Project Documentation: Webby Task Management System

This document provides detailed documentation for the Webby Task Management application, built on Next.js, TypeScript, and Zustand.

1.Application Pages

1.1 Home Dashboard (src/app/page.tsx)

Explanation

1. Identifies Location: It clearly displays the title "Home Dashboard."
2. Sets the Goal: It immediately tells the client the main purpose: "Keep your goals visible and your tasks moving forward."
3. Directs Action: It instructs the client where to go next: "Use the Task Tracker to manage your items!"

```
    'use client';
import { Typography, Box } from '@mui/material';

export default function HomePage() {
  return (
    <Box sx={{ display: 'flex', flexDirection: 'column', gap: 4 }}>
      {/*Bordered Heading Home Dashboard */}
      <Box
        sx={{
          border: '2px solid #1976d2',       // Show the Blue border
          borderRadius: 2,                   // For the Rounded corners
          padding: 2,                        // Make Space inside the box
          display: 'inline-block',           // Keeps border tight around text
          backgroundColor: '#0d47a1',        // Royal blue colour background
          color: '#e3f2fd',                  // Light blue colout text
        }}
      >
        <Typography variant="h2">Home Dashboard</Typography>
      </Box>

      {/* Welcome Text to show  */}
      <Typography variant="h4" sx={{ color: '#bbdefb' }}>
        Welcome to Webby Task Management System.
      </Typography>

      {/* Goal Overview Section */}
      <Box>
        <Typography variant="h6" sx={{ color: '#e3f2fd', mb: 1 }}>
          Goal Overview
        </Typography>
        <Typography variant="body2" sx={{ color: '#bbdefb' }}>
          Keep your goals visible and your tasks moving forward. Use the Task Tracker
to manage your items!
        </Typography>
      </Box>
```

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

1.2 Task Tracker Page (src/app/tasks/page.tsx)

Explanation

1. The Control Center: It takes the list of the tasks from the app's internal memory and displays them.
2. Easy Input: The + Add Task button (which opens the Add Task Modal) is clearly located to ensure that users can immediately capture new work.
3. Quick Management: Each task item includes Edit (pencil icon) and Delete (trash can icon) buttons for immediate editing or deletion.
4. Visual Status: A clear indication of the status of each task by colored labels (Chips) for Type and Status, so users can immediately view the status of their work.

```
'use client';
import { Box, Typography, Button, Chip, IconButton } from '@mui/material';
import { useTaskStore } from '@/stores/taskStore';
import { useState } from 'react';
import AddTaskModal from '@/components/AddTaskModal';
import EditIcon from '@mui/icons-material/Edit';
import DeleteIcon from '@mui/icons-material/Delete';

export default function TasksPage() {
  const tasks = useTaskStore((state) => state.tasks);
  const deleteTask = useTaskStore((state) => state.deleteTask);
  const [open, setOpen] = useState(false);
  const [editingTask, setEditingTask] = useState(null);

  const handleAddClick = () => {
    setEditingTask(null); // New task
    setOpen(true);
  };

  const handleEditClick = (task) => {
    setEditingTask(task); // Existing task
    setOpen(true);
  };

  return (
    <Box sx={{ display: 'flex', flexDirection: 'column', gap: 4 }}>
      {/* Show the full-width Header */}
      <Box
        sx={{
          border: '2px solid #1976d2',
          borderRadius: 2,
          padding: 2,
```

```
      backgroundColor: '#0d47a1',
      color: '#e3f2fd',
    }}
  >
    <Typography variant="h2">Task Tracker</Typography>
  </Box>

  {/* To Add Task Button Below Header */}
  <Box sx={{ display: 'flex', justifyContent: 'flex-end' }}>
    <Button variant="contained" color="primary" onClick={handleAddClick}>
      + Add Task
    </Button>
  </Box>

  {/* For Modal */}
  <AddTaskModal
    open={open}
    onClose={() => setOpen(false)}
    editingTask={editingTask}
  />

  {/* For the Task List */}
  <Box sx={{ display: 'flex', flexDirection: 'column', gap: 2 }}>
    {tasks.length === 0 ? (
      <Typography variant="body1" sx={{ color: '#bbdefb' }}>
        No tasks yet. Click “+ Add Task” to get started!
      </Typography>
    ) : (
      tasks.map((task) => (
        <Box
          key={task.id}
          sx={{
            backgroundColor: '#0d47a1',
            p: 2,
            borderRadius: 2,
            position: 'relative',
            display: 'flex',
            flexDirection: 'column',
            gap: 1,
            color: '#e3f2fd',
          }}
        >
          <Typography variant="h6">{task.name}</Typography>

          <Box sx={{ display: 'flex', gap: 1 }}>
            <Chip label={task.type} color="primary" />
            <Chip
              label={task.status}
              color={
                task.status === 'Pending'
                  ? 'warning'
                  : task.status === 'In Progress'
```

```
                      ? 'info'
                      : 'success'
                  }
                />
              </Box>

              <Box sx={{ position: 'absolute', top: 8, right: 8, display: 'flex', gap:
1 }}>
                <IconButton size="small" color="info" onClick={() =>
handleEditClick(task)}>
                  <EditIcon />
                </IconButton>
                <IconButton size="small" color="error" onClick={() =>
deleteTask(task.id)}>
                  <DeleteIcon />
                </IconButton>
              </Box>
            </Box>
          ))
        )}
      </Box>
    </Box>
  );
}
```

