React Signup Page with Firebase Integration

A Step-by-Step Implementation Guide

This guide walks you through converting your signup page to React while keeping the rest of your site as HTML/CSS/JS. The guide uses the latest Firebase practices and clearly indicates example code versus customization requirements.

Table of Contents

Environment Setup

Install Node.js and NPM

1. Download and install Node.js from https://nodejs.org/ (choose the LTS version)
2. Open a command prompt in your project directory
3. Verify installation:

   ```
   node -v
   npm -v
   ```

Initialize NPM in Your Project

```
npm init -y
```

Install Required Dependencies

Install the latest Firebase SDK (v12.1.0 as of August 2025) and React:

```
npm install firebase@12.1.0 react react-dom
```

Install development tools:

```
npm install --save-dev webpack webpack-cli babel-loader @babel/core @babel/preset-env @babel/preset-react css-loader style-loader
```

Install Firebase CLI (Optional)

If you need to deploy to Firebase Hosting or use other Firebase services from the command line:

```
npm install -g firebase-tools@14.11.1
```

After installation, authenticate the CLI:

```
firebase login
```

Project Structure

Create the following folder structure:

```
src/
  ├── react/
  │    └── signup/
  │         ├── components/
  │         │    ├── AccountInfoStep.jsx
  │         │    ├── ServiceTierStep.jsx
  │         │    ├── PaymentStep.jsx
  │         │    └── ConfirmationStep.jsx
  │         ├── SignupForm.jsx
  │         ├── index.js
  │         └── signup.css
  └── firebase-config.js
webpack.config.js
```

You can create these folders and files using File Explorer or with command line:

```
mkdir -p src/react/signup/components
touch src/react/signup/components/AccountInfoStep.jsx
touch src/react/signup/components/ServiceTierStep.jsx
touch src/react/signup/components/PaymentStep.jsx
touch src/react/signup/components/ConfirmationStep.jsx
touch src/react/signup/SignupForm.jsx
touch src/react/signup/index.js
touch src/react/signup/signup.css
touch src/firebase-config.js
touch webpack.config.js
```

Webpack Configuration

Create a `webpack.config.js` file in your project root:

```
// webpack.config.js
const path = require('path');

module.exports = {
  mode: 'development', // Change to 'production' for production builds
  entry: './src/react/signup/index.js',
  output: {
    filename: 'signup-bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        exclude: /node_modules/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: ['@babel/preset-env', '@babel/preset-react']
          }
        }
      },
      {
        test: /\.css$/,
        use: ['style-loader', 'css-loader']
      }
    ]
  },
  resolve: {
    extensions: ['.js', '.jsx']
  }
};
```

Add build scripts to your `package.json`:

```
"scripts": {
  "build": "webpack --mode production",
  "dev": "webpack --mode development --watch"
}
```

Firebase Configuration

This section covers setting up Firebase with the latest modular SDK approach, which is the current best practice recommended by Firebase.

Firebase SDK Version Approaches

Firebase offers two main SDK approaches:

1. **Modular SDK (Recommended)** - Uses specific function imports for better tree-shaking

   ```
   import { getAuth, createUserWithEmailAndPassword } from 'firebase/auth';
   ```

2. **Compat SDK (Legacy)** - Uses namespace-style imports, compatible with older code

   ```
   import firebase from 'firebase/compat/app';
   // Or via CDN
   // <script src="https://www.gstatic.com/firebasejs/9.x.x/firebase-app-compat.js"></script>
   ```

Important: For new React components, always use the modular SDK. Your existing HTML/JS pages can continue using the compat SDK until you're ready to migrate them.

Authentication Options

Firebase provides several options for implementing authentication:

1. **Custom React Authentication (This Guide)** - Full control over UI and authentication flow
   - Pros: Complete customization, integrates with your site design
   - Cons: More code to write and maintain
2. **FirebaseUI Auth (Alternative)** - Pre-built, customizable authentication UI component
   - Pros: Less code to write, handles many edge cases
   - Cons: Less customization, may not match your site perfectly
   - Usage: `npm install firebaseui` or use the CDN version

Example FirebaseUI Auth implementation:

```
import * as firebaseui from 'firebaseui';
import 'firebaseui/dist/firebaseui.css';
import { getAuth } from 'firebase/auth';

// Initialize FirebaseUI Auth
const ui = new firebaseui.auth.AuthUI(getAuth());

// Configure FirebaseUI
ui.start('#firebaseui-auth-container', {
  signInOptions: [
```

```
      {
        provider: firebase.auth.EmailAuthProvider.PROVIDER_ID,
        requireDisplayName: true
      },
      firebase.auth.GoogleAuthProvider.PROVIDER_ID
    ],
    callbacks: {
      signInSuccessWithAuthResult: (authResult, redirectUrl) => {
        // Handle successful sign-in
        return false; // Prevent redirect
      }
    }
  });
```

Important Note: Firebase Dynamic Links, often used for email authentication, is deprecated and will be shut down on August 25, 2025. If you're implementing email link authentication, you'll need to use a different approach, such as password-based authentication or third-party providers.

