2D Graphics Modeler - Team Alphawolves

Generated by Doxygen 1.13.2

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Topic Documentation

6.1 > Shapes

Global constant PI used for calculating perimater and area.

Global constant PI used for calculating perimater and area.

Chapter 7

Namespace Documentation

7.1 alpha Namespace Reference

Classes

- class vector

7.2 QT_WARNING_DISABLE_DEPRECATED Namespace Reference

Classes

- struct qt_meta_tag_ZN11UserManagerE_t
- struct qt_meta_tag_ZN9ApiClientE_t

7.3 Ui Namespace Reference

Chapter 8

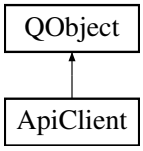
Class Documentation

8.1 ApiClient Class Reference

Documentation Qt Version 6.9.0.

`#include <ApiClient.h>`

Inheritance diagram for ApiClient:



Signals

API Reply Signals

Signals emitted by ApiClient to communicate request outcomes.

These signals notify listeners about the success or failure of GET, POST, and DELETE operations.

- void GoodGetReply (const QString &json)
  Signal for a successful request sent to a Get endpoint.
- void BadGetReply (const QString &error)
  Signal for a failed request sent to a Get endpoint.
- void GoodPostReply ()
  Signal for a successful request sent to a Post endpoint.
- void BadPostReply (const QString &error)
  Signal for a failed request sent to a Post endpoint.
- void GoodDeleteReply ()
  Signal for a successful request sent to a Delete endpoint.
- void BadDeleteReply (const QString &error)
  Signal for a failed request sent to a Delete endpoint.

Public Member Functions

Constructor and Core API Functions

Public methods for ApiClient.

This section includes the constructor and functions for making API requests.

- ApiClient (QObject *parent=nullptr)
  Default Constructor.

Shape Data Management

API endpoints for managing Shape data.

These functions interact with the `/shapes` and `/shapes-all` endpoints to retrieve, submit, or delete shape information.

Note

These endpoints are deprecated

- void GetShapes ()
  Makes a request to get data from the Get /shapes API Endpoint.
- void PostShapes (std::string json)
  Makes a request to send data to the Post /shapes API Endpoint.
- void DeleteShapesAll ()
  Makes a request to delete all shapes via the Delete /shapes-all endpoint.

Render Area Data Management

API endpoints for managing Render Area data.

These functions interact with the `/render_area` and `/render_area-all` endpoints to retrieve, submit, or delete render area configurations.

- void GetRenderArea ()
  Makes a request to get data from the Get /render_area API Endpoint.
- void PostRenderArea (std::string json)
  Makes a request to send data to the Post /render_area API Endpoint.
- void DeleteRenderAreaAll ()
  Makes a request to delete all render area data via Delete /render_area-all.

User Data Management

API endpoints for managing User data.

These functions interact with the `/users` and `/users-all` endpoints to retrieve, submit, or delete user information.

- void GetUsers ()
  Makes a request to get data from the Get /users API Endpoint.
- void PostUsers (std::string json)
  Makes a request to send data to the Post /users API Endpoint.
- void DeleteUsersAll ()
  Makes a request to delete all user data via Delete /users-all.

Testimonial Data Management

API endpoints for managing Testimonial data.

These functions interact with the `/testimonials` endpoint to retrieve, submit, or delete testimonial entries.

- void GetTestimonials ()
  Makes a request to get all testimonials via GET /testimonials.
- void PostTestimonials (std::string json)
  Makes a request to post testimonial data via POST /testimonials.
- void DeleteTestimonialsAll ()
  Makes a request to delete all testimonials via DELETE /testimonials.

Private Slots

Network Reply Analysis

These slots are connected to the finished signal of QNetworkReply and determine whether the operation was successful, emitting the appropriate Good/Bad Get/Post/Delete reply signals.

- void AnalyzeGetReply ()
  Analyzes the reply for GET requests.
- void AnalyzePostReply ()
  Analyzes the reply for POST requests.
- void AnalyzeDeleteReply ()
  Analyzes the reply for DELETE requests.

Private Attributes

- QNetworkAccessManager * manager
  Manages all network operations for the client.

8.1.1 Detailed Description

Documentation Qt Version 6.9.0.

Manages network communication with the backend API.

This class provides methods to perform GET, POST, and DELETE requests to various endpoints of the webservice. It uses QNetworkAccessManager for handling network operations and emits signals to indicate the success or failure of these requests.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 ApiClient()

```
ApiClient::ApiClient (
            QObject * parent = nullptr)  [explicit]
```

Default Constructor.

8.1.3 Member Function Documentation

8.1.3.1 AnalyzeDeleteReply

```
void ApiClient::AnalyzeDeleteReply ()  [private], [slot]
```

Analyzes the reply for DELETE requests.

8.1.3.2 AnalyzeGetReply

```
void ApiClient::AnalyzeGetReply ()  [private], [slot]
```

Analyzes the reply for GET requests.

8.1.3.3 AnalyzePostReply

```
void ApiClient::AnalyzePostReply ()  [private], [slot]
```

Analyzes the reply for POST requests.

8.1.3.4 BadDeleteReply

```
void ApiClient::BadDeleteReply (
            const QString & error)  [signal]
```

Signal for a failed request sent to a Delete endpoint.

8.1.3.5 BadGetReply

```
void ApiClient::BadGetReply (
            const QString & error)  [signal]
```

Signal for a failed request sent to a Get endpoint.

8.1.3.6 BadPostReply

```
void ApiClient::BadPostReply (
            const QString & error)  [signal]
```

Signal for a failed request sent to a Post endpoint.

8.1.3.7 DeleteRenderAreaAll()

```
void ApiClient::DeleteRenderAreaAll ()
```

Makes a request to delete all render area data via Delete /render_area-all.

8.1.3.8 DeleteShapesAll()

```
void ApiClient::DeleteShapesAll ()
```

Makes a request to delete all shapes via the Delete /shapes-all endpoint.

8.1.3.9 DeleteTestimonialsAll()

```
void ApiClient::DeleteTestimonialsAll ()
```

Makes a request to delete all testimonials via DELETE /testimonials.

8.1.3.10 DeleteUsersAll()

```
void ApiClient::DeleteUsersAll ()
```

Makes a request to delete all user data via Delete /users-all.

8.1.3.11 GetRenderArea()

```
void ApiClient::GetRenderArea ()
```

Makes a request to get data from the Get /render_area API Endpoint.

8.1.3.12 GetShapes()

```
void ApiClient::GetShapes ()
```

Makes a request to get data from the Get /shapes API Endpoint.

8.1.3.13 GetTestimonials()

```
void ApiClient::GetTestimonials ()
```

Makes a request to get all testimonials via GET /testimonials.

8.1.3.14 GetUsers()

```
void ApiClient::GetUsers ()
```

Makes a request to get data from the Get /users API Endpoint.

8.1.3.15 GoodDeleteReply

```
void ApiClient::GoodDeleteReply ()  [signal]
```

Signal for a successful request sent to a Delete endpoint.

8.1.3.16 GoodGetReply

```
void ApiClient::GoodGetReply (
            const QString & json)  [signal]
```

Signal for a successful request sent to a Get endpoint.

8.1.3.17 GoodPostReply

```
void ApiClient::GoodPostReply ()  [signal]
```

Signal for a successful request sent to a Post endpoint.

8.1.3.18 PostRenderArea()

```
void ApiClient::PostRenderArea (
            std::string json)
```

Makes a request to send data to the Post /render_area API Endpoint.

8.1.3.19 PostShapes()

```
void ApiClient::PostShapes (
            std::string json)
```

Makes a request to send data to the Post /shapes API Endpoint.

8.1.3.20 PostTestimonials()

```
void ApiClient::PostTestimonials (
            std::string json)
```

Makes a request to post testimonial data via POST /testimonials.

8.1.3.21 PostUsers()

```
void ApiClient::PostUsers (
            std::string json)
```

Makes a request to send data to the Post /users API Endpoint.

8.1.4 Member Data Documentation

8.1.4.1 manager

```
QNetworkAccessManager* ApiClient::manager  [private]
```

Manages all network operations for the client.

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

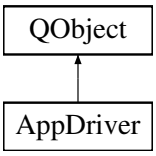
- src/backend/ApiClient.h
- src/backend/ApiClient.cpp
- src/backend/moc_ApiClient.cpp

8.2 AppDriver Class Reference

Orchestrates the main application logic and connections.

`#include <AppDriver.h>`

Inheritance diagram for AppDriver:



Public Member Functions

Core Application Lifecycle

Public interface for the AppDriver.

This section includes the constructor, destructor, and core methods for running, shutting down, and loading data for the application.

- AppDriver (QObject *parent=nullptr)
  Constructs the AppDriver and initializes manager objects.
- ~AppDriver ()
  Destroys the AppDriver and cleans up manager objects.
- void run ()
  Initializes and shows the main application window and loads initial data.
- void shutdown ()
  Saves data from managers before the application closes.
- void loadAllData ()
  Triggers the loading of all necessary data by the managers.

Private Slots

Render Area UI Slots

Slots connecting the render area UI to RenderAreaManager data.

These slots handle signals from the frontend render area, such as shape additions, modifications, and deletions, and delegate these actions to the RenderAreaManager.

- void onRenderShapeAdded (Shape *shape)
  Handles the addition of a new shape from the render area.
- void onRenderShapeChanged (Shape *shape, QString key, int value)
- void onRenderShapeDeleted (const int trackerId)
  Handles the deletion of a single shape from the render area.
- void onRenderDeleteAllShapes ()
  Handles the request to delete all shapes from the render area.

User Management UI Slots

Slots connecting the user login UI to UserManager data.

These slots handle signals related to user management from the frontend, such as new user creation, credential modification, user deletion, and login attempts, delegating these actions to the UserManager.

- void onNewUser (const QString username, const QString password, const bool admin)
- void onUserModified (const QString username, const QString password, const bool admin)
- void onUserDeleted (const QString username)
  Handles the deletion of a single user.
- void onDeleteAllUsers ()
  Handles the request to delete all users.
- void onLoginAttempt (const QString username, const QString password)

Private Member Functions

Connection Setup Helpers

Private helper methods for setting up connections.

These subroutines are called during initialization to establish signal-slot connections between different components of the application.

- void connectFrontendToDriver ()
  Connects signals from the frontend UI to slots in this AppDriver.
- void connectManagersToFrontend ()
  Connects signals from backend managers to slots in the frontend UI.

Private Attributes

Core Components

Private member variables for the AppDriver.

These members hold pointers to the core manager classes and the main window.

- MainWindow * mainWindow
  Pointer to the main application window.
- RenderAreaManager * renderedShapes
  Manages all the shapes that are currently rendered.
- UserManager * user
  Manages the current user and holds all existing users to authenticate against.

8.2.1 Detailed Description

Orchestrates the main application logic and connections.

The AppDriver class is responsible for initializing and managing the main components of the application, including the RenderAreaManager, UserManager, and the MainWindow. It handles the setup of signal-slot connections between the frontend UI elements and the backend data managers. It also manages the application lifecycle, including startup, data loading, and shutdown procedures.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 AppDriver()

```
AppDriver::AppDriver (
            QObject * parent = nullptr)
```

Constructs the AppDriver and initializes manager objects.

8.2.2.2 ~AppDriver()

```
AppDriver::~AppDriver ()
```

Destroys the AppDriver and cleans up manager objects.

8.2.3 Member Function Documentation

8.2.3.1 connectFrontendToDriver()

```
void AppDriver::connectFrontendToDriver ()  [private]
```

Connects signals from the frontend UI to slots in this AppDriver.

8.2.3.2 connectManagersToFrontend()

```
void AppDriver::connectManagersToFrontend ()  [private]
```

Connects signals from backend managers to slots in the frontend UI.

8.2.3.3 loadAllData()

```
void AppDriver::loadAllData ()
```

Triggers the loading of all necessary data by the managers.

8.2.3.4 onDeleteAllUsers

```
void AppDriver::onDeleteAllUsers ()  [private], [slot]
```

Handles the request to delete all users.

8.2.3.5 onLoginAttempt

```
void AppDriver::onLoginAttempt (
            const QString username,
            const QString password)  [private], [slot]
```

Parameters

username	Handles a user login attempt.

8.2.3.6 onNewUser

```
void AppDriver::onNewUser (
            const QString username,
            const QString password,
            const bool admin)  [private], [slot]
```

Parameters

username	Handles the creation of a new user.

8.2.3.7 onRenderDeleteAllShapes

```
void AppDriver::onRenderDeleteAllShapes ()  [private], [slot]
```

Handles the request to delete all shapes from the render area.

8.2.3.8 onRenderShapeAdded

```
void AppDriver::onRenderShapeAdded (
            Shape * shape)  [private], [slot]
```

Handles the addition of a new shape from the render area.

8.2.3.9 onRenderShapeChanged

```
void AppDriver::onRenderShapeChanged (
            Shape * shape,
            QString key,
            int value)  [private], [slot]
```

Parameters

shape	Handles modifications to an existing shape from the render area.

8.2.3.10 onRenderShapeDeleted

```
void AppDriver::onRenderShapeDeleted (
            const int trackerId)  [private], [slot]
```

Handles the deletion of a single shape from the render area.

8.2.3.11 onUserDeleted

```
void AppDriver::onUserDeleted (
            const QString username)  [private], [slot]
```

Handles the deletion of a single user.

8.2.3.12 onUserModified

```
void AppDriver::onUserModified (
            const QString username,
            const QString password,
            const bool admin)  [private], [slot]
```

Parameters

username	Handles modification of an existing user's credentials.

8.2.3.13 run()

```
void AppDriver::run ()
```

Initializes and shows the main application window and loads initial data.

8.2.3.14 shutdown()

```
void AppDriver::shutdown ()
```

Saves data from managers before the application closes.

8.2.4 Member Data Documentation

8.2.4.1 mainWindow

```
MainWindow* AppDriver::mainWindow  [private]
```

Pointer to the main application window.

8.2.4.2 renderedShapes

```
RenderAreaManager* AppDriver::renderedShapes  [private]
```

Manages all the shapes that are currently rendered.

8.2.4.3 user

```
UserManager* AppDriver::user  [private]
```

Manages the current user and holds all existing users to authenticate against.

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

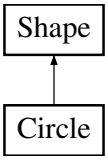
- src/backend/AppDriver.h
- src/backend/AppDriver.cpp

8.3 Circle Class Reference

The Circle class.

`#include "objects/circle.h"`

Inheritance diagram for Circle:



Public Member Functions

- Circle (string shapeType, QPoint coords, QPen pen, QBrush brush, int r)

  Circle Constructor.
- void Draw (QWidget ∗renderArea) override

  Draw - Draws a circle at the assigned coords with the given radius.
- double Perimeter () const override

  Perimeter - Returns the perimeter of the circle.
- double Area () const override

  Area - Returns the area of the circle.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the circle
- int getR () const

  getR - Accessor - Returns the radius
- void setR (int radius)

  Mutator Functions.
- void setX (int x)
- void setY (int y)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ∼Shape ()

  ∼Shape Destructor
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

Accessor Functions - Returns the data named after them.

- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- int r

  Mutator Functions.

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.3.1 Detailed Description

The Circle class.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Circle()

```
Circle::Circle (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            int r)
```

Circle Constructor.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
r	- int radius

8.3.3 Member Function Documentation

8.3.3.1 Area()

```
double Circle::Area () const  [override], [virtual]
```

Area - Returns the area of the circle.

Returns

Implements Shape.

8.3.3.2 Draw()

```
void Circle::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a circle at the assigned coords with the given radius.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.3.3.3 getR()

```
int Circle::getR () const
```

getR - Accessor - Returns the radius

Returns

8.3.3.4 isPointInside()

```
bool Circle::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the circle

Parameters

point	- point being read

Returns

Implements Shape.

8.3.3.5 Perimeter()

```
double Circle::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimeter of the circle.

Returns

Implements Shape.

8.3.3.6 setR()

```
void Circle::setR (
            int radius)
```

Mutator Functions.

8.3.3.7 setX()

```
void Circle::setX (
            int x)
```

Implements Shape.

8.3.3.8 setY()

```
void Circle::setY (
            int y)
```

Implements Shape.

8.3.4 Member Data Documentation

8.3.4.1 r

```
int Circle::r  [private]
```

Mutator Functions.

int radius of the circle

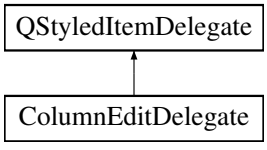
The documentation for this class was generated from the following files:

- src/objects/circle.h
- src/objects/circle.cpp

8.4 ColumnEditDelegate Class Reference

```
#include <ColumnEditDelegate.h>
```

Inheritance diagram for ColumnEditDelegate:



Public Member Functions

- ColumnEditDelegate (QObject ∗parent=nullptr)
- QWidget ∗ createEditor (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const override
- void setCanEdit (bool edit)

Private Attributes

- bool canEdit = false

8.4.1 Constructor & Destructor Documentation

8.4.1.1 ColumnEditDelegate()

```
ColumnEditDelegate::ColumnEditDelegate (
            QObject * parent = nullptr)  [inline]
```

8.4.2 Member Function Documentation

8.4.2.1 createEditor()

```
QWidget * ColumnEditDelegate::createEditor (
            QWidget * parent,
            const QStyleOptionViewItem & option,
            const QModelIndex & index) const  [inline], [override]
```

8.4.2.2 setCanEdit()

```
void ColumnEditDelegate::setCanEdit (
            bool edit)  [inline]
```

8.4.3 Member Data Documentation

8.4.3.1 canEdit

```
bool ColumnEditDelegate::canEdit = false  [private]
```

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

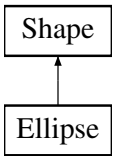
- src/frontend/ColumnEditDelegate.h

8.5 Ellipse Class Reference

The Ellipse class.

```
#include "objects/ellipse.h"
```

Inheritance diagram for Ellipse:



Public Member Functions

- Ellipse (string shapeType, QPoint coords, QPen pen, QBrush brush, int a, int b)

  Ellipse Constructor.
- void Draw (QWidget ∗renderArea) override

  Draw - Draws the ellipse to the passed renderArea.
- double Perimeter () const override

  Perimeter - Returns the perimeter of the ellipse.
- double Area () const override

  Area - Returns the area of the ellipse.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the circle
- int getA () const

  Accessor Functions.
- int getB () const
- void setA (int newA)

  Accessor Functions.
- void setB (int newB)
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ∼Shape ()

  ∼*Shape Destructor*
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.

- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- int a

  Mutator Functions.
- int b

  Semi-Major Axis.

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.5.1 Detailed Description

The Ellipse class.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Ellipse()

```
Ellipse::Ellipse (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            int a,
            int b)
```

Ellipse Constructor.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
a	- The Semi-Minor axis of the Ellipse
b	- The Semi-Major axis of the Ellipse

8.5.3 Member Function Documentation

8.5.3.1 Area()

```
double Ellipse::Area () const  [override], [virtual]
```

Area - Returns the area of the ellipse.

Returns

Implements Shape.

8.5.3.2 Draw()

```
void Ellipse::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws the ellipse to the passed renderArea.

Parameters

renderArea	- The renderArea which the ellipse is drawn to

Implements Shape.

8.5.3.3 getA()

```
int Ellipse::getA () const
```

Accessor Functions.

8.5.3.4 getB()

```
int Ellipse::getB () const
```

8.5.3.5 isPointInside()

```
bool Ellipse::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the circle

Parameters

point	- point being read

Returns

Implements Shape.

8.5.3.6 Perimeter()

```
double Ellipse::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimeter of the ellipse.

Returns

Implements Shape.

8.5.3.7 setA()

```
void Ellipse::setA (
            int newA)
```

Accessor Functions.

Mutator Functions

8.5.3.8 setB()

```
void Ellipse::setB (
            int newB)
```

8.5.3.9 setX()

```
void Ellipse::setX (
            int newX)
```

Implements Shape.

8.5.3.10 setY()

```
void Ellipse::setY (
            int newY)
```

Implements Shape.

8.5.4 Member Data Documentation

8.5.4.1 a

```
int Ellipse::a  [private]
```

Mutator Functions.

Semi-Minor Axis

8.5.4.2 b

```
int Ellipse::b  [private]
```

Semi-Major Axis.

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

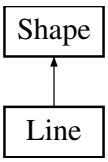
- src/objects/ellipse.h
- src/objects/ellipse.cpp

8.6 Line Class Reference

The Line class.

`#include "objects/line.h"`

Inheritance diagram for Line:



Public Member Functions

- Line (string shapeType, QPoint coords, QPen pen, QBrush brush, QPoint startPoint, QPoint endPoint)

  Line Constructor.
- void Draw (QWidget ∗renderArea) override

  Draw - Draws a line from startPoint to endPoint.
- void Move (int x, int y) override

  Move - Moves the Line coords to the passed x and y.
- double Perimeter () const override

  Perimeter - Returns the perimeter of the Line.
- double Area () const override

  Area - Returns 0, necessary override to avoid being seen as Abstract.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the Line
- QPoint getStartPoint () const

  Accessor Functions.
- QPoint getEndPoint () const
- void setStartPoint (const QPoint &newStartPoint)

  Mutator Functions.
- void setEndPoint (const QPoint &newEndPoint)
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ∼Shape ()

  ∼Shape Destructor
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.

- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- QPoint startPoint

  Starting point of the Line.
- QPoint endPoint

  Ending point of the line.

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.6.1 Detailed Description

The Line class.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 Line()

```
Line::Line (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            QPoint startPoint,
            QPoint endPoint)
```

Line Constructor.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
startPoint	- Point representing the start of the line
endPoint	- Point representing the end of the line

8.6.3 Member Function Documentation

8.6.3.1 Area()

```
double Line::Area () const  [inline], [override], [virtual]
```

Area - Returns 0, necessary override to avoid being seen as Abstract.

Returns

Implements Shape.

8.6.3.2 Draw()

```
void Line::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a line from startPoint to endPoint.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.6.3.3 getEndPoint()

```
QPoint Line::getEndPoint () const
```

8.6.3.4 getStartPoint()

```
QPoint Line::getStartPoint () const
```

Accessor Functions.

8.6.3.5 isPointInside()

```
bool Line::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the Line

Parameters

point	- point being read

Returns

Implements Shape.

8.6.3.6 Move()

```
void Line::Move (
            int x,
            int y)  [override], [virtual]
```

Move - Moves the Line coords to the passed x and y.

Parameters

x	- x coordinate
y	- y coordinate

Implements Shape.

8.6.3.7 Perimeter()

```
double Line::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimeter of the Line.

Returns

Implements Shape.

8.6.3.8 setEndPoint()

```
void Line::setEndPoint (
            const QPoint & newEndPoint)
```

8.6.3.9 setStartPoint()

```
void Line::setStartPoint (
            const QPoint & newStartPoint)
```

Mutator Functions.

8.6.3.10 setX()

```
void Line::setX (
            int newX)
```

Implements Shape.

8.6.3.11 setY()

```
void Line::setY (
            int newY)
```

Implements Shape.

8.6.4 Member Data Documentation

8.6.4.1 endPoint

```
QPoint Line::endPoint  [private]
```

Ending point of the line.

8.6.4.2 startPoint

```
QPoint Line::startPoint  [private]
```

Starting point of the Line.

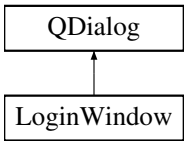
The documentation for this class was generated from the following files:

- src/objects/line.h
- src/objects/line.cpp

8.7 LoginWindow Class Reference

```
#include <loginwindow.h>
```

Inheritance diagram for LoginWindow:



Signals

- void loginRequested (const QString &username, const QString &password)
- void signupRequested (const QString &username, const QString &password, const bool admin)

Public Member Functions

- LoginWindow (QWidget *parent=nullptr)
- QString username () const
- QString password () const

Private Slots

- void showSignupPage ()
- void showLoginPage ()
- void attemptLogin ()
- void attemptSignup ()

Private Attributes

- QStackedWidget * stack
- QWidget * loginPage
- QLineEdit * loginUserEdit
- QLineEdit * loginPassEdit
- QPushButton * loginBtn
- QPushButton * toSignupBtn
- QWidget * signupPage
- QLineEdit * signupUserEdit
- QLineEdit * signupPassEdit
- QPushButton * signupBtn
- QPushButton * backToLoginBtn

8.7.1 Constructor & Destructor Documentation

8.7.1.1 LoginWindow()

```
LoginWindow::LoginWindow (
            QWidget * parent = nullptr)  [explicit]
```

8.7.2 Member Function Documentation

8.7.2.1 attemptLogin

```
void LoginWindow::attemptLogin ()  [private], [slot]
```

8.7.2.2 attemptSignup

```
void LoginWindow::attemptSignup ()  [private], [slot]
```

8.7.2.3 loginRequested

```
void LoginWindow::loginRequested (
            const QString & username,
            const QString & password)  [signal]
```

8.7.2.4 password()

```
QString LoginWindow::password () const
```

8.7.2.5 showLoginPage

```
void LoginWindow::showLoginPage ()  [private], [slot]
```

8.7.2.6 showSignupPage

```
void LoginWindow::showSignupPage ()  [private], [slot]
```

8.7.2.7 signupRequested

```
void LoginWindow::signupRequested (
            const QString & username,
            const QString & password,
            const bool admin)  [signal]
```

8.7.2.8 username()

```
QString LoginWindow::username () const
```

8.7.3 Member Data Documentation

8.7.3.1 backToLoginBtn

```
QPushButton* LoginWindow::backToLoginBtn  [private]
```

8.7.3.2 loginBtn

```
QPushButton* LoginWindow::loginBtn  [private]
```

8.7.3.3 loginPage

```
QWidget* LoginWindow::loginPage  [private]
```

8.7.3.4 loginPassEdit

```
QLineEdit* LoginWindow::loginPassEdit  [private]
```

8.7.3.5 loginUserEdit

```
QLineEdit* LoginWindow::loginUserEdit  [private]
```

8.7.3.6 signupBtn

```
QPushButton* LoginWindow::signupBtn  [private]
```

8.7.3.7 signupPage

```
QWidget* LoginWindow::signupPage  [private]
```

8.7.3.8 signupPassEdit

```
QLineEdit* LoginWindow::signupPassEdit  [private]
```

8.7.3.9 signupUserEdit