1.3 About Us Page (src/app/about/page.tsx)

Explanation

A simple informational page providing necessary metadata. It documents the core technology stack and the architectural philosophy behind the project, using MUI Chip components to visually list the key technologies.

```
'use client';
import { Box, Typography, Chip } from '@mui/material';

export default function AboutPage() {
  return (
    <Box sx={{ display: 'flex', flexDirection: 'column', gap: 4 }}>
      <Box // border heading: creates border heading on the about us page
        sx={{
          border: '2px solid #1976d2',
          borderRadius: 2,
          padding: 2,
          backgroundColor: '#0d47a1',
          color: '#e3f2fd',
        }}
      >
        <Typography variant="h2">About Us</Typography>
      </Box>

      <Typography variant="h4" sx={{ color: '#bbdefb' }}>
        Author: Jun
```

```
      </Typography>

      {/* This part is for Mission Section */}
      <Box>
        <Typography variant="h6" sx={{ color: '#e3f2fd', mb: 1 }}>
          Mission
        </Typography>
        <Typography variant="body2" sx={{ color: '#bbdefb' }}>
          To build a clean, maintainable, and user-friendly interfaces that helps
users to manage thier tasks.
        </Typography>
      </Box>

      {/* This part is for Tech Stack Section */}
      <Box>
        <Typography variant="h6" sx={{ color: '#e3f2fd', mb: 1 }}>
          Tech Stack
        </Typography>
        <Box sx={{ display: 'flex', gap: 1, flexWrap: 'wrap' }}>
          <Chip label="Next.js" color="primary" />
          <Chip label="TypeScript" color="primary" />
          <Chip label="Zustand" color="primary" />
          <Chip label="MUI" color="primary" />
          <Chip label="React Hook Form" color="primary" />
        </Box>
      </Box>
    </Box>
  );
}
```

1.4 API List Page (src/app/api-list/page.tsx)

Explanation

1. Technical Specification: This list contains technical information about other software systems. At present, this is just displaying GitHub public projects, but in a real application this would be a list of tools we might want to integrate against (such as a calendar or accounts management).
2. The Waiting Progress (The Spinner): When this page is loaded, you temporarily see a circular rotating (the CircularProgress) element. This is normal! This is because the app must talk to the internet (GitHub) to get this list of information.
3. The Benefit: The app is programmed to wait for this information without skipping or jumping around. It maintains a smooth and stable user interface while the data is being retrieved.

```
'use client';
import { useEffect, useState } from 'react';
import { Box, Typography, CircularProgress, Card, CardContent } from '@mui/material';

type Repo = {
```

```
  id: number;
  name: string;
  description: string;
  html_url: string;
};

export default function ApiListPage() {
  const [repos, setRepos] = useState<Repo[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch('https://api.github.com/users/github/repos?per_page=5') // this link is from
github
      .then((res) => res.json())
      .then((data) => {
        setRepos(data);
        setLoading(false);
      });
  }, []);

  return (
    <Box sx={{ display: 'flex', flexDirection: 'column', gap: 4 }}>
      {/* This shows the blue bordered header */}
      <Box
        sx={{
          border: '2px solid #1976d2',
          borderRadius: 2,
          padding: 2,
          backgroundColor: '#0d47a1',
          color: '#e3f2fd',
        }}
      >
        <Typography variant="h2">API List</Typography>
      </Box>

      {/* this is for the Loading section */}
      {loading ? (
        <CircularProgress color="primary" />
      ) : (
        repos.map((repo) => (
          <Card key={repo.id} sx={{ backgroundColor: '#1e1e1e', color: 'white' }}>
            <CardContent>
              <Typography variant="h6">{repo.name}</Typography>
              <Typography variant="body2" sx={{ color: '#ccc', mb: 1 }}>
                {repo.description || 'No description provided.'}
              </Typography>
              <Typography
                variant="body2"
                component="a"
                href={repo.html_url}
                target="_blank"
                rel="noopener noreferrer"
```

```
                sx={{ color: '#90caf9', textDecoration: 'underline' }}
              >
                View on GitHub
              </Typography>
            </CardContent>
          </Card>
        ))
      )}
    </Box>
  );
}
```