Creating a Firebase Configuration

Create a shared Firebase configuration file that works with both your React components and existing code:

```
// src/firebase-config.js

// CUSTOMIZE: Use your Firebase config values from auth.js
const firebaseConfig = {
  apiKey: "AIzaSyADF9yuram-pvlzjg6kBtdCk7LuK0M65tk",
  authDomain: "shreyfitweb.firebaseapp.com",
  projectId: "shreyfitweb",
  storageBucket: "shreyfitweb.firebasestorage.app",
  messagingSenderId: "1076359633281",
  appId: "1:1076359633281:web:3687e1675c9e185f0ab080",
  measurementId: "G-5GBP19SXBW"
};

// Use modern modular Firebase SDK
import { initializeApp } from 'firebase/app';
import {
  getAuth,
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  signInWithPopup,
  GoogleAuthProvider
} from 'firebase/auth';
import {
  getFirestore,
  doc,
  setDoc,
  serverTimestamp
} from 'firebase/firestore';
```

```
// Initialize Firebase for React components
let app;
try {
  app = initializeApp(firebaseConfig);
} catch (error) {
  // Handle the duplicate app initialization
  if (error.code === 'app/duplicate-app') {
    console.info('Firebase already initialized, using existing instance');
    app = initializeApp(undefined, 'default');
  } else {
    console.error("Firebase initialization error:", error);
  }
}

export const auth = getAuth(app);
export const db = getFirestore(app);

// Function to create a user with tier information (modern approach)
export async function createUserWithTier(email, password, name, phone, tier) {
  try {
    // Use modern Firebase SDK
    const userCredential = await createUserWithEmailAndPassword(auth, email,
password);
    const userId = userCredential.user.uid;

    await setDoc(doc(db, 'users', userId), {
      name: name,
      email: email,
      phone: phone || null,
      tier: tier,
      role: 'client',
      createdAt: serverTimestamp()
    });

    return { success: true, userId, user: userCredential.user };
  } catch (error) {
    console.error('Error creating user:', error);
    return {
      success: false,
      error: error
    };
  }
}

// Function to sign in a user (modern approach)
export async function signInUser(email, password) {
  try {
    const userCredential = await signInWithEmailAndPassword(auth, email,
password);
    return {
      success: true,
      user: userCredential.user
    };
  } catch (error) {
    console.error('Sign in error:', error);
    return {
      success: false,
      error: error
    };
  }
}

// Helper for Google sign-in
export async function signInWithGoogleAuth() {
  const provider = new GoogleAuthProvider();
  try {
    const result = await signInWithPopup(auth, provider);
    return result;
  } catch (error) {
    console.error('Google sign-in error:', error);
    throw error;
  }
}
```

Hybrid Architecture Approach

Your site can successfully use a hybrid architecture with both React components and traditional HTML/CSS/JS pages, all working with Firebase. Here's how it works:

Firebase Initialization in Different Environments

Firebase must be initialized separately in each standalone environment, but they can share the same configuration:

1. **For React Components**:
   - Initialize Firebase at the module level using the modular SDK (as shown in the firebase-config.js example above)
   - Import and use specific Firebase functions
2. **For HTML/CSS/JS Pages**:
   - Continue using your existing approach with script tags
   - Or gradually migrate to module-style imports with `<script type="module">`

Key Benefits of This Approach

1. You can **convert pages incrementally** - no need to rewrite your entire site at once
2. Each page/component can use the most appropriate technology for its needs
3. Your existing Firebase-powered features continue to work normally
4. All pages share the same Firebase project and data

Example: Initializing Firebase in HTML/CSS/JS Pages

For non-React pages, you can either:

1. **Keep Using the Compat SDK**:

```

```

~~~
2. **Or Migrate to Modular SDK with ES modules**:
```html
<script type="module">
  import { initializeApp } from
"https://www.gstatic.com/firebasejs/12.1.0/firebase-app.js";
  import { getAuth } from "https://www.gstatic.com/firebasejs/12.1.0/firebase-
auth.js";

  const firebaseConfig = {
    // Your Firebase config (same as in firebase-config.js)
  };

  const app = initializeApp(firebaseConfig);
  const auth = getAuth(app);

