

REACT



REACT AND THE ART OF GAMIFICATION

COURTNEY YATTEAU

START

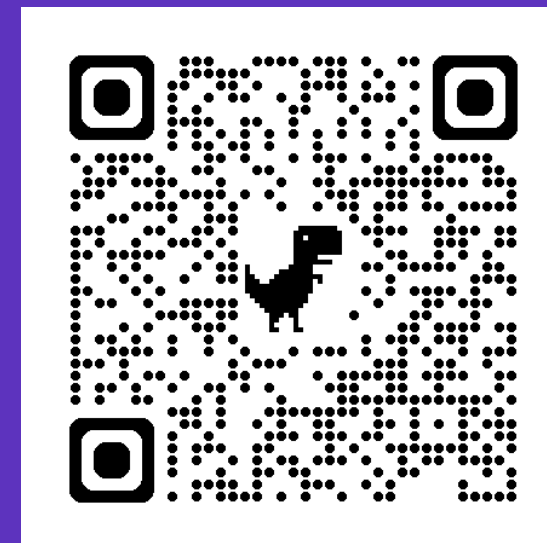


COURTNEY YATTEAU

Developer Advocate, Esri



AGENDA



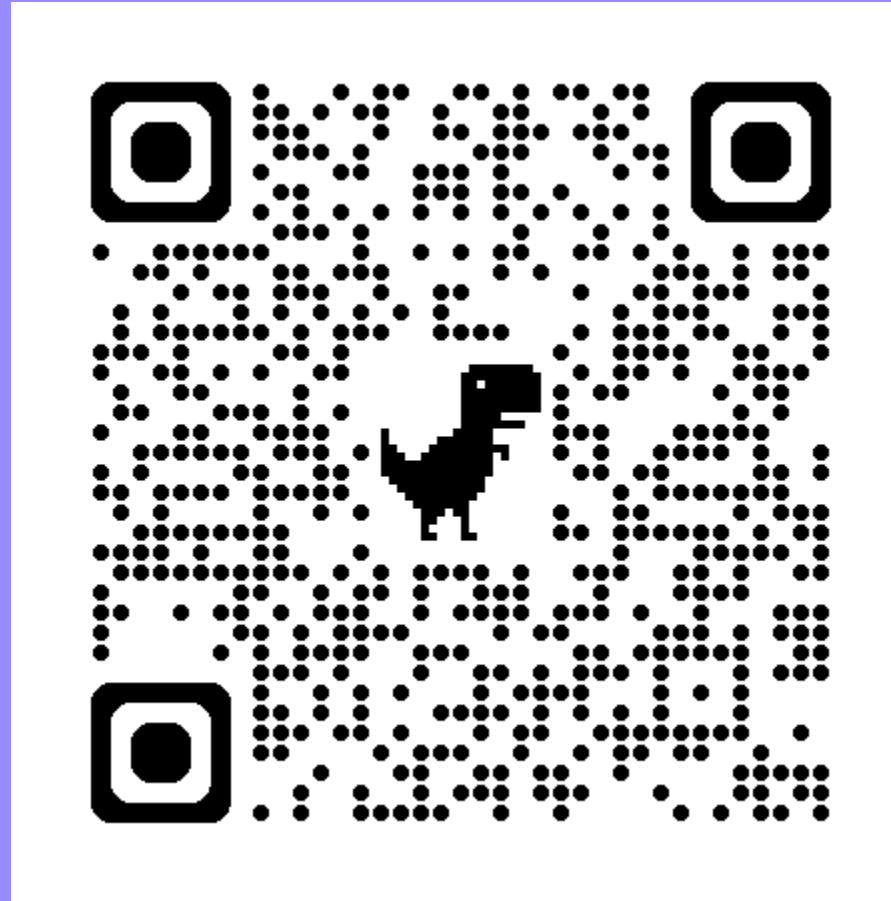
[github.com/cyatteau/
react-gamification-
kcdc2025](https://github.com/cyatteau/react-gamification-kcdc2025)

GAMERS 🎮

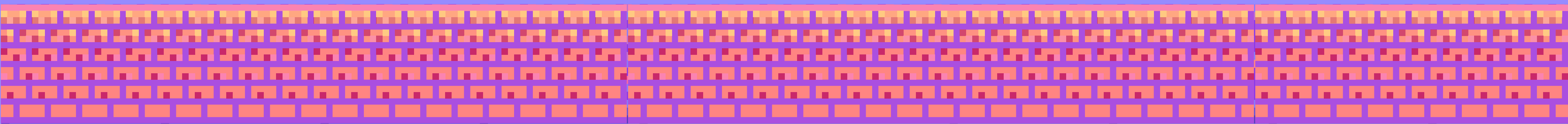
COMPETITIVE 🔥

REWARDS 🏆





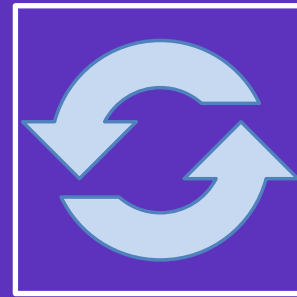
<https://draw-meme-app.vercel.app/>



PLAY BEFORE THEORY



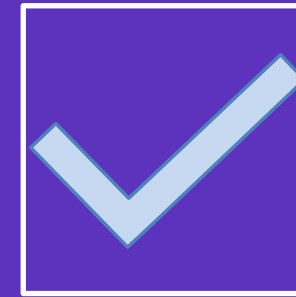
Instant Engagement



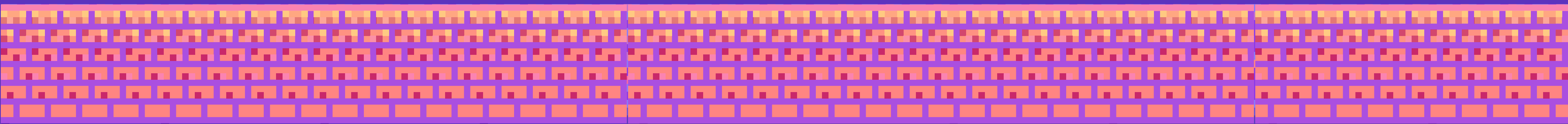
Live Feedback Loop



Shared Experience



Proof in Practice



WHAT IS GAMIFICATION?



“

Gamification is the process of using **game thinking** and **game dynamics** to **engage audiences** and **solve problems**.

Gabe Zichermann

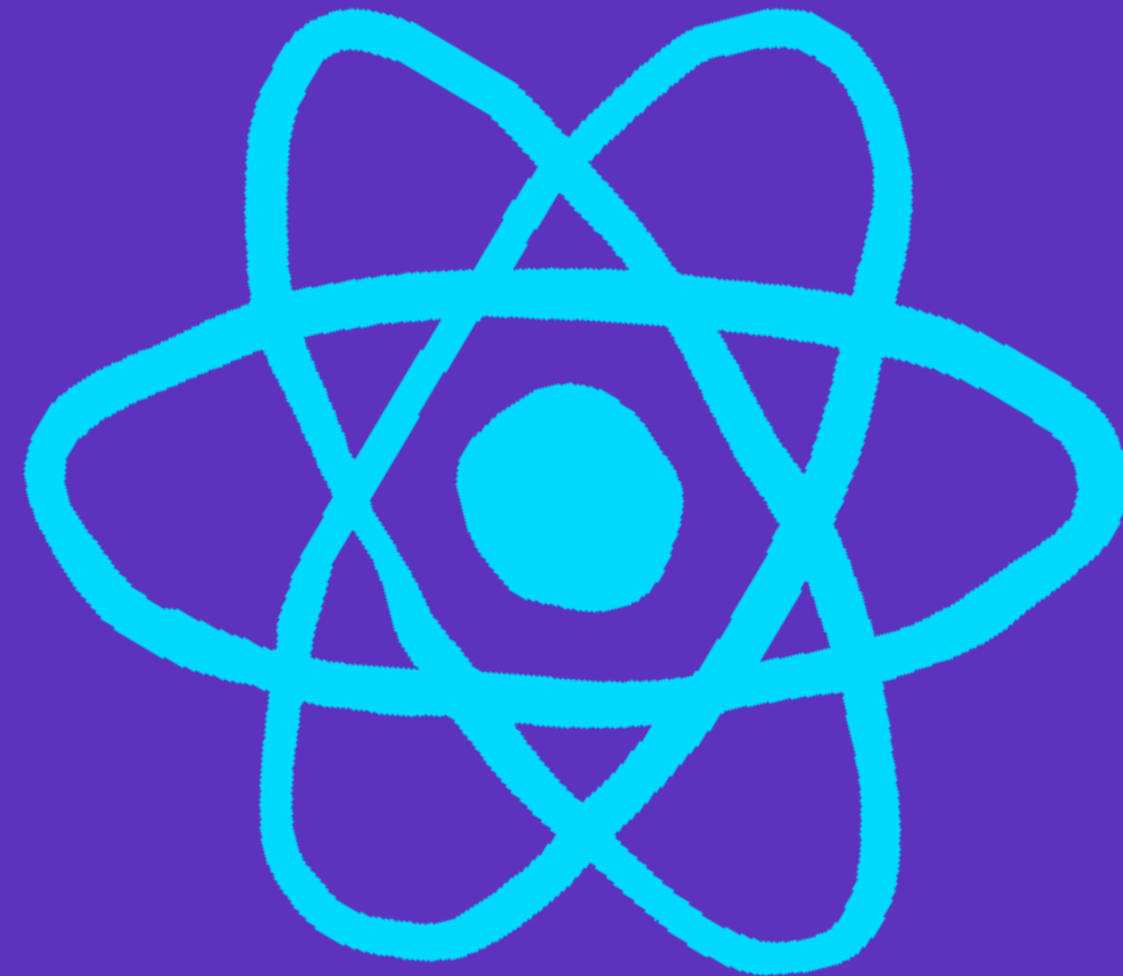
”



WHY GAMIFICATION?



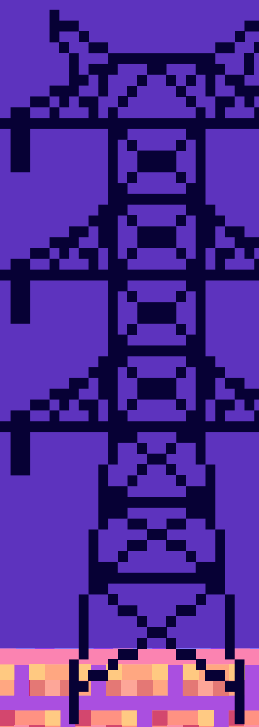
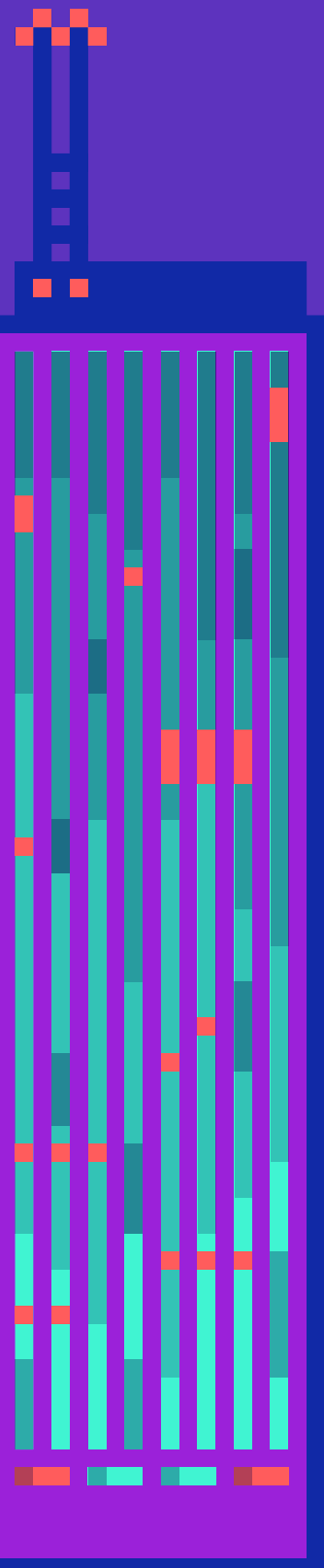
WHAT?



