**Aim:** Implementing multiple screen navigation and nested navigation using solutions provided by React Navigation

**Description:**

**Task Listing:** Displays tasks in a structured format.

**Export Options:** Allows exporting tasks in CSV or PDF formats.

**Progress Chart:** Displays task progress (completed vs. pending) via a pie chart.

**Interactive Buttons:** Toggles visibility of the progress chart and facilitates task deletion or status updates.

This component registers Chart.js components and integrates navigation options to provide a multi-screen, interactive interface.

**Objective:**

The objective is to:

1. Provide tools for users to manage tasks efficiently using features like export options and task status updates.
2. Implement a visual representation of task progress using a pie chart.
3. Facilitate smooth multi-screen navigation and toggle between views like the task list and the progress chart.

**Implementation:**

Key Features:

1. Task Export:
   * CSV Export: The CSVLink from react-csv allows exporting task data with titles and statuses into a CSV file.
   * PDF Export: The jsPDF library is used to generate and download a PDF file containing task details.
2. Progress Chart:
   * Displays the number of completed vs. pending tasks.
   * Uses chart.js for rendering a visually appealing pie chart.
3. Toggle Buttons:
   * A button toggles the visibility of the progress chart.
   * Buttons for exporting tasks are positioned in the same row for better alignment.
4. Task Operations:
   * Tasks can be deleted or updated via onDeleteTask and onUpdateTask props.
5. React Navigation:
   * This component is designed for integration with React Navigation to toggle between screens.

**Dependencies:** Ensure the following packages are installed:

* react-csv
* jspdf
* chart.js
* react-chartjs-2

**Index.html**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8" />*

*<meta name="viewport" content="width=device-width, initial-scale=1.0" />*

*<title>Task Manager</title>*

*</head>*

*<body>*

*<div id="root"></div>*

*</body>*

*</html>*

**Header.js**

*import React from 'react';*

*import { ToastContainer, toast } from 'react-toastify';*

*import 'react-toastify/dist/ReactToastify.css';*

*const Header = ({ onAddTask }) => {*

*const [taskTitle, setTaskTitle] = React.useState("");*

*const handleAddClick = () => {*

*if (taskTitle.trim()) {*

*onAddTask(taskTitle);*

*toast.success('Task added successfully!');*

*setTaskTitle(""); // Reset input field after adding task*

*} else {*

*toast.error('Please enter a task!');*

*}*

*};*

*return (*

*<div className="header">*

*<h1>Task Manager</h1>*

*<input*

*type="text"*

*placeholder="Enter task title"*

*value={taskTitle}*

*onChange={(e) => setTaskTitle(e.target.value)}*

*className="task-input"*

*/>*

*<button onClick={handleAddClick} className="add-task-btn">Add Task</button>*

*<ToastContainer />*

*</div>*

*);*

*};*

*export default Header;*

**Notification.js**

*import { ToastContainer } from 'react-toastify';*

*import 'react-toastify/dist/ReactToastify.css';*

*const Notification = () => {*

*return <ToastContainer position="top-right" autoClose={5000} />;*

*};export default Notification;*

**PieChart.js**

*// src/pages/PieChartPage.js*

*import React from 'react';*

*import { useLocation } from 'react-router-dom';*

*import { PieChart, Pie, Cell, Tooltip, Legend } from 'recharts';*

*const PieChartPage = () => {*

*const location = useLocation();*

*const { completedTasks, pendingTasks } = location.state || {};*

*const data = [*

*{ name: 'Completed', value: completedTasks },*

*{ name: 'Pending', value: pendingTasks }*

*];*

*return (*

*<div>*

*<h1>Task Progress</h1>*

*<PieChart width={400} height={400}>*

*<Pie*

*data={data}*

*dataKey="value"*

*nameKey="name"*

*outerRadius={150}*

*fill="#8884d8"*

*>*

*{data.map((entry, index) => (*

*<Cell key={`cell-${index}`} fill={index % 2 === 0 ? '#82ca9d' : '#8884d8'} />*

*))}*

*</Pie>*

*<Tooltip />*

*<Legend />*

*</PieChart>*

*</div>*

*);*

*};*

*export default PieChartPage;*

**Task.js**

*import React from 'react';*

*import { FaTrash, FaCheckCircle, FaCircle } from 'react-icons/fa';*

*const Task = ({ task, onDelete, onUpdate }) => {*

*return (*

*<div className="task">*

*<div className="task-info">*

*<h3>{task.title}</h3>*

*<span className={`status ${task.status.toLowerCase()}`}>{task.status}</span>*

*</div>*

*<div className="task-actions">*

*<button onClick={() => onUpdate(task.status === 'Completed' ? 'Pending' : 'Completed')} className="update-btn">*