  // Use Firebase services here
</script>
~~~

HTML Integration

Update your `signup.html` to include a container for the React signup form while keeping your existing Firebase initialization:

```html
<!-- Keep your existing header and structure -->
<section class="signup-section">
    <div class="container">
        <div class="section-header">
            <h2>Join SHREY.FIT</h2>
            <p>Create your account to start your fitness journey</p>
        </div>

        <!-- This is where React will mount -->
        <div id="react-signup-form"></div>
    </div>
</section>

<!-- Keep your existing Firebase scripts -->
<script src="https://www.gstatic.com/firebasejs/9.22.0/firebase-app-compat.js">
</script>
<script src="https://www.gstatic.com/firebasejs/9.22.0/firebase-auth-compat.js">
</script>
<script src="https://www.gstatic.com/firebasejs/9.22.0/firebase-firestore-
compat.js"></script>
```

```html
<!-- Keep your existing auth.js script -->
<script src="js/auth.js"></script>

<!-- Add the React bundle at the end -->
<script src="dist/signup-bundle.js"></script>
```

Important Notes:

1. The React components will use their own Firebase initialization via the modular SDK
2. Your existing auth.js script will continue working with the compat SDK
3. Both environments will operate on the same Firebase project
4. No changes to your existing Firebase configuration in the Firebase console are needed

React Components

Entry Point

Create the React entry point:

```javascript
// src/react/signup/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { SignupForm } from './SignupForm';
import './signup.css';

// Wait for DOM to be ready, then mount React
document.addEventListener('DOMContentLoaded', () => {
  const container = document.getElementById('react-signup-form');
  if (container) {
    ReactDOM.render(<SignupForm />, container);
  } else {
    console.error('Could not find react-signup-form container element');
  }
});
```

Main Signup Form Component

Create the main signup form component with multi-step functionality:

```javascript
// src/react/signup/SignupForm.jsx
import React, { useState } from 'react';
import { createUserWithTier } from '../../firebase-config';
import AccountInfoStep from './components/AccountInfoStep';
import ServiceTierStep from './components/ServiceTierStep';
import PaymentStep from './components/PaymentStep';
import ConfirmationStep from './components/ConfirmationStep';

export function SignupForm() {
  const [currentStep, setCurrentStep] = useState(1);
```

```
    const [formData, setFormData] = useState({
      name: '',
      email: '',
      phone: '',
      password: '',
      confirmPassword: '',
      tier: null,
      paymentInfo: null
    });
    const [error, setError] = useState('');
    const [isSubmitting, setIsSubmitting] = useState(false);

    const updateFormData = (data) => {
      setFormData({...formData, ...data});
      // Clear errors when form data changes
      setError('');
    };

    const nextStep = () => setCurrentStep(currentStep + 1);
    const prevStep = () => setCurrentStep(currentStep - 1);

    const handleSubmit = async () => {
      setIsSubmitting(true);
      setError('');

      try {
        // Create user with tier info
        const result = await createUserWithTier(
          formData.email,
          formData.password,
          formData.name,
          formData.phone,
          formData.tier
        );

        if (!result.success) {
          throw new Error(result.error?.message || 'Failed to create account');
        }

        // Dispatch the same success event your current code looks for
        const signupSuccessEvent = new CustomEvent('signupSuccess', {
          detail: {
            email: formData.email,
            name: formData.name,
            autoLogin: true,
            tier: formData.tier
          }
        });
        document.dispatchEvent(signupSuccessEvent);

      } catch (error) {
        console.error('Signup error:', error);
        setError(error.message);
        setIsSubmitting(false);
      }
    };

    // Render the current step
    const renderStep = () => {
      switch(currentStep) {
        case 1:
          return (
            <AccountInfoStep
              formData={formData}
              updateFormData={updateFormData}
              nextStep={nextStep}
              error={error}
            />
          );
        case 2:
          return (
            <ServiceTierStep
              formData={formData}
              updateFormData={updateFormData}
              nextStep={nextStep}
              prevStep={prevStep}
              error={error}
            />
          );
        case 3:
          return (
            <PaymentStep
              formData={formData}
              updateFormData={updateFormData}
              nextStep={nextStep}
              prevStep={prevStep}
              error={error}
            />
          );
        case 4:
          return (
            <ConfirmationStep
              formData={formData}
              handleSubmit={handleSubmit}
              prevStep={prevStep}
              isSubmitting={isSubmitting}
              error={error}
            />
          );
        default:
          return <AccountInfoStep
                   formData={formData}
                   updateFormData={updateFormData}
                   nextStep={nextStep}
                   error={error}
                 />;
      }
    };
```

