OLLI PDF Generator - Source Code Package

Overview

This is the complete source code for the OLLI Walks & Hikes PDF Generator desktop application built with Electron and React.

Package Contents

Plain Text

```
olli-pdf-generator-v2/
├── src/
│   ├── App.js                      # Main React component
│   ├── App.css                     # Application styles
│   ├── index.js                    # React entry point
│   ├── index.css                   # Global styles
│   ├── global.js                   # Global polyfill
│   └── extracted_locations.json   # 108 OLLI locations database
├── public/
│   └── index.html                  # HTML template
├── main.js                         # Electron main process
├── preload.js                      # Electron preload script
├── package.json                    # Dependencies and scripts
├── webpack.config.js               # Build configuration
└── README.md                       # Project documentation
```

System Requirements

For Development

- **Node.js**: v22.13.0 or higher
- **npm**: 10.9.2 or higher
- **Operating System**: Windows 10/11, macOS, or Linux

For Building Windows Installer

- **wine** (on Linux) for cross-platform builds
- **electron-builder** (installed via npm)

Installation & Setup

Step 1: Extract the Archive

Bash

```bash
tar -xzf OLLI_PDF_Generator_SOURCE_CODE.tar.gz
cd olli-pdf-generator-v2
```

Step 2: Install Dependencies

Bash

```bash
npm install
```

This will install all required packages:

- React 18.2.0
- Electron 38.7.2
- jsPDF 2.5.1
- Webpack 5.104.1
- Babel and other build tools

Step 3: Run in Development Mode

Bash

```bash
# Build the React app
npm run build

# Start Electron
npm start
```

The application window will open with the full interface.

Available Scripts

`npm run build`

Builds the React application using Webpack. Output goes to `dist/` folder.

Bash

```
npm run build
```

`npm start`

Starts the Electron application (must run `npm run build` first).

Bash

```
npm start
```

`npm run dist:win`

Builds a Windows installer (.exe file). Output goes to `release/` folder.

Bash

```
npm run dist:win
```

`npm run dist`

Builds installers for the current platform.

Bash

```
npm run dist
```

Project Structure Explained

Main Files

main.js - Electron Main Process

- Creates the application window
- Configures webPreferences (nodeIntegration, contextIsolation)
- Loads the React app from dist/index.html

preload.js - Electron Preload Script

- Runs before web page loads
- Can expose Node.js APIs to renderer process

webpack.config.js - Build Configuration

- Compiles React/JSX to JavaScript
- Bundles all code into dist/bundle.js
- Processes CSS and JSON files
- Sets publicPath for Electron compatibility

package.json - Project Configuration

- Lists all dependencies
- Defines build scripts
- Configures electron-builder for installers

React Application

src/index.js - React Entry Point

- Imports global polyfill
- Mounts React app to DOM

src/App.js - Main Application Component (1000+ lines)

- State management for walks, hikes, locations
- PDF generation logic using jsPDF
- Location database management
- localStorage persistence
- Auto-fill functionality

src/App.css - Application Styles

- Senior-friendly design (large fonts, clear buttons)
- Responsive layout
- Color-coded buttons and sections

src/extracted_locations.json - Location Database

- 108 OLLI locations
- Alphabetically sorted
- Complete data: address, town, comments, meeting notes

Key Features in the Code

1. Location Management (App.js)

JavaScript

```javascript
// Load locations from localStorage or defaults
useEffect(() => {
  const savedLocations = localStorage.getItem('olliLocations');
  if (savedLocations) {
    setLocations(JSON.parse(savedLocations));
  } else {
    setLocations(defaultLocations);
  }
}, []);

// Save to localStorage on changes
useEffect(() => {
  if (locations.length > 0) {
    localStorage.setItem('olliLocations', JSON.stringify(locations));
  }
}, [locations]);
```

2. Auto-Generate Monday Walks

JavaScript

```javascript
const getMondaysInMonth = (month, year) => {
  const mondays = [];
  const date = new Date(year, month - 1, 1);
  while (date.getMonth() === month - 1) {
    if (date.getDay() === 1) {  // Monday
      mondays.push(new Date(date));
    }
    date.setDate(date.getDate() + 1);
  }
  return mondays;
};
```

3. Thursday Hikes Selection

JavaScript

```javascript
const getThursdaysInMonth = (month, year) => {
  const thursdays = [];
  const date = new Date(year, month - 1, 1);
  while (date.getMonth() === month - 1) {
    if (date.getDay() === 4) {  // Thursday
      thursdays.push(new Date(date));
    }
    date.setDate(date.getDate() + 1);
  }
  return thursdays;
};
```

4. PDF Generation with jsPDF

JavaScript

```javascript
const generatePdf = async () => {
  const doc = new jsPDF('landscape', 'pt', 'letter');

  // Add title
  doc.setFontSize(18);
  doc.setFont('helvetica', 'bold');
  doc.text(`OLLI Walks & Hikes - ${monthNames[month - 1]} ${year}`, 50, 40);

  // Add walks section
  // Add hikes section
  // Handle text wrapping
  // Save PDF

  doc.save(`OLLI_Walks_Hikes_${monthNames[month - 1]}_${year}.pdf`);
};
```