*{task.status === 'Completed' ? <FaCircle /> : <FaCheckCircle />}*

*</button>*

*<button onClick={onDelete} className="delete-btn"><FaTrash /></button>*

*</div>*

*</div>*

*);*

*};*

*export default Task;*

**TaskDeatils.js**

*import React, { useState } from 'react';*

*import '../styles/TaskDetails.module.css';*

*const TaskDetails = ({ task, onSave }) => {*

*const [title, setTitle] = useState(task.title);*

*const [description, setDescription] = useState(task.description);*

*const [status, setStatus] = useState(task.status);*

*const handleSave = () => {*

*onSave({ ...task, title, description, status });*

*};*

*return (*

*<div className="task-details">*

*<h1>Edit Task</h1>*

*<input*

*type="text"*

*value={title}*

*onChange={(e) => setTitle(e.target.value)}*

*placeholder="Task title"*

*/>*

*<textarea*

*value={description}*

*onChange={(e) => setDescription(e.target.value)}*

*placeholder="Task description"*

*/>*

*<select*

*value={status}*

*onChange={(e) => setStatus(e.target.value)}*

*>*

*<option value="incomplete">Incomplete</option>*

*<option value="complete">Complete</option>*

*</select>*

*<button onClick={handleSave}>Save Changes</button>*

*</div>*

*);*

*};*

*export default TaskDetails;*

**TaskList.js**

*import React, { useState } from 'react';*

*import { CSVLink } from 'react-csv';*

*import { jsPDF } from 'jspdf';*

*import { Pie } from 'react-chartjs-2';*

*import { Chart as ChartJS, CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend, ArcElement } from 'chart.js';*

*import Task from './Task';*

*// Registering chart.js components*

*ChartJS.register(CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend, ArcElement);*

*const TaskList = ({ tasks, onDeleteTask, onUpdateTask }) => {*

*const [showPieChart, setShowPieChart] = useState(false);*

*// Function to export tasks as PDF*

*const exportPDF = () => {*

*const doc = new jsPDF();*

*doc.setFontSize(12);*

*let y = 20;*

*// Add header*

*doc.text("Task Manager - Task List", 14, y);*

*y += 10;*

*// Loop through tasks and add them to PDF*

*tasks.forEach((task, index) => {*

*doc.text(`${index + 1}. ${task.title} - Status: ${task.status}`, 14, y);*

*y += 10;*

*});*

*doc.save('task-list.pdf');*

*};*

*// Pie Chart Data*

*const completedTasks = tasks.filter(task => task.status === 'Completed').length;*

*const pendingTasks = tasks.length - completedTasks;*

*const data = {*

*labels: ['Completed', 'Pending'],*

*datasets: {*

*data: [completedTasks, pendingTasks],*

*backgroundColor: ['#4caf50', '#f44336'], // Green for completed, Red for pending*

