Software Architecture Design

For project

Home

Version 1.3

Prepared by Tran Ngo Anh Tuan

HoChiMinh 02/12/2022

Document History

Version	Editor	Reviewer	Date	Page	Description
1.0	Tran Ngo Anh Tuan	Pham Ba Thanh	26/11/2022	All	Create
1.1	Tran Ngo Anh Tuan	Pham Ba Thanh	28/11/2022		Translate text, redesign some flow diagrams
1.2	Tran Ngo Anh Tuan	Pham Ba Thah	30/11/2022		Fix flow diagrams

Table of Contents

I. Interface design

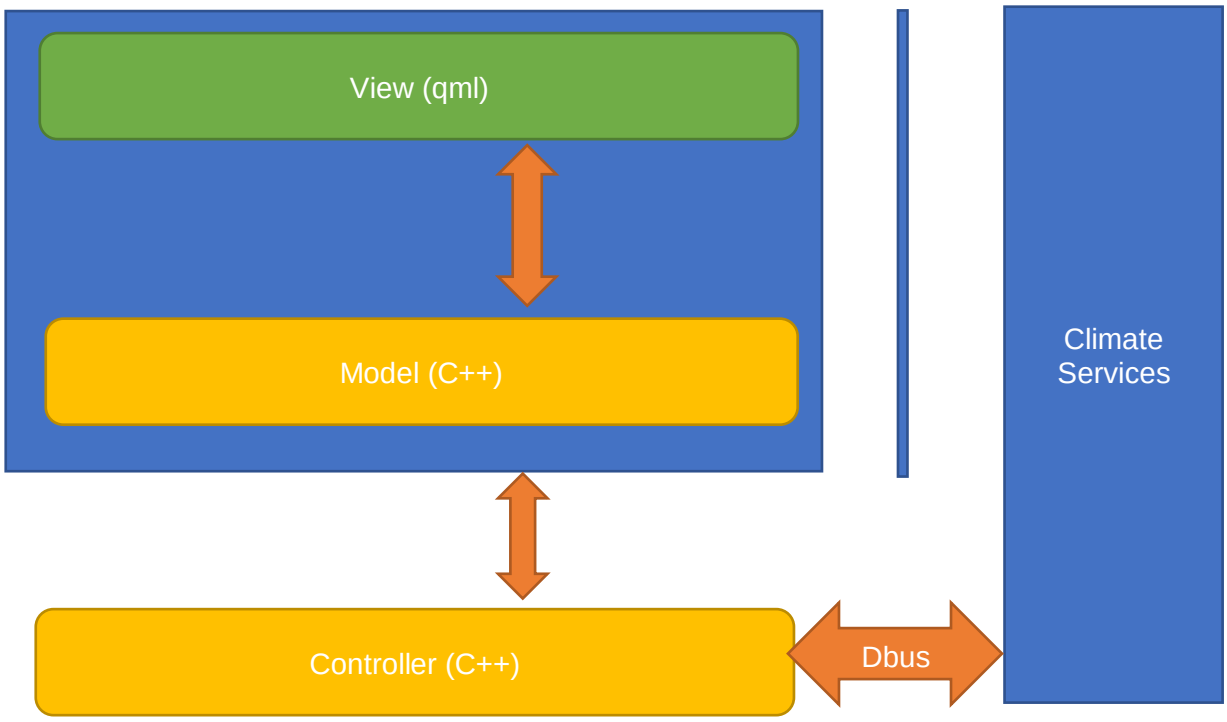
Presented in UI Sample .pdf document

II. Interactive design

Presented in UX Sample.pdf document

Architectural design

1. App overall diagram

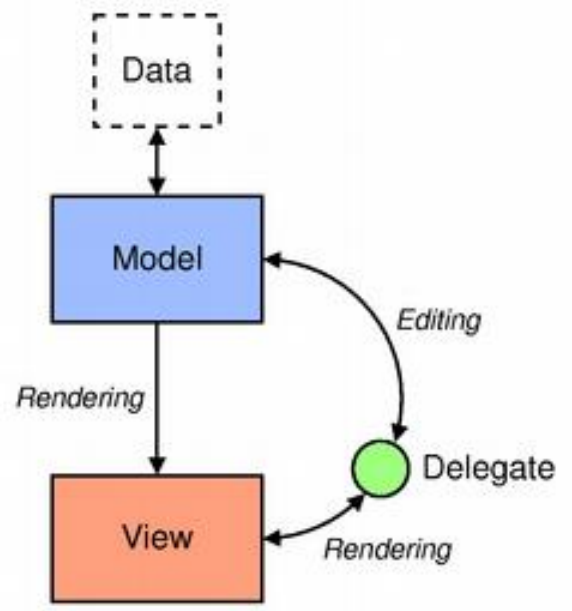


View (qml): This is where monitors, components are built using qml, and the resources of screen building