2. Layout and Styling

2.1 Root Layout (src/app/layout.tsx)

Explanation

1. The Big Picture: It structures the app into a single screen divided into two major sections:
2. Sidebar: The Sidebar component (the dark menu on the left with "Home Dashboard," "Task Tracker," etc.) is permanently located on the left side.
3. Main Content Area: The main area occupies the rest of the screen (in the right) and contains the content which actually changes (the dashboard, the task list, the API list, etc.).

Aesthetics & Usability:

1. It helps the app to use a dark theme (backgroundColor: '#121212', color: 'white').
2. It ensures that the main content area has room to scroll (overflowY: 'auto') if a page is getting too long.

```
import './globals.css';
import Sidebar from '@/components/Sidebar';

export const metadata = {
  title: 'Jun’s Webby Assessment',
  description: 'Task Tracker built with Next.js, TypeScript, Zustand, and MUI',
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <head>
        <meta name="viewport" content="width=device-width, initial-scale=1" />
      </head>
```

```
      <body style={{ margin: 0, padding: 0 }}>
        <div style={{ display: 'flex', height: '100vh', width: '100vw' }}>
          <Sidebar />
          <main style={{ flex: 1, padding: '32px', overflowY: 'auto', backgroundColor:
'#121212', color: 'white' }}>
            {children}
          </main>
        </div>
      </body>
    </html>
  );
}
```

2.2 Global CSS (src/app/globals.css)

Explanation

A minimal global CSS file that sets the base dark-mode colors (--background-rgb, --foreground-rgb) and ensures the application uses the 'Inter' font for visual consistency.

```
@import "tailwindcss";

:root {
  --background: #ffffff;
  --foreground: #171717;
}

@theme inline {
  --color-background: var(--background);
  --color-foreground: var(--foreground);
  --font-sans: var(--font-geist-sans);
  --font-mono: var(--font-geist-mono);
}

@media (prefers-color-scheme: dark) {
  :root {
    --background: #0a0a0a;
    --foreground: #ededed;
  }
}

body {
  background: var(--background);
  color: var(--foreground);
  font-family: Arial, Helvetica, sans-serif;
}

html, body {
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
```

```
  font-family: var(--font-geist-sans);
  background-color: #121212;
  color: white;
}
```

3. Shared Components

3.1 Sidebar Component (src/components/Sidebar.tsx)

Explanation

1. Fixed Navigation: The sidebar is fixed on the screen, making it easily accessible at all times, giving a consistent and reliable way of navigating the Webby Task Manager.
2. Identification: It is marked with the app title of "Jun's Webby Assessment" on the very top so that the user always knows where they are.
3. Key Links: It holds all the links the client requires to utilize the app:
4. Home Dashboard (The initial point).
5. The main work area, known as Task Tracker.
6. About Us (Documentation and information)
7. API List (Technical connection information).

```
'use client';
import Link from 'next/link';
import { Box, Typography } from '@mui/material';

export default function Sidebar() {
  return (
    <Box
      sx={{
        width: 240,
        height: '100vh',
        backgroundColor: '#1e1e1e',
        color: 'white',
        p: 3,
        display: 'flex',
        flexDirection: 'column',
        gap: 3,
        borderRight: '1px solid #333',
      }}
    >
      <Typography variant="h5">Jun's Webby Assessment</Typography>
      <Link href="/" style={linkStyle}>Home Dashboard</Link>
      <Link href="/tasks" style={linkStyle}>Task Tracker</Link>
      <Link href="/about" style={linkStyle}>About Us</Link>
      <Link href="/api-list" style={linkStyle}>API List</Link>
    </Box>
  );
}
```

```
const linkStyle = {
  color: 'white',
  fontSize: '1.1rem',
  textDecoration: 'none',
  padding: '8px 0',
};
```

