Object Oriented Analysis and Design Project Part 2

Team: Dea Allen, Sesha Sailendra Chetlur, Pravin Venkatesh Venkataraman
Title: Media Player
Project Summary: Our goal is to design and build a Media Player that can be extensively customized to specific user's needs. We intend to make this possible by extending the core part of the application through inclusion of modules or plug-ins.

Project Requirements:

Business Requirements				
ID	**Requirement**	**Topic Area**	**User**	**Priority**
BR-001	Copyright laws must be followed.	Legal	All	High
BR-002	Media Player should be a freeware.	Consumer	Developer	High

Non-Functional Requirements				
ID	**Requirement**	**Topic Area**	**User**	**Priority**
NFR-001	Upon clicking any button a user should be given feedback within 2 seconds max.	Performance	Product user	High
NFR-002	Media Player should be platform independent.	Supportability	Product user	High
NFR-003	A user should be able to load and play media in the Media Player without instructions.	Usability	Product user	Critical
NFR-004	A user should be able create and delete a playlist without instructions.	Usability	Product user	Critical
NFR-005	Media Player should come with a help document.	Usability	All	Medium

Functional Requirements				
ID	**Requirement**	**Topic Area**	**User**	**Priority**

| FR-001	Media Player should prevent duplicate media files being added to a playlist.	Usability	Product User	Medium
FR-002	Media Player should save its state upon being closed.	Usability	Product User	High
FR-003	Media Player should validate a newly added module.	Technical	Product Developer	High
FR-004	Media Player should validate the playlist name entered by a user before the playlist is created.	Technical	Product User	Medium
FR-005	Media Player should be able to play any file format so that user doesn't need to worry about format conversions.	Usability	Product User	Medium
FR-006	Media Player should display information about currently playing media.	Usability	Product User	Medium

User Requirements				
ID	**Requirement**	**Topic Area**	**User**	**Priority**
UR-001	As a product user, I want to be able to open a media file so that I can begin playing it.	Technical	Product user	Critical
UR-002	As a product user, I want to be able to play a media file so that I can be entertained.	Technical	Product user	Critical
UR-003	As a product user, I want to be able to stop a media file so that I can leave, come back and resume my interaction with the player as per my convenience.	Technical	Product user	Critical
UR-004	As a product user, I should be able to choose from multiple skin options so that I can customize my view.	User Interface	Product User	Low

UR-005	As a product user, I should be able to seek to any part of my selected media, so that I can watch or listen to whichever part of the media I want without having to wait for it to reach the desired point on its own accord.	Usability	Product User	High
UR-006	As a product user, I should be able to bookmark times in a particular media file so that seeking to my desired part will be easy.	Technical	Product User	Medium
UR-007	As a product user, I should be able to create playlists so that I will be able to organize and play a series of media.	Technical	Product User	Medium
UR-008	As a product user, I should be able to add media to playlists so that I will be able to reorganize my media.	Technical	Product User	Medium
UR-009	As a product user, I should be able to remove media from playlists so that I will be able to reorganize my media.	Technical	Product User	Medium
UR-010	As a product user, I should be able to delete playlists so that I will be able to get rid of playlists that I no longer want.	Technical	Product User	Medium
UR-011	As a product developer, I should be able to add a custom module so that I may customize my Media Player.	Technical	Product Developer	Medium
UR-012	As a product user, I should be able to crop media so that I can edit my media files.	User Interface	Product User	Medium

Use Cases:

    Use Case Overview:
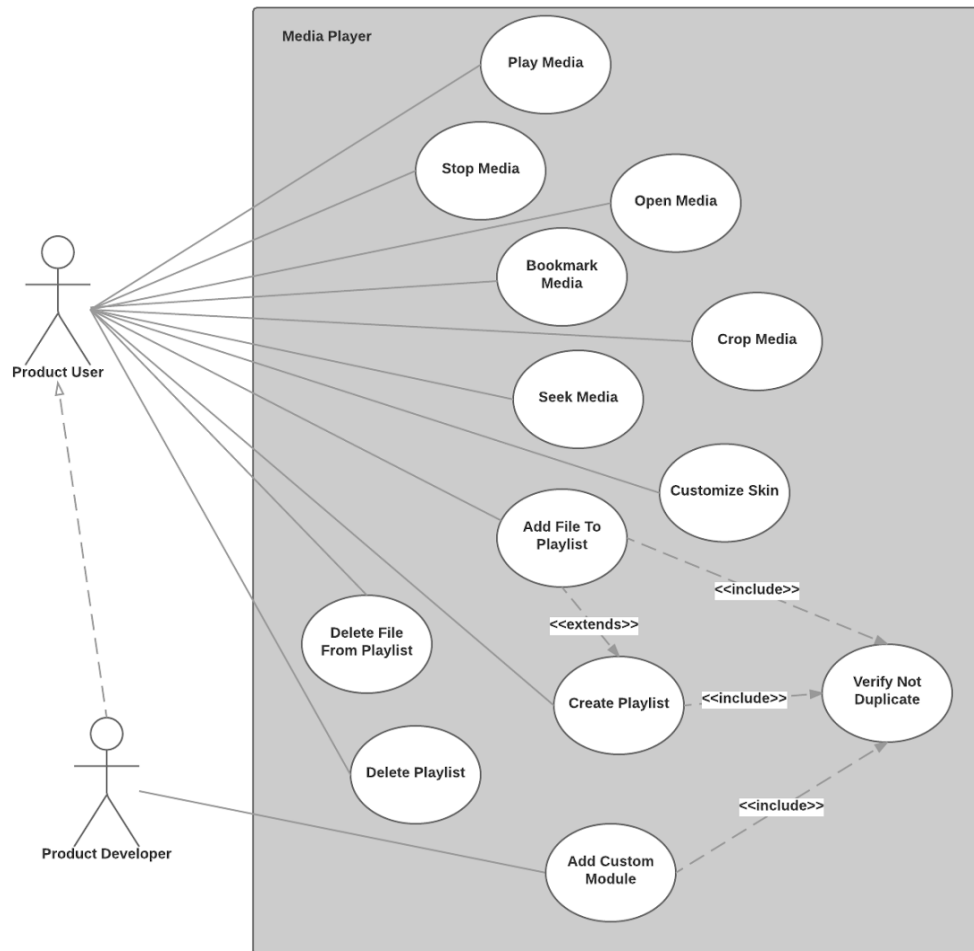
Use Case Documents:

|>

Use Case ID:	UC-001
Use Case Name:	Open Media
Description:	User can open a media file by selecting the 'open' option under the menu bar located in the GUI.

Actors:	Product Users	
Pre-conditions:	Media Player should should be open. Media file should exist.	
Post-conditions:	Media file selected is opened by the player.	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Select 'Open' option in 'File' menu	Open file browser

		from top of GUI	
	2	Select media in file browser	Open media in Media Player
Variations:	None.		
Notes and Issues:			
Developer Notes:			

II>

Use Case ID:	UC-002		
Use Case Name:	Play Media		
Description:	User can play a selected/loaded media by pressing the play button on the media player GUI window.		

Actors:	Product Users		
Pre-conditions:	Media player GUI window should be open, and media to be played should be selected.		
Post-conditions:	Media file selected is played by the player.		
Frequency of Use:	Depends.		
Flow of Events:		Actor Action	System Response
	1	Press play button	Play media
Variations:	None.		
Notes and Issues:			
Developer Notes:			

III>

Use Case ID:	UC-003		
Use Case Name:	Stop Media		
Description:	User can stop a currently playing media by pressing the stop button on the media player GUI window.		

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open, and media to be stopped should be playing.	
Post-conditions:	Media file playing is stopped by the player.	
Frequency of Use:	Every time a media is playing.	
Flow of Events:	Actor Action	System Response
	1 Press stop button	Stop media
Variations:	1. User closes GUI window	
Notes and Issues:		
Developer Notes:		

IV>

Use Case ID:	UC-004
Use Case Name:	Seek Media
Description:	User can seek to a selected location of the media file by controlling the seek bar on the media player GUI window.

