

CodeUpgradeEngine

Complete Repository Source Code

Total Files: 112

Table of Contents

1. .gitignore
2. attached_assets/Pasted-I-told-my-engineer-friend-i-have-a-lod-400-upgrading-engine-i-coded-and-i-use-myself-only-but-it-s-1764572247372_1764572247373.txt
3. client/index.html
4. client/src/App.tsx
5. client/src/components/AppSidebar.tsx
6. client/src/components/ObjectUploader.tsx
7. client/src/components/OrderDetailModal.tsx
8. client/src/components/OrdersTable.tsx
9. client/src/components/OrderTimeline.tsx
10. client/src/components/StatsCard.tsx
11. client/src/components/StatusBadge.tsx
12. client/src/components/ThemeProvider.tsx
13. client/src/components/ThemeToggle.tsx
14. client/src/components/ui/accordion.tsx
15. client/src/components/ui/alert-dialog.tsx
16. client/src/components/ui/alert.tsx
17. client/src/components/ui/aspect-ratio.tsx
18. client/src/components/ui/avatar.tsx
19. client/src/components/ui/badge.tsx
20. client/src/components/ui/breadcrumb.tsx
21. client/src/components/ui/button.tsx
22. client/src/components/ui/calendar.tsx
23. client/src/components/ui/card.tsx
24. client/src/components/ui/carousel.tsx
25. client/src/components/ui/chart.tsx
26. client/src/components/ui/checkbox.tsx
27. client/src/components/ui/collapsible.tsx
28. client/src/components/ui/command.tsx
29. client/src/components/ui/context-menu.tsx
30. client/src/components/ui/dialog.tsx
31. client/src/components/ui/drawer.tsx
32. client/src/components/ui/dropdown-menu.tsx
33. client/src/components/ui/form.tsx
34. client/src/components/ui/hover-card.tsx
35. client/src/components/ui/input-otp.tsx
36. client/src/components/ui/input.tsx
37. client/src/components/ui/label.tsx
38. client/src/components/ui/menubar.tsx
39. client/src/components/ui/navigation-menu.tsx
40. client/src/components/ui/pagination.tsx
41. client/src/components/ui/popover.tsx
42. client/src/components/ui/progress.tsx
43. client/src/components/ui/radio-group.tsx
44. client/src/components/ui/resizable.tsx
45. client/src/components/ui/scroll-area.tsx
46. client/src/components/ui/select.tsx
47. client/src/components/ui/separator.tsx
48. client/src/components/ui/sheet.tsx
49. client/src/components/ui/sidebar.tsx
50. client/src/components/ui/skeleton.tsx
51. client/src/components/ui/slider.tsx
52. client/src/components/ui/switch.tsx
53. client/src/components/ui/table.tsx
54. client/src/components/ui/tabs.tsx
55. client/src/components/ui/textarea.tsx
56. client/src/components/ui/toast.tsx
57. client/src/components/ui/toaster.tsx
58. client/src/components/ui/toggle-group.tsx
59. client/src/components/ui/toggle.tsx
60. client/src/components/ui/tooltip.tsx
61. client/src/hooks/use-mobile.tsx
62. client/src/hooks/use-toast.ts
63. client/src/hooks/useAuth.ts
64. client/src/index.css
65. client/src/lib/authUtils.ts

66. client/src/lib/queryClient.ts
67. client/src/lib/utils.ts
68. client/src/main.tsx
69. client/src/pages/ClientDashboard.tsx
70. client/src/pages/Clients.tsx
71. client/src/pages/Dashboard.tsx
72. client/src/pages/Landing.tsx
73. client/src/pages/not-found.tsx
74. client/src/pages/Orders.tsx
75. client/src/pages/Settings.tsx
76. components.json
77. design_guidelines.md
78. drizzle.config.ts
79. package.json
80. postcss.config.js
81. README.md
82. replit.md
83. revit-addin/LOD400Uploader/App.cs
84. revit-addin/LOD400Uploader/Commands/CheckStatusCommand.cs
85. revit-addin/LOD400Uploader/Commands/UploadSheetsCommand.cs
86. revit-addin/LOD400Uploader/LOD400Uploader.addin
87. revit-addin/LOD400Uploader/LOD400Uploader.csproj
88. revit-addin/LOD400Uploader/Models/Order.cs
89. revit-addin/LOD400Uploader/Services/ApiService.cs
90. revit-addin/LOD400Uploader/Services/PackagingService.cs
91. revit-addin/LOD400Uploader/Views/LoginDialog.xaml
92. revit-addin/LOD400Uploader/Views/LoginDialog.xaml.cs
93. revit-addin/LOD400Uploader/Views/StatusDialog.xaml
94. revit-addin/LOD400Uploader/Views/StatusDialog.xaml.cs
95. revit-addin/LOD400Uploader/Views/UploadDialog.xaml
96. revit-addin/LOD400Uploader/Views/UploadDialog.xaml.cs
97. revit-addin/README.md
98. script/build.ts
99. server/db.ts
100. server/index.ts
101. server/objectStorage.ts
102. server/replitAuth.ts
103. server/routes.ts
104. server/static.ts
105. server/storage.ts
106. server/stripeClient.ts
107. server/vite.ts
108. server/webhookHandlers.ts
109. shared/schema.ts
110. tailwind.config.ts
111. tsconfig.json
112. vite.config.ts

File: .gitignore

```
node_modules
dist
.DS_Store
server/public
vite.config.ts.*
*.tar.gz
```

File: attached_assets/Pasted-I-told-my-engineer-friend-i-have-a-lod-400-upgrading-engine-i-coded-and-i-use-myself-only-but-it-s-1764572247372_1764572247373.txt

I told my engineer friend i have a "lod 400 upgrading engine" i coded and i use myself only but it's local on my pc and i want to know how to make it a company.

The "Bootstrap" Technology Stack

Revit Add-in (Client): C#, .NET Framework, WPF (Free via Visual Studio Community Edition).

Database & Authentication: Supabase (Generous Free Tier).

API Logic & Hosting: Vercel or Netlify (Free Tiers).

Large File Storage: Cloudflare R2 or Backblaze B2 (Extremely low cost).

Payments & VAT (Merchant of Record): Lemon Squeezy or Paddle (No upfront cost; percentage-based).

Email Notifications: Resend (Free Tier).

Phase 1: Infrastructure and Account Setup (Minimal Coding)

This phase focuses on setting up the necessary accounts and tools.

Step 1: Development Environment Setup

1.1. (You/Partner) Install Visual Studio Community Edition (ensure the ".NET desktop development" workload is included for C# and WPF).

1.2. (You/Partner) Install VS Code (for backend/web development) and Git (version control).

1.3. (You) Download the Revit SDK (Software Development Kit) from Autodesk.

Step 2: Source Control Setup

2.1. (You) Create a free GitHub account.

2.2. (Partner) Create three private repositories: LOD400-RevitAddin, LOD400-API, and LOD400-AdminDashboard.

Step 3: Cloud Infrastructure Accounts

3.1. (You) Sign up for Supabase. Create a new project.

3.2. (You) Sign up for Cloudflare R2 (or Backblaze B2). Create a private storage "Bucket" (e.g., lod400-files) and generate API keys.

3.3. (You) Sign up for Vercel (or Netlify). Link it to your GitHub repositories.

3.4. (You) Sign up for Resend.

Step 4: Merchant of Record (MoR) Setup

This is crucial for handling payments and international VAT compliance without hassle.

4.1. (You) Sign up for Lemon Squeezy or Paddle.

4.2. (You) Configure your product: "LOD 400 Sheet Upgrade," priced at 150 SAR.

4.3. (You) Obtain the API Key and the "Webhook Secret" (essential for secure payment confirmation).

Step 5: Environment Configuration

5.1. (Partner) In Vercel/Netlify, securely store all API keys (Supabase, R2, MoR, Resend) as "Environment Variables." This keeps sensitive keys out of the code.

Phase 2: The Backend Engine (Heavy Coding)

This is the central nervous system of your service, built using serverless functions (e.g., Node.js/TypeScript) hosted on Vercel/Netlify.

Step 6: Database Schema Design

6.1. [PARTNER - CODING] Define the database tables in Supabase (PostgreSQL):

Users (Use the built-in Supabase Auth table).

Orders (Columns: OrderID, UserID, SheetCount, TotalPriceSAR, Status [Enum: Pending, Paid, Uploaded, Processing, Complete]).

Files (Columns: FileID, OrderID, FileType [Input/Output], StorageKey).

6.2. [PARTNER - CODING] Implement Row Level Security (RLS) policies on Supabase to ensure users can only access their own orders.

Step 7: The CreateOrder API Endpoint

7.1. [PARTNER - CODING] Develop the endpoint logic:

Receive UserID and SheetCount from the Add-in.

Insert a record into Supabase Orders as 'Pending'.

Call the MoR API (Lemon Squeezy/Paddle) to generate a unique Checkout URL for that specific price, passing the OrderID as metadata.

Return the Checkout URL.

Step 8: The PaymentWebhook Listener

8.1. [PARTNER - CODING] Develop the secure webhook endpoint:

Listen for incoming confirmation from the MoR.

Crucially: Verify the request signature using the Webhook Secret (from Step 4.3) to prevent fraud.

Update the corresponding order in Supabase from 'Pending' to 'Paid'.

Step 9: The RequestUploadURL Endpoint (Pre-signed URLs)

This allows secure, direct-to-storage uploads for large files.

9.1. [PARTNER - CODING] Develop the endpoint logic:

Receive OrderID and verify the status is 'Paid'.

Use the R2/B2 SDK to generate a "Pre-signed PUT URL." This is a temporary (e.g., 1 hour expiry) URL allowing upload only to a specific location (e.g., inputs/

Return the Pre-signed URL.

Step 10: The UploadComplete Endpoint

10.1. [PARTNER - CODING] Develop the endpoint logic:

Receive OrderID.

Update the order status in Supabase to 'Uploaded'.

Use the Resend API to send you an email alert: "New LOD 400 Job Ready."

Phase 3: The Revit Add-in (Heavy C# Coding)

This is the software your clients use inside Revit, developed using C#, WPF, and the Revit API.

Step 11: Add-in Scaffolding and UI

11.1. [PARTNER - CODING] Create the C# Class Library project in Visual Studio. Reference the Revit API DLLs (RevitAPI.dll, RevitAPIUI.dll).

11.2. [PARTNER - CODING] Create the .addin manifest file (XML) so Revit recognizes the plugin.

11.3. [PARTNER - CODING] Implement the IExternalApplication interface to create a custom Ribbon button in Revit.

11.4. [PARTNER - CODING] Design the User Interface using WPF (XAML). Create screens for Login, Sheet Selector, Order Summary, and Order Status.

Step 12: Authentication

12.1. [PARTNER - CODING] Implement the Login/Registration screen using the Supabase Auth C# library to authenticate users securely.

Step 13: Reading the Model and Quoting

13.1. [PARTNER - CODING] Access the active Revit Document.

13.2. [PARTNER - CODING] Use the Revit API's FilteredElementCollector to retrieve all elements of class ViewSheet.

13.3. [PARTNER - CODING] Populate the WPF list and implement the dynamic pricing logic (Count * 150 SAR).

Step 14: The Checkout Process (Client Side)

14.1. [PARTNER - CODING] Implement the "Pay Now" button:

Call the CreateOrder API endpoint (Step 7).

Receive the Checkout URL.

Use System.Diagnostics.Process.Start() to open the user's default web browser.

Implement a background process to poll the backend API until the status changes to 'Paid'.

Step 15: Model Packaging (Crucial Step)

15.1. [PARTNER - CODING] Once 'Paid', identify the main model path.

15.2. [PARTNER - CODING] Use the Revit API to find all linked models (RevitLinkInstance) and resolve their paths.

15.3. [PARTNER - CODING] Use System.IO.Compression (built-in C# library) to create a single ZIP archive containing the main model and all linked files.

Step 16: Robust File Upload (Complex Step)

16.1. [PARTNER - CODING] Call the RequestUploadURL endpoint (Step 9) to get the Pre-signed URL.

16.2. [PARTNER - CODING] Implement Chunked Uploads. Write C# code to read the large ZIP file in smaller segments (e.g., 5-10MB chunks).

16.3. [PARTNER - CODING] Upload each chunk sequentially to the Pre-signed URL, implementing retry logic for network errors.

16.4. [PARTNER - CODING] Display a real-time progress bar in the WPF UI.

16.5. [PARTNER - CODING] Once complete, call the UploadComplete endpoint (Step 10).

Phase 4: Admin Dashboard and Finalization (Web Coding)

This is your interface for managing the jobs, built with web technologies (e.g., React, Vue, or plain HTML/JS) and hosted on Vercel/Netlify.

Step 17: Admin Dashboard UI and Login

17.1. [PARTNER - CODING] Create the web application and implement secure login (using Supabase Auth, restricted to your admin credentials).

17.2. [PARTNER - CODING] Create a dashboard view listing all orders from Supabase, filterable by status.

Step 18: Admin Workflow Logic

18.1. [PARTNER - CODING] Implement "Download Inputs": A button that generates a secure download link (Pre-signed GET URL) from R2/B2 so you can retrieve the c

18.2. (You) You download the files and execute your specialized LOD 300->400 workflow.

18.3. [PARTNER - CODING] Implement "Upload Outputs": An interface for you to upload the finished deliverables. This also uses Pre-signed PUT URLs.

18.4. [PARTNER - CODING] Implement "Mark Complete": A button that updates the status in Supabase to 'Complete' and triggers an email to the client via Resend.

Step 19: Client Download Functionality

19.1. [PARTNER - CODING] In the Revit Add-in (C#), enable the "Download" button in the Order Status tab when the status is 'Complete'.

19.2. [PARTNER - CODING] Implement the download logic by requesting a Pre-signed GET URL from the backend and saving the deliverables locally.

Step 20: Packaging and Protection

20.1. [PARTNER - CODING] Use a free .NET Obfuscator (like Obfuscator) to make the compiled C# code (.dll) difficult to reverse-engineer.

20.2. [PARTNER - CODING] Use Inno Setup (Free/Open Source) to create a professional Windows installer (.exe) for the Add-in.) what do you think ?

File: client/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1" />
    <link rel="icon" type="image/png" href="/favicon.png" />
    <title>LOD 400 Delivery Platform | Professional BIM Model Upgrades</title>
    <meta name="description" content="Professional LOD 300 to LOD 400 BIM model upgrade service. Upload your Revit models and receive production-ready details." />
    <meta property="og:title" content="LOD 400 Delivery Platform" />
    <meta property="og:description" content="Professional BIM model upgrade service - LOD 300 to LOD 400 transformations" />
    <meta property="og:type" content="website" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="stylesheet">
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

File: client/src/App.tsx

```
import { Switch, Route } from "wouter";
import { queryClient } from "../lib/queryClient";
import { QueryClientProvider } from "@tanstack/react-query";
import { Toaster } from "@components/ui/toaster";
import { TooltipProvider } from "@components/ui/tooltip";
import { ThemeProvider } from "@components/ThemeProvider";
import { SidebarProvider } from "@components/ui/sidebar";
import { AppSidebar } from "@components/AppSidebar";
import { useAuth } from "@hooks/useAuth";
import { Loader2 } from "lucide-react";

import Landing from "@pages/Landing";
import Dashboard from "@pages/Dashboard";
import ClientDashboard from "@pages/ClientDashboard";
import Orders from "@pages/Orders";
import Clients from "@pages/Clients";
import Settings from "@pages/Settings";
import NotFound from "@pages/not-found";

function LoadingScreen() {
  return (
    <div className="flex items-center justify-center min-h-screen bg-background">
      <div className="flex flex-col items-center gap-4">
        <Loader2 className="h-8 w-8 animate-spin text-primary" />
        <p className="text-sm text-muted-foreground">Loading...</p>
      </div>
    </div>
  );
}

function AuthenticatedLayout({ children }: { children: React.ReactNode }) {
  const sidebarStyle = {
    "--sidebar-width": "16rem",
    "--sidebar-width-icon": "3rem",
  };

  return (
    <SidebarProvider style={sidebarStyle as React.CSSProperties}>
      <div className="flex h-screen w-full">
        <AppSidebar />
        <div className="flex flex-col flex-1 overflow-hidden">
          {children}
        </div>
      </div>
    </SidebarProvider>
  );
}

function AdminRouter() {
  return (
    <AuthenticatedLayout>
      <Switch>
        <Route path="/" component={Dashboard} />
        <Route path="/orders" component={Orders} />
        <Route path="/clients" component={Clients} />
        <Route path="/settings" component={Settings} />
        <Route component={NotFound} />
      </Switch>
    </AuthenticatedLayout>
  );
}

function ClientRouter() {
  return (
    <AuthenticatedLayout>
      <Switch>
        <Route path="/" component={ClientDashboard} />
        <Route path="/settings" component={Settings} />
        <Route component={NotFound} />
      </Switch>
    </AuthenticatedLayout>
  );
}

function Router() {
  const { isAuthenticated, isLoading, isAdmin } = useAuth();

  if (isLoading) {
    return <LoadingScreen />;
  }

  if (!isAuthenticated) {
    return <Landing />;
  }

  if (isAdmin) {
    return <AdminRouter />;
  }

  return <ClientRouter />;
}

function App() {
  return (
    <QueryClientProvider client={queryClient}>
      <ThemeProvider defaultTheme="light">
        <TooltipProvider>

```

```
        <Toaster />
        <Router />
        <TooltipProvider>
        <ThemeProvider>
        <QueryClientProvider>
    );
}

export default App;
```