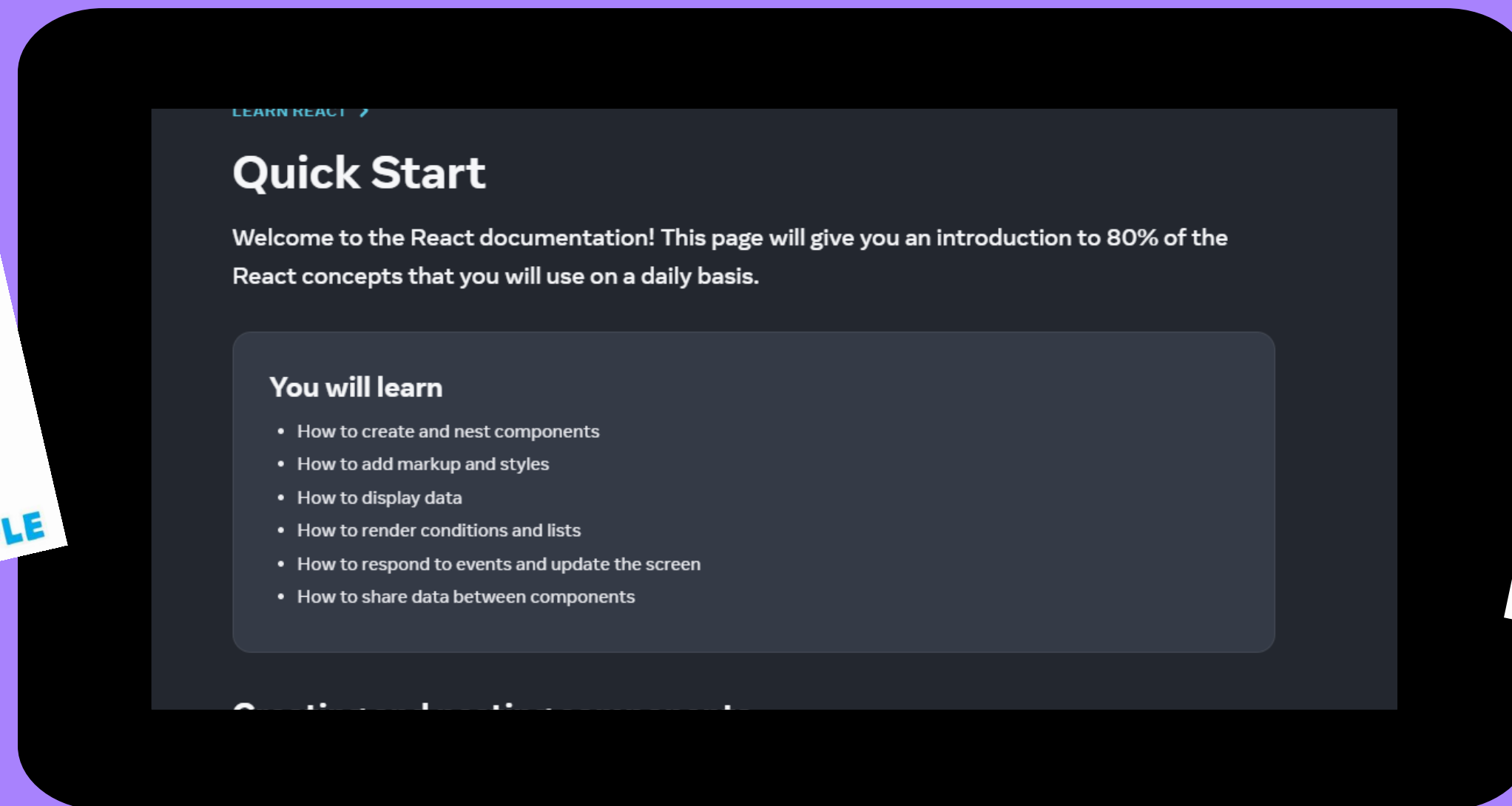
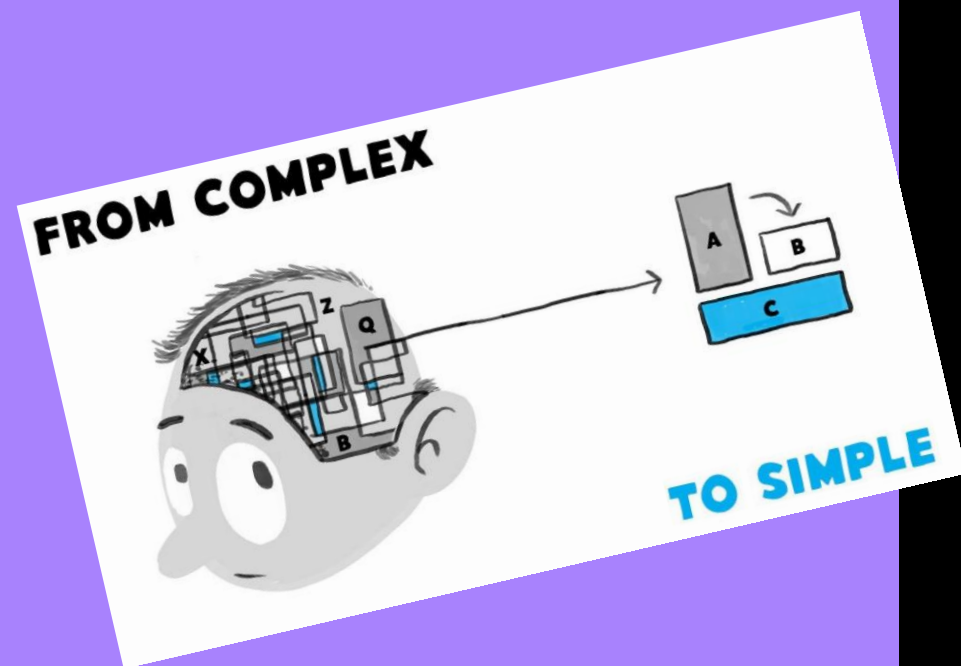
WHY?

HOW?

React JS



GAMIFYING REACT EDUCATION



react.dev/learn

WHAT IS REACT?



JS LIBRARY FOR UI



Declarative



**Component-
based**



Efficient

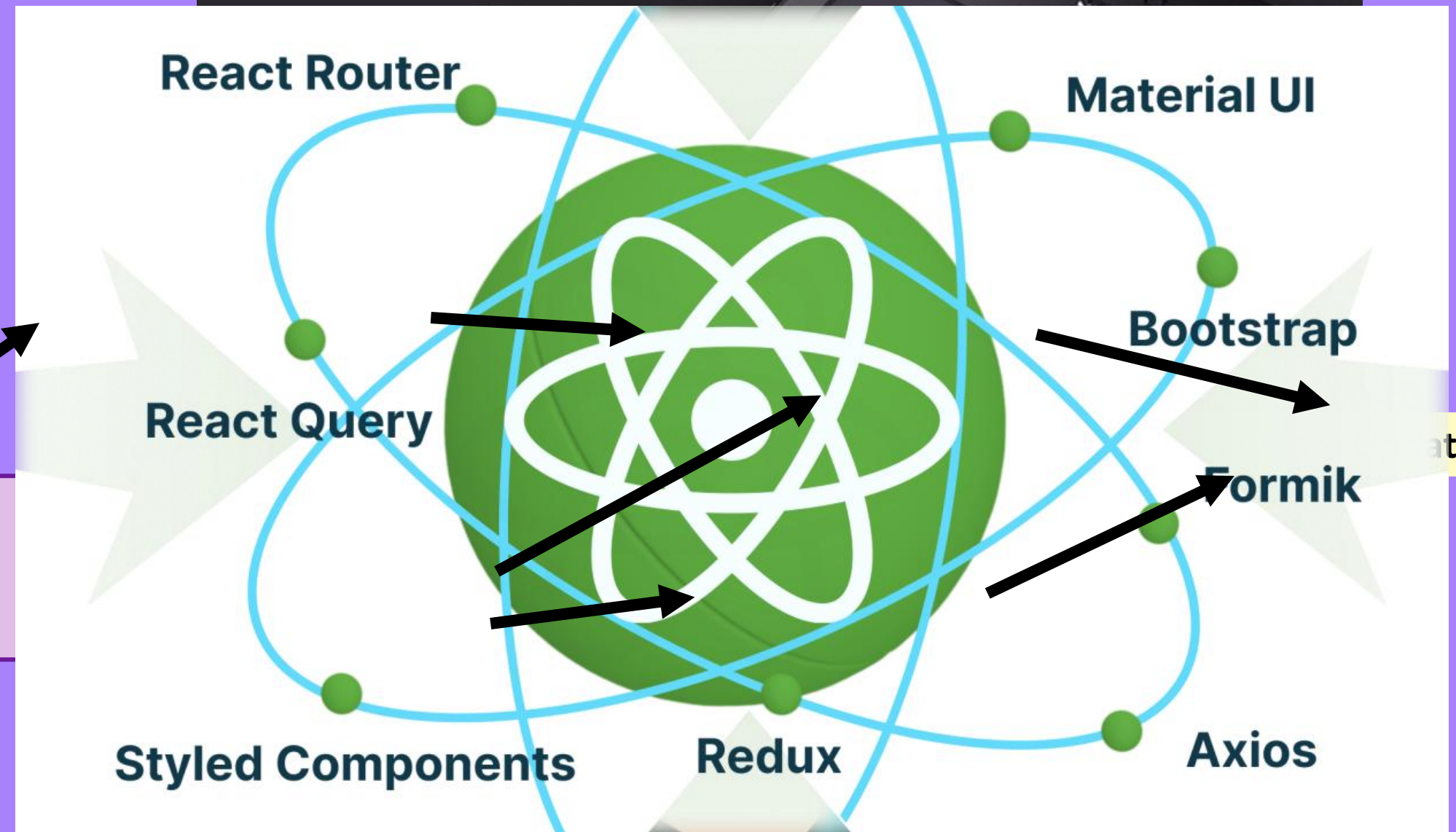


WHY USE REACT?

