Software Architect Design

Home Application

Version 1.1
Prepared by ThangTV

Hanoi 19/10/2021

Version	Editer	Reviewer	Date	Page	Description
1.0	ThangTV	ThanhPB	19-Oct-2021	All	Created
1.1	ThangTV	ThanhPB	30-Oct-2021	All	Updated

Contents

I: UI design

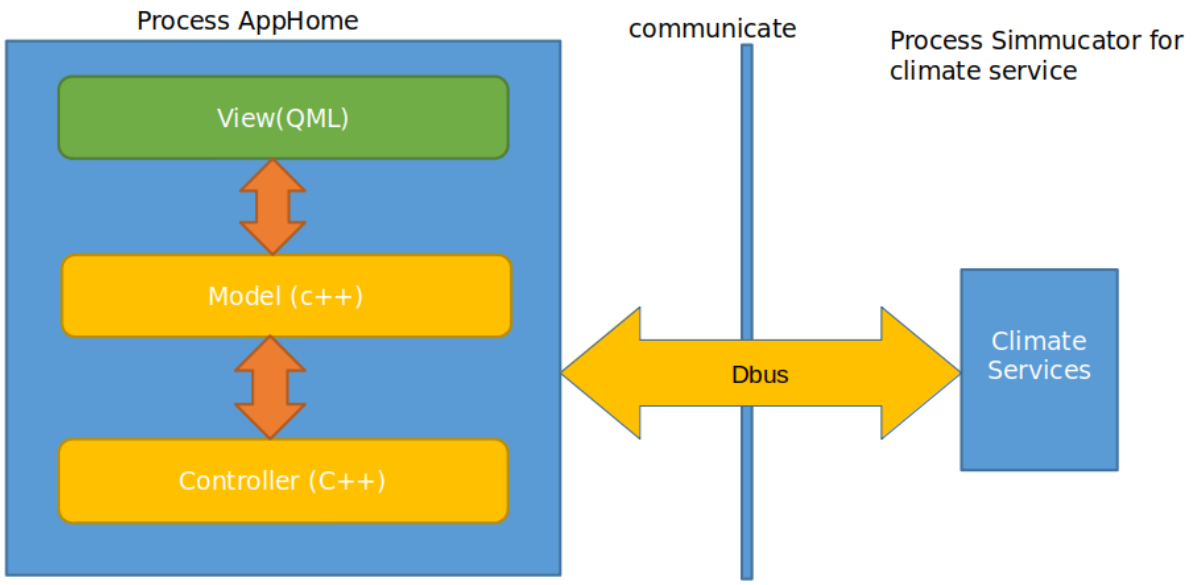
Note: detail refer to UI_FX04527_ThangTV_v1_1.pdf

II: UX design

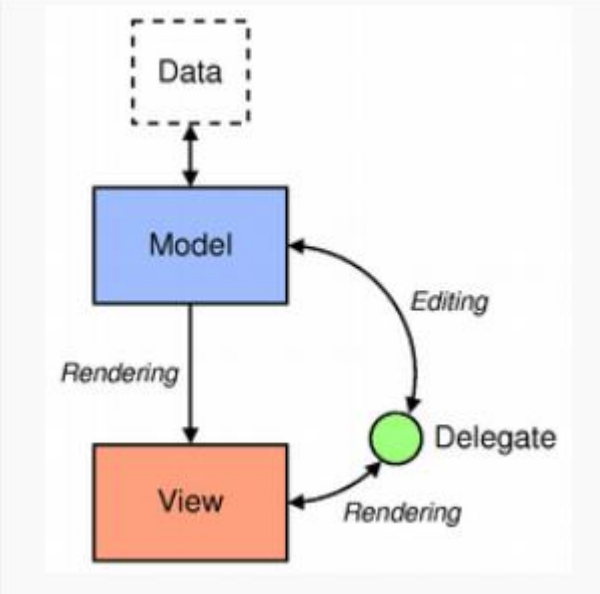
Note: detail refer to UX_FX04527_ThangTV_v1_1.pdf

III: Software architech design

1: Architech design common



- **View (QML)**: this is where manage the screens of the Home Application. The components made by QML and the resources to build the screens.
- **Model(c++)**: this is where the data is built, used to manage the state of the C++ interface. The where the data for constructing the states of the screen is presented.
- **Controller(c++)**: handle, control to the program. Responsible for interactive with third services (climate services).
- **Dbus**: D-Bus is an Inter-Process Communication (IPC) and Remote Procedure Calling (RPC) mechanism originally developed for Linux to replace existing and competing IPC solutions with one unified protocol. It has also been designed to allow communication between system-level processes (such as printer and hardware driver services) and normal user processes. Communication in general happens through a central server application, called the "bus" (hence the name), but direct application-to-application communication is also possible. When communicating on a bus, applications can query which other applications and services are available, as well as activate one on demand.
- The architect designed by MVC model, this is popular model used to build software application.