File: client/src/components/AppSidebar.tsx

```

import {
  Sidebar,
  SidebarContent,
  SidebarFooter,
  SidebarGroup,
  SidebarGroupContent,
  SidebarGroupLabel,
  SidebarHeader,
  SidebarMenu,
  SidebarMenuButton,
  SidebarMenuItem,
} from "@components/ui/sidebar";
import { Avatar, AvatarFallback, AvatarImage } from "@components/ui/avatar";
import { Button } from "@components/ui/button";
import { useAuth } from "@hooks/useAuth";
import { useLocation, Link } from "wouter";
import {
  LayoutDashboard,
  FileBox,
  Settings,
  Logout,
  Building2,
  Users,
} from "lucide-react";

const adminNavItems = [
  {
    title: "Dashboard",
    url: "/",
    icon: LayoutDashboard,
  },
  {
    title: "Orders",
    url: "/orders",
    icon: FileBox,
  },
  {
    title: "Clients",
    url: "/clients",
    icon: Users,
  },
  {
    title: "Settings",
    url: "/settings",
    icon: Settings,
  },
];

const clientNavItems = [
  {
    title: "My Orders",
    url: "/",
    icon: FileBox,
  },
  {
    title: "Settings",
    url: "/settings",
    icon: Settings,
  },
];

export function AppSidebar() {
  const { user, isAdmin } = useAuth();
  const [location] = useLocation();
  const navItems = isAdmin ? adminNavItems : clientNavItems;

  const getUserInitials = () => {
    if (user?.firstName && user?.lastName) {
      return `${user.firstName[0]}${user.lastName[0]}`.toUpperCase();
    }
    if (user?.email) {
      return user.email[0].toUpperCase();
    }
    return "U";
  };

  return (
    <Sidebar>
      <SidebarHeader className="border-b border-sidebar-border px-4 py-4">
        <Link href="/" className="flex items-center gap-3">
          <div className="flex h-9 w-9 items-center justify-center rounded-md bg-primary">
            <Building2 className="h-5 w-5 text-primary-foreground" />
          </div>
          <div className="flex flex-col">
            <span className="font-semibold text-sm">LOD 400</span>
            <span className="text-xs text-muted-foreground">Delivery Platform</span>
          </div>
        </Link>
      </SidebarHeader>

      <SidebarContent>
        <SidebarGroup>
          <SidebarGroupLabel>Navigation</SidebarGroupLabel>
          <SidebarGroupContent>
            <SidebarMenu>
              {navItems.map((item) => (
                <SidebarMenuItem key={item.title}>

```

```

        <SidebarMenuButton
          asChild
            isActive={location === item.url}
            data-testid={`nav-${item.title.toLowerCase().replace(/s+/g, "-")}`}
        >
          <Link href={item.url}>
            <item.icon className="h-4 w-4" />
            <span>{item.title}</span>
          </Link>
        </SidebarMenuButton>
      </SidebarMenuItem>
    )}
  </SidebarMenu>
  </SidebarGroupContent>
  </SidebarGroup>
  </SidebarContent>

  <SidebarFooter className="border-t border-sidebar-border p-4">
    <div className="flex items-center gap-3">
      <Avatar className="h-9 w-9">
        <AvatarImage
          src={user?.profileImageUrl || undefined}
          alt={user?.firstName || "User"}
          className="object-cover"
        />
        <AvatarFallback className="text-xs font-medium">
          {getUserInitials()}
        </AvatarFallback>
      </Avatar>
      <div className="flex-1 min-w-0">
        <p className="text-sm font-medium truncate">
          {user?.firstName} {user?.lastName}
        </p>
        <p className="text-xs text-muted-foreground truncate">
          {user?.email || "No email"}
        </p>
      </div>
      <Button
        variant="ghost"
        size="icon"
        asChild
        data-testid="button-logout"
      >
        <a href="/api/logout">
          <Logout className="h-4 w-4" />
        </a>
      </Button>
    </div>
  </SidebarFooter>
</Sidebar>
);
}

```

File: client/src/components/ObjectUploader.tsx

```
import { useState, useEffect, useMemo } from "react";
import type { ReactNode } from "react";
import Uppy from "@uppy/core";
import DashboardModal from "@uppy/react/dashboard-modal";
import AwsS3 from "@uppy/aws-s3";
import type { UploadResult, UppyFile } from "@uppy/core";
import { Button } from "@components/ui/button";
import "@uppy/core/css/style.min.css";
import "@uppy/dashboard/css/style.min.css";

interface ObjectUploaderProps {
  maxNumberOfFiles?: number;
  maxFileSize?: number;
  allowedFileTypes?: string[];
  getUploadUrl: (fileName: string) => Promise<string>;
  onUploadComplete?: (fileName: string, uploadUrl: string, fileSize: number) => Promise<void>;
  onAllComplete?: () => void;
  buttonClassName?: string;
  buttonVariant?: "default" | "outline" | "secondary" | "ghost" | "link" | "destructive";
  children: ReactNode;
  disabled?: boolean;
}

export function ObjectUploader({
  maxNumberOfFiles = 1,
  maxFileSize = 524288000,
  allowedFileTypes,
  getUploadUrl,
  onUploadComplete,
  onAllComplete,
  buttonClassName,
  buttonVariant = "default",
  children,
  disabled = false,
}: ObjectUploaderProps) {
  const [showModal, setShowModal] = useState(false);

  const uppy = useMemo(() => {
    const uppyInstance = new Uppy({
      restrictions: {
        maxNumberOfFiles,
        maxFileSize,
        allowedFileTypes,
      },
      autoProceed: false,
    });

    uppyInstance.use(AwsS3, {
      shouldUseMultipart: false,
      getUploadParameters: async (file: UppyFile<Record<string, unknown>;, Record<string, unknown>&&>) => {
        const url = await getUploadUrl(file.name);
        (file as any).uploadUrl = url;
        return {
          method: "PUT" as const,
          url,
          headers: {
            "Content-Type": file.type || "application/octet-stream",
          },
        };
      },
    });

    return uppyInstance;
  }, [getUploadUrl, maxNumberOfFiles, maxFileSize, allowedFileTypes]);

  useEffect(() => {
    const handleUploadSuccess = async (file: UppyFile<Record<string, unknown>;, Record<string, unknown>&&> | undefined) => {
      if (file && onUploadComplete) {
        const uploadUrl = (file as any).uploadUrl;
        await onUploadComplete(file.name, uploadUrl, file.size);
      }
    };

    const handleComplete = (result: UploadResult<Record<string, unknown>;, Record<string, unknown>&&>) => {
      if (result.successful.length > 0 && onAllComplete) {
        onAllComplete();
      }
      setShowModal(false);
      uppy.cancelAll();
    };

    uppy.on("upload-success", handleUploadSuccess);
    uppy.on("complete", handleComplete);

    return () => {
      uppy.off("upload-success", handleUploadSuccess);
      uppy.off("complete", handleComplete);
    };
  }, [uppy, onUploadComplete, onAllComplete]);

  useEffect(() => {
    return () => {
      uppy.close();
    };
  }, [uppy]);

  return (
```

```

</div>
<Button
  onClick={() => setShowModal(true)}
  className={buttonClassName}
  variant={buttonVariant}
  disabled={disabled}
  data-testid="button-open-uploader"
/>
</Button>

<DashboardModal
  uppy={uppy}
  open={showModal}
  onRequestClose={() => {
    setShowModal(false);
    uppy.cancelAll();
  }}
  proudlyDisplayPoweredByUppy={false}
  note="Upload your LOD 400 deliverables (ZIP file)"
/>
</div>
);
}

```

File: client/src/components/OrderDetailModal.tsx

```
import { useState } from "react";
import { useMutation } from "@tanstack/react-query";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
} from "@components/ui/dialog";
import { Button } from "@components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from "@components/ui/card";
import { Separator } from "@components/ui/separator";
import { StatusBadge } from "@components/StatusBadge";
import { OrderTimeline } from "@components/OrderTimeline";
import { ObjectUploader } from "@components/ObjectUploader";
import { Download, Upload, FileArchive, ExternalLink, Loader2 } from "lucide-react";
import type { OrderWithFiles } from "@shared/schema";
import { format } from "date-fns";
import { apiRequest, queryClient } from "@lib/queryClient";
import { useToast } from "@hooks/use-toast";

interface OrderDetailModalProps {
  order: OrderWithFiles | null;
  isOpen: boolean;
  onClose: () => void;
  onDownloadInputs?: () => void;
  onMarkComplete?: () => void;
  onDownloadOutputs?: () => void;
  isAdmin?: boolean;
  isMarkingComplete?: boolean;
}

function formatCurrency(amount: number): string {
  return new Intl.NumberFormat("en-SA", {
    style: "currency",
    currency: "SAR",
    minimumFractionDigits: 0,
  }).format(amount);
}

function formatDate(date: Date | string | null | undefined): string {
  if (!date) return "-";
  const d = typeof date === "string" ? new Date(date) : date;
  return format(d, "PPp");
}

function formatFileSize(bytes: number | null | undefined): string {
  if (!bytes) return "-";
  const units = ["B", "KB", "MB", "GB"];
  let size = bytes;
  let unitIndex = 0;
  while (size >= 1024 && unitIndex < units.length - 1) {
    size /= 1024;
    unitIndex++;
  }
  return `${size.toFixed(1)} ${units[unitIndex]}`;
}

export function OrderDetailModal({
  order,
  isOpen,
  onClose,
  onDownloadInputs,
  onMarkComplete,
  onDownloadOutputs,
  isAdmin = false,
  isMarkingComplete = false,
}: OrderDetailModalProps) {
  const { toast } = useToast();
  const [isUploading, setIsUploading] = useState(false);

  const getUploadUrl = async (fileName: string): Promise<string> => {
    if (!order) throw new Error("No order selected");

    const response = await apiRequest("POST", `/api/admin/orders/${order.id}/upload-url`, {
      fileName,
    });
    const data = await response.json();
    return data.uploadURL;
  };

  const handleUploadComplete = async (fileName: string, uploadUrl: string, fileSize: number) => {
    if (!order) return;

    try {
      await apiRequest("POST", `/api/admin/orders/${order.id}/upload-complete`, {
        fileName,
        fileSize,
        uploadURL: uploadUrl,
      });
    } catch (error) {
      console.error("Error completing upload:", error);
      throw error;
    }
  };

  const handleAllUploadsComplete = () => {
    setIsUploading(false);
  };
}
```

```

queryClient.invalidateQueries({ queryKey: ["/api/admin/orders"] });
toast({
  title: "Upload Complete",
  description: "Deliverables have been uploaded successfully. You can now mark the order as complete.",
});
onClose();
};

if (!order) return null;

const inputFiles = order.files?.filter((f) => f.fileType === "input") || [];
const outputFiles = order.files?.filter((f) => f.fileType === "output") || [];

return (
  <Dialog open={isOpen} onOpenChange={(open) => !open && onClose()}>
    <DialogContent className="max-w-2xl max-h-[90vh] overflow-y-auto">
      <DialogHeader>
        <DialogTitle className="flex items-center gap-3">
          <span>Order Details</span>
          <StatusBadge status={order.status} />
        </DialogTitle>
      </DialogHeader>

      <div className="space-y-6">
        <Card>
          <CardHeader className="pb-3">
            <CardTitle className="text-base">Order Summary</CardTitle>
          </CardHeader>
          <CardContent className="grid grid-cols-2 gap-4 text-sm">
            <div>
              <p className="text-muted-foreground">Order ID</p>
              <p className="font-mono text-xs">{order.id}</p>
            </div>
            <div>
              <p className="text-muted-foreground">Created</p>
              <p>{formatDate(order.createdAt)}</p>
            </div>
            <div>
              <p className="text-muted-foreground">Sheet Count</p>
              <p className="font-medium">{order.sheetCount} sheets</p>
            </div>
            <div>
              <p className="text-muted-foreground">Total Price</p>
              <p className="font-medium text-primary">{formatCurrency(order.totalPriceSar)}</p>
            </div>
            <div>
              {isAdmin && order.user && (
                <div>
                  <p className="text-muted-foreground">Client Name</p>
                  <p>{order.user.firstName} {order.user.lastName}</p>
                </div>
                <div>
                  <p className="text-muted-foreground">Client Email</p>
                  <p>{order.user.email || "-"}</p>
                </div>
              )}
            </div>
          </CardContent>
        </Card>

        <div className="grid md:grid-cols-2 gap-6">
          <Card>
            <CardHeader className="pb-3">
              <CardTitle className="text-base">Order Timeline</CardTitle>
            </CardHeader>
            <CardContent>
              <OrderTimeline
                status={order.status}
                createdAt={order.createdAt}
                paidAt={order.paidAt}
                uploadedAt={order.uploadedAt}
                completedAt={order.completedAt}
              />
            </CardContent>
          </Card>

          <div className="space-y-4">
            <Card>
              <CardHeader className="pb-3">
                <CardTitle className="text-base flex items-center gap-2">
                  <Upload className="h-4 w-4" />
                  Input Files
                </CardTitle>
              </CardHeader>
              <CardContent>
                {inputFiles.length === 0 ? (
                  <p className="text-sm text-muted-foreground">No input files uploaded yet.</p>
                ) : (
                  <div className="space-y-2">

```

```

        {inputFiles.map((file) => {
            <div
                key={file.id}
                className="flex items-center gap-3 p-2 rounded-md bg-muted/50"
            >
                <FileArchive className="h-4 w-4 text-muted-foreground shrink-0" />
                <div className="flex-1 min-w-0">
                    <p className="text-sm font-medium truncate">
                        {file.fileName}
                    </p>
                    <p className="text-xs text-muted-foreground">
                        {formatFileSize(file.fileSize)}
                    </p>
                </div>
            </div>
        ))}
        {isAdmin && onDownloadInputs && (
            <Button
                variant="outline"
                size="sm"
                className="w-full mt-2"
                onClick={onDownloadInputs}
                data-testid="button-download-inputs"
            >
                <Download className="h-4 w-4 mr-2" />
                Download All Inputs
            </Button>
        )}
    </div>
)}
</CardContent>
</Card>

<Card>
    <CardHeader className="pb-3">
        <CardTitle className="text-base flex items-center gap-2">
            <Download className="h-4 w-4" />
            Output Files
        </CardTitle>
    </CardHeader>
    <CardContent>
        {outputFiles.length === 0 ? (
            <p className="text-sm text-muted-foreground">
                No output files available yet.
            </p>
        ) : (
            <div className="space-y-2">
                {outputFiles.map((file) => {
                    <div
                        key={file.id}
                        className="flex items-center gap-3 p-2 rounded-md bg-muted/50"
                    >
                        <FileArchive className="h-4 w-4 text-muted-foreground shrink-0" />
                        <div className="flex-1 min-w-0">
                            <p className="text-sm font-medium truncate">
                                {file.fileName}
                            </p>
                            <p className="text-xs text-muted-foreground">
                                {formatFileSize(file.fileSize)}
                            </p>
                        </div>
                    </div>
                ))}
                {onDownloadOutputs && order.status === "complete" && (
                    <Button
                        variant="outline"
                        size="sm"
                        className="w-full mt-2"
                        onClick={onDownloadOutputs}
                        data-testid="button-download-outputs"
                    >
                        <Download className="h-4 w-4 mr-2" />
                        Download Deliverables
                    </Button>
                )}
            </div>
        )}
    </CardContent>
</Card>
</div>

{isAdmin && (
    <Separator />
    <div className="flex flex-wrap gap-3 justify-end">
        {(order.status === "uploaded" || order.status === "processing") && (
            <ObjectUploader
                getUploadUrl={getUploadUrl}
                onUploadComplete={handleUploadComplete}
                onAllComplete={handleAllUploadsComplete}
                allowedFileTypes={[".zip", ".rvt", ".rfa"]}
                buttonVariant="outline"
                disabled={isUploading}
            >
                <Upload className="h-4 w-4 mr-2" />
                {isUploading ? "Uploading..." : "Upload Deliverables"}
            </ObjectUploader>
        )}
        {order.status === "processing" && outputFiles.length > 0 && onMarkComplete && (
            <Button

```

```

        onClick={onMarkComplete}
        disabled={isMarkingComplete}
        data-testid="button-mark-complete"
      <button>
        {isMarkingComplete ? (
          <div>
            <Loader2 className="h-4 w-4 mr-2 animate-spin" />
            Completing...
          </div>
        ) : (
          "Mark Complete & Notify Client"
        )}
      </button>
    </div>
  </div>
)

{!isAdmin & order.status === "pending" & (
  <div>
    <Separator />
    <div className="flex justify-end">
      <button asChild data-testid="button-continue-payment">
        <a href={`\api/orders/${order.id}/checkout`} target="_blank">
          <ExternalLink className="h-4 w-4 mr-2" />
          Continue to Payment
        </a>
      </button>
    </div>
  </div>
)
</div>
</DialogContent>
</Dialog>
);
}

