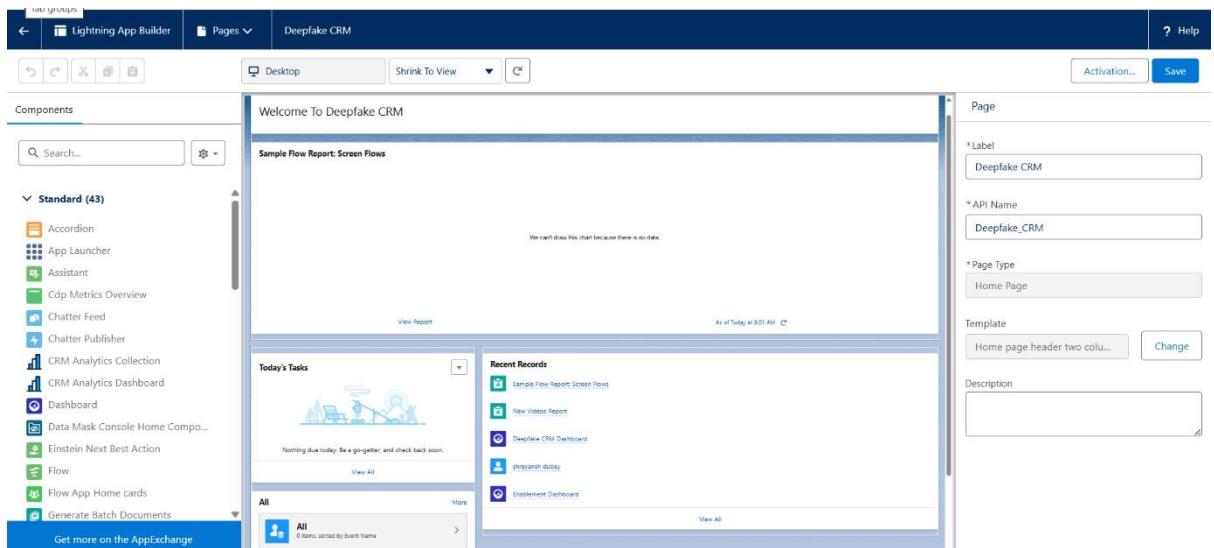


Phase 6: User Interface Development - Salesforce

Project: Deepfake CRM: Intelligent Salesforce Integration Project

1. Lightning App Builder

- Drag-and-drop tool to create **App, Home, or Record pages**.
- Types of pages:
 - **App Page:** Standalone page in App Launcher.
 - **Home Page:** Landing page with dashboards, reports, and components.
 - **Record Page:** Custom layout for specific objects (e.g., Video).
- Features:
 - Dynamic visibility
 - Tabs for organization
 - Custom LWC integration
- **Tip:** Use tabs template for Record Pages to organize Details, Related Lists, and custom components.



2. Record Pages

- Customizes layout for **object records**.
- Include:
 - **Highlights Panel:** Key fields at top.
 - **Tabs:** Organize Details, Related Lists, and Custom LWCs.
 - **Custom Components:** Video Player, Alerts, Detection status.
- Assign pages to **profiles or apps**.