Model: As a place to build data for managing the state of the interface from C++, it is the place to show the data for building the state of the screen

Controller: Is the part that handles, controls the program, and is responsible for connecting to third services (specifically, climate sevices)

Construction architecture for the project is built based on Model View architecture

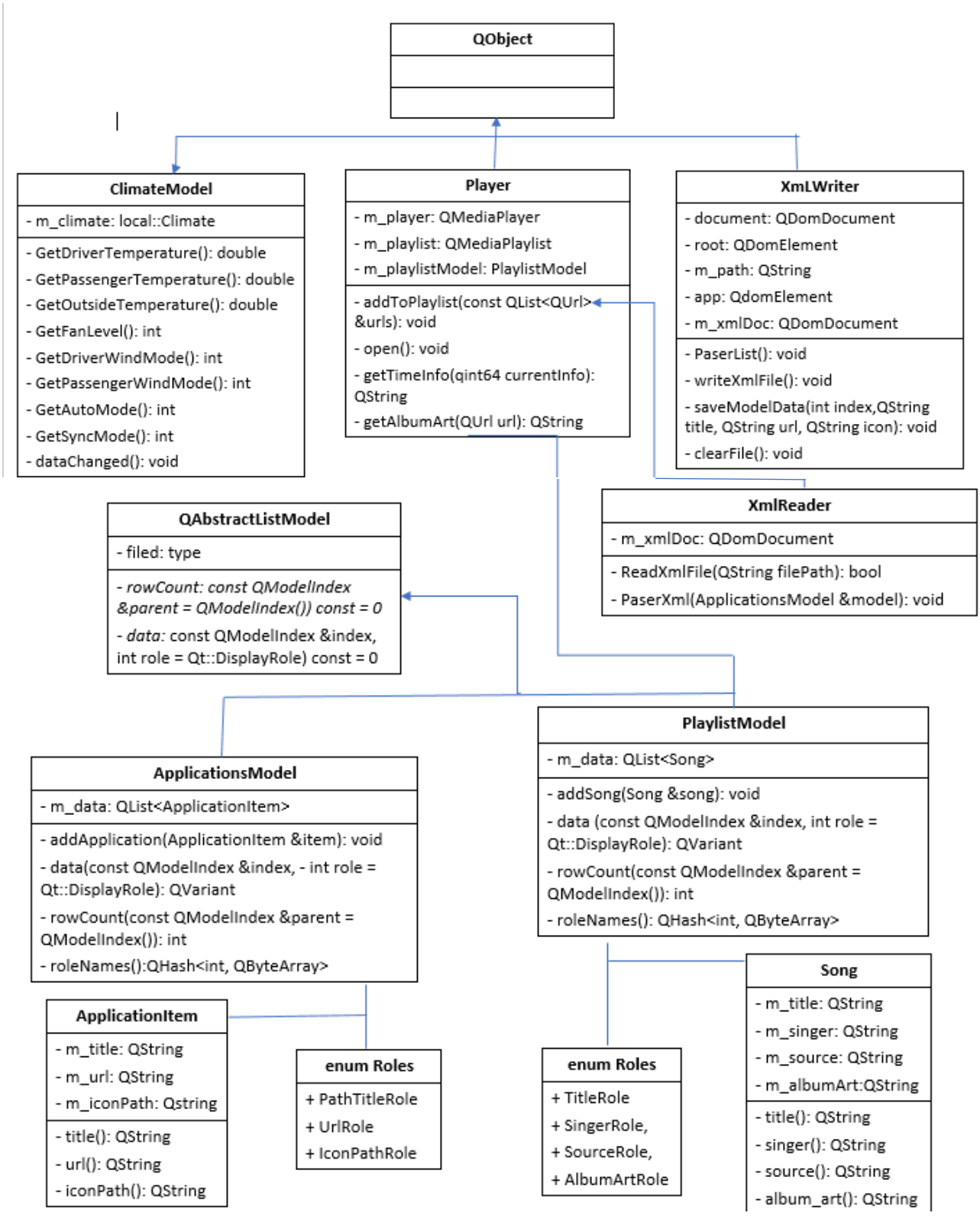


Data: xml contains information about applications in the system

Model: Class stores list of applications read from xml file

View: QML displays application list

III. Class Diagram



1. Class PlaylistModel

- **Attributes**

Property	**Type**	**Description**
m_data	QList<Song>	Store metadata of song

- **Method**

Function	**Description**	**Input**	**Output**
addSong	Add song to model	A variable typed Song	
data	To map data in model	Two variables typed QModelIdenx and int	QVariant
rowCount	Return number of element (song)		Int
roleNames	Define roles for enums		roles

