Software architecture design

For Project

Home

Version 1.0

Prepared by BienNT

TP.HCM, 22/04/2025

I. UI Desgin

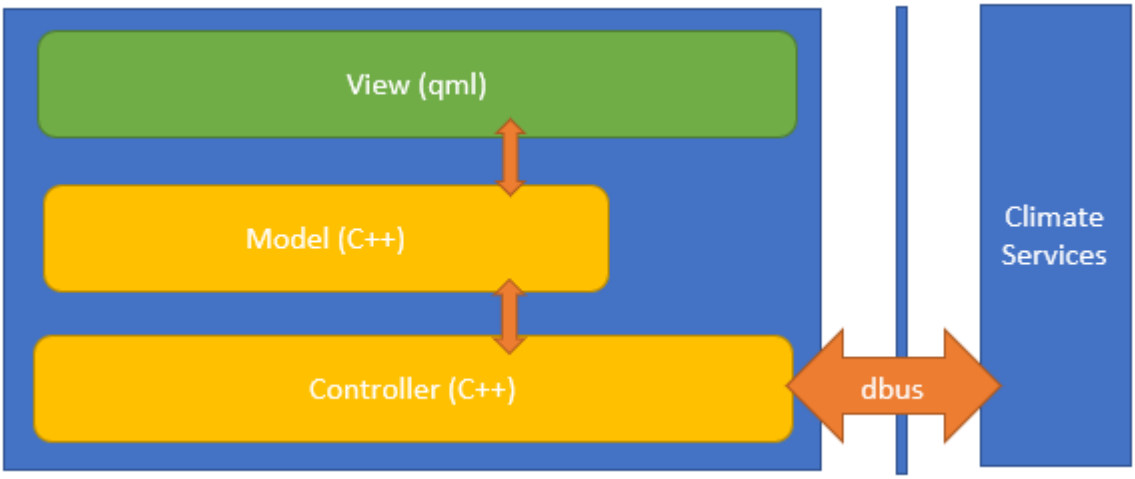
The UI is presented in the UI Sample.pdf document.

II. UX Design

The UX design is presented in the UX Sample.pdf document.

III. Architecture design

1. Application overview diagram



View (qml): This layer manages the screens, UI components built using QML, and resources used for rendering the UI.

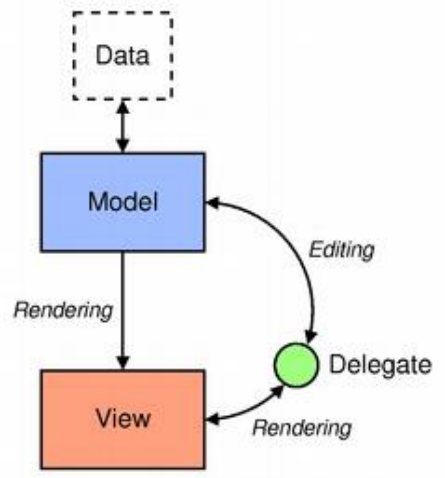
Model: This layer is responsible for managing the data that controls the state of the UI from C++. It represents the data used to build the UI state.

Controller: This layer handles the program logic and is responsible for connecting with third-party services (in this case, the climate services).

D-Bus**:** is an Inter-Process Communication (IPC) and Remote Procedure Call (RPC) mechanism that supports data exchange between multiple applications. Through a central bus, applications can send messages to each other or call remote methods easily. D-Bus acts as a communication bridge that allows the Controller to connect to external services (in this case, the climate services).

