Media Player

By Arden Yan, Davel Radindra, Thang Nguyen

Introduction

1. **Purpose**: "Our project aims to create a JavaFX-based media player that balances advanced features with ease of use."

2. **Significance**: "Addressing the gap in current media players that are either too complex or too limited in functionality."

3. **Target Audience**: "Designed for a wide range of users, compatible with Windows, Mac OS X, and Linux."

Approach

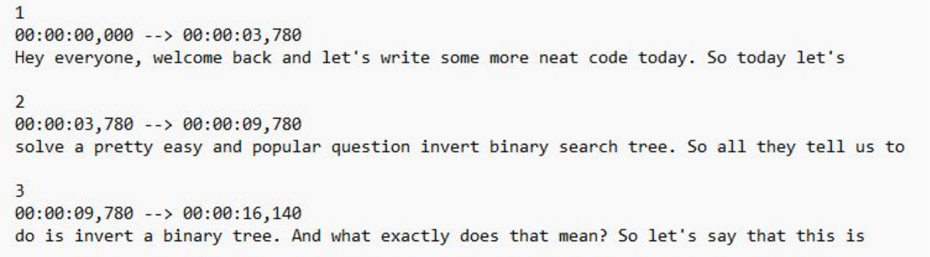
1. **Development Tools and Languages:** Java, JavaFX, FXML, JavaFX CSS, Scene Builder

2. **Key Features and Implementation:**

- Playback controls
- Media library management
- Subtitle support (.srt)

3. **Format Support:** MP4

.srt example

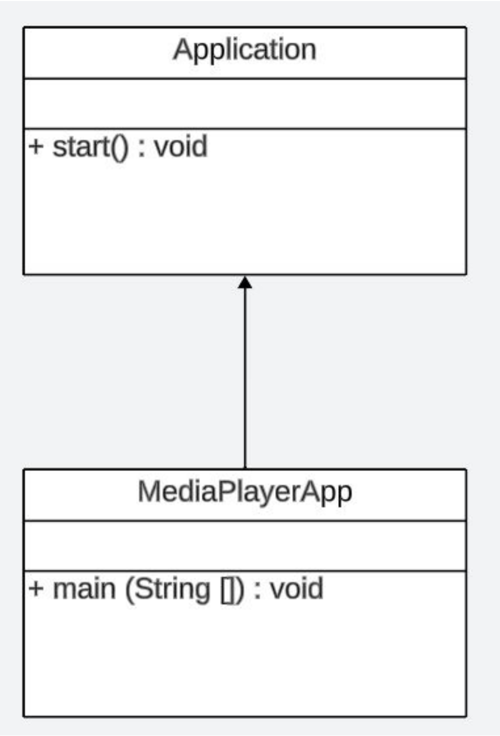


```
1
00:00:00,000 --> 00:00:03,780
Hey everyone, welcome back and let's write some more neat code today. So today let's

2
00:00:03,780 --> 00:00:09,780
solve a pretty easy and popular question invert binary search tree. So all they tell us to

3
00:00:09,780 --> 00:00:16,140
do is invert a binary tree. And what exactly does that mean? So let's say that this is
```