```
QLineEdit* LoginWindow::signupUserEdit  [private]
```

8.7.3.10 stack

```
QStackedWidget* LoginWindow::stack  [private]
```

8.7.3.11 toSignupBtn

```
QPushButton* LoginWindow::toSignupBtn  [private]
```

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

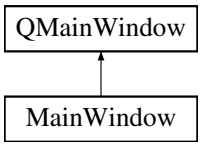
- src/frontend/loginwindow.h
- src/frontend/loginwindow.cpp

8.8 MainWindow Class Reference

The main window of the 2D Graphics Modeler application.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Slots

RenderAreaManager Response Slots

Slots for handling responses from RenderAreaManager.

These slots are connected to signals from the RenderAreaManager to update the UI when the state of rendered shapes changes or status messages are available.

- void onRenderAreaChanged ()
  Slot triggered when the render area's content has changed, requiring a UI update.
- void onRenderAreaNotChanged (const QString &message)
  Slot triggered when an operation on the render area did not result in a change (e.g., error).
- void showRenderStatusMessage (const QString &message)
  Displays a status message related to rendering operations.

Login Process Management Slots

Slots for managing the login process.

These slots handle user interactions related to logging in and signing up.

- void onLoginClicked ()
  Slot triggered when the login button is clicked, typically opening a login dialog.
- void onLoginRequest (const QString &username, const QString &password)
  Slot triggered by a login dialog to initiate a login attempt.
- void onSignupRequest (const QString &username, const QString &password, const bool admin)

UserManager Response Slots

Slots for handling responses from UserManager.

These slots are connected to signals from the UserManager to update the UI based on the outcome of user authentication attempts.

- void onUserAuthentication (const UserAccount *currUser)
  Slot triggered upon successful user authentication. Updates UI with user info.
- void onUserAuthenticationFailure (const QString &message)
  Slot triggered when user authentication fails. Displays an error message.

Signals

Shape Manipulation Signals

Signals related to shape manipulation.

These signals are emitted to notify other components (likely AppDriver or RenderAreaManager) about actions performed on shapes by the user through the UI.

- void shapeAdded (Shape *shape)
  Emitted when a new shape is added by the user.
- void shapeChanged (Shape *shape, QString key, int value)
  Emitted when a property of an existing shape is changed.
- void displayedTextChanged (Text *text, QString newText)
  Emitted when the displayed text of a Text shape is changed.
- void shapeDeleted (int trackerId)
  Emitted when a shape is requested to be deleted.
- void deleteAllShapes ()
  Emitted when a request to delete all shapes is made.

User Authentication Signals

Signals related to user login and signup.

These signals are emitted to the AppDriver or UserManager to handle user authentication and account creation processes.

- void loginAttempt (const QString &username, const QString &password)
  Emitted when a user attempts to log in.
- void newUserAdded (const QString &username, const QString &password, const bool admin)

Dialog Flow Control Signals

Signals for controlling dialog flow.

These signals are used to communicate the success or failure of operations like login, often to control the state of dialog windows.

- void loginSuccess ()
  Emitted when a login attempt is successful.
- void loginFailed (const QString &message)
  Emitted when a login attempt fails, providing an error message.

Public Member Functions

Constructors and Destructor

Manages the lifecycle of MainWindow objects.

These methods handle the creation, initialization, and cleanup of the MainWindow object.

- MainWindow (QWidget *parent=nullptr)
  Default constructor.
- MainWindow (QWidget *parent, const alpha::vector< Shape * > *renderedShapes, const UserAccount *currUser)
- ~MainWindow ()
  Destructor. Cleans up UI elements and other resources.

Shape Display Interface

Public interface methods for shape display.

These methods are responsible for drawing shapes on the render area and populating the tree widget with shape information.

- void drawShapes () const
  Instructs the render area to draw the current shapes. (Note: Actual drawing is typically triggered by `renderArea->update()`).
- void shapes_to_treeWidget ()
  Populates the tree widget with details of the current shapes and sets up editing controls.

Private Slots

UI Customization Slots

Slot for UI style toggling.

This slot handles changes to the application's visual style.

- void onToggleStyle (bool checked=true)
  Toggles the application's stylesheet between predefined styles.

Shape Creation Action Slots

Slots for shape creation actions.

These slots are connected to toolbar buttons or menu actions that allow the user to add new shapes to the canvas.

- void on_actionnew_line_button_triggered ()
  Slot triggered to create and add a new Line shape.
- void on_actionnew_square_button_triggered ()
  Slot triggered to create and add a new Square shape.
- void on_actionnew_rectange_button_triggered ()
  Slot triggered to create and add a new Rectangle shape.
- void on_actionnew_circle_button_triggered ()
  Slot triggered to create and add a new Circle shape.
- void on_actionnew_ellipse_button_triggered ()
  Slot triggered to create and add a new Ellipse shape.
- void on_actionnew_polyline_button_triggered ()
  Slot triggered to create and add a new Polyline shape.
- void on_actionnew_polygon_button_triggered ()
  Slot triggered to create and add a new Polygon shape.
- void on_actionnew_text_button_triggered ()
  Slot triggered to create and add a new Text shape.
- void on_actionremove_shape_button_triggered ()
  Slot triggered to remove the currently selected shape.

Shape Report Sorting Slots

Slots for shape sorting functionality in the shape report.

This slot handles changes in the sorting criteria for the shape table.

- void onSortMethodChanged (int index)
  Slot triggered when the sort method or order in the shape report table changes.

Tree Widget Editing Slots

Slots for modifying shape properties via the tree widget.

These slots respond to user edits made in the shape properties tree, updating the corresponding shape attributes.

- void onTreeWidgetItemChanged (QTreeWidgetItem *item, int column)
  Slot triggered when an item in the shape properties tree widget is changed.
- void onComboBoxChanged (int newIndex)
  Slot triggered when a QComboBox value in the tree widget changes.
- void onSpinBoxChanged ()
  Slot triggered when a QSpinBox value in the tree widget changes.

Testimonial UI Slots

Slots for testimonial UI.

These slots handle the display of testimonial prompts and the testimonial viewing dialog.

- void showTestimonialPrompt ()
  Displays a dialog prompting the user to submit a testimonial.
- void showTestimonialsDisplay ()
  Displays a dialog showing existing testimonials.

Contact Information Slot

Slot for displaying the "Contact Us" window.

This slot is triggered by a menu action to show contact information.

- void onContactUsClicked ()
  Slot triggered to display the 'Contact Us' information window.

Private Member Functions

UI Management Helpers

Private helper methods for UI management.

These methods assist in managing UI elements, such as adding shapes to the tree widget and loading stylesheets.

- void addToShapeTree (Shape *shape)
  Adds a given shape to the properties tree widget and emits shapeAdded signal.
- QString loadStyleSheet (const QString &path)
  Loads a Qt stylesheet from the specified file path.

Editor Widget Factories

Factory methods for creating QComboBox and QSpinBox widgets.

These methods are used to dynamically create and configure editor widgets (like QComboBoxes for color, style, etc., and QSpinBoxes for numerical values) for the shape properties tree. Each method initializes the widget with items and sets its current value based on the shape's property.

- QComboBox * createShapeTypeComboBox (const QString &currentShapeType)
  Creates and configures a QComboBox for selecting shape types.
- QSpinBox * createPenWidthSpinBox (int currentPenWidth)
  Creates and configures a QSpinBox for editing pen width.
- QComboBox * createColorComboBox (int currentColor)
  Creates and configures a QComboBox for selecting colors.
- QComboBox * createPenStyleComboBox (int currentPenStyle)
  Creates and configures a QComboBox for selecting pen styles.
- QComboBox * createPenCapStyleComboBox (int currentPenCapStyle)
  Creates and configures a QComboBox for selecting pen cap styles.
- QComboBox * createPenJoinStyleComboBox (int currentPenJoinStyle)
  Creates and configures a QComboBox for selecting pen join styles.
- QComboBox * createBrushStyleComboBox (int currentBrushStyle)
  Creates and configures a QComboBox for selecting brush styles.
- QComboBox * createAlignmentComboBox (Qt::AlignmentFlag currentAlignment)
  Creates and configures a QComboBox for selecting text alignment.
- QComboBox * createFontComboBox (QFont currentFont)
  Creates and configures a QComboBox for selecting font families.
- QComboBox * createFontStyleComboBox (int currentFontStyle)
  Creates and configures a QComboBox for selecting font styles (normal, italic, oblique).
- QComboBox * createFontWeightComboBox (QFont::Weight currentFontWeight)
  Creates and configures a QComboBox for selecting font weights.

Testimonial Feature Setup

Testimonial feature setup.

This method initializes components related to the testimonial feature.

- void setupTestimonials ()
  Sets up connections and UI elements for the testimonial feature.

Private Attributes

Core UI Components and Data

Core UI components and data pointers.

These members are essential for the MainWindow's operation, including the UI definition, rendering area, and pointers to shared data.

- Ui::MainWindow * ui
  Pointer to the auto-generated UI class.
- RenderArea * renderArea
  Pointer to the custom widget responsible for drawing shapes.
- const alpha::vector< Shape * > * renderShapes
  Pointer to the vector of shapes currently being rendered.
- const UserAccount * currUser
  Pointer to the currently logged-in user account.
- QLabel * userStatusLabel
  Label in the status bar to display the current user's login status.
- ColumnEditDelegate * delegate
  Custom delegate for editing items in the shape properties tree widget.

Contact Us Window Members

Members for the "Contact Us" window.

Pointers to widgets used in the "Contact Us" information display.

- QWidget * contactUsWidget
  Pointer to the main widget for the Contact Us display.
- QLabel * logoLabel
  Label to display the team logo in the Contact Us display.
- QLabel * teamNameLabel
  Label to display the team name in the Contact Us display.
- QWidget * contactWindow
  Pointer to the QWidget that serves as the 'Contact Us' window.

Shape Report Table Management

Methods and members related to the shape report table.

These are used to create, populate, and manage the tab/window that displays a sortable table of shape properties.

- QTabWidget * tabWidget
  Pointer to the tab widget (if used for the shape report, otherwise could be for other purposes).
- QTableWidget * shapeTable
  Pointer to the table widget used to display shape properties in the report.
- QComboBox * sortDropdown
  Dropdown QComboBox for selecting the sorting criterion in the shape report.
- QComboBox * sortOrderDropdown
  Dropdown QComboBox for selecting the sorting order (ascending/descending) in the shape report.
- void createShapeTableTab ()
  Creates and displays a new tab or window containing the shape report table.

Shape Sorting Logic

Sorting algorithm and comparison functions for the shape report.

Implements a selection sort algorithm and provides static comparison functions used to sort shapes by different criteria.

- void selection_sort (alpha::vector< Shape * > &shapes, bool(*compare)(const Shape *, const Shape *), bool ascending)
- void populateShapeTable (const alpha::vector< Shape * > &shapes)

  Populates the shape report table with data from the provided vector of shapes.
- static bool sortById (const Shape *a, const Shape *b)

  Static comparison function to sort shapes by their ID.
- static bool sortByArea (const Shape *a, const Shape *b)

  Static comparison function to sort shapes by their calculated area.
- static bool sortByPerimeter (const Shape *a, const Shape *b)

  Static comparison function to sort shapes by their calculated perimeter.

8.8.1 Detailed Description

The main window of the 2D Graphics Modeler application.

This class serves as the primary user interface, managing shape rendering, user interactions, login, testimonials, and displaying shape information. It coordinates between the UI elements and backend logic.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 MainWindow() [1/2]

```
MainWindow::MainWindow (
            QWidget * parent = nullptr)  [explicit]
```

Default constructor.

8.8.2.2 MainWindow() [2/2]

```
MainWindow::MainWindow (
            QWidget * parent,
            const alpha::vector< Shape * > * renderedShapes,
            const UserAccount * currUser)
```

Parameters

parent	Constructor initializing with shapes and user data.

8.8.2.3 ~MainWindow()

```
MainWindow::~MainWindow ()
```

Destructor. Cleans up UI elements and other resources.

8.8.3 Member Function Documentation

8.8.3.1 addToShapeTree()

```
void MainWindow::addToShapeTree (
            Shape * shape)  [private]
```

Adds a given shape to the properties tree widget and emits shapeAdded signal.

8.8.3.2 createAlignmentComboBox()

```
QComboBox * MainWindow::createAlignmentComboBox (
            Qt::AlignmentFlag currentAlignment)  [private]
```

Creates and configures a QComboBox for selecting text alignment.

8.8.3.3 createBrushStyleComboBox()

```
QComboBox * MainWindow::createBrushStyleComboBox (
            int currentBrushStyle)  [private]
```

Creates and configures a QComboBox for selecting brush styles.

8.8.3.4 createColorComboBox()

```
QComboBox * MainWindow::createColorComboBox (
            int currentColor)  [private]
```

Creates and configures a QComboBox for selecting colors.

8.8.3.5 createFontComboBox()

```
QComboBox * MainWindow::createFontComboBox (
            QFont currentFont)  [private]
```

Creates and configures a QComboBox for selecting font families.

8.8.3.6 createFontStyleComboBox()

```
QComboBox * MainWindow::createFontStyleComboBox (
            int currentFontStyle)  [private]
```

Creates and configures a QComboBox for selecting font styles (normal, italic, oblique).

8.8.3.7 createFontWeightComboBox()

```
QComboBox * MainWindow::createFontWeightComboBox (
            QFont::Weight currentFontWeight)  [private]
```

Creates and configures a QComboBox for selecting font weights.

8.8.3.8 createPenCapStyleComboBox()

```
QComboBox * MainWindow::createPenCapStyleComboBox (
            int currentPenCapStyle)  [private]
```

Creates and configures a QComboBox for selecting pen cap styles.

8.8.3.9 createPenJoinStyleComboBox()

```
QComboBox * MainWindow::createPenJoinStyleComboBox (
            int currentPenJoinStyle)  [private]
```

Creates and configures a QComboBox for selecting pen join styles.

8.8.3.10 createPenStyleComboBox()

```
QComboBox * MainWindow::createPenStyleComboBox (
            int currentPenStyle)  [private]
```

Creates and configures a QComboBox for selecting pen styles.

8.8.3.11 createPenWidthSpinBox()

```
QSpinBox * MainWindow::createPenWidthSpinBox (
            int currentPenWidth)  [private]
```

Creates and configures a QSpinBox for editing pen width.

8.8.3.12 createShapeTableTab()

```
void MainWindow::createShapeTableTab ()  [private]
```

Creates and displays a new tab or window containing the shape report table.

8.8.3.13 createShapeTypeComboBox()

```
QComboBox * MainWindow::createShapeTypeComboBox (
            const QString & currentShapeType)  [private]
```

Creates and configures a QComboBox for selecting shape types.

8.8.3.14 deleteAllShapes

```
void MainWindow::deleteAllShapes ()  [signal]
```

Emitted when a request to delete all shapes is made.

8.8.3.15 displayedTextChanged

```
void MainWindow::displayedTextChanged (
            Text * text,
            QString newText)  [signal]
```

Emitted when the displayed text of a Text shape is changed.

8.8.3.16 drawShapes()

```
void MainWindow::drawShapes () const
```

Instructs the render area to draw the current shapes. (Note: Actual drawing is typically triggered by `renderArea->update()`).

8.8.3.17 loadStyleSheet()

```
QString MainWindow::loadStyleSheet (
            const QString & path)  [private]
```

Loads a Qt stylesheet from the specified file path.

8.8.3.18 loginAttempt

```
void MainWindow::loginAttempt (
            const QString & username,
            const QString & password)  [signal]
```

Emitted when a user attempts to log in.

8.8.3.19 loginFailed

```
void MainWindow::loginFailed (
            const QString & message)  [signal]
```

Emitted when a login attempt fails, providing an error message.

8.8.3.20 loginSuccess

```
void MainWindow::loginSuccess ()  [signal]
```

Emitted when a login attempt is successful.

8.8.3.21 newUserAdded

```
void MainWindow::newUserAdded (
            const QString & username,
            const QString & password,
            const bool admin)  [signal]
```

Parameters

password	Emitted when a new user account is requested to be created.

8.8.3.22 on_actionnew_circle_button_triggered

```
void MainWindow::on_actionnew_circle_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Circle shape.

8.8.3.23 on_actionnew_ellipse_button_triggered

```
void MainWindow::on_actionnew_ellipse_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Ellipse shape.

8.8.3.24 on_actionnew_line_button_triggered

```
void MainWindow::on_actionnew_line_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Line shape.

8.8.3.25 on_actionnew_polygon_button_triggered

```
void MainWindow::on_actionnew_polygon_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Polygon shape.

8.8.3.26 on_actionnew_polyline_button_triggered

```
void MainWindow::on_actionnew_polyline_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Polyline shape.

8.8.3.27 on_actionnew_rectange_button_triggered

```
void MainWindow::on_actionnew_rectange_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Rectangle shape.

8.8.3.28 on_actionnew_square_button_triggered

```
void MainWindow::on_actionnew_square_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Square shape.

8.8.3.29 on_actionnew_text_button_triggered

```
void MainWindow::on_actionnew_text_button_triggered ()  [private], [slot]
```

Slot triggered to create and add a new Text shape.

8.8.3.30 on_actionremove_shape_button_triggered

```
void MainWindow::on_actionremove_shape_button_triggered ()  [private], [slot]
```

Slot triggered to remove the currently selected shape.

8.8.3.31 onComboBoxChanged

```
void MainWindow::onComboBoxChanged (
            int newIndex)  [private], [slot]
```

Slot triggered when a QComboBox value in the tree widget changes.

8.8.3.32 onContactUsClicked

```
void MainWindow::onContactUsClicked ()  [private], [slot]
```

Slot triggered to display the 'Contact Us' information window.

8.8.3.33 onLoginClicked

```
void MainWindow::onLoginClicked ()  [slot]
```

Slot triggered when the login button is clicked, typically opening a login dialog.

8.8.3.34 onLoginRequest

```
void MainWindow::onLoginRequest (
            const QString & username,
            const QString & password)  [slot]
```

Slot triggered by a login dialog to initiate a login attempt.

8.8.3.35 onRenderAreaChanged

```
void MainWindow::onRenderAreaChanged ()  [slot]
```

Slot triggered when the render area's content has changed, requiring a UI update.

8.8.3.36 onRenderAreaNotChanged

```
void MainWindow::onRenderAreaNotChanged (
            const QString & message)  [slot]
```

Slot triggered when an operation on the render area did not result in a change (e.g., error).

8.8.3.37 onSignupRequest

```
void MainWindow::onSignupRequest (
            const QString & username,
            const QString & password,
            const bool admin)  [slot]
```

Parameters

password	Slot triggered by a login/signup dialog to initiate account creation.

8.8.3.38 onSortMethodChanged

```
void MainWindow::onSortMethodChanged (
            int index)  [private], [slot]
```

Slot triggered when the sort method or order in the shape report table changes.

8.8.3.39 onSpinBoxChanged

```
void MainWindow::onSpinBoxChanged ()  [private], [slot]
```

Slot triggered when a QSpinBox value in the tree widget changes.

8.8.3.40 onToggleStyle

```
void MainWindow::onToggleStyle (
            bool checked = true)  [private], [slot]
```

Toggles the application's stylesheet between predefined styles.

8.8.3.41 onTreeWidgetItemChanged

```
void MainWindow::onTreeWidgetItemChanged (
            QTreeWidgetItem * item,
            int column)  [private], [slot]
```

Slot triggered when an item in the shape properties tree widget is changed.

8.8.3.42 onUserAuthentication

```
void MainWindow::onUserAuthentication (
            const UserAccount * currUser)  [slot]
```

Slot triggered upon successful user authentication. Updates UI with user info.

8.8.3.43 onUserAuthenticationFailure

```
void MainWindow::onUserAuthenticationFailure (
            const QString & message)  [slot]
```

Slot triggered when user authentication fails. Displays an error message.

8.8.3.44 populateShapeTable()

```
void MainWindow::populateShapeTable (
            const alpha::vector< Shape * > & shapes)  [private]
```

Populates the shape report table with data from the provided vector of shapes.

8.8.3.45 selection_sort()

```
void MainWindow::selection_sort (
            alpha::vector< Shape * > & shapes,
            bool(* compare )(const Shape *, const Shape *),
            bool ascending)  [private]
```

Parameters

shapes	Implements the selection sort algorithm for a vector of shapes.

8.8.3.46 setupTestimonials()

```
void MainWindow::setupTestimonials ()  [private]
```

Sets up connections and UI elements for the testimonial feature.

8.8.3.47 shapeAdded

```
void MainWindow::shapeAdded (
            Shape * shape)  [signal]
```

Emitted when a new shape is added by the user.

8.8.3.48 shapeChanged

```
void MainWindow::shapeChanged (
            Shape * shape,
            QString key,
            int value)  [signal]
```

Emitted when a property of an existing shape is changed.

8.8.3.49 shapeDeleted

```
void MainWindow::shapeDeleted (
            int trackerId)  [signal]
```

Emitted when a shape is requested to be deleted.

8.8.3.50 shapes_to_treeWidget()

```
void MainWindow::shapes_to_treeWidget ()
```

Populates the tree widget with details of the current shapes and sets up editing controls.

8.8.3.51 showRenderStatusMessage

```
void MainWindow::showRenderStatusMessage (
            const QString & message)  [slot]
```

Displays a status message related to rendering operations.

8.8.3.52 showTestimonialPrompt

```
void MainWindow::showTestimonialPrompt ()  [private], [slot]
```

Displays a dialog prompting the user to submit a testimonial.

8.8.3.53 showTestimonialsDisplay

```
void MainWindow::showTestimonialsDisplay ()  [private], [slot]
```

Displays a dialog showing existing testimonials.

8.8.3.54 sortByArea()

```
bool MainWindow::sortByArea (
            const Shape * a,
            const Shape * b)  [static], [private]
```

Static comparison function to sort shapes by their calculated area.

8.8.3.55 sortById()

```
bool MainWindow::sortById (
            const Shape * a,
            const Shape * b)  [static], [private]
```

Static comparison function to sort shapes by their ID.

8.8.3.56 sortByPerimeter()

```
bool MainWindow::sortByPerimeter (
            const Shape * a,
            const Shape * b)  [static], [private]
```

Static comparison function to sort shapes by their calculated perimeter.

8.8.4 Member Data Documentation

8.8.4.1 contactUsWidget

```
QWidget* MainWindow::contactUsWidget  [private]
```

Pointer to the main widget for the Contact Us display.

8.8.4.2 contactWindow

```
QWidget* MainWindow::contactWindow  [private]
```

Pointer to the QWidget that serves as the 'Contact Us' window.

8.8.4.3 currUser

```
const UserAccount* MainWindow::currUser  [private]
```

Pointer to the currently logged-in user account.

8.8.4.4 delegate

```
ColumnEditDelegate* MainWindow::delegate  [private]
```

Custom delegate for editing items in the shape properties tree widget.

8.8.4.5 logoLabel

```
QLabel* MainWindow::logoLabel  [private]
```

Label to display the team logo in the Contact Us display.

8.8.4.6 renderArea

```
RenderArea* MainWindow::renderArea  [private]
```

Pointer to the custom widget responsible for drawing shapes.

8.8.4.7 renderShapes

```
const alpha::vector<Shape*>* MainWindow::renderShapes  [private]
```

Pointer to the vector of shapes currently being rendered.

8.8.4.8 shapeTable

```
QTableWidget* MainWindow::shapeTable  [private]
```

Pointer to the table widget used to display shape properties in the report.

8.8.4.9 sortDropdown

```
QComboBox* MainWindow::sortDropdown  [private]
```

Dropdown QComboBox for selecting the sorting criterion in the shape report.

8.8.4.10 sortOrderDropdown

```
QComboBox* MainWindow::sortOrderDropdown  [private]
```

Dropdown QComboBox for selecting the sorting order (ascending/descending) in the shape report.

8.8.4.11 tabWidget

```
QTabWidget* MainWindow::tabWidget  [private]
```

Pointer to the tab widget (if used for the shape report, otherwise could be for other purposes).

8.8.4.12 teamNameLabel

```
QLabel* MainWindow::teamNameLabel  [private]
```

Label to display the team name in the Contact Us display.

8.8.4.13 ui

```
Ui::MainWindow* MainWindow::ui  [private]
```

Pointer to the auto-generated UI class.

8.8.4.14 userStatusLabel

```
QLabel* MainWindow::userStatusLabel  [private]
```

Label in the status bar to display the current user's login status.

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

- src/frontend/mainwindow.h
- src/frontend/mainwindow.cpp

8.9 Parser::MorphicShape Struct Reference

Internal accumulator structure for parsing shape data from JSON.

Public Attributes

- std::string shapeType = ""

  The type of the shape (e.g., "Line", "Circle").
- int shapeId = 0

  The unique identifier for the shape type.
- int trackerId = 0

  A tracking ID for the shape instance.
- alpha::vector< int > shapeDimensions

  A vector holding the geometric dimensions of the shape.
- QPen pen = QPen()

  The QPen object defining the shape's outline properties.
- QBrush brush = QBrush()

  The QBrush object defining the shape's fill properties.
- QPoint coords = QPoint()

  The primary coordinates (e.g., top-left point) of the shape.
- QString textString

  The string content for Text shapes.
- GlobalColor textColor

  The color of the text for Text shapes.
- QFont font

  The QFont object for Text shapes.
- AlignmentFlag textAlignment

  The alignment for Text shapes.

8.9.1 Detailed Description

Internal accumulator structure for parsing shape data from JSON.

This structure temporarily holds the properties of a shape as they are parsed from a JSON object. It allows for jumbled key-value pairs and converts string values to their appropriate C++ types before a final Shape object is constructed.

8.9.2 Member Data Documentation

8.9.2.1 brush

```
QBrush Parser::MorphicShape::brush = QBrush()
```

The QBrush object defining the shape's fill properties.

8.9.2.2 coords

```
QPoint Parser::MorphicShape::coords = QPoint()
```

The primary coordinates (e.g., top-left point) of the shape.

8.9.2.3 font

```
QFont Parser::MorphicShape::font
```

The QFont object for Text shapes.

8.9.2.4 pen

```
QPen Parser::MorphicShape::pen = QPen()
```

The QPen object defining the shape's outline properties.

8.9.2.5 shapeDimensions

```
alpha::vector<int> Parser::MorphicShape::shapeDimensions
```

A vector holding the geometric dimensions of the shape.

8.9.2.6 shapeId

```
int Parser::MorphicShape::shapeId = 0
```

The unique identifier for the shape type.

8.9.2.7 shapeType

```
std::string Parser::MorphicShape::shapeType = ""
```

The type of the shape (e.g., "Line", "Circle").

8.9.2.8 textAlignment

```
AlignmentFlag Parser::MorphicShape::textAlignment
```

The alignment for Text shapes.

8.9.2.9 textColor

```
GlobalColor Parser::MorphicShape::textColor
```

The color of the text for Text shapes.

8.9.2.10 textString

```
QString Parser::MorphicShape::textString
```

The string content for Text shapes.

8.9.2.11 trackerId

```
int Parser::MorphicShape::trackerId = 0
```

A tracking ID for the shape instance.

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

- src/backend/Parser.h

8.10 Parser Class Reference

Provides functionality for parsing JSON data to C++ objects and vice-versa.

```
#include <Parser.h>
```

Classes

- struct MorphicShape
  Internal accumulator structure for parsing shape data from JSON.
- struct RawUser
  Internal accumulator structure for parsing user account data from JSON.

Public Member Functions

- Parser ()=default
  Default constructor. Initializes a Parser object.
- ~Parser ()=default
  Default destructor. Cleans up resources used by the Parser object.
- Parser (const Parser &)=delete
  Deleted copy constructor to prevent copying Parser objects.
- Parser & operator= (const Parser &)=delete
  Deleted copy assignment operator to prevent copying Parser objects.
- Parser (Parser &&)=delete
  Deleted move constructor to prevent moving Parser objects.
- Parser & operator= (Parser &&)=delete
  Deleted move assignment operator to prevent moving Parser objects.
- void PrintShapeVector (const alpha::vector< Shape * > &shapes)
  Prints properties of shapes in a vector to the console.
- alpha::vector< Shape * > JsonToShapes (const std::string &json)
  Converts a JSON string representation into a vector of Shape objects.
- std::string ShapesToJson (const alpha::vector< Shape * > &shapes)
  Converts a vector of Shape pointers into a JSON string representation.
- alpha::vector< UserAccount * > JsonToUsers (const std::string &json)
  Converts a JSON string representation into a vector of UserAccount objects.
- std::string UsersToJson (const alpha::vector< UserAccount * > &users)
  Converts a vector of UserAccount pointers into a JSON string representation.

Static Public Member Functions

- static QVector< Testimonial > JsonToTestimonials (const std::string &json)

  Converts a JSON string representation into a QVector of Testimonial objects.
- static std::string TestimonialsToJson (const QVector< Testimonial > &testimonials)

  Converts a QVector of Testimonial objects into a JSON string representation.

Private Member Functions

- MorphicShape ParseJsonObject (const std::string json, size_t &index)

  Parses a single JSON object (data within '{}') into a MorphicShape.
- void UpdateAccumulator (const std::string &key, const std::string &value, MorphicShape &tempShape)

  Updates a MorphicShape accumulator with a parsed key-value pair.
- Shape * BuildShape (MorphicShape tempShape)

  Constructs a concrete Shape object from a populated MorphicShape accumulator.
- void SkipWhitespace (const std::string &json, size_t &index)

  Advances the parsing index past any whitespace characters.
- std::string ExtractKey (const std::string &json, size_t &index)

  Extracts a JSON key (a string enclosed in double quotes) from the input string.
- std::string ExtractValue (const std::string &json, size_t &index)

  Extracts a JSON value from the input string.
- std::string ExtractInteger (const std::string &json, size_t &index)

  Extracts an integer value from the JSON string.
- std::string ExtractArray (const std::string &json, size_t &index)

  Extracts a JSON array (content within '[]') as a string.
- std::string ExtractLiteral (const std::string &json, size_t &index)

  Extracts a JSON literal (true, false, null) as a string.
- alpha::vector< int > StringToVector (const std::string &value)

  Converts a string representation of a JSON array of integers into an alpha::vector<int>.
- std::string AppendCommonShapeData (const Shape *shape)

  Appends common shape properties (ID, Type, Dimensions, Pen properties) to a JSON string.
- std::string AppendBrushData (const Shape *shape)

  Appends QBrush properties (BrushColor, BrushStyle) to a JSON string.
- std::string AppendTextData (const Shape *shape)

  Appends all properties specific to Text objects to a JSON string.
- std::string GetShapeDimensions (const Shape *shape)

  Gets the geometric dimensions of a shape as a JSON array string.
- std::string GetColor (const QColor &objectColor)

  Converts a QColor object to its string representation (e.g., "red", "blue").
- std::string GetPenStyle (const Shape *shape)

  Gets the pen style of a shape as a string (e.g., "SolidLine", "DashLine").
- std::string GetPenCapStyle (const Shape *shape)

  Gets the pen cap style of a shape as a string (e.g., "FlatCap", "RoundCap").
- std::string GetPenJoinStyle (const Shape *shape)

  Gets the pen join style of a shape as a string (e.g., "MiterJoin", "BevelJoin").
- std::string GetBrushStyle (const Shape *shape)

  Gets the brush style of a shape as a string (e.g., "SolidPattern", "NoBrush").
- std::string GetAlignmentFlag (const Text *text)

  Gets the text alignment of a Text object as a string (e.g., "AlignLeft", "AlignCenter").
- std::string GetFontStyle (const Text *text)

  Gets the font style of a Text object as a string (e.g., "StyleNormal", "StyleItalic").
- std::string GetFontWeight (const Text *text)

  Gets the font weight of a Text object as a string (e.g., "Normal", "Bold").