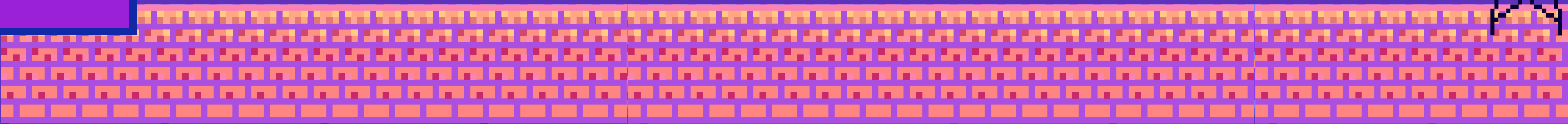
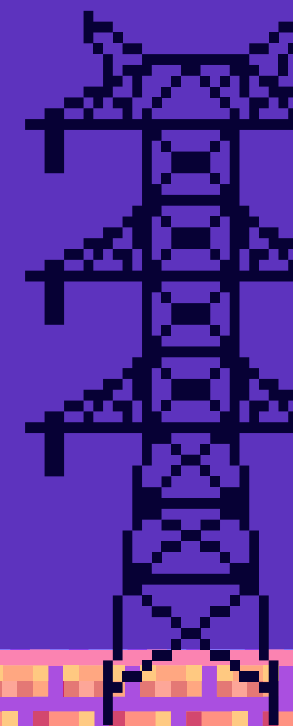
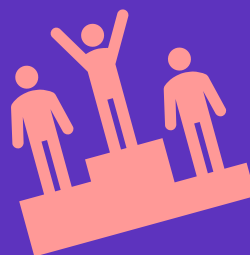
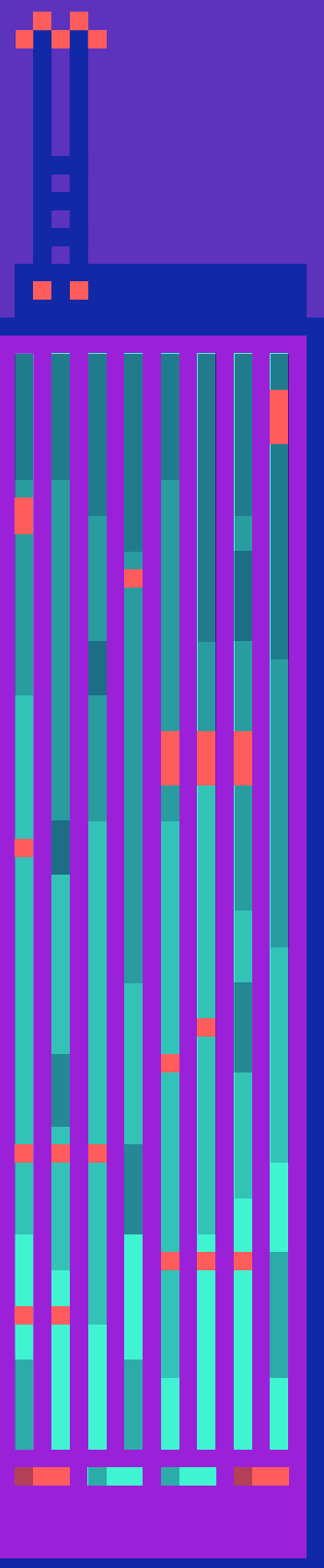
Rich Ecosystem

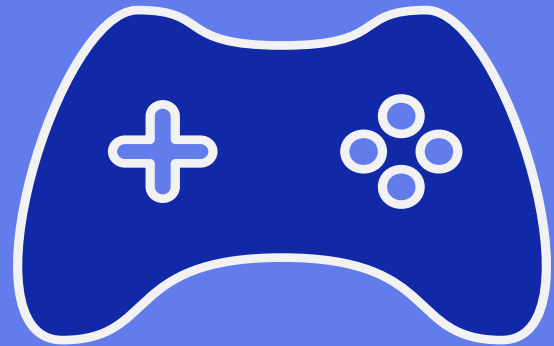
User Action
(e.g., click/tap)

Basic State (useState)



G.A.M.E.S.





G

Gamified UI
Elements

A

M

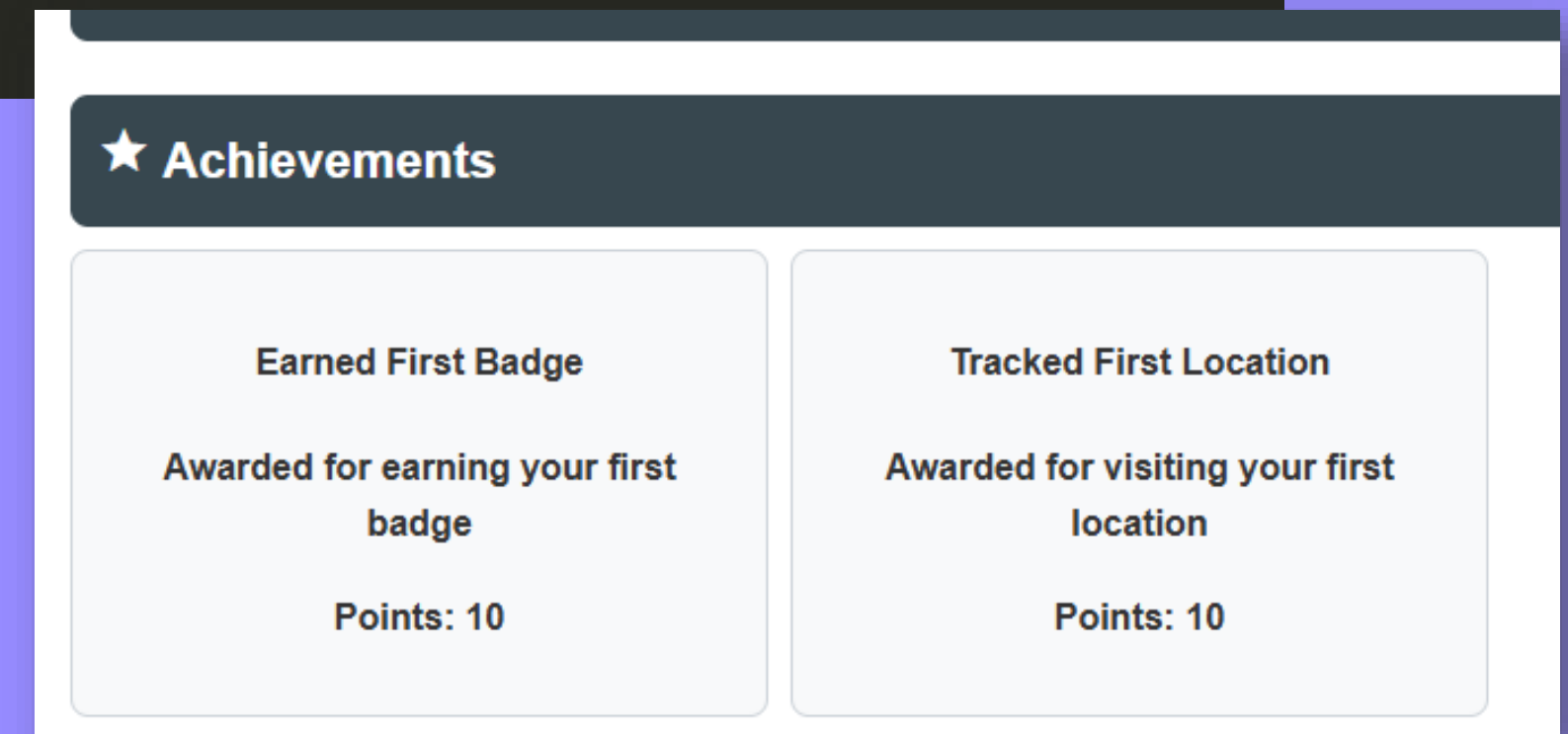
E

S

Gamified UI Elements – Achievements

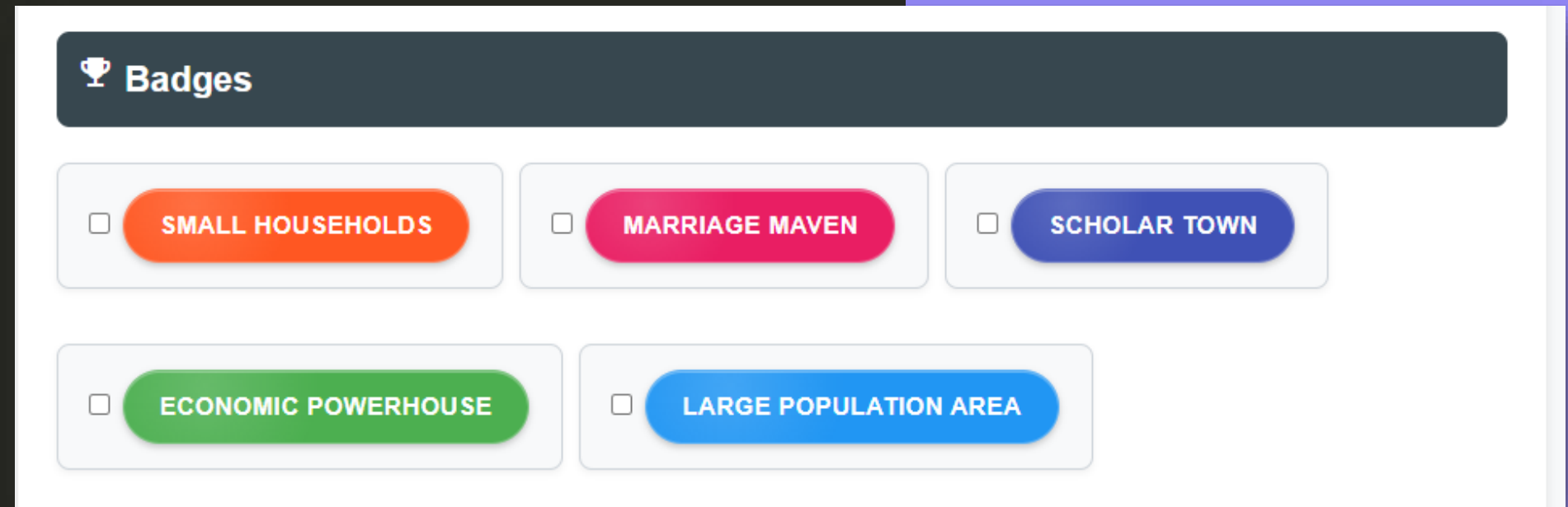
```
const showAchievementPopupMessage = (points, text) => {  
  setAchievementMessage(`Congrats! You've earned ${points} points for ${text}`);  
  setShowAchievementPopup(true);  
  setTimeout(() => {  
    setShowAchievementPopup(false);  
  }, 3000);  
};
```

```
{earnedAchievements.map((achievement, index) => (  
  <div key={index} className="achievement-card">  
    <h4>{achievement.text}</h4>  
    <p>{achievement.description}</p>  
    <p>Points: {achievement.points}</p>  
  </div>  
))}
```