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open, and media to be navigated should be selected.	
Post-conditions:	Media player navigates to the desired location of the media file and is ready to perform the next action.	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Drag seek bar	Navigates to location
Variations:	None.	
Notes and Issues:		

Developer Notes:	

V>

Use Case ID:	UC-101
Use Case Name:	Crop Media
Description:	User can select a part of the media file and delete the selected section using the media player GUI window.

Actors:	Product Users		
Pre-conditions:	Media player GUI window should be open, and media to be cropped should be selected.		
Post-conditions:	Media player crops file.		
Frequency of Use:	Depends.		
Flow of Events:		Actor Action	System Response
	1	Select 'File' menu from top of GUI window	Open 'File' menu
	2	Select 'crop' from File menu	Make cursor crop selector cursor
	3	Select region	Highlights region
	4	Press delete	Crops file
Variations:	None.		
Notes and Issues:			
Developer Notes:			

VI>

Use Case ID:	UC-102
Use Case Name:	Bookmark Media
Description:	User can navigate to a location and set a bookmark using the media player GUI window.

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open, and media to be bookmarked should be selected.	
Post-conditions:	Media player sets bookmark.	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Drag seek bar to a location	Navigates to location
	2 Press bookmark button	Set bookmark wherever seek bar is at the time.
Variations:	None.	
Notes and Issues:		
Developer Notes:		

VII>

Use Case ID:	UC-103
Use Case Name:	Customize Skin
Description:	User can choose a skin to customize the Media Player

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open.	
Post-conditions:	Media player sets skin.	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Select 'Skins' from menu bar	Display existing skins
	2 Select skin from list	Set skin
Variations:	None.	
Notes and Issues:		

Developer Notes:	

VIII>

Use Case ID:	UC-201
Use Case Name:	Create Playlist
Description:	User can create an empty playlist using the media player GUI window.

Actors:	Product Users		
Pre-conditions:	Media player GUI window should be open. The playlist panel should be open.		
Post-conditions:	A playlist is created in the Media player.		
Frequency of Use:	Depends.		
Flow of Events:		Actor Action	System Response
	1	Select 'Create Playlist' in the playlist panel in the GUI	Prompt user for playlist name
	2	Enter playlist name	Create playlist and open file browser
	3	Select media to add to playlist	Add media to playlist
Variations:	3. Choose not to add media to playlist		
Notes and Issues:			
Developer Notes:			

IX>

Use Case ID:	UC-202
Use Case Name:	Add To Playlist
Description:	User can add media files to an existing playlist using the Media player GUI window.

Actors:	Product Users

Pre-conditions:	Media player GUI window should be open. The playlist panel should be open.	
Post-conditions:	Media files added to selected playlist	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Press '+' button in playlist panel	Open file browser
	2 Select media file(s) in file browser	Add media file(s) to playlist
Variations:	None.	
Notes and Issues:		
Developer Notes:		

X>

Use Case ID:	UC-203
Use Case Name:	Delete From Playlist
Description:	User can delete media files from an existing playlist using the Media player GUI window.

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open. The playlist panel should be open.	
Post-conditions:	Media file deleted from selected playlist	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Select playlist in GUI	Open media panel for selected playlist
	2 Press '-' button in selected playlist panel	Delete selected media file
Variations:	None.	
Notes and Issues:		

Developer Notes:	

XI>

Use Case ID:	UC-204
Use Case Name:	Delete Playlist
Description:	User can delete an existing playlist using the Media player GUI window.

Actors:	Product Users	
Pre-conditions:	Media player GUI window should be open. The playlist panel should be open.	
Post-conditions:	Media playlist deleted	
Frequency of Use:	Depends.	
Flow of Events:	Actor Action	System Response
	1 Press 'x' button in playlist panel	Delete selected playlist
Variations:	None.	
Notes and Issues:		
Developer Notes:		

XII>

Use Case ID:	UC-300
Use Case Name:	Add Custom Module
Description:	Product developer can add a custom module that specializes the base Media Player.

Actors:	Product Developer
Pre-conditions:	Product developer should have created a custom module and opened the player
Post-conditions:	Custom module added to Media Player
Frequency of Use:	Frequently by Product Developers.

Flow of Events:		Actor Action	System Response
	1	Select 'Modules' menu from menu bar	Open 'Modules' menu
	2	Select 'Add Module' from dropdown list	Open file browser
	3	Select module to add from file browser	Add selected module
Variations:	None.		
Notes and Issues:			
Developer Notes:	This could also be done by copy and pasting the module snippet into the source code.		

Activity Diagrams:

1>
Requirement ID: UR-001
Use Case ID: UC-001
Use Case Name and Description: Open Media. Opens a media file in the Media Player.
Implementer: Pravin Venkatesh Venkataraman
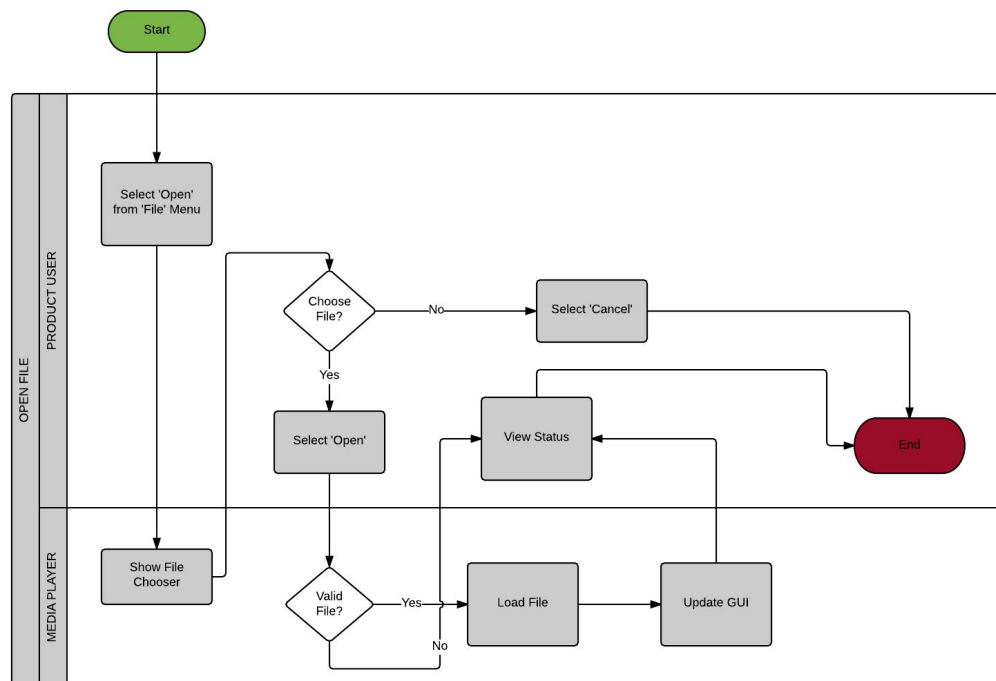
2>
Requirement ID: UR-007
Use Case ID: UC-201
Use Case Name and Description: Create Playlist. Allows user to create a playlist.
Implementer: Sesha Chetlur

3>
Requirement ID: UR-008
Use Case ID: UC-202
Use Case Name and Description: Add to Playlist. Allows user to add media files to a selected playlist.
Implementer: Sesha Chetlur

