💻 Complete Code Explanation - Part 3

Page Components (SignUp, SignIn, EmployeeList, EmployeeDetail)

---

12. src/pages/SignUp.tsx

Purpose: User registration page। নতুন account তৈরি করার জন্য।

```typescript
```

```
import { useState, FormEvent } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import {
  Container,
  Box,
  TextField,
  Button,
  Typography,
  Alert,
  Paper,
} from '@mui/material';
import { authAPI } from '../services/api';

function SignUp() {
  // State for form fields
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  // State for UI feedback
  const [error, setError] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
    // Prevent default form submission
    e.preventDefault();

    // Clear previous errors
    setError(null);
```

```
// Validation
if (!name || !email || !password) {
  setError('全てのフィールドを入力してください');
  return;
}

if (!email.includes('@')) {
  setError('有効なメールアドレスを入力してください');
  return;
}

if (password.length < 8) {
  setError('パスワードは8文字以上である必要があります');
  return;
}

// API call
setLoading(true);

try {
  const response = await authAPI.signUp({
    Name: name,
    Email: email,
    Password: password,
  });

  if (response.data.message === 'Registration successful.') {
    // Success: navigate to signin
    navigate('/signin');
  } else {
    setError(response.data.message);
  }
  } catch (err: any) {
    setError(err.response?.data?.message || '登録に失敗しました');
  } finally {
    setLoading(false);
  }
};

return (
  <Container maxWidth="sm">
    <Box
      sx={{
        marginTop: 8,
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
      }}
    >
      <Paper elevation={3} sx={{ padding: 4, width: '100%' }}>
        <Typography component="h1" variant="h5" align="center" gutterBottom>
          新規登録
        </Typography>

        {error && (
          <Alert severity="error" sx={{ mb: 2 }}>
            {error}
          </Alert>
        )}

        <Box component="form" onSubmit={handleSubmit} noValidate>
          <TextField
            margin="normal"
            required
  fullWidth
  id="name"
  label="氏名"
  name="name"
  autoComplete="name"
  autoFocus
  value={name}
  onChange={(e) => setName(e.target.value)}
/>

<TextField
  margin="normal"
  required
  fullWidth
  id="email"
  label="メールアドレス"
  name="email"
  autoComplete="email"
  value={email}
  onChange={(e) => setEmail(e.target.value)}
/>

<TextField
  margin="normal"
  required
  fullWidth
  name="password"
  label="パスワード"
  type="password"
  id="password"
  autoComplete="new-password"
  value={password}
```

```
            onChange={(e) => setPassword(e.target.value)}
          />

          <Button
            type="submit"
            fullWidth
            variant="contained"
            sx={{ mt: 3, mb: 2 }}
            disabled={loading}
          >
            {loading ? '登録中...' : '登録'}
          </Button>

          <Box sx={{ textAlign: 'center' }}>
            <Link to="/signin" style={{ textDecoration: 'none' }}>
              <Typography variant="body2" color="primary">
                既にアカウントをお持ちですか？ログイン
              </Typography>
            </Link>
          </Box>
        </Box>
      </Paper>
    </Box>
  </Container>
 );
}

export default SignUp;
```

Detailed Line-by-Line Explanation:

This component handles user registration with validation, API communication, and navigation.

Key Concepts:

- Controlled components (form fields tied to state)
- Client-side validation before API call
- Async/await for API calls
- Error handling and user feedback
- Conditional rendering (error alerts, loading states)

For complete line-by-line breakdown with examples and explanations of every single line, see the detailed markdown version.

---

13. src/pages/SignIn.tsx

Purpose: Authentication page for existing users

```typescript

```

```
import { useState, FormEvent } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import {
  Container,
  Box,
  TextField,
  Button,
  Typography,
  Alert,
  Paper,
} from '@mui/material';
import { authAPI } from '../services/api';
import { setAuthData } from '../utils/auth';

function SignIn() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    setError(null);

    if (!email || !password) {
      setError('メールアドレスとパスワードを入力してください');
      return;
    }

    setLoading(true);
    try {
      const response = await authAPI.signIn(email, password);

      if (response.data.message === 'Authentication successful.') {
        setAuthData(email);
        navigate('/employees');
      } else {
        setError(response.data.message);
      }
    } catch (err: any) {
      setError(err.response?.data?.message || 'ログインに失敗しました');
    } finally {
      setLoading(false);
    }
  };

  return (
    <Container maxWidth="sm">
      {/* Similar structure to SignUp but with login-specific elements */}
      {/* Email and Password fields only */}
      {/* Success navigates to /employees instead of /signin */}
      {/* Link goes to /signup instead of /signin */}
    </Container>
  );
}

export default SignIn;
```

Key Differences from SignUp:

1. Only 2 fields (email, password) - no name field

2. Simpler validation (just check not empty)
3. Calls `setAuthData()` on success to save login state
4. Navigates to `/employees` after successful login
5. Link points to signup page for new users

Due to length constraints, I'll create a separate document for the complex EmployeeList and EmployeeDetail components. Let me continue...