Static Private Member Functions

- static void UpdateUserAccumulator (const std::string &key, const std::string &value, RawUser &acc)

  Updates a RawUser accumulator with a parsed key-value pair for user data.

8.10.1 Detailed Description

Provides functionality for parsing JSON data to C++ objects and vice-versa.

The Parser class handles the serialization and deserialization of various data structures used in the application, such as Shapes, UserAccounts, and Testimonials, converting them to and from JSON string representations. It includes methods for both forward parsing (JSON to object) and reverse parsing (object to JSON). This class does not store any data itself and its copy/move operations are disabled.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 Parser() [1/3]

```
Parser::Parser ()  [default]
```

Default constructor. Initializes a Parser object.

8.10.2.2 ~Parser()

```
Parser::~Parser ()  [default]
```

Default destructor. Cleans up resources used by the Parser object.

8.10.2.3 Parser() [2/3]

```
Parser::Parser (
            const Parser & )  [delete]
```

Deleted copy constructor to prevent copying Parser objects.

8.10.2.4 Parser() [3/3]

```
Parser::Parser (
            Parser && )  [delete]
```

Deleted move constructor to prevent moving Parser objects.

8.10.3 Member Function Documentation

8.10.3.1 AppendBrushData()

```
std::string Parser::AppendBrushData (
            const Shape * shape)  [private]
```

Appends QBrush properties (BrushColor, BrushStyle) to a JSON string.

This helper function is used for shapes that have fill properties.

Parameters

shape	A constant pointer to the Shape object whose brush data is to be serialized.

Returns

A std::string containing the JSON representation of brush data.

8.10.3.2 AppendCommonShapeData()

```
std::string Parser::AppendCommonShapeData (
            const Shape * shape)  [private]
```

Appends common shape properties (ID, Type, Dimensions, Pen properties) to a JSON string.

This helper function is used during the serialization of shapes to JSON. It formats 9 common key-value pairs.

Parameters

shape	A constant pointer to the Shape object to serialize.

Returns

A std::string containing the JSON representation of common shape data.

8.10.3.3 AppendTextData()

```
std::string Parser::AppendTextData (
            const Shape * shape)  [private]
```

Appends all properties specific to Text objects to a JSON string.

Serializes properties like TextString, PointSize, Color, Alignment, Font.

Parameters

shape	A constant pointer to the Shape object, expected to be a Text object.

Returns

A std::string containing the JSON representation of text data.

Exceptions

std::runtime_error	If the provided shape cannot be cast to Text.

8.10.3.4 BuildShape()

```
Shape * Parser::BuildShape (
            MorphicShape tempShape)  [private]
```

Constructs a concrete Shape object from a populated MorphicShape accumulator.

Based on the `shapeId` in `tempShape`, this function dynamically allocates and initializes the appropriate derived Shape object (e.g., Line, Circle).

Parameters

tempShape	The MorphicShape object containing the data for the new shape.

Returns

A pointer to the newly instantiated Shape object, or nullptr if construction fails.

8.10.3.5 ExtractArray()

```
std::string Parser::ExtractArray (
            const std::string & json,
            size_t & index)  [private]
```

Extracts a JSON array (content within '[]') as a string.

Assumes `index` is at the opening bracket of the array. Modifies `index` to point after the closing bracket of the array.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

Returns

The extracted array (including brackets) as a std::string.

Exceptions

std::runtime_error	If the closing bracket is missing.

8.10.3.6 ExtractInteger()

```
std::string Parser::ExtractInteger (
            const std::string & json,
            size_t & index)  [private]
```

Extracts an integer value from the JSON string.

Assumes `index` is at the first digit of the integer. Modifies `index` to point after the last digit of the integer.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

Returns

The extracted integer as a std::string.

8.10.3.7 ExtractKey()

```
std::string Parser::ExtractKey (
            const std::string & json,
            size_t & index)  [private]
```

Extracts a JSON key (a string enclosed in double quotes) from the input string.

Assumes `index` is at the opening double quote of the key. Modifies `index` to point immediately after the closing double quote of the key.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

Returns

The extracted key as a std::string.

8.10.3.8 ExtractLiteral()

```
std::string Parser::ExtractLiteral (
            const std::string & json,
            size_t & index)  [private]
```

Extracts a JSON literal (true, false, null) as a string.

Assumes `index` is at the first character of the literal. Modifies `index` to point after the last character of the literal.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

Returns

The extracted literal as a std::string.

8.10.3.9 ExtractValue()

```
std::string Parser::ExtractValue (
            const std::string & json,
            size_t & index)  [private]
```

Extracts a JSON value from the input string.

Handles strings, numbers, arrays, and boolean literals. Modifies `index` to point after the extracted value.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

Returns

The extracted value as a std::string.

Exceptions

std::runtime_error	If the value type is unexpected.

8.10.3.10 GetAlignmentFlag()

```
std::string Parser::GetAlignmentFlag (
            const Text * text)  [private]
```

Gets the text alignment of a Text object as a string (e.g., "AlignLeft", "AlignCenter").

Parameters

text	A constant pointer to the Text object.

Returns

A std::string representing the text alignment.

Exceptions

std::runtime_error	If the alignment flag is unknown.

8.10.3.11 GetBrushStyle()

```
std::string Parser::GetBrushStyle (
            const Shape * shape)  [private]
```

Gets the brush style of a shape as a string (e.g., "SolidPattern", "NoBrush").

Parameters

shape	A constant pointer to the Shape object.

Returns

A std::string representing the brush style.

Exceptions

std::runtime_error	If the brush style is unknown.

8.10.3.12 GetColor()

```
std::string Parser::GetColor (
            const QColor & objectColor)  [private]
```

Converts a QColor object to its string representation (e.g., "red", "blue").

Parameters

objectColor	The QColor to be converted.

Returns

A std::string representing the color name.

Exceptions

std::runtime_error	If the color is not one of the predefined known colors.

8.10.3.13 GetFontStyle()

```
std::string Parser::GetFontStyle (
            const Text * text)  [private]
```

Gets the font style of a Text object as a string (e.g., "StyleNormal", "StyleItalic").

Parameters

text	A constant pointer to the Text object.

Returns

A std::string representing the font style.

Exceptions

std::runtime_error	If the font style is unknown.

8.10.3.14 GetFontWeight()

```
std::string Parser::GetFontWeight (
            const Text * text)  [private]
```

Gets the font weight of a Text object as a string (e.g., "Normal", "Bold").

Parameters

text	A constant pointer to the Text object.

Returns

A std::string representing the font weight.

Exceptions

std::runtime_error	If the font weight is unknown.

8.10.3.15 GetPenCapStyle()

```
std::string Parser::GetPenCapStyle (
            const Shape * shape)  [private]
```

Gets the pen cap style of a shape as a string (e.g., "FlatCap", "RoundCap").

Parameters

shape	A constant pointer to the Shape object.

Returns

A std::string representing the pen cap style.

Exceptions

std::runtime_error	If the pen cap style is unknown.

8.10.3.16 GetPenJoinStyle()

```
std::string Parser::GetPenJoinStyle (
            const Shape * shape)  [private]
```

Gets the pen join style of a shape as a string (e.g., "MiterJoin", "BevelJoin").

Parameters

shape	A constant pointer to the Shape object.

Returns

A std::string representing the pen join style.

Exceptions

std::runtime_error	If the pen join style is unknown.

8.10.3.17 GetPenStyle()

```
std::string Parser::GetPenStyle (
            const Shape * shape)  [private]
```

Gets the pen style of a shape as a string (e.g., "SolidLine", "DashLine").

Parameters

shape	A constant pointer to the Shape object.

Returns

A std::string representing the pen style.

Exceptions

std::runtime_error	If the pen style is unknown.

8.10.3.18 GetShapeDimensions()

```
std::string Parser::GetShapeDimensions (
            const Shape * shape)  [private]
```

Gets the geometric dimensions of a shape as a JSON array string.

The specific dimensions depend on the shape type (e.g., coordinates for Line, points for Polyline/Polygon, x,y,length,width for Rectangle).

Parameters

shape	A constant pointer to the Shape object.

Returns

A std::string representing a JSON array of the shape's dimensions.

Exceptions

std::runtime_error	If the shape type is unknown or casting fails.

8.10.3.19 JsonToShapes()

```
alpha::vector< Shape * > Parser::JsonToShapes (
            const std::string & json)
```

Converts a JSON string representation into a vector of Shape objects.

This is a forward parsing method that takes a JSON string, parses it, and constructs a vector of dynamically allocated Shape objects.

Parameters

json	A constant reference to a string containing the JSON data for shapes.

Returns

An alpha::vector of Shape pointers, each pointing to an instantiated Shape object.

Exceptions

std::runtime_error	If the JSON string is malformed or parsing fails.

8.10.3.20 JsonToTestimonials()

```
QVector< Testimonial > Parser::JsonToTestimonials (
            const std::string & json)  [static]
```

Converts a JSON string representation into a QVector of Testimonial objects.

This static forward parsing method uses Qt's JSON utilities to parse testimonials.

Parameters

json	A constant reference to a string containing the JSON data for testimonials.

Returns

A QVector of Testimonial objects.

Exceptions

std::runtime_error	If JSON parsing fails or the top-level structure is not an array.

8.10.3.21 JsonToUsers()

```
alpha::vector< UserAccount * > Parser::JsonToUsers (
            const std::string & json)
```

Converts a JSON string representation into a vector of UserAccount objects.

This forward parsing method processes a JSON string to create UserAccount objects.

Parameters

json	A constant reference to a string containing the JSON data for user accounts.

Returns

An alpha::vector of UserAccount pointers.

Exceptions

std::runtime_error	If the JSON string is malformed or essential user data is missing.

8.10.3.22 operator=() [1/2]

```
Parser & Parser::operator= (
            const Parser & )  [delete]
```

Deleted copy assignment operator to prevent copying Parser objects.

8.10.3.23 operator=() [2/2]

```
Parser & Parser::operator= (
            Parser && )  [delete]
```

Deleted move assignment operator to prevent moving Parser objects.

8.10.3.24 ParseJsonObject()

```
Parser::MorphicShape Parser::ParseJsonObject (
            const std::string json,
            size_t & index)  [private]
```

Parses a single JSON object (data within '{}') into a MorphicShape.

This helper function processes one JSON object from the input string, extracting key-value pairs and populating a MorphicShape accumulator.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index within the JSON string, modified by the function.

Returns

A MorphicShape object containing the parsed data.

Exceptions

std::runtime_error	If the JSON object is malformed.

8.10.3.25 PrintShapeVector()

```
void Parser::PrintShapeVector (
            const alpha::vector< Shape * > & shapes)
```

Prints properties of shapes in a vector to the console.

Iterates through a vector of Shape pointers and prints their ID, TrackerId, and ShapeType to standard output. Primarily for debugging purposes.

Parameters

shapes	A constant reference to an alpha::vector of Shape pointers.

8.10.3.26 ShapesToJson()

```
std::string Parser::ShapesToJson (
            const alpha::vector< Shape * > & shapes)
```

Converts a vector of Shape pointers into a JSON string representation.

This is a reverse parsing method that takes a vector of Shape objects and serializes them into a JSON formatted string.

Parameters

shapes	A constant reference to an alpha::vector of Shape pointers.

Returns

A string formatted in JSON containing all key:value pairs for the shapes.

8.10.3.27 SkipWhitespace()

```
void Parser::SkipWhitespace (
            const std::string & json,
            size_t & index)  [private]
```

Advances the parsing index past any whitespace characters.

Modifies `index` to point to the next non-whitespace character in the `json` string.

Parameters

json	The JSON string being parsed.
index	A reference to the current parsing index, modified by the function.

8.10.3.28 StringToVector()

```
alpha::vector< int > Parser::StringToVector (
            const std::string & value)  [private]
```

Converts a string representation of a JSON array of integers into an alpha::vector<int>.

Parses a string like "[20, 32, 41, 64]" into a vector of integers.

Parameters

value	The string representation of the integer array.

Returns

An alpha::vector<int> containing the parsed integers.

Exceptions

std::runtime_error	If the string format is invalid.

8.10.3.29 TestimonialsToJson()

```
std::string Parser::TestimonialsToJson (
            const QVector< Testimonial > & testimonials)  [static]
```

Converts a QVector of Testimonial objects into a JSON string representation.

This static reverse parsing method uses Qt's JSON utilities to serialize testimonials.

Parameters

testimonials	A constant reference to a QVector of Testimonial objects.

Returns

A string formatted in JSON representing the testimonials.

8.10.3.30 UpdateAccumulator()

```
void Parser::UpdateAccumulator (
            const std::string & key,
            const std::string & value,
            MorphicShape & tempShape)  [private]
```

Updates a MorphicShape accumulator with a parsed key-value pair.

Converts the string `value` to its appropriate C++ type based on the `key` and stores it in the corresponding member of the `tempShape` accumulator.

Parameters

key	The JSON key as a string.
value	The JSON value as a string.
tempShape	A reference to the MorphicShape accumulator to be updated.

Exceptions

std::runtime_error	If the key is unknown or the value is invalid for the key.

8.10.3.31 UpdateUserAccumulator()

```
void Parser::UpdateUserAccumulator (
            const std::string & key,
            const std::string & value,
            RawUser & acc)  [static], [private]
```

Updates a RawUser accumulator with a parsed key-value pair for user data.

Converts the string `value` to its appropriate C++ type based on the `key` (username, password, admin) and stores it in the `acc` accumulator. Also sets flags in `acc` to indicate which fields were found.

Parameters

key	The JSON key as a string.
value	The JSON value as a string.
acc	A reference to the RawUser accumulator to be updated.

Exceptions

std::runtime_error	If the value for 'admin' is not "true" or "false".

8.10.3.32 UsersToJson()

```
std::string Parser::UsersToJson (
            const alpha::vector< UserAccount * > & users)
```

Converts a vector of UserAccount pointers into a JSON string representation.

This reverse parsing method serializes UserAccount objects into a JSON string.

Parameters

users	A constant reference to an alpha::vector of UserAccount pointers.

Returns

A string formatted in JSON representing the user accounts.

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

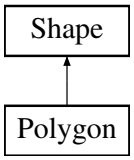
- src/backend/Parser.h
- src/backend/Parser.cpp

8.11 Polygon Class Reference

The Polygon class.

`#include "objects/polygon.h"`

Inheritance diagram for Polygon:



Public Member Functions

- Polygon (string shapeType, QPoint coords, QPen pen, QBrush brush, QPolygon pointsList)

  Polygon Constructor.
- void Draw (QWidget ∗renderArea) override

  Draw - Draws a Polygon at the assigned coords with points in the pointsList.
- void Move (int x, int y) override

  Move - Moves the Polygon to the passed x and y coordinates.
- double Perimeter () const override

  Perimeter - Returns the perimeter of the Polygon.
- double Area () const override

  Area - Returns the area of the Polygon.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the Polygon
- QPolygon getPointsList () const

  getPointsList - Returns the pointsList by value
- void setPointsList (const QPolygon &newPointsList)

  Mutator Functions.
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ∼Shape ()

  ∼Shape Destructor
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

Accessor Functions - Returns the data named after them.

- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- QPolygon pointsList

  list of points

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem
  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems
  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems
  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems
  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems
  vector of QTreeWidgetItem holding data of all brush items*

8.11.1 Detailed Description

The Polygon class.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 Polygon()

```
Polygon::Polygon (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            QPolygon pointsList)
```

Polygon Constructor.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
pointsList	- List of QPoints for each point of the Polygon

8.11.3 Member Function Documentation

8.11.3.1 Area()

```
double Polygon::Area () const  [override], [virtual]
```

Area - Returns the area of the Polygon.

Returns

Implements Shape.

8.11.3.2 Draw()

```
void Polygon::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a Polygon at the assigned coords with points in the pointsList.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.11.3.3 getPointsList()

```
QPolygon Polygon::getPointsList () const
```

getPointsList - Returns the pointsList by value

Returns

8.11.3.4 isPointInside()

```
bool Polygon::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the Polygon

Parameters

point	- point being read

Returns

Implements Shape.

8.11.3.5 Move()

```
void Polygon::Move (
            int x,
            int y)  [override], [virtual]
```

Move - Moves the Polygon to the passed x and y coordinates.

Parameters

x	- x coordinate
y	- y coordinate

Implements Shape.

8.11.3.6 Perimeter()

```
double Polygon::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimeter of the Polygon.

Returns

Implements Shape.

8.11.3.7 setPointsList()

```
void Polygon::setPointsList (
            const QPolygon & newPointsList)
```

Mutator Functions.

setPointsList - Sets the pointsList to the passed newPointsList

Parameters

newPointsList	- New list of points for pointsList to take

8.11.3.8 setX()

```
void Polygon::setX (
            int newX)
```

Implements Shape.

8.11.3.9 setY()

```
void Polygon::setY (
            int newY)
```

Implements Shape.

8.11.4 Member Data Documentation

8.11.4.1 pointsList

```
QPolygon Polygon::pointsList  [private]
```

list of points

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

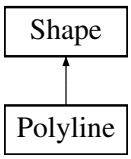
- src/objects/polygon.h
- src/objects/polygon.cpp

8.12 Polyline Class Reference

The Polyline class.

```
#include "objects/polyline.h"
```

Inheritance diagram for Polyline:



Public Member Functions

- Polyline (string shapeType, QPoint coords, QPen pen, QBrush brush, QPolygon pointsList)

  Polyline.
- void Draw (QWidget *renderArea) override

  Draw - Draws a Polyline at the assigne coords with points in the pointsList.
- void Move (int x, int y) override

  Move - Moves the Polyline to the passed x and y coordinates.
- double Perimeter () const override

  Perimeter - Returns the perimater of the Polyline.
- double Area () const override

  Area - Returns 0 (Needs to have implementation to avoid being seen as virtual.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the Polyline
- QPolygon getPointsList () const

  getPointsList - Returns the pointsList by value
- void setPointsList (const QPolygon &newPointsList)

  Mutator Functions.
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ~Shape ()

  ~Shape Destructor
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- QPolygon pointsList
  list of points

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem
  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems
  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems
  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems
  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems
  vector of QTreeWidgetItem holding data of all brush items*

8.12.1 Detailed Description

The Polyline class.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Polyline()

```
Polyline::Polyline (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            QPolygon pointsList)
```

Polyline.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
pointsList	- List of QPoints for each point of the Polyline

8.12.3 Member Function Documentation

8.12.3.1 Area()

```
double Polyline::Area () const  [inline], [override], [virtual]
```

Area - Returns 0 (Needs to have implementation to avoid being seen as virtual.

Returns

Implements Shape.

8.12.3.2 Draw()

```
void Polyline::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a Polyline at the assigne coords with points in the pointsList.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.12.3.3 getPointsList()

```
QPolygon Polyline::getPointsList () const
```

getPointsList - Returns the pointsList by value

Returns

8.12.3.4 isPointInside()

```
bool Polyline::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the Polyline

Parameters

point	- points being read

Returns

Implements Shape.

8.12.3.5 Move()

```
void Polyline::Move (
            int x,
            int y)  [override], [virtual]
```

Move - Moves the Polyline to the passed x and y coordinates.

Parameters

x	- x coordinate
y	- y coordinate

Implements Shape.

8.12.3.6 Perimeter()

```
double Polyline::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimater of the Polyline.

Returns

Implements Shape.

8.12.3.7 setPointsList()

```
void Polyline::setPointsList (
            const QPolygon & newPointsList)
```

Mutator Functions.

setPointsList - Sets the pointsList to the passed newPointsList

Parameters

newPointsList	- New list of points for pointsList to take

8.12.3.8 setX()

```
void Polyline::setX (
            int newX)
```

Implements Shape.

8.12.3.9 setY()

```
void Polyline::setY (
            int newY)
```

Implements Shape.

8.12.4 Member Data Documentation

8.12.4.1 pointsList

```
QPolygon Polyline::pointsList  [private]
```

list of points

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

- src/objects/polyline.h
- src/objects/polyline.cpp

8.13 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN11UserManagerE_t Struct Reference

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

- src/backend/moc_UserManager.cpp

8.14 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN9ApiClientE_t Struct Reference

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

- src/backend/moc_ApiClient.cpp

8.15 Parser::RawUser Struct Reference

Internal accumulator structure for parsing user account data from JSON.

Public Attributes

- QString username
  The username of the user.
- QString password
  The password of the user.
- bool admin = false
  Flag indicating if the user has admin privileges.
- bool hasUsername = false
  Flag indicating if the username was found during parsing.
- bool hasPassword = false
  Flag indicating if the password was found during parsing.
- bool hasAdmin = false
  Flag indicating if the admin status was found during parsing.

8.15.1 Detailed Description

Internal accumulator structure for parsing user account data from JSON.

This structure temporarily holds the properties of a user account as they are parsed from a JSON object, along with flags to track if fields were found.

8.15.2 Member Data Documentation

8.15.2.1 admin

```
bool Parser::RawUser::admin = false
```

Flag indicating if the user has admin privileges.

8.15.2.2 hasAdmin

```
bool Parser::RawUser::hasAdmin = false
```

Flag indicating if the admin status was found during parsing.

8.15.2.3 hasPassword

```
bool Parser::RawUser::hasPassword = false
```

Flag indicating if the password was found during parsing.

8.15.2.4 hasUsername

```
bool Parser::RawUser::hasUsername = false
```

Flag indicating if the username was found during parsing.

8.15.2.5 password

```
QString Parser::RawUser::password
```

The password of the user.

8.15.2.6 username

```
QString Parser::RawUser::username
```

The username of the user.

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

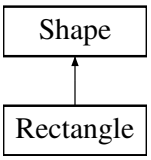
- src/backend/Parser.h

8.16 Rectangle Class Reference

The Rectangle class.

`#include "objects/rectangle.h"`

Inheritance diagram for Rectangle:



Public Member Functions

- Rectangle (string shapeType, QPoint coords, QPen pen, QBrush brush, int length, int width)

  Rectangle.
- void Draw (QWidget ∗renderArea) override

  Draw - Draws a Rectangle at the assigned coords.
- double Perimeter () const override

  Perimeter - Returns the perimeter of the Rectangle.
- double Area () const override

  Area - Returns the area of the Rectangle.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the Rectangle
- int getLength () const

  Accessor Functions.
- int getWidth () const
- void setLength (int newLength)

  Mutator Functions.
- void setWidth (int newWidth)
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ∼Shape ()

  ∼*Shape Destructor*
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.

- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- int length

  length of the rectangle
- int width

  width of the rectangle

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.16.1 Detailed Description

The Rectangle class.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 Rectangle()

```
Rectangle::Rectangle (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            int length,
            int width)
```

Rectangle.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
length	- Length of the Rectangle
width	- Width of the Rectangle

8.16.3 Member Function Documentation

8.16.3.1 Area()

```
double Rectangle::Area () const  [override], [virtual]
```

Area - Returns the area of the Rectangle.

Returns

Implements Shape.

8.16.3.2 Draw()

```
void Rectangle::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a Rectangle at the assigned coords.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.16.3.3 getLength()

```
int Rectangle::getLength () const
```

Accessor Functions.

8.16.3.4 getWidth()

```
int Rectangle::getWidth () const
```

8.16.3.5 isPointInside()

```
bool Rectangle::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the Rectangle

Parameters

point	- point being read

Returns

Implements Shape.