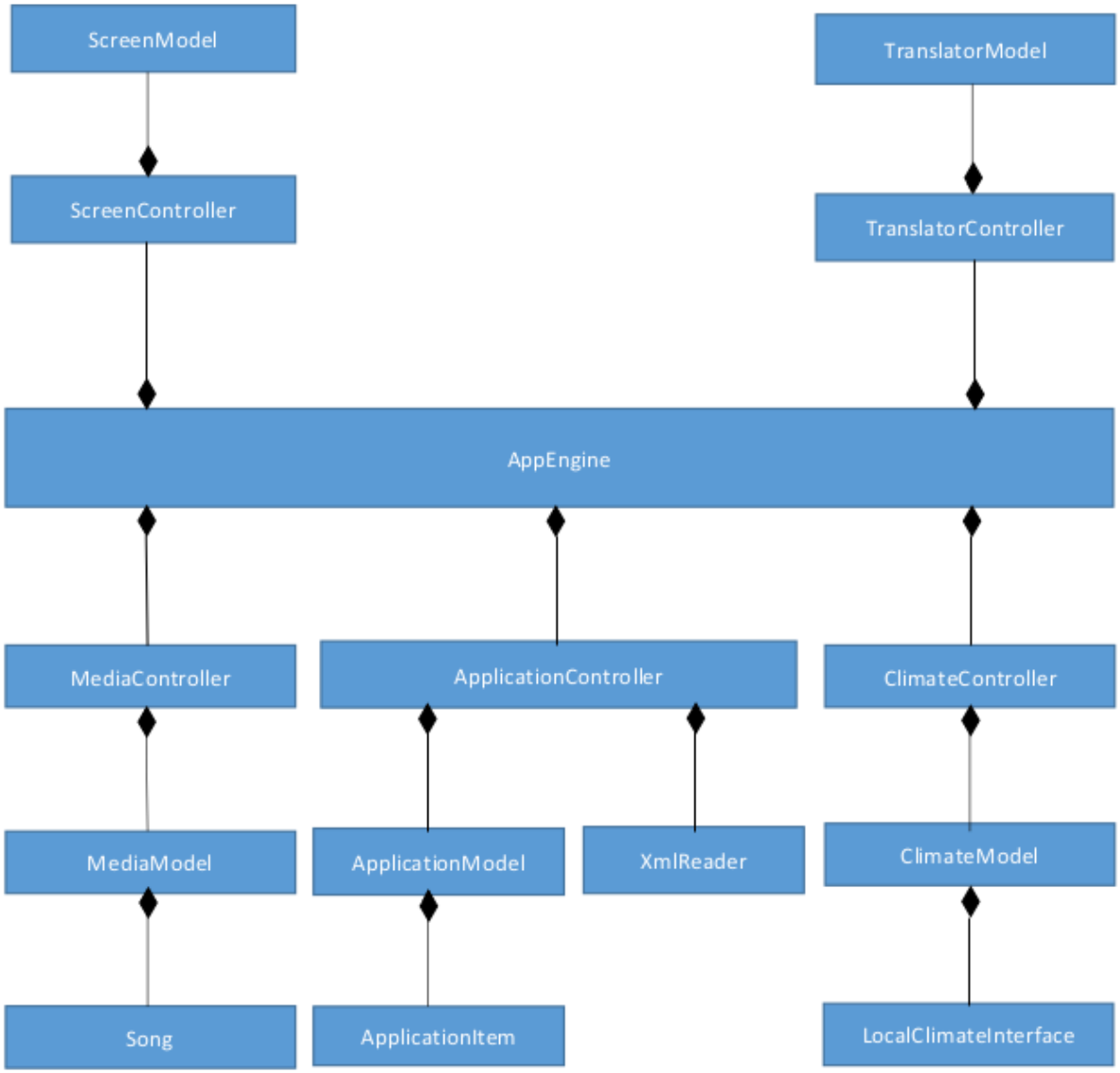
The model/view architecture

The model communicates with a source of data, providing an *interface* for the other components in the architecture. The nature of the communication depends on the type of data source, and the way the model is implemented.

