Project 2: Custom Video Player

---

1. Introduction

This project implements a **custom video player** built with **HTML, CSS, and JavaScript**. Unlike the default browser video player, this version provides a **modern, responsive, and interactive user interface**, offering better user control and design flexibility.

The player allows users to:

- Upload a local video file.
- Play, pause, and seek within the video.
- Adjust volume with a custom slider.
- Track progress via a progress bar.
- Switch to fullscreen mode.
- Use **keyboard shortcuts** for faster interaction.

---

2. Objectives

- To design a visually appealing video player interface.
- To implement custom playback controls.
- To enhance user experience with smooth animations.
- To support keyboard shortcuts for accessibility.

---

3. Tools & Technologies

- **HTML5** → Semantic structure & `<video>` element.

- **CSS3** → Styling, responsive layout, and transitions.
- **Vanilla JavaScript (ES6)** → Playback logic, events, and interactivity.

---

4. Features & Functionalities

1. **Video Upload**
    - Drag-and-drop style upload section.
    - Displays file name and size dynamically.
2. **Playback Controls**
    - **Play / Pause button** (with toggle).
    - **Progress bar** with click-and-drag seek.
    - **Time display** (`current time / total duration`).
3. **Volume Control**
    - Custom slider to adjust audio.
    - Visual feedback via colored volume bar.
4. **Fullscreen Mode**
    - One-click fullscreen toggle.
    - Keyboard shortcut (`f`) supported.
5. **Keyboard Shortcuts**
    - `Space` or `K` → Play/Pause.
    - `F` → Fullscreen.
    - `M` → Mute/Unmute.
    - `→` / `←` → Seek forward/backward.

    - ↑ / ↓ → Volume control.

6. **Responsive Design**
    - Scales smoothly on desktop, tablet, and mobile.

---

5. Design & Architecture

- **Upload Section** → Initial screen with file input.
- **Player Section** → Video element with custom controls.
- **Controls Overlay** → Hidden until hover for a cleaner look.
- **JS Modules** → Handle events (`play`, `pause`, `timeupdate`, `volumechange`, `fullscreen`, `keyboard`).

---

6. Implementation Details

- **HTML**:
    - File input for video upload.
    - `<video>` tag with custom overlay controls.
- **CSS**:
    - Gradient background, rounded containers, and modern UI.
    - Hover-triggered control panel.
- **JavaScript**:
    - Event listeners for buttons, progress, and volume.
    - Dynamic updates for time and file info.
    - Custom fullscreen and keyboard control handling.

---

7. Challenges & Solutions

- **Challenge**: Implementing custom progress bar seek.
  - **Solution**: Used `mousemove` + `mousedown` listeners for drag functionality.
- **Challenge**: Synchronizing time display with progress.
  - **Solution**: Added `timeupdate` event and custom `formatTime()` function.
- **Challenge**: Making controls responsive.
  - **Solution**: Applied media queries and flexible layouts.

---

8. Testing

- ✅ Upload tested with MP4, WebM, and OGG.
- ✅ Controls tested in Chrome, Firefox, and Edge.
- ✅ Responsive testing across desktop, tablet, and mobile.
- ✅ Keyboard controls verified.

---

9. Future Scope

- Add playlist support.
- Add playback speed control.
- Add picture-in-picture (PiP) mode.
- Save playback state (resume from last position).

---

10. Conclusion

This project successfully demonstrates how to create a **fully custom video player** with a polished UI and functional playback controls using **HTML, CSS, and JavaScript**. It enhances usability with responsive design, intuitive controls, and accessibility features like keyboard shortcuts.