3.2 Add Task Modal Component (src/components/AddTaskModal.tsx)

Explanation

1. New or Edit: It smartly changes its title and button from "Add New Task" to "Edit Task" depending on whether you're creating a new item or modifying an old one.
2. Required Fields: It forces you to enter a "Task Name" before saving. If you try to skip it, the app throws an error message.
3. Organization: It makes sure every task is properly labeled using the standard options for Type (Personal, Work, Security) and Status (Pending, In Progress, Completed).

```
'use client';
import { useEffect } from 'react';
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { z } from 'zod';
import {
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  TextField,
  Button,
  MenuItem,
} from '@mui/material';
import { useTaskStore } from '@/stores/taskStore';
import { v4 as uuidv4 } from 'uuid';

const schema = z.object({
  name: z.string().min(1, 'Task name is required'),
  type: z.string().min(1),
  status: z.string().min(1),
});

type FormData = z.infer<typeof schema>;

type Props = {
  open: boolean;
  onClose: () => void;
  editingTask: { id: string } & FormData | null;
};
```

```
export default function AddTaskModal({ open, onClose, editingTask }: Props) {
  const addTask = useTaskStore((state) => state.addTask);
  const updateTask = useTaskStore((state) => state.updateTask);

  const {
    register,
    handleSubmit,
    formState: { errors },
    reset,
  } = useForm<FormData>({
    resolver: zodResolver(schema),
    defaultValues: {
      name: '',
      type: '',
      status: '',
    },
  });

  useEffect(() => {  // Effect to Handle Edit Mode
    if (editingTask) {
      reset({
        name: editingTask.name,
        type: editingTask.type,
        status: editingTask.status,
      });
    } else {
      reset({
        name: '',
        type: '',
        status: '',
      });
    }
  }, [editingTask, reset]);

  const onSubmit = (data: FormData) => {  //this is for the submission Handler
    if (editingTask) {
      updateTask({ id: editingTask.id, ...data });
    } else {
      addTask({ id: uuidv4(), ...data });
    }
    reset();
    onClose();
  };

  return (
    <Dialog open={open} onClose={onClose}>
      <DialogTitle>{editingTask ? 'Edit Task' : 'Add New Task'}</DialogTitle>
      <DialogContent sx={{ display: 'flex', flexDirection: 'column', gap: 2, mt: 1 }}>
        <TextField
          label="Task Name"
          {...register('name')}
```

```
          error={!!errors.name}
          helperText={errors.name?.message}
          fullWidth
        />
        <TextField
          label="Task Type"
          select
          {...register('type')}
          error={!!errors.type}
          helperText={errors.type?.message}
          fullWidth
        >
          <MenuItem value="Personal">Personal</MenuItem>
          <MenuItem value="Work">Work</MenuItem>
          <MenuItem value="Security">Security</MenuItem>
        </TextField>
        <TextField
          label="Task Status"
          select
          {...register('status')}
          error={!!errors.status}
          helperText={errors.status?.message}
          fullWidth
        >
          <MenuItem value="Pending">Pending</MenuItem>
          <MenuItem value="In Progress">In Progress</MenuItem>
          <MenuItem value="Completed">Completed</MenuItem>
        </TextField>
      </DialogContent>
      <DialogActions>
        <Button onClick={onClose}>Cancel</Button>
        <Button onClick={handleSubmit(onSubmit)} variant="contained">
          {editingTask ? 'Update Task' : '+ Create Task'}
        </Button>
      </DialogActions>
    </Dialog>
  );
}
```

***4. Stores* 4.1 Task Store (stores/taskStore.ts)**

Explanation

1. The Master List: It is the single source of truth that holds all the user's tasks that are created, edited, or completed.
2. Real-Time updates: As soon as you add a new task in the pop-up modal, the modal requests the Task Store to add the data and the Task Tracker list is updated immediately to display the new item.
3. Consistency: The data source is one and the same Store so two copies of your task list will never be displayed anywhere in the app.

```
import { create } from 'zustand';

// This seciton is for task type definition
export type Task = {
  id: string;
  name: string;
  type: string;
  status: string;
};

// This section is for Store shape
type TaskStore = {
  tasks: Task[];
  addTask: (task: Task) => void;
  updateTask: (task: Task) => void;
  deleteTask: (id: string) => void;
};

// this section is Zustand store implementation
export const useTaskStore = create<TaskStore>((set) => ({
  tasks: [],

  // this section is to Add a new task
  addTask: (task) =>
    set((state) => ({
      tasks: [...state.tasks, task],
    })),

  // To Update an existing task by ID
  updateTask: (updatedTask) =>
    set((state) => ({
      tasks: state.tasks.map((task) =>
        task.id === updatedTask.id ? updatedTask : task
      ),
    })),

  // To Delete a task by ID
  deleteTask: (id) =>
    set((state) => ({
      tasks: state.tasks.filter((task) => task.id !== id),
    })),
}));
```