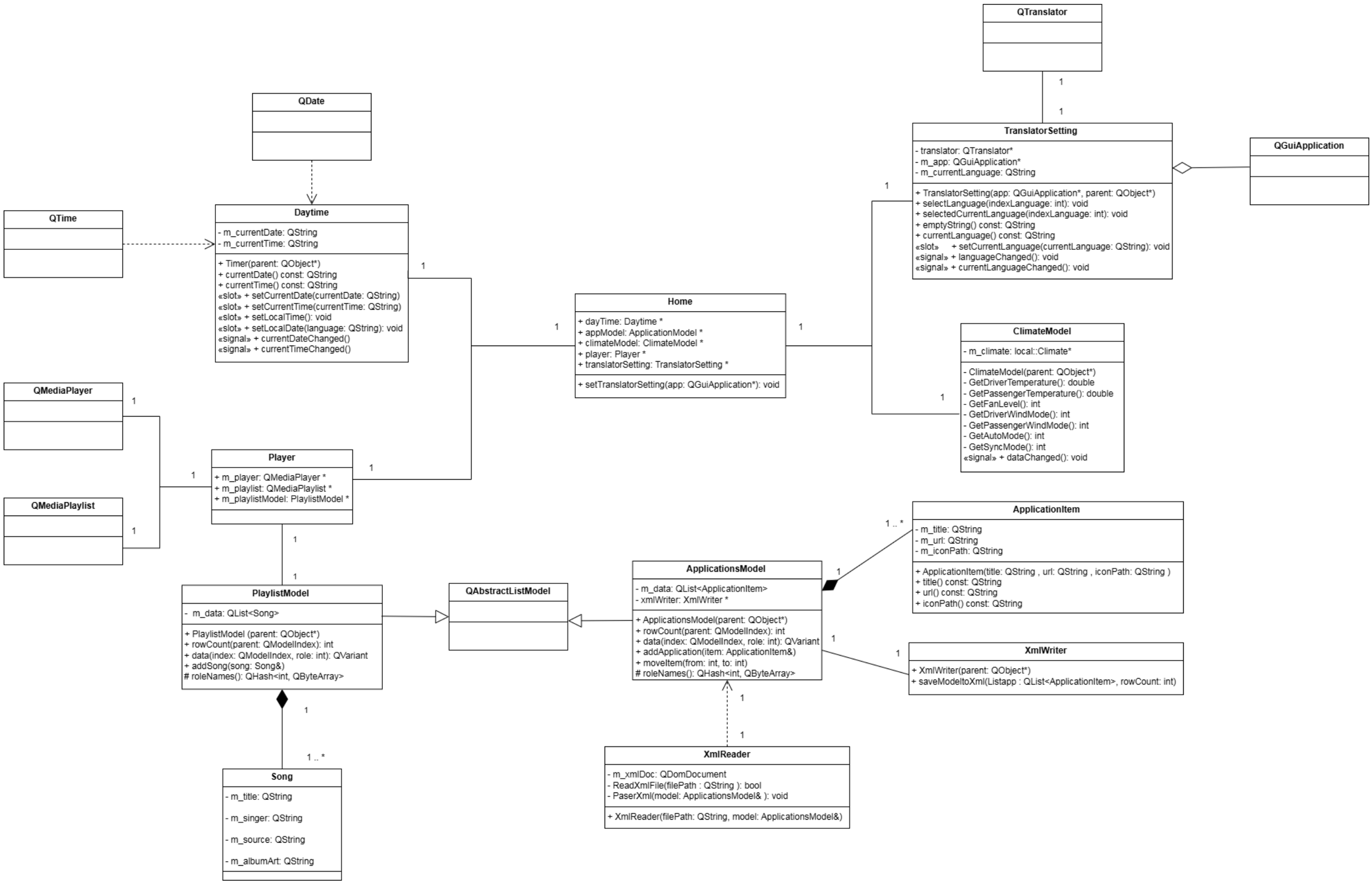
Climate services: provide A/C information for other applications. The data can be sent to D-Bus, and applications can retrieve it from D-Bus through their controllers.

The architecture of the program is built based on the **Model-View** architecture.



Data: An XML file contains information about the applications in the system.
Model: A class that stores the list of applications read from the XML file.
View: QML is used to display the list of applications.

IV. Class Diagram



4.1 ApplicationsModel

- Attribute

Property	Type	Description
m_data	QList<ApplicationItem>	The list contains application elements

- Method

Function	Description	Input	Output
ApplicationsModel()	Contructor function	QObject *parent	
rowCount()	Return the size of App List Model	QModelIndex &parent	int
data()	Get the data of each app from the m_data - app list model	QModelIndex &index int role	QVariant
addApplication()	Add a app into the m_data object	ApplicationItem &item	void
moveItem()	Move an app from preIndex to index to reorder the app list model	int from int to	Q_INVOKABLE void
roleNames()	Expose the app data to QML		QHash<int, QByteArray>

4.2 ApplicationItem

- Attribute

Property	Type	Description
m_title	QString	The title of application
m_url	QString	The url is used to open app
m_iconPath	QString	The iconPath specifies the path to the icon of the corresponding app

- Method

Function	Description	Input	Output
ApplicationItem()	Contructor function	QString title QString url QString iconPath	
title()	Return the app's title	QModelIndex &parent	QString
url()	Return the url title	QModelIndex &index int role	QString
iconPath()	Return the iconPath's title	ApplicationItem &item	QString

4.3 XmlReader

- Attribute

Property	Type	Description
m_xmlDoc	QDomDocument	This object stores the content of the XML file

- Method

Function	Description	Input	Output
XmlReader()	Contructor function	QString filePath ApplicationsModel &model	
ReadXmlFile()	Open XML file and set its content to m_xmlDoc object	QString filePath	bool
PaserXml()	Read app data from m_xmlDoc object and add the apps to the app list	ApplicationsModel &model	void

4.4 XmlWriter

- Method

Function	Description	Input	Output
XmlWriter()	Contructor function	QObject *parent	
saveModeltoXml()	Save the app model to an XML file after reordering the apps	QList<ApplicationItem> Listapp int rowCount	

4.5 ClimateModel

- Attribute

Property	Type	Description
m_climate	local::Climate*	Interacts with the DBus interface to get climate service data

- Method

Function	Description	Input	Output
GetDriverTemperature()	Get data from Dbus		double
GetPassengerTemperature()	Get data from Dbus		double
GetFanLevel()	Get data from Dbus		QString
GetDriverWindMode()	Get data from Dbus		int
GetPassengerWindMode()	Get data from Dbus		int
GetAutoMode()	Get data from Dbus		int
GetSyncMode()	Get data from Dbus		int
dataChanged();	This signal is connected to the dataChanged signal of the m_climate object from D-Bus interface		void

4.6 Timer

- Attribute

Property	Type	Description
m_currentDate	QString	Contains the current date of the system
m_currentTime	QString	Contains the current time of the system

- Method

Function	Description	Input	Output
Timer()	Contructor function	QObject *parent	
currentDate()	Return the value of m_currentDate object		QString
currentTime()	Return the value of m_currentTime object		QString
setCurrentDate()	Set the value for m_currentDate	QString currentDate	void
setCurrentTime()	Set the value for m_currentTime	QString currentTime	void
setLocalTime()	Retrieves the system's current local time from QTime, formats it as "hh:mm"		void
setLocalDate()	Sets the current Date based on the given language locale, formats it as "MMM. dd".	QString language	Q_INVOKABLE void
currentDateChanged()	Emit signal when the currentDate value changes		void
currentTimeChanged()	Emit signal when the currentTime value changes		void

