[0;31mmargin-bottom\\033[0m", "\\033[0;31mleft\\033[0m", "\\033[0;31mright\\033[0m",
00181     "\\033[0;31mbottom\\033[0m", "\\033[0;31moverflow\\033[0m", "\\033[0;32mhidden\\033[0m",
00182     "\\033[0;31mbackground-color\\033[0m", "\\033[0;31mbackground\\033[0m", "\\033[0;31mopacity\\033[0m",
00183     "\\033[0;32mabsolute\\033[0m", "\\033[0;32mfixed\\033[0m", "\\033[0;32mstyle\\033[0m",
00184     "\\033[0;31mspan\\033[0m", "\\033[0;33minput\\033[0m", "\\033[0;32mplaceholder\\033[0m",
00185     "\\033[0;35mifn\\033[0m", "\\033[0;34mdefine\\033[0m", "\\033[0;32mregex\\033[0m",
00186     "\\033[0;36mprintln\\033[0m"
00187 };
00188 static std::vector<std::string> specialCharacter = {
00189     "<", ">", "\"",
00190     "\\", " ", "%",
00191     "|", "$", ":",

```

```

00192     "->", "#", "%"
00193 };
00194 static std::vector<std::string> colorSpecialCharacter = {
00195     "\033[0;32m<\033[0m", "\033[0;32m>\033[0m", "\033[0;32m\" \033[0m",
00196     "\033[0;31m'\033[0m", "\033[0;34m*\033[0m", "\033[0;34m&\033[0m",
00197     "\033[0;33m|\033[0m", "\033[0;34m$\033[0m", "\033[0;32m:\033[0m",
00198     "\033[0;32m->\033[0m", "\033[0;33m#\033[0m", "\033[0;33m%\033[0m"
00199 };
00200 }
00201 #endif //SYNTAX_HPP

```

8.25 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.](#)

```

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

```

8.28 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.](#)

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

8.31 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.](#)

```

00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP
00007
00008 #include <windows.h>
00009 #include <iostream>
00010 #include <thread>
00011 #include <iomanip>
00012 #include "../Globals.hpp"
00013 #include <algorithm>
00014 #include <cctype>
00015
00021 namespace TextColors
00022 {
00023     static int BLUE = 1;
00024     static int GREEN = 2;
00025     static int LIGHTBLUE = 3;
00026     static int RED = 4;
00027     static int PURPLE = 5;
00028     static int YELLOW = 6;
00029     static int WHITE = 7;
00030     static int GREY = 8;
00031     static int BLUEBERRY = 9;
00032     static int LIGHTGREEN = 10;
00033     static int CYAN = 11;
00034     static int ROSE = 12;
00035     static int PINK = 13;
00036     static int BEIGE = 14;
00037 }
00038
00042 namespace TextFunctions{
00043
00048     static void changeTextColor(int color)
00049     {
00050         SetConsoleTextAttribute(cmd::hOutput, color);
00051     }
00052
00058     static void typeWriteMessage(std::string& s, int time)
00059     {
00060         for (const auto c : s) {
00061             std::cout << c << std::flush;
00062             std::this_thread::sleep_for(std::chrono::milliseconds(time));
00063         }
00064         printf("\n");
00065     }
00066
00071     static void print(std::string& message)
00072     {
00073         std::cout<<message<<std::endl;
00074     }
00075
00082     static bool setCursor(short x, short y)
00083     {
00084         return SetConsoleCursorPosition(cmd::hOutput, {x, y});
00085     }
00086     static COORD GetConsoleCursorPosition(HANDLE hConsoleOutput)
00087     {
00088         CONSOLE_SCREEN_BUFFER_INFO cbsi;
00089         if (GetConsoleScreenBufferInfo(hConsoleOutput, &cbsi))
00090         {
00091             return cbsi.dwCursorPosition;
00092         }
00093         else
00094         {
00095             // The function failed. Call GetLastError() for details.
00096             COORD invalid = { 0, 0 };
00097             return invalid;

```

```

00098     }
00099     }
00105     static std::string toLower(std::string data) {
00106
00107         std::transform(data.begin(), data.end(), data.begin(),
00108             [](unsigned char c){ return std::tolower(c); });
00109
00110         return data;
00111     }
00112
00113
00114 }
00115
00116
00117
00118 #endif //INC_2024__TAB_DSA__8_BRODZIAK_TEXTFORMATTER_HPP

```

8.33 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

<code>argv</code>	name of parameter, it has to be <code>-help</code> or <code>-h</code> 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.](#)