Gamified UI Elements – Badge

```
const Badge = ({ badges }) => {
  const [earnedBadges, setEarnedBadges] = useState([]);
  const [selectedBadge, setSelectedBadge] = useState(null);
  const handleBadgeClick = (badge) => {
    setSelectedBadge(badge);
    setEarnedBadges((prev) => [...prev, badge]);
  };
  return (
    <div className="badges-container">
      {badges.map((badge, index) => (
        <div
          key={index}
          className={`badge ${earnedBadges.includes(badge) ? "badge-earned" : ""}`}
          style={{ backgroundColor: badge.color }}
          onClick={() => handleBadgeClick(badge)}
        >
          {badge.text}
        </div>
      ))}
      {selectedBadge && <BadgePopup badge={selectedBadge} onClose={() => setSelectedBadge(null)} />}
    </div>
  );
};
```



Gamified UI Elements – Basemap Options

```
const basemapOptions = [
  {
    id: "11b7300674584eb793129a808290d235",
    name: "Default Basemap",
    unlocked: true,
  },
  {
    id: "456d1df3810e482b8abcb2aa0440d6ac",
    name: "Valentine's Basemap",
    unlocked: earnedAchievements.length >= 1,
  },
  {
    id: "f030ad3c601c4c4f9404197ded54b8e6",
    name: "Chocolate Mint Basemap",
    unlocked: earnedAchievements.length >= 2,
  },
],
```

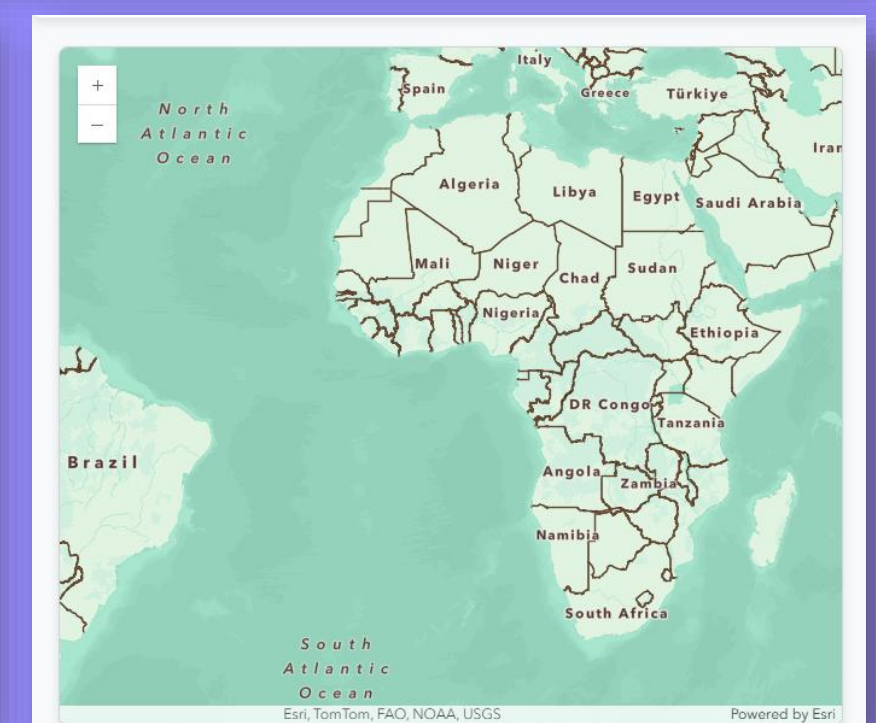
Style Your Map

Default Basemap

Default Basemap

Valentine's Basemap

Chocolate Mint Basemap



Gamified UI Elements – Progress Bar

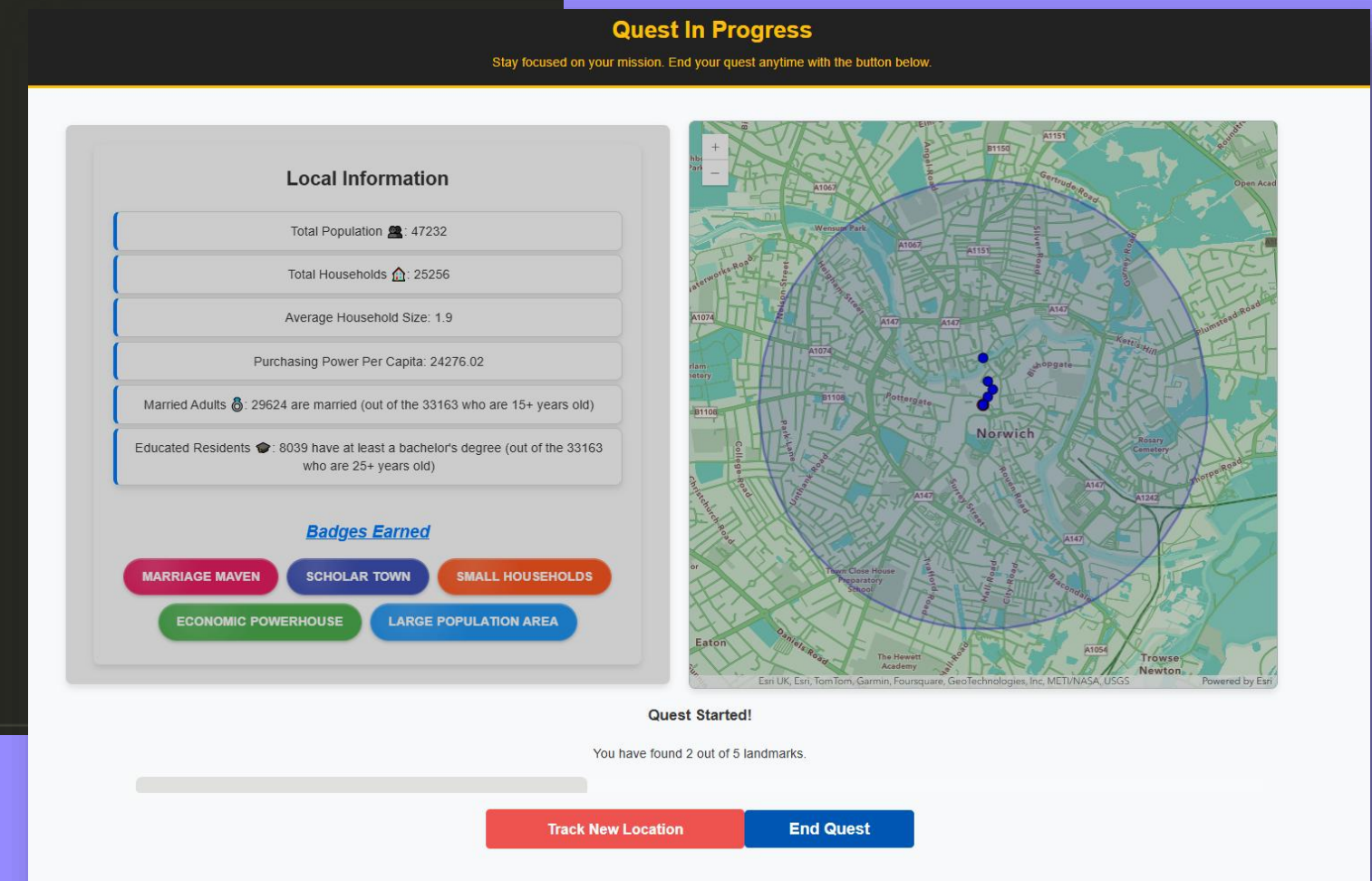
```
const ProgressBar = ({ points, maxPoints }) => {  
  const progress = (points / maxPoints) * 100;  
  return (  
    <div>  
      <div style={{ width: `${progress}%` }} />  
      <span>{points} / {maxPoints} Points</span>  
    </div>  
  );  
};  
  
export default ProgressBar;
```

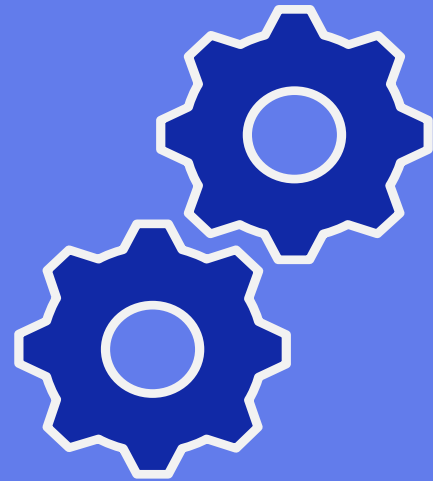
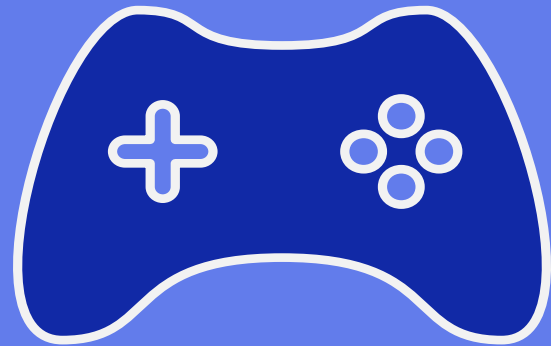
Total Points Earned: 20/500



Gamified UI Elements – Quests

```
const QuestStatus = ({ questStarted, foundLandmarks, totalLandmarks }) => {
  if (!questStarted) return null;
  const progressPercentage = Math.round((foundLandmarks / totalLandmarks) * 100);
  return (
    <div className="quest-status-container">
      <h3>Quest Started!</h3>
      <p>
        You have found {foundLandmarks} out of {totalLandmarks} landmarks.
      </p>
      <div className="progress-bar-container">
        <div
          className="progress-bar"
          style={{ width: `${progressPercentage}%` }}
        ></div>
      </div>
    </div>
  );
};
```