The view obtains *model indexes* from the model; these are references to items of data. By supplying model indexes to the model, the view can retrieve items of data from the data source.

In standard views, a *delegate* renders the items of data. When an item is edited, the delegate communicates with the model directly using model indexes.

2: Class diagram



3:Class detail

3.1 Class Screenmodel

Attribute:

Property	Type	Description
m_currentScreen	QString	Name of the current screen

Method:

Method	Parameter	Type	Description
ScreenModel	-	Constructors	Initialize the class when declare an object
~ScreenModel	-	Destructors	Delete an object of class
currentScreen	-	QString	Get m_currentScreen variable of the class
setCurrentcreent	const QString &currentScreen	void	Set value for m_currentScreen variable of the class
currentScreentChanged	-	void	Signal emits when the screen be changed

3.2 Class ScreenController

Attribute:

Property	Type	Description
m_instance	ScreenController *	An Object pointer of the class, use for singleton pattern
m_initialized	bool	Flag initialize of the class
m_itemFocus	QString	Contain the item is focusing on Home Screen
m_model	ScreenModel	Model of the screen
m_screenStack	QStack<int>	Contain the index of the screen when application run
m_engine	QQmlApplicationEngine *	Qml engine used to set context of c++ object , using on QML.

Method:

Method	Parameter	Type	Description
ScreenController	-	Constructors	-
~ScreenController	-	Destructor	-
getInstance	-	ScreenController *	return root pointer of the class, use singleton pattern
Initialize	QQmlContext *context	void	Set context of c++ object to use on QML
initializeScreen	QQmlApplicationEngine *engine	void	Initialize the Home Screen when application load
pushScreen	QString url	void	Push history of the screen action to stack
replaceScreen	int screenId	void	Replace the current screen
popScreen	-	void	Return to previous screen of the current screen

popToRoot	-	void	Return to the Home screen of application
getItemFocus	-	QString	Return Item is focusing
setFocus	QString itemFocus	void	Set focus for Item from the Home screen
reloadScreen	-	void	Reload the current screen
focusChanged	-	void	Signal emit when item focus changed on screen

3.3 Class Song

Attribute:

Property	Type	Description
m_title	QString	Song name
m_singer	QString	Autho of the song
m_source	QString	source of song (.mp3)
m_albumArt	QString	Cover art of the song

Method:

Method	Parameter	Type	Description
Song	const QString &title, const QString &singer, const QString &source, const QString &albumArt	Constructors	-
tittle	-	QString	Get song name
singer	-	QString	Get author of the song
source	-	QString	Get the source of the song
Album_art	-	QString	Get cover art of the song

3.4 Class MediaModel

Attribute:

Property	Type	Description
m_data	QList<Song>	List used to storage the all Object of Song class
TittleRole	enum	Return song name top use on QML
SingleRole	enum	Return Author of the song to use on QML
SourceRole	enum	Return source of the song to use on QML
AlbumArtRole	enum	Return cover art of the song to use on QML

Method:

Method	Parameter	Type	Description
MediaModel	-	Constructors	-
rowCount	const QModelIndex &parent	Int	Get all the object on model
data	const QModelIndex &index, int role	Qvariant	Get data of model
addSong	Song &song	void	Add object of Song class to model
roleNames	-	QHash<int, QByteArray>	Define name of all song's attribute, to use on QML

3.5 Class MediaController

Attribute:

Property	Type	Description
m_player	QMediaPlayer *	The QMediaPlayer pointer, used to controls the song action
m_playlist	QMediaPlaylist *	The QMediaPlaylist pointer. Used to storage all song.

m_playlistModel	MediaModel	Model of the media
m_duarationTotal	qint64	Duaration of the song
m_totalTime	QString	Duaration of the song
m_currentTime	QString	The current time of the current song is playing
m_currentDuaration	qint64	The current time of the current song is playing
m_initVolume	qint32	The value init volume of application
m_songName	QString	The song name
m_singer	QString	Author of the song
m_volume	qint32	Current volume of application
m_instance	MediaController *	The pointer root of the class, use singleton pattern
m_engine	QQmlApplicationEngine *	Qml engine used to set context of c++ object , using on QML.