8.16.3.6 Perimeter()

```
double Rectangle::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimeter of the Rectangle.

Returns

Implements Shape.

8.16.3.7 setLength()

```
void Rectangle::setLength (
            int newLength)
```

Mutator Functions.

8.16.3.8 setWidth()

```
void Rectangle::setWidth (
            int newWidth)
```

8.16.3.9 setX()

```
void Rectangle::setX (
            int newX)
```

Implements Shape.

8.16.3.10 setY()

```
void Rectangle::setY (
            int newY)
```

Implements Shape.

8.16.4 Member Data Documentation

8.16.4.1 length

```
int Rectangle::length  [private]
```

length of the rectangle

8.16.4.2 width

```
int Rectangle::width  [private]
```

width of the rectangle

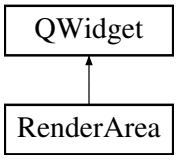
The documentation for this class was generated from the following files:

- src/objects/rectangle.h
- src/objects/rectangle.cpp

8.17 RenderArea Class Reference

`#include <renderarea.h>`

Inheritance diagram for RenderArea:



Public Member Functions

- RenderArea (QWidget *parent=nullptr)
- void setRenderShapes (const alpha::vector< Shape * > *renderShapes)
- const alpha::vector< Shape * > & getShapes () const
- void mousePressEvent (QMouseEvent *event) override
- void mouseMoveEvent (QMouseEvent *event) override
- void mouseDoubleClickEvent (QMouseEvent *event) override
- void mouseReleaseEvent (QMouseEvent *event) override
- void setShapeSelectedIndex (int newIndex)
- void resetSelection ()
- void setEditPrivileges (bool edit)
- void updateShapeDisplayCoords (Shape *item, const QPoint &position) const
- int getShapeSelected () const
- int getShapeSelectedIndex () const

Protected Member Functions

- void paintEvent (QPaintEvent *event) override

Private Attributes

- const alpha::vector< Shape * > * renderShapes
- int shapeSelectedIndex
- bool allowEditing = false

8.17.1 Constructor & Destructor Documentation

8.17.1.1 RenderArea()

```
RenderArea::RenderArea (
            QWidget * parent = nullptr)
```

8.17.2 Member Function Documentation

8.17.2.1 getShapes()

```
const alpha::vector< Shape * > & RenderArea::getShapes () const
```

8.17.2.2 getShapeSelected()

```
int RenderArea::getShapeSelected () const
```

8.17.2.3 getShapeSelectedIndex()

```
int RenderArea::getShapeSelectedIndex () const
```

8.17.2.4 mouseDoubleClickEvent()

```
void RenderArea::mouseDoubleClickEvent (
            QMouseEvent * event)  [override]
```

8.17.2.5 mouseMoveEvent()

```
void RenderArea::mouseMoveEvent (
            QMouseEvent * event)  [override]
```

8.17.2.6 mousePressEvent()

```
void RenderArea::mousePressEvent (
            QMouseEvent * event)  [override]
```

8.17.2.7 mouseReleaseEvent()

```
void RenderArea::mouseReleaseEvent (
            QMouseEvent * event)  [override]
```

8.17.2.8 paintEvent()

```
void RenderArea::paintEvent (
            QPaintEvent * event)  [override], [protected]
```

8.17.2.9 resetSelection()

```
void RenderArea::resetSelection ()
```

8.17.2.10 setEditPrivileges()

```
void RenderArea::setEditPrivileges (
            bool edit)
```

8.17.2.11 setRenderShapes()

```
void RenderArea::setRenderShapes (
            const alpha::vector< Shape * > * renderShapes)
```

8.17.2.12 setShapeSelectedIndex()

```
void RenderArea::setShapeSelectedIndex (
            int newIndex)
```

8.17.2.13 updateShapeDisplayCoords()

```
void RenderArea::updateShapeDisplayCoords (
            Shape * item,
            const QPoint & position) const
```

8.17.3 Member Data Documentation

8.17.3.1 allowEditing

```
bool RenderArea::allowEditing = false  [private]
```

8.17.3.2 renderShapes

```
const alpha::vector<Shape*>* RenderArea::renderShapes  [private]
```

8.17.3.3 shapeSelectedIndex

```
int RenderArea::shapeSelectedIndex  [private]
```

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

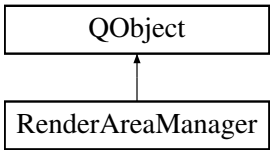
- src/frontend/renderarea.h
- src/frontend/renderarea.cpp

8.18 RenderAreaManager Class Reference

Manages the shapes to be rendered and interacts with the backend API.

`#include <RenderAreaManager.h>`

Inheritance diagram for RenderAreaManager:



Signals

Render Area Signals

Signals emitted by RenderAreaManager to communicate with other parts of the application.

These signals are primarily used to notify the UI about changes in the shape data or to convey status messages.

- void renderAreaChanged ()
  Emitted when the render area has changed and needs redrawing.
- void renderAreaNotChanged (const QString &message)
  Emitted when an operation completes without changing the render area, often with a message.
- void statusMessage (const QString &message)
  Emitted to provide a status message to the user.

Public Member Functions

Core Management Functions

Public methods for RenderAreaManager.

This section includes constructors, destructors, and primary interface functions for managing and accessing shape data.

- RenderAreaManager (QObject *parent=nullptr)
  Constructs a RenderAreaManager object and connects API client signals.
- ~RenderAreaManager ()
  Destroys the RenderAreaManager object and deallocates rendered shapes.
- alpha::vector< Shape * > * getShapesRef ()
  Retrieves a pointer to the vector of rendered shapes.

Shape Manipulation

Functions for manipulating shapes within the RenderAreaManager.

These methods handle the creation, modification, and deletion of shapes. Most of these operations will trigger a signal to update the rendering area and will attempt to save the changes to the backend.

- void addShape (Shape *shape)
  Adds a new shape to the rendering list and triggers a save.
- void modifyShape (Shape *shape, QString key, int value)
  Modifies a property of an existing shape based on a key-value pair.
- void modifyDisplayedText (Text *obj, QString newText)
  Modifies the displayed text of a Text shape.
- void deleteShape (const int trackerId)
  Deletes a shape identified by its tracker ID.
- void deleteAllShapes ()
  Requests the deletion of all shapes from the backend.
- void loadShapes ()
  Initiates loading shapes from the backend webservice.
- void saveShapes ()
  Initiates saving the current shapes to the backend webservice.

Private Slots

API Client Response Handlers

Private slots for handling responses from the ApiClient.

These slots are connected to signals from the ApiClient and process the results of asynchronous network operations (GET, POST, DELETE).

- void onGoodGetResponse (const QString &json)
  Slot to handle successful GET responses from the API client.
- void onBadGetResponse (const QString &errorMsg)
  Slot to handle unsuccessful GET responses from the API client.
- void onGoodPostResponse ()
  Slot to handle successful POST responses from the API client.
- void onBadPostResponse (const QString &errorMsg)
  Slot to handle unsuccessful POST responses from the API client.
- void onGoodDeleteResponse ()
  Slot to handle successful DELETE responses from the API client.
- void onBadDeleteResponse (const QString &errorMsg)
  Slot to handle unsuccessful DELETE responses from the API client.

Private Attributes

Internal Data and Utilities

Private members for storing shape data and utility objects.

This section includes the storage for shapes and instances of helper classes like ApiClient and Parser.

- alpha::vector< Shape * > renderedShapes
  Vector storing pointers to all shapes currently managed.
- ApiClient client
  Client for making API requests to the backend.
- Parser parse
  Parser for converting shapes to/from JSON.

8.18.1 Detailed Description

Manages the shapes to be rendered and interacts with the backend API.

This class is responsible for adding, modifying, deleting, loading, and saving shapes. It communicates with an ApiClient to persist shape data and uses a Parser for serialization and deserialization. It also emits signals when the render area needs to be updated or when status messages should be displayed.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 RenderAreaManager()

```
RenderAreaManager::RenderAreaManager (
            QObject * parent = nullptr)  [explicit]
```

Constructs a RenderAreaManager object and connects API client signals.

8.18.2.2 ~RenderAreaManager()

```
RenderAreaManager::~RenderAreaManager ()
```

Destroys the RenderAreaManager object and deallocates rendered shapes.

8.18.3 Member Function Documentation

8.18.3.1 addShape()

```
void RenderAreaManager::addShape (
            Shape * shape)
```

Adds a new shape to the rendering list and triggers a save.

8.18.3.2 deleteAllShapes()

```
void RenderAreaManager::deleteAllShapes ()
```

Requests the deletion of all shapes from the backend.

8.18.3.3 deleteShape()

```
void RenderAreaManager::deleteShape (
            const int trackerId)
```

Deletes a shape identified by its tracker ID.

8.18.3.4 getShapesRef()

```
alpha::vector< Shape * > * RenderAreaManager::getShapesRef ()
```

Retrieves a pointer to the vector of rendered shapes.

8.18.3.5 loadShapes()

```
void RenderAreaManager::loadShapes ()
```

Initiates loading shapes from the backend webservice.

8.18.3.6 modifyDisplayedText()

```
void RenderAreaManager::modifyDisplayedText (
            Text * obj,
            QString newText)
```

Modifies the displayed text of a Text shape.

8.18.3.7 modifyShape()

```
void RenderAreaManager::modifyShape (
            Shape * shape,
            QString key,
            int value)
```

Modifies a property of an existing shape based on a key-value pair.

8.18.3.8 onBadDeleteResponse

```
void RenderAreaManager::onBadDeleteResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot to handle unsuccessful DELETE responses from the API client.

8.18.3.9 onBadGetResponse

```
void RenderAreaManager::onBadGetResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot to handle unsuccessful GET responses from the API client.

8.18.3.10 onBadPostResponse

```
void RenderAreaManager::onBadPostResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot to handle unsuccessful POST responses from the API client.

8.18.3.11 onGoodDeleteResponse

```
void RenderAreaManager::onGoodDeleteResponse ()  [private], [slot]
```

Slot to handle successful DELETE responses from the API client.

8.18.3.12 onGoodGetResponse

```
void RenderAreaManager::onGoodGetResponse (
            const QString & json)  [private], [slot]
```

Slot to handle successful GET responses from the API client.

8.18.3.13 onGoodPostResponse

```
void RenderAreaManager::onGoodPostResponse ()  [private], [slot]
```

Slot to handle successful POST responses from the API client.

8.18.3.14 renderAreaChanged

```
void RenderAreaManager::renderAreaChanged ()  [signal]
```

Emitted when the render area has changed and needs redrawing.

8.18.3.15 renderAreaNotChanged

```
void RenderAreaManager::renderAreaNotChanged (
            const QString & message)  [signal]
```

Emitted when an operation completes without changing the render area, often with a message.

8.18.3.16 saveShapes()

```
void RenderAreaManager::saveShapes ()
```

Initiates saving the current shapes to the backend webservice.

8.18.3.17 statusMessage

```
void RenderAreaManager::statusMessage (
            const QString & message)  [signal]
```

Emitted to provide a status message to the user.

8.18.4 Member Data Documentation

8.18.4.1 client

```
ApiClient RenderAreaManager::client  [private]
```

Client for making API requests to the backend.

8.18.4.2 parse

```
Parser RenderAreaManager::parse  [private]
```

Parser for converting shapes to/from JSON.

8.18.4.3 renderedShapes

```
alpha::vector<Shape*> RenderAreaManager::renderedShapes  [private]
```

Vector storing pointers to all shapes currently managed.

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

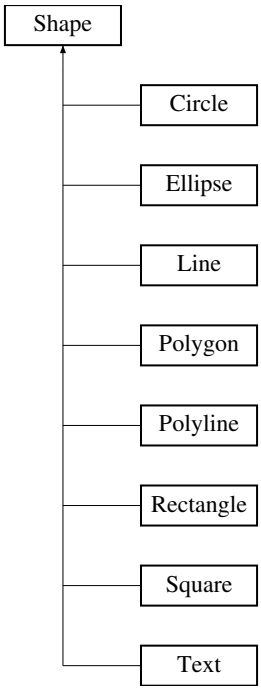
- src/backend/RenderAreaManager.h
- src/backend/RenderAreaManager.cpp

8.19 Shape Interface Reference

The Shape Abstract Base Class.

`#include "objects/shape.h"`

Inheritance diagram for Shape:



Public Member Functions

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ~Shape ()

  ~Shape Destructor
- virtual void Draw (QWidget *renderArea)=0

  Draw - Draws the shape to the associated renderArea.
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- virtual double Perimeter () const =0

  Perimeter - Returns the perimeter of the shape.
- virtual double Area () const =0

  Area - Returns the area of the shape.
- virtual bool isPointInside (const QPoint &point) const =0

  isPointInside - Returns true if point is inside the shape
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.

- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setX (int x)
- void setY (int y)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Static Public Attributes

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

Private Member Functions

- Shape (Shape &shape)=delete
- Shape & operator= (Shape &object)=delete

Private Attributes

- const int shapeId

  int representing the id of a shape, each shapeType is given one id, Line = 1, Polyline = 2, etc.
- int trackerId

  int representing the unique id of each shape, each individual shape has its own trackerId
- string shapeType

  string representing the type of shape, (Line, Circle, etc)
- QPen pen

  QPen for the outline.
- QBrush brush

  QBrush for the fill.
- QPoint coords

  QPoint for the coordinates of the shape.
- QPainter painter

  QPainter used to paint the shape onto a rendering area.
- bool isSelected = false

  Shows the state of the shape, if it is currently selected by the user.

Friends

- bool operator== (const Shape &shape1, const Shape &shape2)

  operator == - Overloaded equality operator for comparing two shapeId's
- bool operator< (const Shape &shape1, const Shape &shape2)

  operator < - Overloaded less than operator for comparing two shapeId's

8.19.1 Detailed Description

The Shape Abstract Base Class.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 Shape() [1/2]

```
Shape::Shape (
            int shapeId,
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush)
```

Shape Constructor.

Parameters

shapeType	- string representing shape type, (Line, Circle, etc)
coords	- QPoint with coordinates of the shape
pen	- QPen for the outline of the shape
brush	- QBrush for the fill of the shape

8.19.2.2 ~Shape()

```
Shape::~Shape ()  [virtual]
```

~Shape Destructor

8.19.2.3 Shape() [2/2]

```
Shape::Shape (
            Shape & shape)  [private], [delete]
```

8.19.3 Member Function Documentation

8.19.3.1 allocateTrackerId()

```
void Shape::allocateTrackerId (
            int shapeId)
```

8.19.3.2 Area()

```
virtual double Shape::Area () const  [pure virtual]
```

Area - Returns the area of the shape.

Returns

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.3.3 CreateBrushChild()

```
void Shape::CreateBrushChild ()
```

CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.

8.19.3.4 CreateParentItem()

```
void Shape::CreateParentItem ()
```

CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.

8.19.3.5 CreatePenChild()

```
void Shape::CreatePenChild ()
```

CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.

8.19.3.6 CreatePointsChild()

```
void Shape::CreatePointsChild (
            const int POINTS_NUM)
```

CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.

Parameters

POINTS_NUM	- Number of points being added

8.19.3.7 Draw()

```
virtual void Shape::Draw (
            QWidget * renderArea)  [pure virtual]
```

Draw - Draws the shape to the associated renderArea.

Parameters

renderArea	- QWidget to be drawn on

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.3.8 getBrush()

```
QBrush Shape::getBrush () const
```

8.19.3.9 getBrushColor()

```
QColor Shape::getBrushColor () const
```

8.19.3.10 getBrushItems()

```
alpha::vector< QTreeWidgetItem * > & Shape::getBrushItems ()
```

8.19.3.11 getBrushItemsEnd()

```
int Shape::getBrushItemsEnd () const
```

Returns the dereferenced penItems.end() - 1 for the last initialized element of the vector

8.19.3.12 getBrushStyle()

```
BrushStyle Shape::getBrushStyle () const
```

8.19.3.13 getChildEnd()

```
int Shape::getChildEnd () const
```

8.19.3.14 getChildItems()

```
alpha::vector< QTreeWidgetItem * > & Shape::getChildItems ()
```

8.19.3.15 getPainter()

```
QPainter & Shape::getPainter ()
```

8.19.3.16 getParentItem()

```
QTreeWidgetItem * Shape::getParentItem ()
```

8.19.3.17 getPen()

```
QPen Shape::getPen () const
```

8.19.3.18 getPenCapStyle()

```
PenCapStyle Shape::getPenCapStyle () const
```

8.19.3.19 getPenColor()

```
QColor Shape::getPenColor () const
```

8.19.3.20 getPenItems()

```
alpha::vector< QTreeWidgetItem * > & Shape::getPenItems ()
```

8.19.3.21 getPenItemsEnd()

```
int Shape::getPenItemsEnd () const
```

Returns the dereferenced childItems.end() - 1 for the last initialized element of the vector

8.19.3.22 getPenJoinStyle()

```
PenJoinStyle Shape::getPenJoinStyle () const
```

8.19.3.23 getPenStyle()

```
PenStyle Shape::getPenStyle () const
```

8.19.3.24 getPenWidth()

```
int Shape::getPenWidth () const
```

8.19.3.25 getPoints()

```
QPoint Shape::getPoints () const
```

8.19.3.26 getPointsItems()

```
alpha::vector< QTreeWidgetItem * > & Shape::getPointsItems ()
```

8.19.3.27 getSelected()

```
bool Shape::getSelected () const
```

8.19.3.28 getShapeId()

```
int Shape::getShapeId () const
```

Accessor Functions - Returns the data named after them.

8.19.3.29 getShapeType()

```
string Shape::getShapeType () const
```

8.19.3.30 getTrackerId()

```
int Shape::getTrackerId () const
```

8.19.3.31 getX()

```
int Shape::getX () const
```

8.19.3.32 getY()

```
int Shape::getY () const
```

8.19.3.33 isPointInside()

```
virtual bool Shape::isPointInside (
            const QPoint & point) const  [pure virtual]
```

isPointInside - Returns true if point is inside the shape

Parameters

point	- QPoint being checked

Returns

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.3.34 Move()

```
void Shape::Move (
            int x,
            int y)  [virtual]
```

Move - Moves the shape to the x and y coords.

Parameters

x	- x coordinate
y	- y coordinate

Implemented in Line, Polygon, and Polyline.

8.19.3.35 operator=()

```
Shape & Shape::operator= (
            Shape & object)  [private], [delete]
```

8.19.3.36 Perimeter()

```
virtual double Shape::Perimeter () const  [pure virtual]
```

Perimeter - Returns the perimeter of the shape.

Returns

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.3.37 setBrush()

```
void Shape::setBrush (
            GlobalColor brushColor,
            BrushStyle brushStyle)
```

8.19.3.38 setInternalBrush()

```
QBrush & Shape::setInternalBrush ()
```

8.19.3.39 setInternalPen()

```
QPen & Shape::setInternalPen ()
```

8.19.3.40 setPen()

```
void Shape::setPen (
            GlobalColor penColor,
            int penWidth,
            PenStyle penStyle,
            PenCapStyle penCapStyle,
            PenJoinStyle penJoinStyle)
```

8.19.3.41 setSelected()

```
void Shape::setSelected (
            bool selected)
```

8.19.3.42 setShapeType()

```
void Shape::setShapeType (
            string shapeType)
```

Accessor Functions.

Returns the dereferenced brushItems.end() - 1 for the last initialized element of the vector Mutator Functions - Sets the data of the item to the passed param

8.19.3.43 setTrackerId()

```
void Shape::setTrackerId (
            int trackerId)
```

8.19.3.44 setX()

```
void Shape::setX (
            int x)
```

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.3.45 setY()

```
void Shape::setY (
            int y)
```

Implemented in Circle, Ellipse, Line, Polygon, Polyline, Rectangle, Square, and Text.

8.19.4 Friends And Related Symbol Documentation

8.19.4.1 operator<

```
bool operator< (
            const Shape & shape1,
            const Shape & shape2)  [friend]
```

operator < - Overloaded less than operator for comparing two shapeId's

Parameters

shape1	
shape2	

Returns

8.19.4.2 operator==

```
bool operator== (
            const Shape & shape1,
            const Shape & shape2)  [friend]
```

operator == - Overloaded equality operator for comparing two shapeId's

Parameters

shape1	
shape2	

Returns

8.19.5 Member Data Documentation

8.19.5.1 brush

```
QBrush Shape::brush  [private]
```

QBrush for the fill.

8.19.5.2 brushItems

```
alpha::vector<QTreeWidgetItem*> Shape::brushItems  [protected]
```

vector of QTreeWidgetItem∗ holding data of all brush items

8.19.5.3 childItems

```
alpha::vector<QTreeWidgetItem*> Shape::childItems  [protected]
```

vector of QTreeWidgetItem∗ holding data of all child items in parentItem

8.19.5.4 coords

```
QPoint Shape::coords  [private]
```

QPoint for the coordinates of the shape.

8.19.5.5 isSelected

```
bool Shape::isSelected = false  [private]
```

Shows the state of the shape, if it is currently selected by the user.

8.19.5.6 nextTracker

```
int Shape::nextTracker = {}  [static]
```

8.19.5.7 painter

```
QPainter Shape::painter  [private]
```

QPainter used to paint the shape onto a rendering area.

8.19.5.8 parentItem

```
QTreeWidgetItem* Shape::parentItem  [protected]
```

Mutator Functions.

QTreeWidgetItem∗ holding treeWidget data of each shape

8.19.5.9 pen

```
QPen Shape::pen  [private]
```

QPen for the outline.

8.19.5.10 penItems

```
alpha::vector<QTreeWidgetItem*> Shape::penItems  [protected]
```

vector of QTreeWidgetItem∗ holding data of all pen items

8.19.5.11 pointsItems

```
alpha::vector<QTreeWidgetItem*> Shape::pointsItems  [protected]
```

vector of QTreeWidgetItem∗ holding data of all points (besides coords) in parentItem

8.19.5.12 shapeId

```
const int Shape::shapeId  [private]
```

int representing the id of a shape, each shapeType is given one id, Line = 1, Polyline = 2, etc.

8.19.5.13 shapeType

```
string Shape::shapeType  [private]
```

string representing the type of shape, (Line, Circle, etc)

8.19.5.14 trackerId

```
int Shape::trackerId  [private]
```

int representing the unique id of each shape, each individual shape has its own trackerId

8.19.5.15 trackersInUse

```
bool Shape::trackersInUse = {}  [static]
```

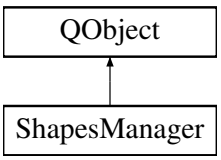
The documentation for this interface was generated from the following files:

- src/objects/shape.h
- src/objects/shape.cpp

8.20 ShapesManager Class Reference

```
#include <ShapesManager.h>
```

Inheritance diagram for ShapesManager:



Signals

- void shapesChanged ()
- void shapesNotChanged (const QString &messsage)
- void statusMessage (const QString &message)

Public Member Functions

- ShapesManager (QObject *parent=nullptr)

  Default Constructor.
- ~ShapesManager ()
- alpha::vector< Shape * > * getShapesRef ()

  Returns the shapes vector as a reference.
- void addShape (Shape *shape)

  These functions are used to add, delete, and load shapes in the ShapesManager.
- void modifyShape (Shape *shape)
- void deleteShape (const int trackerId)
- void deleteAllShapes ()
- void loadShapes ()
- void saveShapes ()

Private Slots

- void onGoodGetResponse (const QString &json)
- void onBadGetResponse (const QString &errorMsg)
- void onGoodPostResponse ()
- void onBadPostResponse (const QString &errorMsg)
- void onGoodDeleteResponse ()
- void onBadDeleteResponse (const QString &errorMsg)

Private Attributes

- alpha::vector< Shape * > shapes
- ApiClient client
- Parser parse

8.20.1 Constructor & Destructor Documentation

8.20.1.1 ShapesManager()

```
ShapesManager::ShapesManager (
            QObject * parent = nullptr)  [explicit]
```

Default Constructor.

This constructor initializes the ShapesManager object and connects to the ApiClient signals for handling shape data.

Parameters

parent	- any QObject to tie this instantiation to ensure automatic deletion

8.20.1.2 ~ShapesManager()

```
ShapesManager::~ShapesManager ()
```

8.20.2 Member Function Documentation

8.20.2.1 addShape()

```
void ShapesManager::addShape (
            Shape * shape)
```

These functions are used to add, delete, and load shapes in the ShapesManager.

All of these functions emit the shapesChanged() signal to notify the frontend that the shapes have changed and need to be redrawn except for saveShapes().

8.20.2.2 deleteAllShapes()

```
void ShapesManager::deleteAllShapes ()
```

8.20.2.3 deleteShape()

```
void ShapesManager::deleteShape (
            const int trackerId)
```

8.20.2.4 getShapesRef()

```
alpha::vector< Shape * > * ShapesManager::getShapesRef ()
```

Returns the shapes vector as a reference.

8.20.2.5 loadShapes()

```
void ShapesManager::loadShapes ()
```

8.20.2.6 modifyShape()

```
void ShapesManager::modifyShape (
            Shape * shape)
```

8.20.2.7 onBadDeleteResponse

```
void ShapesManager::onBadDeleteResponse (
            const QString & errorMsg)  [private], [slot]
```

8.20.2.8 onBadGetResponse

```
void ShapesManager::onBadGetResponse (
            const QString & errorMsg)  [private], [slot]
```

8.20.2.9 onBadPostResponse

```
void ShapesManager::onBadPostResponse (
            const QString & errorMsg)  [private], [slot]
```

8.20.2.10 onGoodDeleteResponse

```
void ShapesManager::onGoodDeleteResponse ()  [private], [slot]
```

8.20.2.11 onGoodGetResponse

```
void ShapesManager::onGoodGetResponse (
            const QString & json)  [private], [slot]
```

These slot functions receive signals from the ApiClient class and hold the code on what to do next after one of these responses from the webservice.

8.20.2.12 onGoodPostResponse

```
void ShapesManager::onGoodPostResponse ()  [private], [slot]
```

8.20.2.13 saveShapes()

```
void ShapesManager::saveShapes ()
```

8.20.2.14 shapesChanged

```
void ShapesManager::shapesChanged ()  [signal]
```

These signals are meant to connect to the Shapes window in the frontend. -shapesChanged(): signal for when the shapes vector is changed in any way so that the frontend knows to refresh the window and redraw the shapes -statusMessage(): signal for slot functions below. It passes the message to the frontend so it can display a popup saying the shapes were saved successfully, or whatever the message is for the user.

8.20.2.15 shapesNotChanged

```
void ShapesManager::shapesNotChanged (
            const QString & messsage)  [signal]
```

8.20.2.16 statusMessage

```
void ShapesManager::statusMessage (
            const QString & message)  [signal]
```

8.20.3 Member Data Documentation

8.20.3.1 client

```
ApiClient ShapesManager::client  [private]
```

8.20.3.2 parse

```
Parser ShapesManager::parse  [private]
```

8.20.3.3 shapes

```
alpha::vector<Shape*> ShapesManager::shapes  [private]
```

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

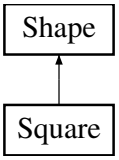
- src/backend/ShapesManager.h
- src/backend/ShapesManager.cpp

8.21 Square Class Reference

The Square class.

```
#include "objects/square.h"
```

Inheritance diagram for Square:



Public Member Functions

- Square (string shapeType, QPoint coords, QPen pen, QBrush brush, int length)

  Square.
- void Draw (QWidget *renderArea) override

  Draw - Draws a Square at the assigned coords.
- double Perimeter () const override

  Perimeter - Returns the perimater of the Square.
- double Area () const override

  Area - Returns the area of the Square.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the Square
- int getLength () const

  Accessor Functions.
- void setLength (int newLength)

  Mutator Functions.
- void setX (int newX)
- void setY (int newY)

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ~Shape ()

  ~Shape Destructor
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const
- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- int length

  side length of the Square

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.21.1 Detailed Description

The Square class.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 Square()

```
Square::Square (
            string shapeType,
            QPoint coords,
            QPen pen,
            QBrush brush,
            int length)
```

Square.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
pen	- Used in Shape constructor MIL
brush	- Used in Shape constructor MIL
length	- Side length of the Square

8.21.3 Member Function Documentation

8.21.3.1 Area()

```
double Square::Area () const  [override], [virtual]
```

Area - Returns the area of the Square.

Returns

Implements Shape.

8.21.3.2 Draw()

```
void Square::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws a Square at the assigned coords.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.21.3.3 getLength()

```
int Square::getLength () const
```

Accessor Functions.

8.21.3.4 isPointInside()

```
bool Square::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the Square

Parameters

point	- point being read

Returns

Implements Shape.