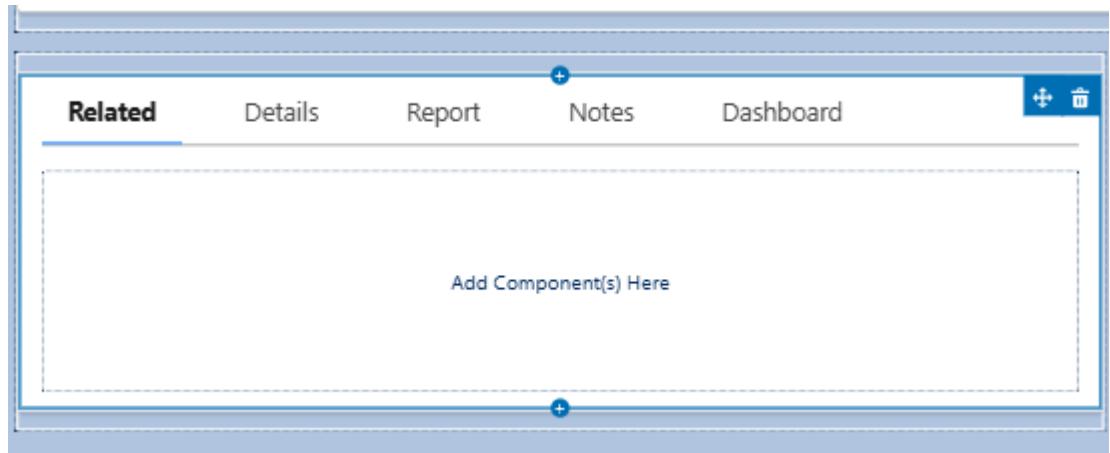
The screenshot shows the Salesforce Page Layout Editor interface. On the left, there's a preview area with a blue background. At the top of the preview, there are two sections labeled 'Source URL' and 'Uploaded By'. Below these are two 'Title' fields. Underneath these sections is a tab bar with 'Details', 'Related', 'Report', and 'Notes'. The 'Related' tab is selected. To the right of the tabs is a component titled 'Activity History (0)' with a 'Log a Call' and 'Email' button. Below the tabs and activity history is a placeholder box with the text 'Add Component(s) Here'. On the right side of the editor, there are several configuration panels:

- * Parent Record:** A search bar with the text 'Use This Video' and a magnifying glass icon.
- * Related List:** A search bar with the text 'Activity History' and a magnifying glass icon.
- Related List Type:** A dropdown menu set to 'Default'.

Below these panels is a note: 'Include the related list you want to use in the page layout for your users. If using a parent record, update the parent record's page layout. Otherwise, users can't see the related list.' At the bottom right, there are buttons for 'Assign Page Layouts', 'Project Layout (previewed)', and 'View all layouts'.