*hoverBackgroundColor: ['#45a049', '#e53935'],*

*},*

*],*

*};*

*return (*

*<div className="task-list">*

*<div className="export-buttons">*

*<CSVLink*

*data={tasks.map(task => ({ title: task.title, status: task.status }))}*

*filename="tasks.csv"*

*className="export-btn"*

*>*

*Export as CSV*

*</CSVLink>*

*<button onClick={exportPDF} className="export-btn">Export as PDF</button>*

*</div>*

*<button*

*onClick={() => setShowPieChart(!showPieChart)}*

*className="pie-chart-btn"*

*>*

*{showPieChart ? 'Hide Progress Chart' : 'Show Progress Chart'}*

*</button>*

*{showPieChart && (*

*<div className="pie-chart-container">*

*<Pie data={data} />*

*</div>*

*)}*

*{tasks.map((task, index) => (*

*<Task*

*key={index}*

*task={task}*

*onDelete={() => onDeleteTask(index)}*

*onUpdate={(status) => onUpdateTask(index, status)}*

*/>*

*))}*

*</div>*

*);*

*};*

*export default TaskList;*

**PieChartPage.js**

*import React from 'react';*

*import { Pie } from 'react-chartjs-2';*

*import { useLocation } from 'react-router-dom';*

*import { Chart as ChartJS, ArcElement, Tooltip, Legend } from 'chart.js';*

*// Registering chart.js components*

*ChartJS.register(ArcElement, Tooltip, Legend);*

*const PieChartPage = () => {*

*const { state } = useLocation(); // Access state passed from TaskListPage*

*const data = {*

*labels: ['Completed', 'Pending'],*

*datasets: [*

*{*

*data: [state.completedTasks, state.pendingTasks],*

*backgroundColor: ['#4caf50', '#f44336'],*

*hoverBackgroundColor: ['#45a049', '#e53935'],*

*},*

*],*

*};*

*return (*

*<div className="pie-chart-container">*

*<h2>Task Progress</h2>*

*<Pie data={data} />*

*</div>*

*);*

*};*

*export default PieChartPage;*

**TaskDeatilsPage.js**

*import React from 'react';*

*import TaskDetails from '../components/TaskDetails';*

*const TaskDetailsPage = () => {*

*return (*

*<div>*

*<h1>Task Details</h1>*

*<TaskDetails task={{ title: "Task Title", description: "Task Description", status: "Incomplete" }} onSave={() => {}} />*

*</div>*

*);*

*};*

*export default TaskDetailsPage;*

**TaskListPage.js**

*import React, { useState } from 'react';*

*import { CSVLink } from 'react-csv';*

*import { jsPDF } from 'jspdf';*

*import { Pie } from 'react-chartjs-2';*

*import { Chart as ChartJS, CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend, ArcElement } from 'chart.js';*

*import Task from './Task';*

*// Registering chart.js components*

*ChartJS.register(CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend, ArcElement);*

*const TaskList = ({ tasks, onDeleteTask, onUpdateTask }) => {*

*const [showPieChart, setShowPieChart] = useState(false);*

*// Function to export tasks as PDF*

*const exportPDF = () => {*

*const doc = new jsPDF();*

*doc.setFontSize(12);*

*let y = 20;*

*// Add header*

*doc.text("Task Manager - Task List", 14, y);*

*y += 10;*

*// Loop through tasks and add them to PDF*

*tasks.forEach((task, index) => {*

*doc.text(`${index + 1}. ${task.title} - Status: ${task.status}`, 14, y);*

*y += 10;*

*});*

*doc.save('task-list.pdf');*

*};*

*// Pie Chart Data*

*const completedTasks = tasks.filter(task => task.status === 'Completed').length;*

*const pendingTasks = tasks.length - completedTasks;*

*const data = {*

*labels: ['Completed', 'Pending'],*

*datasets: [*

*{*

*data: [completedTasks, pendingTasks],*

*backgroundColor: ['#4caf50', '#f44336'], // Green for completed, Red for pending*

