React.js Material-UI

.

```
frontend/
├── public/
│   ├── index.html
│   ├── favicon.ico
│   ├── locales/
│   │   ├── ar/
│   │   │   └── translation.json
│   │   └── en/
│   │       └── translation.json
│   └── assets/
│       └── images/
├── src/
│   ├── components/
│   │   ├── common/
│   │   ├── auth/
│   │   ├── dashboard/
│   │   ├── analysis/
│   │   └── reports/
│   ├── pages/
│   │   ├── Home.js
│   │   ├── Login.js
│   │   ├── Register.js
│   │   ├── Dashboard.js
│   │   ├── Analysis.js
│   │   └── Reports.js
│   ├── services/
│   │   ├── api.js
│   │   ├── auth.js
│   │   └── analysis.js
│   ├── utils/
│   │   ├── i18n.js
│   │   └── theme.js
│   ├── App.js
```

```
│   └── index.js
└── package.json
```

React :

```
npx create-react-app frontend
cd frontend
npm install @mui/material @mui/icons-material @emotion/react @emotion/styled
npm install react-router-dom i18next react-i18next i18next-browser-languagedetector
npm install axios formik yup react-dropzone recharts
```

1. (Home.js)

:

```
import React from 'react';
import {
  Container,
  Typography,
  Button,
  Grid,
  Card,
  CardContent,
  CardMedia,
  Box
} from '@mui/material';
import { useTranslation } from 'react-i18next';
import { Link } from 'react-router-dom';

const Home = () => {
  const { t } = useTranslation();

  const features = [
    {
      title: t('home.features.detection.title'),
      description: t('home.features.detection.description'),
      image: '/assets/images/crack-detection.jpg'
    },
    {
      title: t('home.features.analysis.title'),
```

```
    description: t('home.features.analysis.description'),
    image: '/assets/images/analysis.jpg'
  },
  {
    title: t('home.features.reports.title'),
    description: t('home.features.reports.description'),
    image: '/assets/images/reports.jpg'
  }
];

return (
  <Container maxWidth="lg">
    {/* Hero Section */}
    <Box
      sx={{
        textAlign: 'center',
        py: 8,
        backgroundImage: 'url(/assets/images/hero-bg.jpg)',
        backgroundSize: 'cover',
        backgroundPosition: 'center',
        borderRadius: 2,
        mb: 6,
        color: 'white'
      }}
    >
      <Typography variant="h2" component="h1" gutterBottom>
        {t('home.hero.title')}
      </Typography>
      <Typography variant="h5" component="h2" gutterBottom>
        {t('home.hero.subtitle')}
      </Typography>
      <Button
        component={Link}
        to="/register"
        variant="contained"
        size="large"
        sx={{ mt: 4 }}
      >
        {t('home.hero.cta')}
      </Button>
    </Box>

    {/* Features Section */}
    <Typography variant="h4" component="h2" align="center" gutterBottom>
      {t('home.featuresSection.title')}
    </Typography>
    <Grid container spacing={4} sx={{ mt: 2 }}>
      {features.map((feature, index) => (
        <Grid item xs={12} md={4} key={index}>
          <Card sx={{ height: '100%' }}>
            <CardMedia
              component="img"
```

```
            height="200"
            image={feature.image}
            alt={feature.title}
          />
          <CardContent>
            <Typography variant="h5" component="h3" gutterBottom>
              {feature.title}
            </Typography>
            <Typography variant="body1">
              {feature.description}
            </Typography>
          </CardContent>
        </Card>
      </Grid>
    ))}
  </Grid>

  {/* How It Works Section */}
  <Box sx={{ mt: 8, mb: 6 }}>
    <Typography variant="h4" component="h2" align="center" gutterBottom>
      {t('home.howItWorks.title')}
    </Typography>
    <Grid container spacing={2} sx={{ mt: 4 }}>
      <Grid item xs={12} md={4}>
        <Box sx={{ textAlign: 'center' }}>
          <img src="/assets/images/upload.svg" alt="Upload" height="80" />
          <Typography variant="h6" component="h3" sx={{ mt: 2 }}>
            {t('home.howItWorks.step1.title')}
          </Typography>
          <Typography>
            {t('home.howItWorks.step1.description')}
          </Typography>
        </Box>
      </Grid>
      <Grid item xs={12} md={4}>
        <Box sx={{ textAlign: 'center' }}>
          <img src="/assets/images/analyze.svg" alt="Analyze" height="80" />
          <Typography variant="h6" component="h3" sx={{ mt: 2 }}>
            {t('home.howItWorks.step2.title')}
          </Typography>
          <Typography>
            {t('home.howItWorks.step2.description')}
          </Typography>
        </Box>
      </Grid>
      <Grid item xs={12} md={4}>
        <Box sx={{ textAlign: 'center' }}>
          <img src="/assets/images/report.svg" alt="Report" height="80" />
          <Typography variant="h6" component="h3" sx={{ mt: 2 }}>
            {t('home.howItWorks.step3.title')}
          </Typography>
          <Typography>
```

```
            {t('home.howItWorks.step3.description')}
          </Typography>
        </Box>
      </Grid>
    </Grid>
  </Box>

  {/* Call to Action */}
  <Box sx={{ textAlign: 'center', py: 6 }}>
    <Typography variant="h4" component="h2" gutterBottom>
      {t('home.cta.title')}
    </Typography>
    <Typography variant="body1" gutterBottom sx={{ mb: 4 }}>
      {t('home.cta.description')}
    </Typography>
    <Button
      component={Link}
      to="/register"
      variant="contained"
      size="large"
    >
      {t('home.cta.button')}
    </Button>
  </Box>
 </Container>
 );
};

export default Home;
```

2. (Login.js)

```
 import React, { useState } from 'react';
import {
 Container,
 Typography,
 TextField,
 Button,
 Paper,
 Box,
 Link,
 Alert
} from '@mui/material';
import { useTranslation } from 'react-i18next';
import { Link as RouterLink, useNavigate } from 'react-router-dom';
import { useFormik } from 'formik';
import * as Yup from 'yup';
// import { login } from '../services/auth';

const Login = () => {
```

```
  const { t } = useTranslation();
  const navigate = useNavigate();
  const [error, setError] = useState('');

  const validationSchema = Yup.object({
    email: Yup.string()
      .email(t('validation.email'))
      .required(t('validation.required')),
    password: Yup.string()
      .required(t('validation.required'))
  });

  const formik = useFormik({
    initialValues: {
      email: '',
      password: ''
    },
    validationSchema,
    onSubmit: async (values) => {
      try {
        // Placeholder for actual login API call
        // const response = await login(values.email, values.password);
        console.log('Login submitted:', values);
        // Simulate successful login
        localStorage.setItem('isAuthenticated', 'true');
        navigate('/dashboard');
      } catch (err) {
        setError(t('login.error'));
      }
    }
  });

  return (
    <Container maxWidth="sm">
      <Paper elevation={3} sx={{ p: 4, mt: 8 }}>
        <Typography variant="h4" component="h1" align="center" gutterBottom>
          {t('login.title')}
        </Typography>

        {error && (
          <Alert severity="error" sx={{ mb: 3 }}>
            {error}
          </Alert>
        )}

        <form onSubmit={formik.handleSubmit}>
          <TextField
            fullWidth
            id="email"
            name="email"
            label={t('login.email')}
            value={formik.values.email}
```

```
          onChange={formik.handleChange}
          error={formik.touched.email && Boolean(formik.errors.email)}
          helperText={formik.touched.email && formik.errors.email}
          margin="normal"
        />

        <TextField
          fullWidth
          id="password"
          name="password"
          label={t('login.password')}
          type="password"
          value={formik.values.password}
          onChange={formik.handleChange}
          error={formik.touched.password && Boolean(formik.errors.password)}
          helperText={formik.touched.password && formik.errors.password}
          margin="normal"
        />

        <Button
          type="submit"
          variant="contained"
          fullWidth
          size="large"
          sx={{ mt: 3 }}
        >
          {t('login.submit')}
        </Button>
      </form>

      <Box sx={{ mt: 2, textAlign: 'center' }}>
        <Link component={RouterLink} to="/forgot-password">
          {t('login.forgotPassword')}
        </Link>
      </Box>

      <Box sx={{ mt: 3, textAlign: 'center' }}>
        <Typography variant="body2">
          {t('login.noAccount')}{' '}
          <Link component={RouterLink} to="/register">
            {t('login.register')}
          </Link>
        </Typography>
      </Box>
    </Paper>
  </Container>
 );
};

export default Login;
```