4.7 TranslatorSetting

Attribute

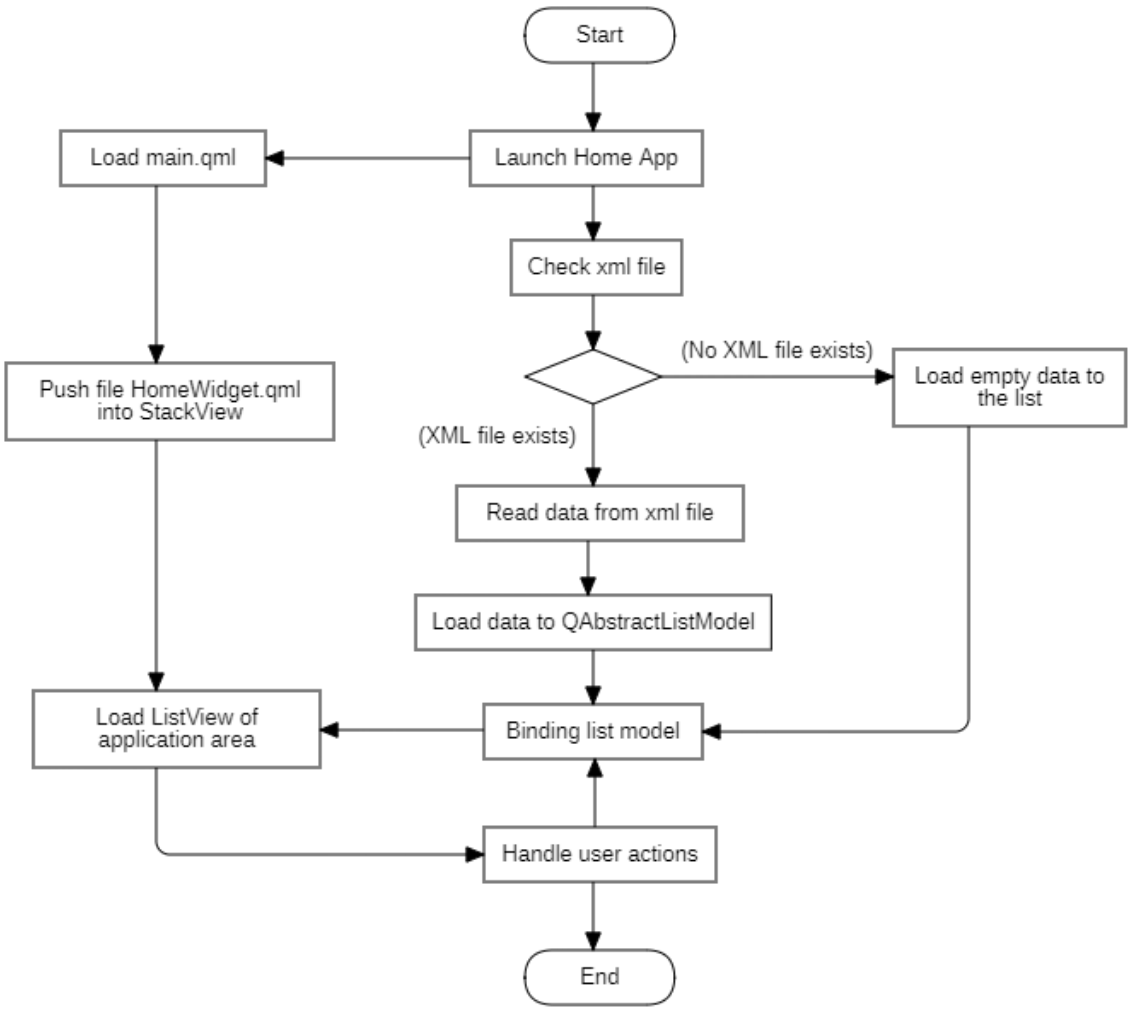
Property	Type	Description
*translator	QTranslator	Be used to load a .qm file
*m_app	QGuiApplication	Be used to install translator
m_currentLanguage	QString	Contains a selected language

- Method

Function	Description	Input	Output
TranslatorSetting()	Contructor function	QGuiApplication *app QObject *parent	
selectLanguage()	Set the selected language for Home App	int indexLanguage	Q_INVOKABLE void
selectedCurrentLanguage()	Install the translator with the selected language	int indexLanguage	void
emptyString()	Return the empty string value		QString
currentLanguage ()	Return the selected language value		QString
setCurrentLanguage ()	Set the value for m_currentLanguage	QString currentLanguage	void
languageChanged ()	Emits signal when the seleted language changes		void

