**EMOJI GAME index.js**

import {Component} from 'react'

import EmojiCard from '../EmojiCard'

import NavBar from '../NavBar'

import WinOrLoseCard from '../WinOrLoseCard'

import './index.css'

class EmojiGame extends Component {

  state = {

    clickedEmojisList: [],

    isGameInProgress: true,

    topScore: 0,

  }

  resetGame = () => {

    this.setState({clickedEmojisList: [], isGameInProgress: true})

  }

  renderScoreCard = () => {

    const {emojisList} = this.props

    const {clickedEmojisList} = this.state

    const isWon = clickedEmojisList.length === emojisList.length

    return (

      <WinOrLoseCard

        isWon={isWon}

        onClickPlayAgain={this.resetGame}

        score={clickedEmojisList.length}

      />

    )

  }

  finishGameAndSetTopScore = currentScore => {

    const {topScore} = this.state

    let newTopScore = topScore

    if (currentScore > topScore) {

      newTopScore = currentScore

    }

    this.setState({topScore: newTopScore, isGameInProgress: false})

  }

  clickEmoji = id => {

    const {emojisList} = this.props

    const {clickedEmojisList} = this.state

    const isEmojiPresent = clickedEmojisList.includes(id)

    const clickedEmojisLength = clickedEmojisList.length

    if (isEmojiPresent) {

      this.finishGameAndSetTopScore(clickedEmojisLength)

    } else {

      if (emojisList.length - 1 === clickedEmojisLength) {

        this.finishGameAndSetTopScore(emojisList.length)

      }

      this.setState(previousState => ({

        clickedEmojisList: [...previousState.clickedEmojisList, id],

      }))

    }

  }

  getShuffledEmojisList = () => {

    const {emojisList} = this.props

    return emojisList.sort(() => Math.random() - 0.5)

  }

  renderEmojisList = () => {

    const shuffledEmojisList = this.getShuffledEmojisList()

    return (

      <ul className="emojis-list-container">

        {shuffledEmojisList.map(emojiObject => (

          <EmojiCard

            key={emojiObject.id}

            emojiDetails={emojiObject}

            clickEmoji={this.clickEmoji}

          />

        ))}

      </ul>

    )

  }

  render() {

    const {clickedEmojisList, isGameInProgress, topScore} = this.state

    return (

      <div className="app-container">

        <NavBar

          currentScore={clickedEmojisList.length}

          isGameInProgress={isGameInProgress}

          topScore={topScore}

        />

        <div className="emoji-game-body">

          {isGameInProgress ? this.renderEmojisList() : this.renderScoreCard()}

        </div>

      </div>

    )

  }

}

export default EmojiGame

**Emoji Game index.css**

.app-container {

  display: flex;

  flex-direction: column;

  background-image: linear-gradient(

    to bottom right,

    #9796f0 0.5%,

    #fbc7d4 150%

  );

  min-height: 100vh;

}

.emoji-game-body {

  display: flex;

  flex-direction: column;

  align-items: center;

  flex: 1;

}

.emojis-list-container {

  display: flex;

  justify-content: space-between;

  flex-wrap: wrap;

  width: 80%;

  margin: auto;

  padding: 0px;

  max-width: 1000px;

}

**Emoji Card index.js**

import './index.css'

const EmojiCard = props => {

  const {emojiDetails, clickEmoji} = props

  const {id, emojiName, emojiUrl} = emojiDetails

  const onClickEmojiCard = () => {

    clickEmoji(id)

  }

  return (

    <li className="emoji-item">

      <button type="button" className="emoji-btn" onClick={onClickEmojiCard}>

        <img className="emoji-icon" src={emojiUrl} alt={emojiName} />

      </button>

    </li>

  )

}

export default EmojiCard

**index.css**

.emoji-item {

  width: 32%;

  list-style-type: none;

  margin-top: 8px;

  margin-bottom: 8px;

}

@media screen and (min-width: 992px) {

  .emoji-item {

    width: 24%;

  }

}

.emoji-btn {

  background-color: #ffffff33;

  cursor: pointer;

  outline: none;

  width: 100%;

  height: 100px;

  border: 3px solid #ffffff30;

  border-radius: 24px;

}

@media screen and (min-width: 768px) {

  .emoji-btn {

    height: 200px;

  }

}

.emoji-icon {

  width: 50px;

}

@media screen and (min-width: 768px) {

  .emoji-icon {

    width: 100px;

  }

}

**NavBar index.js**

import './index.css'

const NavBar = props => {

  const {currentScore, isGameInProgress, topScore} = props

  return (

    <nav className="nav-bar-container">

      <div className="title-with-score-container">

        <div className="logo-and-title-container">

          <img

            className="emoji-logo"

            src="https://assets.ccbp.in/frontend/react-js/game-logo-img.png"

            alt="emoji logo"