Class Diagrams

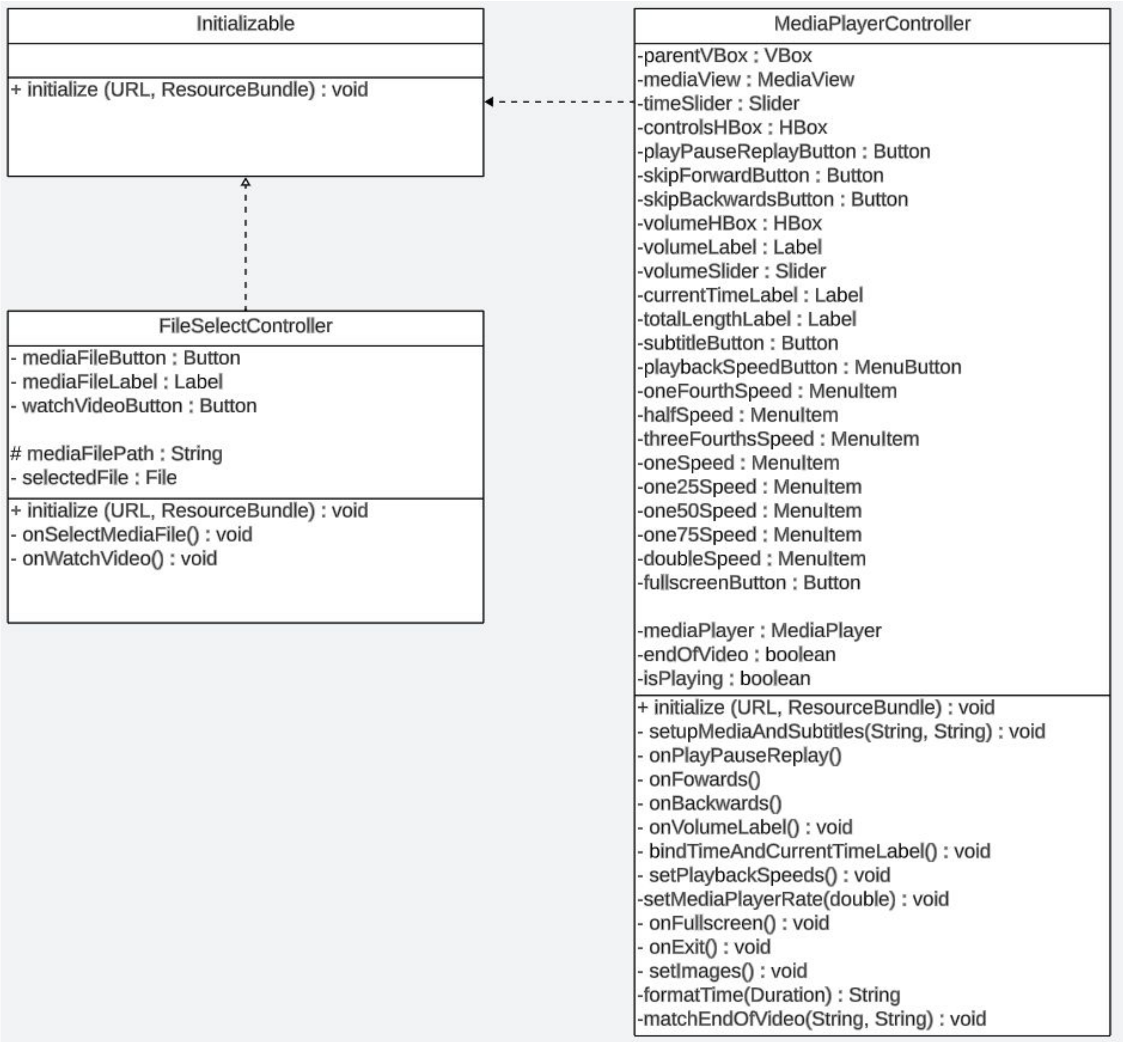


Application

+ start() : void

MediaPlayerApp

+ main (String []) : void



Initializable

+ initialize (URL, ResourceBundle) : void

FileSelectController

- mediaFileButton : Button
- mediaFileLabel : Label
- watchVideoButton : Button

# mediaFilePath : String
- selectedFile : File

+ initialize (URL, ResourceBundle) : void
- onSelectMediaFile() : void
- onWatchVideo() : void