G

Gamified UI
Elements

A

Advanced
State Control

M

E

S

Advanced State Control – Context and Reducer

```
import { createContext, useReducer, useContext } from "react";

const AppContext = createContext();


> const initialState = { ...
};

const reducer = (state, action) => {
>   switch (action.type) { ...
}
};

export const AppProvider = ({ children }) => {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <AppContext.Provider value={{ state, dispatch }}>
      {children}
    </AppContext.Provider>
  );
};

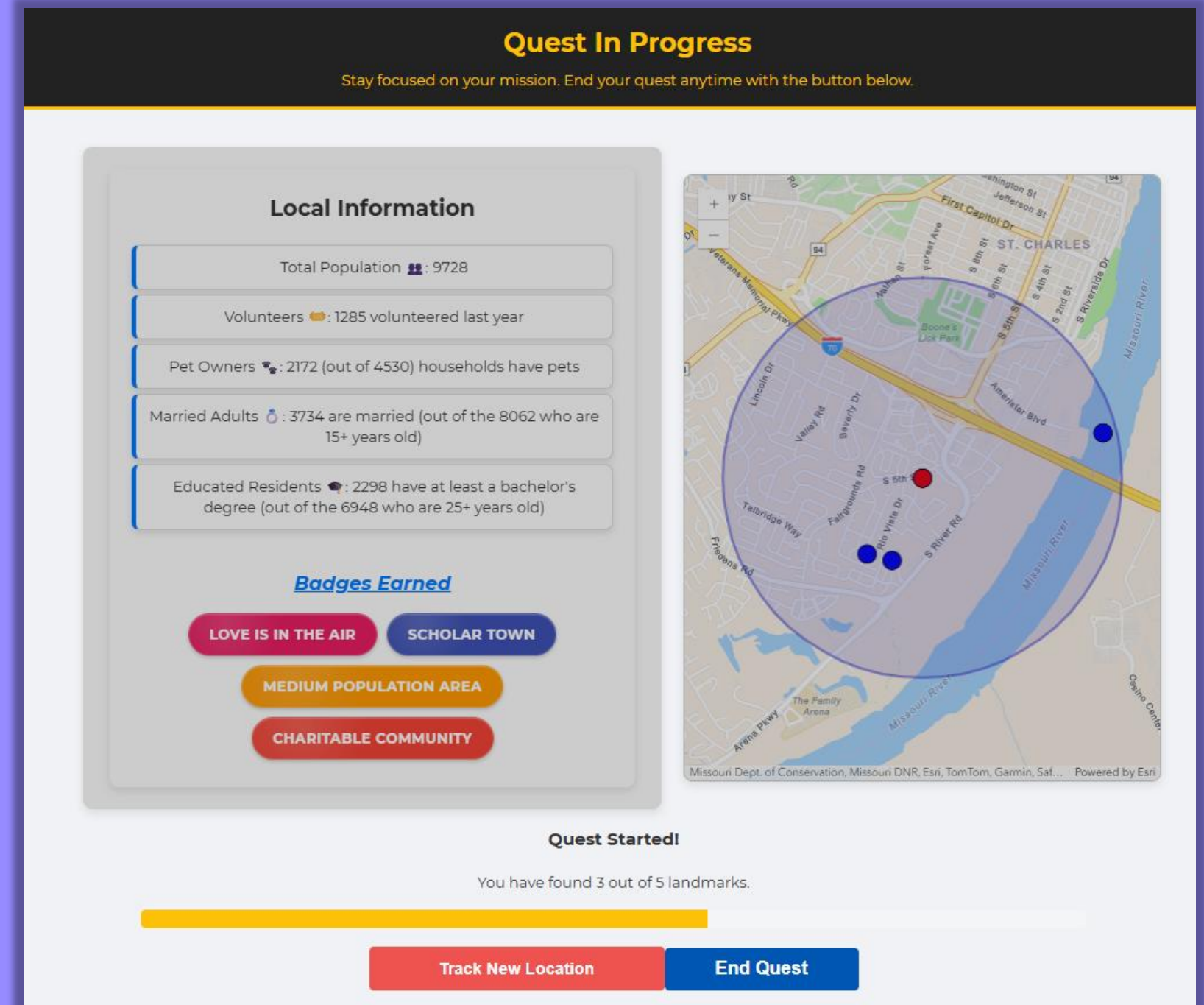
export const useAppContext = () => useContext(AppContext);
```

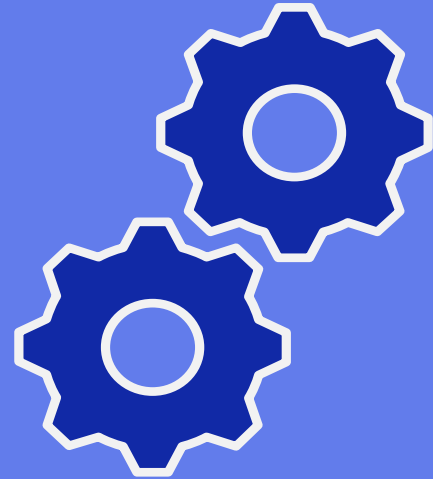
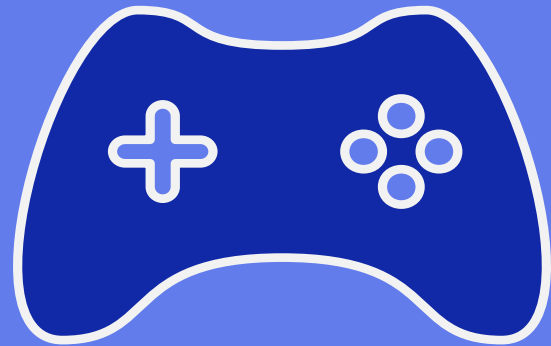


```
switch (action.type) {
  case "SET_LOCATION":
    return { ...state, location: action.payload };
  case "SET_LOCATION_INPUT":
    return { ...state, locationInput: action.payload };
  case "SET_SUBMITTED":
    return { ...state, submitted: action.payload };
  case "RESET":
    return initialState;
  case "SET_SELECTED_BADGE":
    return { ...state, selectedBadge: action.payload };
  case "SET_SELECTED_BASEMAP":
    return { ...state, selectedBasemap: action.payload };
  case "EARN_BADGE":
    return {
      ...state,
      badges: [...state.badges, action.payload],
    };
  default:
    throw new Error(`Unhandled action type: ${action.type}`);
}
```

Advanced State Control – Quest View

```
<QuestStatus
  questStarted={questStarted}
  foundLandmarks={foundLandmarks.length}
  totalLandmarks={nearbyLandmarks.length}
/>
{questStarted && (
  <div className="quest-buttons">
    <button onClick={handleTrackNewLocation}>
      Track New Location
    </button>
    <button onClick={handleEndQuestButtonClick}>
      End Quest
    </button>
  </div>
)}
```





G

Gamified UI
Elements

A

Advanced
State Control

M

Memoization
/Modern
Optimization

E

S

Memoization – Players List

```
import React from 'react';

const Player = React.memo(({ player }) => {
  console.log(`Rendering ${player.name}`);
  return (
    <li>
      {player.name}: {player.score}
    </li>
  );
});

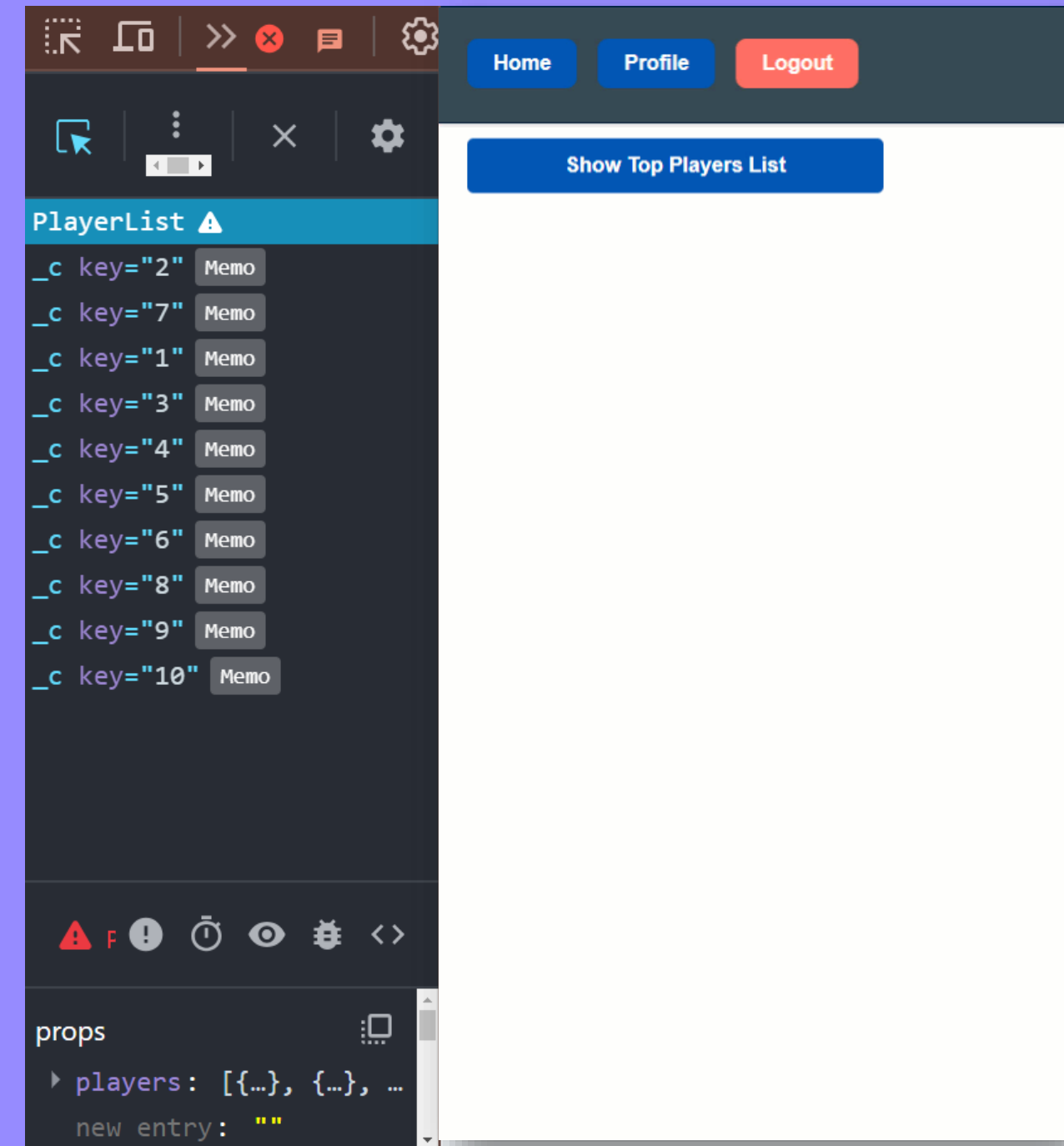
export default Player;

import { useMemo } from 'react';
import Player from './Player';

const PlayerList = ({ players }) => {
  const sortedPlayers = useMemo(() => {
    return players.sort((a, b) => b.score - a.score);
  }, [players]);

  return (
    <div className="player-list">
      <h3>Top Players</h3>
      <ul>
        {sortedPlayers.map(player => (
          <Player key={player.id} player={player} />
        ))}
      </ul>
    </div>
  );
};

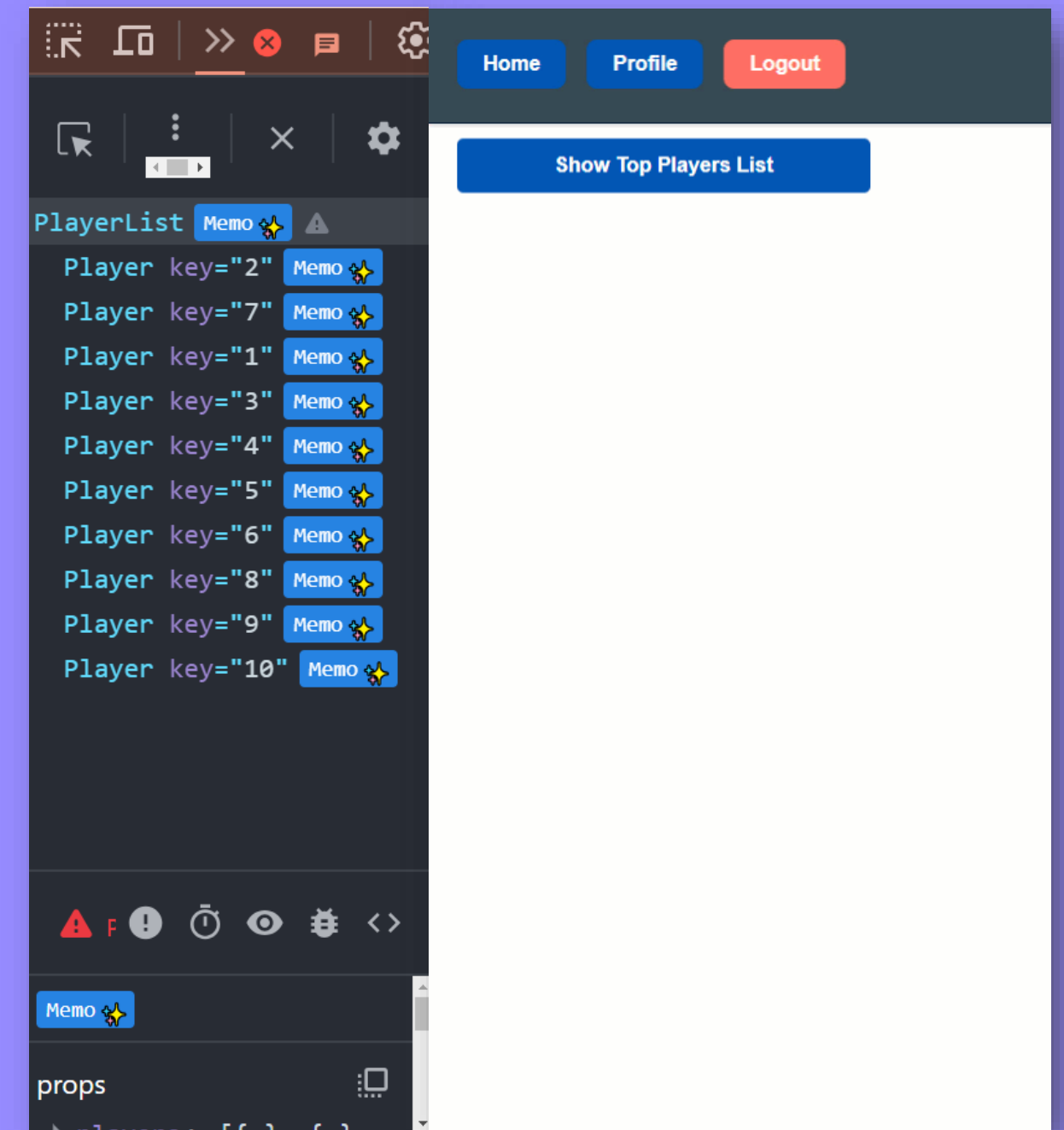
export default PlayerList;
```