Method:

Method	**Parameter**	**Type**	**Description**
MediaController	-	Constructors	-
~MediaCOntroller	-	Destructors	-
getInstance	-	MediaController *	return to the pointer root of the class, use singleton pattern
initialize	QQmlContext *context	void	Set context of c++ object to use on QML
addToPlaylist	const QList<QUrl> &url	void	Add all data of the song to model and playlist
open	-	void	Open path of system, get source of the song

getAlbumArt	QUrl url	QString	Return cover art of the song
initPlaylistMediaPlayer	-	QString	Set playlist to QMedailPlayer, and set volume init for application
getMediaPlaylist	-	QMediaPlaylist *	Return Playlist of the song
getMediaPlayer	-	QMediaPlayer *	Return media player of the song
getModel	-	MediaModel *	Return model of the song
getTotalTime	-	QString	Return duration of the song by string type
getCurrentTime	-	QString	Return current time of the song is playing, by string type
getDuarationTotal	-	qint64	Return duration of the song by qint64 type
getCurrentDuaration	-	qint64	Return current time of the song is playing, by qint64 type
getSongName	-	QString	Return name of the song by string type
getSinger	-	QString	Return author name of the song by string type
continuePlayer	-	void	Continue play when the song is pausing
initPlayer	-	void	Play the song when the state is stop
pausePlayer	-	void	Pause the song is playing

setPosition	qint64 position	void	Set current Position of task bar
setCurrentIndex	int index	void	Set current index of the playlist
checkPlayerState	-	bool	Check the state of play mode
setVolume	qint32 volume	void	Set volume of application by user
getVolume	-	qint32	Get volume of the application
next	-	void	Next the next song on list
previous	-	void	Previous the previos song on list
switchModePlaylist	int mode	void	Change play mode bug user
onMetaDataAvailableChanged	bool available	void	Slot check data available
onDurationChanged	qint64 duration	void	Slot get duration when change index of the song on playlist
onPositionChanged	qint64 position	void	Slot get position of the song is playing
duarationChanged	-	void	Signal emit when index of the song changed
currentTimeChanged	-	void	Signal emit when current time of the song changed
currentIndexChanged	-	void	Signal emit when change index of the song on playlist

volumeChanged	-	void	Signal when change volume value of application

3.6 Class TranslatorModel

Attribute:

Property	Type	Description
m_translator	QTranslator *	A pointer of class QTranslator , used to controll language
m_enMode	bool	Check if English language mode
m_vnModel	bool	Check if Vietnamese language model
m_koModel	bool	Check if Korea language mode
EN	enum	Enum of English language mode
VN	enum	Enum of Vietnamese language mode
Ko	enum	Enum of Korea language mode

Method:

Method	Parameter	Type	Description
TranslatorModel	-	Constructors	-
~TranslatorModel	-	Destructors	-

emptyString	-	QString	Emit signal change language to QML
setLanguage	int lang	void	Set language mode by user from QML
checkModelLang	int lang	bool	Return current language be used

3.7 Class TranslatorController

Attribute:

Property	Type	Description
m_instance	TranslatorController *	The pointer root of the class, use singleton pattern
m_initialized	bool	Flag check if the root pointer initialized
m_model	TranslatorModel	Mode of language use on QML

Method:

Method	Parameter	Type	Description
TranslatorController	-	Constructor	-
~TranslatorController	-	Destructor	-
getInstance	-	TranslatorController *	return to the pointer root of the class, use singleton pattern
initialize	QQmlContext *context	void	Set context of c++ object to use on QML

3.8 Class ApplicationItem

Attribute:

Property	Type	Description
m_title	QString	Title of application
m_url	QString	url of application
m_iconPath	QString	Image icon of application

Method:

Method	Parameter	Type	Description
ApplicationItem	QString title, QString url, QString iconPath	Constructors	-
title	-	QString	Get title of the application
url	-	QString	Get url of the application
iconPath	-	QString	Get imgage icon of the application

3.9 Class ApplicationModel

Attribute:

Property	Type	Descritption
m_data	QList<AppliactionItem>	Storage the list of applications
TitleRole	enum	Title of the application use on QML

UrlRole	enum	Url of the application use on QML
IconPathRole	enum	Image icon of the application use on QML

Method:

Method	Parameter	Type	Description
ApplicationModel	-	Constructors	-
rowCount	const QModelIndex &parent	int	Return size of the list applications
data	const QModelIndex &index, int role	QVariant	Get data of the model of applications
addApplication	ApplicationItem &item	void	Insert application to model
moveItem	int from, int to	void	Used to change index of the list applications when user reoder application on home screen
getListApp		QList<ApplicationItem>	Return the list applications
roleNames		QHash<int , QByteArray>	Define name of all application's attribute, to use on QML

3.10 Class ApplicationController

Attribute:

Property	Type	Description
m_instance	ApplicationController *	The pointer root of the class, use singleton pattern

m_initialize	bool	Flag check if the root pointer initialized
m_model	ApplicationModel	Object of ApplicationModel class
m_xmlReader	XmlReader	Object of XmlReder class

Method:

Method	Parameter	Type	Description
ApplicationController	-	Constructors	-
~ApplicationController	-	Destructor	-
getInstance	-	ApplicationController *	Return to the pointer root of the class, use singleton pattern
initialize	`QQmlContext *context`	void	Set context of c++ object to use on QML
loadDataFromLocal	-	void	Read data from applications.xml file and paser data to application model
updateDataFromQML	`int from, int to`	void	Update data of applications.xml file when user reoder the applications on the home screen

3.11 Class XmlReader

Attribute:

Property	Type	Description
m_xmlDoc	QDomDocument	Read xml data

Method:

Method	Parameter	Type	Description
xmlReader	-	Contructors	-

ReadXmlFile	Qstring filePath	bool	Read file applications.xml , when the application be loaded the first and update by reorder the application
ParserXml	ApplicationsModel &model	void	Paser file applications.xml when the application be loaded the first
XmlUpdateData	QList<ApplicationItem> list	void	Write data to applications.xml when user reodered the applications on home screen

3.12 Class ClimateModel

Attribute:

Property	Type	Description
m_climate	local::Climate *	Name space of the climate simulator application, used get data from the the climate simulator application by dbus protocol

Method:

Method	Parameter	Type	Description
ClimateModel	-	Constructors	-
GetDriverTemperature	-	double	Return temperature of the driver position from climate simulator application
GetPassengerTemperature	-	double	Return temperature of the passenger position from climate simulator application
GetFanLevel	-	int	Return Fan level from climate simulator application
GetDriverWindModel	-	int	Return wind direct of the driver from climate simulator application
GetPassengerWindModel	-	int	Return wind direct of the passenger from climate simulator application
GetAutoModel	-	int	Return the state of the AUTO mode from climate simulator application

GetSyncModel	-	int	Return the state of the SYNC mode from climate simulator application
dataChanged	-	void	Signal emit when data be changed

3.13 Class ClimateController

Attribute:

Property	Type	Description
m_instance	ClimateController *	The pointer root of the class, use singleton pattern
m_initialize	bool	Flag check if the root pointer initialized
m_model	ClimateModel	Model of climate widget

Method:

Method	Parameter	Type	Description
ClimateController	-	Constructors	-
~ClimateController	-	Destructors	-
getInstance	-	ClimateController *	return to the pointer root of the class, use singleton pattern
initialize	QQmlContext *context	void	Set context of c++ object to use on QML

3.14 Class AppEngine

Attribute:

Property	Type	Description
m_instance	AppEngine *	The pointer root of the class, use singleton pattern