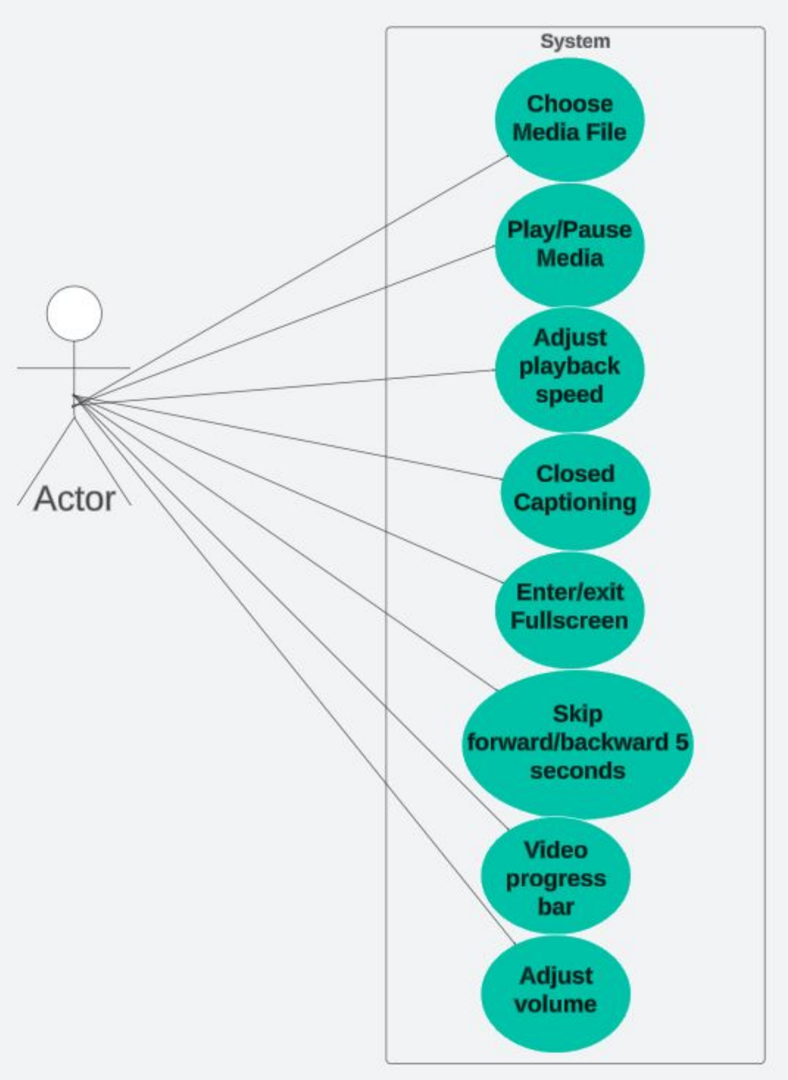
MediaPlayerController

-parentVBox : VBox
-mediaView : MediaView
-timeSlider : Slider
-controlsHBox : HBox
-playPauseReplayButton : Button
-skipForwardButton : Button
-skipBackwardsButton : Button
-volumeHBox : HBox
-volumeLabel : Label
-volumeSlider : Slider
-currentTimeLabel : Label
-totalLengthLabel : Label
-subtitleButton : Button
-playbackSpeedButton : MenuButton
-oneFourthSpeed : MenuItem
-halfSpeed : MenuItem
-threeFourthsSpeed : MenuItem
-oneSpeed : MenuItem
-one25Speed : MenuItem
-one50Speed : MenuItem
-one75Speed : MenuItem
-doubleSpeed : MenuItem
-fullscreenButton : Button