```

File: client/src/components/OrdersTable.tsx

```
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@components/ui/table";
import { Button } from "@components/ui/button";
import { StatusBadge } from "@components/StatusBadge";
import { Eye, Download, Upload, CheckCircle } from "lucide-react";
import type { OrderWithFiles } from "@shared/schema";
import { formatDistanceToNow } from "date-fns";
import { Skeleton } from "@components/ui/skeleton";

interface OrdersTableProps {
  orders: OrderWithFiles[];
  isLoading?: boolean;
  onViewOrder: (order: OrderWithFiles) => void;
  onDownloadInputs?: (order: OrderWithFiles) => void;
  onUploadOutputs?: (order: OrderWithFiles) => void;
  onMarkComplete?: (order: OrderWithFiles) => void;
  isAdmin?: boolean;
}

function formatCurrency(amount: number): string {
  return new Intl.NumberFormat("en-SA", {
    style: "currency",
    currency: "SAR",
    minimumFractionDigits: 0,
  }).format(amount);
}

function formatDate(date: Date | string | null | undefined): string {
  if (!date) return "-";
  const d = typeof date === "string" ? new Date(date) : date;
  return formatDistanceToNow(d, { addSuffix: true });
}

export function OrdersTable({
  orders,
  isLoading,
  onViewOrder,
  onDownloadInputs,
  onUploadOutputs,
  onMarkComplete,
  isAdmin = false,
}: OrdersTableProps) {
  if (isLoading) {
    return (
      <div className="space-y-3">
        {[...Array(5)].map((_, i) => (
          <Skeleton key={i} className="h-16 w-full" />
        ))}
      </div>
    );
  }

  if (orders.length === 0) {
    return (
      <div className="flex flex-col items-center justify-center py-12 text-center">
        <div className="rounded-full bg-muted p-4 mb-4">
          <Eye className="h-8 w-8 text-muted-foreground" />
        </div>
        <h3 className="text-lg font-semibold">No orders yet</h3>
        <p className="text-sm text-muted-foreground mt-1">
          {isAdmin
            ? "Orders will appear here when clients create them."
            : "Create your first order from the Revit add-in."}
        </p>
      </div>
    );
  }

  const hasInputFiles = (order: OrderWithFiles) =>
    order.files?.some((f) => f.fileType === "input");

  const hasOutputFiles = (order: OrderWithFiles) =>
    order.files?.some((f) => f.fileType === "output");

  return (
    <div className="rounded-md border overflow-x-auto">
      <Table>
        <TableHeader>
          <TableRow>
            <TableHead className="w-[120px]">Order ID</TableHead>
            {isAdmin & & <TableHead>Client</TableHead>}
            <TableHead className="text-center">Sheets</TableHead>
            <TableHead className="text-right">Price</TableHead>
            <TableHead>Status</TableHead>
            <TableHead>Created</TableHead>
            <TableHead className="text-right">Actions</TableHead>
          </TableRow>
          <TableBody>
            {orders.map((order) => (
              <TableRow
                key={order.id}
                className="hover-elevate cursor-pointer"
                onClick={() => onViewOrder(order)}
                data-testid={`row-order-${order.id}`}
              >
                <TableCell className="font-mono text-sm">
                  {order.id.slice(0, 8)}...
```

```

</TableCell>
{isAdmin &&& (
  <TableCell>
    <div className="flex flex-col">
      <span className="font-medium text-sm">
        {order.user?.firstName} {order.user?.lastName}
      </span>
      <span className="text-xs text-muted-foreground">
        {order.user?.email || "No email"}
      </span>
    </div>
  </TableCell>
)}
<TableCell className="text-center font-medium">
  {order.sheetCount}
</TableCell>
<TableCell className="text-right font-medium">
  {formatCurrency(order.totalPriceSar)}
</TableCell>
<TableCell>
  <StatusBadge status={order.status} />
</TableCell>
<TableCell className="text-muted-foreground text-sm">
  {formatDate(order.createdAt)}
</TableCell>
<TableCell className="text-right">
  <div className="flex items-center justify-end gap-1">
    <Button
      variant="ghost"
      size="icon"
      onClick={(e) => {
        e.stopPropagation();
        onViewOrder(order);
      }}
      data-testid={`button-view-order-${order.id}`}
    >
    <Eye className="h-4 w-4" />
  </Button>

    {isAdmin &&& hasInputFiles(order) &&& onDownloadInputs &&& (
      <Button
        variant="ghost"
        size="icon"
        onClick={(e) => {
          e.stopPropagation();
          onDownloadInputs(order);
        }}
        data-testid={`button-download-inputs-${order.id}`}
      >
      <Download className="h-4 w-4" />
    </Button>
    )}

    {isAdmin &&& order.status === "uploaded" &&& onUploadOutputs &&& (
      <Button
        variant="ghost"
        size="icon"
        onClick={(e) => {
          e.stopPropagation();
          onUploadOutputs(order);
        }}
        data-testid={`button-upload-outputs-${order.id}`}
      >
      <Upload className="h-4 w-4" />
    </Button>
    )}

    {isAdmin &&& order.status === "processing" &&& hasOutputFiles(order) &&& onMarkComplete &&& (
      <Button
        variant="ghost"
        size="icon"
        onClick={(e) => {
          e.stopPropagation();
          onMarkComplete(order);
        }}
        data-testid={`button-mark-complete-${order.id}`}
      >
      <CheckCircle className="h-4 w-4" />
    </Button>
    )}
  </div>
</TableCell>
</TableRow>
))
</TableBody>
</Table>
</div>
);
}

```

File: client/src/components/OrderTimeline.tsx

```
import { cn } from "@/lib/utils";
import { Check, Clock, CreditCard, Upload, Cog, CheckCircle } from "lucide-react";

type OrderStatus = "pending" | "paid" | "uploaded" | "processing" | "complete";

interface OrderTimelineProps {
  status: OrderStatus;
  createdAt?: Date | string | null;
  paidAt?: Date | string | null;
  uploadedAt?: Date | string | null;
  completedAt?: Date | string | null;
}

const steps = [
  { status: "pending", label: "Order Created", icon: Clock },
  { status: "paid", label: "Payment Received", icon: CreditCard },
  { status: "uploaded", label: "Files Uploaded", icon: Upload },
  { status: "processing", label: "Processing", icon: Cog },
  { status: "complete", label: "Complete", icon: CheckCircle },
] as const;

const statusOrder: OrderStatus[] = ["pending", "paid", "uploaded", "processing", "complete"];

function formatDate(date: Date | string | null | undefined): string {
  if (!date) return "";
  const d = typeof date === "string" ? new Date(date) : date;
  return d.toLocaleDateString("en-US", {
    month: "short",
    day: "numeric",
    hour: "2-digit",
    minute: "2-digit",
  });
}

export function OrderTimeline({
  status,
  createdAt,
  paidAt,
  uploadedAt,
  completedAt,
}: OrderTimelineProps) {
  const currentIndex = statusOrder.indexOf(status);

  const dates: Record<string, Date | string | null | undefined> = {
    pending: createdAt,
    paid: paidAt,
    uploaded: uploadedAt,
    processing: status === "processing" || status === "complete" ? uploadedAt : null,
    complete: completedAt,
  };

  return (
    <div className="space-y-4" data-testid="order-timeline">
      {steps.map((step, index) => {
        const stepIndex = statusOrder.indexOf(step.status);
        const isCompleted = stepIndex < currentIndex;
        const isCurrent = stepIndex === currentIndex;
        const isPending = stepIndex > currentIndex;
        const Icon = step.icon;
        const date = dates[step.status];

        return (
          <div key={step.status} className="flex gap-4">
            <div className="flex flex-col items-center">
              <div
                className={cn(
                  "flex h-8 w-8 items-center justify-center rounded-full border-2 transition-colors",
                  isCompleted
                    ? "border-green-500 bg-green-500 text-white"
                    : isCurrent
                    ? "border-primary bg-primary text-primary-foreground"
                    : "border-muted bg-muted text-muted-foreground"
                )}
              >
                {isCompleted ? (
                  <Check className="h-4 w-4" />
                ) : (
                  <Icon className="h-4 w-4" />
                )}
              </div>
            </div>
            <div>
              {index < steps.length - 1 & & (
                <div
                  className={cn(
                    "w-0.5 flex-1 min-h-6",
                    isCompleted ? "bg-green-500" : "bg-muted"
                  )}
                />
              )}
            </div>
            <div className="flex-1 pb-4">
              <p
                className={cn(
                  "font-medium text-sm",
                  isPending ? "text-muted-foreground" : "text-foreground"
                )}
              >
                {step.label}
              </p>
            </div>
          </div>
        );
      })}
    </div>
  );
}
```

```
        </p>
        {date && !isPending && (
          <p className="text-xs text-muted-foreground mt-0.5">
            {formatDate(date)}
          </p>
        )}
      </div>
    );
  }
  </div>
);
}
```

File: client/src/components/StatsCard.tsx

```
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { cn } from "@/lib/utils";
import { LucideIcon } from "lucide-react";

interface StatsCardProps {
  title: string;
  value: string | number;
  description?: string;
  icon: LucideIcon;
  trend?: {
    value: number;
    isPositive: boolean;
  };
  className?: string;
}

export function StatsCard({
  title,
  value,
  description,
  icon: Icon,
  trend,
  className,
}: StatsCardProps) {
  return (
    <Card className={cn("hover-elevate", className)}>
      <CardHeader className="flex flex-row items-center justify-between gap-2 space-y-0 pb-2">
        <CardTitle className="text-sm font-medium text-muted-foreground">
          {title}
        </CardTitle>
        <Icon className="h-4 w-4 text-muted-foreground" />
      </CardHeader>
      <CardContent>
        <div className="text-2xl font-bold">{value}</div>
        {description & & (
          <p className="text-xs text-muted-foreground mt-1">{description}</p>
        )}
        {trend & & (
          <p
            className={cn(
              "text-xs mt-1 font-medium",
              trend.isPositive ? "text-green-600 dark:text-green-400" : "text-red-600 dark:text-red-400"
            )}
          >
            {trend.isPositive ? "+" : "-"}
            {Math.abs(trend.value)}% from last month
          </p>
        )}
      </CardContent>
    </Card>
  );
}
```

File: client/src/components/StatusBadge.tsx

```
import { Badge } from "@components/ui/badge";
import { cn } from "@lib/utils";
import { Clock, CreditCard, Upload, Cog, CheckCircle } from "lucide-react";

type OrderStatus = "pending" | "paid" | "uploaded" | "processing" | "complete";

interface StatusBadgeProps {
  status: OrderStatus;
  className?: string;
}

const statusConfig: Record<
  OrderStatus,
  { label: string; className: string; icon: typeof Clock }
> = {
  pending: {
    label: "Pending Payment",
    className: "bg-amber-100 text-amber-800 dark:bg-amber-900/30 dark:text-amber-400 border-amber-200 dark:border-amber-800",
    icon: Clock,
  },
  paid: {
    label: "Paid",
    className: "bg-blue-100 text-blue-800 dark:bg-blue-900/30 dark:text-blue-400 border-blue-200 dark:border-blue-800",
    icon: CreditCard,
  },
  uploaded: {
    label: "Files Uploaded",
    className: "bg-purple-100 text-purple-800 dark:bg-purple-900/30 dark:text-purple-400 border-purple-200 dark:border-purple-800",
    icon: Upload,
  },
  processing: {
    label: "Processing",
    className: "bg-orange-100 text-orange-800 dark:bg-orange-900/30 dark:text-orange-400 border-orange-200 dark:border-orange-800",
    icon: Cog,
  },
  complete: {
    label: "Complete",
    className: "bg-green-100 text-green-800 dark:bg-green-900/30 dark:text-green-400 border-green-200 dark:border-green-800",
    icon: CheckCircle,
  },
};

export function StatusBadge({ status, className }: StatusBadgeProps) {
  const config = statusConfig[status];
  const Icon = config.icon;

  return (
    <Badge
      variant="outline"
      className={cn(
        "font-medium gap-1.5 no-default-hover-elevate",
        config.className,
        className
      )}
      data-testid={`badge-status-${status}`}
    >
      <Icon className="h-3 w-3" />
      {config.label}
    </Badge>
  );
}
```

File: client/src/components/ThemeProvider.tsx

```
import { createContext, useContext, useEffect, useState } from "react";

type Theme = "dark" | "light";

type ThemeProviderContextType = {
  theme: Theme;
  setTheme: (theme: Theme) => void;
  toggleTheme: () => void;
};

const ThemeProviderContext = createContext<ThemeProviderContextType | undefined>(undefined);

interface ThemeProviderProps {
  children: React.ReactNode;
  defaultTheme?: Theme;
  storageKey?: string;
}

export function ThemeProvider({
  children,
  defaultTheme = "light",
  storageKey = "lod400-theme",
}: ThemeProviderProps) {
  const [theme, setTheme] = useState<Theme>(() => {
    if (typeof window !== "undefined") {
      const stored = localStorage.getItem(storageKey) as Theme | null;
      return stored || defaultTheme;
    }
    return defaultTheme;
  });

  useEffect(() => {
    const root = window.document.documentElement;
    root.classList.remove("light", "dark");
    root.classList.add(theme);
    localStorage.setItem(storageKey, theme);
  }, [theme, storageKey]);

  const toggleTheme = () => {
    setTheme(theme === "light" ? "dark" : "light");
  };

  return (
    <ThemeProviderContext.Provider value={{ theme, setTheme, toggleTheme }}>
      {children}
    </ThemeProviderContext.Provider>
  );
}

export function useTheme() {
  const context = useContext(ThemeProviderContext);
  if (context === undefined) {
    throw new Error("useTheme must be used within a ThemeProvider");
  }
  return context;
}
```

File: client/src/components/ThemeToggle.tsx

```
import { Moon, Sun } from "lucide-react";
import { Button } from "@/components/ui/button";
import { useTheme } from "@/components/ThemeProvider";

export function ThemeToggle() {
  const { theme, toggleTheme } = useTheme();

  return (
    <Button
      variant="ghost"
      size="icon"
      onClick={toggleTheme}
      data-testid="button-theme-toggle"
    >
      {theme === "light" ? (
        <Moon className="h-5 w-5" />
      ) : (
        <Sun className="h-5 w-5" />
      )}
      <span className="sr-only">Toggle theme</span>
    </Button>
  );
}
```

File: client/src/components/ui/accordion.tsx

```
import * as React from "react"
import * as AccordionPrimitive from "@radix-ui/react-accordion"
import { ChevronDown } from "lucide-react"

import { cn } from "@/lib/utlils"

const Accordion = AccordionPrimitive.Root

const AccordionItem = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Item>;
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Item>;
>(({ className, ...props }, ref) => {
  <AccordionPrimitive.Item
    ref={ref}
    className={cn("border-b", className)}
    {...props}
  />
))
AccordionItem.displayName = "AccordionItem"

const AccordionTrigger = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Trigger>;
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Trigger>;
>(({ className, children, ...props }, ref) => {
  <AccordionPrimitive.Header className="flex">
    <AccordionPrimitive.Trigger
      ref={ref}
      className={cn(
        "flex flex-1 items-center justify-between py-4 font-medium transition-all hover:underline [&[data-state=open]&svg]:rotate-180",
        className
      )}
      {...props}
    >
      {children}
      <ChevronDown className="h-4 w-4 shrink-0 transition-transform duration-200" />
    </AccordionPrimitive.Trigger>
  </AccordionPrimitive.Header>
))
AccordionTrigger.displayName = AccordionPrimitive.Trigger.displayName

const AccordionContent = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Content>;
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Content>;
>(({ className, children, ...props }, ref) => {
  <AccordionPrimitive.Content
    ref={ref}
    className="overflow-hidden text-sm transition-all data-[state=closed]:animate-accordion-up data-[state=open]:animate-accordion-down"
    {...props}
  >
    <div className={cn("pb-4 pt-0", className)}>{children}</div>
  </AccordionPrimitive.Content>
))
AccordionContent.displayName = AccordionPrimitive.Content.displayName

export { Accordion, AccordionItem, AccordionTrigger, AccordionContent }
```

File: client/src/components/ui/alert-dialog.tsx

```
import * as React from "react"
import * as AlertDialogPrimitive from "@radix-ui/react-alert-dialog"

import { cn } from "@/lib/utlils"
import { buttonVariants } from "@/components/ui/button"

const AlertDialog = AlertDialogPrimitive.Root

const AlertDialogTrigger = AlertDialogPrimitive.Trigger

const AlertDialogPortal = AlertDialogPrimitive.Portal

const AlertDialogOverlay = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Overlay>;,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Overlay>;
>(({ className, ...props }, ref) => {
  <AlertDialogPrimitive.Overlay
    className={cn(
      "fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0"
    )}
    {...props}
    ref={ref}
  />
})
AlertDialogOverlay.displayName = AlertDialogPrimitive.Overlay.displayName

const AlertDialogContent = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Content>;,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Content>;
>(({ className, ...props }, ref) => {
  <AlertDialogPortal>
    <AlertDialogOverlay />
    <AlertDialogPrimitive.Content
      ref={ref}
      className={cn(
        "fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0"
      )}
      {...props}
    />
  </AlertDialogPortal>
</>
})
AlertDialogContent.displayName = AlertDialogPrimitive.Content.displayName

const AlertDialogHeader = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => {
  <div
    className={cn(
      "flex flex-col space-y-2 text-center sm:text-left",
      className
    )}
    {...props}
  />
}
AlertDialogHeader.displayName = "AlertDialogHeader"

const AlertDialogFooter = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => {
  <div
    className={cn(
      "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
      className
    )}
    {...props}
  />
}
AlertDialogFooter.displayName = "AlertDialogFooter"

const AlertDialogTitle = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Title>;,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Title>;
>(({ className, ...props }, ref) => {
  <AlertDialogPrimitive.Title
    ref={ref}
    className={cn("text-lg font-semibold", className)}
    {...props}
  />
})
AlertDialogTitle.displayName = AlertDialogPrimitive.Title.displayName

const AlertDialogDescription = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Description>;,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Description>;
>(({ className, ...props }, ref) => {
  <AlertDialogPrimitive.Description
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
})
AlertDialogDescription.displayName =
  AlertDialogPrimitive.Description.displayName
```

```

const AlertDialogAction = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Action>;
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Action>;
  &gt;(({ className, ...props }, ref) => {
    <AlertDialogPrimitive.Action
      ref={ref}
      className={cn(buttonVariants(), className)}
      {...props}
    />
  ))
AlertDialogAction.displayName = AlertDialogPrimitive.Action.displayName

const AlertDialogCancel = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Cancel>;
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Cancel>;
  &gt;(({ className, ...props }, ref) => {
    <AlertDialogPrimitive.Cancel
      ref={ref}
      className={cn(
        buttonVariants({ variant: "outline" }),
        "mt-2 sm:mt-0",
        className
      )}
      {...props}
    />
  ))
AlertDialogCancel.displayName = AlertDialogPrimitive.Cancel.displayName

export {
  AlertDialog,
  AlertDialogPortal,
  AlertDialogOverlay,
  AlertDialogTrigger,
  AlertDialogContent,
  AlertDialogHeader,
  AlertDialogFooter,
  AlertDialogTitle,
  AlertDialogDescription,
  AlertDialogAction,
  AlertDialogCancel,
}