```
00001 //
00002 // Created by Michin on 01.05.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP
00007
00008
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012 #include "../Logic/PromptSingleton.hpp"
00013
00014 #include <iostream>
00015 #include <string>
00016 #include <vector>
00017
00021 class StateAbout : public State<States> {
00022     PromptSingleton* prompt = PromptSingleton::GetInstance();
00023     std::vector<std::string> dict = {
00024         "return"
00025     };
00026 public:
```

```

00027     explicit StateAbout (FiniteStateMachine<States>& fsm)
00028         : State<States>(fsm, States::ABOUT, "ABOUT") {}
00029
00030     void OnEnter() override;
00031     void OnUpdate() override;
00032     void OnExit() override;
00033 };
00034
00035
00036 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEABOUT_HPP

```

8.38 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.](#)

```

00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEEXIT_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/PromptSingleton.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015
00016 class StateExit : public State<States> {
00017     PromptSingleton* prompt = PromptSingleton::GetInstance();
00018 public:
00019     explicit StateExit (FiniteStateMachine<States>& fsm)
00020         : State<States>(fsm, States::EXIT, "EXIT") {}
00021
00022     void OnEnter() override;
00023     void OnUpdate() override;
00024     void OnExit() override;
00025 };
00026
00027
00028 #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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEFAVOURITES_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/Database/DBmanager.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016
00017 class StateFavourites : public State<States> {
00018     PromptSingleton* prompt = PromptSingleton::GetInstance();
00019     std::vector<std::string> dict = {
00020         "return"
00021     };
00022
00023     DBmanager db;
00024     std::vector<int> indexes;
00025     std::vector<std::pair<std::string, std::string>> data;
00026     std::vector<std::string> trimmedData;
00027
00028     void ManageData();
00029     int CheckFav(std::string);
00030 public:
00031     explicit StateFavourites(FiniteStateMachine<States>& fsm)
00032         : State<States>(fsm, States::FAVOURITES, "FAVOURITES") {}
00033
00034     void OnEnter() override;
00035     void OnUpdate() override;
00036     void OnExit() override;
00037 };
00038
00039 #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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/Database/DBmanager.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016
00017 class StateHistory : public State<States> {
00018     PromptSingleton* prompt = PromptSingleton::GetInstance();
00019     std::vector<std::string> dict = {
00020         "return"
00021     };
00022     DBmanager db;
00023     std::vector<std::string> trimmedData;
00024
00025     void ManageData();
00026     int CheckFav(std::string);
00027
00028 public:
00029     explicit StateHistory(FiniteStateMachine<States>& fsm)
00030         : State<States>(fsm, States::HISTORY, "HISTORY") {}
00031
00032     void OnEnter() override;
00033     void OnUpdate() override;
00034     void OnExit() override;
00035 };
00036
00037 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEHISTORY_HPP
```

8.47 States/StatIdle.cpp File Reference

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

8.48 States/StatIdle.hpp File Reference

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

Classes

- class [StatIdle](#)

8.49 StatIdle.hpp

[Go to the documentation of this file.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00007
00008 #include <iostream>
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012 #include "../Logic/PromptSingleton.hpp"
00013 #include <vector>
00014
00015
00016 class StateIdle : public State<States>{
00017     PromptSingleton* prompt = PromptSingleton::GetInstance();
00018     std::vector<std::string> dict = {
00019         "login",
00020         "register",
00021         "about"
00022     };
00023 public:
00024     explicit StateIdle(FiniteStateMachine<States>& fsm)
00025         : State<States>(fsm, States::IDLE, "IDLE") {}
00026
00027     void OnEnter() override;
00028     void OnUpdate() override;
00029     void OnExit() override;
00030 };
00031
00032 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEIDLE_HPP
00033
```

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.](#)

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


8.53 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.](#)

