### Frontend Development

\*\*Technology Stack:\*\*

- \*\*Framework:\*\* Flutter Web for a unified development experience and to leverage Dart across both frontend and backend (if needed).

- \*\*State Management:\*\* Built-in `setState` for simplicity, suitable for managing the UI state based on user interactions and API responses.

- \*\*UI Components:\*\*

- Splash screen with a timer.

- Buttons with icons (e.g., movie reel for random recommendations).

- Input fields and selectors for search filters (genre, type, length, rating).

- Toggle switch for light and dark mode.

- Snackbar and other feedback mechanisms for error handling.

\*\*Design:\*\*

- Responsive design to accommodate various screen sizes and devices.

- Implement a theme system for light and dark modes.

- Use placeholders for images or trailers when not available.

### Backend/Integration

\*\*Data Source:\*\*

- \*\*APIs:\*\* Use movie database APIs like TMDB for movie/show data. For streaming platform information, explore available options like JustWatch API (considering legal and availability constraints).

\*\*Local Storage:\*\*

- \*\*Package:\*\* `shared\_preferences` for persisting user preferences and watchlist data locally.

### Error Handling

- Implement robust error handling throughout the app, utilizing Flutter’s built-in widgets like Snackbar for user feedback on errors (e.g., network issues, data fetching errors).

### Architecture

- \*\*Pattern:\*\* MVC (Model-View-Controller) to separate business logic, UI, and data models, ensuring clean code organization and maintainability.

### Development Tools and Practices

- \*\*Version Control:\*\* GitHub for code repository and version control.

- \*\*Debugging:\*\* Use Dart’s debugging tools and print statements for console debugging.

- \*\*Documentation:\*\* Ensure major functions and complex logic are well-commented to aid in readability and maintenance.

### Deployment

- \*\*Platform:\*\* Netlify for hosting the Flutter Web app.

- \*\*CI/CD:\*\* Utilize Netlify CLI for continuous integration and deployment, automating the deployment process from GitHub commits to production.

### Security and Compliance

- Ensure API keys and sensitive information are not exposed in the frontend code. Use environment variables and Netlify Functions for secure API requests if necessary.

### Performance and Optimization

- Optimize image and video content for web delivery to ensure fast load times.

- Implement lazy loading for images and videos to improve performance.

### Accessibility

- Ensure the app is accessible, considering color contrast for light and dark modes and providing alt text for images and descriptive titles for videos.

These technical specifications provide a comprehensive framework for developing the movie/show recommendation web app, focusing on user experience, performance, and maintainability.