V. Design the processing flow

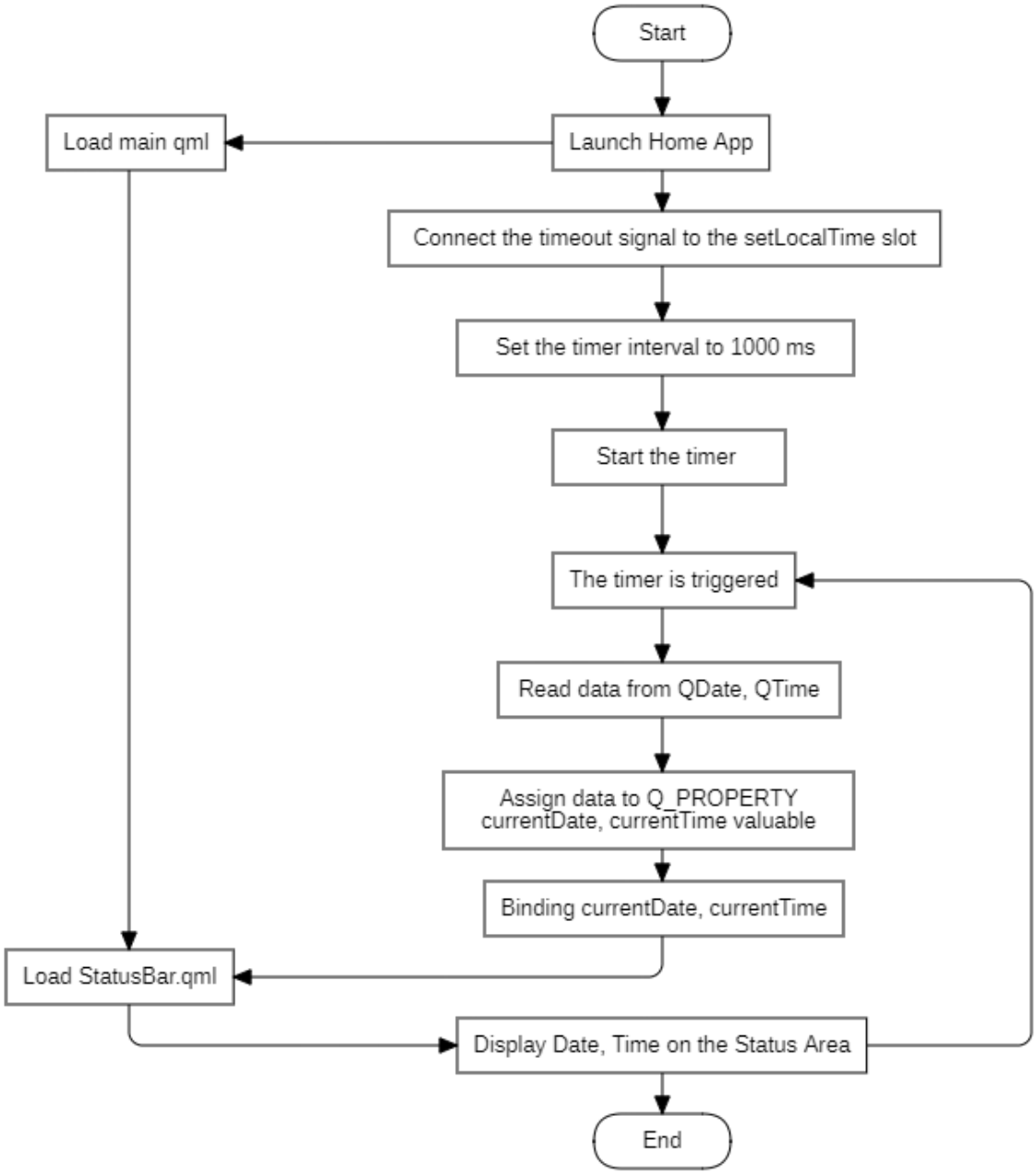
1. Starting Home Application



- Steps to start the Home Application:

Step 1: Create the engine object of QQmlApplicationEngine
Step 2: Create the appsModel object of ApplicationsModel
Step 3 & 4: Create the xmlReader object of XmlReader, passing in the path to the XML file and the appsModel object
Step 5: Read the XML file
Step 6: Parse data from the XML into the ApplicationsModel object
Step 7: Bind appsModel to QML using setContextProperty
Step 8: Launch the QML engine by loading the main.qml file of URL

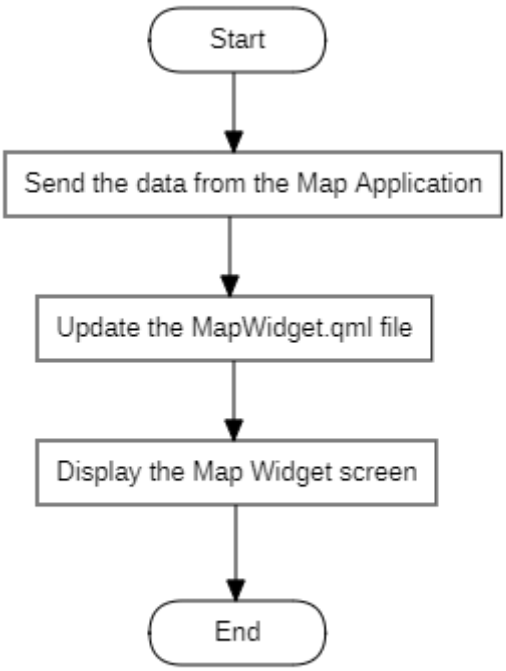
2. Updating System Time



- Steps to update system time:

Step 1: Create dateTime object of Timer
Step 2: Create timer object or QTimer
Step 3: Connect timeout signal of QTimer to setLocalTime slot
Step 5 & 6: Set the interval of timer and start timer
Step 7: Check whether the timer is triggered
Step 8: Get the data into dateTime object
Step 9: Binding dateTime to QML using setContextProperty
Step 10: Display the Date and Time on the system
Step 11: Back to step 7