```

File: client/src/components/ui/alert.tsx

```
import * as React from "react"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utls"

const alertVariants = cva(
  "relative w-full rounded-lg border p-4 [&#x26;#x26;svg~*]:p1-7 [&#x26;#x26;svg+div]:translate-y-[-3px] [&#x26;#x26;svg]:absolute [&#x26;#x26;svg]:left-4 [&#x26;#x26;svg]:",
  {
    variants: {
      variant: {
        default: "bg-background text-foreground",
        destructive:
          "border-destructive/50 text-destructive dark:border-destructive [&#x26;#x26;svg]:text-destructive",
      },
    },
    defaultVariants: {
      variant: "default",
    },
  },
)

const Alert = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement> &#x26;#x26; VariantProps<typeof alertVariants>
>(({ className, variant, ...props }, ref) => {
  &#x26;#x26;div
    ref={ref}
    role="alert"
    className={cn(alertVariants({ variant }), className)}
    {...props}
  /&#x26;#x26;
})
Alert.displayName = "Alert"

const AlertTitle = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLHeadingElement>
>(({ className, ...props }, ref) => {
  &#x26;#x26;h5
    ref={ref}
    className={cn("mb-1 font-medium leading-none tracking-tight", className)}
    {...props}
  /&#x26;#x26;
})
AlertTitle.displayName = "AlertTitle"

const AlertDescription = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>(({ className, ...props }, ref) => {
  &#x26;#x26;div
    ref={ref}
    className={cn("text-sm [&#x26;#x26;p]:leading-relaxed", className)}
    {...props}
  /&#x26;#x26;
})
AlertDescription.displayName = "AlertDescription"

export { Alert, AlertTitle, AlertDescription }
```

File: client/src/components/ui/aspect-ratio.tsx

```
import * as AspectRatioPrimitive from "@radix-ui/react-aspect-ratio"

const AspectRatio = AspectRatioPrimitive.Root

export { AspectRatio }
```

File: client/src/components/ui/avatar.tsx

```
"use client"

import * as React from "react"
import * as AvatarPrimitive from "@radix-ui/react-avatar"

import { cn } from "@/lib/utils"

const Avatar = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Root>;
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>;
>(({ className, ...props }, ref) => {
  <AvatarPrimitive.Root
    ref={ref}
    className={cn(
      "after:content-[''] after:block after:absolute after:inset-0 after:rounded-full after:pointer-events-none after:border after:border-black/10 dark:after:b",
      relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full`,
      className
    )}
    {...props}
  />
})
Avatar.displayName = AvatarPrimitive.Root.displayName

const AvatarImage = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Image>;
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>;
>(({ className, ...props }, ref) => {
  <AvatarPrimitive.Image
    ref={ref}
    className={cn("aspect-square h-full w-full", className)}
    {...props}
  />
})
AvatarImage.displayName = AvatarPrimitive.Image.displayName

const AvatarFallback = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Fallback>;
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>;
>(({ className, ...props }, ref) => {
  <AvatarPrimitive.Fallback
    ref={ref}
    className={cn(
      "flex h-full w-full items-center justify-center rounded-full bg-muted",
      className
    )}
    {...props}
  />
})
AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName

export { Avatar, AvatarImage, AvatarFallback }
```

File: client/src/components/ui/badge.tsx

```
import * as React from "react"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const badgeVariants = cva(
  // Whitespace-nowrap: Badges should never wrap.
  "whitespace-nowrap inline-flex items-center rounded-md border px-2.5 py-0.5 text-xs font-semibold transition-colors focus:outline-none focus:ring-2 focus:ring-primary",
  {
    variants: {
      variant: {
        default: "border-transparent bg-primary text-primary-foreground shadow-xs",
        secondary: "border-transparent bg-secondary text-secondary-foreground",
        destructive: "border-transparent bg-destructive text-destructive-foreground shadow-xs",
        outline: "border [border-color:var(--badge-outline)] shadow-xs",
      },
    },
    defaultVariants: {
      variant: "default",
    },
  },
)

export interface BadgeProps
  extends React.HTMLAttributes<HTMLDivElement>,
    VariantProps<typeof badgeVariants> {}

function Badge({ className, variant, ...props }: BadgeProps) {
  return (
    <div className={cn(badgeVariants({ variant }), className)} {...props} />
  );
}

export { Badge, badgeVariants }
```

File: client/src/components/ui/breadcrumb.tsx

```
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { ChevronRight, MoreHorizontal } from "lucide-react"

import { cn } from "@lib/utlils"

const Breadcrumb = React.forwardRef<
  HTMLElement,
  React.ComponentPropsWithoutRef< "nav"> & {
    separator?: React.ReactNode
  }
>(({ ...props }, ref) => {
  <nav ref={ref} aria-label="breadcrumb" {...props} />
  Breadcrumb.displayName = "Breadcrumb"

  const BreadcrumbList = React.forwardRef<
    HTMLListElement,
    React.ComponentPropsWithoutRef< "ol">
  >(({ className, ...props }, ref) => {
    <ol
      ref={ref}
      className={cn(
        "flex flex-wrap items-center gap-1.5 break-words text-sm text-muted-foreground sm:gap-2.5",
        className
      )}
      {...props}
    />
  ))
  BreadcrumbList.displayName = "BreadcrumbList"

  const BreadcrumbItem = React.forwardRef<
    HTMLLIElement,
    React.ComponentPropsWithoutRef< "li">
  >(({ className, ...props }, ref) => {
    <li
      ref={ref}
      className={cn("inline-flex items-center gap-1.5", className)}
      {...props}
    />
  ))
  BreadcrumbItem.displayName = "BreadcrumbItem"

  const BreadcrumbLink = React.forwardRef<
    HTMLAnchorElement,
    React.ComponentPropsWithoutRef< "a"> & {
      asChild?: boolean
    }
  >(({ asChild, className, ...props }, ref) => {
    const Comp = asChild ? Slot : "a"

    return (
      <Comp
        ref={ref}
        className={cn("transition-colors hover:text-foreground", className)}
        {...props}
      />
    )
  })
  BreadcrumbLink.displayName = "BreadcrumbLink"

  const BreadcrumbPage = React.forwardRef<
    HTMLSpanElement,
    React.ComponentPropsWithoutRef< "span">
  >(({ className, ...props }, ref) => {
    <span
      ref={ref}
      role="link"
      aria-disabled="true"
      aria-current="page"
      className={cn("font-normal text-foreground", className)}
      {...props}
    />
  ))
  BreadcrumbPage.displayName = "BreadcrumbPage"

  const BreadcrumbSeparator = ({
    children,
    className,
    ...props
  }): React.ComponentProps< "li"> => {
    <li
      role="presentation"
      aria-hidden="true"
      className={cn("[&svg]:w-3.5 [&svg]:h-3.5", className)}
      {...props}
    >
    {children ?? <ChevronRight />}
    </li>
  }
  BreadcrumbSeparator.displayName = "BreadcrumbSeparator"

  const BreadcrumbEllipsis = ({
    className,
    ...props
  }): React.ComponentProps< "span"> => {
    <span
      role="presentation"
      aria-hidden="true"

```

```
      className={cn("flex h-9 w-9 items-center justify-center", className)}
      {...props}
    &gt;
      &lt;MoreHorizontal className="h-4 w-4" /&gt;
      &lt;span className="sr-only"&gt;More&lt;/span&gt;
    &lt;/span&gt;
  )
  BreadcrumbEllipsis.displayName = "BreadcrumbElipssis"

  export {
    Breadcrumb,
    BreadcrumbList,
    BreadcrumbItem,
    BreadcrumbLink,
    BreadcrumbPage,
    BreadcrumbSeparator,
    BreadcrumbEllipsis,
  }
```

File: client/src/components/ui/button.tsx

```
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@lib/utlils"

const buttonVariants = cva(
  "inline-flex items-center justify-center gap-2 whitespace-nowrap rounded-md text-sm font-medium focus-visible:outline-none focus-visible:ring-1 focus-visible:ring-ring hover-elevate active-elevate-2",
  {
    variants: {
      variant: {
        default:
          "bg-primary text-primary-foreground border border-primary-border",
        destructive:
          "bg-destructive text-destructive-foreground border border-destructive-border",
        outline:
          // Shows the background color of whatever card / sidebar / accent background it is inside of.
          // Inherits the current text color.
          "border [border-color:var(--button-outline)] shadow-xs active:shadow-none ",
        secondary: "border bg-secondary text-secondary-foreground border border-secondary-border ",
        // Add a transparent border so that when someone toggles a border on later, it doesn't shift layout/size.
        ghost: "border border-transparent",
      },
      size: {
        default: "min-h-9 px-4 py-2",
        sm: "min-h-8 rounded-md px-3 text-xs",
        lg: "min-h-10 rounded-md px-8",
        icon: "h-9 w-9",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>((
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button"
    return (
      <Comp
        className={cn(buttonVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  },
)

Button.displayName = "Button"

export { Button, buttonVariants }
```

File: client/src/components/ui/calendar.tsx

```
import * as React from "react"
import { ChevronLeft, ChevronRight } from "lucide-react"
import { DayPicker } from "react-day-picker"

import { cn } from "@/lib/utils"
import { buttonVariants } from "@/components/ui/button"

export type CalendarProps = React.ComponentProps<typeof DayPicker>

function Calendar({
  className,
  classNames,
  showOutsideDays = true,
  ...props
}: CalendarProps) {
  return (
    <DayPicker
      showOutsideDays={showOutsideDays}
      className={cn("p-3", className)}
      classNames={{
        months: "flex flex-col sm:flex-row space-y-4 sm:space-x-4 sm:space-y-0",
        month: "space-y-4",
        caption: "flex justify-center pt-1 relative items-center",
        caption_label: "text-sm font-medium",
        nav: "space-x-1 flex items-center",
        nav_button: cn(
          buttonVariants({ variant: "outline" }),
          "h-7 w-7 bg-transparent p-0 opacity-50 hover:opacity-100"
        ),
        nav_button_previous: "absolute left-1",
        nav_button_next: "absolute right-1",
        table: "w-full border-collapse space-y-1",
        head_row: "flex",
        head_cell:
          "text-muted-foreground rounded-md w-9 font-normal text-[0.8rem]",
        row: "flex w-full mt-2",
        cell: "h-9 w-9 text-center text-sm p-0 relative [&:has([aria-selected].day-range-end)]:rounded-r-md [&:has([aria-selected].day-outside)]:bg-accent",
        day: cn(
          buttonVariants({ variant: "ghost" }),
          "h-9 w-9 p-0 font-normal aria-selected:opacity-100"
        ),
        day_range_end: "day-range-end",
        day_selected:
          "bg-primary text-primary-foreground hover:bg-primary hover:text-primary-foreground focus:bg-primary focus:text-primary-foreground",
        day_today: "bg-accent text-accent-foreground",
        day_outside:
          "day-outside text-muted-foreground aria-selected:bg-accent/50 aria-selected:text-muted-foreground",
        day_disabled: "text-muted-foreground opacity-50",
        day_range_middle:
          "aria-selected:bg-accent aria-selected:text-accent-foreground",
        day_hidden: "invisible",
        ...classNames,
      }}
      components={{
        IconLeft: ({ className, ...props }) => (
          <ChevronLeft className={cn("h-4 w-4", className)} {...props} />
        ),
        IconRight: ({ className, ...props }) => (
          <ChevronRight className={cn("h-4 w-4", className)} {...props} />
        ),
      }}
      {...props}
    />
  )
}

Calendar.displayName = "Calendar"

export { Calendar }
```

File: client/src/components/ui/card.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div
    ref={ref}
    className={cn(
      "shadcn-card rounded-xl border bg-card border-card-border text-card-foreground shadow-sm",
      className
    )}
    {...props}
  />
));
Card.displayName = "Card"

const CardHeader = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div
    ref={ref}
    className={cn("flex flex-col space-y-1.5 p-6", className)}
    {...props}
  />
});
CardHeader.displayName = "CardHeader"

const CardTitle = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div
    ref={ref}
    className={cn(
      "text-2xl font-semibold leading-none tracking-tight",
      className
    )}
    {...props}
  />
});
CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
});
CardDescription.displayName = "CardDescription"

const CardContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div ref={ref} className={cn("p-6 pt-0", className)} {...props} />
});
CardContent.displayName = "CardContent"

const CardFooter = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  <div
    ref={ref}
    className={cn("flex items-center p-6 pt-0", className)}
    {...props}
  />
});
CardFooter.displayName = "CardFooter"

export {
  Card,
  CardHeader,
  CardFooter,
  CardTitle,
  CardDescription,
  CardContent,
}
```

File: client/src/components/ui/carousel.tsx

```
import * as React from "react"
import useEmblaCarousel, {
  type UseEmblaCarouselType,
} from "embla-carousel-react"
import { ArrowLeft, ArrowRight } from "lucide-react"

import { cn } from "@/lib/utils"
import { Button } from "@/components/ui/button"

type CarouselApi = UseEmblaCarouselType[1]
type UseCarouselParameters = Parameters<typeof useEmblaCarousel>
type CarouselOptions = UseCarouselParameters[0]
type CarouselPlugin = UseCarouselParameters[1]

type CarouselProps = {
  opts?: CarouselOptions
  plugins?: CarouselPlugin
  orientation?: "horizontal" | "vertical"
  setApi?: (api: CarouselApi) => void
}

type CarouselContextProps = {
  carouselRef: ReturnType<typeof useEmblaCarousel>[0]
  api: ReturnType<typeof useEmblaCarousel>[1]
  scrollPrev: () => void
  scrollNext: () => void
  canScrollPrev: boolean
  canScrollNext: boolean
} & CarouselProps

const CarouselContext = React.createContext<CarouselContextProps | null>(null)

function useCarousel() {
  const context = React.useContext(CarouselContext)

  if (!context) {
    throw new Error("useCarousel must be used within a <Carousel />")
  }

  return context
}

const Carousel = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement> & CarouselProps
>((
  {
    orientation = "horizontal",
    opts,
    setApi,
    plugins,
    className,
    children,
    ...props
  },
  ref
) => {
  const [carouselRef, api] = useEmblaCarousel(
    {
      ...opts,
      axis: orientation === "horizontal" ? "x" : "y",
    },
    plugins
  )
  const [canScrollPrev, setCanScrollPrev] = React.useState(false)
  const [canScrollNext, setCanScrollNext] = React.useState(false)

  const onSelect = React.useCallback((api: CarouselApi) => {
    if (!api) {
      return
    }

    setCanScrollPrev(api.canScrollPrev())
    setCanScrollNext(api.canScrollNext())
  }, [])

  const scrollPrev = React.useCallback(() => {
    api?.scrollPrev()
  }, [api])

  const scrollNext = React.useCallback(() => {
    api?.scrollNext()
  }, [api])

  const handleKeyDown = React.useCallback(
    (event: React.KeyboardEvent<HTMLDivElement>) => {
      if (event.key === "ArrowLeft") {
        event.preventDefault()
        scrollPrev()
      } else if (event.key === "ArrowRight") {
        event.preventDefault()
        scrollNext()
      }
    },
    [scrollPrev, scrollNext]
  )

  return (
    <CarouselContext.Provider value={{ carouselRef, api, scrollPrev, scrollNext, canScrollPrev, canScrollNext }}>
      <div
        ref={ref}
        className={cn("relative", className)}
        {...props}
      >
        {children}
      </div>
    </CarouselContext.Provider>
  )
)
```

```

    React.useEffect(() => {
      if (!api || !setApi) {
        return
      }

      setApi(api)
    }, [api, setApi])

    React.useEffect(() => {
      if (!api) {
        return
      }

      onSelect(api)
      api.on("reInit", onSelect)
      api.on("select", onSelect)

      return () => {
        api?.off("select", onSelect)
      }
    }, [api, onSelect])

    return (
      <CarouselContext.Provider
        value={{
          carouselRef,
          api: api,
          opts,
          orientation:
            orientation || (opts?.axis === "y" ? "vertical" : "horizontal"),
          scrollPrev,
          scrollNext,
          canScrollPrev,
          canScrollNext,
        }}
      >
        <div
          ref={ref}
          onKeyDownCapture={handleKeyDown}
          className={cn("relative", className)}
          role="region"
          aria-roledescription="carousel"
          {...props}
        >
          {children}
        </div>
      </CarouselContext.Provider>
    )
  }
)
Carousel.displayName = "Carousel"

const CarouselContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { carouselRef, orientation } = useCarousel()

  return (
    <div ref={carouselRef} className="overflow-hidden">
      <div
        ref={ref}
        className={cn(
          "flex",
          orientation === "horizontal" ? "-ml-4" : "-mt-4 flex-col",
          className
        )}
        {...props}
      />
    </div>
  )
})
CarouselContent.displayName = "CarouselContent"

const CarouselItem = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { orientation } = useCarousel()

  return (
    <div
      ref={ref}
      role="group"
      aria-roledescription="slide"
      className={cn(
        "min-w-0 shrink-0 grow-0 basis-full",
        orientation === "horizontal" ? "pl-4" : "pt-4",
        className
      )}
      {...props}
    />
  )
})
CarouselItem.displayName = "CarouselItem"

const CarouselPrevious = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<typeof Button>
>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
  const { orientation, scrollPrev, canScrollPrev } = useCarousel()

```