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

8.56 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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011 #include "../Logic/PromptSingleton.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include <vector>
00016
00021 class StateMenu : public State<States> {
00022     PromptSingleton* prompt = PromptSingleton::GetInstance();
00023     std::vector<std::string> dict = {
00024         "question",
00025         "history",
00026         "tags",
00027         "favourites"
00028     };
00029
00030 public:
00031     explicit StateMenu(FiniteStateMachine<States>& fsm)
00032         : State<States>(fsm, States::MENU, "MENU") {}
00033
00034     void OnEnter() override;
00035     void OnUpdate() override;
00036     void OnExit() override;
00037 };
00038
00039
00040 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEMENU_HPP
```

8.59 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.](#)

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

8.62 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.](#)

```
00001 //
00002 // Created by Michin on 21.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATEREREGISTER_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATEREREGISTER_HPP
00007
00008 #include "StatesConf.hpp"
00009 #include "../FSM/StateMachine.hpp"
00010 #include "../FSM/State.hpp"
00011
00012 #include <iostream>
00013 #include <string>
00014 #include "../Logic/PromptSingleton.hpp"
00015
00016 class StateRegister : public State<States> {
00017     PromptSingleton* prompt = PromptSingleton::GetInstance();
00018     std::string log;
00019     std::string pass;
00020     std::string email;
00021 public:
00022     explicit StateRegister(FiniteStateMachine<States>& fsm)
00023         : State<States>(fsm, States::REGISTER, "REGISTER") {}
00024
00025     void OnEnter() override;
00026     void OnUpdate() override;
00027     void OnExit() override;
00028 };
00029
00030 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATEREREGISTER_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.](#)

```
00001 //
00002 // Created by Michin on 24.04.2024.
00003 //
00004
00005 #ifndef INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
00006 #define INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
00007
00008
00009 #include "StatesConf.hpp"
00010 #include "../FSM/StateMachine.hpp"
00011 #include "../FSM/State.hpp"
00012
00013 #include <iostream>
00014 #include <string>
00015 #include "../Logic/PromptSingleton.hpp"
00016 #include "../Logic/StackApi/StackManager.hpp"
00017 #include "../Logic/StackApi/SyntaxHighlighting.hpp"
00018
00023 class StateResult : public State<States> {
00024     PromptSingleton* prompt = PromptSingleton::GetInstance();
00025     std::string question;
00026     std::string answer;
00027     StackManager sm = StackManager();
00028     SyntaxHighlighting sh = SyntaxHighlighting();
00029     std::vector<std::string> dict = {
00030         "question",
00031         "return"
00032     };
00033 public:
00034     explicit StateResult(FiniteStateMachine<States>& fsm)
00035         : State<States>(fsm, States::RESULT, "RESULT") {}
00036
00037     void OnEnter() override;
00038     void OnUpdate() override;
00039     void OnExit() override;
00040
00041     void QuestionManage();
00042 };
00043
00044
00045 #endif //INC_2024__TAB_DSA__8_BRODZIAK_STATERESULT_HPP
```

8.68 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.](#)

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

8.71 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.](#)

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

8.73 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.](#)

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

8.75 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.](#)

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

8.78 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](#)

Generated by Doxygen

[illegible]

```
00236         "type return to go back,\n"
00237         "type number of question to move to this question,\n"
00238         "type f$ where $ is number of question to add it to favourites\n"
00239         "\n"
00240         "Favourites:\n"
00241         "lists your favourites questions,\n"
00242         "type return to go back,\n"
00243         "type number of question to move to this question,\n"
00244         "type d$ where $ is number of question to delete it from favourites\n"
00245         "\n"
00246         "Result:\n"
00247         "you can move to result from different states,\n"
00248         "from question by prompting question,\n"
00249         "from list of tags, history or favourites by choosing number from
list\n"
00250         "it always gives back question with max 3 top rated answers\n"
00251         "and possibility to type return to go back to previous state\n"
00252         "\n"
00253         "MOST OF THE STATES HAVE AUTOCOMPLETE SO YOU CAN TYPE\n"
00254         "R INSTEAD OF RETURN TO EXECUTE DESIRED COMMAND";
00255
00256     static std::string exit = "\nPress any key + enter to exit\n";
00257     static std::string help = "\nYou can also double check commands in the new file HELP.txt which
just got created"
00258         "in the exe directory\n";
00259 }
00260
00261
00262 #endif //INC_2024__TAB_DSA__8_BRODZIAK_ALLTEXTS_HPP
```