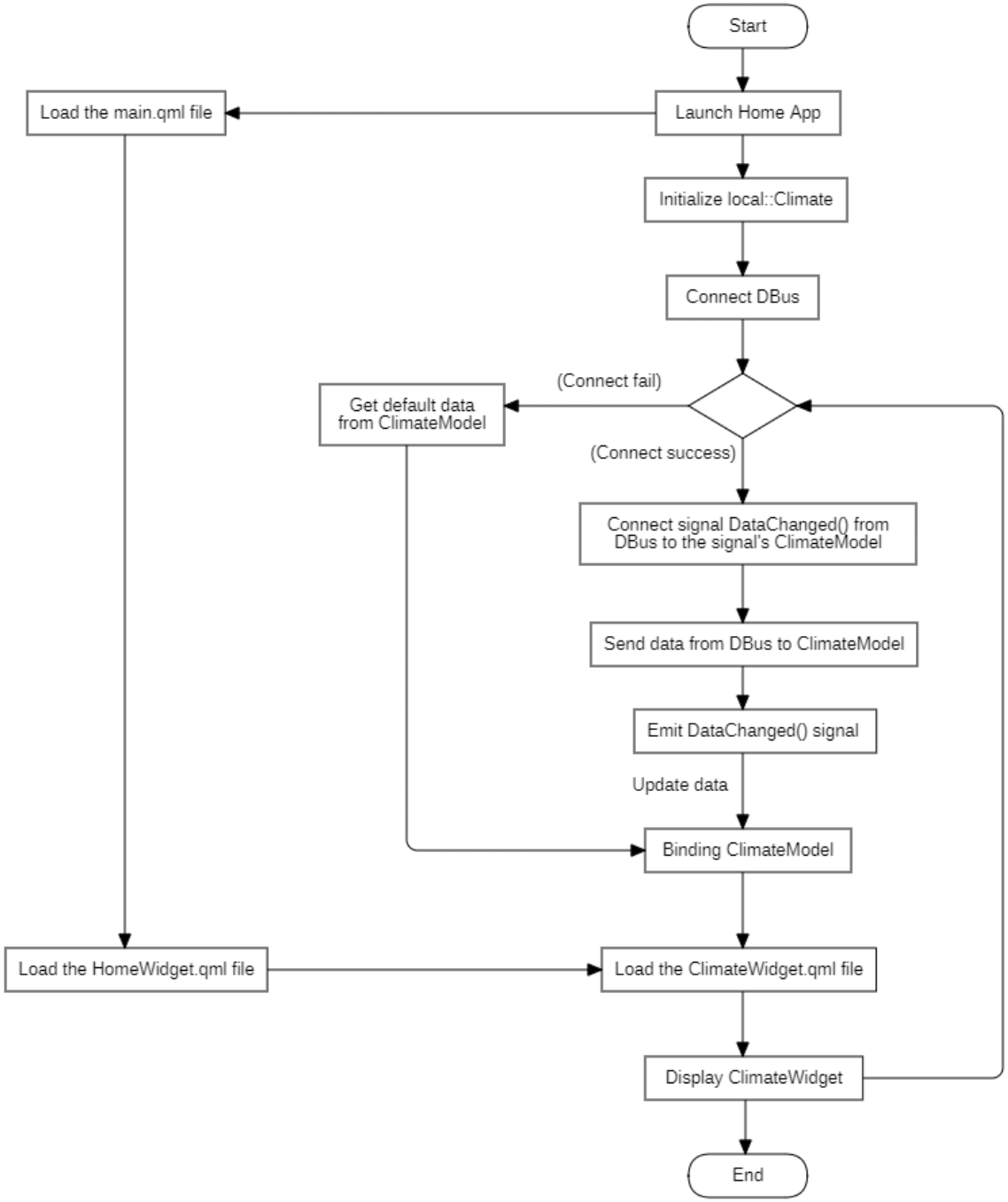
3. Receiving data on the Map Widget



- Steps to receive the data on the Map:

Step 1: Receive the data from the Map App
Step 2: Load file MapWidget.qml
Step 3: Load Map

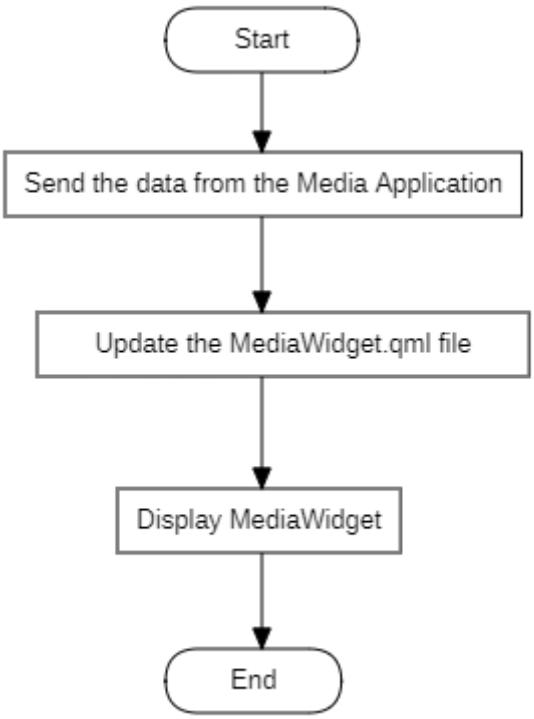
4. Receiving data on the Climate Widget



- Steps to receive the data on the Climate Widget:

Step 1: Create climate object of ClimateModel
Step 2: Create m_climate object of local::Climate
Step 3: Connect DataChanged signal of m_climate object to DataChanged signal of climate object
Step 4: Get data from Dbus to climate object
Step 5: Emit the DataChanged signal of ClimateModel
Step 6: Binding climate object of ClimateModel to QML using setContextProperty
Step 7: Update file ClimateWidget.qml
Step 8: Display Climate Widget