2. Class ApplicationModel

- **Attributes**

Property	**Type**	**Description**
m_data	QList<ApplicationItem>	Store metadata of an application

- **Method**

Function	**Description**	**Input**	**Output**
addApplication	Add app to model	A variable typed Song	
data	To map data in model	Two variables typed QModelIdenx and int	QVariant
rowCount	Return number of element (ApplicationItem)		Int
roleNames	Define roles for enums		roles
getModel	Get ApplicationItem from model		ApplicationItem
writeData	Write data into file xml	QString	

3. Class ApplicationItem

- **Attributes**

Property	**Type**	**Description**
m_title	QString	Store title of app
m_url	QString	Store url of app
m_iconPath	QString	Store iconPath of app

- **Method**

Function	**Description**	**Input**	**Output**
title	Return tile of app		QString
url	Return url of app		QString
iconPath	Return path of icon app		QString

4. Class Song

- **Attribute**

Property	Type	Description
m_title	QString	Store title of app
m_singer	QString	Store singer name of app
m_source	QString	Store directory of song

m_albumArt	QString	Store directory of album image

- **Method**

Function	Description	Input	Output
title	Return tile of app		QString
singer	Return url of app		QString
Source	Return directory of song		QString
album_art	Return directory of album image		QString

5. Class XmlReader

- **Attribute**

Property	Type	Description
m_xmlDoc	QDomDocument	Store data of app

- **Method**

Function	**Description**	**Input**	**Output**
ReadXmlFile	Check if file can be read	Path of file	bool
PaserXml	Extract data and add into model	ApplicationModel	

	(ApplicationModel)		

6. Class XmlWriter

- **Attribute**

Property	Type	Description
document	QDomDocument	Store data of app
root	QDomElement	createElement("APPLICATIONS")
m_path	QString	path
app	QdomElement	document.createElement("APP")

- **Method**

Function	Description	Input	Output
- PaserList()			
- writeXmlFile()	Write file xml	ApplicationModel	string
- saveModelData(int index,QString title, QString url, QString icon)	Save data to document		
- clearFile(): void	Delete data in document		

7. Class Player

- **Attribute**

Property	Type	Description
m_player	QMediaPlayer	Media controller
m_playlist	QMediaPlaylist	Song list manager
m_playlistModel	PlaylistModel	Data model

- **Method**

Function	Description	Input	Output
addToPlaylist	Add song into Playlist	QList<QUrl>	
open	Extract url		
getTimeInfo	Return the time process of time	qint64	QString
getAlbumArt	Extract data to get album picture from song	QUrl	QString