8.21.3.5 Perimeter()

```
double Square::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimater of the Square.

Returns

Implements Shape.

8.21.3.6 setLength()

```
void Square::setLength (
            int newLength)
```

Mutator Functions.

8.21.3.7 setX()

```
void Square::setX (
            int newX)
```

Implements Shape.

8.21.3.8 setY()

```
void Square::setY (
            int newY)
```

Implements Shape.

8.21.4 Member Data Documentation

8.21.4.1 length

```
int Square::length  [private]
```

side length of the Square

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

- src/objects/square.h
- src/objects/square.cpp

8.22 Testimonial Class Reference

Represents a user testimonial.

`#include <Testimonial.h>`

Public Member Functions

Constructor

Constructs a Testimonial object.

Initializes a new testimonial with the provided author, content, and guest status. The timestamp is automatically set to the current date and time, and `isSatisfactory` defaults to true.

- Testimonial (const QString &author="", const QString &content="", bool isGuest=true)
  Constructor with optional parameters.

Getters

Public getters for accessing testimonial data.

These methods provide read-only access to the private member variables of the Testimonial object.

- QString getAuthor () const
  Returns the author of the testimonial.
- QString getContent () const
  Returns the content of the testimonial.
- QDateTime getTimestamp () const
  Returns the timestamp when the testimonial was submitted.
- bool isGuest () const
  Returns true if the author is a guest, false otherwise.
- bool isSatisfactory () const
  Returns true if the testimonial is marked as satisfactory for display.

Setters

Public setter for the satisfaction flag.

This method allows modification of the `m_isSatisfactory` flag, typically used for moderation purposes.

- void setIsSatisfactory (bool value)
  Sets the satisfactory status of the testimonial.

Private Attributes

Member Data

Private member variables storing testimonial attributes.

These variables store the core information for each testimonial.

- QString m_author
  Name of the person giving the testimonial.
- QString m_content
  Actual testimonial text.
- QDateTime m_timestamp
  Timestamp of when the testimonial was submitted.
- bool m_isGuest
  Flag indicating if the author is a guest or a registered user.
- bool m_isSatisfactory
  Flag indicating if the testimonial is deemed satisfactory for display (e.g., after moderation).

JSON Conversion

JSON conversion methods for database storage and retrieval.

These static and member methods handle the serialization of a Testimonial object to a QJsonObject and deserialization from a QJsonObject.

- QJsonObject toJson () const

  Converts the Testimonial object to a QJsonObject.
- static Testimonial fromJson (const QJsonObject &json)

  Creates a Testimonial object from a QJsonObject.

8.22.1 Detailed Description

Represents a user testimonial.

This class encapsulates the data for a testimonial, including the author's name, the content of the testimonial, a timestamp, whether the author is a guest, and a flag indicating if the testimonial is satisfactory for display. It also provides methods for converting testimonial data to and from JSON format.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 Testimonial()

```
Testimonial::Testimonial (
            const QString & author = "",
            const QString & content = "",
            bool isGuest = true)
```

Constructor with optional parameters.

8.22.3 Member Function Documentation

8.22.3.1 fromJson()

```
Testimonial Testimonial::fromJson (
            const QJsonObject & json)  [static]
```

Creates a Testimonial object from a QJsonObject.

8.22.3.2 getAuthor()

```
QString Testimonial::getAuthor () const
```

Returns the author of the testimonial.

8.22.3.3 getContent()

```
QString Testimonial::getContent () const
```

Returns the content of the testimonial.

8.22.3.4 getTimestamp()

```
QDateTime Testimonial::getTimestamp () const
```

Returns the timestamp when the testimonial was submitted.

8.22.3.5 isGuest()

```
bool Testimonial::isGuest () const
```

Returns true if the author is a guest, false otherwise.

8.22.3.6 isSatisfactory()

```
bool Testimonial::isSatisfactory () const
```

Returns true if the testimonial is marked as satisfactory for display.

8.22.3.7 setIsSatisfactory()

```
void Testimonial::setIsSatisfactory (
            bool value)
```

Sets the satisfactory status of the testimonial.

8.22.3.8 toJson()

```
QJsonObject Testimonial::toJson () const
```

Converts the Testimonial object to a QJsonObject.

8.22.4 Member Data Documentation

8.22.4.1 m_author

```
QString Testimonial::m_author  [private]
```

Name of the person giving the testimonial.

8.22.4.2 m_content

```
QString Testimonial::m_content  [private]
```

Actual testimonial text.

8.22.4.3 m_isGuest

```
bool Testimonial::m_isGuest  [private]
```

Flag indicating if the author is a guest or a registered user.

8.22.4.4 m_isSatisfactory

```
bool Testimonial::m_isSatisfactory  [private]
```

Flag indicating if the testimonial is deemed satisfactory for display (e.g., after moderation).

8.22.4.5 m_timestamp

```
QDateTime Testimonial::m_timestamp  [private]
```

Timestamp of when the testimonial was submitted.

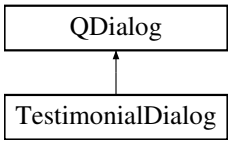
The documentation for this class was generated from the following files:

- src/backend/Testimonial.h
- src/backend/Testimonial.cpp

8.23 TestimonialDialog Class Reference

```
#include <TestimonialDialog.h>
```

Inheritance diagram for TestimonialDialog:



Public Member Functions

- TestimonialDialog (QWidget *parent=nullptr)

Private Slots

- void onSubmit ()
- void onCancel ()

Private Attributes

- QLineEdit * m_authorEdit
- QTextEdit * m_contentEdit
- QCheckBox * m_doNotShowAgain

8.23.1 Constructor & Destructor Documentation

8.23.1.1 TestimonialDialog()

```
TestimonialDialog::TestimonialDialog (
            QWidget * parent = nullptr)  [explicit]
```

8.23.2 Member Function Documentation

8.23.2.1 onCancel

```
void TestimonialDialog::onCancel ()  [private], [slot]
```

8.23.2.2 onSubmit

```
void TestimonialDialog::onSubmit ()  [private], [slot]
```

8.23.3 Member Data Documentation

8.23.3.1 m_authorEdit

```
QLineEdit* TestimonialDialog::m_authorEdit  [private]
```

8.23.3.2 m_contentEdit

```
QTextEdit* TestimonialDialog::m_contentEdit  [private]
```

8.23.3.3 m_doNotShowAgain

```
QCheckBox* TestimonialDialog::m_doNotShowAgain  [private]
```

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

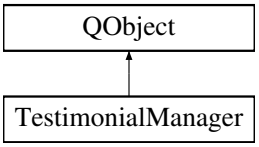
- src/frontend/TestimonialDialog.h
- src/frontend/TestimonialDialog.cpp

8.24 TestimonialManager Class Reference

Manages user testimonials, including storage, retrieval, and prompting logic.

`#include <TestimonialManager.h>`

Inheritance diagram for TestimonialManager:



Signals

User Interaction Signals

Signal emitted when conditions are met to prompt the user for a testimonial.

This signal is typically connected to a UI element that will display the prompt.

- void shouldPromptForTestimonial ()
  Emitted to prompt the user for a testimonial.

Public Member Functions

Core Testimonial Operations

Core testimonial management functions.

These methods allow adding new testimonials, retrieving satisfactory ones for display, and checking if a specific user has already submitted a testimonial.

- void addTestimonial (const Testimonial &testimonial)
  Adds a new testimonial to the manager and persists it.
- QVector< Testimonial > getSatisfactoryTestimonials () const
  Returns a vector of testimonials marked as satisfactory.
- bool hasUserGivenTestimonial (const QString &username) const
  Checks if a user with the given username has already submitted a testimonial.

User Prompt Preferences

User preference management for testimonial prompts.

These methods handle the "do not show again" preference for testimonial prompts on a per-user basis.

- void setDoNotShowAgain (const QString &username, bool value)
  Sets the "do not show again" preference for a user.
- bool getDoNotShowAgain (const QString &username) const
  Retrieves the "do not show again" preference for a user.

Activity Time Tracking

Time tracking control for testimonial prompts.

These methods start and stop the timer that tracks user activity, which is used to determine when to prompt for a testimonial.

- void startTrackingTime ()
  Starts the timer for tracking user activity time.
- void stopTrackingTime ()
  Stops the timer for tracking user activity time.

Static Public Member Functions

Singleton Access

Provides access to the singleton instance of TestimonialManager.

Ensures that only one instance of TestimonialManager exists throughout the application.

- static TestimonialManager & getInstance ()
  Returns a reference to the singleton TestimonialManager instance.

Private Slots

API Response Slots

API client response handlers.

These private slots are connected to signals from the ApiClient to process the results of network requests related to testimonials.

- void onGoodGetResponse (const QString &json)
  Slot called when a GET request for testimonials succeeds.
- void onBadGetResponse (const QString &error)
  Slot called when a GET request for testimonials fails.
- void onGoodPostResponse ()
  Slot called when a POST request to save testimonials succeeds.
- void onBadPostResponse (const QString &error)
  Slot called when a POST request to save testimonials fails.

Private Member Functions

Singleton Constructor/Destructor

Private constructor and destructor for the singleton pattern.

Ensures that TestimonialManager cannot be instantiated directly from outside the class.

- TestimonialManager ()
  Private constructor to enforce singleton pattern. Initializes API connections and timer.
- ~TestimonialManager ()
  Private destructor. Saves testimonials before destruction.

Internal Logic

Internal data persistence and prompt logic.

These methods handle the loading and saving of testimonial data (now via API) and the logic for checking if a testimonial prompt should be displayed.

- void loadTestimonials ()
  Loads testimonials (now primarily via API client in constructor).
- void saveTestimonials ()
  Saves all current testimonials (via API client).
- void checkTimeAndPrompt ()
  Checks accumulated user time and emits `shouldPromptForTestimonial` if criteria are met.

Member Data

Private member variables for TestimonialManager.

These members store testimonials, manage timing, user preferences, and utility objects.

- QVector< Testimonial > m_testimonials

  In-memory storage for all loaded testimonials.
- QTimer * m_trackingTimer

  Timer used to periodically check if a testimonial prompt is needed.
- QHash< QString, bool > m_doNotShowAgain

  Stores user preferences for not showing the testimonial prompt again. Key is username.
- QHash< QString, int > m_userTimeTracking

  Tracks accumulated active time for each user in minutes. Key is username.
- ApiClient client

  API client instance for network communication.
- Parser parse

  Parser instance for JSON serialization/deserialization.
- static const int INITIAL_PROMPT_TIME = 30 * 60

  Time in seconds before the first testimonial prompt (30 minutes).
- static const int REPEAT_PROMPT_TIME = 60 * 60

  Time in seconds for subsequent testimonial prompts if not dismissed (1 hour).

8.24.1 Detailed Description

Manages user testimonials, including storage, retrieval, and prompting logic.

This class is a singleton that handles all operations related to testimonials. It interacts with an ApiClient to load and save testimonials from/to a webservice, tracks user activity time to determine when to prompt for a testimonial, and manages user preferences regarding these prompts.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 TestimonialManager()

```
TestimonialManager::TestimonialManager ()  [private]
```

Private constructor to enforce singleton pattern. Initializes API connections and timer.

8.24.2.2 ~TestimonialManager()

```
TestimonialManager::~TestimonialManager ()  [private]
```

Private destructor. Saves testimonials before destruction.

8.24.3 Member Function Documentation

8.24.3.1 addTestimonial()

```
void TestimonialManager::addTestimonial (
            const Testimonial & testimonial)
```

Adds a new testimonial to the manager and persists it.

8.24.3.2 checkTimeAndPrompt()

```
void TestimonialManager::checkTimeAndPrompt ()  [private]
```

Checks accumulated user time and emits `shouldPromptForTestimonial` if criteria are met.

8.24.3.3 getDoNotShowAgain()

```
bool TestimonialManager::getDoNotShowAgain (
            const QString & username) const
```

Retrieves the "do not show again" preference for a user.

8.24.3.4 getInstance()

```
TestimonialManager & TestimonialManager::getInstance ()  [static]
```

Returns a reference to the singleton TestimonialManager instance.

8.24.3.5 getSatisfactoryTestimonials()

```
QVector< Testimonial > TestimonialManager::getSatisfactoryTestimonials () const
```

Returns a vector of testimonials marked as satisfactory.

8.24.3.6 hasUserGivenTestimonial()

```
bool TestimonialManager::hasUserGivenTestimonial (
            const QString & username) const
```

Checks if a user with the given username has already submitted a testimonial.

8.24.3.7 loadTestimonials()

```
void TestimonialManager::loadTestimonials ()  [private]
```

Loads testimonials (now primarily via API client in constructor).

8.24.3.8 onBadGetResponse

```
void TestimonialManager::onBadGetResponse (
            const QString & error)  [private], [slot]
```

Slot called when a GET request for testimonials fails.

8.24.3.9 onBadPostResponse

```
void TestimonialManager::onBadPostResponse (
            const QString & error)  [private], [slot]
```

Slot called when a POST request to save testimonials fails.

8.24.3.10 onGoodGetResponse

```
void TestimonialManager::onGoodGetResponse (
            const QString & json)  [private], [slot]
```

Slot called when a GET request for testimonials succeeds.

8.24.3.11 onGoodPostResponse

```
void TestimonialManager::onGoodPostResponse ()  [private], [slot]
```

Slot called when a POST request to save testimonials succeeds.

8.24.3.12 saveTestimonials()

```
void TestimonialManager::saveTestimonials ()  [private]
```

Saves all current testimonials (via API client).

8.24.3.13 setDoNotShowAgain()

```
void TestimonialManager::setDoNotShowAgain (
            const QString & username,
            bool value)
```

Sets the "do not show again" preference for a user.

8.24.3.14 shouldPromptForTestimonial

```
void TestimonialManager::shouldPromptForTestimonial ()  [signal]
```

Emitted to prompt the user for a testimonial.

8.24.3.15 startTrackingTime()

```
void TestimonialManager::startTrackingTime ()
```

Starts the timer for tracking user activity time.

8.24.3.16 stopTrackingTime()

```
void TestimonialManager::stopTrackingTime ()
```

Stops the timer for tracking user activity time.

8.24.4 Member Data Documentation

8.24.4.1 client

```
ApiClient TestimonialManager::client  [private]
```

API client instance for network communication.

8.24.4.2 INITIAL_PROMPT_TIME

```
const int TestimonialManager::INITIAL_PROMPT_TIME = 30 * 60  [static], [private]
```

Time in seconds before the first testimonial prompt (30 minutes).

8.24.4.3 m_doNotShowAgain

```
QHash<QString, bool> TestimonialManager::m_doNotShowAgain  [private]
```

Stores user preferences for not showing the testimonial prompt again. Key is username.

8.24.4.4 m_testimonials

```
QVector<Testimonial> TestimonialManager::m_testimonials  [private]
```

In-memory storage for all loaded testimonials.

8.24.4.5 m_trackingTimer

```
QTimer* TestimonialManager::m_trackingTimer  [private]
```

Timer used to periodically check if a testimonial prompt is needed.

8.24.4.6 m_userTimeTracking

```
QHash<QString, int> TestimonialManager::m_userTimeTracking  [private]
```

Tracks accumulated active time for each user in minutes. Key is username.

8.24.4.7 parse

```
Parser TestimonialManager::parse  [private]
```

Parser instance for JSON serialization/deserialization.

8.24.4.8 REPEAT_PROMPT_TIME

```
const int TestimonialManager::REPEAT_PROMPT_TIME = 60 * 60  [static], [private]
```

Time in seconds for subsequent testimonial prompts if not dismissed (1 hour).

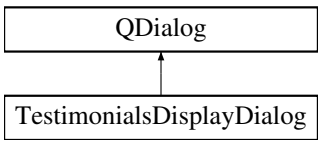
The documentation for this class was generated from the following files:

- src/backend/TestimonialManager.h
- src/backend/TestimonialManager.cpp

8.25 TestimonialsDisplayDialog Class Reference

```
#include <TestimonialsDisplayDialog.h>
```

Inheritance diagram for TestimonialsDisplayDialog:



Public Member Functions

- TestimonialsDisplayDialog (QWidget *parent=nullptr)

Private Member Functions

- void refreshTestimonials ()

Private Attributes

- QVBoxLayout * m_testimonialsLayout

8.25.1 Constructor & Destructor Documentation

8.25.1.1 TestimonialsDisplayDialog()

```
TestimonialsDisplayDialog::TestimonialsDisplayDialog (
            QWidget * parent = nullptr)  [explicit]
```

8.25.2 Member Function Documentation

8.25.2.1 refreshTestimonials()

```
void TestimonialsDisplayDialog::refreshTestimonials ()  [private]
```

8.25.3 Member Data Documentation

8.25.3.1 m_testimonialsLayout

```
QVBoxLayout* TestimonialsDisplayDialog::m_testimonialsLayout  [private]
```

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

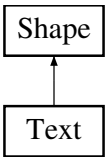
- src/frontend/TestimonialsDisplayDialog.h
- src/frontend/TestimonialsDisplayDialog.cpp

8.26 Text Class Reference

The Text class.

```
#include "objects/text.h"
```

Inheritance diagram for Text:



Public Member Functions

- Text (string shapeType, QPoint coords, QString textString, GlobalColor textColor, AlignmentFlag textAlignment, QFont font, int length, int width)

  Text.
- void Draw (QWidget *renderArea) override

  Draw - Draws the text to the assigned coords.
- double Perimeter () const override

  Perimeter - Returns the perimater of the text box.
- double Area () const override

  Area - Returns the area of the text box.
- bool isPointInside (const QPoint &point) const override

  isPointInside - Returns True if point is inside the text box
- int getLength () const

  Accessor Functions.
- int getWidth () const
- QString getTextString () const
- GlobalColor getTextColor () const
- QFont getFont () const
- AlignmentFlag getTextAlignment () const
- int getFontStyle () const
- QFont::Weight getFontWeight () const
- void setText (QString text)

  Mutator Functions.
- void setLength (int newLength)
- void setWidth (int newWidth)
- void setX (int newX)
- void setY (int newY)
- void setAlignment (Qt::AlignmentFlag alignment)
- QFont & setInternalFont ()

Public Member Functions inherited from Shape

- Shape (int shapeId, string shapeType, QPoint coords, QPen pen, QBrush brush)

  Shape Constructor.
- virtual ~Shape ()

  ~Shape Destructor
- virtual void Move (int x, int y)

  Move - Moves the shape to the x and y coords.
- void CreateParentItem ()

  CreateParentItem - Adds data cooresponding to all shapes to parentItem for a QTreeWidget.
- void CreatePenChild ()

  CreatePenChild - Adds pen data to penItems vector for a QTreeWidget.
- void CreateBrushChild ()

  CreateBrushChild - Adds brush data to brushItems vector for a QTreeWidget.
- void CreatePointsChild (const int POINTS_NUM)

  CreatePointsChild - Adds points data to pointsItems vector for a QTreeWidget.
- int getShapeId () const

  Accessor Functions - Returns the data named after them.
- int getTrackerId () const
- string getShapeType () const

- bool getSelected () const
- int getX () const
- int getY () const
- QPainter & getPainter ()
- QTreeWidgetItem * getParentItem ()
- alpha::vector< QTreeWidgetItem * > & getChildItems ()
- alpha::vector< QTreeWidgetItem * > & getPointsItems ()
- alpha::vector< QTreeWidgetItem * > & getPenItems ()
- alpha::vector< QTreeWidgetItem * > & getBrushItems ()
- int getPenWidth () const
- PenStyle getPenStyle () const
- PenCapStyle getPenCapStyle () const
- PenJoinStyle getPenJoinStyle () const
- QColor getPenColor () const
- QColor getBrushColor () const
- BrushStyle getBrushStyle () const
- QPen getPen () const
- QBrush getBrush () const
- QPoint getPoints () const
- int getChildEnd () const
- int getPenItemsEnd () const
- int getBrushItemsEnd () const
- void setShapeType (string shapeType)

  Accessor Functions.
- void setSelected (bool selected)
- void setTrackerId (int trackerId)
- void allocateTrackerId (int shapeId)
- void setPen (GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle, PenJoin↩ Style penJoinStyle)
- void setBrush (GlobalColor brushColor, BrushStyle brushStyle)
- QPen & setInternalPen ()
- QBrush & setInternalBrush ()

Private Attributes

- int length

  length of text box
- int width

  width of text box
- QString textString

  string of text being displayed
- GlobalColor textColor

  text color
- QFont font

  text font
- AlignmentFlag textAlignment

  text alignment

Additional Inherited Members

Static Public Attributes inherited from Shape

- static int nextTracker [9] = {}
- static bool trackersInUse [9000] = {}

Protected Attributes inherited from Shape

- QTreeWidgetItem * parentItem

  Mutator Functions.
- alpha::vector< QTreeWidgetItem * > childItems

  vector of QTreeWidgetItem holding data of all child items in parentItem*
- alpha::vector< QTreeWidgetItem * > pointsItems

  vector of QTreeWidgetItem holding data of all points (besides coords) in parentItem*
- alpha::vector< QTreeWidgetItem * > penItems

  vector of QTreeWidgetItem holding data of all pen items*
- alpha::vector< QTreeWidgetItem * > brushItems

  vector of QTreeWidgetItem holding data of all brush items*

8.26.1 Detailed Description

The Text class.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 Text()

```
Text::Text (
            string shapeType,
            QPoint coords,
            QString textString,
            GlobalColor textColor,
            AlignmentFlag textAlignment,
            QFont font,
            int length,
            int width)
```

Text.

Parameters

shapeType	- Used in Shape constructor MIL
coords	- Used in Shape constructor MIL
textString	- String of text being displayed
textColor	- Color of text
textAlignment	- Alignment of text
font	- Text font
length	- Length of text box
width	- Width of text box

8.26.3 Member Function Documentation

8.26.3.1 Area()

```
double Text::Area () const  [override], [virtual]
```

Area - Returns the area of the text box.

Returns

Implements Shape.

8.26.3.2 Draw()

```
void Text::Draw (
            QWidget * renderArea)  [override], [virtual]
```

Draw - Draws the text to the assigned coords.

Parameters

renderArea	- The renderArea being drawn on

Implements Shape.

8.26.3.3 getFont()

```
QFont Text::getFont () const
```

8.26.3.4 getFontStyle()

```
int Text::getFontStyle () const
```

8.26.3.5 getFontWeight()

```
QFont::Weight Text::getFontWeight () const
```

8.26.3.6 getLength()

```
int Text::getLength () const
```

Accessor Functions.

8.26.3.7 getTextAlignment()

```
AlignmentFlag Text::getTextAlignment () const
```

8.26.3.8 getTextColor()

```
GlobalColor Text::getTextColor () const
```

8.26.3.9 getTextString()

```
QString Text::getTextString () const
```

8.26.3.10 getWidth()

```
int Text::getWidth () const
```

8.26.3.11 isPointInside()

```
bool Text::isPointInside (
            const QPoint & point) const  [override], [virtual]
```

isPointInside - Returns True if point is inside the text box

Parameters

point	- point being read

Returns

Implements Shape.

8.26.3.12 Perimeter()

```
double Text::Perimeter () const  [override], [virtual]
```

Perimeter - Returns the perimater of the text box.

Returns

Implements Shape.

8.26.3.13 setAlignment()

```
void Text::setAlignment (
            Qt::AlignmentFlag alignment)
```

8.26.3.14 setInternalFont()

```
QFont & Text::setInternalFont ()
```

8.26.3.15 setLength()

```
void Text::setLength (
            int newLength)
```

8.26.3.16 setText()

```
void Text::setText (
            QString text)
```

Mutator Functions.

8.26.3.17 setWidth()

```
void Text::setWidth (
            int newWidth)
```

8.26.3.18 setX()

```
void Text::setX (
            int newX)
```

Implements Shape.

8.26.3.19 setY()

```
void Text::setY (
            int newY)
```

Implements Shape.

8.26.4 Member Data Documentation

8.26.4.1 font

```
QFont Text::font  [private]
```

text font

8.26.4.2 length

```
int Text::length  [private]
```

length of text box

8.26.4.3 textAlignment

```
AlignmentFlag Text::textAlignment  [private]
```

text alignment

8.26.4.4 textColor

```
GlobalColor Text::textColor  [private]
```

text color

8.26.4.5 textString

```
QString Text::textString  [private]
```

string of text being displayed

8.26.4.6 width

```
int Text::width  [private]
```

width of text box

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

- src/objects/text.h
- src/objects/text.cpp

8.27 UserAccount Class Reference

Represents a user account within the application.

```
#include <UserAccount.h>
```

Public Member Functions

Constructors and Destructor

Manages the lifecycle of UserAccount objects.

These methods handle the creation, initialization, and destruction of UserAccount objects.

- UserAccount ()
  Default constructor. Initializes as a guest user.
- UserAccount (QString username, QString password, bool admin)
  Parameterized constructor. Initializes with specified credentials and admin status.
- ~UserAccount ()
  Destructor.

Copy and Move Semantics

Defines how UserAccount objects are copied and moved.

These methods define how UserAccount objects are copied and moved, ensuring proper resource management.

- UserAccount (const UserAccount &other)
  Copy constructor.
- UserAccount & operator= (const UserAccount &other)
  Copy assignment operator.
- UserAccount (UserAccount &&other) noexcept
  Move constructor.
- UserAccount & operator= (UserAccount &&other) noexcept
  Move assignment operator.

Getters

Public getters for accessing user account data.

These methods provide read-only access to the private member variables of the UserAccount object.

- QString getUsername () const
  Returns the username of the account.
- QString getPassword () const
  Returns the password of the account.
- bool isAdmin () const
  Returns true if the user has administrative privileges, false otherwise.

Setters

Public setters for modifying user account data.

These methods allow modification of the username, password, and administrative status of the UserAccount object.

- void setUsername (const QString &username)
  Sets the username for the account.
- void setPassword (const QString &password)
  Sets the password for the account.
- void setAdmin (bool admin)
  Sets the administrative status for the account.
- void setUserAccount (const QString &username, const QString &password, bool admin)
  Sets all properties of the user account at once.

Private Attributes

Member Data

Private member variables storing user account attributes.

These variables store the core information for each user account.

- QString username
  The username associated with the account.
- QString password
  The password for the account. For guest users, this might be NULL or empty.
- bool admin
  Flag indicating whether the user has administrative privileges.

8.27.1 Detailed Description

Represents a user account within the application.

This class encapsulates user credentials (username and password) and administrative status. It provides constructors, destructors, copy/move semantics, and getters/setters for its properties.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 UserAccount() [1/4]

```
UserAccount::UserAccount ()
```

Default constructor. Initializes as a guest user.

8.27.2.2 UserAccount() [2/4]

```
UserAccount::UserAccount (
            QString username,
            QString password,
            bool admin)
```

Parameterized constructor. Initializes with specified credentials and admin status.

8.27.2.3 ~UserAccount()

```
UserAccount::~UserAccount ()
```

Destructor.

8.27.2.4 UserAccount() [3/4]

```
UserAccount::UserAccount (
            const UserAccount & other)
```

Copy constructor.

8.27.2.5 UserAccount() [4/4]

```
UserAccount::UserAccount (
            UserAccount && other)  [noexcept]
```

Move constructor.

8.27.3 Member Function Documentation

8.27.3.1 getPassword()

```
QString UserAccount::getPassword () const
```

Returns the password of the account.

8.27.3.2 getUsername()

```
QString UserAccount::getUsername () const
```

Returns the username of the account.

8.27.3.3 isAdmin()

```
bool UserAccount::isAdmin () const
```

Returns true if the user has administrative privileges, false otherwise.

8.27.3.4 operator=() [1/2]

```
UserAccount & UserAccount::operator= (
            const UserAccount & other)
```

Copy assignment operator.

8.27.3.5 operator=() [2/2]

```
UserAccount & UserAccount::operator= (
            UserAccount && other)  [noexcept]
```

Move assignment operator.

8.27.3.6 setAdmin()

```
void UserAccount::setAdmin (
            bool admin)
```

Sets the administrative status for the account.

8.27.3.7 setPassword()

```
void UserAccount::setPassword (
            const QString & password)
```

Sets the password for the account.

8.27.3.8 setUserAccount()

```
void UserAccount::setUserAccount (
            const QString & username,
            const QString & password,
            bool admin)
```

Sets all properties of the user account at once.

8.27.3.9 setUsername()

```
void UserAccount::setUsername (
            const QString & username)
```

Sets the username for the account.

8.27.4 Member Data Documentation

8.27.4.1 admin

```
bool UserAccount::admin  [private]
```

Flag indicating whether the user has administrative privileges.

8.27.4.2 password

```
QString UserAccount::password  [private]
```

The password for the account. For guest users, this might be NULL or empty.

8.27.4.3 username

```
QString UserAccount::username  [private]
```

The username associated with the account.

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

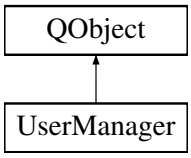
- src/backend/UserAccount.h
- src/backend/UserAccount.cpp

8.28 UserManager Class Reference

Manages user accounts, including creation, authentication, and persistence.

`#include <UserManager.h>`

Inheritance diagram for UserManager:



Signals

User Status Signals

Signals for UI interaction and status updates.

These signals are emitted to notify the frontend about changes in user data, authentication status, or general operational messages.

- void userChanged ()
  Emitted when the list of users changes (add, modify, delete).
- void userNotChanged (const QString &message)
  Emitted when an operation that would change users fails (e.g., user not found, username taken).
- void statusMessage (const QString &message)
  Emitted to provide general status updates or error messages.
- void userAuthenticated (const UserAccount *currUser)
  Emitted upon successful user authentication, providing the authenticated user.
- void authenticationFailed (const QString &message)
  Emitted when user authentication fails.

Public Member Functions

Constructor and Destructor

Manages the lifecycle of UserManager objects.

Handles the initialization and cleanup of the UserManager. The constructor sets up connections to the ApiClient for handling responses to user data requests and loads initial user data.

- UserManager (QObject *parent=nullptr)
  Constructor. Initializes ApiClient connections and loads users.
- ~UserManager ()
  Destructor. Cleans up dynamically allocated UserAccount objects.

Current User Access

Accessor for the current user.

Provides a reference to the currently authenticated user account. If no user is authenticated, this typically points to a guest account.

- UserAccount * getCurrUserRef ()
  Returns a pointer to the currently authenticated UserAccount object.

User Account Management

Core functions for managing user accounts.

These methods provide functionalities to add, modify, delete, load, save, and authenticate user accounts. Most operations will trigger signals to update the UI or indicate status.

- void addUser (const QString username, const QString password, const bool admin)
- void modifyUser (const QString username, const QString password, const bool admin)
- void deleteUser (QString username)

  Deletes a specific user account by username.
- void deleteAllUsers ()

  Deletes all user accounts from the backend.
- void loadUsers ()

  Initiates loading of all user accounts from the webservice.
- void saveUsers ()

  Saves all current user accounts to the webservice.
- void authenticate (const QString username, const QString password)

Private Slots

API Response Slots

API client response handlers.

These private slots are connected to signals from the ApiClient to process the results of network requests related to user data.

- void onGoodGetResponse (const QString &json)

  Slot called when a GET request for user data succeeds.
- void onBadGetResponse (const QString &errorMsg)

  Slot called when a GET request for user data fails.
- void onGoodPostResponse ()

  Slot called when a POST request to save user data succeeds.
- void onBadPostResponse (const QString &errorMsg)

  Slot called when a POST request to save user data fails.
- void onGoodDeleteResponse ()

  Slot called when a DELETE request for user data succeeds.
- void onBadDeleteResponse (const QString &errorMsg)

  Slot called when a DELETE request for user data fails.

Private Attributes

Member Data

Private member variables for UserManager.

These members store the current user, all known users, and utility objects.

- UserAccount * currUser

  Pointer to the currently authenticated user. Defaults to a guest user.
- alpha::vector< UserAccount * > users

  Vector storing pointers to all loaded UserAccount objects.
- ApiClient client

  API client instance for network communication with the user data backend.
- Parser parse

  Parser instance for JSON serialization/deserialization of user data.

8.28.1 Detailed Description

Manages user accounts, including creation, authentication, and persistence.

This class handles all operations related to user accounts, such as adding, modifying, deleting, and authenticating users. It interacts with an ApiClient to load and save user data from/to a webservice and uses a Parser for data serialization/deserialization.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 UserManager()

```
UserManager::UserManager (
            QObject * parent = nullptr)  [explicit]
```

Constructor. Initializes ApiClient connections and loads users.

8.28.2.2 ~UserManager()

```
UserManager::~UserManager ()
```

Destructor. Cleans up dynamically allocated UserAccount objects.

8.28.3 Member Function Documentation

8.28.3.1 addUser()

```
void UserManager::addUser (
            const QString username,
            const QString password,
            const bool admin)
```

Parameters

username	Adds a new user account and attempts to authenticate with the new credentials.

8.28.3.2 authenticate()

```
void UserManager::authenticate (
            const QString username,
            const QString password)
```

Parameters

username	Authenticates a user against the loaded user accounts.

8.28.3.3 authenticationFailed

```
void UserManager::authenticationFailed (
            const QString & message)  [signal]
```

Emitted when user authentication fails.

8.28.3.4 deleteAllUsers()

```
void UserManager::deleteAllUsers ()
```

Deletes all user accounts from the backend.

8.28.3.5 deleteUser()

```
void UserManager::deleteUser (
            QString username)
```

Deletes a specific user account by username.

8.28.3.6 getCurrUserRef()

```
UserAccount * UserManager::getCurrUserRef ()
```

Returns a pointer to the currently authenticated UserAccount object.

8.28.3.7 loadUsers()

```
void UserManager::loadUsers ()
```

Initiates loading of all user accounts from the webservice.

8.28.3.8 modifyUser()

```
void UserManager::modifyUser (
            const QString username,
            const QString password,
            const bool admin)
```

Parameters

username	Modifies an existing user account's details (password, admin status).

8.28.3.9 onBadDeleteResponse

```
void UserManager::onBadDeleteResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot called when a DELETE request for user data fails.