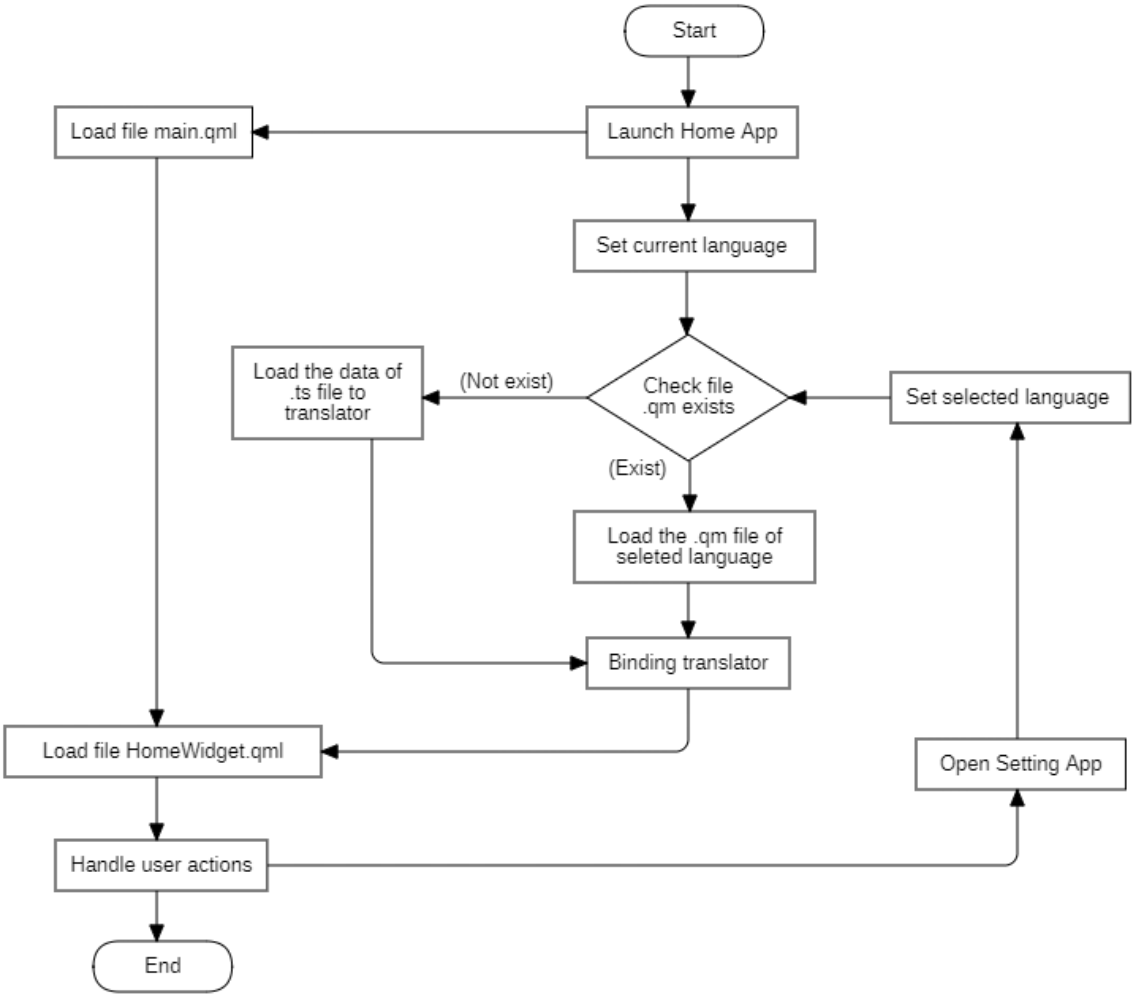
5. Receiving data on the Media Widget



- Steps to receive the data on the Media Widget:

Step 1: Receive the data from the Media Application
Step 2: Update the MediaWidget.qml file
Step 3: Display Media Widget

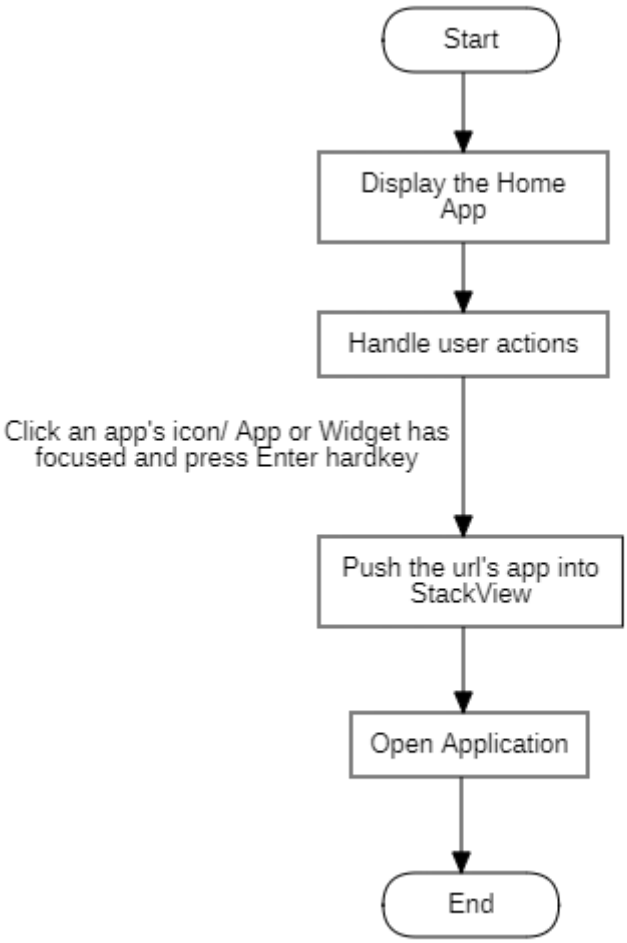
6. Changing Language in the Home Application