```
  return (
    <div className="signup-form-container">
      {/* Progress indicator */}
      <div className="progress-steps">
        <div className={`step ${currentStep >= 1 ? 'active' : ''}`}>
          <div className="step-number">1</div>
          <div className="step-label">Account</div>
        </div>
        <div className={`step-connector ${currentStep >= 2 ? 'active' : ''}`}>
</div>
        <div className={`step ${currentStep >= 2 ? 'active' : ''}`}>
          <div className="step-number">2</div>
          <div className="step-label">Select Plan</div>
        </div>
        <div className={`step-connector ${currentStep >= 3 ? 'active' : ''}`}>
</div>
        <div className={`step ${currentStep >= 3 ? 'active' : ''}`}>
          <div className="step-number">3</div>
          <div className="step-label">Payment</div>
        </div>
        <div className={`step-connector ${currentStep >= 4 ? 'active' : ''}`}>
</div>
        <div className={`step ${currentStep >= 4 ? 'active' : ''}`}>
          <div className="step-number">4</div>
          <div className="step-label">Confirm</div>
        </div>
      </div>

      {/* Current step */}
      {renderStep()}
    </div>
  );
}

export default SignupForm;
```

Step Components

Account Info Step

```
// src/react/signup/components/AccountInfoStep.jsx
import React from 'react';

function AccountInfoStep({ formData, updateFormData, nextStep, error }) {
  const handleChange = (e) => {
    updateFormData({ [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    // Form validation
    if (!formData.name || !formData.email || !formData.password ||
!formData.confirmPassword) {
      updateFormData({ error: 'Please fill in all required fields' });
      return;
    }

    if (formData.password !== formData.confirmPassword) {
      updateFormData({ error: 'Passwords do not match' });
      return;
    }

    // Proceed to next step
    nextStep();
  };

  return (
    <form className="signup-form" onSubmit={handleSubmit}>
      <h3>Create Your Account</h3>

      {error && <div className="error-message">{error}</div>}

      <div className="form-group">
        <label htmlFor="fullname">
          <i className="fas fa-user"></i> Full Name <span className="required">*
</span>
        </label>
        <input
          type="text"
          id="fullname"
          name="name"
          value={formData.name}
          onChange={handleChange}
          placeholder="Enter your full name"
          required
        />
      </div>

      <div className="form-group">
        <label htmlFor="email">
          <i className="fas fa-envelope"></i> Email Address <span
className="required">*</span>
        </label>
        <input
          type="email"
          id="email"
          name="email"
          value={formData.email}
          onChange={handleChange}
          placeholder="Enter your email address"
          required
        />
      </div>
      <div className="form-group">
        <label htmlFor="phone">
          <i className="fas fa-phone"></i> Phone Number
        </label>
        <input
          type="tel"
          id="phone"
          name="phone"
          value={formData.phone}
          onChange={handleChange}
          placeholder="Enter your phone number"
        />
      </div>

      <div className="form-group">
        <label htmlFor="password">
          <i className="fas fa-lock"></i> Password <span className="required">*
</span>
        </label>
        <input
          type="password"
          id="password"
          name="password"
          value={formData.password}
          onChange={handleChange}
          placeholder="Create a password"
          required
        />
      </div>

      <div className="form-group">
        <label htmlFor="confirmPassword">
          <i className="fas fa-lock"></i> Confirm Password <span
className="required">*</span>
        </label>
        <input
          type="password"
          id="confirmPassword"
          name="confirmPassword"
          value={formData.confirmPassword}
          onChange={handleChange}
          placeholder="Confirm your password"
          required
        />
      </div>

      <button type="submit" className="btn-primary-enhanced">
        <i className="fas fa-arrow-right"></i>
        Continue
      </button>
    </form>
  );
}
```