```

return (
  <Button
    ref={ref}
    variant={variant}
    size={size}
    className={cn(
      "absolute h-8 w-8 rounded-full",
      orientation === "horizontal"
        ? "-left-12 top-1/2 -translate-y-1/2"
        : "-top-12 left-1/2 -translate-x-1/2 rotate-90",
      className
    )}
    disabled={!canScrollPrev}
    onClick={scrollPrev}
    {...props}
  />
  <ArrowLeft className="h-4 w-4" />
  <span className="sr-only">Previous slide</span>
  </Button>
)
CarouselPrevious.displayName = "CarouselPrevious"

const CarouselNext = React.forwardRef(
  HTMLButtonElement,
  React.ComponentProps<Button>,
  >({ className, variant = "outline", size = "icon", ...props }, ref) => {
    const { orientation, scrollNext, canScrollNext } = useCarousel()

    return (
      <Button
        ref={ref}
        variant={variant}
        size={size}
        className={cn(
          "absolute h-8 w-8 rounded-full",
          orientation === "horizontal"
            ? "-right-12 top-1/2 -translate-y-1/2"
            : "-bottom-12 left-1/2 -translate-x-1/2 rotate-90",
          className
        )}
        disabled={!canScrollNext}
        onClick={scrollNext}
        {...props}
      />
      <ArrowRight className="h-4 w-4" />
      <span className="sr-only">Next slide</span>
      </Button>
    )
  }
)
CarouselNext.displayName = "CarouselNext"

export {
  type CarouselApi,
  Carousel,
  CarouselContent,
  CarouselItem,
  CarouselPrevious,
  CarouselNext,
}

```

File: client/src/components/ui/chart.tsx

```
"use client"

import * as React from "react"
import * as RechartsPrimitive from "recharts"

import { cn } from "@/lib/utils"

// Format: { THEME_NAME: CSS_SELECTOR }
const THEMES = { light: "", dark: ".dark" } as const

export type ChartConfig = {
  [k in string]: {
    label?: React.ReactNode
    icon?: React.ComponentType
  } & (
    | { color?: string; theme?: never }
    | { color?: never; theme: Record<keyof typeof THEMES, string> }
  )
}

type ChartContextProps = {
  config: ChartConfig
}

const ChartContext = React.createContext<ChartContextProps | null>(null)

function useChart() {
  const context = React.useContext(ChartContext)

  if (!context) {
    throw new Error("useChart must be used within a <ChartContainer />")
  }

  return context
}

const ChartContainer = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & {
    config: ChartConfig
    children: React.ComponentProps<
      typeof RechartsPrimitive.ResponsiveContainer
    >["children"]
  }
>(({ id, className, children, config, ...props }, ref) => {
  const uniqueId = React.useId()
  const chartId = `chart-${id || uniqueId.replace(/:/g, "")}`

  return (
    <ChartContext.Provider value={{ config }}>
      <div
        data-chart={chartId}
        ref={ref}
        className={cn(
          "flex aspect-video justify-center text-xs [&:_recharts-cartesian-axis-tick_text]:fill-muted-foreground [&:_recharts-cartesian-grid_line[stroke]]:stroke-muted-foreground",
          className
        )}
        {...props}
      >
        <ChartStyle id={chartId} config={config} />
        <RechartsPrimitive.ResponsiveContainer>
          {children}
        </RechartsPrimitive.ResponsiveContainer>
      </div>
    </ChartContext.Provider>
  )
})
ChartContainer.displayName = "Chart"

const ChartStyle = ({ id, config }: { id: string; config: ChartConfig }) => {
  const colorConfig = Object.entries(config).filter(
    ([, config]) => config.theme || config.color
  )

  if (!colorConfig.length) {
    return null
  }

  return (
    <style
      dangerouslySetInnerHTML={{
        __html: Object.entries(THEMES)
          .map(
            ([theme, prefix]) => `
${prefix} [data-chart=${id}] {
${colorConfig
      .map(([, itemConfig]) => {
        const color =
          itemConfig.theme?.[theme as keyof typeof itemConfig.theme] ||
          itemConfig.color
        return color ? `--color-${key}: ${color};` : null
      })
      .join("\n")}
    `
          )
      }
    >
  )
}

```

```

    }}
    /&gt;
  )
}

const ChartTooltip = RechartsPrimitive.Tooltip

const ChartTooltipContent = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<typeof RechartsPrimitive.Tooltip> & {
    React.ComponentProps<"div"> & {
      hideLabel?: boolean
      hideIndicator?: boolean
      indicator?: "line" | "dot" | "dashed"
      nameKey?: string
      labelKey?: string
    }
  }
>({
  (
    {
      active,
      payload,
      className,
      indicator = "dot",
      hideLabel = false,
      hideIndicator = false,
      label,
      labelFormatter,
      labelClassName,
      formatter,
      color,
      nameKey,
      labelKey,
    },
    ref
  ) => {
    const { config } = useChart()

    const tooltipLabel = React.useMemo(() => {
      if (hideLabel || !payload?.length) {
        return null
      }

      const [item] = payload
      const key = `${labelKey || item?.dataKey || item?.name || "value"}`
      const itemConfig = getPayloadConfigFromPayload(config, item, key)
      const value =
        !labelKey & & typeof label === "string"
          ? config[label as keyof typeof config]?.label || label
          : itemConfig?.label

      if (labelFormatter) {
        return (
          <div className={cn("font-medium", labelClassName)}>
            {labelFormatter(value, payload)}
          </div>
        )
      }

      if (!value) {
        return null
      }

      return <div className={cn("font-medium", labelClassName)}>{value}</div>
    }, [
      label,
      labelFormatter,
      payload,
      hideLabel,
      labelClassName,
      config,
      labelKey,
    ])

    if (!active || !payload?.length) {
      return null
    }

    const nestLabel = payload.length === 1 & & indicator !== "dot"

    return (
      <div
        ref={ref}
        className={cn(
          "grid min-w-[8rem] items-start gap-1.5 rounded-lg border border-border/50 bg-background px-2.5 py-1.5 text-xs shadow-xl",
          className
        )}
      >
        {!nestLabel ? tooltipLabel : null}
        <div className="grid gap-1.5">
          {payload.map((item, index) => {
            const key = `${nameKey || item.name || item.dataKey || "value"}`
            const itemConfig = getPayloadConfigFromPayload(config, item, key)
            const indicatorColor = color || item.payload.fill || item.color

            return (
              <div
                key={item.dataKey}
                className={cn(
                  "flex w-full flex-wrap items-stretch gap-2 [& & ;svg]:h-2.5 [& & ;svg]:w-2.5 [& & ;svg]:text-muted-foreground",
                  indicator === "dot" & & "items-center"
                )}
              >

```

```

    })
    &gt;
    {formatter &amp;&amp; item?.value !== undefined &amp;&amp; item.name ? (
      formatter(item.value, item.name, item, index, item.payload)
    ) : (
      &lt;&gt;
      {itemConfig?.icon ? (
        &lt;itemConfig.icon /&gt;
      ) : (
        !hideIndicator &amp;&amp; (
          &lt;div
            className={cn(
              "shrink-0 rounded-[2px] border-[--color-border] bg-[--color-bg]",
              {
                "h-2.5 w-2.5": indicator === "dot",
                "w-1": indicator === "line",
                "w-0 border-[1.5px] border-dashed bg-transparent":
                  indicator === "dashed",
                "my-0.5": nestLabel &amp;&amp; indicator === "dashed",
              }
            )}
            style={
              {
                "--color-bg": indicatorColor,
                "--color-border": indicatorColor,
              } as React.CSSProperties
            }
          /&gt;
        )
      )}
      &lt;div
        className={cn(
          "flex flex-1 justify-between leading-none",
          nestLabel ? "items-end" : "items-center"
        )}
      &gt;
        &lt;div className="grid gap-1.5"&gt;
          {nestLabel ? tooltipLabel : null}
          &lt;span className="text-muted-foreground"&gt;
            {itemConfig?.label || item.name}
          &lt;/span>
        &lt;/div>
        {item.value &amp;&amp; (
          &lt;span className="font-mono font-medium tabular-nums text-foreground"&gt;
            {item.value.toLocaleString()}
          &lt;/span>
        )}
      &lt;/div>
    &lt;/&gt;
  )}
  &lt;/div>
  &lt;/div>
)
}
)
ChartTooltipContent.displayName = "ChartTooltip"

const ChartLegend = RechartsPrimitive.Legend

const ChartLegendContent = React.forwardRef&lt;&lt;
  HTMLDivElement,
  React.ComponentProps&lt;&lt;"div"&gt;& &amp;
  Pick&lt;&lt;RechartsPrimitive.LegendProps, "payload" | "verticalAlign"&gt;& &amp; {
    hideIcon?: boolean
    nameKey?: string
  }
&gt;({
  (
    { className, hideIcon = false, payload, verticalAlign = "bottom", nameKey },
    ref
  ) =&gt; {
    const { config } = useChart()

    if (!payload?.length) {
      return null
    }

    return (
      &lt;div
        ref={ref}
        className={cn(
          "flex items-center justify-center gap-4",
          verticalAlign === "top" ? "pb-3" : "pt-3",
          className
        )}
      &gt;
        {payload.map((item) =&gt; {
          const key = `${nameKey || item.dataKey || "value"}`
          const itemConfig = getPayloadConfigFromPayload(config, item, key)

          return (
            &lt;div
              key={item.value}
              className={cn(
                "flex items-center gap-1.5 [&amp;&svg]:h-3 [&amp;&svg]:w-3 [&amp;&svg]:text-muted-foreground"
              )}
            &gt;
              {itemConfig?.icon &amp;&amp; !hideIcon ? (

```

```

        <itemConfig.icon />
      ) : (
        <div
          className="h-2 w-2 shrink-0 rounded-[2px]"
          style={{
            backgroundColor: item.color,
          }}
        />
      )
    }
    {itemConfig?.label}
  </div>
)
}}
</div>
)
}
}
ChartLegendContent.displayName = "ChartLegend"

// Helper to extract item config from a payload.
function getPayloadConfigFromPayload(
  config: ChartConfig,
  payload: unknown,
  key: string
) {
  if (typeof payload !== "object" || payload === null) {
    return undefined
  }

  const payloadPayload =
    "payload" in payload &&
    typeof payload.payload === "object" &&
    payload.payload !== null
      ? payload.payload
      : undefined

  let configLabelKey: string = key

  if (
    key in payload &&
    typeof payload[key as keyof typeof payload] === "string"
  ) {
    configLabelKey = payload[key as keyof typeof payload] as string
  } else if (
    payloadPayload &&
    key in payloadPayload &&
    typeof payloadPayload[key as keyof typeof payloadPayload] === "string"
  ) {
    configLabelKey = payloadPayload[
      key as keyof typeof payloadPayload
    ] as string
  }

  return configLabelKey in config
    ? config[configLabelKey]
    : config[key as keyof typeof config]
}

export {
  ChartContainer,
  ChartTooltip,
  ChartTooltipContent,
  ChartLegend,
  ChartLegendContent,
  ChartStyle,
}

```

File: client/src/components/ui/checkbox.tsx

```
import * as React from "react"
import * as CheckboxPrimitive from "@radix-ui/react-checkbox"
import { Check } from "lucide-react"

import { cn } from "@/lib/utlis"

const Checkbox = React.forwardRef<
  React.ElementRef<typeof CheckboxPrimitive.Root>;,
  React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>;
>(({ className, ...props }, ref) => {
  <CheckboxPrimitive.Root
    ref={ref}
    className={cn(
      "peer h-4 w-4 shrink-0 rounded-sm border border-primary ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring f
      className
    )}
    {...props}
  >
    <CheckboxPrimitive.Indicator
      className={cn("flex items-center justify-center text-current")}
    >
      <Check className="h-4 w-4" />
    </CheckboxPrimitive.Indicator>
  </CheckboxPrimitive.Root>
)}
Checkbox.displayName = CheckboxPrimitive.Root.displayName

export { Checkbox }
```

File: client/src/components/ui/collapsible.tsx

```
"use client"

import * as CollapsiblePrimitive from "@radix-ui/react-collapsible"

const Collapsible = CollapsiblePrimitive.Root

const CollapsibleTrigger = CollapsiblePrimitive.CollapsibleTrigger

const CollapsibleContent = CollapsiblePrimitive.CollapsibleContent

export { Collapsible, CollapsibleTrigger, CollapsibleContent }
```

File: client/src/components/ui/command.tsx

```
import * as React from "react"
import { type DialogProps } from "@radix-ui/react-dialog"
import { Command as CommandPrimitive } from "cmdk"
import { Search } from "lucide-react"

import { cn } from "@/lib/utils"
import { Dialog, DialogContent } from "@/components/ui/dialog"

const Command = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive>;
  React.ComponentPropsWithoutRef<typeof CommandPrimitive>;
>(({ className, ...props }, ref) => {
  <CommandPrimitive
    ref={ref}
    className={cn(
      "flex h-full w-full flex-col overflow-hidden rounded-md bg-popover text-popover-foreground",
      className
    )}
    {...props}
  />
})
Command.displayName = CommandPrimitive.displayName

const CommandDialog = ({ children, ...props }: DialogProps) => {
  return (
    <Dialog {...props}>
      <DialogContent className="overflow-hidden p-0 shadow-lg">
        <Command className="[&_[:cmdk-group-heading]]:px-2 [&_[:cmdk-group-heading]]:font-medium [&_[:cmdk-group-heading]]:text-muted-foreground">
          {children}
        </Command>
      </DialogContent>
    </Dialog>
  )
}

const CommandInput = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.Input>;
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.Input>;
>(({ className, ...props }, ref) => {
  <div className="flex items-center border-b px-3 cmdk-input-wrapper">
    <Search className="mr-2 h-4 w-4 shrink-0 opacity-50" />
    <CommandPrimitive.Input
      ref={ref}
      className={cn(
        "flex h-11 w-full rounded-md bg-transparent py-3 text-sm outline-none placeholder:text-muted-foreground disabled:cursor-not-allowed disabled:opacity-50",
        className
      )}
      {...props}
    />
  </div>
})
CommandInput.displayName = CommandPrimitive.Input.displayName

const CommandList = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.List>;
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.List>;
>(({ className, ...props }, ref) => {
  <CommandPrimitive.List
    ref={ref}
    className={cn("max-h-[300px] overflow-y-auto overflow-x-hidden", className)}
    {...props}
  />
})
CommandList.displayName = CommandPrimitive.List.displayName

const CommandEmpty = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.Empty>;
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.Empty>;
>((props, ref) => {
  <CommandPrimitive.Empty
    ref={ref}
    className="py-6 text-center text-sm"
    {...props}
  />
})
CommandEmpty.displayName = CommandPrimitive.Empty.displayName

const CommandGroup = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.Group>;
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.Group>;
>(({ className, ...props }, ref) => {
  <CommandPrimitive.Group
    ref={ref}
    className={cn(
      "overflow-hidden p-1 text-foreground [&_[:cmdk-group-heading]]:px-2 [&_[:cmdk-group-heading]]:py-1.5 [&_[:cmdk-group-heading]]:text-xs [&_[:cmdk-group-heading]]:font-medium",
      className
    )}
    {...props}
  />
})
CommandGroup.displayName = CommandPrimitive.Group.displayName

const CommandSeparator = React.forwardRef<
```

```

    React.ElementRef<typeof CommandPrimitive.Separator>;
    React.ComponentPropsWithoutRef<typeof CommandPrimitive.Separator>;
    &gt;(({ className, ...props }, ref) => {
      <CommandPrimitive.Separator
        ref={ref}
        className={cn("-mx-1 h-py bg-border", className)}
        {...props}
      />
    ))
    CommandSeparator.displayName = CommandPrimitive.Separator.displayName

    const CommandItem = React.forwardRef<
      React.ElementRef<typeof CommandPrimitive.Item>;
      React.ComponentPropsWithoutRef<typeof CommandPrimitive.Item>;
      &gt;(({ className, ...props }, ref) => {
        <CommandPrimitive.Item
          ref={ref}
          className={cn(
            "relative flex cursor-default gap-2 select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none data-[disabled=true]:pointer-events-none data-[focus=true]:bg-accent data-[focus=true]:text-accent-foreground",
            className
          )}
          {...props}
        />
      ))
    CommandItem.displayName = CommandPrimitive.Item.displayName

    const CommandShortcut = ({
      className,
      ...props
    }: React.HTMLAttributes<HTMLSpanElement>) => {
      return (
        <span
          className={cn(
            "ml-auto text-xs tracking-widest text-muted-foreground",
            className
          )}
          {...props}
        />
      )
    }
    CommandShortcut.displayName = "CommandShortcut"

    export {
      Command,
      CommandDialog,
      CommandInput,
      CommandList,
      CommandEmpty,
      CommandGroup,
      CommandItem,
      CommandShortcut,
      CommandSeparator,
    }

```

File: client/src/components/ui/context-menu.tsx

```
import * as React from "react"
import * as ContextMenuPrimitive from "@radix-ui/react-context-menu"
import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utlils"

const ContextMenu = ContextMenuPrimitive.Root

const ContextMenuTrigger = ContextMenuPrimitive.Trigger

const ContextMenuGroup = ContextMenuPrimitive.Group

const ContextMenuPortal = ContextMenuPrimitive.Portal

const ContextMenuSub = ContextMenuPrimitive.Sub

const ContextMenuRadioGroup = ContextMenuPrimitive.RadioGroup

const ContextMenuSubTrigger = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.SubTrigger>;,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubTrigger>; & {
    inset?: boolean
  }
>(({ className, inset, children, ...props }, ref) => {
  <ContextMenuItemPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",
      inset & "pl-8",
      className
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto h-4 w-4" />
  </ContextMenuItemPrimitive.SubTrigger>
</>
))
ContextMenuItemSubTrigger.displayName = ContextMenuPrimitive.SubTrigger.displayName

const ContextMenuItemSubContent = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.SubContent>;,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubContent>;
>(({ className, ...props }, ref) => {
  <ContextMenuItemPrimitive.SubContent
    ref={ref}
    className={cn(
      "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",
      className
    )}
    {...props}
  >
    {children}
  </>
</>
))
ContextMenuItemSubContent.displayName = ContextMenuPrimitive.SubContent.displayName

const ContextMenuItemContent = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.Content>;,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Content>;
>(({ className, ...props }, ref) => {
  <ContextMenuItemPrimitive.Portal>
    <ContextMenuItemPrimitive.Content
      ref={ref}
      className={cn(
        "z-50 max-h-[--radix-context-menu-content-available-height] min-w-[8rem] overflow-y-auto overflow-x-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow",
        className
      )}
      {...props}
    >
      {children}
    </>
  </ContextMenuItemPrimitive.Portal>
</>
))
ContextMenuItemContent.displayName = ContextMenuPrimitive.Content.displayName

const ContextMenuItemItem = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.Item>;,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Item>; & {
    inset?: boolean
  }
>(({ className, inset, ...props }, ref) => {
  <ContextMenuItemPrimitive.Item
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none",
      inset & "pl-8",
      className
    )}
    {...props}
  >
    {children}
  </>
</>
))
ContextMenuItemItem.displayName = ContextMenuPrimitive.Item.displayName

const ContextMenuItemCheckboxItem = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.CheckboxItem>;,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.CheckboxItem>;
>(({ className, children, checked, ...props }, ref) => {
  <ContextMenuItemPrimitive.CheckboxItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none",
      className
    )}
    checked={checked}
    {...props}
  >
    <input checked={checked} type="checkbox" />
    {children}
  </>
</>
))
ContextMenuItemCheckboxItem.displayName = ContextMenuPrimitive.CheckboxItem.displayName
```