3. (Register.js)

```
 import React, { useState } from 'react';
import {
  Container,
  Typography,
  TextField,
  Button,
  Paper,
  Box,
  Link,
  Alert,
  FormControl,
  InputLabel,
  Select,
  MenuItem
} from '@mui/material';
import { useTranslation } from 'react-i18next';
import { Link as RouterLink, useNavigate } from 'react-router-dom';
import { useFormik } from 'formik';
import * as Yup from 'yup';
// import { register } from '../services/auth';

const Register = () => {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const [error, setError] = useState('');

  const validationSchema = Yup.object({
    fullName: Yup.string()
      .required(t('validation.required')),
    email: Yup.string()
      .email(t('validation.email'))
      .required(t('validation.required')),
    password: Yup.string()
      .min(8, t('validation.passwordLength'))
      .required(t('validation.required')),
    confirmPassword: Yup.string()
      .oneOf([Yup.ref('password'), null], t('validation.passwordMatch'))
      .required(t('validation.required')),
    userType: Yup.string()
      .required(t('validation.required'))
  });

  const formik = useFormik({
    initialValues: {
      fullName: '',
      email: '',
      password: '',
      confirmPassword: '',
      userType: ''
```

```
  },
  validationSchema,
  onSubmit: async (values) => {
    try {
      // Placeholder for actual register API call
      // const response = await register(values);
      console.log('Registration submitted:', values);
      navigate('/login');
    } catch (err) {
      setError(t('register.error'));
    }
  }
});

return (
  <Container maxWidth="sm">
    <Paper elevation={3} sx={{ p: 4, mt: 8, mb: 8 }}>
      <Typography variant="h4" component="h1" align="center" gutterBottom>
        {t('register.title')}
      </Typography>

      {error && (
        <Alert severity="error" sx={{ mb: 3 }}>
          {error}
        </Alert>
      )}

      <form onSubmit={formik.handleSubmit}>
        <TextField
          fullWidth
          id="fullName"
          name="fullName"
          label={t('register.fullName')}
          value={formik.values.fullName}
          onChange={formik.handleChange}
          error={formik.touched.fullName && Boolean(formik.errors.fullName)}
          helperText={formik.touched.fullName && formik.errors.fullName}
          margin="normal"
        />

        <TextField
          fullWidth
          id="email"
          name="email"
          label={t('register.email')}
          value={formik.values.email}
          onChange={formik.handleChange}
          error={formik.touched.email && Boolean(formik.errors.email)}
          helperText={formik.touched.email && formik.errors.email}
          margin="normal"
        />
```

```
        <TextField
          fullWidth
          id="password"
          name="password"
          label={t('register.password')}
          type="password"
          value={formik.values.password}
          onChange={formik.handleChange}
          error={formik.touched.password && Boolean(formik.errors.password)}
          helperText={formik.touched.password && formik.errors.password}
          margin="normal"
        />

        <TextField
          fullWidth
          id="confirmPassword"
          name="confirmPassword"
          label={t('register.confirmPassword')}
          type="password"
          value={formik.values.confirmPassword}
          onChange={formik.handleChange}
          error={formik.touched.confirmPassword &&
Boolean(formik.errors.confirmPassword)}
          helperText={formik.touched.confirmPassword &&
formik.errors.confirmPassword}
          margin="normal"
        />

        <FormControl fullWidth margin="normal">
          <InputLabel id="userType-label">{t('register.userType')}</InputLabel>
          <Select
            labelId="userType-label"
            id="userType"
            name="userType"
            value={formik.values.userType}
            onChange={formik.handleChange}
            error={formik.touched.userType && Boolean(formik.errors.userType)}
            label={t('register.userType')}
          >
            <MenuItem value="engineer">{t('register.userTypes.engineer')}</
MenuItem>
            <MenuItem value="inspector">{t('register.userTypes.inspector')}</
MenuItem>
            <MenuItem value="company">{t('register.userTypes.company')}</
MenuItem>
          </Select>
          {formik.touched.userType && formik.errors.userType && (
            <Typography color="error" variant="caption">
              {formik.errors.userType}
            </Typography>
          )}
        </FormControl>
```

```
          <Button
            type="submit"
            variant="contained"
            fullWidth
            size="large"
            sx={{ mt: 3 }}
          >
            {t('register.submit')}
          </Button>
        </form>

        <Box sx={{ mt: 3, textAlign: 'center' }}>
          <Typography variant="body2">
            {t('register.haveAccount')}{' '}
            <Link component={RouterLink} to="/login">
              {t('register.login')}
            </Link>
          </Typography>
        </Box>
      </Paper>
    </Container>
  );
};

export default Register;
```