```
export default AccountInfoStep;
```

Service Tier Step

```
// src/react/signup/components/ServiceTierStep.jsx
import React from 'react';

function ServiceTierStep({ formData, updateFormData, nextStep, prevStep, error })
{
  // CUSTOMIZE: Match your service tiers from services.html
  const tiers = [
    {
      id: 'in-person-training',
      title: 'In-Person Training',
      price: '$70/session',
      description: 'Expert in-person coaching sessions focused on technique, form,
and effective workouts.',
      details: 'Seattle Area Only'
    },
    {
      id: 'online-coaching',
      title: 'Online Coaching',
      price: '$199/month',
      description: 'Complete transformation system with custom programs, nutrition
guidance, and support.',
      details: 'Remote Coaching'
    },
    {
      id: 'complete-transformation',
      title: 'Complete Transformation',
      price: '$199/month + $60/session',
      description: 'Comprehensive coaching plus hands-on training for maximum
results.',
      details: 'Seattle Premium Experience'
    }
  ];

  const handleTierSelect = (tier) => {
    updateFormData({ tier });
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!formData.tier) {
      updateFormData({ error: 'Please select a service tier' });
      return;
    }

    nextStep();
  };

  return (
    <form className="signup-form" onSubmit={handleSubmit}>
      <h3>Select Your Service Tier</h3>

      {error && <div className="error-message">{error}</div>}

      <p className="form-description">
        Choose the service that best fits your fitness goals and preferences.
      </p>

      <div className="service-tiers">
        {tiers.map(tier => (
          <div
            key={tier.id}
            className={`service-tier ${formData.tier === tier.id ? 'selected' :
''}`}
            onClick={() => handleTierSelect(tier.id)}
          >
            <div className="tier-header">
              <input
                type="radio"
                name="tier"
                id={tier.id}
                className="tier-radio"
                checked={formData.tier === tier.id}
                onChange={() => handleTierSelect(tier.id)}
              />
              <h4 className="tier-title">{tier.title}</h4>
            </div>
            <div className="tier-price">{tier.price}</div>
            <p className="tier-description">{tier.description}</p>
            <span className="tier-details">{tier.details}</span>
          </div>
        ))}
      </div>

      <div className="button-row">
        <button
          type="button"
          className="btn-secondary"
          onClick={prevStep}
        >
          <i className="fas fa-arrow-left"></i> Back
        </button>
        <button type="submit" className="btn-primary-enhanced">
          Continue <i className="fas fa-arrow-right"></i>
        </button>
      </div>
    </form>
  );
}
```

```
export default ServiceTierStep;
```

Payment Step

```
// src/react/signup/components/PaymentStep.jsx
import React, { useState } from 'react';

function PaymentStep({ formData, updateFormData, nextStep, prevStep, error }) {
  // This is a simplified payment form
  // In production, you would integrate with Stripe or another payment processor

  const [cardInfo, setCardInfo] = useState({
    cardNumber: '',
    cardName: '',
    expiry: '',
    cvv: ''
  });

  const handleChange = (e) => {
    setCardInfo({
      ...cardInfo,
      [e.target.name]: e.target.value
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Basic validation
    if (!cardInfo.cardNumber || !cardInfo.cardName || !cardInfo.expiry ||
!cardInfo.cvv) {
      updateFormData({ error: 'Please complete all payment fields' });
      return;
    }

    // In a real app, you would process payment here
    // For this example, we'll just collect the info and store it in formData
    updateFormData({
      paymentInfo: {
        ...cardInfo,
        // Mask card number for security in state
        cardNumber: '**** **** **** ' + cardInfo.cardNumber.slice(-4)
      }
    });

    nextStep();
  };

  // Calculate pricing based on selected tier
  const getTierPricing = () => {
    switch(formData.tier) {
      case 'in-person-training':
        return {
          name: 'In-Person Training',
          price: '$70.00',
          recurring: false,
          frequency: 'per session'
        };
      case 'online-coaching':
        return {
          name: 'Online Coaching',
          price: '$199.00',
          recurring: true,
          frequency: 'monthly'
        };
      case 'complete-transformation':
        return {
          name: 'Complete Transformation',
          price: '$199.00',
          recurring: true,
          frequency: 'monthly + $60 per session'
        };
      default:
        return {
          name: 'Selected Plan',
          price: '$0.00',
          recurring: false,
          frequency: ''
        };
    }
  };

  const pricingInfo = getTierPricing();

  return (
    <form className="signup-form" onSubmit={handleSubmit}>
      <h3>Payment Information</h3>

      {error && <div className="error-message">{error}</div>}

      <div className="payment-summary">
        <h4>Order Summary</h4>
        <div className="summary-item">
          <span className="item-name">{pricingInfo.name}</span>
          <span className="item-price">{pricingInfo.price}</span>
        </div>
        {pricingInfo.recurring && (
          <p className="recurring-notice">
            You will be charged {pricingInfo.price} {pricingInfo.frequency}
          </p>
        )}
      </div>

      <div className="form-group">
```

```
          <label htmlFor="cardName">
            <i className="fas fa-user"></i> Name on Card <span
className="required">*</span>
          </label>
          <input
            type="text"
            id="cardName"
            name="cardName"
            value={cardInfo.cardName}
            onChange={handleChange}
            placeholder="Name as it appears on card"
            required
          />
        </div>