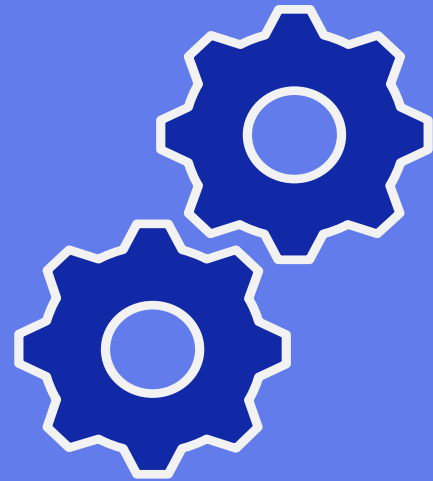
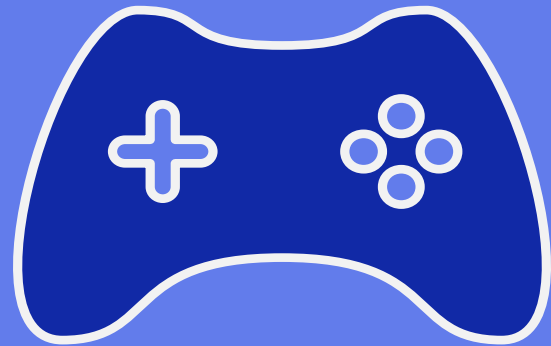


Memoization – Players List with React Compiler

```
const Player = React.memo(( { player } ) => {  
  console.log(`Rendering ${player.name}`);  
  return (  
    <li>
```

```
import { useMemo } from 'react';  
import Player from './Player';  
  
const PlayerList = ({ players }) => {  
  const sortedPlayers = useMemo(() => {  
    return players.sort((a, b) => b.score - a.score);  
  }, [players]);
```





G

Gamified UI
Elements

A

Advanced
State Control

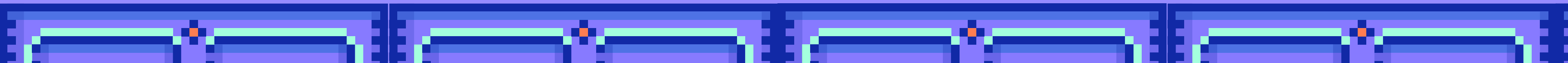
M

Memoization

E

Efficient
Rendering

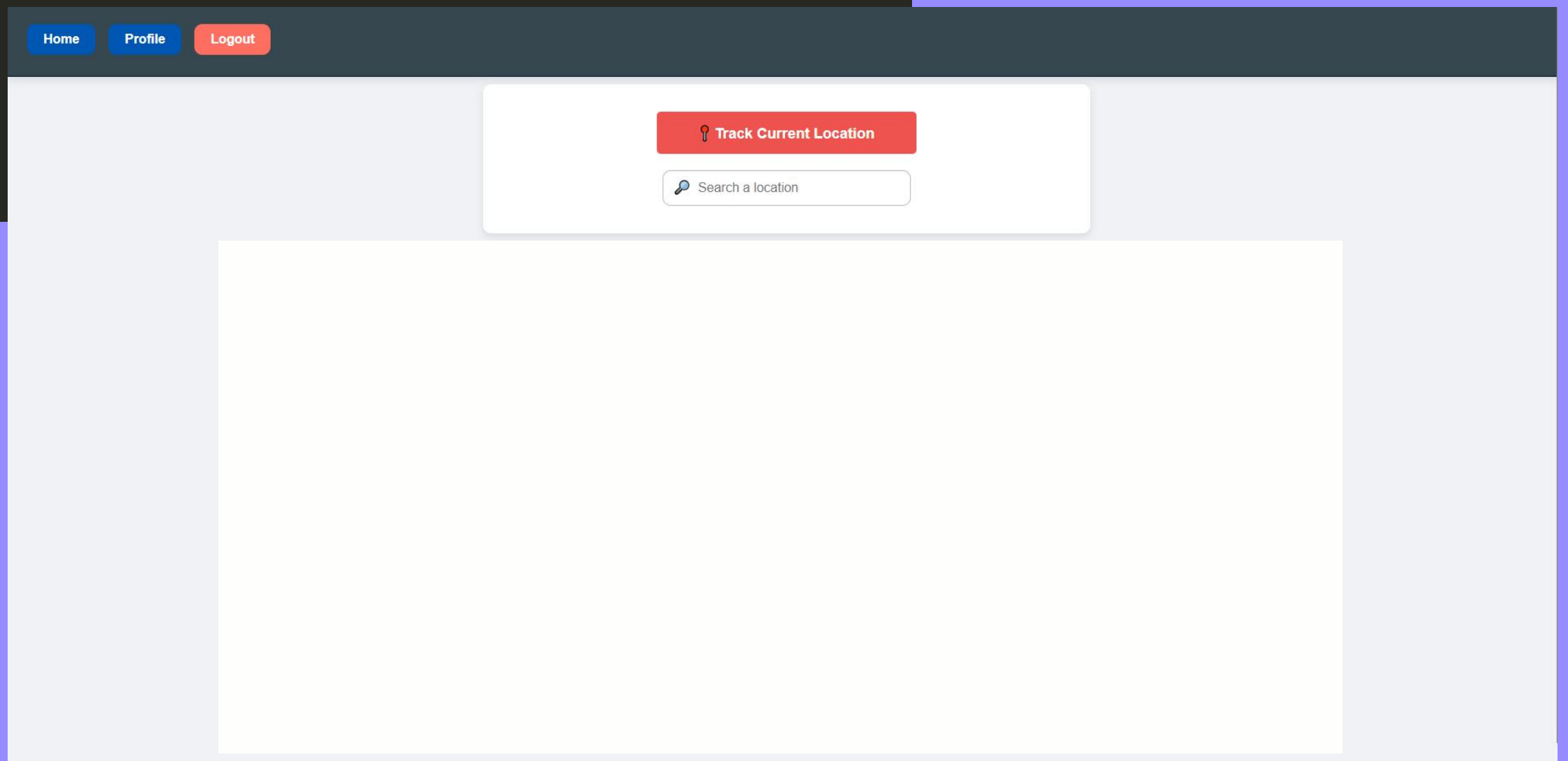
S



Efficient Rendering – Lazy and Suspense

```
const MapViewComponent = lazy(() => import("./components/MapViewComponent"));
const SimpleMapComponent = lazy(() => import("./components/SimpleMapComponent"));
```

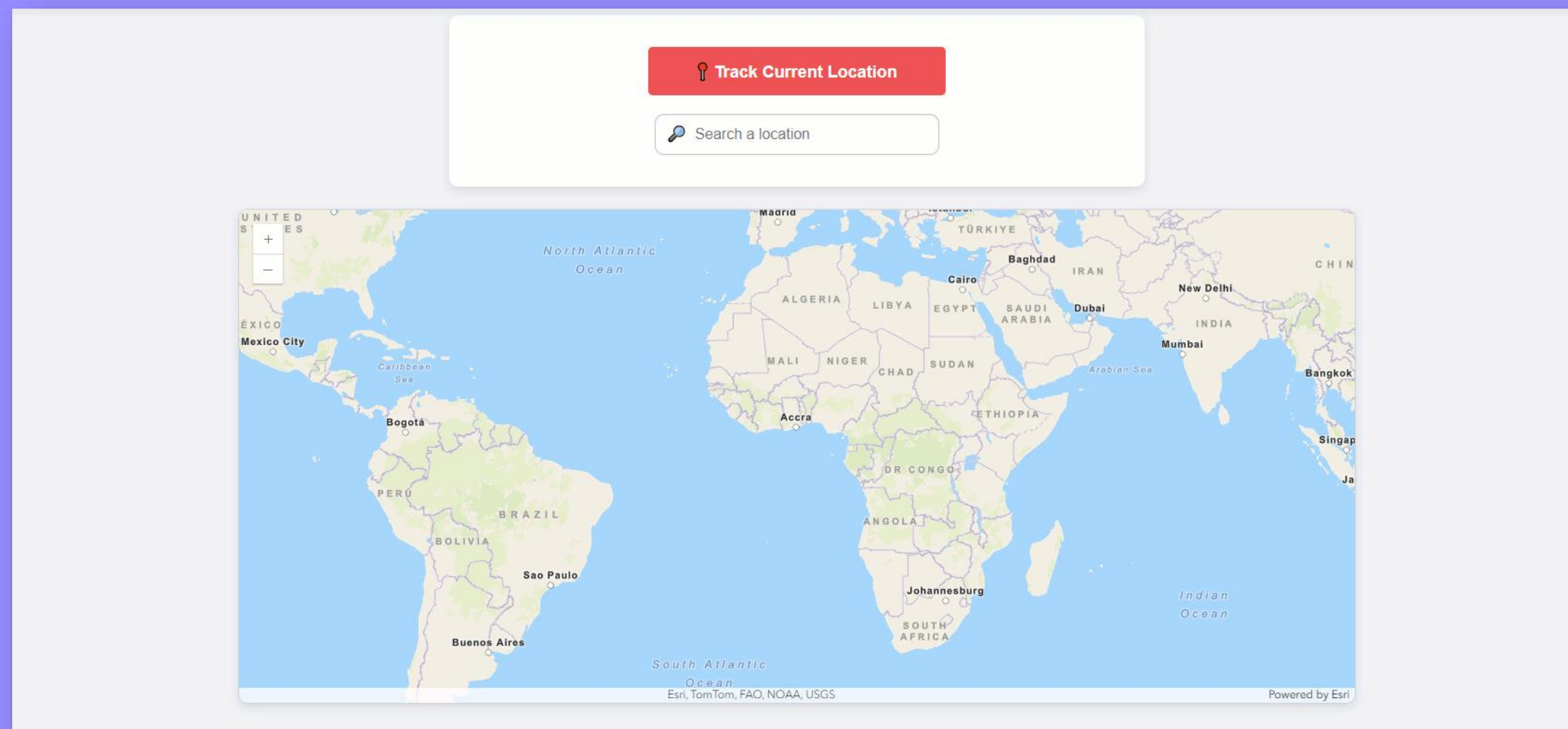
```
<Suspense fallback={<div>Loading Map...</div>}>
  {state.location ? (
    <MapViewComponent location={state.location} landmarks={nearbyLandmarks} />
  ) : (
    <SimpleMapComponent />
  )}
</Suspense>
```