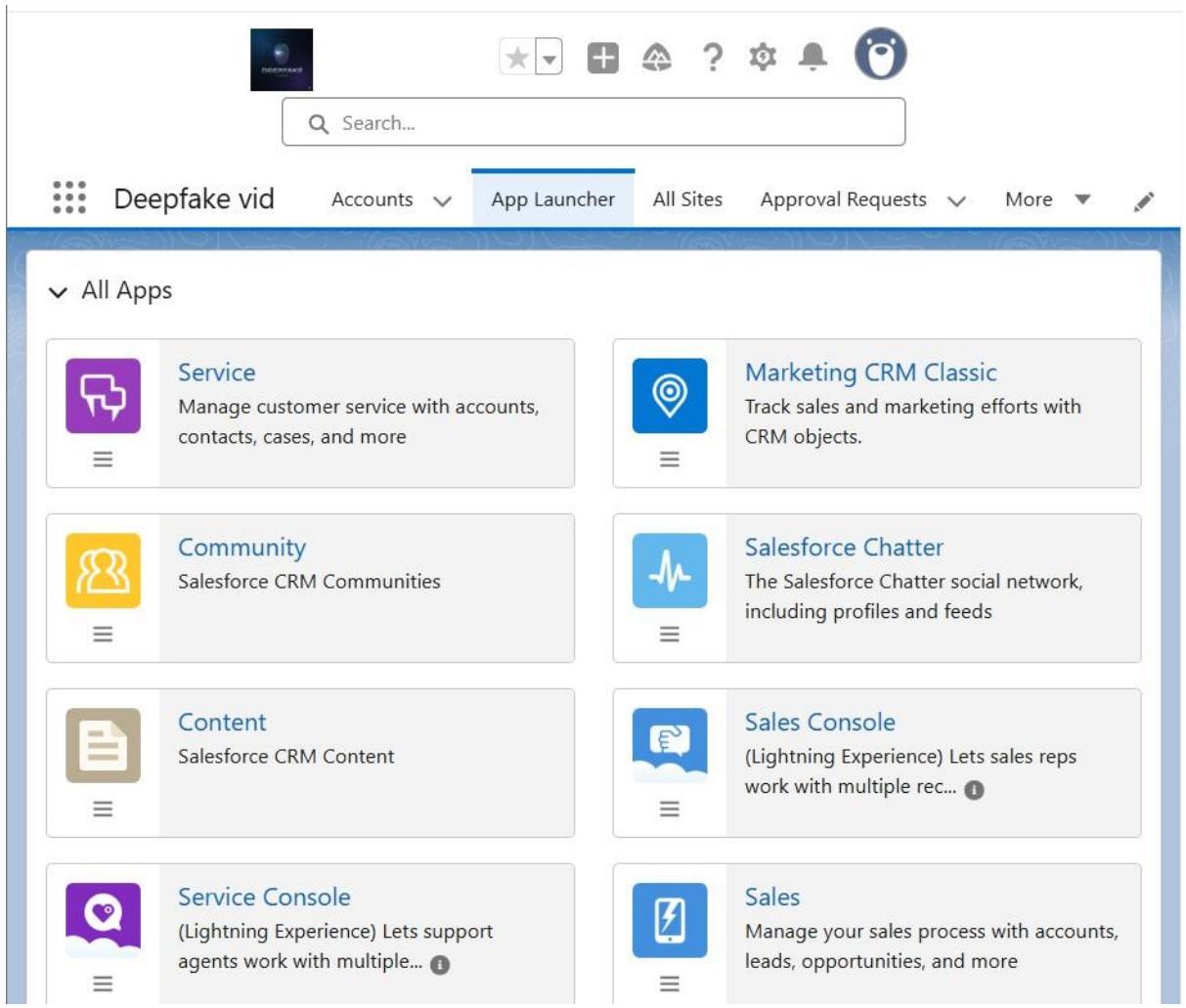
3. Tabs

- Organize information efficiently.
- Examples for Video object:
 - Details (Record Fields)
 - Related (Alerts, Comments)
 - Video Player (LWC)
 - Detection/Info (Custom LWC or Rich Text)
 - Report
 - Notes
 - Dashboard



4. Home Page Layouts

- Custom landing page for users.
- Components:
 - Reports, Dashboards
 - Announcements
 - Custom LWC (e.g., Recent Videos)
- Can assign different layouts for different **profiles**.



5. Utility Bar

- Persistent footer in Lightning apps.
- Quick access tools for productivity.
- Examples for Video project:
 - Video Uploader (LWC)
 - Recent Videos
 - Alerts from Detection System
- Add via **App Manager → Utility Bar → Add Utility**.

The screenshot shows the 'Utility Items (Desktop Only)' configuration screen. On the left, a sidebar lists 'App Settings' with options like 'App Details & Branding' and 'App Options'. Below that is a section titled 'Utility Items (Desktop Only)' which contains 'Navigation Items' and 'User Profiles'. The main area is titled 'Utility Items (Desktop Only)' and includes a sub-instruction: 'Give your users quick access to productivity tools and add background utility items to your app.' A 'Add Utility Item' button is at the top left. To its right are 'Utility Bar Alignment' (set to 'Default') and 'Remove' buttons. A list of utility items is shown, each with an icon and a label: 'Omni-Channel', 'Notes', 'History' (selected), 'To Do List', 'List View', 'History', 'My Appointments', and 'Report Chart'. To the right of the list are 'Properties' for the selected item ('History'): 'Label' (set to 'History'), 'Icon' (set to 'clock'), 'Panel Width' (set to '340'), 'Panel Height' (set to '480'), and a checked 'Start automatically' checkbox. At the bottom are 'Cancel' and 'Save' buttons.

6. Lightning Web Components (LWC)

- Modern framework based on **JavaScript ES6+**.
- Files:
 - .html → UI
 - .js → Logic
 - .css → Styling (optional)
- Can be added to Record Pages, App Pages, and Utility Bar.
- Example: Video Player, Video Alerts, Upload Widget.

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left displays the project structure under 'DEEP...', including '.husky', '.sfdx', '.vscode', 'config', 'force-app/main/def...', 'applications', 'aura', 'classes', 'contentassets', 'flexipages', 'layouts', 'lwc/videoPlayer' (which is expanded), '_tests_', 'videoPlayer.html' (selected), 'videoPlayer.js', 'videoPlayer.js-meta.xml', 'objects', 'permissionsets', 'staticresources', 'tabs', 'triggers', 'scripts/apex/hello.apex', 'scripts/sql/account.sql', '.forceignore', '.gitignore', '.prettierignore', 'TIMELINE', and 'OUTLINE'. The main editor area shows the 'videoPlayer.html' file with the following code:

```
1 <template>
2   <lightning-card title="Video Player">
3     <video width="100%" controls>
4       <source src={videoUrl} type="video/mp4" />
5     </video>
6   </lightning-card>
7 </template>
```

The bottom right corner of the editor shows the status bar with 'Salesforce CLI' and other icons.