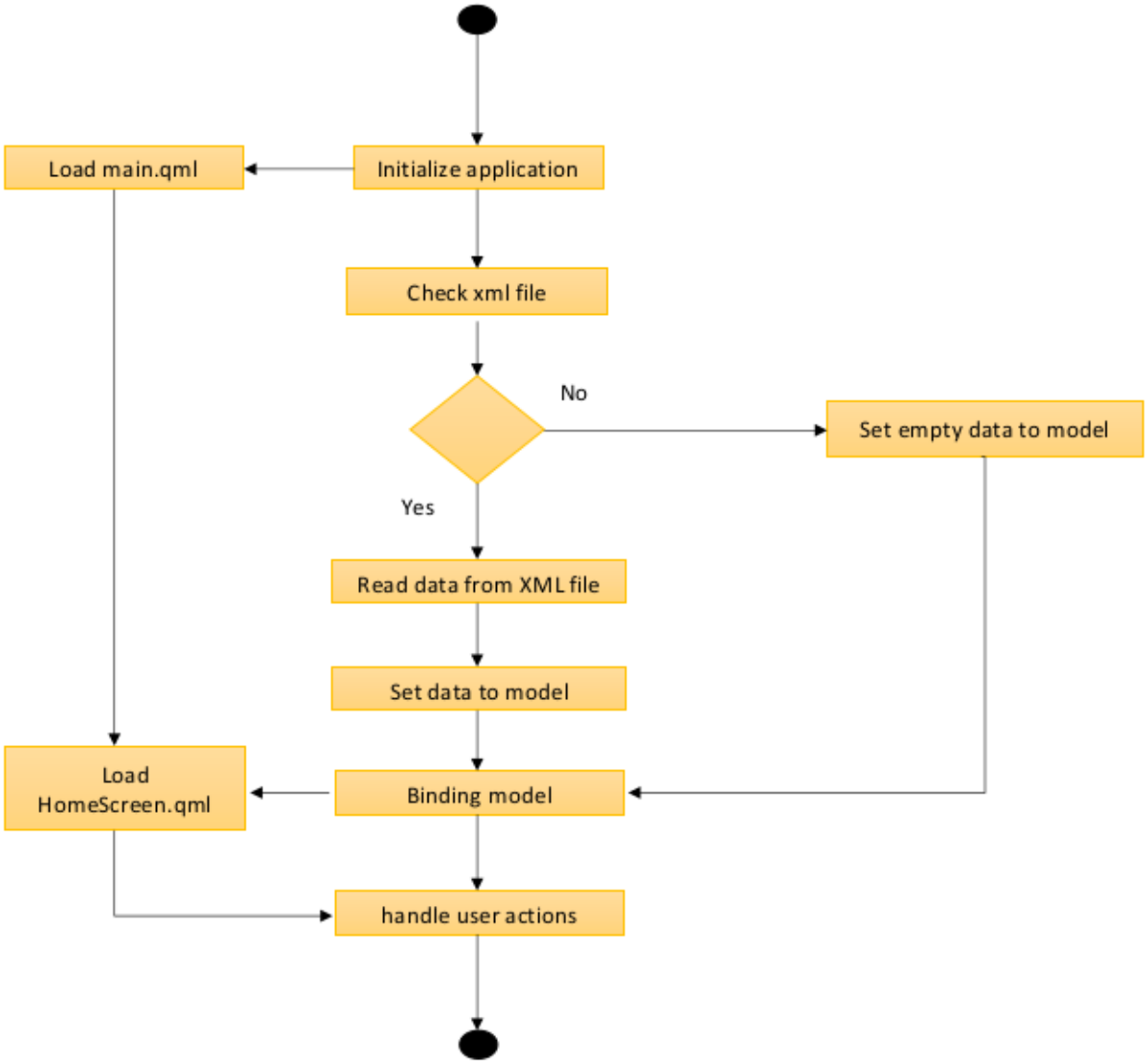
m_initialized	bool	Flag check if the root pointer initialized
m_app	QGuiApplication *	GUI engine of the Application
m_engine	QQmlApplicationEngine	Qml engine used to set context of c++ object , using on QML.

Method:

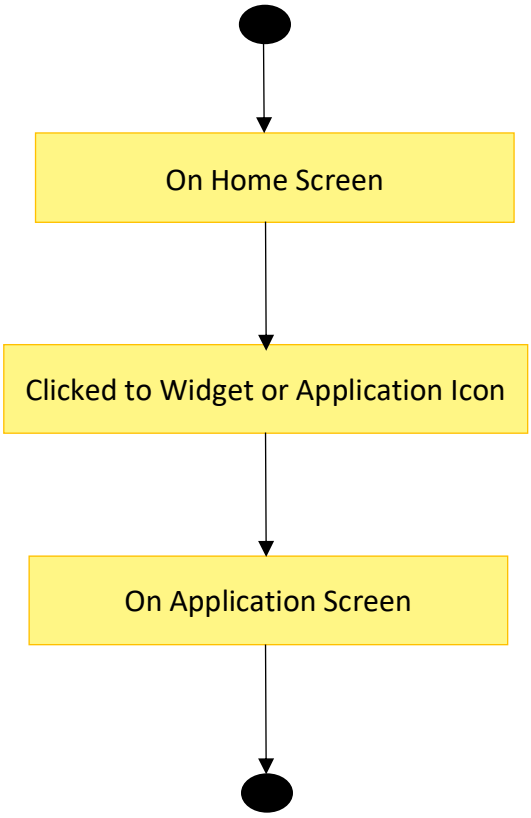
Method	Parameter	Type	Description
AppEngine	-	Constructors	-
~AppEngine	-	Destructors	-
getInstance	-	AppEngine	return to the pointer root of the class, use singleton pattern
initialize	`QGuiApplication *app`	Void	Init home application
registerQmlObjects	-	void	Register an object usd on QML
createControllers	-	void	Create All controllers of the application
initControllers	-	void	Init context property for all controller of the application
initScreens	-	void	Init the home screen, when application loaded