- Steps to change language in the Home Application:

Step 1: Create translator object of TranslatorSetting
Step 2: Set the initial language for Home App
Step 3: Check file .qm and load corresonding .qm file or and load corresonding .ts file
Step 4: Binding translator object to QML using setContextProperty
Step 5: Load file .qml of Home Screen
Step 6: Open Setting App
Step 7: Set the select language for Home App
Step 8: Back to step 3

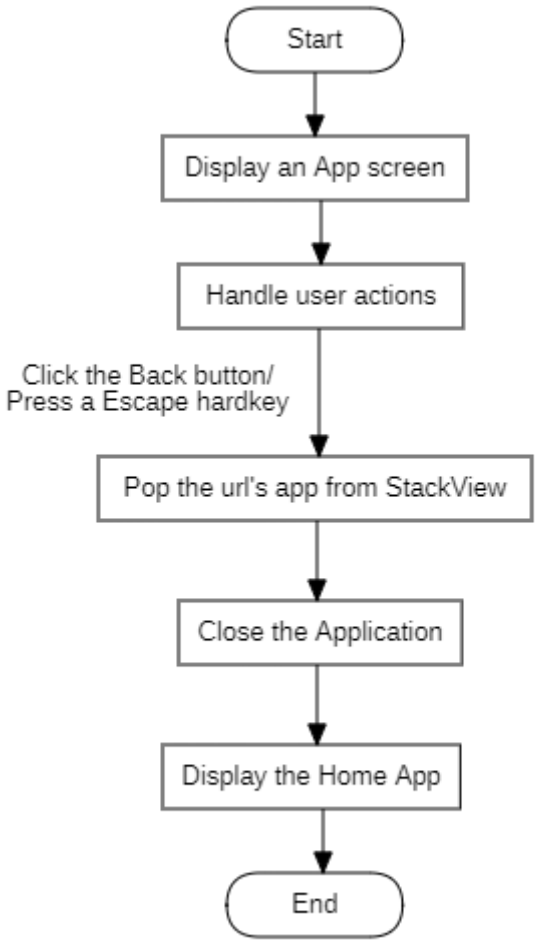
7. Opening Application



- Steps to open application:

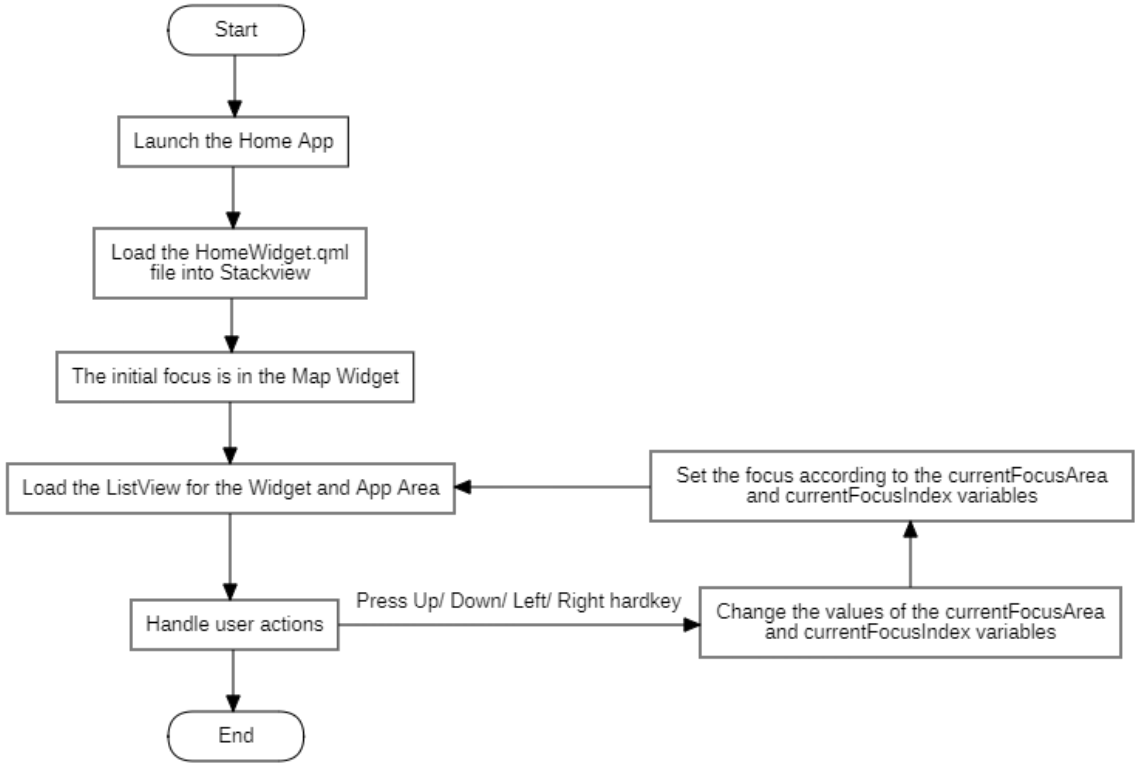
Step 1: Click/ App or Widget has focused and press Enter hardkey to open app
Step 2: Push the url of corresponding application into StackView
Step 3: Open an application