The bottom right panel shows the 'OUTPUT' tab with the following logs:

```
PROBLEMS 1 OUTPUT ... Salesforce CLI Filter
target dir = c:\Users\Shreyansh Dubey\Desktop\TCS
20:50:15.335 Finished SFDX: Create Lightning Web Component
```

The screenshot shows the Visual Studio Code (VS Code) interface with a project titled "Deepfake CRM". The Explorer sidebar on the left lists various files and folders, including ".hus", ".sfdx", ".vscode", "config", "force-app\main\def...", "applications", "aura", "classes", "contentassets", "flexipages", "layouts", "lwc\videoPlayer", "_tests_", "videoPlayer.html", "videoPlayer.js", "videoPlayer.js-meta.xml", "objects", "permissionsets", "staticresources", "tabs", "triggers", "scripts", "apex", "hello.apex", "soql", "account.soql", ".forceignore", ".gitignore", ".prettierignore", "TIMELINE", and "OUTLINE". The "videoPlayer.js" file is currently open in the editor tab, displaying LWC code. The code imports LightningElement, api, and wire from 'lwc', and getRecord from 'lightning/uiRecordApi'. It defines a class VideoPlayer extending LightningElement and uses @wire to fetch a record. The "OUTPUT" tab at the bottom shows the command "target dir = c:\Users\Shreyansh Dubey\Desktop\TCS" and the message "20:50:15.335 Finished SFDX: Create Lightning Web Component".

```
1 import { LightningElement, api, wire } from 'lwc';
2 import { getRecord } from 'lightning/uiRecordApi';
3 import VIDEO_URL_FIELD from '@salesforce/schema/Video_
4
5 export default class VideoPlayer extends LightningElem
6   @api recordId; // Automatically set when added to
7   videoUrl;
8
9   @wire(getRecord, { recordId: '$recordId', fields:
10     wiredRecord({ error, data }) {
11       if (data) {
12         this.videoUrl = data.fields.Source_URL_c.
13       }
14       if (error) {
15         console.error(error);
16       }
17     }
18   }
19 }
```

PROBLEMS 1 OUTPUT ... Salesforce CLI Filter

```
target dir = c:\Users\Shreyansh Dubey\Desktop\TCS
20:50:15.335 Finished SFDX: Create Lightning Web Component
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "DEEPF...". Key items include ".husky", ".sfdx", ".vscode", "config", "force-app\main\def...", "applications", "aura", "classes", "contentassets", "flexipages", "layouts", "lwc\videoPlayer", "_tests_", "videoPlayer.html", "videoPlayer.js", and "videoPlayer.js-met...".
- EDITOR**: Displays the "videoPlayer.js-meta.xml" file content:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>false</isExposed>
</LightningComponentBundle>
```
- OUTPUT**: Shows the Salesforce CLI output:

```
target dir = c:\Users\Shreyansh Dubey\Desktop\TCS
20:50:15.335 Finished SFDX: Create Lightning Web Component
```

7. Apex with LWC

- Apex handles **server-side logic**.
- **Read-only data:** Use `@AuraEnabled(cacheable=true)` with wire adapter.
- **User actions:** Use **imperative Apex calls** for buttons/events.
- Example:
 - Fetch Video details

- Update Video status
 - Analyze video
-

8. Events in LWC

- **Custom events:** Child → Parent communication.
 - **Pub/Sub / Application events:** Component → unrelated component communication.
 - Use case: Update Detection Status across multiple components when video is analyzed.
-

G. Wire Adapters

- Reactive data fetching.
 - Examples:
 - getRecord
 - getObjectInfo
 - getPicklistValues
 - Automatically updates component when data changes.
-

10. Imperative Apex Calls

- Explicit Apex call triggered by user action.
 - Example:
 - Analyze video
 - Approve video
 - Flexible control over when server-side code runs.
-

11. Navigation Service

- Use NavigationMixin in LWC.
- Programmatic navigation:
 - Standard pages (record, object list)

- URLs
 - Lightning pages
 - Example: Redirect user after saving or analyzing a video.
-