3:Flow chart design

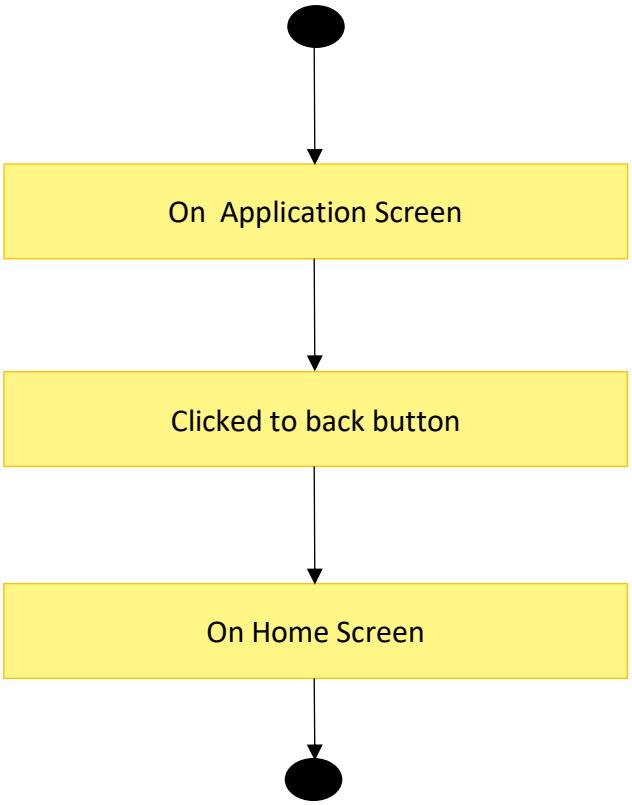
A. Initialze home application



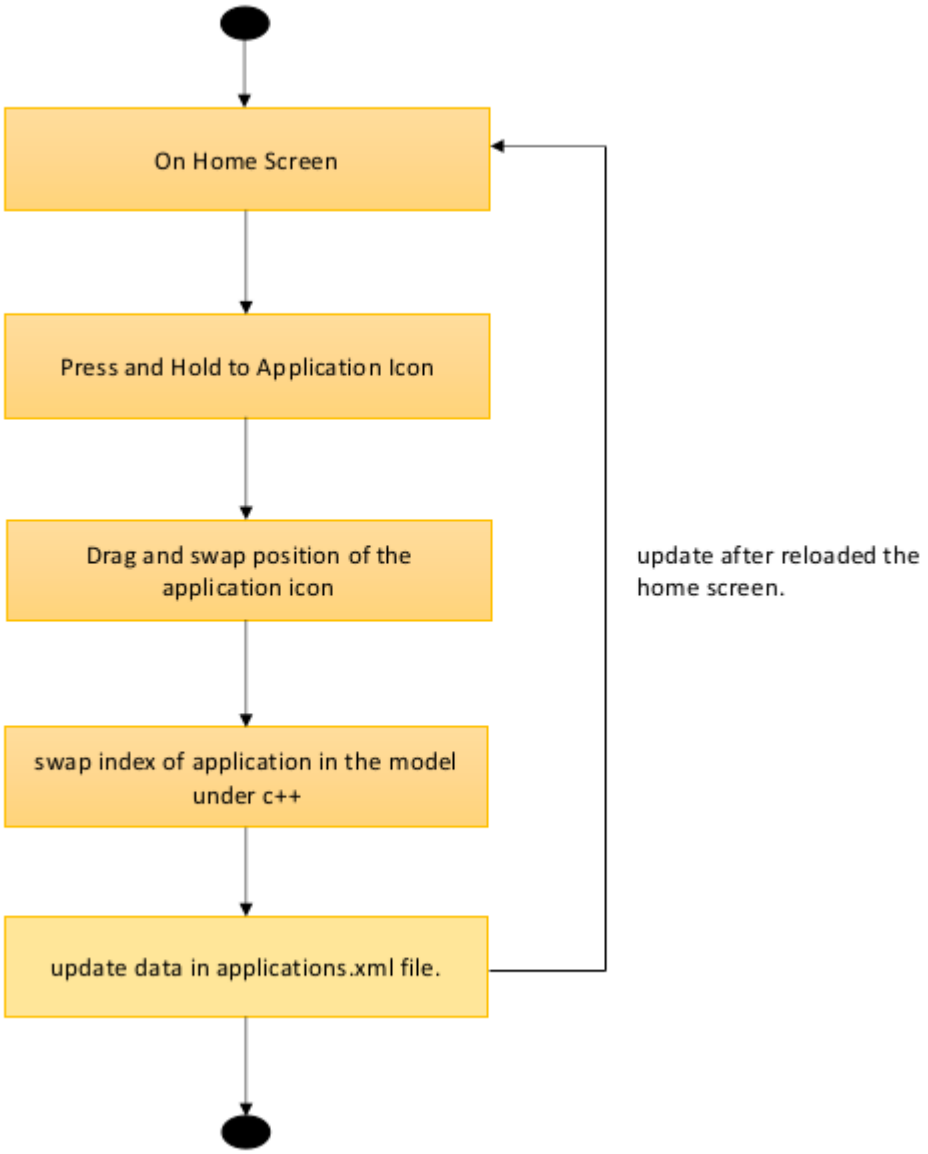
B. Open application



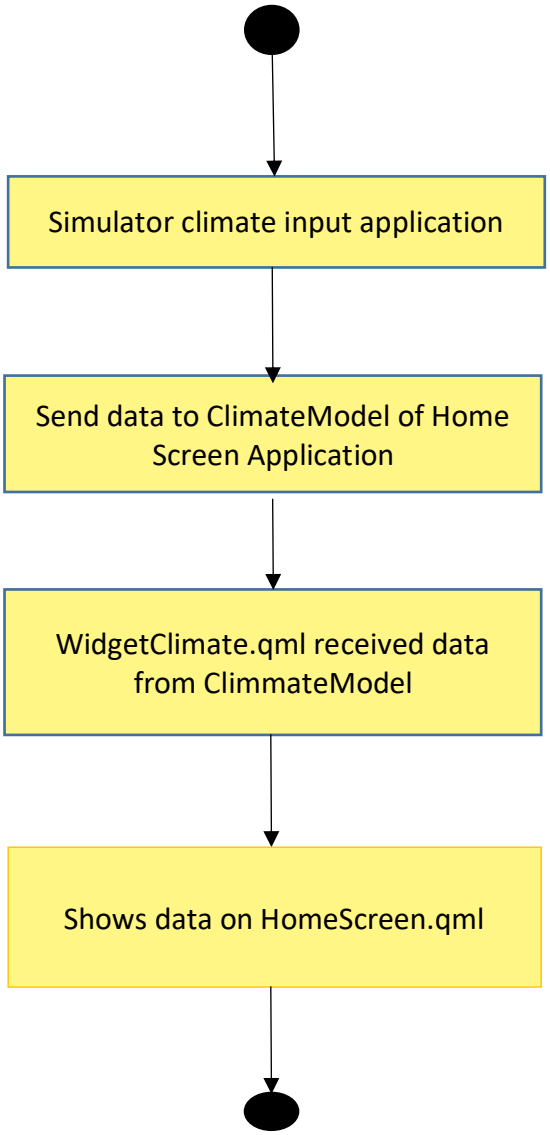