Index

- ~DBmanager
 - DBmanager, [17](#)
- ~State
 - State< T >, [34](#)
- ABOUT
 - StatesConf.hpp, [91](#)
- AboutTexts, [11](#)
- Add
 - FiniteStateMachine< T >, [21](#)
- AskQuestion
 - StackManager, [31](#)
- bestAnswer
 - StackManager, [33](#)
- ChangeJsonToString
 - StackManager, [31](#)
- ChangingSpecialChar
 - StackManager, [31](#)
- checkTagQuestionList
 - StackManager, [31](#)
- ChoosingTitle
 - StateListTags, [46](#)
- cmd, [11](#)
- ColorBracket
 - SyntaxHighlighting, [60](#)
- ColorChar
 - SyntaxHighlighting, [60](#)
- conanfile, [11](#)
- conanfile.ConanApplication, [15](#)
 - generate, [15](#)
 - generators, [16](#)
 - layout, [15](#)
 - package_type, [16](#)
 - requirements, [15](#)
 - settings, [16](#)
- conanfile.py, [63](#)
- connectTagToPhrase
 - DBmanager, [17](#)
 - QueryHelper, [26](#)
- createAdminTable
 - QueryHelper, [26](#)
- createPhraseTable
 - QueryHelper, [26](#)
- createPhraseTagTable
 - QueryHelper, [27](#)
- createTagTable
 - QueryHelper, [27](#)
- createUserTable
 - QueryHelper, [27](#)
- DBmanager, [16](#)
 - ~DBmanager, [17](#)
 - connectTagToPhrase, [17](#)
 - DBmanager, [17](#)
 - deleteAdmin, [17](#)
 - deleteFavourite, [17](#)
 - deletePhrase, [17](#)
 - deleteTag, [17](#)
 - deleteUser, [17](#)
 - getAdmins, [18](#)
 - getFavourites, [18](#)
 - getPhrase, [18](#)
 - getPhrases, [18](#)
 - getPhraseWithTag, [18](#)
 - getTags, [18](#)
 - getUsers, [18](#)
 - insertAdmin, [18](#)
 - insertFavourite, [18](#)
 - insertPhrase, [18](#)
 - insertTag, [19](#)
 - insertUser, [19](#)
 - loginUser, [19](#)
 - updateUserPassword, [19](#)
- DBmanager.cpp
 - receivedData, [67](#)
 - sqlite3_callback, [67](#)
- deleteAdmin
 - DBmanager, [17](#)
 - QueryHelper, [27](#)
- deleteFavourite
 - DBmanager, [17](#)
 - QueryHelper, [27](#)
- deletePhrase
 - DBmanager, [17](#)
 - QueryHelper, [27](#)
- deleteTag
 - DBmanager, [17](#)
 - QueryHelper, [27](#)
- deleteUser
 - DBmanager, [17](#)
 - QueryHelper, [27](#)
- Engine, [19](#)
 - Engine, [20](#)
 - Run, [20](#)
- Engine.cpp, [63](#)
- Engine.hpp, [63](#)
- EXIT