4. (Dashboard.js)

```
 import React from 'react';
import {
  Container,
  Typography,
  Grid,
  Paper,
  Box,
  Card,
  CardContent,
  CardHeader,
  List,
  ListItem,
  ListItemText,
  Divider,
  Button
} from '@mui/material';
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
```

```
  CartesianGrid,
  Tooltip,
  Legend,
  PieChart,
  Pie,
  Cell
} from 'recharts';
import { useTranslation } from 'react-i18next';
import { Link } from 'react-router-dom';

const Dashboard = () => {
  const { t } = useTranslation();

  // Sample data for charts
  const defectTypeData = [
    { name: t('dashboard.charts.defectTypes.cracks'), value: 65 },
    { name: t('dashboard.charts.defectTypes.corrosion'), value: 15 },
    { name: t('dashboard.charts.defectTypes.exposedBars'), value: 10 },
    { name: t('dashboard.charts.defectTypes.dampness'), value: 10 }
  ];

  const monthlyAnalysisData = [
    { name: t('months.jan'), analyses: 12 },
    { name: t('months.feb'), analyses: 19 },
    { name: t('months.mar'), analyses: 25 },
    { name: t('months.apr'), analyses: 32 },
    { name: t('months.may'), analyses: 20 },
    { name: t('months.jun'), analyses: 18 }
  ];

  const recentProjects = [
    { id: 1, name: t('dashboard.recentProjects.project1'), date: '2025-04-15', status:
t('status.active') },
    { id: 2, name: t('dashboard.recentProjects.project2'), date: '2025-04-10', status:
t('status.completed') },
    { id: 3, name: t('dashboard.recentProjects.project3'), date: '2025-04-05', status:
t('status.active') }
  ];

  const recentAnalyses = [
    { id: 1, name: t('dashboard.recentAnalyses.analysis1'), date: '2025-04-20', defects:
5 },
    { id: 2, name: t('dashboard.recentAnalyses.analysis2'), date: '2025-04-18', defects:
2 },
    { id: 3, name: t('dashboard.recentAnalyses.analysis3'), date: '2025-04-15', defects:
8 }
  ];

  const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042'];

  return (
    <Container maxWidth="lg" sx={{ mt: 4, mb: 4 }}>
```

```
<Typography variant="h4" component="h1" gutterBottom>
 {t('dashboard.title')}
</Typography>

{/* Summary Cards */}
<Grid container spacing={3} sx={{ mb: 4 }}>
 <Grid item xs={12} sm={6} md={3}>
  <Paper
   sx={{
    p: 2,
    display: 'flex',
    flexDirection: 'column',
    height: 140,
    bgcolor: '#bbdefb'
   }}
  >
   <Typography variant="h6" gutterBottom>
    {t('dashboard.summary.projects')}
   </Typography>
   <Typography variant="h3" component="div" sx={{ flexGrow: 1 }}>
    12
   </Typography>
  </Paper>
 </Grid>
 <Grid item xs={12} sm={6} md={3}>
  <Paper
   sx={{
    p: 2,
    display: 'flex',
    flexDirection: 'column',
    height: 140,
    bgcolor: '#c8e6c9'
   }}
  >
   <Typography variant="h6" gutterBottom>
    {t('dashboard.summary.analyses')}
   </Typography>
   <Typography variant="h3" component="div" sx={{ flexGrow: 1 }}>
    48
   </Typography>
  </Paper>
 </Grid>
 <Grid item xs={12} sm={6} md={3}>
  <Paper
   sx={{
    p: 2,
    display: 'flex',
    flexDirection: 'column',
    height: 140,
    bgcolor: '#fff9c4'
   }}
  >
```

```
        <Typography variant="h6" gutterBottom>
          {t('dashboard.summary.defects')}
        </Typography>
        <Typography variant="h3" component="div" sx={{ flexGrow: 1 }}>
          156
        </Typography>
      </Paper>
    </Grid>
    <Grid item xs={12} sm={6} md={3}>
      <Paper
        sx={{
          p: 2,
          display: 'flex',
          flexDirection: 'column',
          height: 140,
          bgcolor: '#ffccbc'
        }}
      >
        <Typography variant="h6" gutterBottom>
          {t('dashboard.summary.reports')}
        </Typography>
        <Typography variant="h3" component="div" sx={{ flexGrow: 1 }}>
          24
        </Typography>
      </Paper>
    </Grid>
  </Grid>

  {/* Charts */}
  <Grid container spacing={3} sx={{ mb: 4 }}>
    <Grid item xs={12} md={8}>
      <Paper sx={{ p: 2 }}>
        <Typography variant="h6" gutterBottom>
          {t('dashboard.charts.monthlyAnalysis')}
        </Typography>
        <Box sx={{ height: 300 }}>
          <BarChart
            width={600}
            height={300}
            data={monthlyAnalysisData}
            margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
          >
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Bar dataKey="analyses" fill="#8884d8" />
          </BarChart>
        </Box>
      </Paper>
    </Grid>
```

```
        <Grid item xs={12} md={4}>
          <Paper sx={{ p: 2 }}>
            <Typography variant="h6" gutterBottom>
              {t('dashboard.charts.defectDistribution')}
            </Typography>
            <Box sx={{ height: 300, display: 'flex', justifyContent: 'center' }}>
              <PieChart width={300} height={300}>
                <Pie
                  data={defectTypeData}
                  cx="50%"
                  cy="50%"
                  labelLine={false}
                  outerRadius={100}
                  fill="#8884d8"
                  dataKey="value"
                  label={({ name, percent }) => `${name} ${(percent * 100).toFixed(0)}%`}
                >
                  {defectTypeData.map((entry, index) => (
                    <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
                  ))}
                </Pie>
                <Tooltip />
              </PieChart>
            </Box>
          </Paper>
        </Grid>
      </Grid>

      {/* Recent Activity */}
      <Grid container spacing={3}>
        <Grid item xs={12} md={6}>
          <Card>
            <CardHeader title={t('dashboard.recentProjects.title')} />
            <CardContent>
              <List>
                {recentProjects.map((project, index) => (
                  <React.Fragment key={project.id}>
                    <ListItem>
                      <ListItemText
                        primary={project.name}
                        secondary={`${t('dashboard.date')}: ${project.date} | $
{t('dashboard.status')}: ${project.status}`}
                      />
                    </ListItem>
                    {index < recentProjects.length - 1 && <Divider />}
                  </React.Fragment>
                ))}
              </List>
              <Button
                component={Link}
                to="/projects"
                variant="outlined"
```

```
            fullWidth
            sx={{ mt: 2 }}
          >
            {t('dashboard.viewAll')}
          </Button>
        </CardContent>
      </Card>
    </Grid>
    <Grid item xs={12} md={6}>
      <Card>
        <CardHeader title={t('dashboard.recentAnalyses.title')} />
        <CardContent>
          <List>
            {recentAnalyses.map((analysis, index) => (
              <React.Fragment key={analysis.id}>
                <ListItem>
                  <ListItemText
                    primary={analysis.name}
                    secondary={`${t('dashboard.date')}: ${analysis.date} | $
{t('dashboard.defectsFound')}: ${analysis.defects}`}
                  />
                </ListItem>
                {index < recentAnalyses.length - 1 && <Divider />}
              </React.Fragment>
            ))}
          </List>
          <Button
            component={Link}
            to="/analysis"
            variant="outlined"
            fullWidth
            sx={{ mt: 2 }}
          >
            {t('dashboard.viewAll')}
          </Button>
        </CardContent>
      </Card>
    </Grid>
  </Grid>
 </Container>
);
};

export default Dashboard;
```