8.28.3.10 onBadGetResponse

```
void UserManager::onBadGetResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot called when a GET request for user data fails.

8.28.3.11 onBadPostResponse

```
void UserManager::onBadPostResponse (
            const QString & errorMsg)  [private], [slot]
```

Slot called when a POST request to save user data fails.

8.28.3.12 onGoodDeleteResponse

```
void UserManager::onGoodDeleteResponse ()  [private], [slot]
```

Slot called when a DELETE request for user data succeeds.

8.28.3.13 onGoodGetResponse

```
void UserManager::onGoodGetResponse (
            const QString & json)  [private], [slot]
```

Slot called when a GET request for user data succeeds.

8.28.3.14 onGoodPostResponse

```
void UserManager::onGoodPostResponse ()  [private], [slot]
```

Slot called when a POST request to save user data succeeds.

8.28.3.15 saveUsers()

```
void UserManager::saveUsers ()
```

Saves all current user accounts to the webservice.

8.28.3.16 statusMessage

```
void UserManager::statusMessage (
            const QString & message)  [signal]
```

Emitted to provide general status updates or error messages.

8.28.3.17 userAuthenticated

```
void UserManager::userAuthenticated (
            const UserAccount * currUser)  [signal]
```

Emitted upon successful user authentication, providing the authenticated user.

8.28.3.18 userChanged

```
void UserManager::userChanged ()  [signal]
```

Emitted when the list of users changes (add, modify, delete).

8.28.3.19 userNotChanged

```
void UserManager::userNotChanged (
            const QString & message)  [signal]
```

Emitted when an operation that would change users fails (e.g., user not found, username taken).

8.28.4 Member Data Documentation

8.28.4.1 client

```
ApiClient UserManager::client  [private]
```

API client instance for network communication with the user data backend.

8.28.4.2 currUser

```
UserAccount* UserManager::currUser  [private]
```

Pointer to the currently authenticated user. Defaults to a guest user.

8.28.4.3 parse

```
Parser UserManager::parse  [private]
```

Parser instance for JSON serialization/deserialization of user data.

8.28.4.4 users

```
alpha::vector<UserAccount*> UserManager::users  [private]
```

Vector storing pointers to all loaded UserAccount objects.

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

- src/backend/UserManager.h
- src/backend/moc_UserManager.cpp
- src/backend/UserManager.cpp

8.29 alpha::vector< T > Class Template Reference

`#include <vector.h>`

Public Types

- using iterator = T*
- using const_iterator = const T*

Public Member Functions

- vector ()
- vector (int s)
- vector (const vector &other)
- vector & operator= (const vector &other)
- vector (vector &&other) noexcept
- vector & operator= (vector &&other) noexcept
- ~vector ()
- T & operator[ ] (int n)
- const T & operator[ ] (int n) const
- int size () const
- int capacity () const
- void resize (int newsize)
- void push_back (const T val)
- void reserve (int newalloc)
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- iterator insert (iterator p, const T &v)
- iterator erase (iterator p)

Private Attributes

- int size_v
- T * elem
- int space

8.29.1 Member Typedef Documentation

8.29.1.1 const_iterator

```
template<class T>
using alpha::vector< T >::const_iterator = const T*
```

8.29.1.2 iterator

```
template<class T>
using alpha::vector< T >::iterator = T*
```

8.29.2 Constructor & Destructor Documentation

8.29.2.1 vector() [1/4]

```
template<class T>
alpha::vector< T >::vector ()  [inline]
```

8.29.2.2 vector() [2/4]

```
template<class T>
alpha::vector< T >::vector (
            int s)  [inline], [explicit]
```

8.29.2.3 vector() [3/4]

```
template<class T>
alpha::vector< T >::vector (
            const vector< T > & other)  [inline]
```

8.29.2.4 vector() [4/4]

```
template<class T>
alpha::vector< T >::vector (
            vector< T > && other)  [inline], [noexcept]
```

8.29.2.5 ~vector()

```
template<class T>
alpha::vector< T >::~vector ()  [inline]
```

8.29.3 Member Function Documentation

8.29.3.1 begin() [1/2]

```
template<class T>
iterator alpha::vector< T >::begin ()  [inline]
```

8.29.3.2 begin() [2/2]

```
template<class T>
const_iterator alpha::vector< T >::begin () const  [inline]
```

8.29.3.3 capacity()

```
template<class T>
int alpha::vector< T >::capacity () const  [inline]
```

8.29.3.4 end() [1/2]

```
template<class T>
iterator alpha::vector< T >::end ()  [inline]
```

8.29.3.5 end() [2/2]

```
template<class T>
const_iterator alpha::vector< T >::end () const  [inline]
```

8.29.3.6 erase()

```
template<class T>
iterator alpha::vector< T >::erase (
            iterator p)  [inline]
```

8.29.3.7 insert()

```
template<class T>
iterator alpha::vector< T >::insert (
            iterator p,
            const T & v)  [inline]
```

8.29.3.8 operator=() [1/2]

```
template<class T>
vector & alpha::vector< T >::operator= (
            const vector< T > & other)  [inline]
```

8.29.3.9 operator=() [2/2]

```
template<class T>
vector & alpha::vector< T >::operator= (
            vector< T > && other)  [inline], [noexcept]
```

8.29.3.10 operator[]() [1/2]

```
template<class T>
T & alpha::vector< T >::operator[] (
            int n)  [inline]
```

8.29.3.11 operator[]() [2/2]

```
template<class T>
const T & alpha::vector< T >::operator[] (
            int n) const  [inline]
```

8.29.3.12 push_back()

```
template<class T>
void alpha::vector< T >::push_back (
            const T val)  [inline]
```

8.29.3.13 reserve()

```
template<class T>
void alpha::vector< T >::reserve (
            int newalloc)  [inline]
```

8.29.3.14 resize()

```
template<class T>
void alpha::vector< T >::resize (
            int newsize)  [inline]
```

8.29.3.15 size()

```
template<class T>
int alpha::vector< T >::size () const  [inline]
```

8.29.4 Member Data Documentation

8.29.4.1 elem

```
template<class T>
T* alpha::vector< T >::elem  [private]
```

8.29.4.2 size_v

```
template<class T>
int alpha::vector< T >::size_v  [private]
```

8.29.4.3 space

```
template<class T>
int alpha::vector< T >::space  [private]
```

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

- src/objects/vector.h

Chapter 9

File Documentation

9.1 src/backend/ApiClient.cpp File Reference

```
#include "ApiClient.h"
```

9.2 src/backend/ApiClient.h File Reference

Implements the ApiClient class that makes API requests to the Crow Webservice.

```
#include <QObject>
#include <QNetworkAccessManager>
#include <QNetworkReply>
#include <QNetworkRequest>
#include <QUrl>
#include <QByteArray>
#include <QString>
```

Classes

- class ApiClient

Documentation Qt Version 6.9.0.

9.2.1 Detailed Description

Implements the ApiClient class that makes API requests to the Crow Webservice.

This class interacts with the following endpoints through the corresponding member functions Get /shapes : Get↩ Shapes() Post /shapes : PostShapes() Get /render_area : GetRenderArea() Post /render_area : PostRenderArea()

9.3 ApiClient.h

Go to the documentation of this file.

```
00001
00011 #ifndef API_CLIENT_H
00012 #define API_CLIENT_H
00014 #include <QObject>                  // https://doc.qt.io/qt-6/qobject.html
00015 #include <QNetworkAccessManager>    // https://doc.qt.io/qt-6/qnetworkaccessmanager.html
00016 #include <QNetworkReply>            // https://doc.qt.io/qt-6/qnetworkreply.html
00017 #include <QNetworkRequest>          // https://doc.qt.io/qt-6/qnetworkrequest.html
00018 #include <QUrl>                     // https://doc.qt.io/qt-6/qurl.html
00019 #include <QByteArray>               // https://doc.qt.io/qt-6/qbytearray.html
00020 #include <QString>                  // https://doc.qt.io/qt-6/qstring.html
00021
00030 class ApiClient : public QObject {
00031     Q_OBJECT    //necessary macro for qt's compiler
00032
00033 public:
00040     explicit ApiClient(QObject* parent = nullptr);
00042
00043
00053     void GetShapes();
00054     void PostShapes(std::string json);
00055     void DeleteShapesAll();
00057
00058
00066     void GetRenderArea();
00067     void PostRenderArea(std::string json);
00068     void DeleteRenderAreaAll();
00070
00071
00079     void GetUsers();
00080     void PostUsers(std::string json);
00081     void DeleteUsersAll();
00083
00084
00092     void GetTestimonials();
00093     void PostTestimonials(std::string json);
00094     void DeleteTestimonialsAll();
00096
00097
00098 signals:
00106     void GoodGetReply(const QString& json);
00107     void BadGetReply(const QString& error);
00108     void GoodPostReply();
00109     void BadPostReply(const QString& error);
00110     void GoodDeleteReply();
00111     void BadDeleteReply(const QString& error);
00113
00114
00115 private slots:
00123     void AnalyzeGetReply();
00124     void AnalyzePostReply();
00125     void AnalyzeDeleteReply();
00127
00128
00129 private:
00130     QNetworkAccessManager* manager;
00131 };
00132
00133 #endif //API_CLIENT_H
```

9.4 src/backend/AppDriver.cpp File Reference

```
#include "AppDriver.h"
```

9.5 src/backend/AppDriver.h File Reference

```
#include <QCoreApplication>
#include <QApplication>
```

```
#include <QObject>
#include <QTimer>
#include <QDebug>
#include "RenderAreaManager.h"
#include "UserManager.h"
#include "../frontend/mainwindow.h"
#include "../frontend/renderarea.h"
```

Classes

- class AppDriver

  Orchestrates the main application logic and connections.

9.6 AppDriver.h

Go to the documentation of this file.

```
00001 #ifndef APP_DRIVER_H
00002 #define APP_DRIVER_H
00003 #include <QCoreApplication>
00004 #include <QApplication>
00005 #include <QObject>
00006 #include <QTimer>
00007 #include <QDebug>
00008 #include "RenderAreaManager.h"
00009 #include "UserManager.h"
00010 #include "../frontend/mainwindow.h"
00011 #include "../frontend/renderarea.h"
00012
00022 class AppDriver : public QObject {
00023     Q_OBJECT
00024 public:
00032     AppDriver(QObject* parent = nullptr);
00033     ~AppDriver();
00034     void run();
00035     void shutdown();
00036     void loadAllData();
00038
00039
00040 private slots:
00049     void onRenderShapeAdded(Shape* shape);
00050     void onRenderShapeChanged(Shape* shape,
00051                               QString key,
00052                               int value);
00053     void onRenderShapeDeleted(const int trackerId);
00054     void onRenderDeleteAllShapes();
00056
00066     void onNewUser(const QString username,
00067                    const QString password,
00068                    const bool admin);
00069     void onUserModified(const QString username,
00070                         const QString password,
00071                         const bool admin);
00072     void onUserDeleted(const QString username);
00073     void onDeleteAllUsers();
00074     void onLoginAttempt(const QString username,
00075                         const QString password);
00077
00078
00079 private:
00086     MainWindow* mainWindow;
00087     RenderAreaManager* renderedShapes;
00088     UserManager* user;
00090
00091
00099     void connectFrontendToDriver();
00100     void connectManagersToFrontend();
00102 };
00103
00104 #endif // APP_DRIVER_H
```

9.7 src/backend/moc_ApiClient.cpp File Reference

```
#include "ApiClient.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
```

Classes

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN9ApiClientE_t

Namespaces

- namespace QT_WARNING_DISABLE_DEPRECATED

Macros

- #define Q_CONSTINIT

Variables

- static Q_CONSTINIT const uint qt_meta_data_ZN9ApiClientE [ ]

9.7.1 Macro Definition Documentation

9.7.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

9.7.2 Variable Documentation

9.7.2.1 qt_meta_data_ZN9ApiClientE

```
Q_CONSTINIT const uint qt_meta_data_ZN9ApiClientE[]  [static]
```

9.8 src/backend/moc_UserManager.cpp File Reference

```
#include "UserManager.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
```

Classes

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN11UserManagerE_t

Namespaces

- namespace QT_WARNING_DISABLE_DEPRECATED

Macros

- #define Q_CONSTINIT

Variables

- static Q_CONSTINIT const uint qt_meta_data_ZN11UserManagerE [ ]

9.8.1 Macro Definition Documentation

9.8.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

9.8.2 Variable Documentation

9.8.2.1 qt_meta_data_ZN11UserManagerE

```
Q_CONSTINIT const uint qt_meta_data_ZN11UserManagerE[]  [static]
```

9.9 src/backend/Parser.cpp File Reference

```
#include "Parser.h"
```

9.10 src/backend/Parser.h File Reference

```
#include <iostream>
#include <cctype>
#include <QVector>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
#include <string>
#include "../objects/vector.h"
#include "../objects/all_shapes.h"
#include "UserAccount.h"
#include "Testimonial.h"
```

Classes

- class Parser

  Provides functionality for parsing JSON data to C++ objects and vice-versa.
- struct Parser::MorphicShape

  Internal accumulator structure for parsing shape data from JSON.
- struct Parser::RawUser

  Internal accumulator structure for parsing user account data from JSON.

9.11 Parser.h

Go to the documentation of this file.

```
00001 #ifndef PARSER_H
00002 #define PARSER_H
00003
00004 #include <iostream>
00005 #include <cctype>
00006 #include <QVector>
00007 #include <QJsonDocument>
00008 #include <QJsonArray>
00009 #include <QJsonObject>
00010 #include <string>
00011 #include "../objects/vector.h"
00012 #include "../objects/all_shapes.h"
00013 #include "UserAccount.h"
00014 #include "Testimonial.h"
00015
00026 class Parser {
00027 public:
00029     Parser() = default;
00031     ~Parser() = default;
00032
00033     //Disable copying and moving
00035     Parser(const Parser&) = delete;
00037     Parser& operator=(const Parser&) = delete;
00039     Parser(Parser&&) = delete;
00041     Parser& operator=(Parser&&) = delete;
00042
00049     void PrintShapeVector(const alpha::vector<Shape*> &shapes);
00050
00059     alpha::vector<Shape*> JsonToShapes(const std::string& json);
00060
00068     std::string ShapesToJson(const alpha::vector<Shape*>& shapes);
00069
00077     alpha::vector<UserAccount*> JsonToUsers(const std::string& json);
00078
00085     std::string UsersToJson(const alpha::vector<UserAccount*>& users);
00086
00094     static QVector<Testimonial> JsonToTestimonials(const std::string& json);
00095
00102     static std::string TestimonialsToJson(const QVector<Testimonial>& testimonials);
00103
00104 private:
00105     /*==================================== Forward Parser Subroutines
    ==========================================*/
00113     struct MorphicShape {
00114         std::string shapeType = "";
00115         int shapeId = 0;
00116         int trackerId = 0;
00117         alpha::vector<int> shapeDimensions;
00118         QPen pen = QPen();
00119         QBrush brush = QBrush();
00120         QPoint coords = QPoint();
00121
00122         QString textString;
00123         GlobalColor textColor;
00124         QFont font;
00125         AlignmentFlag textAlignment;
00126     };
00127
00133     struct RawUser {
00134             QString username;
00135             QString password;
00136             bool    admin       = false;
00137             bool    hasUsername = false;
00138             bool    hasPassword = false;
```

```
00139             bool    hasAdmin    = false;
00140         };
00141
00151     MorphicShape ParseJsonObject(const std::string json, size_t &index);
00152
00162     void UpdateAccumulator(const std::string &key, const std::string &value, MorphicShape &tempShape);
00163
00171     Shape* BuildShape(MorphicShape tempShape);
00172
00179     void SkipWhitespace(const std::string& json, size_t& index);
00180
00189     std::string ExtractKey(const std::string& json, size_t &index);
00190
00200     std::string ExtractValue(const std::string& json, size_t &index);
00201
00210     std::string ExtractInteger(const std::string& json, size_t &index);
00211
00221     std::string ExtractArray(const std::string& json, size_t &index);
00222
00231     std::string ExtractLiteral(const std::string& json, size_t& index);
00232
00240     alpha::vector<int> StringToVector(const std::string &value);
00241
00242     /*================================== Reverse Parser Subroutines
      ==========================================*/
00243
00251     std::string AppendCommonShapeData(const Shape* shape);
00252
00259     std::string AppendBrushData(const Shape* shape);
00260
00268     std::string AppendTextData(const Shape* shape);
00269
00278     std::string GetShapeDimensions(const Shape* shape);
00279
00286     std::string GetColor(const QColor &objectColor);
00287
00294     std::string GetPenStyle(const Shape* shape);
00295
00302     std::string GetPenCapStyle(const Shape* shape);
00303
00310     std::string GetPenJoinStyle(const Shape* shape);
00311
00318     std::string GetBrushStyle(const Shape* shape);
00319
00326     std::string GetAlignmentFlag(const Text* text);
00327
00334     std::string GetFontStyle(const Text* text);
00335
00342     std::string GetFontWeight(const Text* text);
00343
00354     static void   UpdateUserAccumulator(const std::string& key, const std::string& value, RawUser&
      acc);
00355 };
00356
00357 #endif // PARSER_H
```

9.12 src/backend/RenderAreaManager.cpp File Reference

```
#include "RenderAreaManager.h"
```

9.13 src/backend/RenderAreaManager.h File Reference

```
#include <QObject>
#include <QString>
#include <QDebug>
#include <QBrush>
#include <QPen>
#include "ApiClient.h"
#include "Parser.h"
#include "../objects/all_shapes.h"
#include "../objects/vector.h"
```

Classes

- class RenderAreaManager

  Manages the shapes to be rendered and interacts with the backend API.

9.14 RenderAreaManager.h

Go to the documentation of this file.

```
00001 #ifndef RENDER_AREA_MANAGER
00002 #define RENDER_AREA_MANAGER
00003
00004 #include <QObject>
00005 #include <QString>
00006 #include <QDebug>
00007 #include <QBrush>
00008 #include <QPen>
00009 #include "ApiClient.h"
00010 #include "Parser.h"
00011 #include <QString>
00012 #include "../objects/all_shapes.h" // ShapeId enum located here
00013 #include "../objects/vector.h"
00014
00023 class RenderAreaManager : public QObject {
00024     Q_OBJECT
00025
00026 public:
00034     explicit RenderAreaManager(QObject* parent = nullptr);
00035     ~RenderAreaManager();
00036
00037     alpha::vector<Shape*>* getShapesRef();
00039
00048     void addShape(Shape* shape);
00049     void modifyShape(Shape* shape, QString key, int value);
00050     void modifyDisplayedText(Text* obj, QString newText);
00051     void deleteShape(const int trackerId);
00052     void deleteAllShapes();
00053     void loadShapes();
00054     void saveShapes();
00056
00057
00058 signals:
00066     void renderAreaChanged();
00067     void renderAreaNotChanged(const QString &message);
00068     void statusMessage(const QString &message);
00070
00071
00072 private slots:
00080     void onGoodGetResponse(const QString &json);
00081     void onBadGetResponse(const QString &errorMsg);
00082     void onGoodPostResponse();
00083     void onBadPostResponse(const QString &errorMsg);
00084     void onGoodDeleteResponse();
00085     void onBadDeleteResponse(const QString &errorMsg);
00087
00088
00089 private:
00097     alpha::vector<Shape*> renderedShapes;
00098     ApiClient client;
00099     Parser parse;
00101 };
00102
00103 #endif // RENDER_AREA_MANAGER
```

9.15 src/backend/ShapesManager.cpp File Reference

```
#include "ShapesManager.h"
```

9.16 src/backend/ShapesManager.h File Reference

```
#include <QObject>
#include <QString>
#include <QDebug>
#include "ApiClient.h"
#include "Parser.h"
#include "../objects/all_shapes.h"
#include "../objects/vector.h"
```

Classes

- class ShapesManager

9.16.1 Detailed Description

Note

This class is deprecated and no longer used.

9.17 ShapesManager.h

Go to the documentation of this file.

```
00001
00006
00007 #ifndef SHAPES_MANAGER
00008 #define SHAPES_MANAGER
00009
00010 #include <QObject>
00011 #include <QString>
00012 #include <QDebug>
00013 #include "ApiClient.h"
00014 #include "Parser.h"
00015 #include "../objects/all_shapes.h"
00016 #include "../objects/vector.h"
00017
00018 class ShapesManager : public QObject {
00019     Q_OBJECT
00020 public:
00029     explicit ShapesManager(QObject* parent = nullptr);
00030     ~ShapesManager();
00031
00035     alpha::vector<Shape*>* getShapesRef();
00036
00045     void addShape(Shape* shape);
00046     void modifyShape(Shape* shape);
00047     void deleteShape(const int trackerId);
00048     void deleteAllShapes();
00049     void loadShapes();
00050     void saveShapes();
00051
00052
00053
00054 signals:
00063     void shapesChanged();
00064     void shapesNotChanged(const QString &messsage);
00065     void statusMessage(const QString &message);
00066
00067
00068 private slots:
00074     void onGoodGetResponse(const QString &json);
00075     void onBadGetResponse(const QString &errorMsg);
00076     void onGoodPostResponse();
00077     void onBadPostResponse(const QString &errorMsg);
00078     void onGoodDeleteResponse();
```

```
00079     void onBadDeleteResponse(const QString &errorMsg);
00080
00081
00082 private:
00083     alpha::vector<Shape*> shapes;
00084     ApiClient client;
00085     Parser parse;
00086 };
00087
00088 #endif // SHAPES_MANAGER
```

9.18 src/backend/Testimonial.cpp File Reference

```
#include "Testimonial.h"
```

9.19 src/backend/Testimonial.h File Reference

```
#include <QString>
#include <QDateTime>
#include <QJsonObject>
#include <QJsonArray>
#include <QJsonValue>
```

Classes

- class Testimonial

  Represents a user testimonial.

9.20 Testimonial.h

Go to the documentation of this file.

```
00001 #ifndef TESTIMONIAL_H
00002 #define TESTIMONIAL_H
00003
00004 #include <QString>
00005 #include <QDateTime>
00006 #include <QJsonObject>
00007 #include <QJsonArray>
00008 #include <QJsonValue>
00009
00017 class Testimonial {
00018 public:
00027     Testimonial(const QString& author = "", const QString& content = "", bool isGuest = true);
00029
00037     QString getAuthor() const;
00038     QString getContent() const;
00039     QDateTime getTimestamp() const;
00040     bool isGuest() const;
00041     bool isSatisfactory() const;
00043
00051     void setIsSatisfactory(bool value);
00053
00061     QJsonObject toJson() const;
00062     static Testimonial fromJson(const QJsonObject& json);
00064
00065 private:
00072     QString m_author;
00073     QString m_content;
00074     QDateTime m_timestamp;
00075     bool m_isGuest;
00076     bool m_isSatisfactory;
00078 };
00079
00080 #endif // TESTIMONIAL_H
```

9.21 src/backend/TestimonialManager.cpp File Reference

```
#include "TestimonialManager.h"
```

9.22 src/backend/TestimonialManager.h File Reference

```
#include <QObject>
#include <QVector>
#include <QTimer>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
#include <QDebug>
#include <QHash>
#include "Testimonial.h"
#include "ApiClient.h"
#include "Parser.h"
```

Classes

- class TestimonialManager

  Manages user testimonials, including storage, retrieval, and prompting logic.

9.23 TestimonialManager.h

Go to the documentation of this file.

```
00001 #ifndef TESTIMONIALMANAGER_H
00002 #define TESTIMONIALMANAGER_H
00003
00004 #include <QObject>
00005 #include <QVector>
00006 #include <QTimer>
00007 #include <QJsonDocument>
00008 #include <QJsonArray>
00009 #include <QJsonObject>
00010 #include <QDebug>
00011 #include <QHash>
00012 #include "Testimonial.h"
00013 #include "ApiClient.h"
00014 #include "Parser.h"
00015
00023 class TestimonialManager : public QObject {
00024     Q_OBJECT
00025 public:
00032     static TestimonialManager& getInstance();
00034
00042     void addTestimonial(const Testimonial& testimonial);
00043     QVector<Testimonial> getSatisfactoryTestimonials() const;
00044     bool hasUserGivenTestimonial(const QString& username) const;
00046
00054     void setDoNotShowAgain(const QString& username, bool value);
00055     bool getDoNotShowAgain(const QString& username) const;
00057
00065     void startTrackingTime();
00066     void stopTrackingTime();
00068
00069 signals:
00076     void shouldPromptForTestimonial();
00078
00079 private slots:
```

```
00087     void onGoodGetResponse(const QString& json);
00088     void onBadGetResponse(const QString& error);
00089     void onGoodPostResponse();
00090     void onBadPostResponse(const QString& error);
00092
00093 private:
00100     TestimonialManager();
00101     ~TestimonialManager();
00103
00111     void loadTestimonials();
00112     void saveTestimonials();
00113     void checkTimeAndPrompt();
00115
00122     QVector<Testimonial> m_testimonials;
00123     QTimer* m_trackingTimer;
00124     QHash<QString, bool> m_doNotShowAgain;
00125     QHash<QString, int> m_userTimeTracking;
00126
00127     ApiClient client;
00128     Parser parse;
00129
00130     // timing constants for testimonial prompts
00131     static const int INITIAL_PROMPT_TIME = 30 * 60;
00132     static const int REPEAT_PROMPT_TIME = 60 * 60;
00134 };
00135
00136 #endif // TESTIMONIALMANAGER_H
```

9.24 src/backend/UserAccount.cpp File Reference

```
#include "UserAccount.h"
```

9.25 src/backend/UserAccount.h File Reference

```
#include <QString>
#include <utility>
```

Classes

- class UserAccount

  Represents a user account within the application.

9.26 UserAccount.h

Go to the documentation of this file.