Efficient Rendering – useEffect and useRef

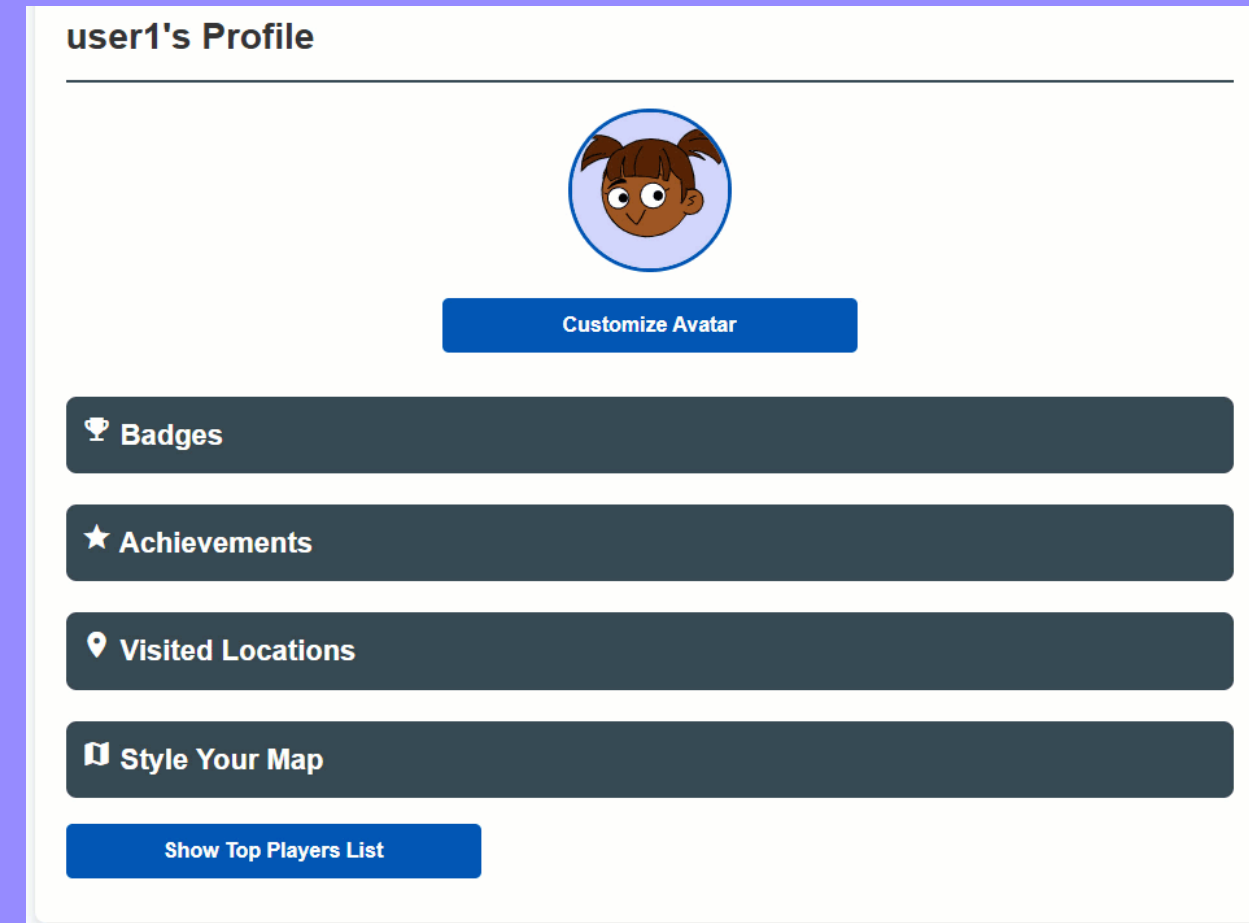
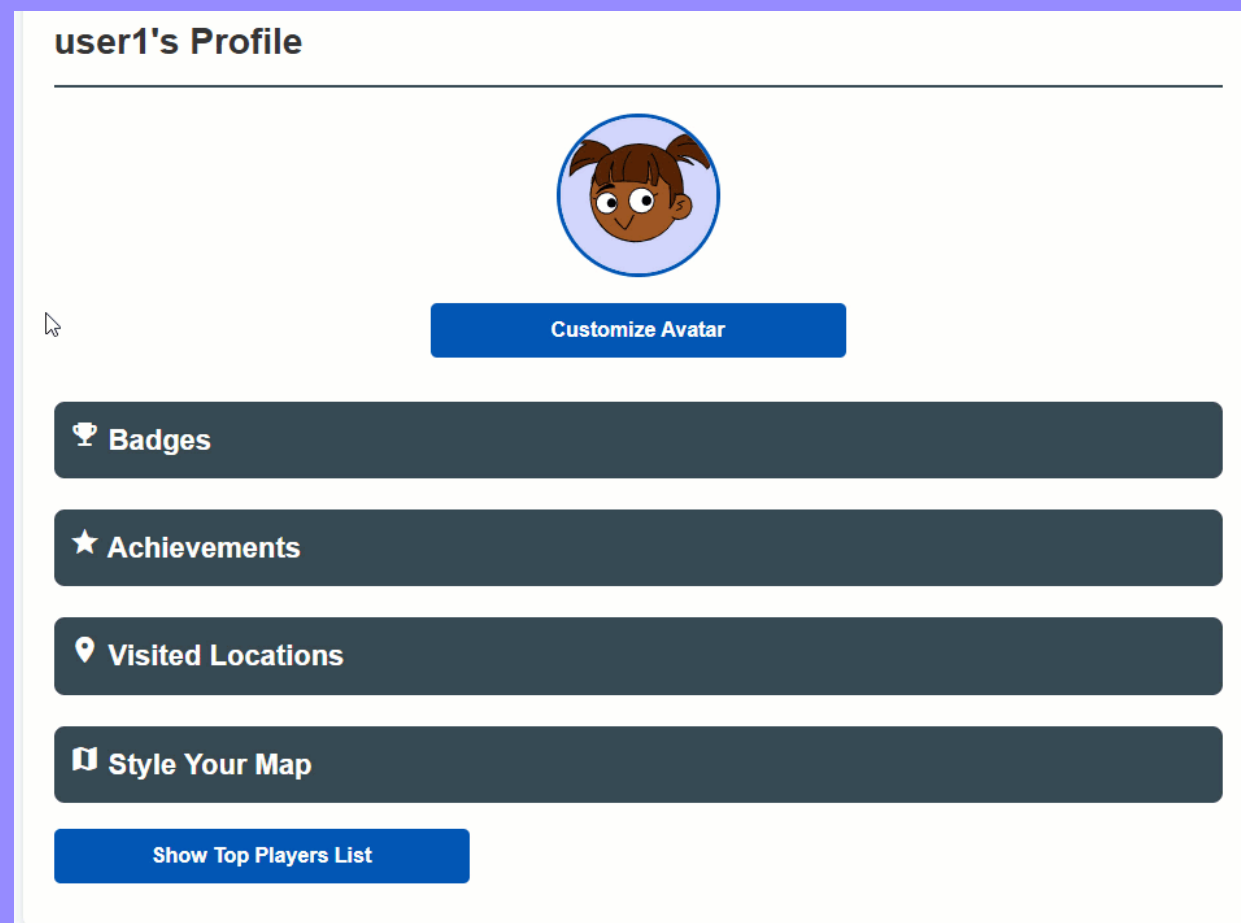
```
const mapDiv = useRef(null);
```

```
useEffect(() => {  
  if (!location || !mapDiv.current) return;  
  // Map initialization code...  
  return () => view.destroy();  
}, [location, landmarks, selectedBasemap]);
```



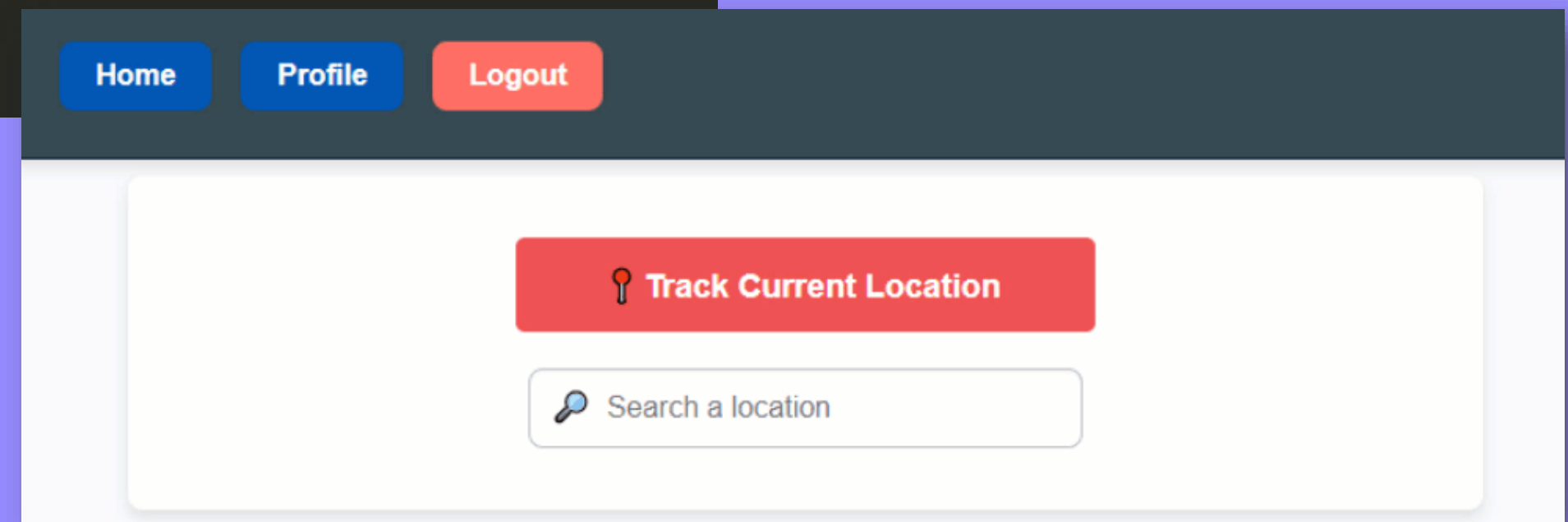
Efficient Rendering – On-Demand Badge Rendering