*hoverBackgroundColor: ['#45a049', '#e53935'],*

*},*

*],*

*};*

*return (*

*<div className="task-list">*

*<div className="export-buttons">*

*<CSVLink*

*data={tasks.map(task => ({ title: task.title, status: task.status }))}*

*filename="tasks.csv"*

*className="export-btn"*

*>*

*Export as CSV*

*</CSVLink>*

*<button onClick={exportPDF} className="export-btn">Export as PDF</button>*

*</div>*

*<button*

*onClick={() => setShowPieChart(!showPieChart)}*

*className="pie-chart-btn"*

*>*

*{showPieChart ? 'Hide Progress Chart' : 'Show Progress Chart'}*

*</button>*

*{showPieChart && (*

*<div className="pie-chart-container">*

*<Pie data={data} />*

*</div>*

*)}*

*{tasks.map((task, index) => (*

*<Task*

*key={index}*

*task={task}*

*onDelete={() => onDeleteTask(index)}*

*onUpdate={(status) => onUpdateTask(index, status)}*

*/>*

*))}*

*</div>*

*);*

*};*

*export default TaskList;*

**Global.css**

*body {*

*font-family: Arial, sans-serif;*

*background-color: #f4f7f6;*

*margin: 0;*

*padding: 0;*

*}*

*.app {*

*max-width: 800px;*

*margin: 0 auto;*

*padding: 20px;*

*}*

*.header {*

*background-color: #4CAF50;*

*color: white;*

*padding: 20px;*

*text-align: center;*

*}*

*.task-input {*

*padding: 10px;*

*width: 70%;*

*margin-right: 10px;*

*border: 1px solid #ccc;*

*border-radius: 4px;*

*}*

*.add-task-btn {*

*padding: 10px 20px;*

*background-color: #4CAF50;*

*color: white;*

*border: none;*

*border-radius: 4px;*

*cursor: pointer;*

*}*

*.add-task-btn:hover {*

*background-color: #45a049;*

*}*

*.task-list {*

*margin-top: 20px;*