4>
Requirement ID: UR-009
Use Case ID: UC-203
Use Case Name and Description: Delete from Playlist. Allows user to delete media files from a selected playlist.
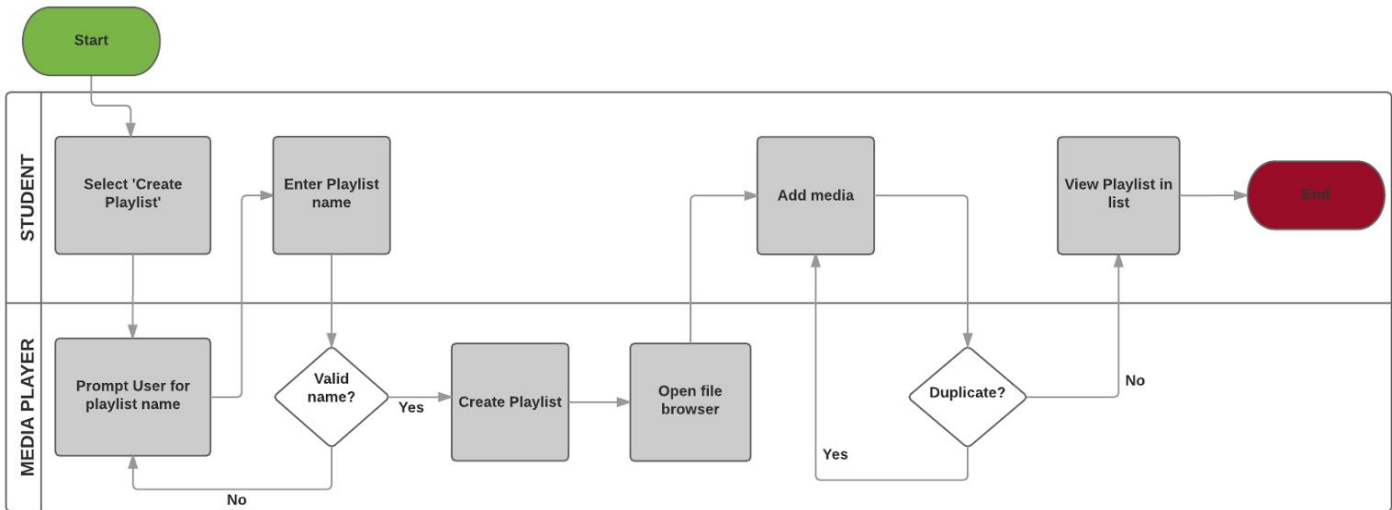Implementer: Sesha Chetlur
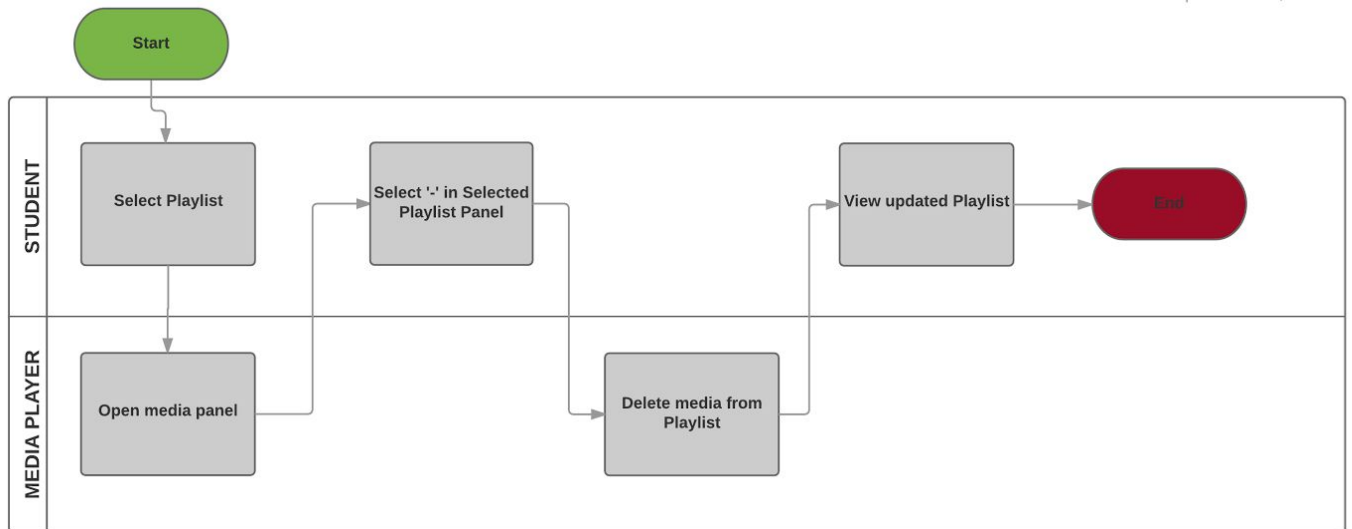
5>
Requirement ID: UR-010
Use Case ID: UC-204
Use Case Name and Description: Delete Playlist. Allows user to delete a playlist.
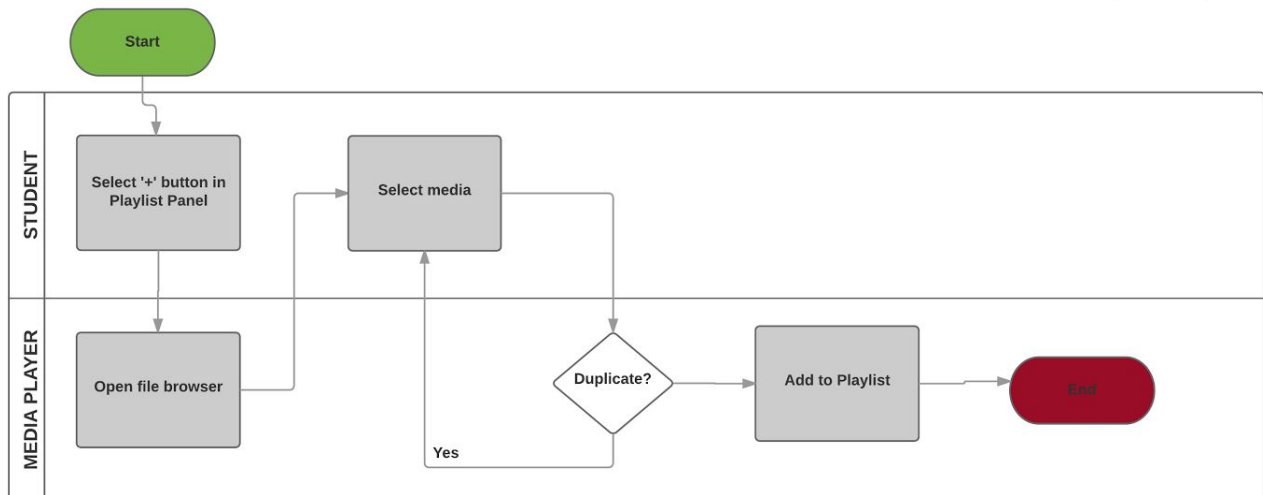Implementer: Sesha Chetlur
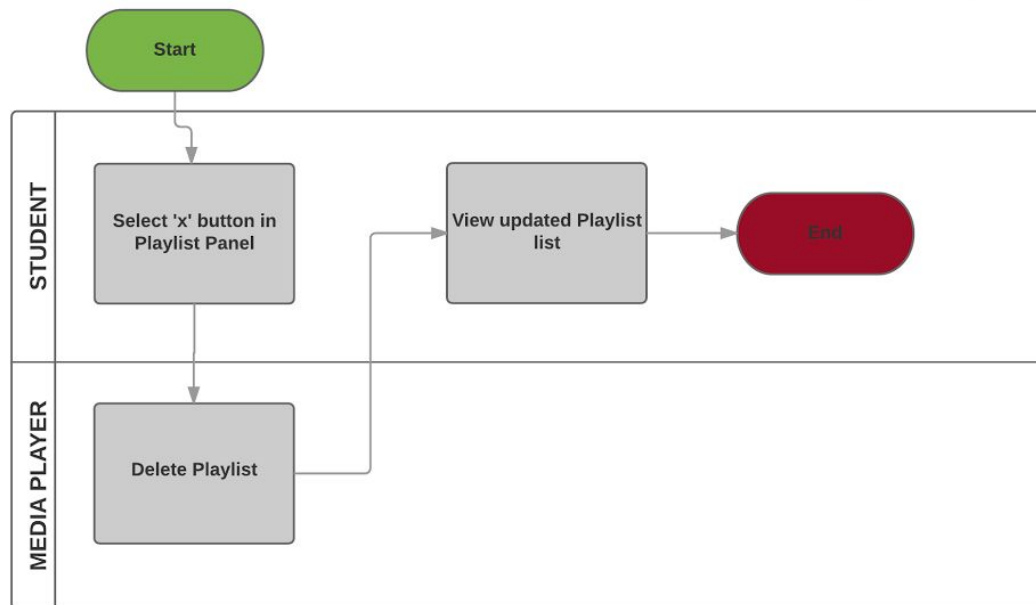
6>
Requirement ID: UR-006
Use Case ID: UC-300
Use Case Name and Description: Add Custom Module. Allows third party developers to add custom modules to customize their Media Players.
Implementer: Dea Allen
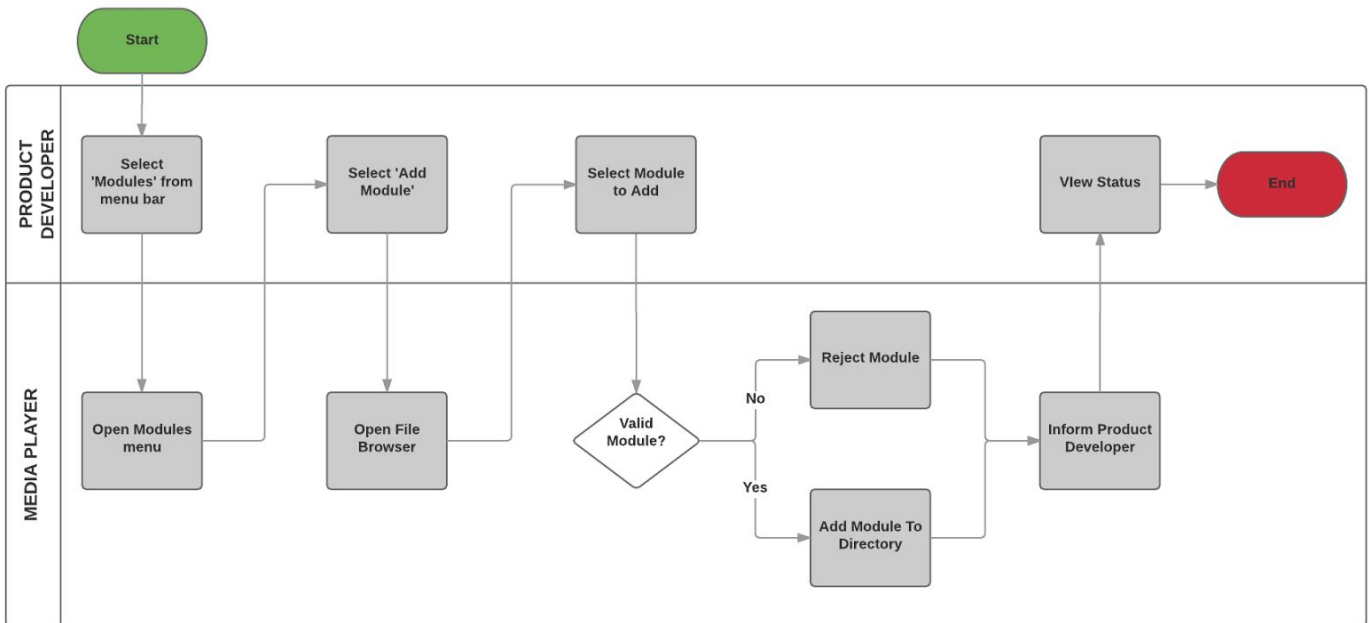
Data Storage:

- Playlists and State Information will be stored in JSON format in text files.
- Classes:
    - PlaylistModel class to store information about individual playlists. The PlaylistModel includes a list of MediaModels, which include the full paths of media files and a list of bookmarks for each media file.
    - PlayerStateModel class to store the state of the media player when it is closed. The PlayerStateModel will store the following information: current gui model, currently playing media, currently selected playlist, and a list of the enabled modules.

UI Mockups:

- Initial View
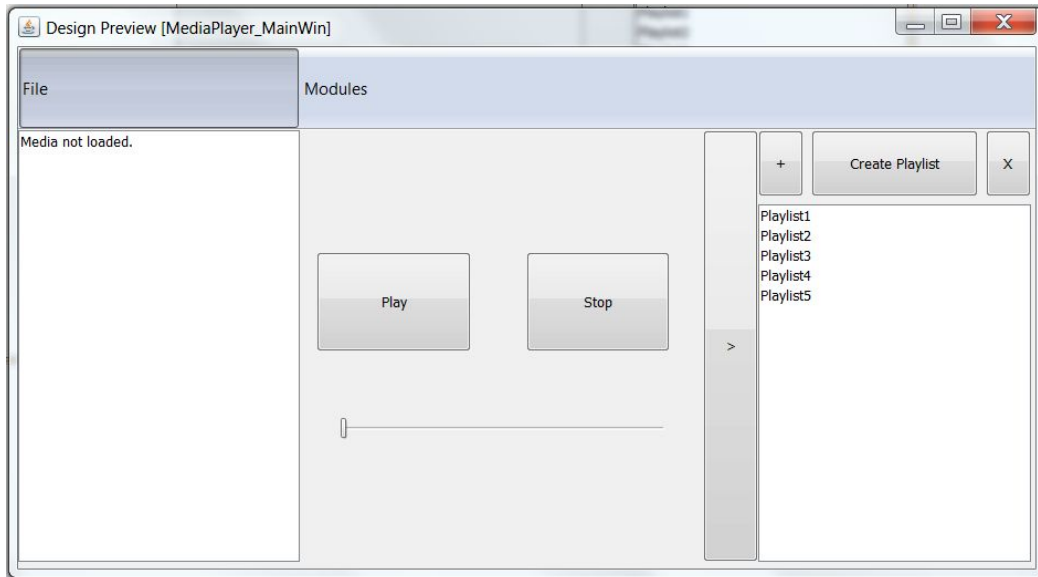
- Opening a file
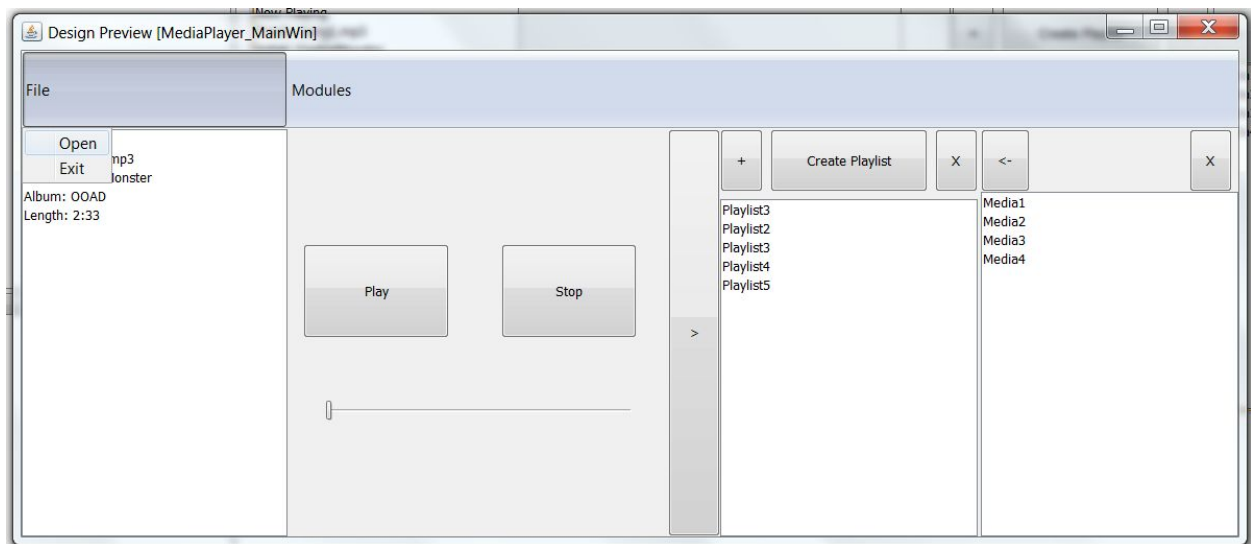
- Choosing or Adding Modules
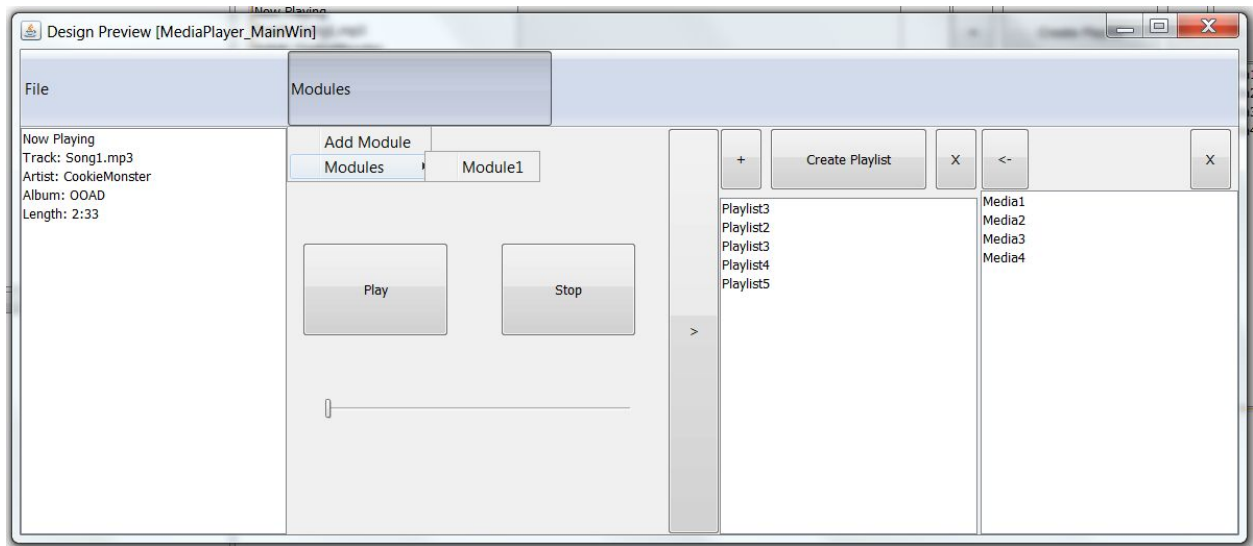
- Playing a file from playlist
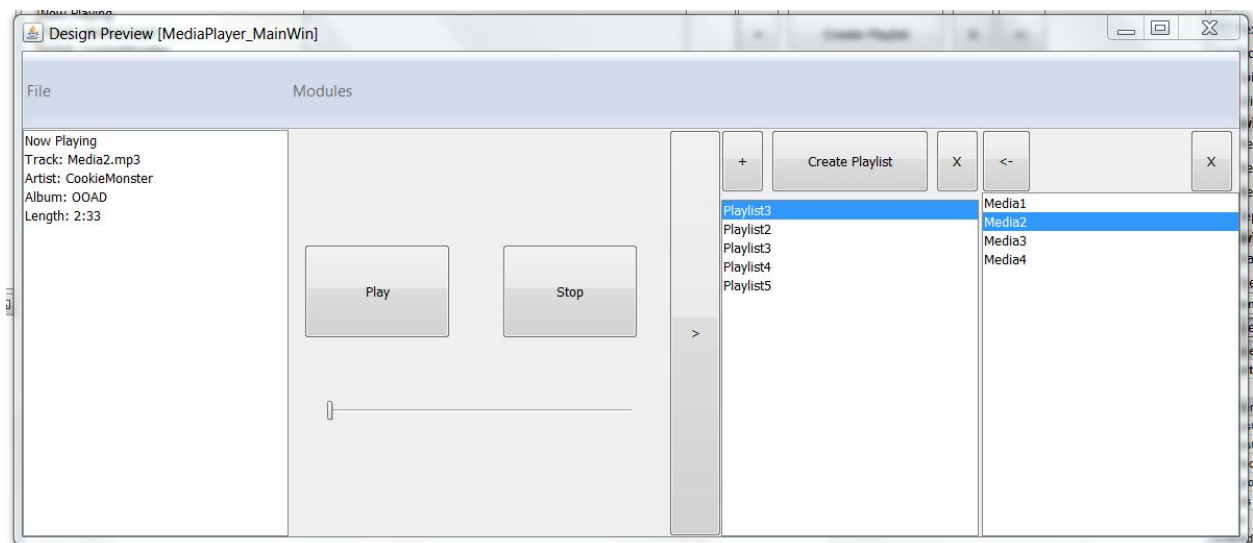
- Playing a file with the playlist panel collapsed
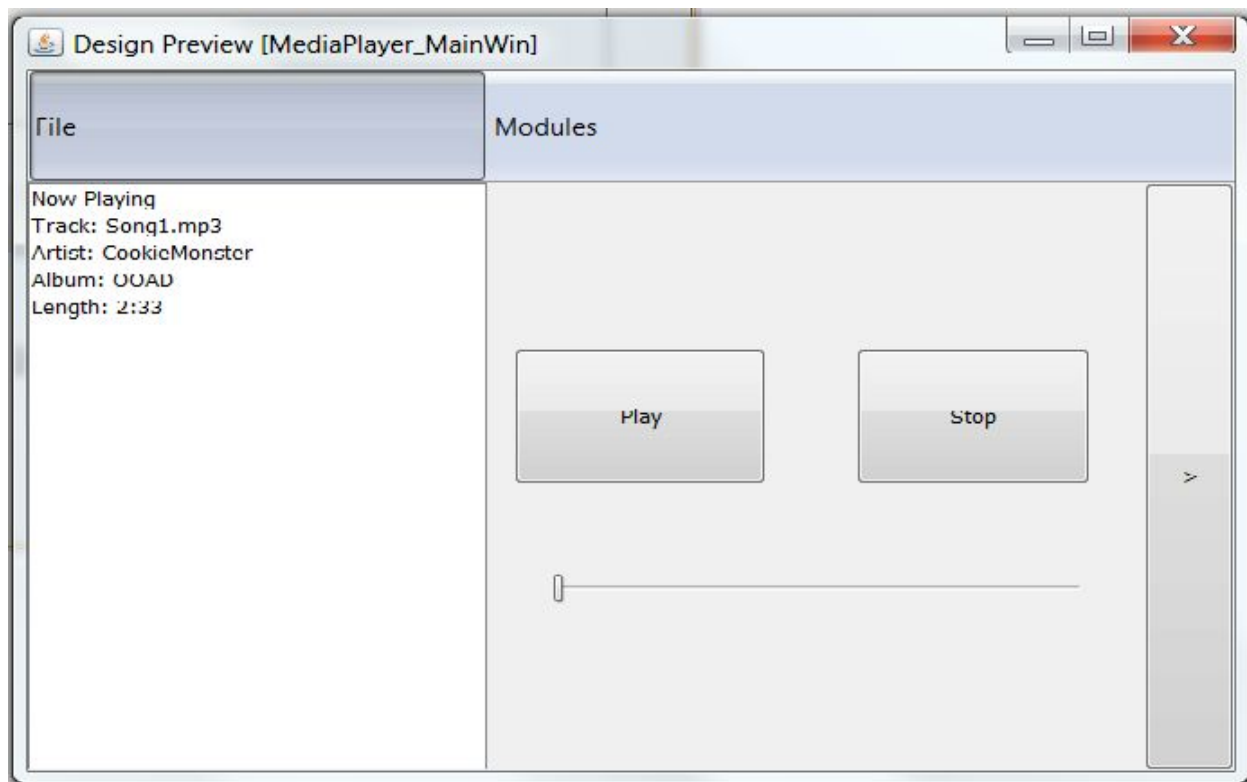
User Interactions:
Note: These sequence diagrams were created in Lucidcharts using the default template. The class names appear in all capital letters; we were not able to change this behavior. Our class diagram shows the class names in camelcase.