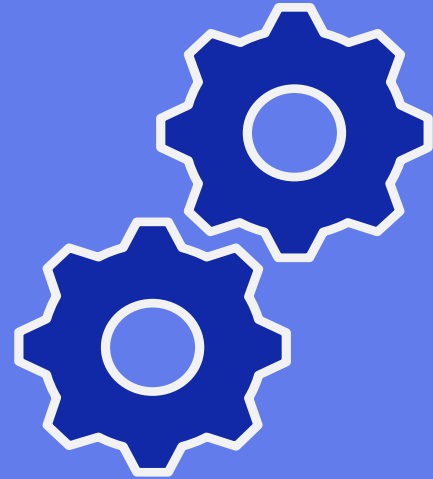
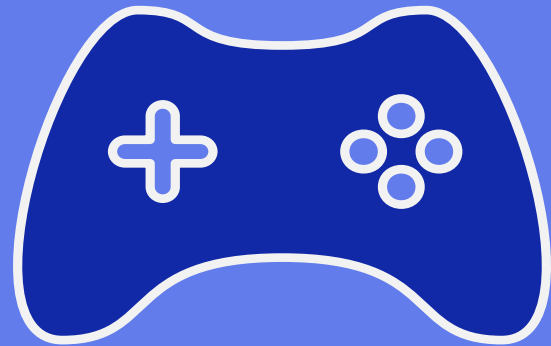
```
{uniqueBadges.length > 0 ? (  
  // render badge list and sharing options  
  <> ...  
  </>  
) : (  
  <p>No badges earned yet.</p>  
)}
```



Efficient Rendering – On-Demand Achievement Popups

```
const queueAchievementPopup = (points, text) => {  
  setAchievementQueue((prevQueue) => [...prevQueue, { points, text }]);  
};  
  
useEffect(() => {  
  if (achievementQueue.length > 0 && !showAchievementPopup) {  
    const { points, text } = achievementQueue[0];  
    showAchievementPopupMessage(points, text);  
    setAchievementQueue((prevQueue) => prevQueue.slice(1));  
  }  
}, [achievementQueue, showAchievementPopup]);
```





G

Gamified UI
Elements

A

Advanced
State Control

M

Memoization

E


Efficient
Rendering

S

Social
Interaction

Social Interaction – Share Badges

```
<div className="share-selected">
  <h4>Share Selected Badges:</h4>
  <div className="social-sharing">
    <FacebookShareButton ...
  </FacebookShareButton>
    <LinkedInShareButton ...
  </LinkedInShareButton>
    <WhatsappShareButton ...
  </WhatsappShareButton>
  </div>
</div>
```

 Badges

☐ SMALL HOUSEHOLDS


☐ MARRIAGE MAVEN


☐ SCHOLAR TOWN


☒ ECONOMIC POWERHOUSE

☐ LARGE POPULATION AREA

Share Selected Badges:

 f

 in



Social Interaction – Player List

```
const PlayerList = ({ players }) => {  
  const sortedPlayers = useMemo(() => {  
    return players.sort((a, b) => b.score - a.score);  
  }, [players]);  
  return (  
    <div className="player-list">  
      <h3>Top Players</h3>  
      <ul>  
        {sortedPlayers.map(player => (  
          <Player key={player.id} player={player} />  
        ))}  
      </ul>  
    </div>  
  );  
};
```





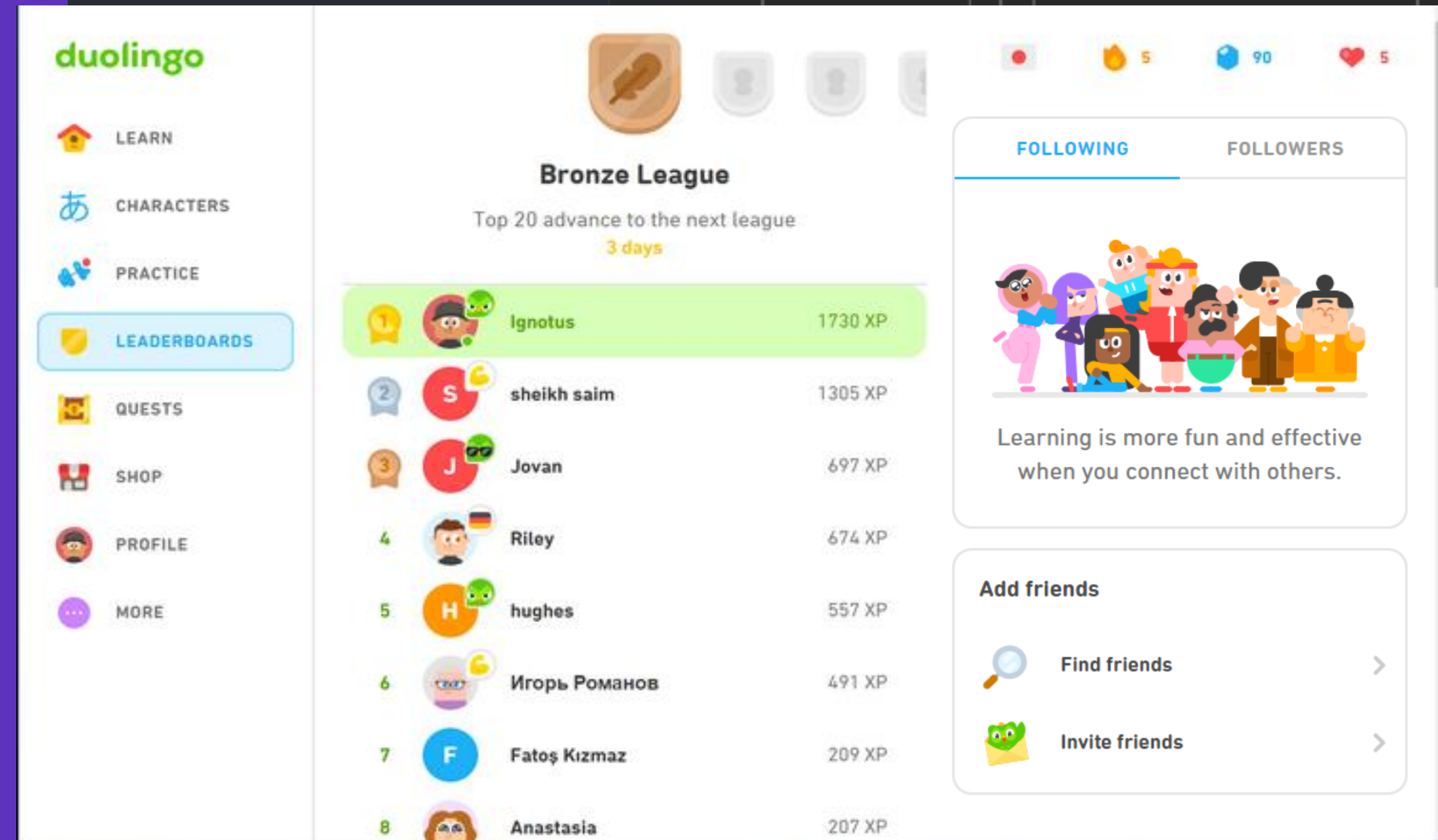
G Streak Counters & XP Bars

A Advanced State Management

M Optimized Lesson Rendering

E Efficient Component Rendering

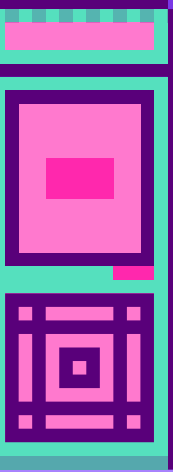
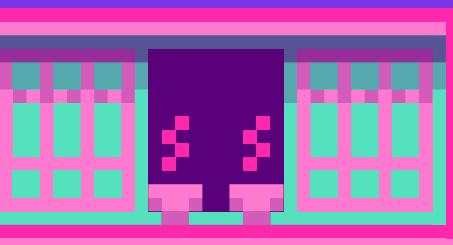
S Leaderboards & Challenges



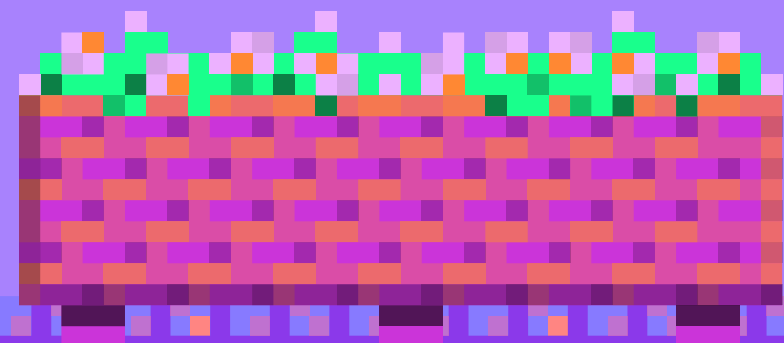


GAMES

WELCOME



GIMKIT.COM/JOIN



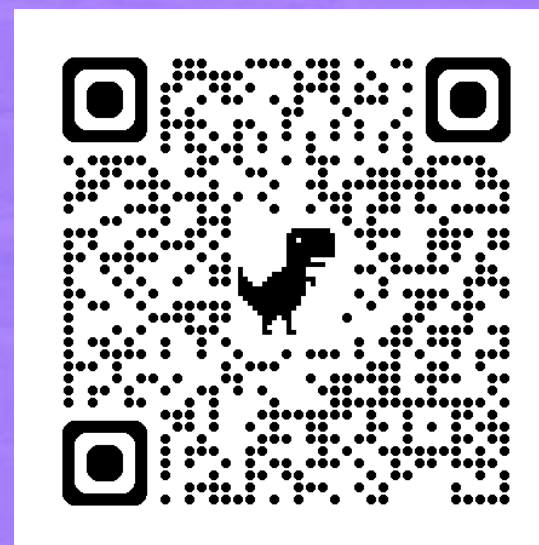
THANK YOU, KCDC!

Courtney Yatteau

  @c_yatteau

 courtneyyatteau

 @cyatteau.bsky.social



[github.com/cyatteau/
react-gamification-
kcdc2025](https://github.com/cyatteau/react-gamification-kcdc2025)