```
00001 #ifndef USERACCOUNT_H
00002 #define USERACCOUNT_H
00003
00004 #include <QString>
00005 #include <utility> // For std::move
00006
00013 class UserAccount {
00014 public:
00022     UserAccount();
00023     UserAccount(QString username, QString password, bool admin);
00024     ~UserAccount();
00026
00034     UserAccount(const UserAccount& other);
00035     UserAccount& operator=(const UserAccount& other);
00036     UserAccount(UserAccount&& other) noexcept;
```

```
00037     UserAccount& operator=(UserAccount&& other) noexcept;
00039
00047     QString getUsername() const;
00048     QString getPassword() const;
00049     bool isAdmin() const;
00051
00059     void setUsername(const QString& username);
00060     void setPassword(const QString& password);
00061     void setAdmin(bool admin);
00062     void setUserAccount(const QString& username, const QString& password, bool admin);
00064
00065 private:
00072     QString username;
00073     QString password;
00074     bool admin;
00076 };
00077
00078 #endif // USERACCOUNT_H
```

9.27 src/backend/UserManager.cpp File Reference

```
#include "UserManager.h"
```

9.28 src/backend/UserManager.h File Reference

```
#include <QObject>
#include <QString>
#include <QDebug>
#include <cctype>
#include <string>
#include "ApiClient.h"
#include "Parser.h"
#include "UserAccount.h"
#include "../objects/vector.h"
```

Classes

- class UserManager

  Manages user accounts, including creation, authentication, and persistence.

9.29 UserManager.h

Go to the documentation of this file.

```
00001 #ifndef USER_MANAGER_H
00002 #define USER_MANAGER_H
00003
00004 #include <QObject>
00005 #include <QString>
00006 #include <QDebug>
00007 #include <cctype>
00008 #include <string>
00009 #include "ApiClient.h"
00010 #include "Parser.h"
00011 #include "UserAccount.h"
00012 #include "../objects/vector.h" // Assuming this is a custom vector implementation
00013
00021 class UserManager : public QObject {
```

```
00022     Q_OBJECT
00023 public:
00032     explicit UserManager(QObject* parent = nullptr);
00033     ~UserManager();
00035
00043     UserAccount* getCurrUserRef();
00045
00054     void addUser(const QString username,
00055                  const QString password,
00056                  const bool admin);
00057     void modifyUser(const QString username,
00058                     const QString password,
00059                     const bool admin);
00060     void deleteUser(QString username);
00061     void deleteAllUsers();
00062     void loadUsers();
00063     void saveUsers();
00064     void authenticate(const QString username,
00065                       const QString password);
00067
00068 signals:
00076     void userChanged();
00077     void userNotChanged(const QString &message);
00078     void statusMessage(const QString &message);
00079     void userAuthenticated(const UserAccount* currUser);
00080     void authenticationFailed(const QString &message);
00082
00083 private slots:
00091     void onGoodGetResponse(const QString &json);
00092     void onBadGetResponse(const QString &errorMsg);
00093     void onGoodPostResponse();
00094     void onBadPostResponse(const QString &errorMsg);
00095     void onGoodDeleteResponse();
00096     void onBadDeleteResponse(const QString &errorMsg);
00098
00099 private:
00106     UserAccount* currUser;
00107     alpha::vector<UserAccount*> users;
00108     ApiClient client;
00109     Parser parse;
00111 };
00112
00113 #endif // USER_MANAGER_H
```

9.30 src/frontend/ColumnEditDelegate.h File Reference

```
#include <QStyledItemDelegate>
```

Classes

- class ColumnEditDelegate

9.31 ColumnEditDelegate.h

Go to the documentation of this file.

```
00001 #ifndef COLUMNEDITDELEGATE_H
00002 #define COLUMNEDITDELEGATE_H
00003
00004 #include <QStyledItemDelegate>
00005
00006 class ColumnEditDelegate : public QStyledItemDelegate
00007 {
00008 public:
00009     ColumnEditDelegate(QObject *parent = nullptr) : QStyledItemDelegate(parent) {}
00010
00011     QWidget *createEditor(QWidget *parent,
00012                           const QStyleOptionViewItem &option,
00013                           const QModelIndex &index) const override
00014     {
```

```
00015         if (index.column() != 1 || !canEdit)
00016         {
00017             return nullptr;  // Only allows editing in column 1
00018         }
00019 
00020         return QStyledItemDelegate::createEditor(parent, option, index);
00021     }
00022 
00023     void setCanEdit(bool edit)
00024     {
00025         canEdit = edit;
00026     }
00027 
00028 
00029 private:
00030     bool canEdit = false;  // Flag to control whether editing is allowed
00031 };
00032 
00033 #endif // COLUMNEDITDELEGATE_H
```

9.32 src/frontend/darkstyle.qss File Reference

9.33 src/frontend/Geoo.qss File Reference

9.34 src/frontend/lightstyle.qss File Reference

9.35 src/frontend/loginwindow.cpp File Reference

```
#include "loginwindow.h"
```

9.36 src/frontend/loginwindow.h File Reference

```
#include <QDialog>
#include <QStackedWidget>
#include <QLineEdit>
#include <QPushButton>
#include <QVBoxLayout>
#include <QFormLayout>
#include <QHBoxLayout>
#include <QLabel>
```

Classes

- class LoginWindow

9.37 loginwindow.h

Go to the documentation of this file.

```
00001 #ifndef LOGIN_WINDOW_H
00002 #define LOGIN_WINDOW_H
00003
00004 #include <QDialog>
00005 #include <QStackedWidget>
00006 #include <QLineEdit>
00007 #include <QPushButton>
00008 #include <QVBoxLayout>
00009 #include <QFormLayout>
00010 #include <QHBoxLayout>
00011 #include <QLabel>
00012
00013 class LoginWindow : public QDialog {
00014     Q_OBJECT
00015
00016 public:
00017     explicit LoginWindow(QWidget *parent = nullptr);
00018
00019     QString username() const;
00020     QString password() const;
00021
00022 signals:
00023     void loginRequested(const QString &username, const QString &password);
00024     void signupRequested(const QString &username, const QString &password, const bool admin);
00025
00026 private slots:
00027     void showSignupPage();
00028     void showLoginPage();
00029     void attemptLogin();
00030     void attemptSignup();
00031
00032 private:
00033     QStackedWidget *stack;
00034
00035     // --- login page widgets
00036     QWidget *loginPage;
00037     QLineEdit *loginUserEdit;
00038     QLineEdit *loginPassEdit;
00039     QPushButton *loginBtn;
00040     QPushButton *toSignupBtn;
00041
00042     // --- signup page widgets
00043     QWidget *signupPage;
00044     QLineEdit *signupUserEdit;
00045     QLineEdit *signupPassEdit;
00046     QPushButton *signupBtn;
00047     QPushButton *backToLoginBtn;
00048 };
00049
00050 #endif // LOGINDIALOG_H
```

9.38 src/backend/main.cpp File Reference

```
#include <QCoreApplication>
#include <QObject>
#include "ApiClient.h"
#include "Parser.h"
#include "UserManager.h"
#include "AppDriver.h"
```

Functions

- void OnGoodGetResponseTest (const std::string &json)
- void OnBadGetResponseTest (const std::string &errorMsg)
- void OnGoodPostResponseTest ()
- void OnBadPostResponseTest (const std::string &errorMsg)

- ApiClient * GetConnectedClient ()
- std::string GetUsersTestString ()
- std::string GetShapeTestString ()
- std::string GetRenderAreaTestString ()
- int main (int argc, char *argv[])

Variables

- QCoreApplication * pApp = nullptr
- ApiClient * pClient = nullptr
- Parser parse

9.38.1 Function Documentation

9.38.1.1 GetConnectedClient()

```
ApiClient * GetConnectedClient ()
```

9.38.1.2 GetRenderAreaTestString()

```
std::string GetRenderAreaTestString ()
```

9.38.1.3 GetShapeTestString()

```
std::string GetShapeTestString ()
```

9.38.1.4 GetUsersTestString()

```
std::string GetUsersTestString ()
```

9.38.1.5 main()

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

9.38.1.6 OnBadGetResponseTest()

```
void OnBadGetResponseTest (
            const std::string & errorMsg)
```

9.38.1.7 OnBadPostResponseTest()

```
void OnBadPostResponseTest (
            const std::string & errorMsg)
```

9.38.1.8 OnGoodGetResponseTest()

```
void OnGoodGetResponseTest (
            const std::string & json)
```

9.38.1.9 OnGoodPostResponseTest()

```
void OnGoodPostResponseTest ()
```

9.38.2 Variable Documentation

9.38.2.1 pApp

```
QCoreApplication* pApp = nullptr
```

9.38.2.2 parse

```
Parser parse
```

9.38.2.3 pClient

```
ApiClient* pClient = nullptr
```

9.39 src/frontend/main.cpp File Reference

Main entry point for the 2D Graphics Modeler application.

```
#include "../backend/AppDriver.h"
#include <QApplication>
```

Functions

- int main (int argc, char *argv[])
  The main function of the application.

9.39.1 Detailed Description

Main entry point for the 2D Graphics Modeler application.

Initializes the QApplication and the AppDriver, then starts the application event loop.

9.39.2 Function Documentation

9.39.2.1 main()

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

The main function of the application.

Parameters

argc	Number of command-line arguments.
argv	Array of command-line arguments.

Returns

Exit status of the application.

9.40 webservice-dockerized/main.cpp File Reference

```
#include <crow.h>
#include <crow/json.h>
#include <string>
#include <fstream>
#include <filesystem>
```

Functions

- crow::json::rvalue get_json_file (const std::string &path)

  Reads and parses a JSON file.
- int main ()

9.40.1 Function Documentation

9.40.1.1 get_json_file()

```
crow::json::rvalue get_json_file (
            const std::string & path)
```

Reads and parses a JSON file.

Opens the file at the specified path, loads its contents into a stringstream, and uses Crow's JSON parser to convert the data into a crow::json::rvalue. Throws a std::runtime_error if the file cannot be opened or parsed.

Parameters

path	The file system path to the JSON file.

Returns

crow::json::rvalue The parsed JSON data.

9.40.1.2 main()

```
int main ()
```

Test endpoint.

Returns a simple greeting message to verify that the web service is running.

Retrieves shapes data.

Reads the JSON data from "/webservice/database/shapes.json", parses it using Crow's JSON parser, and returns the data with the Content-Type header set to "application/json".

Updates shapes.json with new shapes.

Accepts a POST request with JSON data in the body and writes the contents to the file "/webservice/database/shapes.↩ json". If the file cannot be opened, a 500 response is returned.

Clears all shapes.

Deletes the shapes.json file content by truncating the file, leaving it empty (an empty JSON array).

Retrieves render area data.

Reads JSON data from "/webservice/database/render_area.json", parses it using Crow's JSON parser, and returns the data with the Content-Type header set to "application/json".

Updates the render area data.

Accepts a POST request with JSON data and writes it to the file "/webservice/database/render_area.json". Returns a 500 response if the file cannot be opened.

Clears all render area data.

Deletes the render_area.json file content by truncating the file, leaving it empty (an empty JSON array).

Retrieves user account data.

Reads the JSON data from "/webservice/database/users.json", parses it, and returns it with Content-Type "application/json".

Updates users.json with new user data.

Accepts a POST request with JSON body and writes it to "/webservice/database/users.json".

Clears all user account data.

Deletes the users.json file content by truncating it, leaving an empty JSON array.

Retrieves testimonial data.

Reads the JSON array from "/webservice/database/testimonials.json", sets Content-Type to "application/json", and returns the data. Returns 500 if file cannot be opened or parsed.

Creates or updates testimonial data.

Accepts a POST with JSON body containing an array of testimonials or a single testimonial object. Writes the body directly to "/webservice/database/testimonials.json", creating the directory if needed. Returns 200 on success or 500 on failure.

Clears all testimonial data.

Truncates the testimonials.json file at "/webservice/database", writing an empty JSON array. Returns 200 on success or 500 on failure.

9.41 src/frontend/mainwindow.cpp File Reference

```
#include "mainwindow.h"
```

9.42 src/frontend/mainwindow.h File Reference

```
#include <QMainWindow>
#include <algorithm>
#include <QTabWidget>
#include <QTableWidget>
#include <QComboBox>
#include <QApplication>
#include <QGridLayout>
#include <QLabel>
#include <QFile>
#include <QTimer>
#include <QStatusBar>
#include <QVBoxLayout>
#include <QPushButton>
#include <QSpinBox>
#include <QAction>
#include <QMenuBar>
#include "ui_mainwindow.h"
#include "renderarea.h"
#include "loginwindow.h"
#include "TestimonialDialog.h"
#include "TestimonialsDisplayDialog.h"
#include "ColumnEditDelegate.h"
#include "../backend/UserAccount.h"
#include "../backend/TestimonialManager.h"
```

Classes

- class MainWindow

  The main window of the 2D Graphics Modeler application.

Namespaces

- namespace Ui

9.43 mainwindow.h

Go to the documentation of this file.

```
00001 // mainwindow.h
00002 #ifndef MAINWINDOW_H
00003 #define MAINWINDOW_H
00004
00005 #include <QMainWindow>
00006 #include <algorithm>
```

```
00007 #include <QTabWidget>
00008 #include <QTableWidget>
00009 #include <QComboBox>
00010 #include <QApplication>
00011 #include <QGridLayout>
00012 #include <QLabel>
00013 #include <QFile>
00014 #include <QTimer>
00015 #include <QStatusBar>
00016 #include <QVBoxLayout>
00017 #include <QPushButton>
00018 #include <QSpinBox>
00019 #include <QAction>
00020 #include <QMenuBar>
00021 #include "ui_mainwindow.h"
00022 #include "renderarea.h"
00023 #include "loginwindow.h"
00024 #include "TestimonialDialog.h"
00025 #include "TestimonialsDisplayDialog.h"
00026 #include "ColumnEditDelegate.h"
00027 #include "../backend/UserAccount.h"
00028 #include "../backend/TestimonialManager.h"
00029
00030 QT_BEGIN_NAMESPACE
00031 namespace Ui {
00032 class MainWindow;
00033 }
00034 QT_END_NAMESPACE
00035
00042 class MainWindow : public QMainWindow
00043 {
00044     Q_OBJECT
00045
00046 public:
00054     explicit MainWindow(QWidget *parent = nullptr);
00055     MainWindow(QWidget *parent,
00056                const alpha::vector<Shape*>* renderedShapes,
00057                const UserAccount* currUser);
00058     ~MainWindow();
00060
00068     void drawShapes() const;
00069     void shapes_to_treeWidget();
00071
00072 signals:
00080     void shapeAdded(Shape* shape);
00081     void shapeChanged(Shape* shape, QString key, int value);
00082     void displayedTextChanged(Text* text, QString newText);
00083     void shapeDeleted(int trackerId);
00084     void deleteAllShapes();
00086
00094     void loginAttempt(const QString &username, const QString &password);
00095     void newUserAdded(const QString &username, const QString &password,
00096                                                const bool admin);
00098
00106     void loginSuccess();
00107     void loginFailed(const QString &message);
00109
00110 public slots:
00118     void onRenderAreaChanged();
00119     void onRenderAreaNotChanged(const QString& message);
00120     void showRenderStatusMessage(const QString &message);
00122
00129     void onLoginClicked();
00130     void onLoginRequest(const QString &username, const QString &password);
00131     void onSignupRequest(const QString &username, const QString &password,
00132                                                   const bool admin);
00134
00142     void onUserAuthentication(const UserAccount* currUser);
00143     void onUserAuthenticationFailure(const QString& message);
00145
00146 private slots:
00153     void onToggleStyle(bool checked = true);
00155
00163     void on_actionnew_line_button_triggered();
00164     void on_actionnew_square_button_triggered();
00165     void on_actionnew_rectange_button_triggered();
00166     void on_actionnew_circle_button_triggered();
00167     void on_actionnew_ellipse_button_triggered();
00168     void on_actionnew_polyline_button_triggered();
00169     void on_actionnew_polygon_button_triggered();
00170     void on_actionnew_text_button_triggered();
00171     void on_actionremove_shape_button_triggered();
00173
00180     void onSortMethodChanged(int index);
00182
00190     void onTreeWidgetItemChanged(QTreeWidgetItem* item, int column);
00191     void onComboBoxChanged(int newIndex);
```

```
00192     void onSpinBoxChanged();
00194
00201     void showTestimonialPrompt();
00202     void showTestimonialsDisplay();
00204
00211     void onContactUsClicked();
00213
00214 private:
00222     Ui::MainWindow *ui;
00223     RenderArea *renderArea;
00224     const alpha::vector<Shape*>* renderShapes;
00225     const UserAccount* currUser;
00226     QLabel *userStatusLabel;
00227     ColumnEditDelegate* delegate;
00229
00237     void addToShapeTree(Shape* shape);
00238     QString loadStyleSheet(const QString &path);
00240
00250     QComboBox* createShapeTypeComboBox(const QString& currentShapeType);
00251     QSpinBox*  createPenWidthSpinBox(int currentPenWidth);
00252     QComboBox* createColorComboBox(int currentColor);
00253     QComboBox* createPenStyleComboBox(int currentPenStyle);
00254     QComboBox* createPenCapStyleComboBox(int currentPenCapStyle);
00255     QComboBox* createPenJoinStyleComboBox(int currentPenJoinStyle);
00256     QComboBox* createBrushStyleComboBox(int currentBrushStyle);
00257     QComboBox* createAlignmentComboBox(Qt::AlignmentFlag currentAlignment);
00258     QComboBox* createFontComboBox(QFont currentFont);
00259     QComboBox* createFontStyleComboBox(int currentFontStyle);
00260     QComboBox* createFontWeightComboBox(QFont::Weight currentFontWeight);
00262
00270     void createShapeTableTab();
00271     QTabWidget* tabWidget;
00272     QTableWidget* shapeTable;
00273     QComboBox* sortDropdown;
00274     QComboBox* sortOrderDropdown;
00276
00284     void selection_sort(alpha::vector<Shape*>& shapes,
00285                         bool (*compare)(const Shape*, const Shape*),
00286                         bool ascending);
00287     void populateShapeTable(const alpha::vector<Shape*>& shapes);
00288     static bool sortById(const Shape* a, const Shape* b);
00289     static bool sortByArea(const Shape* a, const Shape* b);
00290     static bool sortByPerimeter(const Shape* a, const Shape* b);
00292
00299     void setupTestimonials();
00301
00308     QWidget *contactUsWidget;
00309     QLabel *logoLabel;
00310     QLabel *teamNameLabel;
00311     QWidget *contactWindow;
00313 };
00314
00315 #endif // MAINWINDOW_H
```

9.44 src/frontend/mainwindow.ui File Reference

9.45 src/frontend/Medize.qss File Reference

9.46 src/frontend/renderarea.cpp File Reference

```
#include "renderarea.h"
```

9.47 src/frontend/renderarea.h File Reference

```
#include <QWidget>
#include <QPainter>
#include <QPen>
```

```
#include <QMouseEvent>
#include "../objects/all_shapes.h"
#include "../objects/vector.h"
```

Classes

- class RenderArea

9.48 renderarea.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <QWidget>
00004 #include <QPainter>
00005 #include <QPen>
00006 #include <QMouseEvent>
00007
00008 #include "../objects/all_shapes.h"
00009 #include "../objects/vector.h"
00010
00011 class RenderArea : public QWidget
00012 {
00013     Q_OBJECT
00014
00015 public:
00016     RenderArea(QWidget *parent = nullptr);
00017     void setRenderShapes(const alpha::vector<Shape*>* renderShapes);
00018     const alpha::vector<Shape*>& getShapes() const;
00019
00020     void mousePressEvent(QMouseEvent* event) override;
00021     void mouseMoveEvent(QMouseEvent* event) override;
00022     void mouseDoubleClickEvent(QMouseEvent* event) override;
00023     void mouseReleaseEvent(QMouseEvent* event) override;
00024
00025     void setShapeSelectedIndex(int newIndex);
00026     void resetSelection();
00027     void setEditPrivileges(bool edit);
00028
00029     void updateShapeDisplayCoords(Shape* item, const QPoint& position) const;
00030     int  getShapeSelected() const;
00031     int  getShapeSelectedIndex() const;
00032
00033 protected:
00034     void paintEvent(QPaintEvent *event) override;
00035
00036 private:
00037     const alpha::vector<Shape*>* renderShapes;      // Holds currently renderedShapes
00038     int shapeSelectedIndex; // This is the vector index of the current shape selected, this is done so
    we prevent multiple shapes being selected at once
00039     bool allowEditing = false; // by default you cannot edit shapes
00040 };
```

9.49 src/frontend/resources.qrc File Reference

9.50 src/frontend/TestimonialDialog.cpp File Reference

```
#include "TestimonialDialog.h"
#include "../backend/TestimonialManager.h"
```

9.51 src/frontend/TestimonialDialog.h File Reference

```
#include <QDialog>
#include <QCheckBox>
#include <QTextEdit>
#include <QLineEdit>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QPushButton>
#include <QLabel>
#include <QScrollArea>
```

Classes

- class TestimonialDialog

9.52 TestimonialDialog.h

Go to the documentation of this file.

```
00001 #ifndef TESTIMONIALDIALOG_H
00002 #define TESTIMONIALDIALOG_H
00003
00004 #include <QDialog>
00005 #include <QCheckBox>
00006 #include <QTextEdit>
00007 #include <QLineEdit>
00008 #include <QVBoxLayout>
00009 #include <QHBoxLayout>
00010 #include <QPushButton>
00011 #include <QLabel>
00012 #include <QScrollArea>
00013
00014 // dialog for submitting new testimonials
00015 class TestimonialDialog : public QDialog {
00016     Q_OBJECT
00017
00018 public:
00019     explicit TestimonialDialog(QWidget* parent = nullptr);
00020
00021 private slots:
00022     void onSubmit();
00023     void onCancel();
00024
00025 private:
00026     QLineEdit* m_authorEdit;     // name input
00027     QTextEdit* m_contentEdit;    // testimonial input
00028     QCheckBox* m_doNotShowAgain; // checkbox for future prompts
00029 };
00030
00031 #endif // TESTIMONIALDIALOG_H
```

9.53 src/frontend/TestimonialsDisplayDialog.cpp File Reference

```
#include "TestimonialsDisplayDialog.h"
#include "../backend/TestimonialManager.h"
```

9.54 src/frontend/TestimonialsDisplayDialog.h File Reference

```
#include <QDialog>
#include <QVBoxLayout>
#include "TestimonialDialog.h"
```

Classes

- class TestimonialsDisplayDialog

9.55 TestimonialsDisplayDialog.h

Go to the documentation of this file.

```
00001 #ifndef TESTIMONIALSDISPLAYDIALOG_H
00002 #define TESTIMONIALSDISPLAYDIALOG_H
00003
00004 #include <QDialog>
00005 #include <QVBoxLayout>
00006 #include "TestimonialDialog.h"
00007
00008 // dialog to show all testimonials
00009 class TestimonialsDisplayDialog : public QDialog {
00010     Q_OBJECT
00011
00012 public:
00013     explicit TestimonialsDisplayDialog(QWidget* parent = nullptr);
00014
00015 private:
00016     void refreshTestimonials();
00017     QVBoxLayout* m_testimonialsLayout;
00018 };
00019
00020 #endif // TESTIMONIALSDISPLAYDIALOG_H
```

9.56 src/objects/all_shapes.h File Reference

```
#include "circle.h"
#include "ellipse.h"
#include "line.h"
#include "polygon.h"
#include "polyline.h"
#include "rectangle.h"
#include "shape.h"
#include "square.h"
#include "text.h"
```

Enumerations

- enum ShapeIDs {
  LINE = 1 , POLYLINE , POLYGON , RECTANGLE ,
  SQUARE , ELLIPSE , CIRCLE , TEXT }

9.56.1 Enumeration Type Documentation

9.56.1.1 ShapeIDs

```
enum ShapeIDs
```

Enumerator

LINE	
POLYLINE	
POLYGON	
RECTANGLE	
SQUARE	
ELLIPSE	
CIRCLE	
TEXT	

9.57 all_shapes.h

Go to the documentation of this file.

```
00001 // all_shapes.h | Include this file to automatically include all of these shapes in one line
00002 #ifndef ALL_SHAPES_H
00003 #define ALL_SHAPES_H
00004
00005 #include "circle.h"
00006 #include "ellipse.h"
00007 #include "line.h"
00008 #include "polygon.h"
00009 #include "polyline.h"
00010 #include "rectangle.h"
00011 #include "shape.h"
00012 #include "square.h"
00013 #include "text.h"
00014
00015 enum ShapeIDs {LINE = 1, POLYLINE, POLYGON, RECTANGLE, SQUARE, ELLIPSE, CIRCLE, TEXT};
00016
00017 #endif // ALL_SHAPES_H
```

9.58 src/objects/circle.cpp File Reference

```
#include "circle.h"
```

9.59 src/objects/circle.h File Reference

```
#include "shape.h"
```

Classes

- class Circle

  The Circle class.

9.60 circle.h

Go to the documentation of this file.

```
00001 #ifndef CIRCLE_H
00002 #define CIRCLE_H
00003
00004 #include "shape.h"
00005
00012
00013 class Circle : public Shape
00014 {
00015 public:
00024     Circle(string shapeType,
00025            QPoint coords,
00026            QPen   pen,
00027            QBrush brush,
00028            int r);
00029
00034     void Draw(QWidget* renderArea) override;
00035
00040     double Perimeter() const override;
00041
00046     double Area()      const override;
00047
00053     bool isPointInside(const QPoint& point) const override;
00054
00059     int getR() const;
00060
00062     void setR(int radius);
00063     void setX(int x);
00064     void setY(int y);
00066
00067 private:
00068     int r;
00069 };
00070
00071 #endif // CIRCLE_H
00072
```

9.61 src/objects/ellipse.cpp File Reference

```
#include "ellipse.h"
```

9.62 src/objects/ellipse.h File Reference

```
#include "shape.h"
```

Classes

- class Ellipse

  The Ellipse class.

9.63 ellipse.h

Go to the documentation of this file.

```
00001 #ifndef ELLIPSE_H
00002 #define ELLIPSE_H
00003
00004 #include "shape.h"
00005
00012 class Ellipse : public Shape
00013 {
00014 public:
00024     Ellipse(string shapeType,
00025             QPoint coords,
00026             QPen   pen,
00027             QBrush brush,
00028             int    a,
00029             int    b);
00030
00035     void Draw(QWidget* renderArea) override;
00036
00041     double Perimeter() const override;
00042
00047     double Area()      const override;
00048
00054     bool isPointInside(const QPoint& point) const override;
00055
00057     int getA() const;
00058     int getB() const;
00060
00062     void setA(int newA);
00063     void setB(int newB);
00064     void setX(int newX);
00065     void setY(int newY);
00067
00068 private:
00069     int a;
00070     int b;
00071 };
00072
00073 #endif // ELLIPSE_H
```

9.64 src/objects/line.cpp File Reference

```
#include "line.h"
```

9.65 src/objects/line.h File Reference

```
#include "shape.h"
```

Classes

- class Line

  The Line class.

9.66 line.h

Go to the documentation of this file.

```
00001 #ifndef LINE_H
00002 #define LINE_H
00003
00004 #include "shape.h"
00005
00012 class Line : public Shape
00013 {
00014 public:
00024     Line(string shapeType,
00025          QPoint coords,
00026          QPen   pen,
00027          QBrush brush,
00028          QPoint startPoint,
00029          QPoint endPoint);
00030
00035     void Draw(QWidget* renderArea) override;
00036
00042     void Move(int x, int y) override;
00043
00048     double Perimeter() const override;
00049
00054     double Area() const override { return 0; }  // Need to implement this to instantiate Line
00055
00061     bool isPointInside(const QPoint& point) const override;
00062
00064     QPoint getStartPoint() const;
00065     QPoint getEndPoint()   const;
00066
00068     void setStartPoint(const QPoint& newStartPoint);
00069     void setEndPoint(const QPoint& newEndPoint);
00070     void setX(int newX);
00071     void setY(int newY);
00072
00073 private:
00074     QPoint startPoint;
00075     QPoint endPoint;
00076 };
00077
00078 #endif // LINE_H
```

9.67 src/objects/polygon.cpp File Reference

```
#include "polygon.h"
```

9.68 src/objects/polygon.h File Reference

```
#include "shape.h"
```

Classes

- class Polygon

  The Polygon class.