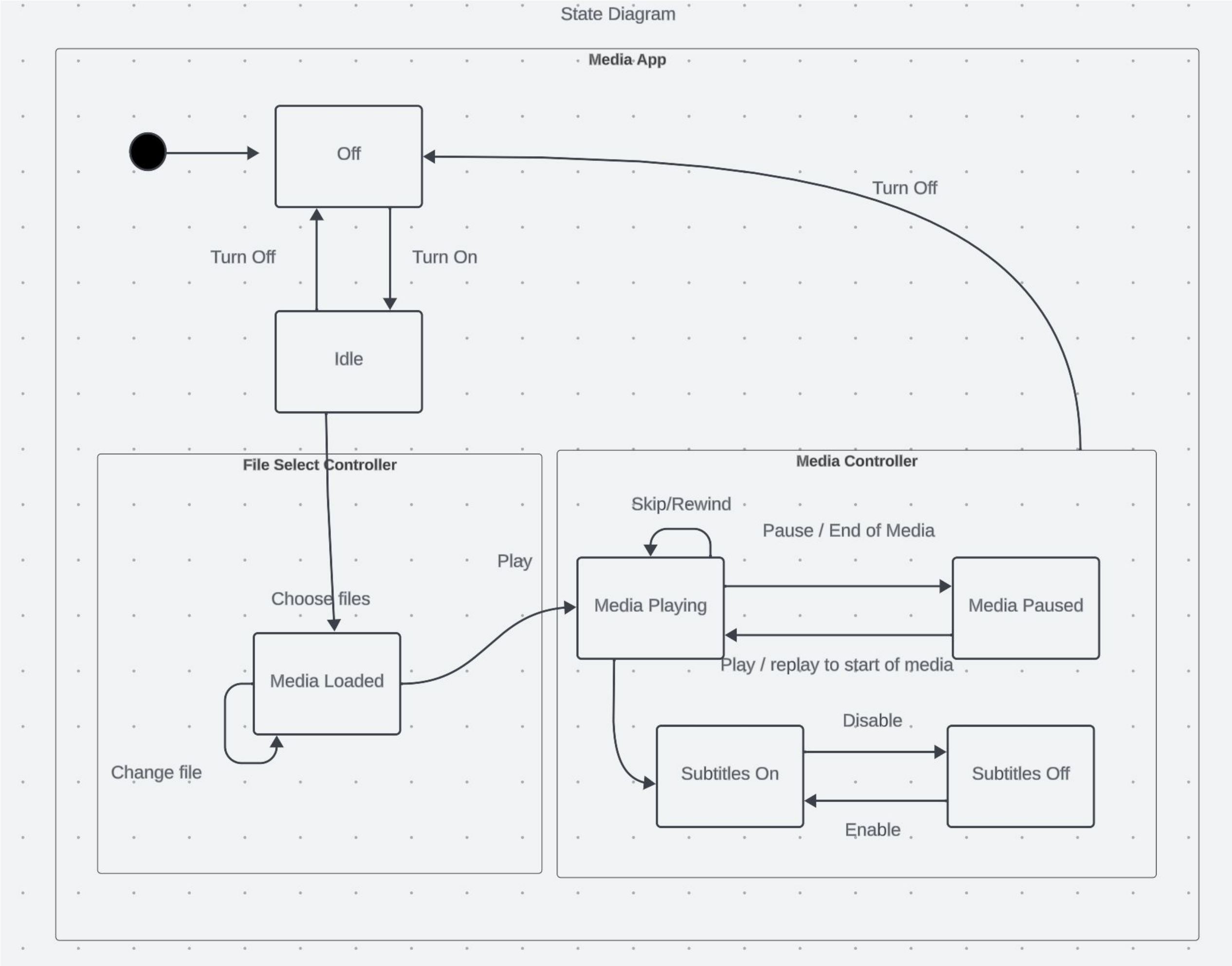
-mediaPlayer : MediaPlayer
-endOfVideo : boolean
-isPlaying : boolean

+ initialize (URL, ResourceBundle) : void
- setupMediaAndSubtitles(String, String) : void
- onPlayPauseReplay()
- onFowards()
- onBackwards()
- onVolumeLabel() : void
- bindTimeAndCurrentTimeLabel() : void
- setPlaybackSpeeds() : void
-setMediaPlayerRate(double) : void
- onFullscreen() : void
- onExit() : void
- setImages() : void
-formatTime(Duration) : String
-matchEndOfVideo(String, String) : void