5. Text Wrapping for PDF

JavaScript

```javascript
const wrapText = (text, maxWidth) => {
  const words = text.split(' ');
  const lines = [];
  let currentLine = '';

  words.forEach(word => {
    const testLine = currentLine + (currentLine ? ' ' : '') + word;
    const width = doc.getTextWidth(testLine);

    if (width > maxWidth && currentLine) {
      lines.push(currentLine);
      currentLine = word;
    } else {
      currentLine = testLine;
    }
  });

  if (currentLine) lines.push(currentLine);
  return lines;
};
```

Configuration Details

Electron Configuration (main.js)

JavaScript

```
webPreferences: {
  preload: path.join(__dirname, 'preload.js'),
  nodeIntegration: true,      // Allows Node.js APIs in renderer
  contextIsolation: false     // Allows direct access to Node APIs
}
```

Why nodeIntegration is enabled:

- jsPDF requires Node.js `require()` and `module` APIs
- Safe for this desktop app (no external web content)

Webpack Configuration

JavaScript

```
output: {
  publicPath: './',  // Critical for Electron file:// protocol
}

target: 'electron-renderer'  // Optimizes for Electron
```

electron-builder Configuration (package.json)

JSON

```
"build": {
  "appId": "com.olli.pdfgenerator",
  "productName": "OLLI PDF Generator",
  "win": {
    "target": "nsis",
    "icon": "build/icon.ico"
  },
  "nsis": {
    "oneClick": false,
    "allowToChangeInstallationDirectory": true
  }
}
```

Building for Windows

On Windows

Bash

```bash
npm install
npm run build
npm run dist:win
```

Output: `release/OLLI PDF Generator Setup 2.0.0.exe`

On Linux (Cross-compile)

Bash

```bash
# Install wine
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install wine wine32 wine64

# Build
npm install
npm run build
npm run dist:win
```

Troubleshooting

Issue: Blank Screen

Cause: Incorrect webpack publicPath

Fix: Ensure `publicPath: './'` in webpack.config.js

Issue: "global is not defined"

Cause: jsPDF expects Node.js global

Fix: Add `<script>window.global = window;</script>` in index.html

Issue: "require is not defined"

Cause: Node.js APIs not available in renderer

Fix: Set `nodeIntegration: true` in main.js

Issue: Only 5 Locations Load

Cause: Old localStorage cache

Fix: Clear `%APPDATA%\olli-pdf-generator` folder

Modifying the Location Database

Option 1: Edit JSON File

Edit `src/extracted_locations.json`:

JSON

```json
[
  {
    "location": "New Park Name",
    "address": "123 Main Street",
    "town": "Denver",
    "comments": "Trail info here",
    "meetingNotes": "Meet at parking lot"
  },
  ...
]
```

Then rebuild:

Bash

```bash
npm run build
npm run dist:win
```

Option 2: Use Python Script

Use the included extraction script:

Python

```
# extract_all_locations.py
import openpyxl
import json

# Load Excel file
wb = openpyxl.load_workbook('your_file.xlsx')
# Extract locations
# Save to JSON
```

Customization

Change App Name

1. Edit `package.json`:

   JSON

   ```json
   "name": "your-app-name",
   "productName": "Your App Name"
   ```

2. Edit `public/index.html`:

   HTML

   ```html
   <title>Your App Name</title>
   ```

Change Window Size

Edit `main.js`:

JavaScript

```javascript
const mainWindow = new BrowserWindow({
  width: 1600,  // Change width
  height: 1000  // Change height
});
```

Change PDF Layout

Edit `src/App.js` in the `generatePdf()` function:

JavaScript

```
// Change orientation
const doc = new jsPDF('portrait', 'pt', 'letter');

// Change font size
doc.setFontSize(14);

// Change colors
doc.setTextColor(255, 0, 0);  // Red text
```

Add Custom Styles

Edit `src/App.css`:

CSS

```
.custom-button {
  background-color: #your-color;
  font-size: 16px;
}
```

Dependencies

Production Dependencies

- **react**: ^18.2.0 - UI library
- **react-dom**: ^18.2.0 - React DOM rendering
- **jspdf**: ^2.5.1 - PDF generation

Development Dependencies

- **electron**: ^38.7.2 - Desktop app framework
- **electron-builder**: ^26.0.12 - Installer builder
- **webpack**: ^5.104.1 - Module bundler
- **babel**: Various packages - JavaScript compiler
- **@babel/preset-react**: React JSX support
- **style-loader**, **css-loader**: CSS processing

File Sizes

- Source code (compressed): ~96 KB

- node_modules (after install): ~500 MB
- Built app (dist folder): ~1 MB
- Windows installer: ~96 MB

Version Information

- **App Version**: 2.0.0
- **Node.js**: 22.13.0
- **Electron**: 38.7.2
- **React**: 18.2.0
- **Build Date**: January 4, 2026

License

This is a custom application for OLLI (Osher Lifelong Learning Institute). All rights reserved.

Support

For questions or issues:

1. Check the troubleshooting section above
2. Review the code comments in App.js
3. Consult Electron and React documentation

Additional Resources

- **Electron Docs**: https://www.electronjs.org/docs
- **React Docs**: https://react.dev
- **jsPDF Docs**: https://github.com/parallax/jsPDF
- **Webpack Docs**: https://webpack.js.org

Created: January 4, 2026

Last Updated: January 4, 2026

Status: Production Ready