*}*

*.task {*

*display: flex;*

*justify-content: space-between;*

*align-items: center;*

*background-color: #fff;*

*padding: 15px;*

*margin: 10px 0;*

*border: 1px solid #ddd;*

*border-radius: 8px;*

*}*

*.task-info {*

*display: flex;*

*align-items: center;*

*}*

*.task-info h3 {*

*margin-right: 10px;*

*}*

*.status {*

*padding: 5px 10px;*

*border-radius: 4px;*

*font-weight: bold;*

*}*

*.status.pending {*

*background-color: #f39c12;*

*color: white;*

*}*

*.status.completed {*

*background-color: #2ecc71;*

*color: white;*

*}*

*.task-actions button {*

*background: none;*

*border: none;*

*cursor: pointer;*

*padding: 5px;*

*font-size: 18px;*

*}*

*.update-btn {*

*color: #3498db;*

*}*

*.delete-btn {*

*color: #e74c3c;*

*}*

*.update-btn:hover,*

*.delete-btn:hover {*

*opacity: 0.7;*

*}*

*/\* Add these styles to your existing Global.css \*/*

*.export-buttons {*

*display: flex;*

*justify-content: flex-start;*

*margin-bottom: 20px;*

*}*

*.export-btn {*

*padding: 10px 20px;*

*margin-right: 10px;*

*background-color: #007bff;*

*color: white;*

*border: none;*

*border-radius: 4px;*

*cursor: pointer;*

*font-size: 16px;*

*}*

*.export-btn:hover {*

*background-color: #0056b3;*

*}*

*/\* Add these styles to your existing Global.css \*/*

*/\* Button to toggle the Pie Chart visibility \*/*

*.pie-chart-btn {*

*padding: 10px 20px;*

*background-color: #007bff;*

*color: white;*

*border: none;*

*border-radius: 4px;*

*cursor: pointer;*

*font-size: 16px;*

*margin-top: 20px;*

*margin-bottom: 20px;*

*}*

*.pie-chart-btn:hover {*