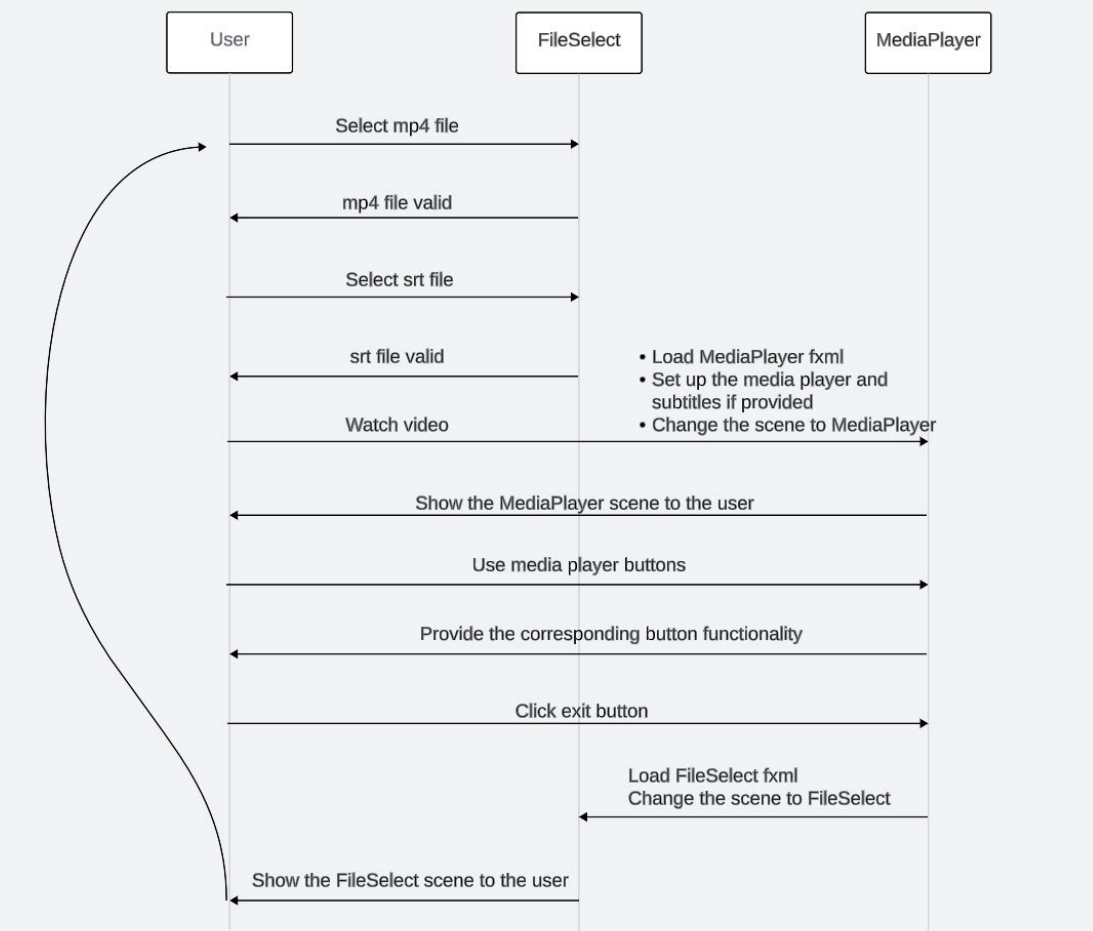
Use Case Diagram



State Diagram

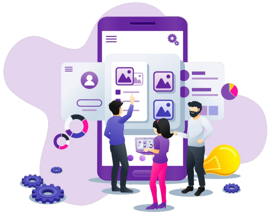


Sequence Diagram


User
FileSelect
MediaPlayer
Select mp4 file
mp4 file valid
Select srt file
srt file valid
• Load MediaPlayer fxml
• Set up the media player and subtitles if provided
• Change the scene to MediaPlayer
Watch video
Show the MediaPlayer scene to the user
Use media player buttons
Provide the corresponding button functionality
Click exit button
Load FileSelect fxml
Change the scene to FileSelect
Show the FileSelect scene to the user