```

        className
    )}
    checked={checked}
    {...props}
  </>
  <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
    <ContextMenuPrimitive.ItemIndicator>
    <Check className="h-4 w-4" />
    </ContextMenuPrimitive.ItemIndicator>
  </span>
  {children}
  </ContextMenuPrimitive.CheckboxItem>
)
ContextMenuCheckboxItem.displayName =
  ContextMenuPrimitive.CheckboxItem.displayName

const ContextMenuRadioItem = React.forwardRef(
  React.ElementRef<typeof ContextMenuPrimitive.RadioItem>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.RadioItem>
  &gt;({ className, children, ...props }, ref) => {
    <ContextMenuPrimitive.RadioItem
      ref={ref}
      className={cn(
        "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-
        className
      )}
      {...props}
    >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <ContextMenuPrimitive.ItemIndicator>
      <Circle className="h-2 w-2 fill-current" />
      </ContextMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </ContextMenuPrimitive.RadioItem>
)
ContextMenuRadioItem.displayName = ContextMenuPrimitive.RadioItem.displayName

const ContextMenuLabel = React.forwardRef(
  React.ElementRef<typeof ContextMenuPrimitive.Label>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Label> & {
    inset?: boolean
  }
  &gt;({ className, inset, ...props }, ref) => {
    <ContextMenuPrimitive.Label
      ref={ref}
      className={cn(
        "px-2 py-1.5 text-sm font-semibold text-foreground",
        inset & "pl-8",
        className
      )}
      {...props}
    >
    </>
  >
  </>
  ContextMenuLabel.displayName = ContextMenuPrimitive.Label.displayName

const ContextMenuSeparator = React.forwardRef(
  React.ElementRef<typeof ContextMenuPrimitive.Separator>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Separator>
  &gt;({ className, ...props }, ref) => {
    <ContextMenuPrimitive.Separator
      ref={ref}
      className={cn("-mx-1 my-1 h-px bg-border", className)}
      {...props}
    >
    </>
  >
  </>
  ContextMenuSeparator.displayName = ContextMenuPrimitive.Separator.displayName

const ContextMenuShortcut = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLSpanElement>) => {
  return (
    <span
      className={cn(
        "ml-auto text-xs tracking-widest text-muted-foreground",
        className
      )}
      {...props}
    >
    </>
  )
}
ContextMenuShortcut.displayName = "ContextMenuShortcut"

export {
  ContextMenu,
  ContextMenuTrigger,
  ContextMenuContent,
  ContextMenuItem,
  ContextMenuCheckboxItem,
  ContextMenuRadioItem,
  ContextMenuLabel,
  ContextMenuSeparator,
  ContextMenuShortcut,
  ContextMenuGroup,
  ContextMenuPortal,
  ContextMenuSub,
  ContextMenuSubContent,
  ContextMenuSubTrigger,
  ContextMenuRadioGroup,
}

```

File: client/src/components/ui/dialog.tsx

```
"use client"

import * as React from "react"
import * as DialogPrimitive from "@radix-ui/react-dialog"
import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const Dialog = DialogPrimitive.Root

const DialogTrigger = DialogPrimitive.Trigger

const DialogPortal = DialogPrimitive.Portal

const DialogClose = DialogPrimitive.Close

const DialogOverlay = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Overlay>;,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Overlay>;
>(({ className, ...props }, ref) => {
  <DialogPrimitive.Overlay
    ref={ref}
    className={cn(
      "fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0"
      className
    )}
    {...props}
  />
})
DialogOverlay.displayName = DialogPrimitive.Overlay.displayName

const DialogContent = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Content>;,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Content>;
>(({ className, children, ...props }, ref) => {
  <DialogPortal>
    <DialogOverlay />
    <DialogPrimitive.Content
      ref={ref}
      className={cn(
        "fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0"
        className
      )}
      {...props}
    >
      <DialogPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2" />
      <X className="h-4 w-4" />
      <span className="sr-only">Close</span>
    </DialogPrimitive.Content>
  </DialogPortal>
})
DialogContent.displayName = DialogPrimitive.Content.displayName

const DialogHeader = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLElement>) => {
  <div
    className={cn(
      "flex flex-col space-y-1.5 text-center sm:text-left",
      className
    )}
    {...props}
  />
)
DialogHeader.displayName = "DialogHeader"

const DialogFooter = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLElement>) => {
  <div
    className={cn(
      "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
      className
    )}
    {...props}
  />
)
DialogFooter.displayName = "DialogFooter"

const DialogTitle = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Title>;,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Title>;
>(({ className, ...props }, ref) => {
  <DialogPrimitive.Title
    ref={ref}
    className={cn(
      "text-lg font-semibold leading-none tracking-tight",
      className
    )}
    {...props}
  />
})
DialogTitle.displayName = DialogPrimitive.Title.displayName
```

```

const DialogDescription = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Description>,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Description>
>(({ className, ...props }, ref) => {
  <DialogPrimitive.Description
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
))
DialogDescription.displayName = DialogPrimitive.Description.displayName

export {
  Dialog,
  DialogPortal,
  DialogOverlay,
  DialogClose,
  DialogTrigger,
  DialogContent,
  DialogHeader,
  DialogFooter,
  DialogTitle,
  DialogDescription,
}

```

File: client/src/components/ui/drawer.tsx

```
"use client"

import * as React from "react"
import { Drawer as DrawerPrimitive } from "vaul"

import { cn } from "@/lib/utils"

const Drawer = ({
  shouldScaleBackground = true,
  ...props
}: React.ComponentProps<typeof DrawerPrimitive.Root>) => { (
  <DrawerPrimitive.Root
    shouldScaleBackground={shouldScaleBackground}
    {...props}
  />
)
Drawer.displayName = "Drawer"

const DrawerTrigger = DrawerPrimitive.Trigger

const DrawerPortal = DrawerPrimitive.Portal

const DrawerClose = DrawerPrimitive.Close

const DrawerOverlay = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Overlay>;
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Overlay>
>(({ className, ...props }, ref) => { (
  <DrawerPrimitive.Overlay
    ref={ref}
    className={cn("fixed inset-0 z-50 bg-black/80", className)}
    {...props}
  />
)
DrawerOverlay.displayName = DrawerPrimitive.Overlay.displayName

const DrawerContent = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Content>;
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Content>
>(({ className, children, ...props }, ref) => { (
  <DrawerPortal>
    <DrawerOverlay />
    <DrawerPrimitive.Content
      ref={ref}
      className={cn(
        "fixed inset-x-0 bottom-0 z-50 mt-24 flex h-auto flex-col rounded-t-[10px] border bg-background",
        className
      )}
      {...props}
    />
    <div className="mx-auto mt-4 h-2 w-[100px] rounded-full bg-muted" />
    {children}
  </DrawerPrimitive.Content>
</DrawerPortal>
)
DrawerContent.displayName = "DrawerContent"

const DrawerHeader = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => { (
  <div
    className={cn("grid gap-1.5 p-4 text-center sm:text-left", className)}
    {...props}
  />
)
DrawerHeader.displayName = "DrawerHeader"

const DrawerFooter = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => { (
  <div
    className={cn("mt-auto flex flex-col gap-2 p-4", className)}
    {...props}
  />
)
DrawerFooter.displayName = "DrawerFooter"

const DrawerTitle = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Title>;
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Title>
>(({ className, ...props }, ref) => { (
  <DrawerPrimitive.Title
    ref={ref}
    className={cn(
      "text-lg font-semibold leading-none tracking-tight",
      className
    )}
    {...props}
  />
)
DrawerTitle.displayName = DrawerPrimitive.Title.displayName

const DrawerDescription = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Description>;
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Description>
```

```
&gt;({ className, ...props }, ref) => {  
  <DrawerPrimitive.Description  
    ref={ref}  
    className={cn("text-sm text-muted-foreground", className)}  
    {...props}  
  />  
})  
DrawerDescription.displayName = DrawerPrimitive.Description.displayName  
  
export {  
  Drawer,  
  DrawerPortal,  
  DrawerOverlay,  
  DrawerTrigger,  
  DrawerClose,  
  DrawerContent,  
  DrawerHeader,  
  DrawerFooter,  
  DrawerTitle,  
  DrawerDescription,  
}
```

File: client/src/components/ui/dropdown-menu.tsx

```
import * as React from "react"
import * as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu"
import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utlils"

const DropdownMenu = DropdownMenuPrimitive.Root

const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger

const DropdownMenuGroup = DropdownMenuPrimitive.Group

const DropdownMenuPortal = DropdownMenuPrimitive.Portal

const DropdownMenuSub = DropdownMenuPrimitive.Sub

const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup

const DropdownMenuSubTrigger = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {
    inset?: boolean
  }
>(({ className, inset, children, ...props }, ref) => {
  <DropdownMenuPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center gap-2 rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent data-[state=open]:bg-accent [&svg]:size-4",
      inset & & "pl-8",
      className
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto" />
  </DropdownMenuPrimitive.SubTrigger>
</>
))
DropdownMenuSubTrigger.displayName =
  DropdownMenuPrimitive.SubTrigger.displayName

const DropdownMenuSubContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>
>(({ className, ...props }, ref) => {
  <DropdownMenuPrimitive.SubContent
    ref={ref}
    className={cn(
      "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out",
      className
    )}
    {...props}
  >
  </>
</>
))
DropdownMenuSubContent.displayName =
  DropdownMenuPrimitive.SubContent.displayName

const DropdownMenuContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>
>(({ className, sideOffset = 4, ...props }, ref) => {
  <DropdownMenuPrimitive.Portal>
    <DropdownMenuPrimitive.Content
      ref={ref}
      sideOffset={sideOffset}
      className={cn(
        "z-50 max-h-[var(--radix-dropdown-menu-content-available-height)] min-w-[8rem] overflow-y-auto overflow-x-hidden rounded-md border bg-popover p-1 text-popover-foreground",
        className
      )}
      {...props}
    >
  </>
  </DropdownMenuPrimitive.Portal>
</>
))
DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName

const DropdownMenuItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {
    inset?: boolean
  }
>(({ className, inset, ...props }, ref) => {
  <DropdownMenuPrimitive.Item
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center gap-2 rounded-sm px-2 py-1.5 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground",
      inset & & "pl-8",
      className
    )}
    {...props}
  >
  </>
</>
))
DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName

const DropdownMenuCheckboxItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>
>(({ className, children, checked, ...props }, ref) => {
  <DropdownMenuPrimitive.CheckboxItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground",
      checked & & "text-accent-foreground",
      className
    )}
    checked={checked}
    {...props}
  >
    {children}
  </>
</>
))
DropdownMenuCheckboxItem.displayName =
  DropdownMenuPrimitive.CheckboxItem.displayName
```

```

    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground",
      className
    )}
    checked={checked}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <DropdownMenuPrimitive.ItemIndicator>
      <Check className="h-4 w-4" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.CheckboxItem>
)
DropdownMenuCheckboxItem.displayName =
  DropdownMenuPrimitive.CheckboxItem.displayName

const DropdownMenuRadioItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>
>(({ className, children, ...props }, ref) => {
  <DropdownMenuPrimitive.RadioItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground",
      className
    )}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <DropdownMenuPrimitive.ItemIndicator>
      <Circle className="h-2 w-2 fill-current" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.RadioItem>
)
DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName

const DropdownMenuLabel = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Label>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {
    inset?: boolean
  }
>(({ className, inset, ...props }, ref) => {
  <DropdownMenuPrimitive.Label
    ref={ref}
    className={cn(
      "px-2 py-1.5 text-sm font-semibold",
      inset & & "pl-8",
      className
    )}
    {...props}
  />
)
DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName

const DropdownMenuSeparator = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Separator>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>
>(({ className, ...props }, ref) => {
  <DropdownMenuPrimitive.Separator
    ref={ref}
    className={cn("-mx-1 my-1 h-px bg-muted", className)}
    {...props}
  />
)
DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName

const DropdownMenuShortcut = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLSpanElement>) => {
  return (
    <span
      className={cn("ml-auto text-xs tracking-widest opacity-60", className)}
      {...props}
    />
  )
}
DropdownMenuShortcut.displayName = "DropdownMenuShortcut"

export {
  DropdownMenu,
  DropdownMenuTrigger,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuCheckboxItem,
  DropdownMenuRadioItem,
  DropdownMenuLabel,
  DropdownMenuSeparator,
  DropdownMenuShortcut,
  DropdownMenuGroup,
  DropdownMenuPortal,
  DropdownMenuSub,
  DropdownMenuSubContent,
  DropdownMenuSubTrigger,
  DropdownMenuRadioGroup,
}

```

File: client/src/components/ui/form.tsx

```
"use client"

import * as React from "react"
import * as LabelPrimitive from "@radix-ui/react-label"
import { Slot } from "@radix-ui/react-slot"
import {
  Controller,
  FormProvider,
  useFormContext,
  type ControllerProps,
  type FieldPath,
  type FieldValues,
} from "react-hook-form"

import { cn } from "@/lib/utils"
import { Label } from "@/components/ui/label"

const Form = FormProvider

type FormFieldContextValue<T> = {
  fieldValues: TFieldValues extends FieldValues ? FieldValues : TFieldValues
  &TName extends FieldPath<TFieldValues> ? FieldPath<TFieldValues> : TName
}

const FormFieldContext = React.createContext<FormFieldContextValue<>({
  } as FormFieldContextValue
)>

const FormField = <T> {
  TFieldValues extends FieldValues = FieldValues,
  TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>
} >({
  ...props
}): ControllerProps<TFieldValues, TName> => {
  return (
    <FormFieldContext.Provider value={{ name: props.name }}>
      <Controller {...props} />
    </FormFieldContext.Provider>
  )
}

const useFormField = () => {
  const fieldContext = React.useContext(FormFieldContext)
  const itemContext = React.useContext(FormItemContext)
  const { getFieldState, formState } = useFormContext()

  const fieldState = getFieldState(fieldContext.name, formState)

  if (!fieldContext) {
    throw new Error("useFormField should be used within <FormField>")
  }

  const { id } = itemContext

  return {
    id,
    name: fieldContext.name,
    formItemId: `${id}-form-item`,
    formDescriptionId: `${id}-form-item-description`,
    formMessageId: `${id}-form-item-message`,
    ...fieldState,
  }
}

type FormItemContextValue = {
  id: string
}

const FormItemContext = React.createContext<FormItemContextValue>({
  } as FormItemContextValue
)

const FormItem = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
> &({ className, ...props }, ref) => {
  const id = React.useId()

  return (
    <FormItemContext.Provider value={{ id }}>
      <div ref={ref} className={cn("space-y-2", className)} {...props} />
    </FormItemContext.Provider>
  )
})

FormItem.displayName = "FormItem"

const FormLabel = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root>
> &({ className, ...props }, ref) => {
  const { error, formItemId } = useFormField()

  return (
    <Label
      ref={ref}
    />
  )
}
```

```

      className={cn(error && "text-destructive", className)}
      htmlFor={formItemId}
      {...props}
    />
  )
})
FormLabel.displayName = "FormLabel"

const FormControl = React.forwardRef<
  React.ElementRef<typeof Slot>,
  React.ComponentPropsWithoutRef<typeof Slot>
>((({ ...props }, ref) => {
  const { error, formItemId, formDescriptionId, formMessageId } = useFormField()

  return (
    <Slot
      ref={ref}
      id={formItemId}
      aria-describedby={
        !error
          ? `${formDescriptionId}`
          : `${formDescriptionId} ${formMessageId}`
      }
      aria-invalid={!!error}
      {...props}
    />
  )
})
FormControl.displayName = "FormControl"

const FormDescription = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>((({ className, ...props }, ref) => {
  const { formDescriptionId } = useFormField()

  return (
    <p
      ref={ref}
      id={formDescriptionId}
      className={cn("text-sm text-muted-foreground", className)}
      {...props}
    />
  )
})
FormDescription.displayName = "FormDescription"

const FormMessage = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>((({ className, children, ...props }, ref) => {
  const { error, formMessageId } = useFormField()
  const body = error ? String(error?.message ?? "") : children

  if (!body) {
    return null
  }

  return (
    <p
      ref={ref}
      id={formMessageId}
      className={cn("text-sm font-medium text-destructive", className)}
      {...props}
    >
      {body}
    </p>
  )
})
FormMessage.displayName = "FormMessage"

export {
  useFormField,
  Form,
  FormItem,
  FormLabel,
  FormControl,
  FormDescription,
  FormMessage,
  FormField,
}
```

File: client/src/components/ui/hover-card.tsx