- StatesConf.hpp, 91
- FAVOURITES
 - StatesConf.hpp, 91
- FavouriteTexts, 11
- FillTabel
 - StackManager, 31
- FiniteStateMachine
 - FiniteStateMachine< T >, 21
- FiniteStateMachine< T >, 20
 - Add, 21
 - FiniteStateMachine, 21
 - GetCurrentState, 22
 - GetState, 22
 - mCurrentState, 23
 - mStates, 23
 - OnUpdate, 22
 - SetCurrentState, 23
- FSM/State.hpp, 64
- FSM/StateMachine.hpp, 65
- generate
 - conanfile.ConanApplication, 15
- generators
 - conanfile.ConanApplication, 16
- getAdmins
 - DBmanager, 18
 - QueryHelper, 27
- GetAnswer
 - StackManager, 31
- GetCurrentState
 - FiniteStateMachine< T >, 22
- getFavourites
 - DBmanager, 18
 - QueryHelper, 28
- GetID
 - TagsList, 62
- getID
 - State< T >, 35
- GetInstance
 - PromptSingleton, 24
- GetMatch
 - PromptSingleton.cpp, 70
- GetName
 - State< T >, 35
- getPhrase
 - DBmanager, 18
 - QueryHelper, 28
- getPhrases
 - DBmanager, 18
 - QueryHelper, 28
- getPhrasesWithTag
 - QueryHelper, 28
- getPhraseWithTag
 - DBmanager, 18
- GetPrompt
 - PromptSingleton, 24
- GetPromptAuto
 - PromptSingleton, 25
- GetQuestionFromID
 - StackManager, 31
- GetQuestionId
 - StackManager, 32
- getQuestionList
 - StackManager, 32
- GetState
 - FiniteStateMachine< T >, 22
- getTags
 - DBmanager, 18
 - QueryHelper, 28
- GetTitle
 - StackManager, 32
 - TagsList, 62
- getUsers
 - DBmanager, 18
 - QueryHelper, 28
- Globals.hpp, 66
- Hightlighting
 - SyntaxHighlighting, 60
- HISTORY
 - StatesConf.hpp, 91
- HistoryTexts, 12
- IDLE
 - StatesConf.hpp, 91
- IdleTexts, 12
- insertAdmin
 - DBmanager, 18
 - QueryHelper, 28
- insertFavourite
 - DBmanager, 18
 - QueryHelper, 28
- insertPhrase
 - DBmanager, 18
 - QueryHelper, 28
- insertTag
 - DBmanager, 19
 - QueryHelper, 29
- insertUser
 - DBmanager, 19
 - QueryHelper, 29
- layout
 - conanfile.ConanApplication, 15
- ListState, 12
- LISTTAGS
 - StatesConf.hpp, 91
- Logic/Database/DBmanager.cpp, 67
- Logic/Database/DBmanager.hpp, 67, 68
- Logic/Database/QueryHelper.cpp, 68
- Logic/Database/QueryHelper.hpp, 69
- Logic/PromptSingleton.cpp, 69
- Logic/PromptSingleton.hpp, 70
- Logic/StackApi/StackManager.cpp, 71
- Logic/StackApi/StackManager.hpp, 71
- Logic/StackApi/Syntax.hpp, 72
- Logic/StackApi/SyntaxHighlighting.cpp, 75

- Logic/StackApi/SyntaxHighlighting.hpp, 75
- Logic/TagList/TagsList.cpp, 76
- Logic/TagList/TagsList.hpp, 76
- Logic/TextFormatter.hpp, 76, 77
- LOGIN
 - StatesConf.hpp, 91
- LoginTexts, 12
- loginUser
 - DBmanager, 19
 - QueryHelper, 29
- LookForByTags
 - StackManager, 32
- main
 - main.cpp, 78
- main.cpp, 78
 - main, 78
 - PrintHelp, 78
- ManageList
 - StateListTags, 46
- Manual, 13
- mCurrentState
 - FiniteStateMachine< T >, 23
- MENU
 - StatesConf.hpp, 91
- MenuTexts, 13
- mFsm
 - State< T >, 36
- mID
 - State< T >, 36
- mName
 - State< T >, 36
- mStates
 - FiniteStateMachine< T >, 23
- OnEnter
 - State< T >, 35
 - StateAbout, 37
 - StateExit, 39
 - StateFavourites, 41
 - StateHistory, 42
 - StateIdle, 44
 - StateListTags, 46
 - StateLogin, 48
 - StateMenu, 50
 - StatePrompt, 52
 - StateRegister, 53
 - StateResult, 55
 - StateResultTags, 57
 - StateTags, 59
- OnExit
 - State< T >, 35
 - StateAbout, 37
 - StateExit, 39
 - StateFavourites, 41
 - StateHistory, 42
 - StateIdle, 44
 - StateListTags, 47
 - StateLogin, 48
- StateMenu, 50
- StatePrompt, 52
- StateRegister, 53
- StateResult, 55
- StateResultTags, 57
- StateTags, 59
- OnUpdate
 - FiniteStateMachine< T >, 22
 - State< T >, 35
 - StateAbout, 38
 - StateExit, 39
 - StateFavourites, 41
 - StateHistory, 43
 - StateIdle, 45
 - StateListTags, 47
 - StateLogin, 49
 - StateMenu, 50
 - StatePrompt, 52
 - StateRegister, 54
 - StateResult, 55
 - StateResultTags, 57
 - StateTags, 59
- package_type
 - conanfile.ConanApplication, 16
- PrintHelp
 - main.cpp, 78
- PROMPT
 - StatesConf.hpp, 91
- PromptSingleton, 24
 - GetInstance, 24
 - GetPrompt, 24
 - GetPromptAuto, 25
 - PromptSingleton, 24
 - RetValues, 25
 - SetValues, 25
- PromptSingleton.cpp
 - GetMatch, 70
- PromptTexts, 13
- QueryHelper, 26
 - connectTagToPhrase, 26
 - createAdminTable, 26
 - createPhraseTable, 26
 - createPhraseTagTable, 27
 - createTagTable, 27
 - createUserTable, 27
 - deleteAdmin, 27
 - deleteFavourite, 27
 - deletePhrase, 27
 - deleteTag, 27
 - deleteUser, 27
 - getAdmins, 27
 - getFavourites, 28
 - getPhrase, 28
 - getPhrases, 28
 - getPhrasesWithTag, 28
 - getTags, 28
 - getUsers, 28