5. (Analysis.js)

```
import React, { useState } from 'react';
import {
 Container,
```

```
  Typography,
  Paper,
  Box,
  Button,
  Grid,
  Card,
  CardContent,
  CardMedia,
  Chip,
  CircularProgress,
  Alert,
  Divider,
  List,
  ListItem,
  ListItemText,
  ListItemIcon
} from '@mui/material';
import {
  CloudUpload as CloudUploadIcon,
  BugReport as BugReportIcon,
  CheckCircle as CheckCircleIcon,
  Warning as WarningIcon
} from '@mui/icons-material';
import { useTranslation } from 'react-i18next';
import { useDropzone } from 'react-dropzone';

const Analysis = () => {
  const { t } = useTranslation();
  const [files, setFiles] = useState([]);
  const [isAnalyzing, setIsAnalyzing] = useState(false);
  const [analysisComplete, setAnalysisComplete] = useState(false);
  const [analysisResults, setAnalysisResults] = useState(null);
  const [error, setError] = useState('');

  const onDrop = acceptedFiles => {
    setFiles(acceptedFiles.map(file => Object.assign(file, {
      preview: URL.createObjectURL(file)
    })));
  };

  const { getRootProps, getInputProps, isDragActive } = useDropzone({
    onDrop,
    accept: {
      'image/*': ['.jpeg', '.jpg', '.png'],
      'video/*': ['.mp4', '.mov']
    },
    maxFiles: 5
  });

  const handleAnalyze = () => {
    if (files.length === 0) {
      setError(t('analysis.noFilesError'));
```

```
    return;
  }

  setError('');
  setIsAnalyzing(true);

  // Simulate API call for analysis
  setTimeout(() => {
    // Mock analysis results
    const mockResults = {
      totalDefects: 7,
      defectTypes: {
        cracks: 4,
        corrosion: 2,
        exposedBars: 1
      },
      defects: [
        {
          id: 1,
          type: 'crack',
          severity: 'high',
          location: { x: 120, y: 250, width: 80, height: 15 },
          confidence: 0.92
        },
        {
          id: 2,
          type: 'crack',
          severity: 'medium',
          location: { x: 320, y: 150, width: 60, height: 10 },
          confidence: 0.87
        },
        {
          id: 3,
          type: 'crack',
          severity: 'low',
          location: { x: 420, y: 350, width: 40, height: 5 },
          confidence: 0.78
        },
        {
          id: 4,
          type: 'crack',
          severity: 'medium',
          location: { x: 220, y: 450, width: 70, height: 12 },
          confidence: 0.85
        },
        {
          id: 5,
          type: 'corrosion',
          severity: 'medium',
          location: { x: 520, y: 250, width: 100, height: 100 },
          confidence: 0.81
        },
```

```
        {
          id: 6,
          type: 'corrosion',
          severity: 'high',
          location: { x: 620, y: 350, width: 120, height: 80 },
          confidence: 0.94
        },
        {
          id: 7,
          type: 'exposedBars',
          severity: 'high',
          location: { x: 320, y: 550, width: 150, height: 30 },
          confidence: 0.89
        }
      ]
    };

    setAnalysisResults(mockResults);
    setIsAnalyzing(false);
    setAnalysisComplete(true);
  }, 3000);
};

const clearAnalysis = () => {
  setFiles([]);
  setAnalysisComplete(false);
  setAnalysisResults(null);
  setError('');

  // Revoke object URLs to avoid memory leaks
  files.forEach(file => URL.revokeObjectURL(file.preview));
};

const getSeverityColor = (severity) => {
  switch (severity) {
    case 'high':
      return '#f44336'; // red
    case 'medium':
      return '#ff9800'; // orange
    case 'low':
      return '#4caf50'; // green
    default:
      return '#2196f3'; // blue
  }
};

return (
  <Container maxWidth="lg" sx={{ mt: 4, mb: 4 }}>
    <Typography variant="h4" component="h1" gutterBottom>
      {t('analysis.title')}
    </Typography>
```

```
{error && (
  <Alert severity="error" sx={{ mb: 3 }}>
    {error}
  </Alert>
)}

{!analysisComplete ? (
  <Paper sx={{ p: 3, mb: 4 }}>
    <Box
      {...getRootProps()}
      sx={{
        border: '2px dashed #cccccc',
        borderRadius: 2,
        p: 3,
        textAlign: 'center',
        bgcolor: isDragActive ? '#f0f8ff' : 'background.paper',
        cursor: 'pointer'
      }}
    >
      <input {...getInputProps()} />
      <CloudUploadIcon sx={{ fontSize: 48, color: 'primary.main', mb: 2 }} />
      <Typography variant="h6" gutterBottom>
        {isDragActive
          ? t('analysis.dropzone.active')
          : t('analysis.dropzone.inactive')}
      </Typography>
      <Typography variant="body2" color="textSecondary">
        {t('analysis.dropzone.hint')}
      </Typography>
    </Box>

    {files.length > 0 && (
      <Box sx={{ mt: 3 }}>
        <Typography variant="h6" gutterBottom>
          {t('analysis.selectedFiles')}
        </Typography>
        <Grid container spacing={2}>
          {files.map((file, index) => (
            <Grid item xs={6} sm={4} md={3} key={index}>
              <Card>
                <CardMedia
                  component="img"
                  height="140"
                  image={file.preview}
                  alt={file.name}
                />
                <CardContent sx={{ py: 1 }}>
                  <Typography variant="body2" noWrap>
                    {file.name}
                  </Typography>
                  <Typography variant="caption" color="textSecondary">
                    {(file.size / 1024 / 1024).toFixed(2)} MB
```

```
                </Typography>
              </CardContent>
            </Card>
          </Grid>
        ))}
      </Grid>

      <Box sx={{ mt: 3, display: 'flex', justifyContent: 'center' }}>
        <Button
          variant="contained"
          color="primary"
          size="large"
          onClick={handleAnalyze}
          disabled={isAnalyzing}
          startIcon={isAnalyzing ? <CircularProgress size={24} color="inherit" /> :
<BugReportIcon />}
          sx={{ mr: 2 }}
        >
          {isAnalyzing ? t('analysis.analyzing') : t('analysis.analyze')}
        </Button>
        <Button
          variant="outlined"
          color="secondary"
          size="large"
          onClick={clearAnalysis}
          disabled={isAnalyzing}
        >
          {t('analysis.clear')}
        </Button>
      </Box>
    </Box>
  )}
 </Paper>
) : (
 <Box>
  <Alert severity="success" sx={{ mb: 3 }}>
   {t('analysis.analysisComplete')}
  </Alert>

  <Paper sx={{ p: 3, mb: 4 }}>
   <Typography variant="h5" gutterBottom>
    {t('analysis.results.summary')}
   </Typography>

   <Grid container spacing={3} sx={{ mb: 3 }}>
    <Grid item xs={12} sm={4}>
     <Paper
      sx={{
       p: 2,
       textAlign: 'center',
       bgcolor: '#e3f2fd'
      }}
```

```
            >
              <Typography variant="h6" gutterBottom>
                {t('analysis.results.totalDefects')}
              </Typography>
              <Typography variant="h3">
                {analysisResults.totalDefects}
              </Typography>
            </Paper>
          </Grid>
          <Grid item xs={12} sm={8}>
            <Paper sx={{ p: 2, height: '100%' }}>
              <Typography variant="h6" gutterBottom>
                {t('analysis.results.defectTypes')}
              </Typography>
              <Box sx={{ display: 'flex', flexWrap: 'wrap', gap: 1 }}>
                <Chip
                  label={`${t('defects.cracks')}: ${analysisResults.defectTypes.cracks}`}
                  color="primary"
                />
                <Chip
                  label={`${t('defects.corrosion')}: $
{analysisResults.defectTypes.corrosion}`}
                  color="secondary"
                />
                <Chip
                  label={`${t('defects.exposedBars')}: $
{analysisResults.defectTypes.exposedBars}`}
                  color="warning"
                />
              </Box>
            </Paper>
          </Grid>
        </Grid>

        <Divider sx={{ mb: 3 }} />

        <Typography variant="h5" gutterBottom>
          {t('analysis.results.detailedResults')}
        </Typography>

        <Grid container spacing={3}>
          <Grid item xs={12} md={6}>
            <Card>
              <CardMedia
                component="img"
                height="400"
                image={files[0].preview}
                alt="Analyzed image"
              />
            </Card>
          </Grid>
          <Grid item xs={12} md={6}>
```

```
            <Paper sx={{ p: 2, height: '100%', maxHeight: 400, overflow: 'auto' }}>
              <Typography variant="h6" gutterBottom>
                {t('analysis.results.defectList')}
              </Typography>
              <List>
                {analysisResults.defects.map((defect) => (
                  <ListItem key={defect.id} divider>
                    <ListItemIcon>
                      {defect.severity === 'high' ? (
                        <WarningIcon sx={{ color: getSeverityColor(defect.severity) }} />
                      ) : (
                        <CheckCircleIcon sx={{ color: getSeverityColor(defect.severity) }} />
                      )}
                    </ListItemIcon>
                    <ListItemText
                      primary={`${t(`defects.${defect.type}`)} - ${t(`severity.${defect.severity}`)}`}
                      secondary={`${t('analysis.results.confidence')}: ${(defect.confidence * 100).toFixed(0)}%`}
                    />
                    <Chip
                      label={t(`severity.${defect.severity}`)}
                      sx={{
                        bgcolor: getSeverityColor(defect.severity),
                        color: 'white'
                      }}
                      size="small"
                    />
                  </ListItem>
                ))}
              </List>
            </Paper>
          </Grid>
        </Grid>

        <Box sx={{ mt: 3, display: 'flex', justifyContent: 'center' }}>
          <Button
            variant="contained"
            color="primary"
            size="large"
            sx={{ mr: 2 }}
          >
            {t('analysis.results.generateReport')}
          </Button>
          <Button
            variant="outlined"
            color="secondary"
            size="large"
            onClick={clearAnalysis}
          >
            {t('analysis.newAnalysis')}
          </Button>
```

```
          </Box>
        </Paper>
      </Box>
    ))}
  </Container>
 );
};

export default Analysis;
```