8. Class ClimateModel

- **Attribute**

Property	Type	Description
m_climate	local::Climate	Store data of climate app

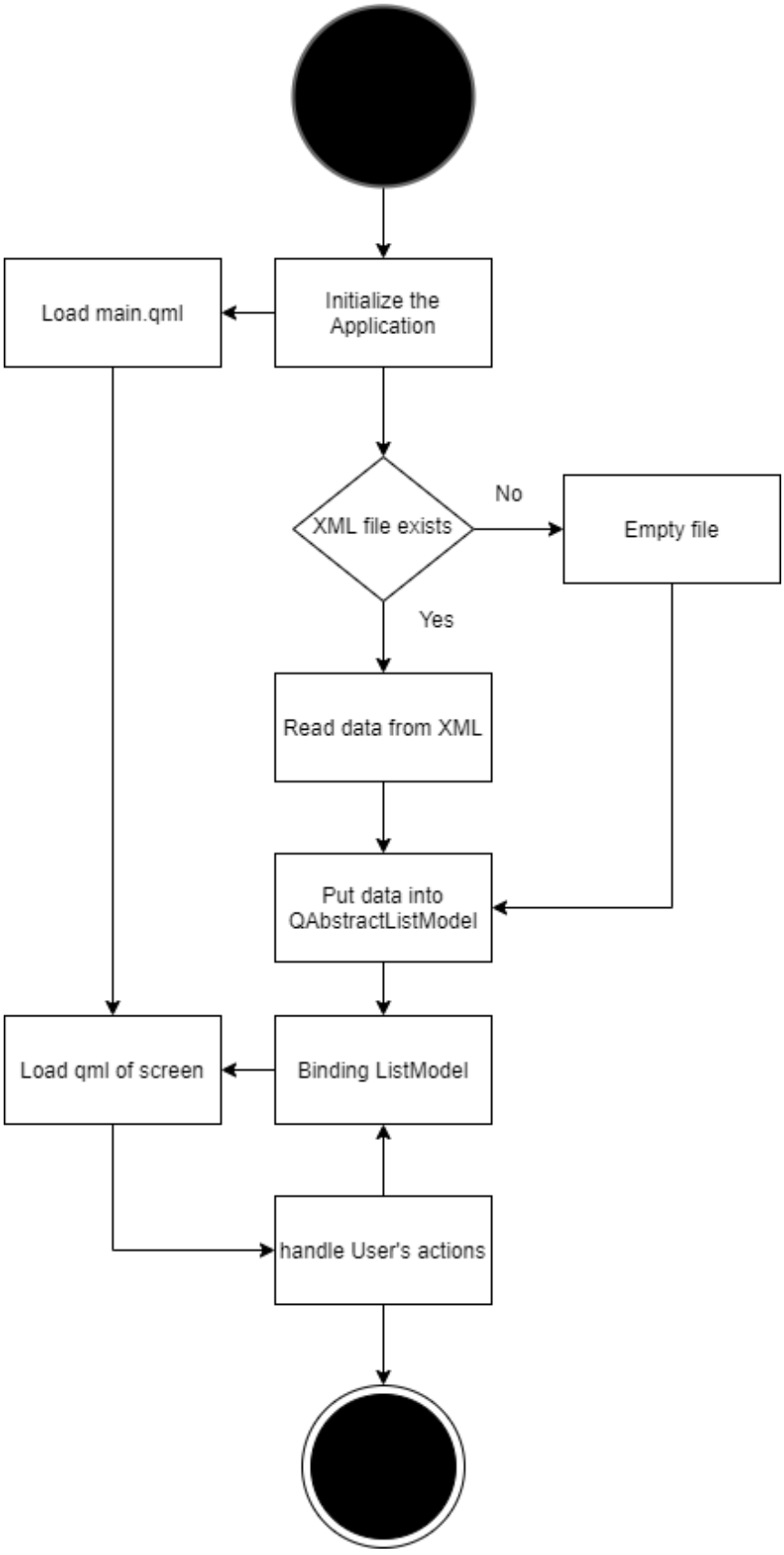
- **Method**

Function	Description	Input	Output
GetDriverTemperature	Return temperature of driver		double
GetPassengerTemperature	Return temperature of passenger		double
GetOutsideTemperature	Return temperature of outside		double
GetFanLevel	Return speed of fan		Int
GetDriverWindMode	Turn direction of win at driver seat		int
GetPassengerWindMode	Turn direction of win at		int

	passenger seat		
GetAutoMode	Turn ON/OFF of AUTO mode		int
GetSyncMode	Turn ON/OFF of SYNC mode		int

IV. Work flow design

1. Processing flow when starting Home app



Steps to start the home program:

Step 1: Create the engine object of QQmlApplicationEngine

Step 2: Create appsModel object of ApplicationsModel

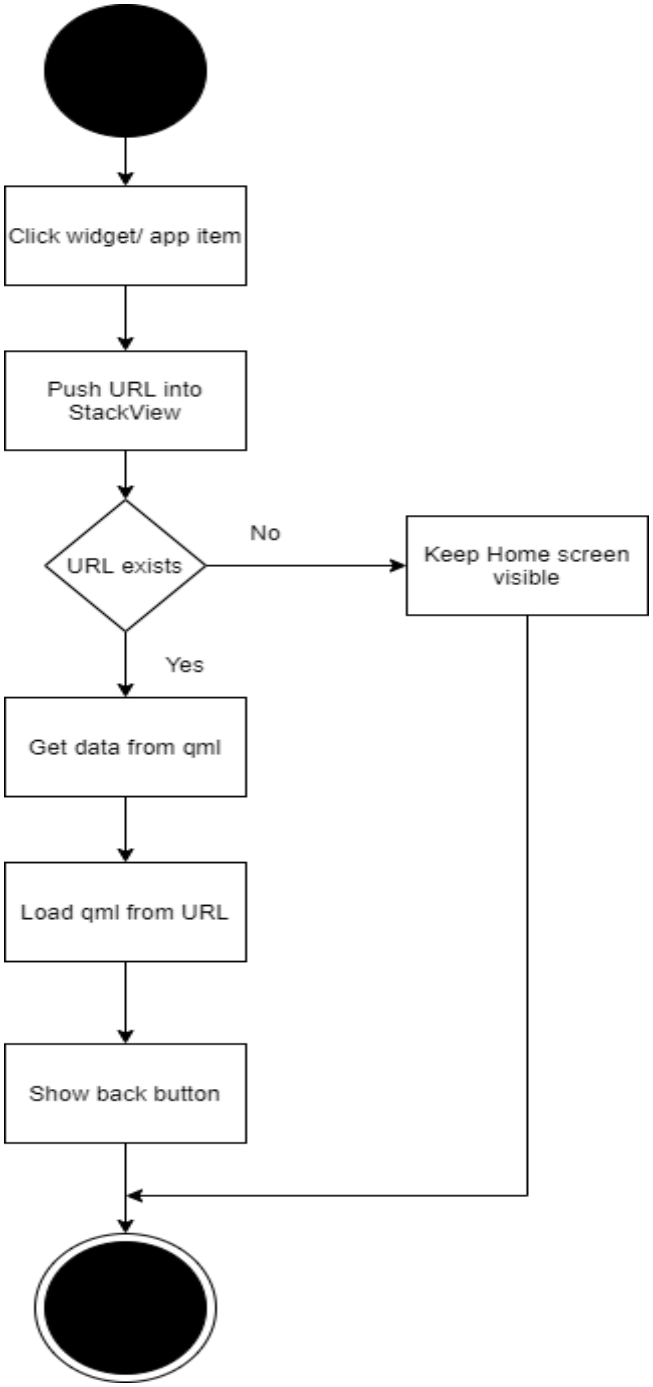
Step 3.4: Create xmlReader object of xmlReader with the transfer value as the path to the xml file and appsModel object

Step 5: Read the xml file

Step 6: Paser information from xml to ApplicationsModel object

Step 7: Binding appsModel to QML by settingContextProperty **Step 8:** Start the QML engine by loading the url of the main qml file

2. Processing flow for Open app



Steps to open one app on widget:

Step 1: Click on the App or widget. Besides, users can use Shortcut and hard key to open widgets:

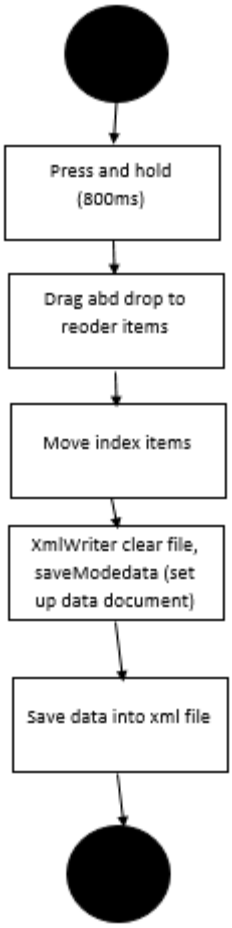
- Press number M key to open Map widget
- Press number C key to open Climate widget
- Press number P key to open Media widget
- Press Enter key to open any Focus app.

Step 2: Pass Url into StackView by function openApplication()

Step 3. Open file from StackView

Step 3: Show Back button (isShowBackBtn = true)

3. Processing flow for Reorder application menu



Steps to reorder app items:

Step 1: reorder item by drag and drop

Step 2: Move index of item

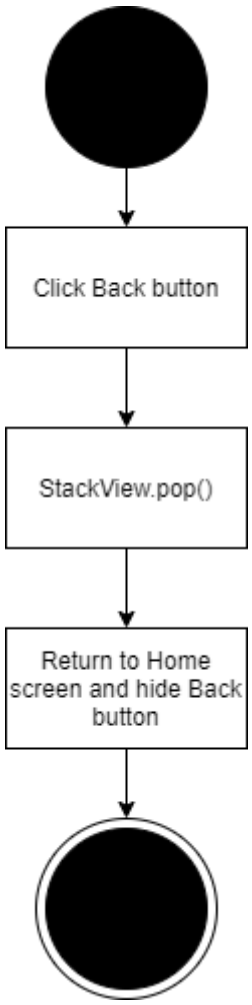
Step 3: move items

Step 4: clear data from xml file and save mode data in document

Step 5: open xml file and write data into it

4. Processing flow for Close App

.