- insertAdmin, [28](#)
- insertFavourite, [28](#)
- insertPhrase, [28](#)
- insertTag, [29](#)
- insertUser, [29](#)
- loginUser, [29](#)
- updateUserPass, [29](#)
- QuestionManage
 - StateResult, [55](#)
 - StateResultTags, [57](#)
- README, [1](#)
- README.md, [79](#)
- receivedData
 - DBmanager.cpp, [67](#)
- RecognizeSyntax
 - SyntaxHighlighting, [61](#)
- REGISTER
 - StatesConf.hpp, [91](#)
- RegisterTexts, [13](#)
- RemoveHtmlTags
 - StackManager, [32](#)
- RemoveTags
 - SyntaxHighlighting, [61](#)
- requirements
 - conanfile.ConanApplication, [15](#)
- RESULT
 - StatesConf.hpp, [91](#)
- RESULTTAGS
 - StatesConf.hpp, [91](#)
- ResultTexts, [13](#)
- ReturnNiceCode
 - StackManager, [32](#)
- RetValues
 - PromptSingleton, [25](#)
- Run
 - Engine, [20](#)
- SetCurrentState
 - FiniteStateMachine< T >, [23](#)
- SetQuestion
 - StackManager, [32](#)
- SetQuestionByTags
 - StackManager, [33](#)
- SetQuestionId
 - StackManager, [33](#)
- settings
 - conanfile.ConanApplication, [16](#)
- SetValues
 - PromptSingleton, [25](#)
- sqlite3_callback
 - DBmanager.cpp, [67](#)
- StackManager, [29](#)
 - AskQuestion, [31](#)
 - bestAnswer, [33](#)
 - ChangeJsonToString, [31](#)
 - ChangingSpecialChar, [31](#)
 - checkTagQuestionList, [31](#)
 - FillTabel, [31](#)
 - GetAnswer, [31](#)
 - GetQuestionFromID, [31](#)
 - GetQuestionId, [32](#)
 - getQuestionList, [32](#)
 - GetTitle, [32](#)
 - LookForByTags, [32](#)
 - RemoveHtmlTags, [32](#)
 - ReturnNiceCode, [32](#)
 - SetQuestion, [32](#)
 - SetQuestionByTags, [33](#)
 - SetQuestionId, [33](#)
- State
 - State< T >, [34](#)
- State< T >, [33](#)
 - ~State, [34](#)
 - getID, [35](#)
 - GetName, [35](#)
 - mFsm, [36](#)
 - mID, [36](#)
 - mName, [36](#)
 - OnEnter, [35](#)
 - OnExit, [35](#)
 - OnUpdate, [35](#)
 - State, [34](#)
- StateAbout, [36](#)
 - OnEnter, [37](#)
 - OnExit, [37](#)
 - OnUpdate, [38](#)
 - StateAbout, [37](#)
- StateExit, [38](#)
 - OnEnter, [39](#)
 - OnExit, [39](#)
 - OnUpdate, [39](#)
 - StateExit, [39](#)
- StateFavourites, [40](#)
 - OnEnter, [41](#)
 - OnExit, [41](#)
 - OnUpdate, [41](#)
 - StateFavourites, [41](#)
- StateHistory, [41](#)
 - OnEnter, [42](#)
 - OnExit, [42](#)
 - OnUpdate, [43](#)
 - StateHistory, [42](#)
- Stateldle, [43](#)
 - OnEnter, [44](#)
 - OnExit, [44](#)
 - OnUpdate, [45](#)
 - Stateldle, [44](#)
- StateListTags, [45](#)
 - ChoosingTitle, [46](#)
 - ManageList, [46](#)
 - OnEnter, [46](#)
 - OnExit, [47](#)
 - OnUpdate, [47](#)
 - StateListTags, [46](#)
- StateLogin, [47](#)
 - OnEnter, [48](#)