1. (translation.json ar)

```
{
 "app": {
  "title": "                    ",
  "tagline": "                                    "
 },
 "nav": {
  "home": "          ",
  "login": "            ",
  "register": "           ",
  "dashboard": "              ",
  "analysis": "            ",
  "reports": "          ",
  "projects": "          ",
  "profile": "            ",
  "logout": "            ",
  "language": "      "
 },
 "home": {
  "hero": {
   "title": "                                    ",
   "subtitle": "
                  ",
   "cta": "          "
  },
  "featuresSection": {
   "title": "              "
  },
  "features": {
   "detection": {
    "title": "                    ",
    "description": "
                                                  "
   },
   "analysis": {
```

```
      "title": "                    ",
      "description": "
                      "
    },
    "reports": {
      "title": "                  ",
      "description": "
                  "
    }
  },
  "howItWorks": {
    "title": "                  ",
    "step1": {
      "title": "                      ",
      "description": "                                                    "
    },
    "step2": {
      "title": "                       ",
      "description": "                                                      "
    },
    "step3": {
      "title": "                      ",
      "description": "                                                        "
    }
  },
  "cta": {
    "title": "                          ",
    "description": "
                          ",
    "button": "                "
  }
},
"login": {
  "title": "                ",
  "email": "                    ",
  "password": "              ",
  "submit": "                ",
  "forgotPassword": "                    ",
  "noAccount": "                  ",
  "register": "                  ",
  "error": "                                                  .
                  ."
},
"register": {
  "title": "                    ",
  "fullName": "                ",
  "email": "                    ",
  "password": "              ",
  "confirmPassword": "                    ",
  "userType": "                ",
  "userTypes": {
    "engineer": "                ",
```

```
    "inspector": "              ",
    "company": "                "
  },
  "submit": "              ",
  "haveAccount": "                    ",
  "login": "                ",
  "error": "                          .                          ."
 },
 "validation": {
  "required": "                    ",
  "email": "                          ",
  "passwordLength": "                  8                          ",
  "passwordMatch": "                          "
 },
 "dashboard": {
  "title": "                ",
  "summary": {
   "projects": "            ",
   "analyses": "              ",
   "defects": "                    ",
   "reports": "            "
  },
  "charts": {
   "monthlyAnalysis": "                      ",
   "defectDistribution": "                        ",
   "defectTypes": {
    "cracks": "          ",
    "corrosion": "       ",
    "exposedBars": "                      ",
    "dampness": "        "
   }
  },
  "recentProjects": {
   "title": "                      ",
   "project1": "          -            ",
   "project2": "      -              ",
   "project3": "          -              "
  },
  "recentAnalyses": {
   "title": "                        ",
   "analysis1": "              -                ",
   "analysis2": "                -              ",
   "analysis3": "                -                    "
  },
  "date": "            ",
  "status": "          ",
  "defectsFound": "                    ",
  "viewAll": "            "
 },
 "analysis": {
  "title": "                          ",
  "dropzone": {
```

```
    "active": "                    ",
    "inactive": "                                                    ,"
    "hint": "                        JPG    PNG    MP4 (       5 :            )"
  },
  "selectedFiles": "                    ",
  "analyze": "                  ",
  "analyzing": "                  ...",
  "clear": "     ",
  "noFilesError": "                                                ",
  "analysisComplete": "                          !",
  "results": {
    "summary": "                ",
    "totalDefects": "                  ",
    "defectTypes": "                ",
    "detailedResults": "                      ",
    "defectList": "                        ",
    "confidence": "              ",
    "generateReport": "                "
  },
  "newAnalysis": "              "
},
"defects": {
  "crack": "       ",
  "cracks": "          ",
  "corrosion": "       ",
  "exposedBars": "                      ",
  "dampness": "         ",
  "mould": "     "
},
"severity": {
  "high": "         ",
  "medium": "          ",
  "low": "          "
},
"status": {
  "active": "     ",
  "completed": "         ",
  "pending": "                  "
},
"months": {
  "jan": "         ",
  "feb": "          ",
  "mar": "        ",
  "apr": "         ",
  "may": "        ",
  "jun": "         ",
  "jul": "        ",
  "aug": "         ",
  "sep": "          ",
  "oct": "          ",
  "nov": "          ",
  "dec": "          "
```

```
  }
}
```

2. (translation.json en)

```
{
 "app": {
  "title": "Structural Defect Detection System",
  "tagline": "AI-Powered Structural Defect Detection"
 },
 "nav": {
  "home": "Home",
  "login": "Login",
  "register": "Register",
  "dashboard": "Dashboard",
  "analysis": "Analysis",
  "reports": "Reports",
  "projects": "Projects",
  "profile": "Profile",
  "logout": "Logout",
  "language": "Language"
 },
 "home": {
  "hero": {
   "title": "AI-Powered Structural Defect Detection",
   "subtitle": "Advanced system for detecting and analyzing structural defects in buildings and bridges using computer vision",
   "cta": "Get Started"
  },
  "featuresSection": {
   "title": "Features"
  },
  "features": {
   "detection": {
    "title": "High-Precision Defect Detection",
    "description": "Detect cracks, corrosion, exposed reinforcement bars, and other structural defects using advanced AI algorithms"
   },
   "analysis": {
    "title": "Comprehensive Defect Analysis",
    "description": "Detailed analysis of detected defects with location, severity, area, and classification by risk level"
   },
   "reports": {
    "title": "Professional Reports",
    "description": "Generate detailed and comprehensive reports on the condition of structures with recommendations for repair and maintenance"
   }
  },
  "howItWorks": {
```

```
      "title": "How It Works",
      "step1": {
        "title": "Upload Images & Videos",
        "description": "Upload images or videos of the structures you want to inspect"
      },
      "step2": {
        "title": "AI Analysis",
        "description": "The system analyzes the images and videos to detect structural defects"
      },
      "step3": {
        "title": "Get Reports",
        "description": "View analysis results and get detailed reports on detected defects"
      }
    },
    "cta": {
      "title": "Start Using the System Today",
      "description": "Join hundreds of engineers and inspectors who use our system to improve the efficiency of inspection and maintenance operations",
      "button": "Create Free Account"
    }
  },
  "login": {
    "title": "Login",
    "email": "Email",
    "password": "Password",
    "submit": "Login",
    "forgotPassword": "Forgot Password?",
    "noAccount": "Don't have an account?",
    "register": "Create a new account",
    "error": "An error occurred during login. Please check your credentials and try again."
  },
  "register": {
    "title": "Create New Account",
    "fullName": "Full Name",
    "email": "Email",
    "password": "Password",
    "confirmPassword": "Confirm Password",
    "userType": "User Type",
    "userTypes": {
      "engineer": "Structural Engineer",
      "inspector": "Building Inspector",
      "company": "Construction Company"
    },
    "submit": "Create Account",
    "haveAccount": "Already have an account?",
    "login": "Login",
    "error": "An error occurred during account creation. Please try again."
  },
  "validation": {
```

```
    "required": "This field is required",
    "email": "Please enter a valid email",
    "passwordLength": "Password must be at least 8 characters",
    "passwordMatch": "Passwords do not match"
  },
  "dashboard": {
    "title": "Dashboard",
    "summary": {
      "projects": "Projects",
      "analyses": "Analyses",
      "defects": "Detected Defects",
      "reports": "Reports"
    },
    "charts": {
      "monthlyAnalysis": "Monthly Analyses",
      "defectDistribution": "Defect Type Distribution",
      "defectTypes": {
        "cracks": "Cracks",
        "corrosion": "Corrosion",
        "exposedBars": "Exposed Bars",
        "dampness": "Dampness"
      }
    },
    "recentProjects": {
      "title": "Recent Projects",
      "project1": "Residential Building - Riyadh",
      "project2": "Pedestrian Bridge - Jeddah",
      "project3": "Commercial Building - Dammam"
    },
    "recentAnalyses": {
      "title": "Recent Analyses",
      "analysis1": "Ground Floor - Residential Building",
      "analysis2": "Support Columns - Pedestrian Bridge",
      "analysis3": "Exterior Facade - Commercial Building"
    },
    "date": "Date",
    "status": "Status",
    "defectsFound": "Defects Found",
    "viewAll": "View All"
  },
  "analysis": {
    "title": "Image & Video Analysis",
    "dropzone": {
      "active": "Drop the files here",
      "inactive": "Drag & drop images or videos here, or click to select",
      "hint": "You can upload JPG, PNG, or MP4 files (Max: 5 files)"
    },
    "selectedFiles": "Selected Files",
    "analyze": "Analyze Files",
    "analyzing": "Analyzing...",
    "clear": "Clear",
    "noFilesError": "Please select at least one file for analysis",
```

```
    "analysisComplete": "Analysis completed successfully!",
    "results": {
      "summary": "Results Summary",
      "totalDefects": "Total Defects",
      "defectTypes": "Defect Types",
      "detailedResults": "Detailed Results",
      "defectList": "Detected Defects List",
      "confidence": "Confidence",
      "generateReport": "Generate Report"
    },
    "newAnalysis": "New Analysis"
  },
  "defects": {
    "crack": "Crack",
    "cracks": "Cracks",
    "corrosion": "Corrosion",
    "exposedBars": "Exposed Reinforcement Bars",
    "dampness": "Dampness",
    "mould": "Mould"
  },
  "severity": {
    "high": "High",
    "medium": "Medium",
    "low": "Low"
  },
  "status": {
    "active": "Active",
    "completed": "Completed",
    "pending": "Pending"
  },
  "months": {
    "jan": "January",
    "feb": "February",
    "mar": "March",
    "apr": "April",
    "may": "May",
    "jun": "June",
    "jul": "July",
    "aug": "August",
    "sep": "September",
    "oct": "October",
    "nov": "November",
    "dec": "December"
  }
}
```