        <div className="form-group">
          <label htmlFor="cardNumber">
            <i className="fas fa-credit-card"></i> Card Number <span
className="required">*</span>
          </label>
          <input
            type="text"
            id="cardNumber"
            name="cardNumber"
            value={cardInfo.cardNumber}
            onChange={handleChange}
            placeholder="XXXX XXXX XXXX XXXX"
            maxLength={19}
            required
          />
        </div>

        <div className="form-row">
          <div className="form-group half">
            <label htmlFor="expiry">
              <i className="fas fa-calendar"></i> Expiry Date <span
className="required">*</span>
            </label>
            <input
              type="text"
              id="expiry"
              name="expiry"
              value={cardInfo.expiry}
              onChange={handleChange}
              placeholder="MM/YY"
              maxLength={5}
              required
            />
          </div>
          <div className="form-group half">
            <label htmlFor="cvv">
              <i className="fas fa-lock"></i> CVV <span className="required">*
</span>
            </label>
            <input
              type="text"
              id="cvv"
              name="cvv"
              value={cardInfo.cvv}
              onChange={handleChange}
              placeholder="123"
              maxLength={4}
              required
            />
          </div>
        </div>

        <div className="button-row">
          <button
            type="button"
            className="btn-secondary"
            onClick={prevStep}
          >
            <i className="fas fa-arrow-left"></i> Back
          </button>
          <button type="submit" className="btn-primary-enhanced">
            Review Order <i className="fas fa-arrow-right"></i>
          </button>
        </div>
      </form>
    );
  }

  export default PaymentStep;
```

Confirmation Step

```
// src/react/signup/components/ConfirmationStep.jsx
import React from 'react';

function ConfirmationStep({ formData, handleSubmit, prevStep, isSubmitting, error
}) {
  // Format selected tier for display
  const formatTier = (tierId) => {
    switch(tierId) {
      case 'in-person-training':
        return {
          name: 'In-Person Training',
          price: '$70 per session'
        };
      case 'online-coaching':
        return {
          name: 'Online Coaching',
          price: '$199 per month'
        };
      case 'complete-transformation':
        return {
          name: 'Complete Transformation',
          price: '$199 per month + $60 per training session'
        };
      default:
        return {
          name: 'Unknown tier',
          price: 'Price not available'
        };
    }
  };

  const tierInfo = formatTier(formData.tier);

  return (
    <div className="signup-form">
      <h3>Review and Confirm</h3>

      {error && <div className="error-message">{error}</div>}

      <div className="review-section">
        <h4>Account Information</h4>
        <div className="review-item">
          <span className="item-label">Name:</span>
          <span className="item-value">{formData.name}</span>
        </div>
        <div className="review-item">
          <span className="item-label">Email:</span>
          <span className="item-value">{formData.email}</span>
        </div>
        {formData.phone && (
          <div className="review-item">
            <span className="item-label">Phone:</span>
            <span className="item-value">{formData.phone}</span>
          </div>
        )}
      </div>

      <div className="review-section">
        <h4>Selected Plan</h4>
        <div className="review-item">
          <span className="item-label">Service:</span>
          <span className="item-value">{tierInfo.name}</span>
        </div>
        <div className="review-item">
          <span className="item-label">Price:</span>
          <span className="item-value">{tierInfo.price}</span>
        </div>
      </div>

      <div className="review-section">
        <h4>Payment Information</h4>
        <div className="review-item">
          <span className="item-label">Card:</span>
          <span className="item-value">{formData.paymentInfo?.cardNumber || 'Not
provided'}</span>
        </div>
      </div>

      <div className="terms-agreement">
        <label className="checkbox-container">
          <input type="checkbox" required />
          <span className="checkmark"></span>
          I agree to the <a href="#" target="_blank">Terms of Service</a> and <a
href="#" target="_blank">Privacy Policy</a>
        </label>
      </div>

      <div className="next-steps-info">
        <h4>What happens next?</h4>
        <p>
          After completing signup, you will be directed to your dashboard where
you can schedule your 30-minute consultation to discuss your fitness goals and
create a personalized plan.
        </p>
      </div>