9.69 polygon.h

Go to the documentation of this file.

```
00001 #ifndef POLYGON_H
00002 #define POLYGON_H
00003
00004 #include "shape.h"
00005
00012 class Polygon : public Shape
00013 {
00014 public:
00023     Polygon(string shapeType,
00024             QPoint coords,
00025             QPen   pen,
00026             QBrush brush,
00027             QPolygon pointsList);
00028
00033     void Draw(QWidget* renderArea) override;
00034
00040     void Move(int x, int y) override;
00041
00046     double Perimeter() const override;
00047
00052     double Area() const override;
00053
00059     bool isPointInside(const QPoint& point) const override;
00060
00065     QPolygon getPointsList() const;
00066
00068
00072     void setPointsList(const QPolygon& newPointsList);
00073     void setX(int newX);
00074     void setY(int newY);
00075
00076 private:
00077     QPolygon pointsList;
00078 };
00079 #endif // POLYGON_H
```

9.70 src/objects/polyline.cpp File Reference

```
#include "polyline.h"
```

9.71 src/objects/polyline.h File Reference

```
#include "shape.h"
```

Classes

- class Polyline

  The Polyline class.

9.72 polyline.h

Go to the documentation of this file.

```
00001 #ifndef POLYLINE_H
00002 #define POLYLINE_H
00003
00004 #include "shape.h"
00005
00012 class Polyline : public Shape
00013 {
00014 public:
00023     Polyline(string shapeType,
00024              QPoint coords,
00025              QPen   pen,
00026              QBrush brush,
00027              QPolygon pointsList);
00028
00033     void Draw(QWidget* renderArea) override;
00034
00040     void Move(int x, int y) override;
00041
00046     double Perimeter() const override;
00047
00052     double Area() const override {return 0;}
00053
00059     bool isPointInside(const QPoint& point) const override;
00060
00065     QPolygon getPointsList() const;
00066
00068
00072     void setPointsList(const QPolygon& newPointsList);
00073     void setX(int newX);
00074     void setY(int newY);
00075
00076 private:
00077     QPolygon pointsList;
00078 };
00079
00080 #endif // POLYLINE_H
```

9.73 src/objects/rectangle.cpp File Reference

```
#include "rectangle.h"
```

9.74 src/objects/rectangle.h File Reference

```
#include "shape.h"
```

Classes

- class Rectangle

  The Rectangle class.

9.75 rectangle.h

Go to the documentation of this file.

```
00001 #ifndef RECTANGLE_H
00002 #define RECTANGLE_H
00003
00004 #include "shape.h"
00005
00012 class Rectangle : public Shape
00013 {
00014 public:
00024     Rectangle(string shapeType,
00025               QPoint coords,
00026               QPen   pen,
00027               QBrush brush,
00028               int length,
00029               int width);
00030
00035     void Draw(QWidget* renderArea) override;
00036
00041     double Perimeter() const override;
00042
00047     double Area()      const override;
00048
00054     bool isPointInside(const QPoint& point) const override;
00055
00057     int getLength() const;
00058     int getWidth()  const;
00059
00061     void setLength(int newLength);
00062     void setWidth(int newWidth);
00063     void setX(int newX);
00064     void setY(int newY);
00065
00066 private:
00067     int length;
00068     int width;
00069 };
00070
00071 #endif // RECTANGLE_H
00072
```

9.76 src/objects/shape.cpp File Reference

```
#include "shape.h"
```

Functions

- bool operator== (const Shape &shape1, const Shape &shape2)
- bool operator< (const Shape &shape1, const Shape &shape2)

9.76.1 Function Documentation

9.76.1.1 operator<()

```
bool operator< (
            const Shape & shape1,
            const Shape & shape2)
```

Parameters

shape1	
shape2	

Returns

9.76.1.2 operator==()

```
bool operator== (
            const Shape & shape1,
            const Shape & shape2)
```

Parameters

shape1	
shape2	

Returns

9.77 src/objects/shape.h File Reference

```
#include <string>
#include <cmath>
#include <QWidget>
#include <QColor>
#include <QFont>
#include <QPen>
#include <QPainter>
#include <QList>
#include <QPolygon>
#include <QTreeWidget>
#include <QComboBox>
#include "vector.h"
```

Classes

- interface Shape

  The Shape Abstract Base Class.

Variables

- const double PI = 3.14

9.77.1 Variable Documentation

9.77.1.1 PI

```
const double PI = 3.14
```

9.78 shape.h

Go to the documentation of this file.

```
00001 #ifndef SHAPE_H
00002 #define SHAPE_H
00003
00004 #include <string>
00005 #include <cmath>
00006 #include <QWidget>
00007 #include <QColor>
00008 #include <QFont>
00009 #include <QPen>
00010 #include <QPainter>
00011 #include <QList>
00012 #include <QPolygon>
00013 #include <QTreeWidget>
00014 #include <QComboBox>
00015 #include "vector.h"
00016
00017
00018 using std::string;
00019
00020 using Qt::GlobalColor;
00021 using Qt::PenCapStyle;
00022 using Qt::PenStyle;
00023 using Qt::PenJoinStyle;
00024 using Qt::BrushStyle;
00025 using Qt::AlignmentFlag;
00026
00027 const double PI = 3.14;
00028
00035
00036 class Shape
00037 {
00044     friend bool operator==(const Shape& shape1, const Shape& shape2);
00051     friend bool operator<(const Shape& shape1, const Shape& shape2);
00052
00053 public:
00054
00062     Shape(int shapeId,
00063           string shapeType,
00064           QPoint coords,
00065           QPen   pen,
00066           QBrush brush);
00067
00071     virtual ~Shape();
00072
00073     static int nextTracker[9];
00074     static bool trackersInUse[9000];
00075
00082     virtual void Draw(QWidget* renderArea) = 0;
00083
00089     virtual void Move(int x, int y);
00090
00097     virtual double Perimeter() const = 0;
00098
00105     virtual double Area()      const = 0;
00106
00114     virtual bool isPointInside(const QPoint& point) const = 0;
00115
00119     void CreateParentItem();
00120
00124     void CreatePenChild();
00125
00129     void CreateBrushChild();
00130
00135     void CreatePointsChild(const int POINTS_NUM);
00136
00138     int    getShapeId()   const;
00139     int    getTrackerId() const;
00140     string getShapeType() const;
00141     bool   getSelected()  const;
00142
00143     int getX() const;
00144     int getY() const;
00145
00146     QPainter& getPainter();
00147     QTreeWidgetItem* getParentItem();
00148     alpha::vector<QTreeWidgetItem*>& getChildItems();
00149     alpha::vector<QTreeWidgetItem*>& getPointsItems();
00150     alpha::vector<QTreeWidgetItem*>& getPenItems();
00151     alpha::vector<QTreeWidgetItem*>& getBrushItems();
00152
00153     int          getPenWidth()      const;
00154     PenStyle     getPenStyle()      const;
```

```
00155     PenCapStyle  getPenCapStyle()   const;
00156     PenJoinStyle getPenJoinStyle()  const;
00157     QColor       getPenColor()      const;
00158     QColor       getBrushColor()    const;
00159     BrushStyle   getBrushStyle()    const;
00160     QPen         getPen()           const;
00161     QBrush       getBrush()         const;
00162     QPoint       getPoints()        const;
00163     int          getChildEnd()      const;
00164     int          getPenItemsEnd()   const;
00165     int          getBrushItemsEnd() const;
00167
00169     void setShapeType(string shapeType);
00170     void setSelected(bool selected);
00171
00172     void setTrackerId(int trackerId);
00173     void allocateTrackerId(int shapeId);
00174
00175     void setX(int x);
00176     void setY(int y);
00177
00178     void setPen(GlobalColor penColor, int penWidth, PenStyle penStyle, PenCapStyle penCapStyle,
      PenJoinStyle penJoinStyle);
00179     void setBrush(GlobalColor brushColor, BrushStyle brushStyle);
00180
00181     // These functions make it easier to change pen and brush properties in
      RenderAreaManager::modifyShape()
00182     QPen& setInternalPen();
00183     QBrush& setInternalBrush();
00185
00186 protected:
00187     QTreeWidgetItem* parentItem;
00188     alpha::vector<QTreeWidgetItem*> childItems;
00189     alpha::vector<QTreeWidgetItem*> pointsItems;
00190     alpha::vector<QTreeWidgetItem*> penItems;
00191     alpha::vector<QTreeWidgetItem*> brushItems;
00192
00193 private:
00194     const int shapeId;
00195     int      trackerId;
00196     string   shapeType;
00197
00198     QPen     pen;
00199     QBrush   brush;
00200     QPoint   coords;
00201
00202     QPainter painter;
00203
00204     bool isSelected = false;
00205
00206     // Disable Copy Operations
00207     Shape(Shape& shape) = delete;
00208     Shape& operator=(Shape& object) = delete;
00209 };
00210
00211 #endif // SHAPE_H
00212
```

9.79 src/objects/square.cpp File Reference

```
#include "square.h"
```

9.80 src/objects/square.h File Reference

```
#include "shape.h"
```

Classes

- class Square

  The Square class.

9.81 square.h

Go to the documentation of this file.

```
00001 #ifndef SQUARE_H
00002 #define SQUARE_H
00003
00004 #include "shape.h"
00005
00012 class Square : public Shape
00013 {
00014 public:
00023     Square(string shapeType,
00024            QPoint coords,
00025            QPen   pen,
00026            QBrush brush,
00027            int length);
00028
00033     void Draw(QWidget* renderArea) override;
00034
00039     double Perimeter() const override;
00040
00045     double Area()      const override;
00046
00052     bool isPointInside(const QPoint& point) const override;
00053
00055     int getLength() const;
00056
00058     void setLength(int newLength);
00059     void setX(int newX);
00060     void setY(int newY);
00061
00062 private:
00063     int length;
00064 };
00065
00066 #endif // SQUARE_H
```

9.82 src/objects/text.cpp File Reference

```
#include "text.h"
```

9.83 src/objects/text.h File Reference

```
#include "shape.h"
```

Classes

- class Text

  The Text class.

9.84 text.h

Go to the documentation of this file.

```
00001 #ifndef TEXT_H
00002 #define TEXT_H
00003
00004 #include "shape.h"
00005
```

```
00012 class Text : public Shape
00013 {
00014 public:
00026     Text(string shapeType,
00027          QPoint coords,
00028          QString textString,
00029          GlobalColor   textColor,
00030          AlignmentFlag textAlignment,
00031          QFont font,
00032          int length,
00033          int width);
00034
00039     void Draw(QWidget* renderArea) override;
00040
00045     double Perimeter() const override;
00046
00051     double Area()      const override;
00052
00058     bool isPointInside(const QPoint& point) const override;
00059
00061     int           getLength()        const;
00062     int           getWidth()         const;
00063     QString       getTextString()    const;
00064     GlobalColor   getTextColor()     const;
00065     QFont         getFont()          const;
00066     AlignmentFlag getTextAlignment() const;
00067     int           getFontStyle()     const;
00068     QFont::Weight getFontWeight()    const;
00069
00071     void setText(QString text);
00072     void setLength(int newLength);
00073     void setWidth(int newWidth);
00074     void setX(int newX);
00075     void setY(int newY);
00076     void setAlignment(Qt::AlignmentFlag alignment);
00077     QFont& setInternalFont();
00078
00079
00080 private:
00081     int length;
00082     int width;
00083
00084     QString       textString;
00085     GlobalColor   textColor;
00086     QFont         font;
00087     AlignmentFlag textAlignment;
00088 };
00089
00090 #endif // TEXT_H
```

9.85 src/objects/vector.h File Reference

Classes

- class alpha::vector< T >

Namespaces

- namespace alpha

Macros

- #define ALPHA_VECTOR_H

9.85.1 Macro Definition Documentation

9.85.1.1 ALPHA_VECTOR_H

```
#define ALPHA_VECTOR_H
```

9.86 vector.h

Go to the documentation of this file.

```
00001 #ifndef ALPHA_VECTOR_H
00002 #define ALPHA_VECTOR_H  //This is necessary for inclusion guards. DO NOT DELETE
00003 /*******************************************************************************
00004  * vector.h
00005  * -----------------------------------------------------------------------------
00006  * Worked on by: Aram, Aspen, Luke
00007  * -----------------------------------------------------------------------------
00008  * This vector will be used to hold the shapes that will be displayed.
00009  * It supports the following basic operations:
00010  *      - constructors for one or more arguments
00011  *      - default constructors
00012  *      - copy constructor
00013  *      - copy assignment
00014  *      - move constructor
00015  *      - move assignment
00016  *      - destructor
00017  *
00018  * Vector also supports:
00019  *      - a basic iterator member type and member function
00020  *      - begin()
00021  *      - end() operations
00022 *******************************************************************************/
00023 namespace alpha {
00024 template<class T>
00025
00026 class vector
00027 {
00028     int size_v;     // the size
00029     T* elem;        // a pointer to the elements
00030     int space;      // size + free_space
00031
00032     public:
00033         /*************************************************************************
00034          * DEFAULT CONSTRUCTOR
00035          *************************************************************************/
00036         vector() : size_v(0), space(1) {
00037             elem = new T[space];
00038         } // END vector()
00039
00040         /*************************************************************************
00041          * ALTERNATE CONSTRUCTOR - size given
00042          *************************************************************************/
00043         explicit vector(int s) : size_v(s), space(s) {
00044             elem = new T[space];
00045         } // END explicit vector(int s)
00046
00047         /*************************************************************************
00048          * COPY CONSTRUCTOR
00049          *************************************************************************/
00050         vector(const vector& other) : size_v(other.size_v), space(other.space) {
00051             elem = new T[space];
00052
00053             for (int i = 0; i < size_v; i++) {
00054                 elem[i] = other.elem[i];
00055             } // END for (int i = 0; i < size_v; i++)
00056         } // END vector(const vector& other)
00057
00058         /*************************************************************************
00059          * COPY ASSIGNMENT
00060          *************************************************************************/
00061         vector& operator=(const vector& other) {
00062             /*********************************************************************
00063             * CHECKS IF SELF-ASSIGNMENT
00064             *********************************************************************/
00065             if (this == &other) {
00066                 return *this;
00067             }
00068
00069             /*********************************************************************
00070             * IF NOT SELF-ASSIGNMENT
00071             *********************************************************************/
00072             delete[] elem;
00073
00074             size_v = other.size_v;
00075             space = other.space;
00076             elem = new T[space];
00077
00078             for (int i = 0; i < size_v; i++) {
00079                 elem[i] = other.elem[i];
00080             }
00081
00082             return *this;
```

```
00083         } // END vector& operator=vector
00084
00085         /**************************************************************************
00086          * MOVE CONSTRUCTOR
00087          *************************************************************************/
00088         vector(vector&& other) noexcept
00089             : size_v(other.size_v), elem(other.elem), space(other.space) {
00090             other.size_v = 0;
00091             other.elem = nullptr;
00092             other.space = 0;
00093         }
00094
00095         /**************************************************************************
00096          * MOVE ASSIGNMENT
00097          *************************************************************************/
00098         vector& operator=(vector&& other) noexcept {
00099             /**********************************************************************
00100             * CHECKS IF SELF-ASSIGNMENT
00101             **********************************************************************/
00102             if (this == &other) {
00103                 return *this;
00104             }
00105
00106             /**********************************************************************
00107             * IF NOT SELF-ASSIGNMENT
00108             **********************************************************************/
00109             delete[] elem;
00110
00111             size_v = other.size_v;
00112             space = other.space;
00113             elem = other.elem;
00114
00115             other.size_v = 0;
00116             other.space = 0;
00117             other.elem = nullptr;
00118
00119            return *this;
00120         } // END vector& operator=(const vector&& other) noexcept
00121
00122         /**************************************************************************
00123          * DESTRUCTOR
00124          *************************************************************************/
00125         ~vector() { delete[] elem; }
00126
00127         /**************************************************************************
00128          * ACCESSOR - RETURN REFERENCE - MODIFIABLE
00129          *************************************************************************/
00130         T& operator[] (int n) { return elem[n]; }
00131
00132         /**************************************************************************
00133          * ACCESSOR - RETURN REFERENCE
00134          *************************************************************************/
00135         const T& operator[] (int n) const { return elem[n]; }
00136
00137         /**************************************************************************
00138          * ACCESSOR - RETURN CURRENT SIZE
00139          *************************************************************************/
00140         int size() const { return size_v; }
00141
00142         /**************************************************************************
00143          * ACCESSOR - RETURN CURRENT AVAILABLE SPACE
00144          *************************************************************************/
00145         int capacity() const { return space; }
00146
00147         /**************************************************************************
00148          * void resize(int newsize)
00149          * ------------------------------------------------------------------------
00150          * This function will be passed a number:
00151          *      newsize     - size to increase vector by
00152          *
00153          * depending on size_v the following will happen:
00154          *      size_v = newsize
00155          *      nothing will happen
00156          *
00157          *      size_v < newsize
00158          *      will change size_v to newsize
00159          *
00160          *      size_v > newsize (default)
00161          *      error will occur
00162          * ------------------------------------------------------------------------
00163          * PRE-CONDITIONS
00164          *      size_v  - original size of vector
00165          *
00166          * POST-CONDITIONS
00167          *      newsize  - new size of vector
00168          *************************************************************************/
00169         void resize(int newsize) {
```

```
00170
00171             /*****************************************************************
00172             * DOES NOTHING - EQUALS EACH OTHER
00173             *****************************************************************/
00174             if (size_v == newsize) {}
00175
00176             /*****************************************************************
00177             * OG SIZE LESS THAN NEW SIZE
00178             *****************************************************************/
00179             else if (size_v < newsize) {
00180                 reserve(newsize);
00181                 size_v = newsize;
00182             } // END else if (size_v < newsize)
00183
00184             /*****************************************************************
00185             * DOES NOTHING - DEFAULT - SMALLER THAN OG / INVALID INPUT / ERROR
00186             *****************************************************************/
00187             else { }
00188
00189         } // END void resize(int newsize)
00190
00191         /*********************************************************************
00192          * FUNCTION - ADD ELEMENT
00193          * ------------------------------------------------------------------
00194          * This function will be passed a value:
00195          *      val     - data to add to vector
00196          *
00197          * Function adds the data to the back of the vector. If there is no
00198          * space then the function will increase the space before adding
00199          * ------------------------------------------------------------------
00200          * PRE-CONDITIONS
00201          *      size_v  - original size of vector
00202          *
00203          * POST-CONDITIONS
00204          *      newsize  - new size of vector
00205          *********************************************************************/
00206         void push_back(const T val) {
00207             /*****************************************************************
00208              * IF NO SPACE - MAKE ROOM
00209              *****************************************************************/
00210             if (size_v == space) {
00211                 reserve(space + 1);
00212             }
00213             /*****************************************************************
00214              * IF SPACE
00215              *****************************************************************/
00216             elem[size_v] = val;
00217             size_v++;
00218         } // END void push_back(const T val)
00219
00220          /*********************************************************************
00221          * void reserve(int newalloc)
00222          * ------------------------------------------------------------------
00223          * This function will be passed a number:
00224          *      newalloc     - size to increase vector capacity by
00225          *
00226          * depending on space the following will happen:
00227          *      space = newalloc
00228          *      nothing will happen
00229          *
00230          *      space < newalloc
00231          *      will change space to newalloc
00232          *
00233          *      size_v > newalloc (default)
00234          *      error will occur
00235          * ------------------------------------------------------------------
00236          * PRE-CONDITIONS
00237          *      space  - original size of vector
00238          *
00239          * POST-CONDITIONS
00240          *      newalloc  - new size of vector
00241          *********************************************************************/
00242         void reserve(int newalloc) {
00243             /*****************************************************************
00244             * DOES NOTHING - EQUALS EACH OTHER
00245             *****************************************************************/
00246             if (space == newalloc) {}
00247
00248             /*****************************************************************
00249             * OG SIZE LESS THAN NEW SIZE
00250             *****************************************************************/
00251             else if (space < newalloc) {
00252
00253                 // CREATE BIGGER ARRAY
00254                 T* new_elem = new T[newalloc];
00255
00256                 // COPY DATA
```

```
00257                 for (int i = 0; i < size_v; i++) {
00258                     new_elem[i] = elem[i];
00259                 }
00260 
00261                 delete[] elem;
00262                 elem = new_elem;
00263 
00264                 // UPDATE SPACE
00265                 space = newalloc;
00266             } // END else if (space < newalloc)
00267 
00268             /*************************************************************
00269             * DOES NOTHING - DEFAULT - SMALLER THAN OG / INVALID INPUT / ERROR
00270             *************************************************************/
00271             else { }
00272 
00273         } // END void reserve(int newalloc)
00274 
00275 
00276         using iterator = T*;
00277         using const_iterator = const T*;
00278 
00279         /*************************************************************************
00280          * POINTS TO FIRST ELEMENT
00281          *************************************************************************/
00282         iterator begin() { return elem; }
00283 
00284         /*************************************************************************
00285          * CONSTANT - POINTS TO FIRST ELEMENT
00286          *************************************************************************/
00287         const_iterator begin() const { return elem; }
00288 
00289         /*************************************************************************
00290          * POINTS TO ONE BEYOND THE LAST ELEMENT
00291          *************************************************************************/
00292         iterator end() { return elem + size_v; }
00293 
00294         /*************************************************************************
00295          * CONSTANT - POINTS TO ONE BEYOND THE LAST ELEMENT
00296          *************************************************************************/
00297         const_iterator end() const { return elem + size_v; }
00298 
00299         /*************************************************************************
00300          * INSERT NEW ELEMENT (V) BEFORE P
00301          *************************************************************************/
00302         iterator insert(iterator p, const T& v) {
00303             /*********************************************************************
00304              * CHECK IF ENOUGH SPACE
00305              *********************************************************************/
00306             // record the index where we want to insert
00307             int index = static_cast<int>(p - elem);
00308 
00309             // grow storage if needed
00310             if (size_v == space) {
00311                 reserve(space + 1);
00312             }
00313 
00314             // iterator next = end();
00315             // while (next != p) {
00316             //     *next = *(next - 1);
00317             //     next--;
00318             // }
00319 
00320             // recompute pointers into the (possibly moved) data
00321             p = elem + index;
00322             iterator end_it = elem + size_v;
00323 
00324             // shift elements [index..size_v-1] one position to the right
00325             for (iterator it = end_it; it != p; --it)
00326                 *it = *(it - 1);
00327 
00328             *p = v;
00329             size_v++;
00330 
00331             return p;
00332         } // END iterator insert(iterator p, const T& v)
00333 
00334         /*************************************************************************
00335          * REMOVE ELEMENT POINTED TO BY P
00336          *************************************************************************/
00337         iterator erase(iterator p) {
00338             /*********************************************************************
00339              * IF P AT THE END
00340              *********************************************************************/
00341             if (p == end()) {
00342                 return p;
00343             }
```

```
00344
00345             iterator next = p + 1;
00346             while (next != end()) {
00347                 *p = *next;
00348                 p++;
00349                 next++;
00350             }
00351
00352             p--;
00353             p->~T();
00354             size_v--;
00355
00356             return p;
00357         } // END iterator erase(iterator p)
00358
00359 }; // END class vector
00360 }; // END namespace alpha
00361 #endif // ALPHA_VECTOR_H
```

9.87 src/webservice/webservice.cpp File Reference

Implements the Crow web service for handling shapes, render area, user, and testimonial data.

```
#include <crow.h>
#include <crow/json.h>
#include <string>
#include <fstream>
#include <filesystem>
```

Functions

- crow::json::rvalue get_json_file (const std::string &path)

  Reads and parses a JSON file.
- int main ()

9.87.1 Detailed Description

Implements the Crow web service for handling shapes, render area, user, and testimonial data.

This web service provides the following endpoints, grouped by entity:

- TEST GET "/" : A test endpoint that returns a greeting.
- Shapes Endpoints:
- GET /shapes : Returns the JSON-formatted shapes data from ../../database/shapes.json.
- POST /shapes : Accepts JSON data to update the shapes file (../../database/shapes.json).
- DELETE /shapes-all : Clears all shapes data by truncating the shapes.json file.
- Render Area Endpoints:
- GET /render_area : Returns the JSON-formatted render area data from ../../database/render_area.json.
- POST /render_area : Accepts JSON data to update the render area file (../../database/render_area.json).
- DELETE /render_area-all : Clears all render area data by truncating the render_area.json file.
- Users Endpoints:

- GET /users : Returns the JSON-formatted user data from ../../database/users.json.
- POST /users : Accepts JSON data to update the user file (../../database/users.json).
- DELETE /users-all : Clears all user data by truncating the users.json file.
- Testimonials Endpoints:
- GET /testimonials : Returns JSON-formatted testimonial data from ../../database/testimonials.json.
- POST /testimonials : Accepts JSON data to update the testimonials file (../../database/testimonials.json).
- DELETE /testimonials : Clears all testimonial data by truncating the testimonials.json file. (Note: Original was DELETE /testimonials, implying delete all)

Note

Uses the Crow framework for handling HTTP requests. Make sure the database directory exists or is created.

This web service provides the following endpoints, grouped by entity:

- TEST GET "/" : A test endpoint that returns a greeting.
- Shapes Endpoints:
- GET /shapes : Returns the JSON-formatted shapes data from /webservice/database/shapes.json.
- POST /shapes : Accepts JSON data to update the shapes file (/webservice/database/shapes.json).
- DELETE /shapes-all : Clears all shapes data by truncating the shapes.json file.
- Render Area Endpoints:
- GET /render_area : Returns the JSON-formatted render area data from /webservice/database/render_↩ area.json.
- POST /render_area : Accepts JSON data to update the render area file (/webservice/database/render_↩ area.json).
- DELETE /render_area-all : Clears all render area data by truncating the render_area.json file.
- Users Endpoints:
- GET /users : Returns the JSON-formatted user data from /webservice/database/users.json.
- POST /users : Accepts JSON data to update the user file (/webservice/database/users.json).
- DELETE /users-all : Clears all user data by truncating the users.json file.
- Testimonials Endpoints:
- GET /testimonials : Returns JSON-formatted testimonial data from /webservice/database/testimonials.json.
- POST /testimonials : Accepts JSON data to update the testimonials file (/webservice/database/testimonials.json).
- DELETE /testimonials : Clears all testimonial data by truncating the testimonials.json file.

Note

Uses the Crow framework for handling HTTP requests. Make sure the database directory exists or is created.

9.87.2 Function Documentation

9.87.2.1 get_json_file()

```
crow::json::rvalue get_json_file (
            const std::string & path)
```

Reads and parses a JSON file.

Opens the file at the specified path, loads its contents into a stringstream, and uses Crow's JSON parser to convert the data into a crow::json::rvalue. Throws a std::runtime_error if the file cannot be opened or parsed.

Parameters

path	The file system path to the JSON file.

Returns

crow::json::rvalue The parsed JSON data.

9.87.2.2 main()

```
int main ()
```

Test endpoint.

Returns a simple greeting message to verify that the web service is running.

Retrieves shapes data.

Reads the JSON data from "../../database/shapes.json", parses it using Crow's JSON parser, and returns the data with the Content-Type header set to "application/json".

Updates shapes.json with new shapes.

Accepts a POST request with JSON data in the body and writes the contents to the file "../../database/shapes.json". If the file cannot be opened, a 500 response is returned.

Clears all shapes.

Deletes the shapes.json file content by truncating the file, leaving it empty (an empty JSON array).

Retrieves render area data.

Reads JSON data from "../../database/render_area.json", parses it using Crow's JSON parser, and returns the data with the Content-Type header set to "application/json".

Updates the render area data.

Accepts a POST request with JSON data and writes it to the file "../../database/render_area.json". Returns a 500 response if the file cannot be opened.

Clears all render area data.

Deletes the render_area.json file content by truncating the file, leaving it empty (an empty JSON array).

Retrieves user account data.

Reads the JSON data from "../../database/users.json", parses it, and returns it with Content-Type "application/json".

Updates users.json with new user data.

Accepts a POST request with JSON body and writes it to "../../database/users.json".

Clears all user account data.

Deletes the users.json file content by truncating it, leaving an empty JSON array.

Retrieves testimonial data.

Reads the JSON array from "../../database/testimonials.json", sets Content-Type to "application/json", and returns the data. Returns 500 if file cannot be opened or parsed.

Creates or updates testimonial data.

Accepts a POST with JSON body containing an array of testimonials or a single testimonial object. Writes the body directly to "../../database/testimonials.json", creating the directory if needed. Returns 200 on success or 500 on failure.

Clears all testimonial data.

Truncates the testimonials.json file, writing an empty JSON array. Returns 200 on success or 500 on failure.

Index