1. App.js

```
 import React from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-
dom';
import { ThemeProvider, createTheme, CssBaseline } from '@mui/material';
import { I18nextProvider } from 'react-i18next';
import i18n from './utils/i18n';
import rtlPlugin from 'stylis-plugin-rtl';
import { CacheProvider } from '@emotion/react';
import createCache from '@emotion/cache';
import { prefixer } from 'stylis';

// Pages
import Home from './pages/Home';
import Login from './pages/Login';
import Register from './pages/Register';
import Dashboard from './pages/Dashboard';
import Analysis from './pages/Analysis';
import Reports from './pages/Reports';
import NotFound from './pages/NotFound';

// Components
import Layout from './components/common/Layout';

// Create rtl cache
const cacheRtl = createCache({
  key: 'muirtl',
  stylisPlugins: [prefixer, rtlPlugin],
});

// Create ltr cache
const cacheLtr = createCache({
  key: 'muiltr',
  stylisPlugins: [prefixer],
});

const App = () => {
  const direction = i18n.language === 'ar' ? 'rtl' : 'ltr';

  const theme = createTheme({
    direction,
    palette: {
      primary: {
        main: '#1976d2',
      },
      secondary: {
        main: '#f50057',
```

```
    },
  },
  typography: {
    fontFamily: direction === 'rtl'
      ? '"Tajawal", "Roboto", "Helvetica", "Arial", sans-serif'
      : '"Roboto", "Helvetica", "Arial", sans-serif',
  },
});

const isAuthenticated = () => {
  return localStorage.getItem('isAuthenticated') === 'true';
};

const ProtectedRoute = ({ children }) => {
  if (!isAuthenticated()) {
    return <Navigate to="/login" />;
  }
  return children;
};

return (
  <CacheProvider value={direction === 'rtl' ? cacheRtl : cacheLtr}>
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <I18nextProvider i18n={i18n}>
        <Router>
          <Layout>
            <Routes>
              <Route path="/" element={<Home />} />
              <Route path="/login" element={<Login />} />
              <Route path="/register" element={<Register />} />
              <Route
                path="/dashboard"
                element={
                  <ProtectedRoute>
                    <Dashboard />
                  </ProtectedRoute>
                }
              />
              <Route
                path="/analysis"
                element={
                  <ProtectedRoute>
                    <Analysis />
                  </ProtectedRoute>
                }
              />
              <Route
                path="/reports"
                element={
                  <ProtectedRoute>
                    <Reports />
```

```
              </ProtectedRoute>
            }
           />
           <Route path="*" element={<NotFound />} />
          </Routes>
         </Layout>
        </Router>
       </I18nextProvider>
     </ThemeProvider>
   </CacheProvider>
  );
};

export default App;
```

2. i18n.js

```
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import LanguageDetector from 'i18next-browser-languagedetector';
import Backend from 'i18next-http-backend';

i18n
  .use(Backend)
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    fallbackLng: 'ar',
    debug: process.env.NODE_ENV === 'development',
    interpolation: {
      escapeValue: false,
    },
    backend: {
      loadPath: '/locales/{{lng}}/translation.json',
    },
    detection: {
      order: ['localStorage', 'navigator'],
      caches: ['localStorage'],
    },
  });

export default i18n;
```

3. Layout.js

```
import React, { useState } from 'react';
import {
  AppBar,
```

```
  Toolbar,
  Typography,
  Button,
  IconButton,
  Drawer,
  List,
  ListItem,
  ListItemIcon,
  ListItemText,
  Box,
  Container,
  Menu,
  MenuItem,
  Divider,
  useMediaQuery,
  useTheme
} from '@mui/material';
import {
  Menu as MenuIcon,
  Home as HomeIcon,
  Dashboard as DashboardIcon,
  Image as ImageIcon,
  Description as DescriptionIcon,
  Folder as FolderIcon,
  Person as PersonIcon,
  Logout as LogoutIcon,
  Translate as TranslateIcon
} from '@mui/icons-material';
import { Link as RouterLink, useNavigate, useLocation } from 'react-router-dom';
import { useTranslation } from 'react-i18next';

const Layout = ({ children }) => {
  const { t, i18n } = useTranslation();
  const navigate = useNavigate();
  const location = useLocation();
  const theme = useTheme();
  const isMobile = useMediaQuery(theme.breakpoints.down('md'));

  const [drawerOpen, setDrawerOpen] = useState(false);
  const [languageMenu, setLanguageMenu] = useState(null);

  const isAuthenticated = localStorage.getItem('isAuthenticated') === 'true';

  const handleDrawerToggle = () => {
    setDrawerOpen(!drawerOpen);
  };

  const handleLanguageMenuOpen = (event) => {
    setLanguageMenu(event.currentTarget);
  };

  const handleLanguageMenuClose = () => {
```

```
    setLanguageMenu(null);
  };

  const changeLanguage = (lng) => {
    i18n.changeLanguage(lng);
    handleLanguageMenuClose();
  };

  const handleLogout = () => {
    localStorage.removeItem('isAuthenticated');
    navigate('/login');
  };

  const menuItems = [
    { text: t('nav.home'), icon: <HomeIcon />, path: '/' },
    { text: t('nav.dashboard'), icon: <DashboardIcon />, path: '/dashboard', auth:
true },
    { text: t('nav.analysis'), icon: <ImageIcon />, path: '/analysis', auth: true },
    { text: t('nav.reports'), icon: <DescriptionIcon />, path: '/reports', auth: true },
    { text: t('nav.projects'), icon: <FolderIcon />, path: '/projects', auth: true },
    { text: t('nav.profile'), icon: <PersonIcon />, path: '/profile', auth: true }
  ];

  const drawer = (
    <Box sx={{ width: 250 }} role="presentation" onClick={handleDrawerToggle}>
      <List>
        {menuItems.map((item) => (
          (!item.auth || (item.auth && isAuthenticated)) && (
            <ListItem
              button
              key={item.text}
              component={RouterLink}
              to={item.path}
              selected={location.pathname === item.path}
            >
              <ListItemIcon>{item.icon}</ListItemIcon>
              <ListItemText primary={item.text} />
            </ListItem>
          )
        ))}
      </List>
      {isAuthenticated && (
        <>
          <Divider />
          <List>
            <ListItem button onClick={handleLogout}>
              <ListItemIcon><LogoutIcon /></ListItemIcon>
              <ListItemText primary={t('nav.logout')} />
            </ListItem>
          </List>
        </>
      )}
```

```
  </Box>
);

return (
  <Box sx={{ display: 'flex', flexDirection: 'column', minHeight: '100vh' }}>
    <AppBar position="static">
      <Toolbar>
        {isAuthenticated && (
          <IconButton
            color="inherit"
            aria-label="open drawer"
            edge="start"
            onClick={handleDrawerToggle}
            sx={{ mr: 2 }}
          >
            <MenuIcon />
          </IconButton>
        )}

        <Typography
          variant="h6"
          component={RouterLink}
          to="/"
          sx={{
            flexGrow: 1,
            textDecoration: 'none',
            color: 'inherit',
            display: 'flex',
            alignItems: 'center'
          }}
        >
          {t('app.title')}
        </Typography>

        {!isMobile && (
          <Box sx={{ display: 'flex' }}>
            <Button
              color="inherit"
              component={RouterLink}
              to="/"
              sx={{ mx: 1 }}
            >
              {t('nav.home')}
            </Button>

            {isAuthenticated ? (
              <>
                <Button
                  color="inherit"
                  component={RouterLink}
                  to="/dashboard"
                  sx={{ mx: 1 }}
```

```
          >
            {t('nav.dashboard')}
          </Button>
          <Button
            color="inherit"
            component={RouterLink}
            to="/analysis"
            sx={{ mx: 1 }}
          >
            {t('nav.analysis')}
          </Button>
          <Button
            color="inherit"
            component={RouterLink}
            to="/reports"
            sx={{ mx: 1 }}
          >
            {t('nav.reports')}
          </Button>
        </>
      ) : (
        <>
          <Button
            color="inherit"
            component={RouterLink}
            to="/login"
            sx={{ mx: 1 }}
          >
            {t('nav.login')}
          </Button>
          <Button
            color="inherit"
            component={RouterLink}
            to="/register"
            sx={{ mx: 1 }}
          >
            {t('nav.register')}
          </Button>
        </>
      )}
    </Box>
  )}

  <IconButton
    color="inherit"
    onClick={handleLanguageMenuOpen}
    aria-controls="language-menu"
    aria-haspopup="true"
  >
    <TranslateIcon />
  </IconButton>
```

```
        <Menu
          id="language-menu"
          anchorEl={languageMenu}
          keepMounted
          open={Boolean(languageMenu)}
          onClose={handleLanguageMenuClose}
        >
          <MenuItem onClick={() => changeLanguage('ar')}>          </MenuItem>
          <MenuItem onClick={() => changeLanguage('en')}>English</MenuItem>
        </Menu>

        {isAuthenticated && !isMobile && (
          <Button
            color="inherit"
            onClick={handleLogout}
            startIcon={<LogoutIcon />}
            sx={{ ml: 1 }}
          >
            {t('nav.logout')}
          </Button>
        )}
      </Toolbar>
    </AppBar>

    <Drawer
      anchor={i18n.language === 'ar' ? 'right' : 'left'}
      open={drawerOpen}
      onClose={handleDrawerToggle}
    >
      {drawer}
    </Drawer>

    <Box component="main" sx={{ flexGrow: 1 }}>
      {children}
    </Box>

    <Box
      component="footer"
      sx={{
        py: 3,
        px: 2,
        mt: 'auto',
        backgroundColor: (theme) => theme.palette.grey[200]
      }}
    >
      <Container maxWidth="lg">
        <Typography variant="body2" color="text.secondary" align="center">
          © {new Date().getFullYear()} {t('app.title')}
        </Typography>
      </Container>
    </Box>
  </Box>
```

```
  );
};

export default Layout;
```