      <div className="button-row">
        <button
          type="button"
          className="btn-secondary"
          onClick={prevStep}
          disabled={isSubmitting}
        >
          <i className="fas fa-arrow-left"></i> Back
        </button>
        <button
          type="button"
          className="btn-primary-enhanced"
          onClick={handleSubmit}
          disabled={isSubmitting}
        >
          {isSubmitting ? 'Creating Account...' : 'Complete Signup'}
          {!isSubmitting && <i className="fas fa-check"></i>}
        </button>
      </div>
    </div>
  );
}

export default ConfirmationStep;
```

Styling

Add CSS styles to match your existing site's look and feel:

```css
/* src/react/signup/signup.css */
/* Progress steps */
.progress-steps {
  display: flex;
  align-items: center;
  margin-bottom: 2rem;
  justify-content: center;
}

.step {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.step-number {
  width: 32px;
  height: 32px;
  border-radius: 50%;
  background-color: #e0e0e0;
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight: 600;
  margin-bottom: 5px;
}

.step.active .step-number {
  background-color: var(--primary, #4caf50);
  color: white;
}

.step-label {
  font-size: 14px;
  color: #666;
}

.step.active .step-label {
  color: var(--primary, #4caf50);
  font-weight: 600;
}

.step-connector {
  height: 2px;
  flex-grow: 1;
  background-color: #e0e0e0;
  margin: 0 10px;
  margin-bottom: 20px;
  min-width: 40px;
}

.step-connector.active {
  background-color: var(--primary, #4caf50);
}

/* Form styling */
.signup-form {
  max-width: 600px;
  margin: 0 auto;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  background-color: white;
}

.service-tiers {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  margin: 2rem 0;
}

.service-tier {
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  padding: 1.5rem;
  flex: 1;
  min-width: 250px;
  cursor: pointer;
  transition: all 0.3s ease;
}

.service-tier.selected {
  border-color: var(--primary, #4caf50);
  background-color: rgba(76, 175, 80, 0.05);
}

.tier-header {
  display: flex;
  align-items: center;
  margin-bottom: 1rem;
}

.tier-radio {
  margin-right: 10px;
}

.tier-title {
  font-weight: 600;
  font-size: 1.2rem;
  margin: 0;
}

.tier-price {
  font-weight: 700;
  font-size: 1.5rem;
  margin: 1rem 0;
  color: var(--primary-dark, #388e3c);
}

.tier-description {
  color: #666;
  margin-bottom: 1rem;
}

/* Button row */
.button-row {
  display: flex;
  justify-content: space-between;
  margin-top: 2rem;
}

.btn-secondary {
  background-color: transparent;
  color: var(--primary, #4caf50);
  border: 1px solid var(--primary, #4caf50);
}

.error-message {
  background-color: #ffebee;
  color: #d32f2f;
  padding: 10px;
  border-radius: 4px;
  margin-bottom: 1rem;
  border-left: 4px solid #d32f2f;
}

/* Review step styling */
.review-section {
  margin-bottom: 1.5rem;
  padding-bottom: 1rem;
  border-bottom: 1px solid #eee;
}

.review-section h4 {
  margin-bottom: 0.5rem;
  color: #333;
}

.review-item {
  display: flex;
  justify-content: space-between;
  margin-bottom: 0.5rem;
}

.item-label {
  font-weight: 500;
  color: #666;
}
```

```css
.item-value {
  color: #333;
}

.payment-summary {
  background-color: #f9f9f9;
  border-radius: 4px;
  padding: 1rem;
  margin-bottom: 1.5rem;
}

.summary-item {
  display: flex;
  justify-content: space-between;
  font-weight: 600;
}

.recurring-notice {
  margin-top: 0.5rem;
  font-size: 0.9rem;
  color: #666;
}

/* Form row for splitting inputs */
.form-row {
  display: flex;
  gap: 1rem;
}

.form-group.half {
  flex: 1;
}

/* Terms agreement styling */
.terms-agreement {
  margin: 1.5rem 0;
}

.checkbox-container {
  display: flex;
  align-items: center;
}

.next-steps-info {
  background-color: #e8f5e9;
  padding: 1rem;
  border-radius: 4px;
  margin: 1rem 0;
}
```

Building and Testing

Build the React Component

Run the webpack development build to create the React bundle:

```
npm run dev
```

This will:

1. Start webpack in watch mode
2. Compile your React code
3. Create the `dist/signup-bundle.js` file
4. Automatically update the bundle when you make changes

Testing the Implementation

1. Open your signup page in a web browser
2. Verify that the multi-step form appears and functions correctly
3. Test each step of the signup process:
   - Account information entry
   - Service tier selection
   - Payment information entry
   - Review and confirmation
4. Complete a test signup to verify that:
   - The user is created in Firebase
   - The tier information is saved correctly
   - The user is redirected to the success page or dashboard

Debugging Tips

If you encounter issues:

1. Check the browser's developer console for errors
2. Verify that all Firebase scripts are loaded correctly
3. Ensure the React bundle is being loaded in the HTML
4. Check that your Firebase configuration matches between your existing code and the new configuration

Future Enhancements

Once the signup page is working, you can follow the same pattern to convert:

1. Login page
2. User dashboard
3. Admin dashboard

For each page:

1. Create a new entry point and components
2. Update the webpack configuration to build multiple bundles
3. Modify the HTML to include the React mount point and bundle

Firestore Security Rules

If you're using Firestore security rules, ensure they allow the `tier` field to be written:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
  }
}
```

Migration Strategy

This section outlines the recommended approach for gradually migrating your site to React with modern Firebase SDK.

Recommended Migration Path

1. **Start with High-Value, Complex Pages**
   - Your plan to convert signup first, then login, then dashboards is ideal
   - These pages benefit most from React's state management and component structure
   - The existing implementation of these pages often requires the most maintenance

2. **Upgrade Path for Pages**

Page Type	Current Status	Recommended Approach
Signup Page	Converting now	React + modular Firebase SDK
Login Page	Convert next	React + modular Firebase SDK
Dashboards	Future conversion	React + modular Firebase SDK
"Send a message" form	HTML/CSS/JS	Keep as-is or gradually migrate to modular SDK
Content pages (About, etc.)	HTML/CSS/JS	Low priority for conversion - can stay as-is

3. **Technical Approach for Each Page**
   - Convert one page completely before moving to the next
   - Create separate webpack entry points for each React page
   - Use the same shared Firebase configuration

Benefits of This Approach

1. **Risk Reduction**
   - Each page can be converted independently

    - Existing functionality remains working throughout the process
    - You can test thoroughly before deploying each conversion

2. **Technical Advantages**
    - Modern, maintainable code for complex features
    - Better performance through code splitting and tree shaking
    - Improved developer experience for new features

3. **Business Value**
    - Focus development effort on highest-impact pages first
    - Deliver improved user experience where it matters most
    - Gradual learning curve for developers

This hybrid approach gives you the benefits of modern architecture without requiring a complete rewrite. As you get more comfortable with React and the modular Firebase SDK, you can continue converting more pages at your own pace.

Implementation Checklist

Use this checklist to track your progress when implementing the React signup page:

Environment Setup

- [ ] Install Node.js and NPM
- [ ] Initialize NPM in project directory
- [ ] Install Firebase SDK v12.1.0 and React
- [ ] Install development dependencies (webpack, babel, etc.)
- [ ] Install Firebase CLI v14.11.1 (if deploying to Firebase Hosting)

Project Structure

- [ ] Create folder structure for React components
- [ ] Create webpack configuration file
- [ ] Set up babel configuration
- [ ] Create shared Firebase config file

Firebase Configuration

- [ ] Copy Firebase config from existing auth.js
- [ ] Set up modular Firebase SDK imports
- [ ] Create user authentication functions
- [ ] Test Firebase connection

React Components

- [ ] Create React entry point (index.js)
- [ ] Create main SignupForm component
- [ ] Create AccountInfoStep component
- [ ] Create ServiceTierStep component

- [ ] Create PaymentStep component
- [ ] Create ConfirmationStep component
- [ ] Add styling (CSS)

Integration

- [ ] Update signup.html with React mounting point
- [ ] Configure event dispatching to work with existing code
- [ ] Build React bundle with webpack
- [ ] Test React components mounting in existing page

Testing & Deployment

- [ ] Test form validation
- [ ] Test user creation with tier information
- [ ] Test payment flow (or mock if not implementing real payments yet)
- [ ] Test integration with existing authentication system
- [ ] Test redirect to dashboard after signup
- [ ] Optimize bundle size for production
- [ ] Deploy to production environment

Next Steps After Implementation

- [ ] Convert login page to React
- [ ] Convert dashboard to React
- [ ] Set up more sophisticated state management if needed
- [ ] Consider adding unit tests for React components

Conclusion

This guide has walked you through converting your signup page to React while keeping the rest of your site as HTML/CSS/JS. By using the latest Firebase modular SDK and a hybrid architecture approach, you can gradually modernize your codebase without disrupting existing functionality.

The implementation provides a seamless user experience while establishing a foundation for future enhancements. As you continue your migration journey, you can apply the same patterns to convert your login page and dashboards, always maintaining compatibility with your existing pages.

Remember that Firebase fully supports this incremental approach - you do not need to convert your entire application to React or to the modular SDK all at once.