```
"use client"

import * as React from "react"
import * as HoverCardPrimitive from "@radix-ui/react-hover-card"

import { cn } from "@/lib/utils"

const HoverCard = HoverCardPrimitive.Root

const HoverCardTrigger = HoverCardPrimitive.Trigger

const HoverCardContent = React.forwardRef<
  React.ElementRef<typeof HoverCardPrimitive.Content> &
  React.ComponentPropsWithoutRef<typeof HoverCardPrimitive.Content> &
  &gt;({ className, align = "center", sideOffset = 4, ...props }, ref) => {
  &lt;HoverCardPrimitive.Content
    ref={ref}
    align={align}
    sideOffset={sideOffset}
    className={cn(
      "z-50 w-64 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out
    )}
    {...props}
  /&gt;
)

HoverCardContent.displayName = HoverCardPrimitive.Content.displayName

export { HoverCard, HoverCardTrigger, HoverCardContent }
```

File: client/src/components/ui/input-otp.tsx

```
import * as React from "react"
import { OTPInput, OTPInputContext } from "input-otp"
import { Dot } from "lucide-react"

import { cn } from "@/lib/utlils"

const InputOTP = React.forwardRef<
  React.ElementRef<typeof OTPInput> &
  React.ComponentPropsWithoutRef<typeof OTPInput>
>(({ className, containerClassName, ...props }, ref) => {
  <OTPInput
    ref={ref}
    containerClassName={cn(
      "flex items-center gap-2 has-[:disabled]:opacity-50",
      containerClassName
    )}
    className={cn("disabled:cursor-not-allowed", className)}
    {...props}
  />
})
InputOTP.displayName = "InputOTP"

const InputOTPGroup = React.forwardRef<
  React.ElementRef<"div"> &
  React.ComponentPropsWithoutRef<"div">
>(({ className, ...props }, ref) => {
  <div ref={ref} className={cn("flex items-center", className)} {...props} />
})
InputOTPGroup.displayName = "InputOTPGroup"

const InputOTPSlot = React.forwardRef<
  React.ElementRef<"div"> &
  React.ComponentPropsWithoutRef<"div"> & { index: number }
>(({ index, className, ...props }, ref) => {
  const inputOTPContext = React.useContext(OTPInputContext)
  const { char, hasFakeCaret, isActive } = inputOTPContext.slots[index]

  return (
    <div
      ref={ref}
      className={cn(
        "relative flex h-10 w-10 items-center justify-center border-y border-r border-input text-sm transition-all first:rounded-l-md first:border-l last:rounded-r",
        isActive & "z-10 ring-2 ring-ring ring-offset-background",
        className
      )}
      {...props}
    >
      {char}
      {hasFakeCaret & (
        <div className="pointer-events-none absolute inset-0 flex items-center justify-center">
          <div className="h-4 w-px animate-caret-blink bg-foreground duration-1000" />
        </div>
      )}
    </div>
  )
})
InputOTPSlot.displayName = "InputOTPSlot"

const InputOTPSeparator = React.forwardRef<
  React.ElementRef<"div"> &
  React.ComponentPropsWithoutRef<"div">
>(({ ...props }, ref) => {
  <div ref={ref} role="separator" {...props}>
    <Dot />
  </div>
})
InputOTPSeparator.displayName = "InputOTPSeparator"

export { InputOTP, InputOTPGroup, InputOTPSlot, InputOTPSeparator }
```

File: client/src/components/ui/input.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>>({
  className, type, ...props }, ref) => {
  // h-9 to match icon buttons and default buttons.
  return (
    <input
      type={type}
      className={cn(
        "flex h-9 w-full rounded-md border border-input bg-background px-3 py-2 text-base ring-offset-background file:border-0 file:bg-transparent file:text-
      className
    )}
    ref={ref}
    {...props}
  />
  )
}

Input.displayName = "Input"

export { Input }
```

File: client/src/components/ui/label.tsx

```
import * as React from "react"
import * as LabelPrimitive from "@radix-ui/react-label"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@lib/Utils"

const labelVariants = cva(
  "text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70"
)

const Label = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> & {
    VariantProps<typeof labelVariants>
  } >(( { className, ...props }, ref) => {
  <LabelPrimitive.Root
    ref={ref}
    className={cn(labelVariants(), className)}
    {...props}
  />
))
Label.displayName = LabelPrimitive.Root.displayName

export { Label }
```

File: client/src/components/ui/menubar.tsx

```
"use client"

import * as React from "react"
import * as MenubarPrimitive from "@radix-ui/react-menubar"
import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utils"

function MenubarMenu({
  ...props
}: React.ComponentProps<typeof MenubarPrimitive.Menu>): {
  return <MenubarPrimitive.Menu {...props} />;
}

function MenubarGroup({
  ...props
}: React.ComponentProps<typeof MenubarPrimitive.Group>): {
  return <MenubarPrimitive.Group {...props} />;
}

function MenubarPortal({
  ...props
}: React.ComponentProps<typeof MenubarPrimitive.Portal>): {
  return <MenubarPrimitive.Portal {...props} />;
}

function MenubarRadioGroup({
  ...props
}: React.ComponentProps<typeof MenubarPrimitive.RadioGroup>): {
  return <MenubarPrimitive.RadioGroup {...props} />;
}

function MenubarSub({
  ...props
}: React.ComponentProps<typeof MenubarPrimitive.Sub>): {
  return <MenubarPrimitive.Sub data-slot="menubar-sub" {...props} />;
}

const Menubar = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.Root>;,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Root> &
  <({ className, ...props }, ref) => (
    <MenubarPrimitive.Root
      ref={ref}
      className={cn(
        "flex h-10 items-center space-x-1 rounded-md border bg-background p-1",
        className
      )}
      {...props}
    />
  )
>() {
  Menubar.displayName = MenubarPrimitive.Root.displayName

  const MenubarTrigger = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.Trigger>;,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Trigger> &
    <({ className, ...props }, ref) => (
      <MenubarPrimitive.Trigger
        ref={ref}
        className={cn(
          "flex cursor-default select-none items-center rounded-sm px-3 py-1.5 text-sm font-medium outline-none focus:bg-accent focus:text-accent-foreground data-
          className
        )}
        {...props}
      />
    )
  >() {
    MenubarTrigger.displayName = MenubarPrimitive.Trigger.displayName

    const MenubarSubTrigger = React.forwardRef<
      React.ElementRef<typeof MenubarPrimitive.SubTrigger>;,
      React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubTrigger> & {
        inset?: boolean
      }
    >({ className, inset, children, ...props }, ref) => (
      <MenubarPrimitive.SubTrigger
        ref={ref}
        className={cn(
          "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]
          inset & & "pl-8",
          className
        )}
        {...props}
      >
        {children}
      </MenubarPrimitive.SubTrigger>
    )
    MenubarSubTrigger.displayName = MenubarPrimitive.SubTrigger.displayName

    const MenubarSubContent = React.forwardRef<
      React.ElementRef<typeof MenubarPrimitive.SubContent>;,
      React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubContent> &
      <({ className, ...props }, ref) => (
        <MenubarPrimitive.SubContent
          ref={ref}
          className={cn(
            "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground data-[state=open]:animate-in data-[state=closed]:animate-out

```

```

        className
      )}
      {...props}
    />
  ))
  MenubarSubContent.displayName = MenubarPrimitive.SubContent.displayName

  const MenubarContent = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.Content>,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Content>
  >((
    {
      { className, align = "start", alignOffset = -4, sideOffset = 8, ...props },
      ref
    } = > {
      <MenubarPrimitive.Portal>
        <MenubarPrimitive.Content
          ref={ref}
          align={align}
          alignOffset={alignOffset}
          sideOffset={sideOffset}
          className={cn(
            "z-50 min-w-[12rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",
            className
          )}
          {...props}
        />
      </MenubarPrimitive.Portal>
    }
  ))
  MenubarContent.displayName = MenubarPrimitive.Content.displayName

  const MenubarItem = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.Item>,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Item> & {
      inset?: boolean
    }
  >(({ className, inset, ...props }, ref) = > {
    <MenubarPrimitive.Item
      ref={ref}
      className={cn(
        "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none",
        inset & "pl-8",
        className
      )}
      {...props}
    />
  ))
  MenubarItem.displayName = MenubarPrimitive.Item.displayName

  const MenubarCheckboxItem = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.CheckboxItem>,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.CheckboxItem>
  >(({ className, children, checked, ...props }, ref) = > {
    <MenubarPrimitive.CheckboxItem
      ref={ref}
      className={cn(
        "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none",
        className
      )}
      checked={checked}
      {...props}
    >
      <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
        <MenubarPrimitive.ItemIndicator>
        <Check className="h-4 w-4" />
        </MenubarPrimitive.ItemIndicator>
      </span>
      {children}
    </MenubarPrimitive.CheckboxItem>
  ))
  MenubarCheckboxItem.displayName = MenubarPrimitive.CheckboxItem.displayName

  const MenubarRadioItem = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.RadioItem>,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.RadioItem>
  >(({ className, children, ...props }, ref) = > {
    <MenubarPrimitive.RadioItem
      ref={ref}
      className={cn(
        "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none",
        className
      )}
      {...props}
    >
      <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
        <MenubarPrimitive.ItemIndicator>
        <Circle className="h-2 w-2 fill-current" />
        </MenubarPrimitive.ItemIndicator>
      </span>
      {children}
    </MenubarPrimitive.RadioItem>
  ))
  MenubarRadioItem.displayName = MenubarPrimitive.RadioItem.displayName

  const MenubarLabel = React.forwardRef<
    React.ElementRef<typeof MenubarPrimitive.Label>,
    React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Label> & {
      inset?: boolean
    }
  >(({ className, inset, ...props }, ref) = > {

```

```

    <MenuPrimitive.Label
      ref={ref}
      className={cn(
        "px-2 py-1.5 text-sm font-semibold",
        inset & "pl-8",
        className
      )}
      {...props}
    />
  )
  MenuLabel.displayName = MenuPrimitive.Label.displayName

  const MenuSeparator = React.forwardRef(
    React.ElementRef<MenuPrimitive.Separator>,
    React.ComponentPropsWithoutRef<MenuPrimitive.Separator>
  )({ className, ...props }, ref) => (
    <MenuPrimitive.Separator
      ref={ref}
      className={cn("-mx-1 my-1 h-px bg-muted", className)}
      {...props}
    />
  )
  MenuSeparator.displayName = MenuPrimitive.Separator.displayName

  const MenuShortcut = ({
    className,
    ...props
  }): React.HTMLAttributes<HTMLSpanElement> => {
    return (
      <span
        className={cn(
          "ml-auto text-xs tracking-widest text-muted-foreground",
          className
        )}
        {...props}
      />
    )
  }
  MenuShortcut.displayName = "MenuShortcut"

  export {
    Menu,
    MenuMenu,
    MenuTrigger,
    MenuContent,
    MenuItem,
    MenuSeparator,
    MenuLabel,
    MenuCheckboxItem,
    MenuRadioGroup,
    MenuRadioItem,
    MenuPortal,
    MenuSubContent,
    MenuSubTrigger,
    MenuGroup,
    MenuSub,
    MenuShortcut,
  }

```

File: client/src/components/ui/navigation-menu.tsx

```
import * as React from "react"
import * as NavigationMenuPrimitive from "@radix-ui/react-navigation-menu"
import { cva } from "class-variance-authority"
import { ChevronDown } from "lucide-react"

import { cn } from "@/lib/utils"

const NavigationMenu = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Root>;,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Root> &
  &gt;({ className, children, ...props }, ref) => {
  <NavigationMenuPrimitive.Root
    ref={ref}
    className={cn(
      "relative z-10 flex max-w-max flex-1 items-center justify-center",
      className
    )}
    {...props}
  &gt;
    {children}
    <NavigationMenuViewport /&gt;
  </NavigationMenuPrimitive.Root>
)
NavigationMenu.displayName = NavigationMenuPrimitive.Root.displayName

const NavigationMenuList = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.List>;,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.List> &
  &gt;({ className, ...props }, ref) => {
  <NavigationMenuPrimitive.List
    ref={ref}
    className={cn(
      "group flex flex-1 list-none items-center justify-center space-x-1",
      className
    )}
    {...props}
  &gt;
  </&gt;
)
NavigationMenuList.displayName = NavigationMenuPrimitive.List.displayName

const NavigationMenuItem = NavigationMenuPrimitive.Item

const navigationMenuTriggerStyle = cva(
  "group inline-flex h-10 w-max items-center justify-center rounded-md bg-background px-4 py-2 text-sm font-medium transition-colors hover:bg-accent hover:text-accent-foreground"
)

const NavigationMenuTrigger = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Trigger>;,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Trigger> &
  &gt;({ className, children, ...props }, ref) => {
  <NavigationMenuPrimitive.Trigger
    ref={ref}
    className={cn(navigationMenuTriggerStyle(), "group", className)}
    {...props}
  &gt;
    {children}{" "}
    <ChevronDown
      className="relative top-[1px] ml-1 h-3 w-3 transition duration-200 group-data-[state=open]:rotate-180"
      aria-hidden="true"
    /&gt;
  </NavigationMenuPrimitive.Trigger>
)
NavigationMenuTrigger.displayName = NavigationMenuPrimitive.Trigger.displayName

const NavigationMenuContent = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Content>;,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Content> &
  &gt;({ className, ...props }, ref) => {
  <NavigationMenuPrimitive.Content
    ref={ref}
    className={cn(
      "left-0 top-0 w-full data-[motion^=from-]:animate-in data-[motion^=to-]:animate-out data-[motion^=from-]:fade-in data-[motion^=to-]:fade-out data-[motion=from-]:opacity-0 data-[motion=to-]:opacity-100",
      className
    )}
    {...props}
  &gt;
  </&gt;
)
NavigationMenuContent.displayName = NavigationMenuPrimitive.Content.displayName

const NavigationMenuLink = NavigationMenuPrimitive.Link

const NavigationMenuViewport = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Viewport>;,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Viewport> &
  &gt;({ className, ...props }, ref) => {
  <div className={cn("absolute left-0 top-full flex justify-center")} &gt;
    <NavigationMenuPrimitive.Viewport
      ref={ref}
      className={cn(
        "origin-top-center relative mt-1.5 h-[var(--radix-navigation-menu-viewport-height)] w-full overflow-hidden rounded-md border bg-popover text-popover-foreground",
        className
      )}
      {...props}
    /&gt;
  </div>
)
NavigationMenuViewport.displayName =
```

```

    NavigationMenuPrimitive.Viewport.displayName

const NavigationMenuIndicator = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Indicator>,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Indicator>
>(({ className, ...props }, ref) => {
  <NavigationMenuPrimitive.Indicator
    ref={ref}
    className={cn(
      "top-full z-[1] flex h-1.5 items-end justify-center overflow-hidden data-[state=visible]:animate-in data-[state=hidden]:animate-out data-[state=hidden]:animate-pulse",
      className
    )}
    {...props}
  />
  <div className="relative top-[60%] h-2 w-2 rotate-45 rounded-tl-sm bg-border shadow-md" />
  /NavigationMenuPrimitive.Indicator
)

NavigationMenuIndicator.displayName =
  NavigationMenuPrimitive.Indicator.displayName

export {
  navigationMenuTriggerStyle,
  NavigationMenu,
  NavigationMenuList,
  NavigationMenuItem,
  NavigationMenuContent,
  NavigationMenuTrigger,
  NavigationMenuLink,
  NavigationMenuIndicator,
  NavigationMenuViewport,
}

```

File: client/src/components/ui/pagination.tsx

```
import * as React from "react"
import { ChevronLeft, ChevronRight, MoreHorizontal } from "lucide-react"

import { cn } from "@/lib/utlils"
import { ButtonProps, buttonVariants } from "@/components/ui/button"

const Pagination = ({ className, ...props }: React.ComponentProps<"nav">) => {
  <nav
    role="navigation"
    aria-label="pagination"
    className={cn("mx-auto flex w-full justify-center", className)}
    {...props}
  />
)
Pagination.displayName = "Pagination"

const PaginationContent = React.forwardRef<
  HTMLUListElement,
  React.ComponentProps<"ul">
>(({ className, ...props }, ref) => {
  <ul
    ref={ref}
    className={cn("flex flex-row items-center gap-1", className)}
    {...props}
  />
)
PaginationContent.displayName = "PaginationContent"

const PaginationItem = React.forwardRef<
  HTMLLIElement,
  React.ComponentProps<"li">
>(({ className, ...props }, ref) => {
  <li ref={ref} className={cn("", className)} {...props} />
)
PaginationItem.displayName = "PaginationItem"

type PaginationLinkProps = {
  isActive?: boolean
} & Pick<ButtonProps, "size"> &
  React.ComponentProps<"a">

const PaginationLink = ({
  className,
  isActive,
  size = "icon",
  ...props
}: PaginationLinkProps) => {
  <a
    aria-current={isActive ? "page" : undefined}
    className={cn(
      buttonVariants({
        variant: isActive ? "outline" : "ghost",
        size,
      }),
      className
    )}
    {...props}
  />
)
PaginationLink.displayName = "PaginationLink"

const PaginationPrevious = ({
  className,
  ...props
}: React.ComponentProps<typeof PaginationLink>) => {
  <PaginationLink
    aria-label="Go to previous page"
    size="default"
    className={cn("gap-1 pl-2.5", className)}
    {...props}
  >
    <ChevronLeft className="h-4 w-4" />
    <span>Previous</span>
  </PaginationLink>
)
PaginationPrevious.displayName = "PaginationPrevious"

const PaginationNext = ({
  className,
  ...props
}: React.ComponentProps<typeof PaginationLink>) => {
  <PaginationLink
    aria-label="Go to next page"
    size="default"
    className={cn("gap-1 pr-2.5", className)}
    {...props}
  >
    <span>Next</span>
    <ChevronRight className="h-4 w-4" />
  </PaginationLink>
)
PaginationNext.displayName = "PaginationNext"

const PaginationEllipsis = ({
  className,
  ...props
}: React.ComponentProps<"span">) => {
  <span
    aria-label="More"
    {...props}
  />
)
PaginationEllipsis.displayName = "PaginationEllipsis"

