# FragPunk Games : "Your Story, Our Craft."

An Angular Application

Submitted by

Jashwanth E M

# Application Details

**Purpose:** FragPunk is a basic web application designed to showcase games developed by FragPunk, a fictional game development company inspired by industry leaders like Rockstar. It provides a user-friendly interface to browse game titles, view game details, and submit feedback. The application is a prototype with static elements (e.g., buttons) and limited functionality, focusing on demonstrating Angular concepts for educational purposes.

**Key Features:**

**1.Homepage:**

- A welcome section with the tagline: "Experience the next generation of gaming with FragPunk’s innovative titles.
- Navigation links to key sections: Games, Add Game, and Feedback.
- A static "Explore Games" button to direct users to the game list.

# Application Details

**2) Games Section:**

- Displays a static list of games fetched from a local JSON file (`api.json`).
- Each game card includes: - Game title and description. - Genre, release date. - A static "View Details" button (placeholder, no functionality).
- Sample games: - *FragPunk: Arena* ($79, FPS, Release: Dec 15, 2024, "Fast-paced first-person shooter"). - *CyberFrag* ($89, RPG, Release: Mar 20, 2025, "Open-world cyberpunk adventure").

**3) Add Game Section:**

- A form to input new game details (title, genre, release date, description).
- On submission, logs data to the console (no backend integration).
- Includes basic validation (e.g., required fields, minimum description length).
- Filter Options based on Games’s Genre to filter games(e.g.,Action , Adventure ,Racing , Casual , RPG , Strategy , etc..).

# Application Details

**4)News Section:**

- A static page displaying recent company updates and game-related announcements.
- Sample news items:
- "FragPunk: Arena Beta Testing Begins!" (Dec 1, 2024).
- "CyberFrag Wins Best RPG Concept Award" (Feb 15, 2025)
- No interactive features (static text display).

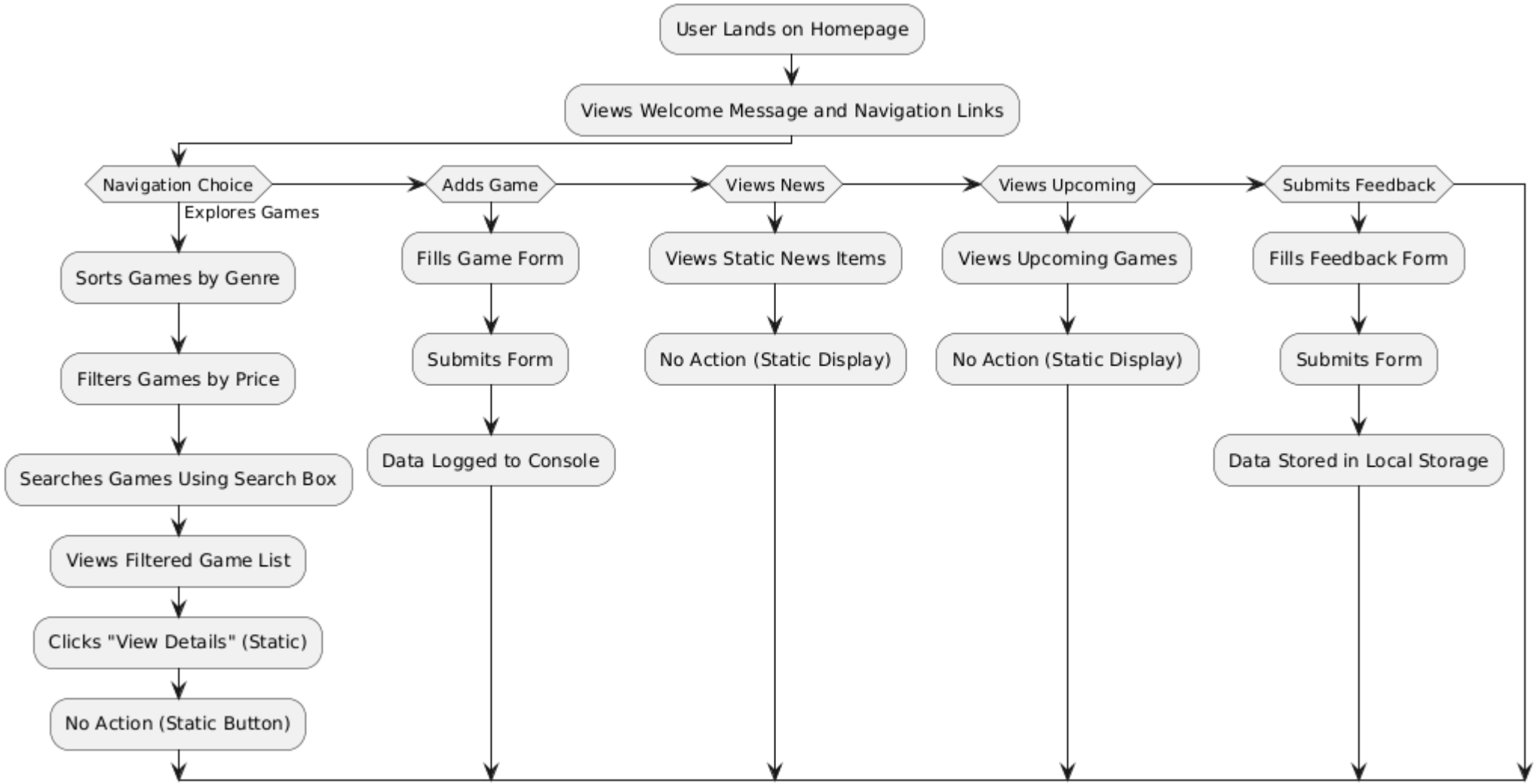
**5)Upcoming Section:**

- A static page showcasing games in development or soon to be released.
- Each upcoming game card includes: Game title, genre, and expected release date. - A brief teaser description. Sample upcoming games: **FragPunk: Skies** (Flight Simulator, Q3 2025, "Soar through dystopian skies"). **Neon Blade** (Action, Q4 2025, "Hack and slash in a neon-lit world").
- No functionality for pre-orders or notifications.