1>
Requirement ID: UR-001
Use Case ID: UC-001
Use Case Name and Description: Open Media. Opens a media file in the Media Player.
Implementer: Pravin

2>
Requirement ID: UR-007
Use Case ID: UC-201
Use Case Name and Description: Create Playlist. Allows user to create a playlist.
Implementer: Sesha Chetlur

3>
Requirement ID: UR-006
Use Case ID: UC-300
Use Case Name and Description: Add Custom Module. Allows third party developers to add custom modules to customize their Media Players.
Implementer: Dea Allen

Product Developer

:GUIVIEW   :EVENTMAPPER   :GUIMODEL

Click 'Modules' menu
Display Modules menu
Click 'Add Module'

actionPerformed(event)
<<create>>   :MODULECONTROLLER

addModule()
<<create>>   FC:JFILECHOOSER

showOpenDialog()
Display File Browser
Select moduleFile
showDialogReturnStatus
[showDialogReturnStatus] getSelectedFile()
moduleFile
valid = validateModule(moduleFile)

[valid] addToDirectory(moduleFile)

getInstance()

[instance has not been created] <<create>>   GUIINSTANCE:GUIMODEL

guiInstance
guiInstance

setModuleList(moduleFile)

updateGuiView()
Display Updated View
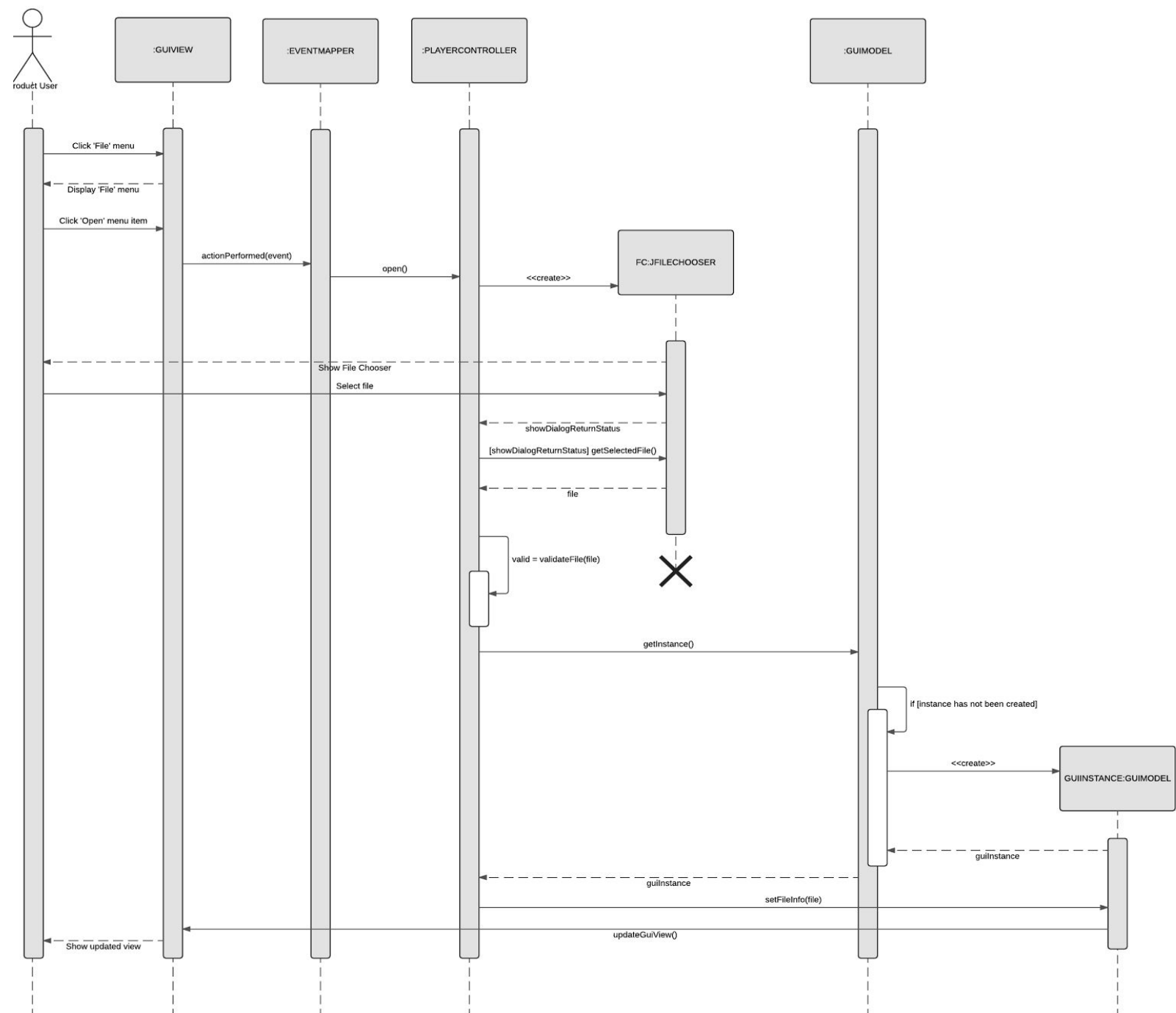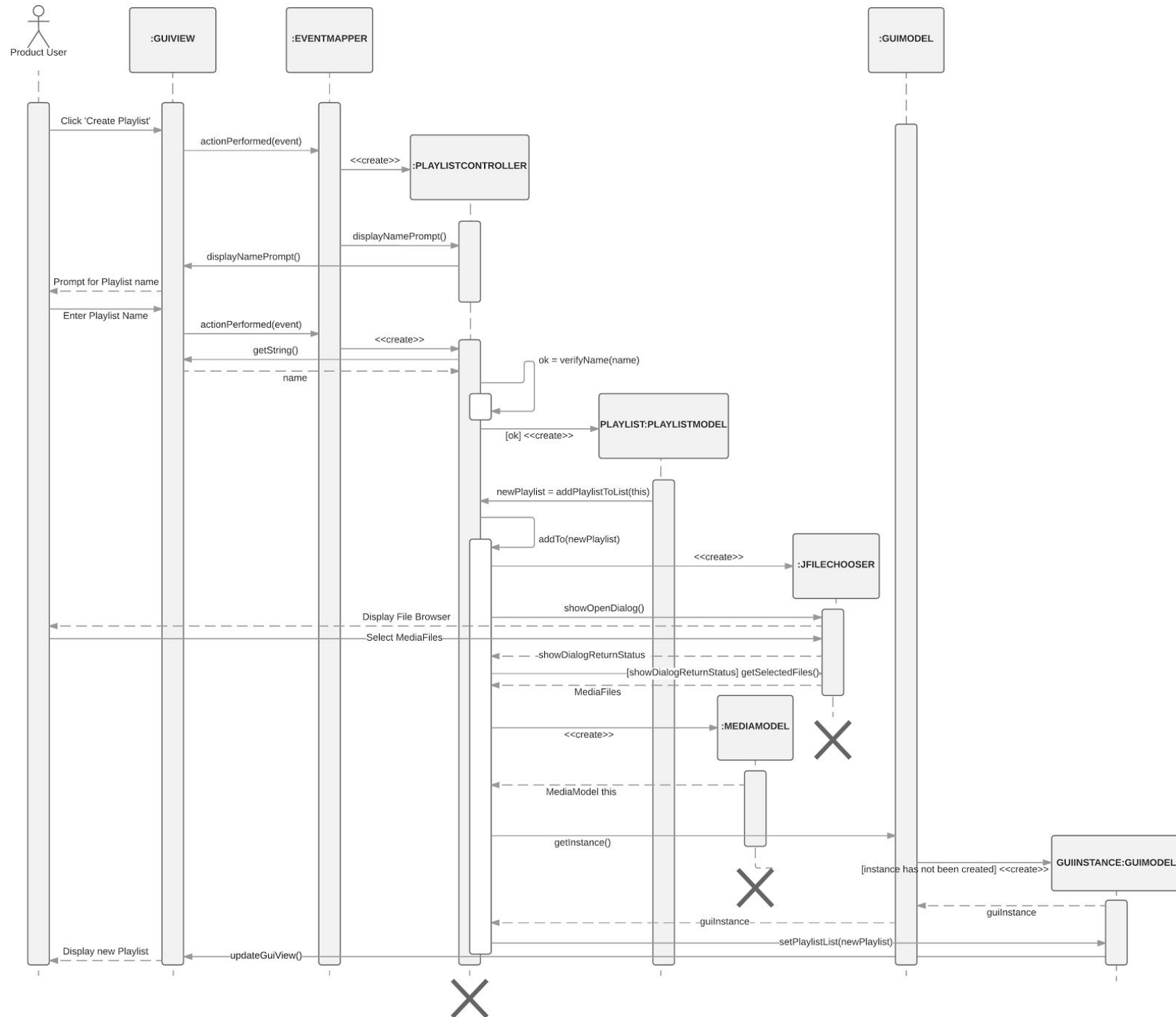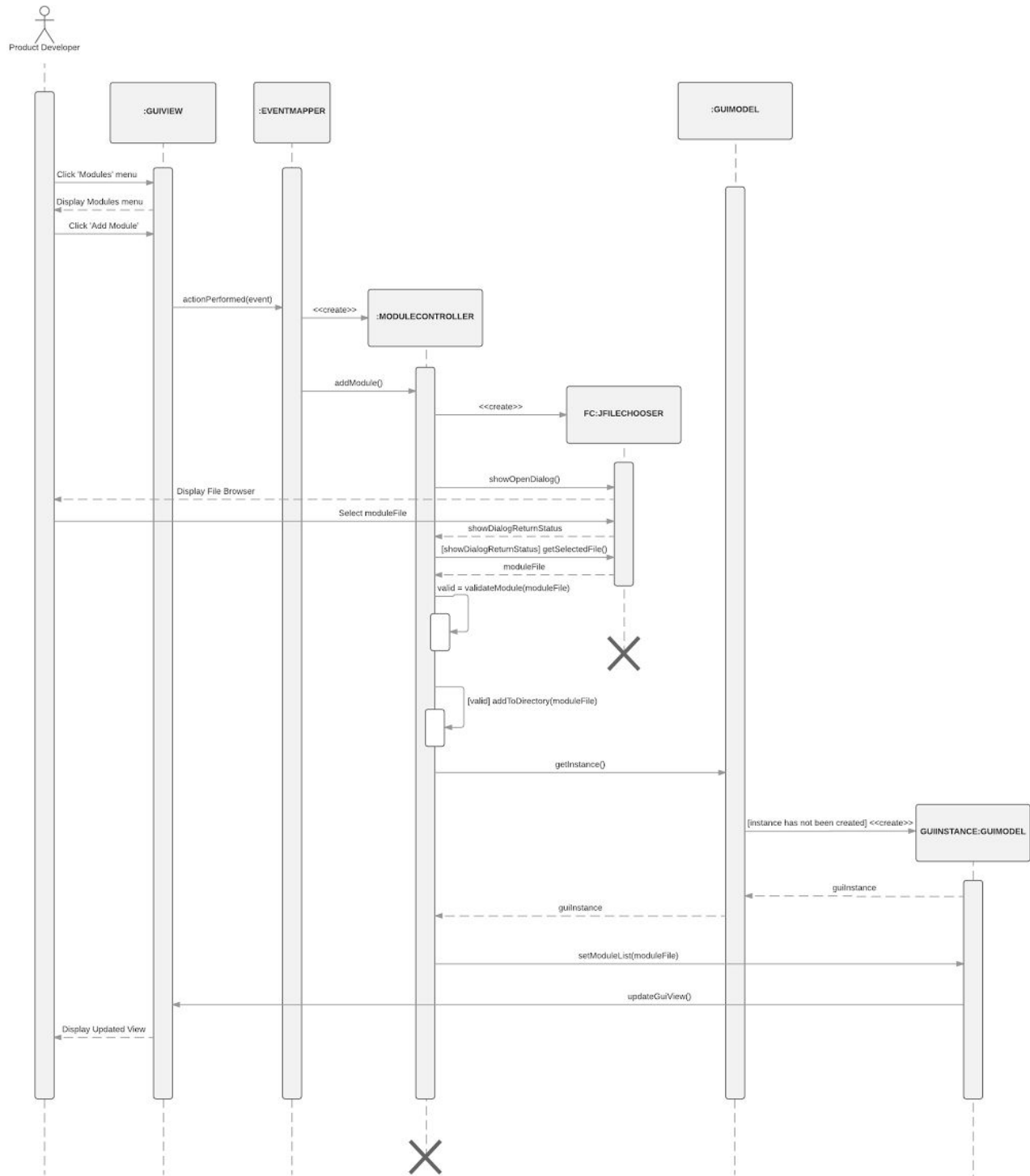
Classes:
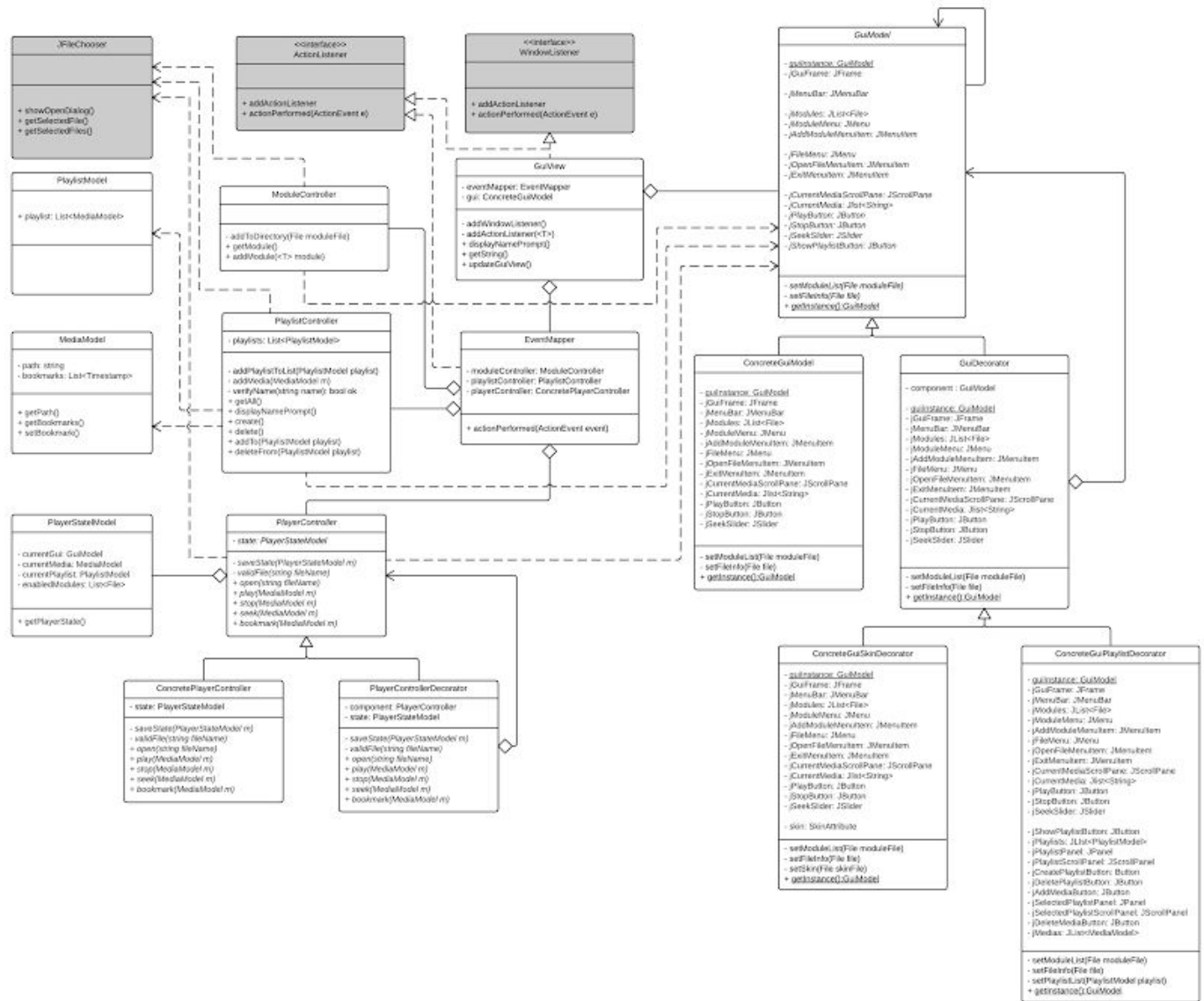
- **Class Diagram:**