C. Close Application



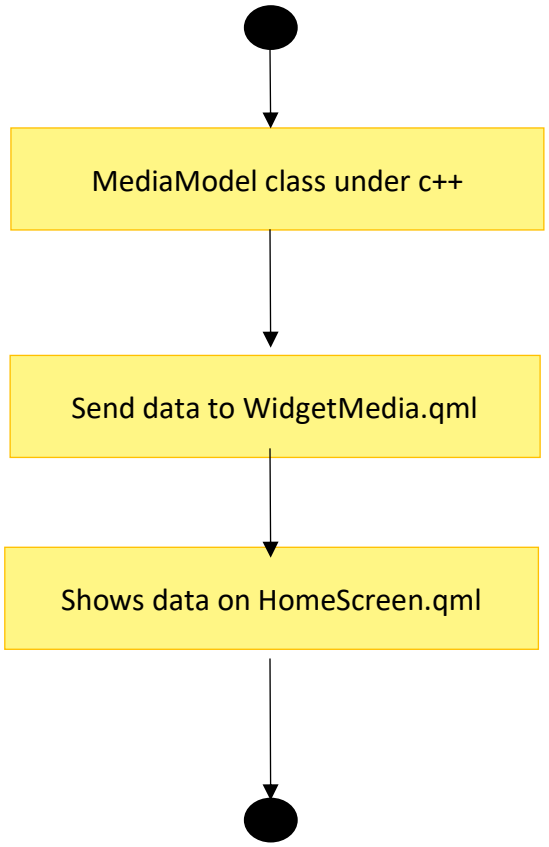
D.Reorder application



E.Send data to Climate Widget



F.Send data to Music Widget



G. Scollbar visible