8. Closing



Step 1: Click the Back button/press the Escape hardkey to close app
Step 2: Pop null from StackView
Step 3: Close an application
Step 4: Return to the Home App

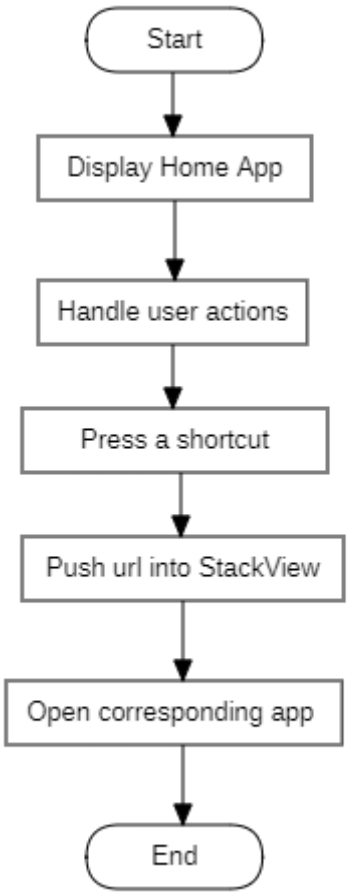
9. Hard Key Actions: Up / Down / Left / Right



- Steps to navigate focus using hardkey:

Step 1: Load main.qml
Step 2: Load HomeWidget.qml
Step 3: The Map Widget is initially focused
Step 4: Press Up/Down/Left/Right hardkey to change the value of the currentFocusArea and currentFocusIndex variables.
Step 5: Set focus according to these two variables.

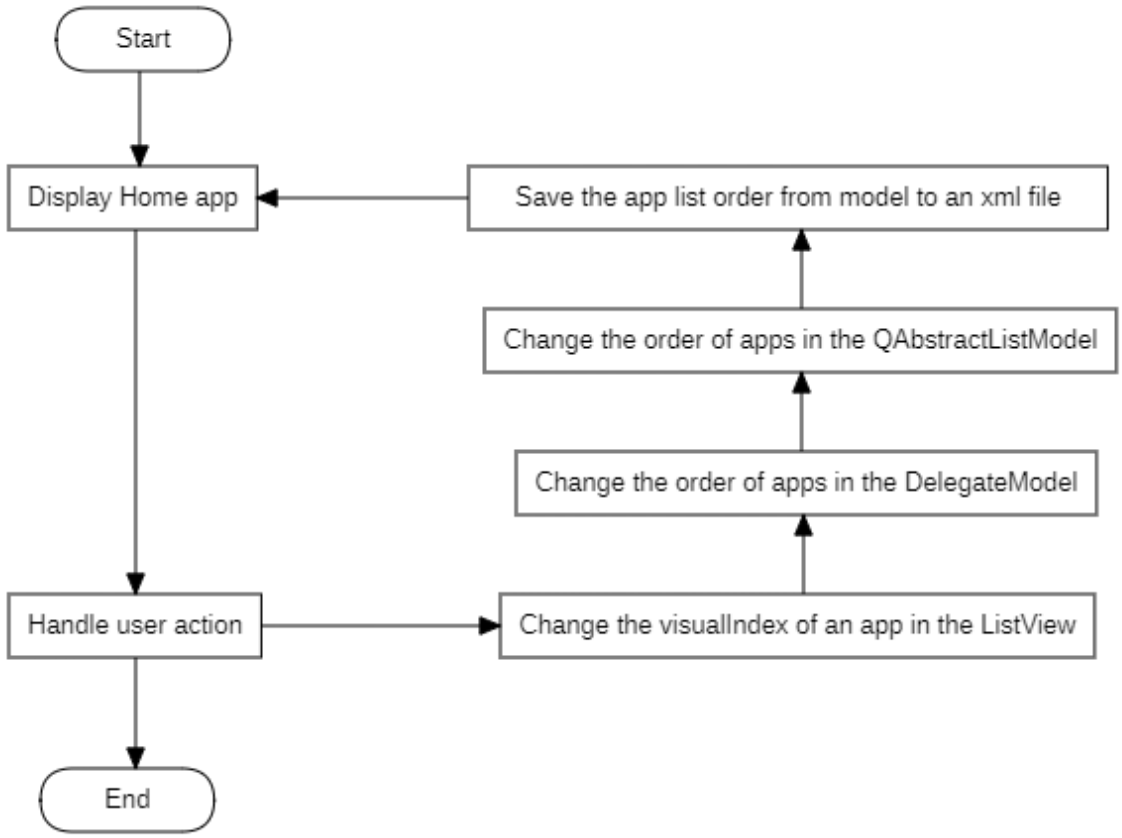
10. Shortcuts



- Steps to open app using shortcut:

Step 1: Press shortcut to open app
Step 2: Push the url into StackView
Step 3: Open corresponding app

11. Updating the order of application list



- Steps to update the order of application list:

Step 1: User reorder the application list

Step 2: Update the order of application list in the DelegateModel inside the ListView

Step 3 & 4: Update the order of application list in the appsModel object of ApplicationsModel and save model data to an XML file.

Step 5: The order of the app list is changed