*background-color: #0056b3;*

*}*

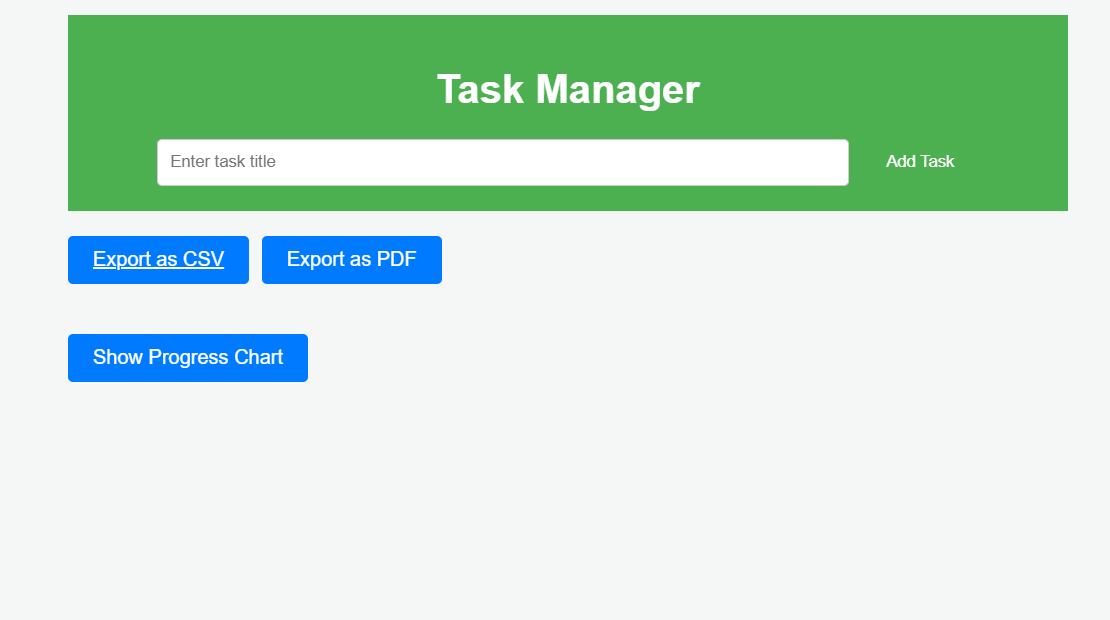
*/\* Container for Pie Chart \*/*

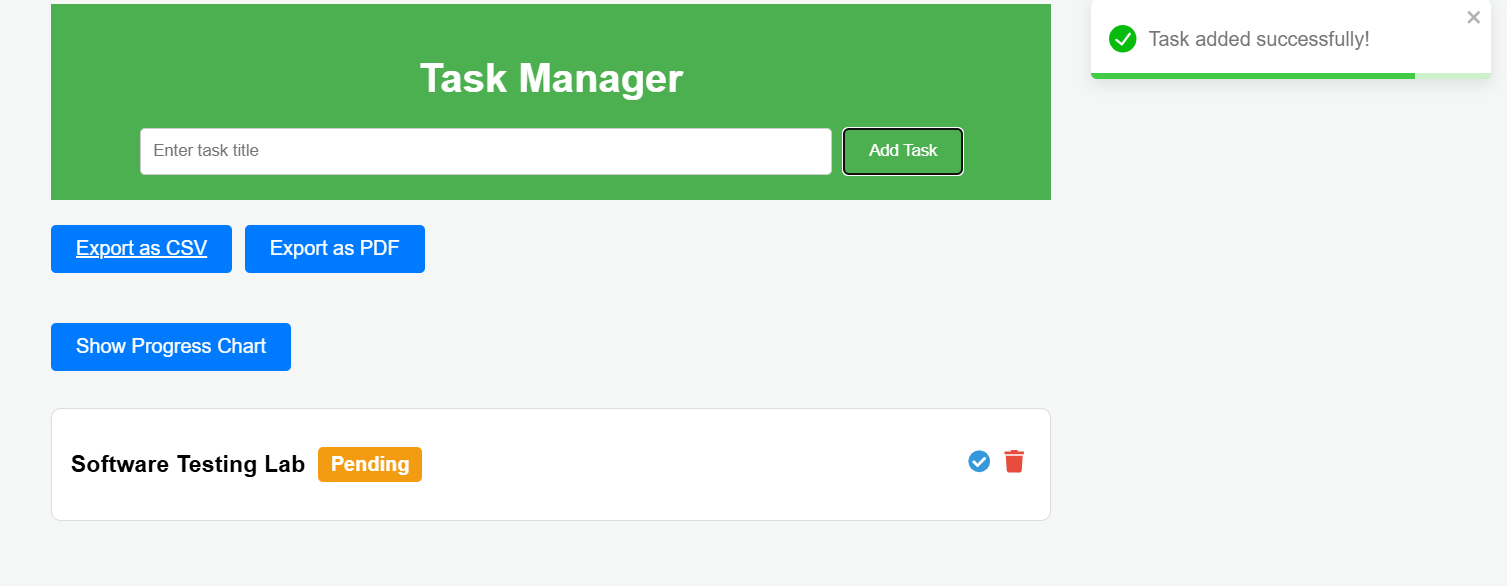
*.pie-chart-container {*

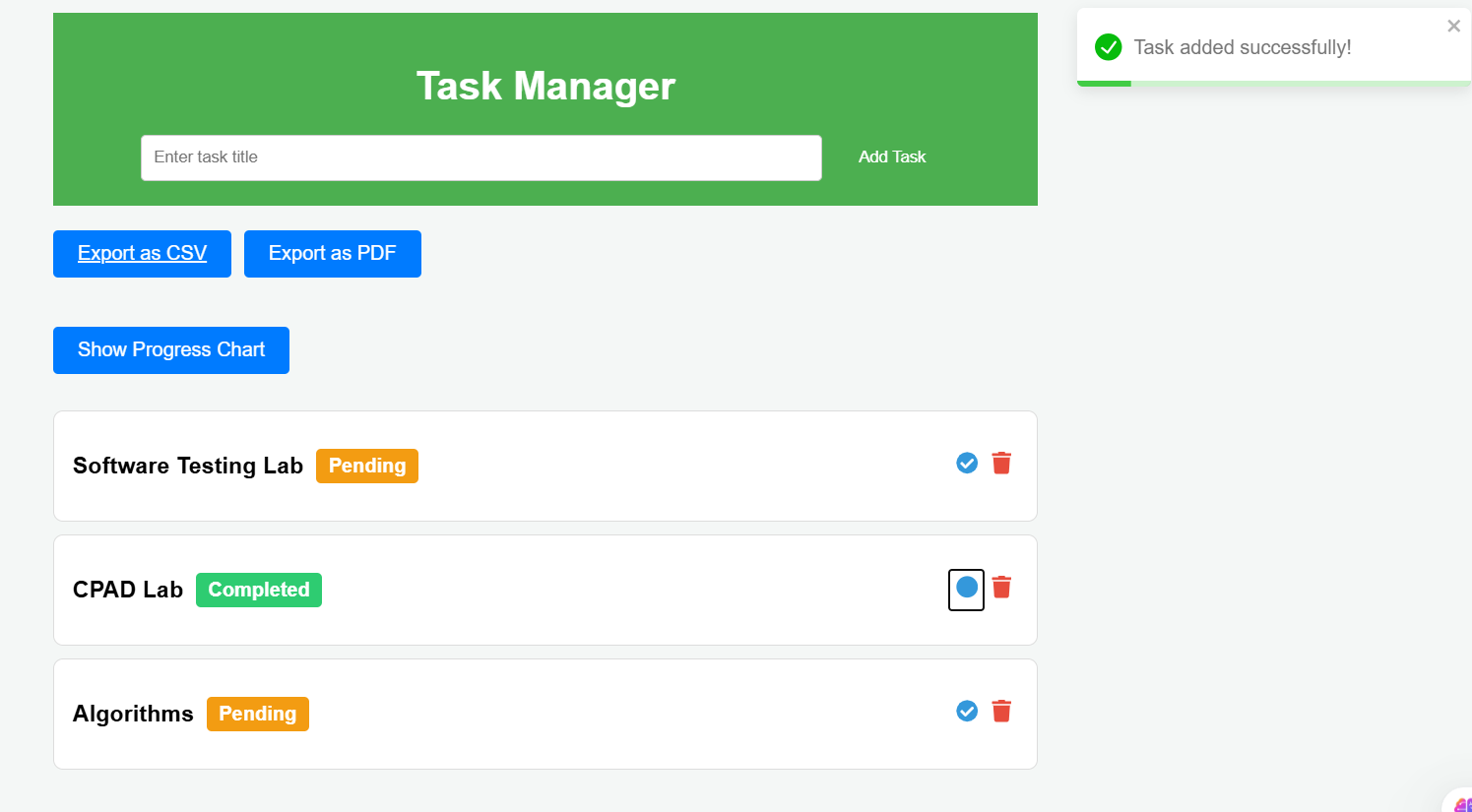
*max-width: 500px;*

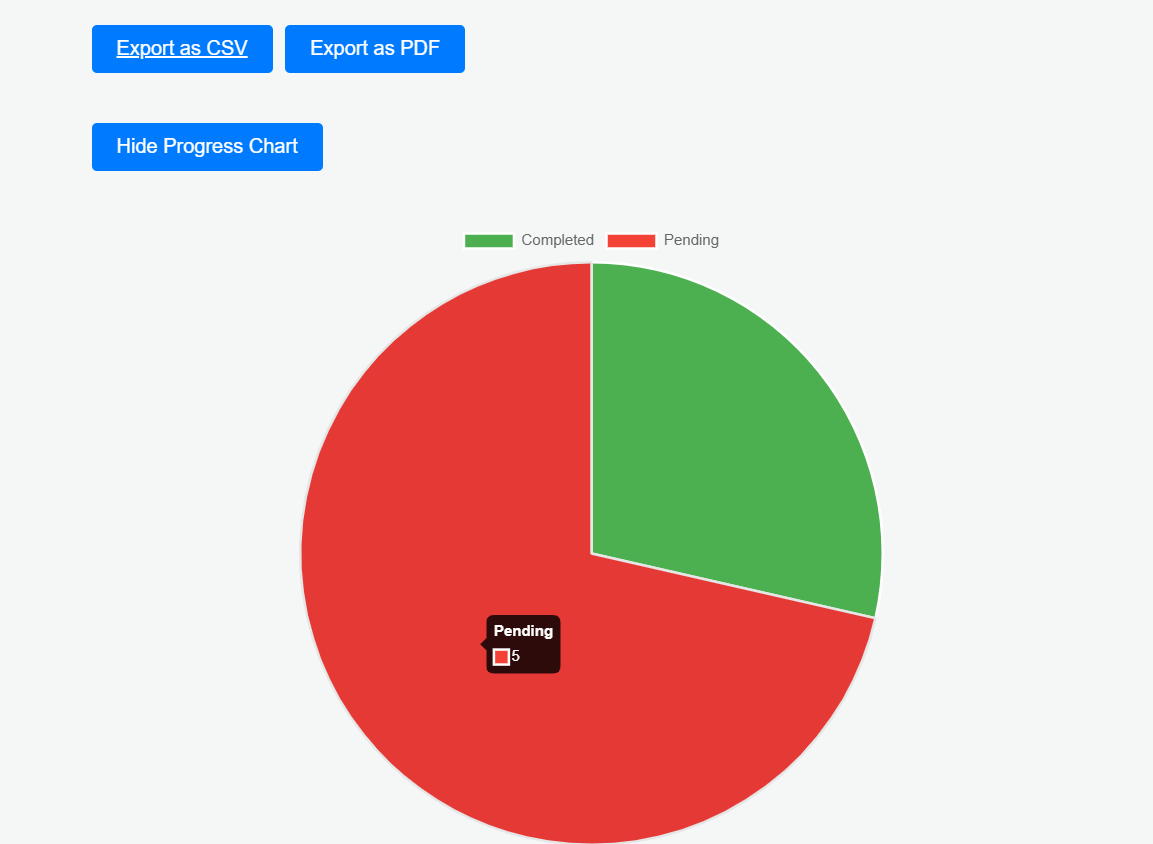
*margin: 20px auto;*

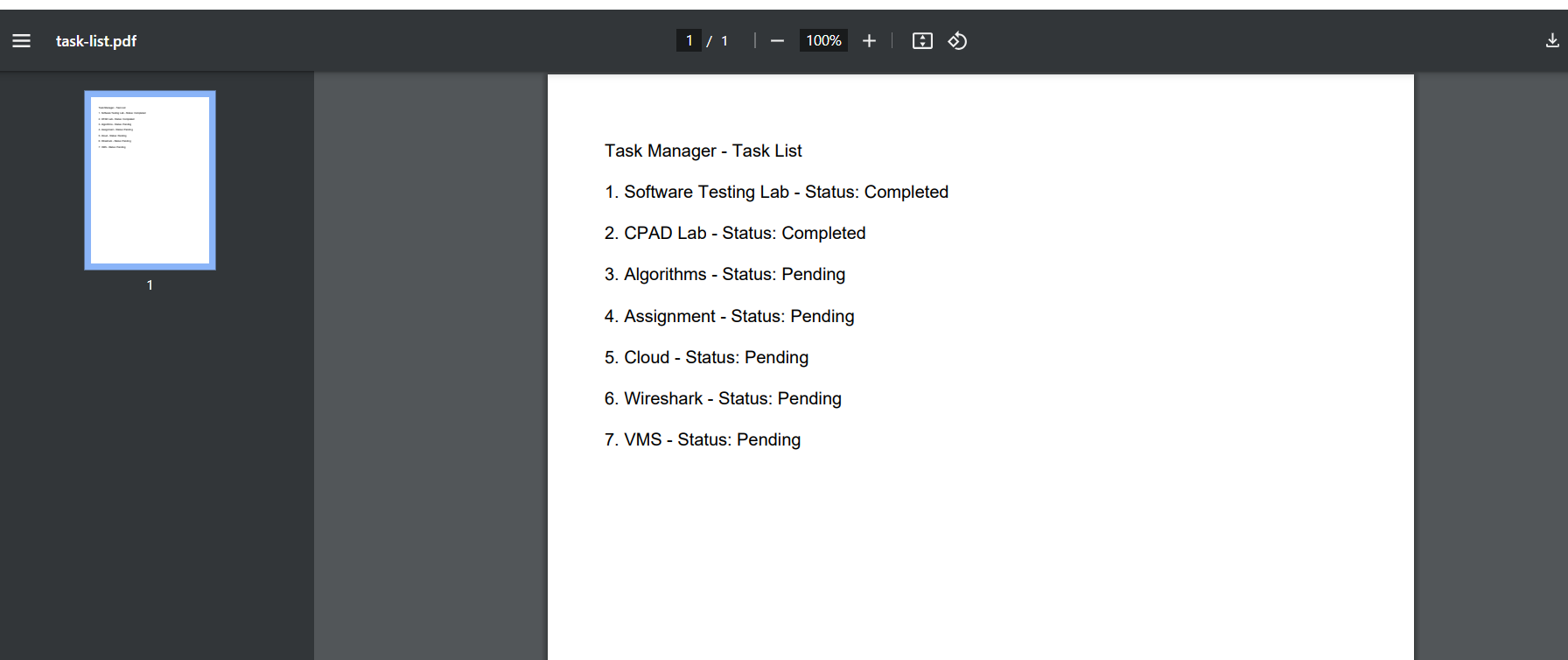
*}*

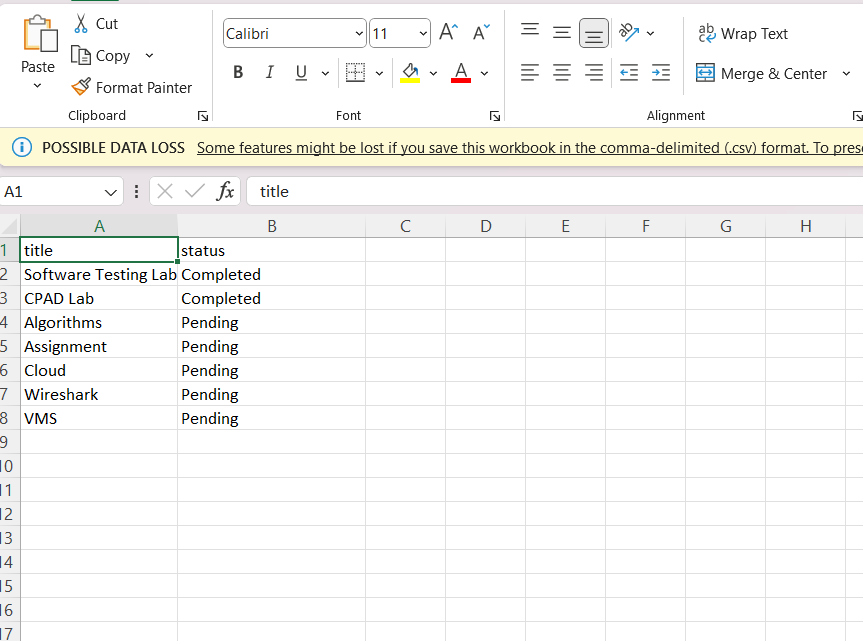
**

**

**

**

**

**

**Conclusion:**

The TaskList component combines robust functionality with visual analytics, enhancing the user experience. By integrating export options, a progress chart, and task status updates, it provides users with comprehensive task management tools. The design aligns with the principles of multi-screen navigation, making it a valuable addition to the Task Manager application.