**6) Feedback Section:**

- A functional feedback form where users can input their name, email, rating (1-5), and comments.
- On submission, form data is stored in the browser's local storage (no backend integration).
- Purpose: Collect user feedback to improve the platform.

# Workflow Diagram



# Workflow Diagram

**User Initiation:** User begins by landing on the FragPunk homepage, marking the starting point of the application.

**Welcome and Navigation:**

- User views the welcome message and navigation links (Games, Add Game, Feedback).
- Navigation choice determines the subsequent user action based on the selected section.

**Exploring Games**:

- User selects the "Games" option from the navigation or clicks "Explore Games."
- Views a list of games fetched from api.json.
- Clicks the "View Details" button on a game card (static and non-functional).
- No action is triggered due to the button being a placeholder.
- User sorts games by genre to organize the list.

# Workflow Diagram

**Adding a Game**:

- User selects the "Add Game" option from the navigation.
- Fills the game form with details (title, genre, release date, description).
- Submits the form, triggering a console log of the data.
- No persistent storage or backend action occurs (prototype limitation).

**Viewing News**:

- User selects the "News" option from the navigation.
- User views static news items (e.g., "FragPunk: Arena Beta Begins!").
- User scrolls through news archive (static content).
- No action is available (static display).

**Submitting Feedback**:

- User selects the "Feedback" option from the navigation.
- Fills the feedback form with details (name, email, rating, comments).
- Submits the form, storing data in local storage.
- Provides basic interactivity without backend support.

# Angular Concepts Used

**1. Angular Modules**:

- The application uses Angular modules (NgModule).
- AppModule: The root module that bootstraps the application and imports necessary modules (e.g., HttpClientModule, FormsModule).

**2. Components**:

- **AppComponent**: Root component rendering the main layout (header, navigation, router outlet).
- **GameListComponent**: Parent component displaying the list of games.
- **GameCardComponent**: Child component rendering individual game details.
- **GameFormComponent**: Manages the add game form with reactive form functionality
- **PlansPricingComponent:** Shows the subscription plans (static display).
- **AboutUsComponent:** Displays the "About Us" text (static).
- **FeedbackComponent:** Manages the feedback form with local storage functionality.

# Angular Concepts Used

**3. Parent-Child Components**:

- **GameListComponent** (parent) passes game data to **GameCardComponent** (child) using @Input().
- Example:
  <app-game-card [game]="game"></app-game-card> in game-list.component.html.

**4. Templates and Data Binding**:

- **Templates**:Each component has an HTML template (e.g., game-card.component.html for game details).
- **Interpolation ({{}})**: Displays dynamic data (e.g., {{ game.title }}).
- **Property Binding ([property])**: Binds data to attributes (e.g., [game]="game").
- **Event Binding ((event))**: Handles form submission (e.g., (ngSubmit)="onSubmit()").

**5. Directives**:

**Structural Directives**:

- ***ngFor**: Loops through games (e.g., *ngFor="let game of games").
- ***ngIf**: Conditionally shows loading state (e.g., *ngIf="isLoading").

**Attribute Directives**:

- **[ngClass]:** Applies styles conditionally (e.g., [ngClass]="{'featured': game.id === 1}").

# Angular Concepts Used

**6. Services**:

- **GameService**: Fetches game data from api.json using HttpClient.
- Injected into GameListComponent via dependency injection.

**7. Routing**:

- **AppRoutingModule:** Manages navigation between sections.
- **Routes:**
  - / : GameListComponent
  - /add-game: GameFormComponent
- **RouterLink**: Used in navigation (e.g., <a routerLink="/">Games</a>).
- **RouterOutlet**: Renders the current route’s component (e.g., <router-outlet>).

8. **Forms**:

- **Reactive Forms**: Used in GameFormComponent for adding games.
- **Example:** Form controls with validators (e.g., title: ['', Validators.required]).
- Submission logs to console.

# Angular Concepts Used

**9. Observables**:

- Used in GameService to handle HTTP requests (e.g., getGames(): Observable<Game[]>).
- Subscribed in GameListComponent to fetch and display games.

**8. API Calls**:

- HttpClient fetches static data from assets/api.json.
- Example: this.http.get<Game[]>(this.apiUrl).

**9. Pipes**:

- **Custom ReleaseDatePipe**: Formats release dates (e.g., {{ game.releaseDate | releaseDate }}).

**10. Local Storage**:

- Could be added to GameFormComponent to store submitted game data locally.
- **Example:**

```
onSubmit() {
if (this.gameForm.valid) {
localStorage.setItem('newGame', JSON.stringify(this.gameForm.value));
  }
 }
```

# Angular Concepts Used

**13. Styling**:

- Component-specific CSS (e.g., game-card.component.css for card styling).
- Global styles in styles.css for consistent layout.

**14. File Structure**:

- src/app/components/: game-list, game-card, game-form.
- src/app/services/: game.service.ts.
- src/app/models/: game.model.ts.
- src/app/pipes/: release-date.pipe.ts.
- app-routing.module.ts, app.module.ts, app.component.*.

**12. Future Enhancements with Angular Concepts:**

- **HTTP Client:** Connect to a real backend API for dynamic game data.
- **Reactive Forms Enhancements:** Add complex validation (e.g., unique titles).
- **State Management:** Use NgRx for managing game list state.
- **Dynamic Features:** Implement game details view or user authentication.
- **Lazy Loading:** Load components on demand for better performance.