(Reports.js)

```
 import React, { useState } from 'react';
import {
  Container,
  Typography,
  Paper,
  Box,
  Button,
  Grid,
  Card,
  CardContent,
  CardActions,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Chip,
  TextField,
  InputAdornment,
  IconButton,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  Tabs,
  Tab
} from '@mui/material';
import {
  Search as SearchIcon,
  GetApp as DownloadIcon,
  Share as ShareIcon,
  Delete as DeleteIcon,
  Visibility as ViewIcon,
  Add as AddIcon,
  FilterList as FilterIcon
} from '@mui/icons-material';
import { useTranslation } from 'react-i18next';

const Reports = () => {
  const { t } = useTranslation();
  const [searchTerm, setSearchTerm] = useState('');
```

```
const [tabValue, setTabValue] = useState(0);
const [openDialog, setOpenDialog] = useState(false);
const [selectedReport, setSelectedReport] = useState(null);

// Sample data for reports
const reports = [
 {
  id: 1,
  title: '          -                    ',
  date: '2025-04-15',
  project: '          -            ',
  defects: 12,
  status: 'completed'
 },
 {
  id: 2,
  title: '     -                    ',
  date: '2025-04-10',
  project: '     -            ',
  defects: 8,
  status: 'completed'
 },
 {
  id: 3,
  title: '          -                    ',
  date: '2025-04-05',
  project: '          -            ',
  defects: 15,
  status: 'completed'
 },
 {
  id: 4,
  title: '            -                ',
  date: '2025-03-25',
  project: '          -            ',
  defects: 5,
  status: 'completed'
 },
 {
  id: 5,
  title: '            -                    ',
  date: '2025-03-15',
  project: '     -            ',
  defects: 3,
  status: 'completed'
 }
];

const handleTabChange = (event, newValue) => {
 setTabValue(newValue);
};
```

```
  const handleSearchChange = (event) => {
    setSearchTerm(event.target.value);
  };

  const handleViewReport = (report) => {
    setSelectedReport(report);
    setOpenDialog(true);
  };

  const handleCloseDialog = () => {
    setOpenDialog(false);
  };

  const filteredReports = reports.filter(report =>
    report.title.toLowerCase().includes(searchTerm.toLowerCase()) ||
    report.project.toLowerCase().includes(searchTerm.toLowerCase())
  );

  return (
    <Container maxWidth="lg" sx={{ mt: 4, mb: 4 }}>
      <Typography variant="h4" component="h1" gutterBottom>
        {t('reports.title')}
      </Typography>

      <Paper sx={{ p: 2, mb: 3 }}>
        <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center',
mb: 2 }}>
          <TextField
            placeholder={t('reports.search')}
            variant="outlined"
            size="small"
            value={searchTerm}
            onChange={handleSearchChange}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <SearchIcon />
                </InputAdornment>
              ),
            }}
            sx={{ width: 300 }}
          />
          <Box>
            <Button
              variant="outlined"
              startIcon={<FilterIcon />}
              sx={{ mr: 1 }}
            >
              {t('reports.filter')}
            </Button>
            <Button
              variant="contained"
```

```
          startIcon={<AddIcon />}
        >
          {t('reports.createNew')}
        </Button>
      </Box>
    </Box>

    <Tabs value={tabValue} onChange={handleTabChange} sx={{ mb: 2 }}>
      <Tab label={t('reports.tabs.all')} />
      <Tab label={t('reports.tabs.recent')} />
      <Tab label={t('reports.tabs.shared')} />
    </Tabs>

    <TableContainer>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>{t('reports.table.title')}</TableCell>
            <TableCell>{t('reports.table.project')}</TableCell>
            <TableCell>{t('reports.table.date')}</TableCell>
            <TableCell>{t('reports.table.defects')}</TableCell>
            <TableCell>{t('reports.table.status')}</TableCell>
            <TableCell>{t('reports.table.actions')}</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {filteredReports.map((report) => (
            <TableRow key={report.id}>
              <TableCell>{report.title}</TableCell>
              <TableCell>{report.project}</TableCell>
              <TableCell>{report.date}</TableCell>
              <TableCell>{report.defects}</TableCell>
              <TableCell>
                <Chip
                  label={t(`status.${report.status}`)}
                  color={report.status === 'completed' ? 'success' : 'warning'}
                  size="small"
                />
              </TableCell>
              <TableCell>
                <IconButton size="small" onClick={() => handleViewReport(report)}>
                  <ViewIcon />
                </IconButton>
                <IconButton size="small">
                  <DownloadIcon />
                </IconButton>
                <IconButton size="small">
                  <ShareIcon />
                </IconButton>
                <IconButton size="small" color="error">
                  <DeleteIcon />
                </IconButton>
```

```
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  </Paper>

  <Typography variant="h5" component="h2" gutterBottom>
    {t('reports.recentReports')}
  </Typography>

  <Grid container spacing={3}>
    {reports.slice(0, 3).map((report) => (
      <Grid item xs={12} md={4} key={report.id}>
        <Card>
          <CardContent>
            <Typography variant="h6" component="h3" gutterBottom>
              {report.title}
            </Typography>
            <Typography variant="body2" color="textSecondary" gutterBottom>
              {report.project}
            </Typography>
            <Typography variant="body2">
              {t('reports.date')}: {report.date}
            </Typography>
            <Typography variant="body2">
              {t('reports.defectsFound')}: {report.defects}
            </Typography>
          </CardContent>
          <CardActions>
            <Button size="small" startIcon={<ViewIcon />} onClick={() =>
handleViewReport(report)}>
              {t('reports.view')}
            </Button>
            <Button size="small" startIcon={<DownloadIcon />}>
              {t('reports.download')}
            </Button>
            <Button size="small" startIcon={<ShareIcon />}>
              {t('reports.share')}
            </Button>
          </CardActions>
        </Card>
      </Grid>
    ))}
  </Grid>

  <Dialog
    open={openDialog}
    onClose={handleCloseDialog}
    maxWidth="md"
    fullWidth
```

```
      >
        {selectedReport && (
          <>
            <DialogTitle>{selectedReport.title}</DialogTitle>
            <DialogContent dividers>
              <Typography variant="subtitle1" gutterBottom>
                {t('reports.project')}: {selectedReport.project}
              </Typography>
              <Typography variant="subtitle2" gutterBottom>
                {t('reports.date')}: {selectedReport.date}
              </Typography>
              <Typography variant="subtitle2" gutterBottom>
                {t('reports.defectsFound')}: {selectedReport.defects}
              </Typography>

              <Box sx={{ mt: 3 }}>
                <Typography variant="h6" gutterBottom>
                  {t('reports.summary')}
                </Typography>
                <Typography variant="body1" paragraph>
                  {t('reports.sampleContent.summary')}
                </Typography>
              </Box>

              <Box sx={{ mt: 3 }}>
                <Typography variant="h6" gutterBottom>
                  {t('reports.defectDetails')}
                </Typography>
                <Typography variant="body1" paragraph>
                  {t('reports.sampleContent.defectDetails')}
                </Typography>

                <TableContainer component={Paper} sx={{ mt: 2 }}>
                  <Table size="small">
                    <TableHead>
                      <TableRow>
                        <TableCell>{t('reports.defectTable.type')}</TableCell>
                        <TableCell>{t('reports.defectTable.location')}</TableCell>
                        <TableCell>{t('reports.defectTable.severity')}</TableCell>
                        <TableCell>{t('reports.defectTable.recommendation')}</TableCell>
                      </TableRow>
                    </TableHead>
                    <TableBody>
                      <TableRow>
                        <TableCell>{t('defects.crack')}</TableCell>
                        <TableCell>{t('reports.sampleContent.locations.wall')}</TableCell>
                        <TableCell>
                          <Chip
                            label={t('severity.high')}
                            color="error"
                            size="small"
                          />
```

```
                  </TableCell>
                  <TableCell>{t('reports.sampleContent.recommendations.immediate')}
</TableCell>
                </TableRow>
                <TableRow>
                  <TableCell>{t('defects.crack')}</TableCell>
                  <TableCell>{t('reports.sampleContent.locations.ceiling')}</TableCell>
                  <TableCell>
                    <Chip
                      label={t('severity.medium')}
                      color="warning"
                      size="small"
                    />
                  </TableCell>
                  <TableCell>{t('reports.sampleContent.recommendations.repair')}</
TableCell>
                </TableRow>
                <TableRow>
                  <TableCell>{t('defects.corrosion')}</TableCell>
                  <TableCell>{t('reports.sampleContent.locations.column')}</
TableCell>
                  <TableCell>
                    <Chip
                      label={t('severity.high')}
                      color="error"
                      size="small"
                    />
                  </TableCell>
                  <TableCell>{t('reports.sampleContent.recommendations.immediate')}
</TableCell>
                </TableRow>
              </TableBody>
            </Table>
          </TableContainer>
        </Box>

        <Box sx={{ mt: 3 }}>
          <Typography variant="h6" gutterBottom>
            {t('reports.conclusion')}
          </Typography>
          <Typography variant="body1" paragraph>
            {t('reports.sampleContent.conclusion')}
          </Typography>
        </Box>
      </DialogContent>
      <DialogActions>
        <Button onClick={handleCloseDialog}>
          {t('reports.close')}
        </Button>
        <Button variant="contained" startIcon={<DownloadIcon />}>
          {t('reports.download')}
        </Button>
```

```
          </DialogActions>
        </>
      )}
    </Dialog>
  </Container>
 );
};

export default Reports;
```