- OnExit, [48](#)
- OnUpdate, [49](#)
- StateLogin, [48](#)
- StateMenu, [49](#)
 - OnEnter, [50](#)
 - OnExit, [50](#)
 - OnUpdate, [50](#)
 - StateMenu, [50](#)
- StatePrompt, [51](#)
 - OnEnter, [52](#)
 - OnExit, [52](#)
 - OnUpdate, [52](#)
 - StatePrompt, [52](#)
- StateRegister, [52](#)
 - OnEnter, [53](#)
 - OnExit, [53](#)
 - OnUpdate, [54](#)
 - StateRegister, [53](#)
- StateResult, [54](#)
 - OnEnter, [55](#)
 - OnExit, [55](#)
 - OnUpdate, [55](#)
 - QuestionManage, [55](#)
 - StateResult, [55](#)
- StateResultTags, [56](#)
 - OnEnter, [57](#)
 - OnExit, [57](#)
 - OnUpdate, [57](#)
 - QuestionManage, [57](#)
 - StateResultTags, [57](#)
- States
 - StatesConf.hpp, [91](#)
 - States/StateAbout.cpp, [79](#)
 - States/StateAbout.hpp, [79](#)
 - States/StateExit.cpp, [80](#)
 - States/StateExit.hpp, [80](#)
 - States/StateFavourites.cpp, [81](#)
 - States/StateFavourites.hpp, [81](#)
 - States/StateHistory.cpp, [82](#)
 - States/StateHistory.hpp, [82](#)
 - States/StateIdle.cpp, [83](#)
 - States/StateIdle.hpp, [83](#)
 - States/StateListTags.cpp, [84](#)
 - States/StateListTags.hpp, [84](#)
 - States/StateLogin.cpp, [85](#)
 - States/StateLogin.hpp, [85](#)
 - States/StateMenu.cpp, [85](#)
 - States/StateMenu.hpp, [86](#)
 - States/StatePrompt.cpp, [86](#)
 - States/StatePrompt.hpp, [87](#)
 - States/StateRegister.cpp, [87](#)
 - States/StateRegister.hpp, [88](#)
 - States/StateResult.cpp, [88](#)
 - States/StateResult.hpp, [89](#)
 - States/StateResultTags.cpp, [89](#)
 - States/StateResultTags.hpp, [90](#)
 - States/StatesConf.hpp, [90, 91](#)
 - States/StatesWrapper.hpp, [91, 92](#)
 - States/StateTags.cpp, [92](#)
 - States/StateTags.hpp, [92, 93](#)
 - StatesConf.hpp
 - ABOUT, [91](#)
 - EXIT, [91](#)
 - FAVOURITES, [91](#)
 - HISTORY, [91](#)
 - IDLE, [91](#)
 - LISTTAGS, [91](#)
 - LOGIN, [91](#)
 - MENU, [91](#)
 - PROMPT, [91](#)
 - REGISTER, [91](#)
 - RESULT, [91](#)
 - RESULTTAGS, [91](#)
 - States, [91](#)
 - TAGS, [91](#)
 - StateTags, [58](#)
 - OnEnter, [59](#)
 - OnExit, [59](#)
 - OnUpdate, [59](#)
 - StateTags, [59](#)
 - Syntax, [14](#)
 - SyntaxHighlighting, [59](#)
 - ColorBracket, [60](#)
 - ColorChar, [60](#)
 - Hightlighting, [60](#)
 - RecognizeSyntax, [61](#)
 - RemoveTags, [61](#)
 - SyntaxHighlighting, [60](#)
 - TAGS
 - StatesConf.hpp, [91](#)
 - TagsList, [61](#)
 - GetID, [62](#)
 - GetTitle, [62](#)
 - TagsList, [61](#)
 - TagsTexts, [14](#)
 - TextColors, [14](#)
 - TextFunctions, [14](#)
 - Texts/AllTexts.hpp, [93, 94](#)
 - updateUserPass
 - QueryHelper, [29](#)
 - updateUserPassword
 - DBmanager, [19](#)