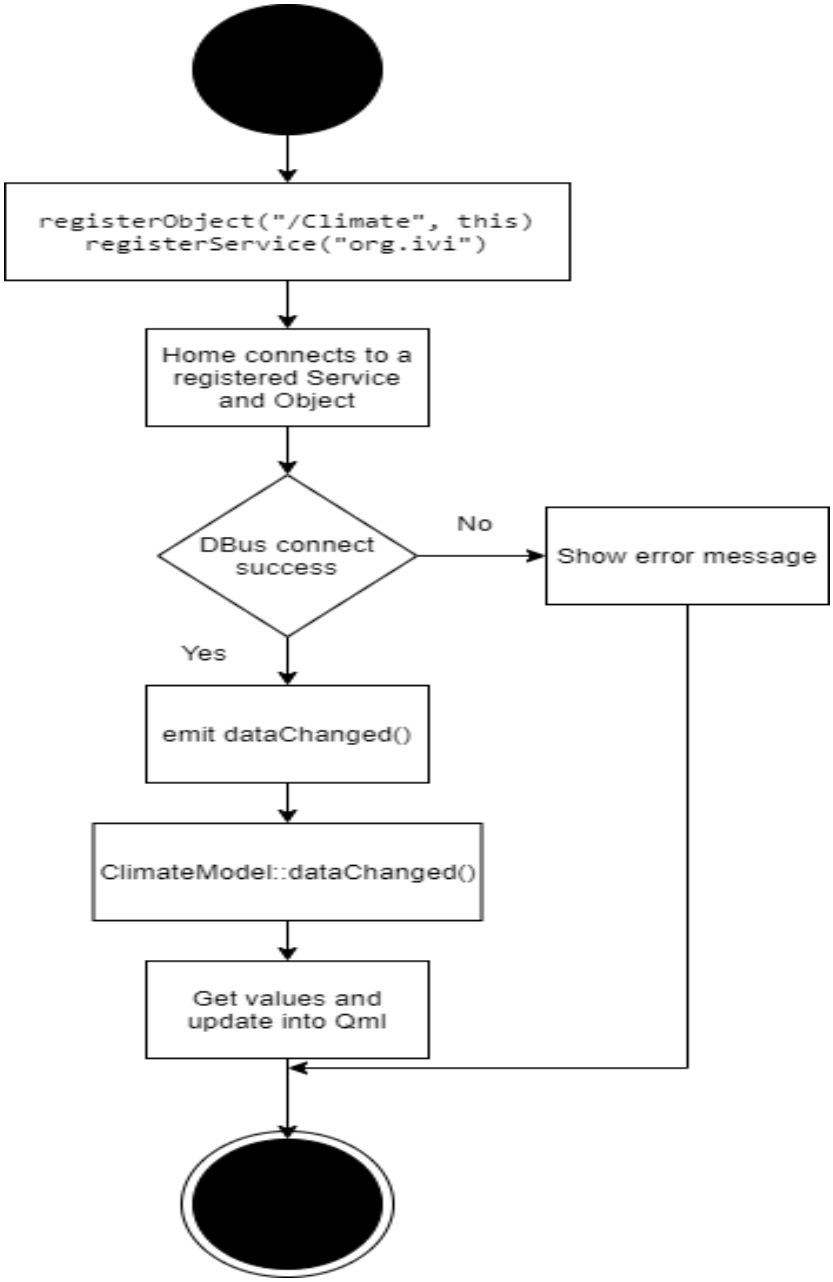
Steps to Close App:

Step 1: Click the Back button or Backspace key to return to the Home page.

Step 2: Use the pop () function to get the element in StackView.

Step 3: Display the Home screen again.

5. Processing flow for connecting Dbus for updating Climate app real-time:



Steps to send Climate:

Step 1: Simulator must register service of DBus and an Object via QDBusConnection

before sending the value to the receiver.

Step 2: Receiving App must connect to QDBusConnection through the registered service

Step 3: Get the value through connection DBus Step

Step 4: Set the value for receiving app