**System Design Report**  
**Application Name**: *Virtual Tour of Maseru Landmarks*  
**Developer**: Teboho Matela  
**Date**: 5/12/2025  
**Language & Framework**: Java with JavaFX

**1. Introduction**

The Virtual Tour application provides an interactive educational experience focused on the city of Maseru. The system allows users to explore landmark locations via an interactive map, view related multimedia (images, audio, and videos), and participate in quizzes. This report outlines the design process and rationale behind each architectural and functional decision made during the system's development.

**2. Requirements Analysis**

**2.1 Functional Requirements**

* Display a map of Maseru with interactive hotspot markers.
* Present landmark-specific information (images, audio, and video).
* Enable users to switch between landmarks.
* Allow media control (play, pause, stop).
* Provide quiz functionality with questions per landmark.
* Maintain consistent UI behavior and user feedback.

**2.2 Non-Functional Requirements**

* Responsive and user-friendly interface.
* Smooth media playback and resource management.
* Maintainable and extensible architecture.

**3. System Architecture and Design**

**3.1 High-Level Design**

The system uses a **JavaFX GUI architecture** composed of multiple panes and control elements layered using StackPane and VBox. The following modules were designed:

* **Map Module**: Displays an interactive map with hotspots.
* **Landmark Module**: Displays images and descriptions of landmarks.
* **Media Module**: Handles audio and video playback.
* **Quiz Module**: Presents multiple-choice questions based on selected landmarks.
* **Gallery Module**: Shows image galleries for each location.

**3.2 Package Structure**

java

CopyEdit

package org.example.virtual\_tour;

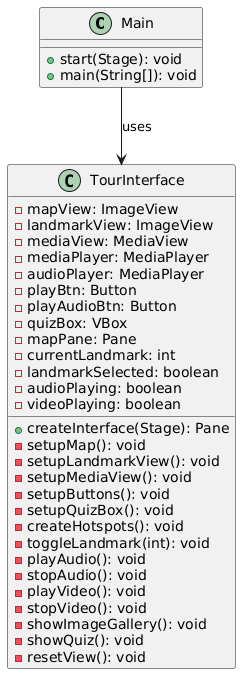
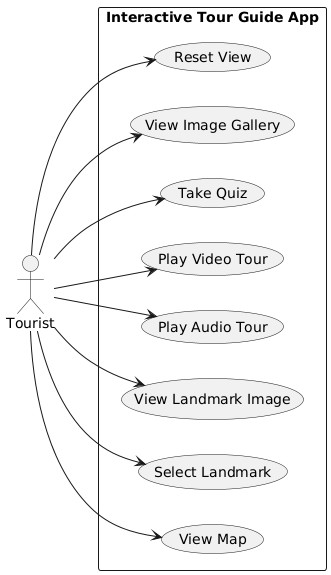
This contains a single class TourInterface responsible for creating and managing all UI elements and logic.

**3.3 Class Design**

* **TourInterface**: Main controller class for all UI components, state management, and event handling.
* Uses JavaFX Stage, Scene, Pane, VBox, HBox, MediaPlayer, and other controls.
* Media is loaded dynamically using getClass().getResource() for platform compatibility.

**3.4 UML diagrams**

Use case diagram class diagram



**4. UI Design and Layout**

**4.1 Layout**

* Top: HBox with the title and a “Back to Map” button.
* Center: StackPane containing the map, landmark image, and video view.
* Bottom: Button panel and quiz panel (VBox).

**4.2 Buttons and Interactions**

Buttons such as “Play Video”, “Play Audio”, “View Images”, and “Go to Quiz” are styled uniformly and dynamically displayed based on context. Custom button styles and hover effects enhance usability.

**4.3 Hotspots**

Each hotspot is a Circle overlaid on the map and associated with:

* An image icon and label (HBox).
* Hover and click effects.
* Interactive behavior that displays the landmark data.

**5. Multimedia Handling**

* **Audio Playback**: Implemented using MediaPlayer. Audio clips are named after the landmark and stored in /audios/.
* **Video Segments**: A single video file (maseru\_video.mp4) is segmented using timecodes for each landmark, ensuring efficient resource use.
* **Image Gallery**: Images are loaded sequentially from the /images/landmark\_name/ directory and navigated using "Next" and "Previous" buttons.

**6. Quiz Functionality**

The quiz system is initialized with pre-defined data in a 2D array:

java

CopyEdit

String[][] quizData = {

{"Question", "Option1", "Option2", "Option3", "CorrectIndex"},

};

* Displayed dynamically based on the selected landmark.
* Provides user feedback on correctness.

**7. State Management**

The application maintains and updates several boolean flags and index trackers:

* currentLandmark: Tracks the selected hotspot.
* landmarkSelected, videoPlaying, audioPlaying: Maintain state for media control.
* UI is updated conditionally based on the state flags to prevent errors and improve experience.

**8. Error Handling and Alerts**

* Fallbacks for missing resources (map, images, media).
* User-friendly error messages via alert boxes (Alert dialogs).
* MediaPlayer objects are explicitly disposed of to prevent memory leaks.

**9. Conclusion**

The Virtual Tour application leverages JavaFX’s capabilities to provide a rich and interactive multimedia experience. Its modular design promotes maintainability, while thoughtful UI and state handling ensure a smooth user journey. This report outlines a complete system design from concept to implementation, showcasing best practices in GUI development and user experience integration.