:

(ar/translation.json)

```
"reports": {
 "title": "          ",
 "search": "                    ",
 "filter": "       ",
 "createNew": "                    ",
 "tabs": {
  "all": "                ",
  "recent": "                    ",
  "shared": "                    "
 },
 "table": {
  "title": "                ",
  "project": "          ",
  "date": "          ",
  "defects": "          ",
  "status": "          ",
  "actions": "              "
 },
 "recentReports": "                ",
 "date": "          ",
 "defectsFound": "                    ",
 "view": "      ",
 "download": "        ",
 "share": "          ",
 "project": "            ",
 "summary": "                ",
 "defectDetails": "                ",
 "conclusion": "                    ",
 "close": "          ",
 "defectTable": {
  "type": "            ",
  "location": "          ",
```

```
    "severity": "         ",
    "recommendation": "            "
  },
  "sampleContent": {
    "summary":
"                          .
                                                                    ,"
    "defectDetails": "                                                              :"
    "conclusion": "
       .
                                                                  ,"
    "locations": {
      "wall": "                    -                  ",
      "ceiling": "                   -                          ",
      "column": "           -                  "
    },
    "recommendations": {
      "immediate": "                      ",
      "repair": "       3              ",
      "monitor": "                  "
    }
  }
}
```

(en/translation.json)

```
 "reports": {
  "title": "Reports",
  "search": "Search reports",
  "filter": "Filter",
  "createNew": "Create New Report",
  "tabs": {
    "all": "All Reports",
    "recent": "Recent Reports",
    "shared": "Shared Reports"
  },
  "table": {
    "title": "Report Title",
    "project": "Project",
    "date": "Date",
    "defects": "Defects",
    "status": "Status",
    "actions": "Actions"
  },
  "recentReports": "Recent Reports",
  "date": "Date",
  "defectsFound": "Defects Found",
  "view": "View",
  "download": "Download",
  "share": "Share",
```

```
  "project": "Project",
  "summary": "Report Summary",
  "defectDetails": "Defect Details",
  "conclusion": "Conclusion & Recommendations",
  "close": "Close",
  "defectTable": {
    "type": "Defect Type",
    "location": "Location",
    "severity": "Severity",
    "recommendation": "Recommendation"
  },
  "sampleContent": {
    "summary": "A comprehensive inspection of the building was conducted and several structural defects requiring attention were discovered. AI technologies were used to analyze images and detect defects with high precision.",
    "defectDetails": "Below are the details of the detected defects and repair recommendations:",
    "conclusion":
"Based on the analysis, we recommend immediate repairs for high severity defects, and scheduling repairs for medium severity defects within the next three months. We also recommend a follow-up inspection after repairs to ensure their effectiveness.",
    "locations": {
      "wall": "North Wall - Ground Floor",
      "ceiling": "Main Room Ceiling - First Floor",
      "column": "Main Column - Entrance"
    },
    "recommendations": {
      "immediate": "Immediate repair required",
      "repair": "Repair within 3 months",
      "monitor": "Periodic monitoring"
    }
  }
}
```