- **Link to high quality image of class diagram:**
  https://github.com/dea-allen/OOAD_MediaPlayer/blob/master/MediaPlayer_ClassDiagra
  m.png

- Our class diagram split into three zoomed in sections:

1. GuiView, EventMapper, ModuleController, and PlaylistController classes where we are implementing an Observer Pattern.

2. GuiModel class where we are implementing the Decorator Pattern.

GuiModel

- *guiInstance: GuiModel*
- *jGuiFrame: JFrame*

- *jMenuBar: JMenuBar*

- *jModules: JList<File>*
- *jModuleMenu: JMenu*
- *jAddModuleMenuItem: JMenuItem*

- *jFileMenu: JMenu*
- *jOpenFileMenuItem: JMenuItem*
- *jExitMenuItem: JMenuItem*

- *jCurrentMediaScrollPane: JScrollPane*
- *jCurrentMedia: Jlist<String>*
- *jPlayButton: JButton*
- *jStopButton: JButton*
- *jSeekSlider: JSlider*
- *jShowPlaylistButton: JButton*

- *setModuleList(File moduleFile)*
- *setFileInfo(File file)*
- + *getInstance():GuiModel*

ConcreteGuiModel

- *guiInstance: GuiModel*
- jGuiFrame: JFrame
- jMenuBar: JMenuBar
- jModules: JList<File>
- jModuleMenu: JMenu
- jAddModuleMenuItem: JMenuItem
- jFileMenu: JMenu
- jOpenFileMenuItem: JMenuItem
- jExitMenuItem: JMenuItem
- jCurrentMediaScrollPane: JScrollPane
- jCurrentMedia: Jlist<String>
- jPlayButton: JButton
- jStopButton: JButton
- jSeekSlider: JSlider

- setModuleList(File moduleFile)
- setFileInfo(File file)
- + getInstance():GuiModel

GuiDecorator

- component : GuiModel

- *guiInstance: GuiModel*
- jGuiFrame: JFrame
- jMenuBar: JMenuBar
- jModules: JList<File>
- jModuleMenu: JMenu
- jAddModuleMenuItem: JMenuItem
- jFileMenu: JMenu
- jOpenFileMenuItem: JMenuItem
- jExitMenuItem: JMenuItem
- jCurrentMediaScrollPane: JScrollPane
- jCurrentMedia: Jlist<String>
- jPlayButton: JButton
- jStopButton: JButton
- jSeekSlider: JSlider

- setModuleList(File moduleFile)
- setFileInfo(File file)
- + getInstance():GuiModel

ConcreteGuiSkinDecorator

- *guiInstance: GuiModel*
- jGuiFrame: JFrame
- jMenuBar: JMenuBar
- jModules: JList<File>
- jModuleMenu: JMenu
- jAddModuleMenuItem: JMenuItem
- jFileMenu: JMenu
- jOpenFileMenuItem: JMenuItem
- jExitMenuItem: JMenuItem
- jCurrentMediaScrollPane: JScrollPane
- jCurrentMedia: Jlist<String>
- jPlayButton: JButton
- jStopButton: JButton
- jSeekSlider: JSlider

- skin: SkinAttribute

- setModuleList(File moduleFile)
- setFileInfo(File file)
- setSkin(File skinFile)
- + getInstance():GuiModel

ConcreteGuiPlaylistDecorator

- *guiInstance: GuiModel*
- jGuiFrame: JFrame
- jMenuBar: JMenuBar
- jModules: JList<File>
- jModuleMenu: JMenu
- jAddModuleMenuItem: JMenuItem
- jFileMenu: JMenu
- jOpenFileMenuItem: JMenuItem
- jExitMenuItem: JMenuItem
- jCurrentMediaScrollPane: JScrollPane
- jCurrentMedia: Jlist<String>
- jPlayButton: JButton
- jStopButton: JButton
- jSeekSlider: JSlider

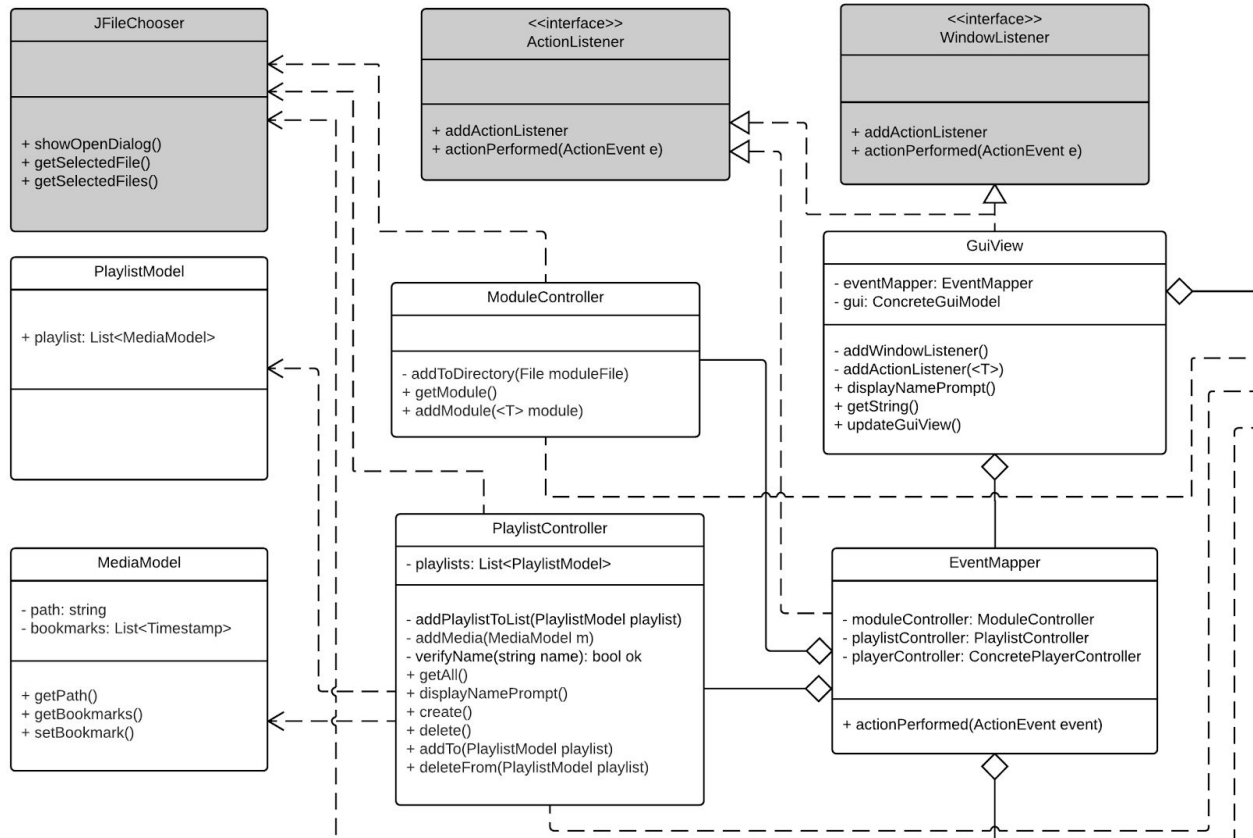- jShowPlaylistButton: JButton
- jPlaylists: JLIst<PlaylistModel>
- jPlaylistPanel: JPanel
- jPlaylistScrollPanel: JScrollPanel
- jCreatePlaylistButton: Button
- jDeletePlaylistButton: JButton
- jAddMediaButton: JButton
- jSelectedPlaylistPanel: JPanel
- jSelectedPlaylistScrollPanel: JScrollPanel
- jDeleteMediaButton: JButton
- jMedias: JList<MediaModel>

- setModuleList(File moduleFile)
- setFileInfo(File file)
- setPlaylistList(PlaylistModel playlist)
- + getInstance():GuiModel

3. PlayerController class where we are implementing the Decorator Pattern.

PlayerStatelModel

- currentGui: GuiModel
- currentMedia: MediaModel
- currentPlaylist: PlaylistModel
- enabledModules: List<File>

+ getPlayerState()

PlayerController

- *state: PlayerStateModel*

- *saveState(PlayerStateModel m)*
- *validFile(string fileName)*
+ *open(string fileName)*
+ *play(MediaModel m)*
+ *stop(MediaModel m)*
+ *seek(MediaModel m)*
+ *bookmark(MediaModel m)*

ConcretePlayerController

- state: PlayerStateModel

- *saveState(PlayerStateModel m)*
- *validFile(string fileName)*
+ *open(string fileName)*
+ *play(MediaModel m)*
+ *stop(MediaModel m)*
+ *seek(MediaModel m)*
+ *bookmark(MediaModel m)*

PlayerControllerDecorator

- component: PlayerController
- state: PlayerStateModel

- *saveState(PlayerStateModel m)*
- *validFile(string fileName)*
+ *open(string fileName)*
+ *play(MediaModel m)*
+ *stop(MediaModel m)*
+ *seek(MediaModel m)*
+ *bookmark(MediaModel m)*