          />

          <h1 className="title">Emoji Game</h1>

        </div>

        {isGameInProgress && (

          <div className="scores-container">

            <p className="score">Score: {currentScore}</p>

            <p className="score">Top Score: {topScore}</p>

          </div>

        )}

      </div>

    </nav>

  )

}

export default NavBar

**index.css**

.nav-bar-container {

  display: flex;

  justify-content: center;

  background-color: #ffffff33;

  padding: 8px;

}

.title-with-score-container {

  display: flex;

  justify-content: space-between;

  flex-basis: 80%;

  max-width: 1000px;

}

.logo-and-title-container {

  display: flex;

  align-items: center;

}

.emoji-logo {

  width: 24px;

}

@media screen and (min-width: 768px) {

  .emoji-logo {

    width: 48px;

  }

}

.title {

  color: #ffffff;

  font-family: 'Roboto';

  font-size: 16px;

  font-weight: bold;

}

@media screen and (min-width: 768px) {

  .title {

    font-size: 24px;

  }

}

.scores-container {

  display: flex;

}

.score {

  color: #ffffff;

  font-family: 'Roboto';

  font-size: 16px;

  font-weight: 500;

  margin-left: 16px;

}

@media screen and (min-width: 768px) {

  .score {

    font-size: 24px;

    margin-left: 32px;

  }

}

**winorLose index.js**

import './index.css'

const LOSE\_IMAGE = 'https://assets.ccbp.in/frontend/react-js/lose-game-img.png'

const WON\_IMAGE = 'https://assets.ccbp.in/frontend/react-js/won-game-img.png'

const WinOrLoseCard = props => {

  const {isWon, onClickPlayAgain, score} = props

  const imageUrl = isWon ? WON\_IMAGE : LOSE\_IMAGE

  const gameStatus = isWon ? 'You Won' : 'You Lose'

  const scoreLabel = isWon ? 'Best Score' : 'Score'

  return (

    <div className="win-or-lose-card">

      <div className="details-section">

        <h1 className="game-status">{gameStatus}</h1>

        <p className="current-score-label">{scoreLabel}</p>

        <p className="current-score-value">{score}/12</p>

        <button

          type="button"

          className="play-again-button"

          onClick={onClickPlayAgain}

        >

          Play Again

        </button>

      </div>

      <div className="image-section">

        <img className="win-or-lose-image" src={imageUrl} alt="win or lose" />

      </div>

    </div>

  )

}

export default WinOrLoseCard

**index.css**

.win-or-lose-card {

  display: flex;

  flex-direction: column;

  justify-content: space-between;

  background-color: #ffffff33;

  width: 85%;

  border: 3px solid #ffffff30;

  border-radius: 24px;

  margin: auto;

  padding: 16px;

}

@media screen and (min-width: 768px) {

  .win-or-lose-card {

    width: 80%;

    padding: 32px;

  }

}

@media screen and (min-width: 992px) {

  .win-or-lose-card {

    flex-direction: row;

    width: 70%;

  }

}

@media screen and (min-width: 1200px) {

  .win-or-lose-card {

    width: 60%;

  }

}

.details-section {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  order: 1;

}

@media screen and (min-width: 992px) {

  .details-section {

    order: 0;

  }

}

.game-status {

  color: #ffffff;

  font-family: 'Roboto';

  font-size: 32px;

  font-weight: bold;

}

@media screen and (min-width: 992px) {

  .game-status {

    font-size: 64px;

  }

}

.current-score-label {

  text-align: center;

  color: #ffffff;

  font-family: 'Roboto';

  font-size: 18px;

  font-weight: bold;

  margin: 0px;

}

@media screen and (min-width: 992px) {

  .current-score-label {

    font-size: 36px;

  }

}

.current-score-value {

  text-align: center;

  color: #6a59ff;

  font-family: 'Roboto';

  font-size: 32px;

  font-weight: bold;

  margin: 0px;

}

@media screen and (min-width: 768px) {

  .current-score-value {

    font-size: 48px;

  }

}

.play-again-button {

  color: #3d3d3d;

  background-color: #ffce27;

  font-family: 'Roboto';

  font-size: 16px;

  font-weight: bold;

  border: none;

  border-radius: 20px;

  margin: 16px;

  padding: 16px;

  padding-right: 24px;

  padding-left: 24px;

  cursor: pointer;

  outline: none;

}

.image-section {

  display: flex;

  justify-content: center;

  order: 0;

}

@media screen and (min-width: 992px) {

  .image-section {

    order: 1;

  }

}

.win-or-lose-image {

  width: 150px;

  height: 200px;

}

@media screen and (min-width: 992px) {

  .win-or-lose-image {

    width: 350px;

    height: 450px;

  }

}