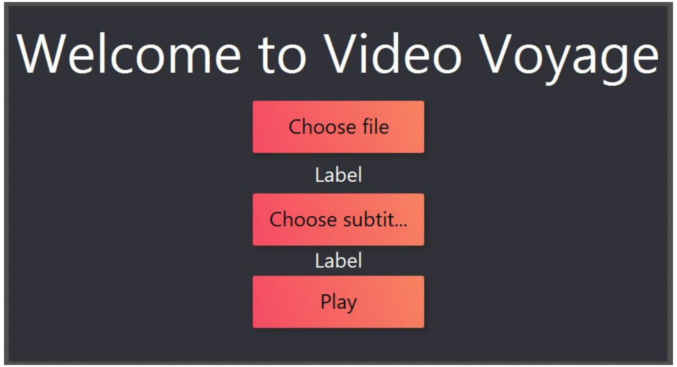
UI/UX Design

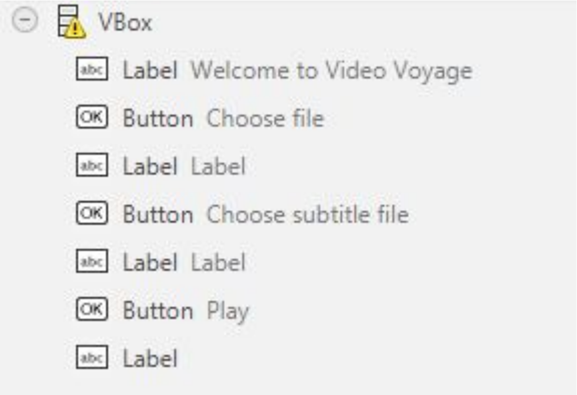
We intended to make our UI as friendly as possible. Using coolors.co, we selected and examined thoroughly to make sure that our colors contrasted well with each other. We also imported Google Fonts into our project as it offers a vast variety of fonts that can easily change the overall aesthetic. Additionally, we implemented informative messages throughout the media player, providing users with clear and reassuring information about their actions.



Scene Builder

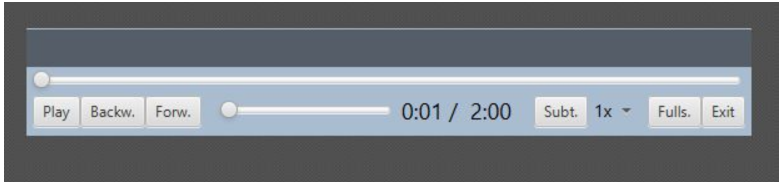
FileSelect.fxml

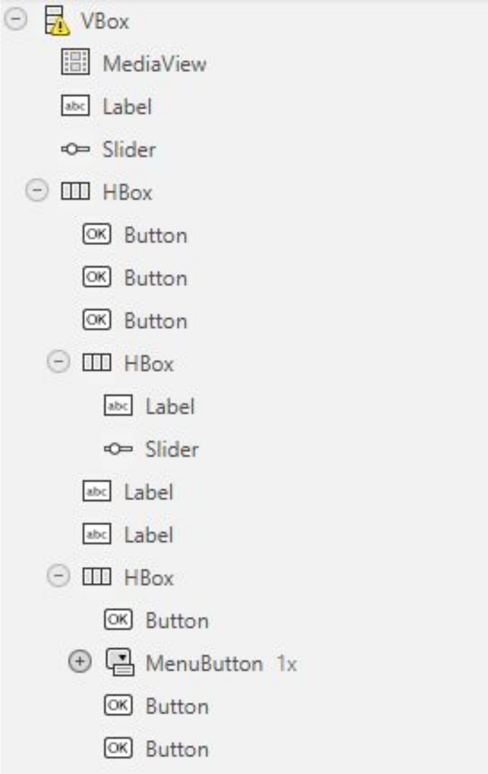




- VBox
  - Label Welcome to Video Voyage
  - Button Choose file
  - Label Label
  - Button Choose subtitle file
  - Label Label
  - Button Play
  - Label

MediaPlayer.fxml





- VBox
  - MediaView
  - Label
  - Slider
  - HBox
    - Button
    - Button
    - Button
    - HBox
      - Label
      - Slider
    - Label
    - Label
    - HBox
      - Button
      - MenuButton 1x
      - Button
      - Button

Testing

Whenever we introduce new features, we make sure to do **Manual Testing** its interactions with existing components to ensure it has proper communication with each other. Our testing approach tested different scenarios and explored various edge cases. Our goal is to prevent conflicts and guarantee a seamless implementation.

Challenges

One big challenge that we had to overcome was implementing the subtitles functionality… but after the parsing logic is done, the implementation becomes not so hard.

Styling in JavaFX was another challenge we faced since it lacks some modern properties and techniques. It also requires us to have some basic knowledge of HTML and CSS and transfer it over to FXML and JavaFX CSS.

Demo

Questions?