```

```

    </span>
    aria-hidden
    className={cn("flex h-9 w-9 items-center justify-center", className)}
    {...props}
  </>
  <MoreHorizontal className="h-4 w-4" />
  <span className="sr-only">More pages</span>
  </span>
)
PaginationEllipsis.displayName = "PaginationEllipsis"

export {
  Pagination,
  PaginationContent,
  PaginationEllipsis,
  PaginationItem,
  PaginationLink,
  PaginationNext,
  PaginationPrevious,
}

```

File: client/src/components/ui/popover.tsx

```
import * as React from "react"
import * as PopoverPrimitive from "@radix-ui/react-popover"

import { cn } from "@lib/utls"

const Popover = PopoverPrimitive.Root

const PopoverTrigger = PopoverPrimitive.Trigger

const PopoverContent = React.forwardRef<
  React.ElementRef<typeof PopoverPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof PopoverPrimitive.Content>
>(( { className, align = "center", sideOffset = 4, ...props }, ref) => {
  <PopoverPrimitive.Portal>
    <PopoverPrimitive.Content
      ref={ref}
      align={align}
      sideOffset={sideOffset}
      className={cn(
        "z-50 w-72 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out",
        className
      )}
      {...props}
    />
  </PopoverPrimitive.Portal>
))
PopoverContent.displayName = PopoverPrimitive.Content.displayName

export { Popover, PopoverTrigger, PopoverContent }
```

File: client/src/components/ui/progress.tsx

```

use client"

import * as React from "react"
import * as ProgressPrimitive from "@radix-ui/react-progress"

import { cn } from "@lib/utls"

const Progress = React.forwardRef<
  React.ElementRef<typeof ProgressPrimitive.Root>;
  React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>;
>(({ className, value, ...props }, ref) => {
  <ProgressPrimitive.Root
    ref={ref}
    className={cn(
      "relative h-4 w-full overflow-hidden rounded-full bg-secondary",
      className
    )}
    {...props}
  >
    <ProgressPrimitive.Indicator
      className="h-full w-full flex-1 bg-primary transition-all"
      style={{ transform: `translateX(-${100 - (value || 0)}%)` }}
    />
  </ProgressPrimitive.Root>
  >
  </>
  <Progress.displayName = ProgressPrimitive.Root.displayName

export { Progress }

```

File: client/src/components/ui/radio-group.tsx

```
import * as React from "react"
import * as RadioGroupPrimitive from "@radix-ui/react-radio-group"
import { Circle } from "lucide-react"

import { cn } from "@lib/utlils"

const RadioGroup = React.forwardRef<
  React.ElementRef<typeof RadioGroupPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Root>
>(({ className, ...props }, ref) => {
  return (
    <RadioGroupPrimitive.Root
      className={cn("grid gap-2", className)}
      {...props}
      ref={ref}
    />
  )
})
RadioGroup.displayName = RadioGroupPrimitive.Root.displayName

const RadioGroupItem = React.forwardRef<
  React.ElementRef<typeof RadioGroupPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Item>
>(({ className, ...props }, ref) => {
  return (
    <RadioGroupPrimitive.Item
      ref={ref}
      className={cn(
        "aspect-square h-4 w-4 rounded-full border border-primary text-primary ring-offset-background focus:outline-none focus-visible:ring-2 focus-visible:ring-primary",
        className
      )}
      {...props}
    />
    <RadioGroupPrimitive.Indicator className="flex items-center justify-center">
      <Circle className="h-2.5 w-2.5 fill-current text-current" />
    </RadioGroupPrimitive.Indicator>
  )
})
RadioGroupItem.displayName = RadioGroupPrimitive.Item.displayName

export { RadioGroup, RadioGroupItem }
```

File: client/src/components/ui/resizable.tsx

```
"use client"

import { GripVertical } from "lucide-react"
import * as ResizablePrimitive from "react-resizable-panels"

import { cn } from "@/lib/utils"

const ResizablePanelGroup = ({
  className,
  ...props
}: React.ComponentProps<typeof ResizablePrimitive.PanelGroup>) => {
  <ResizablePrimitive.PanelGroup
    className={cn(
      "flex h-full w-full data-[panel-group-direction=vertical]:flex-col",
      className
    )}
    {...props}
  />
}

const ResizablePanel = ResizablePrimitive.Panel

const ResizableHandle = ({
  withHandle,
  className,
  ...props
}: React.ComponentProps<typeof ResizablePrimitive.PanelResizeHandle> & {
  withHandle?: boolean
}) => {
  <ResizablePrimitive.PanelResizeHandle
    className={cn(
      "relative flex w-px items-center justify-center bg-border after:absolute after:inset-y-0 after:left-1/2 after:w-1 after:-translate-x-1/2 focus-visible:outline",
      className
    )}
    {...props}
  >
    {withHandle & & (
      <div className="z-10 flex h-4 w-3 items-center justify-center rounded-sm border bg-border">
        <GripVertical className="h-2.5 w-2.5" />
      </div>
    )}
  </ResizablePrimitive.PanelResizeHandle>
}

export { ResizablePanelGroup, ResizablePanel, ResizableHandle }
```

File: client/src/components/ui/scroll-area.tsx

```
import * as React from "react"
import * as ScrollAreaPrimitive from "@radix-ui/react-scroll-area"

import { cn } from "@lib/utlils"

const ScrollArea = React.forwardRef<
  React.ElementRef<typeof ScrollAreaPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root> &
  &gt;({ className, children, ...props }, ref) => {
  <ScrollAreaPrimitive.Root
    ref={ref}
    className={cn("relative overflow-hidden", className)}
    {...props}>
    <ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">
      {children}
    </ScrollAreaPrimitive.Viewport>
    <ScrollBar />
    <ScrollAreaPrimitive.Corner />
  </ScrollAreaPrimitive.Root>
)
ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName

const ScrollBar = React.forwardRef<
  React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,
  React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar> &
  &gt;({ className, orientation = "vertical", ...props }, ref) => {
  <ScrollAreaPrimitive.ScrollAreaScrollbar
    ref={ref}
    orientation={orientation}
    className={cn(
      "flex touch-none select-none transition-colors",
      orientation === "vertical" & "h-full w-2.5 border-l border-l-transparent p-[1px]",
      orientation === "horizontal" & "h-2.5 flex-col border-t border-t-transparent p-[1px]",
      className
    )}>
    <ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />
  </ScrollAreaPrimitive.ScrollAreaScrollbar>
)
ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName

export { ScrollArea, ScrollBar }
```

File: client/src/components/ui/select.tsx

```
"use client"

import * as React from "react"
import * as SelectPrimitive from "@radix-ui/react-select"
import { Check, ChevronDown, ChevronUp } from "lucide-react"

import { cn } from "@/lib/utlils"

const Select = SelectPrimitive.Root

const SelectGroup = SelectPrimitive.Group

const SelectValue = SelectPrimitive.Value

const SelectTrigger = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Trigger>;
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Trigger>;
>(({ className, children, ...props }, ref) => {
  <SelectPrimitive.Trigger
    ref={ref}
    className={cn(
      "flex h-9 w-full items-center justify-between rounded-md border border-input bg-background px-3 py-2 text-sm ring-offset-background data-[placeholder]:text-muted-foreground focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",
      className
    )}
    {...props}
  >
    {children}
    <SelectPrimitive.Icon asChild>
      <ChevronDown className="h-4 w-4 opacity-50" />
    </SelectPrimitive.Icon>
  </SelectPrimitive.Trigger>
))
SelectTrigger.displayName = SelectPrimitive.Trigger.displayName

const SelectScrollUpButton = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.ScrollUpButton>;
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollUpButton>;
>(({ className, ...props }, ref) => {
  <SelectPrimitive.ScrollUpButton
    ref={ref}
    className={cn(
      "flex cursor-default items-center justify-center py-1",
      className
    )}
    {...props}
  >
    <ChevronUp className="h-4 w-4" />
  </SelectPrimitive.ScrollUpButton>
))
SelectScrollUpButton.displayName = SelectPrimitive.ScrollUpButton.displayName

const SelectScrollDownButton = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.ScrollDownButton>;
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollDownButton>;
>(({ className, ...props }, ref) => {
  <SelectPrimitive.ScrollDownButton
    ref={ref}
    className={cn(
      "flex cursor-default items-center justify-center py-1",
      className
    )}
    {...props}
  >
    <ChevronDown className="h-4 w-4" />
  </SelectPrimitive.ScrollDownButton>
))
SelectScrollDownButton.displayName =
  SelectPrimitive.ScrollDownButton.displayName

const SelectContent = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Content>;
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Content>;
>(({ className, children, position = "popper", ...props }, ref) => {
  <SelectPrimitive.Portal>
    <SelectPrimitive.Content
      ref={ref}
      className={cn(
        "relative z-50 max-h-[--radix-select-content-available-height] min-w-[8rem] overflow-y-auto overflow-x-hidden rounded-md border bg-popover text-popover-foreground",
        position === "popper" &&
        "data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=top]:-translate-y-1",
        className
      )}
      position={position}
      {...props}
    >
      <SelectScrollUpButton />
      <SelectPrimitive.Viewport
        className={cn(
          "p-1",
          position === "popper" &&
          "h-[var(--radix-select-trigger-height)] w-full min-w-[var(--radix-select-trigger-width)]"
        )}
      >
        {children}
      </SelectPrimitive.Viewport>
      <SelectScrollDownButton />
    </SelectPrimitive.Content>
  </SelectPrimitive.Portal>
))
SelectContent.displayName = SelectPrimitive.Content.displayName
```

```

    </SelectPrimitive.Portal>
  )
  SelectContent.displayName = SelectPrimitive.Content.displayName

  const SelectLabel = React.forwardRef(
    React.ElementRef<typeof SelectPrimitive.Label>,
    React.ComponentPropsWithoutRef<typeof SelectPrimitive.Label>,
    &gt;({ className, ...props }, ref) => {
      <SelectPrimitive.Label
        ref={ref}
        className={cn("py-1.5 pl-8 pr-2 text-sm font-semibold", className)}
        {...props}
      />
    )
  )
  SelectLabel.displayName = SelectPrimitive.Label.displayName

  const SelectItem = React.forwardRef(
    React.ElementRef<typeof SelectPrimitive.Item>,
    React.ComponentPropsWithoutRef<typeof SelectPrimitive.Item>,
    &gt;({ className, children, ...props }, ref) => {
      <SelectPrimitive.Item
        ref={ref}
        className={cn(
          "relative flex w-full cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground",
          className
        )}
        {...props}
      >
        <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
          <SelectPrimitive.ItemIndicator>
          <Check className="h-4 w-4" />
          </SelectPrimitive.ItemIndicator>
        </span>
        <SelectPrimitive.ItemText>{children}</SelectPrimitive.ItemText>
      </SelectPrimitive.Item>
    )
  )
  SelectItem.displayName = SelectPrimitive.Item.displayName

  const SelectSeparator = React.forwardRef(
    React.ElementRef<typeof SelectPrimitive.Separator>,
    React.ComponentPropsWithoutRef<typeof SelectPrimitive.Separator>,
    &gt;({ className, ...props }, ref) => {
      <SelectPrimitive.Separator
        ref={ref}
        className={cn("-mx-1 my-1 h-px bg-muted", className)}
        {...props}
      />
    )
  )
  SelectSeparator.displayName = SelectPrimitive.Separator.displayName

  export {
    Select,
    SelectGroup,
    SelectValue,
    SelectTrigger,
    SelectContent,
    SelectLabel,
    SelectItem,
    SelectSeparator,
    SelectScrollUpButton,
    SelectScrollDownButton,
  }

```

File: client/src/components/ui/separator.tsx

```
import * as React from "react"
import * as SeparatorPrimitive from "@radix-ui/react-separator"

import { cn } from "@lib/utils"

const Separator = React.forwardRef<
  React.ElementRef<typeof SeparatorPrimitive.Root>;
  React.ComponentPropsWithoutRef<typeof SeparatorPrimitive.Root>;
>({
  (
    { className, orientation = "horizontal", decorative = true, ...props },
    ref
  ) => {
    <SeparatorPrimitive.Root
      ref={ref}
      decorative={decorative}
      orientation={orientation}
      className={cn(
        "shrink-0 bg-border",
        orientation === "horizontal" ? "h-[1px] w-full" : "h-full w-[1px]",
        className
      )}
      {...props}
    />
  )
})
Separator.displayName = SeparatorPrimitive.Root.displayName

export { Separator }
```

File: client/src/components/ui/sheet.tsx

```
"use client"

import * as React from "react"
import * as SheetPrimitive from "@radix-ui/react-dialog"
import { cva, type VariantProps } from "class-variance-authority"
import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const Sheet = SheetPrimitive.Root

const SheetTrigger = SheetPrimitive.Trigger

const SheetClose = SheetPrimitive.Close

const SheetPortal = SheetPrimitive.Portal

const SheetOverlay = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof SheetPrimitive.Overlay>
>(({ className, ...props }, ref) => {
  <SheetPrimitive.Overlay
    className={cn(
      "fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",
      className
    )}
    {...props}
    ref={ref}
  />
})
SheetOverlay.displayName = SheetPrimitive.Overlay.displayName

const sheetVariants = cva(
  "fixed z-50 gap-4 bg-background p-6 shadow-lg transition ease-in-out data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:duration-300 data-[state=open]:duration-500",
  {
    variants: {
      side: {
        top: "inset-x-0 top-0 border-b data-[state=closed]:slide-out-to-top data-[state=open]:slide-in-from-top",
        bottom:
          "inset-x-0 bottom-0 border-t data-[state=closed]:slide-out-to-bottom data-[state=open]:slide-in-from-bottom",
        left: "inset-y-0 left-0 h-full w-3/4 border-r data-[state=closed]:slide-out-to-left data-[state=open]:slide-in-from-left sm:max-w-sm",
        right:
          "inset-y-0 right-0 h-full w-3/4 border-l data-[state=closed]:slide-out-to-right data-[state=open]:slide-in-from-right sm:max-w-sm",
      },
    },
    defaultVariants: {
      side: "right",
    },
  }
)

interface SheetContentProps
  extends React.ComponentPropsWithoutRef<typeof SheetPrimitive.Content>,
    VariantProps<typeof sheetVariants> {}

const SheetContent = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Content>,
  SheetContentProps
>(({ side = "right", className, children, ...props }, ref) => {
  <SheetPortal>
    <SheetOverlay />
    <SheetPrimitive.Content
      ref={ref}
      className={cn(sheetVariants({ side }), className)}
      {...props}
    />
    <div>
      <div>
        <SheetPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2" />
        <span>
          <X className="h-4 w-4" />
          </span>
        </div>
        <SheetPrimitive.Close />
      </div>
      <SheetPrimitive.Content />
    </div>
  </SheetPortal>
</div>
SheetContent.displayName = SheetPrimitive.Content.displayName

const SheetHeader = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => {
  <div
    className={cn(
      "flex flex-col space-y-2 text-center sm:text-left",
      className
    )}
    {...props}
  />
}
SheetHeader.displayName = "SheetHeader"

const SheetFooter = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => {
  <div
    className={cn(
      "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
      className
    )}
    {...props}
  />
}
```

```

        className
      )}
      {...props}
    />
  )
  SheetFooter.displayName = "SheetFooter"

  const SheetTitle = React.forwardRef<
    React.ElementRef<typeof SheetPrimitive.Title>;
    React.ComponentPropsWithoutRef<typeof SheetPrimitive.Title>
  >(({ className, ...props }, ref) => {
    <SheetPrimitive.Title
      ref={ref}
      className={cn("text-lg font-semibold text-foreground", className)}
      {...props}
    />
  ))
  SheetTitle.displayName = SheetPrimitive.Title.displayName

  const SheetDescription = React.forwardRef<
    React.ElementRef<typeof SheetPrimitive.Description>;
    React.ComponentPropsWithoutRef<typeof SheetPrimitive.Description>
  >(({ className, ...props }, ref) => {
    <SheetPrimitive.Description
      ref={ref}
      className={cn("text-sm text-muted-foreground", className)}
      {...props}
    />
  ))
  SheetDescription.displayName = SheetPrimitive.Description.displayName

  export {
    Sheet,
    SheetPortal,
    SheetOverlay,
    SheetTrigger,
    SheetClose,
    SheetContent,
    SheetHeader,
    SheetFooter,
    SheetTitle,
    SheetDescription,
  }

```

File: client/src/components/ui/sidebar.tsx

```
"use client"

import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, VariantProps } from "class-variance-authority"
import { PanelLeftIcon } from "lucide-react"

import { useIsMobile } from "@hooks/use-mobile"
import { cn } from "@lib/utils"
import { Button } from "@components/ui/button"
import { Input } from "@components/ui/input"
import { Separator } from "@components/ui/separator"
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
} from "@components/ui/sheet"
import { Skeleton } from "@components/ui/skeleton"
import {
  Tooltip,
  TooltipContent,
  TooltipProvider,
  TooltipTrigger,
} from "@components/ui/tooltip"

const SIDEBAR_COOKIE_NAME = "sidebar_state"
const SIDEBAR_COOKIE_MAX_AGE = 60 * 60 * 24 * 7
const SIDEBAR_WIDTH = "16rem"
const SIDEBAR_WIDTH_MOBILE = "18rem"
const SIDEBAR_WIDTH_ICON = "3rem"
const SIDEBAR_KEYBOARD_SHORTCUT = "b"

type SidebarContextProps = {
  state: "expanded" | "collapsed"
  open: boolean
  setOpen: (open: boolean) => void
  openMobile: boolean
  setOpenMobile: (open: boolean) => void
  isMobile: boolean
  toggleSidebar: () => void
}

const SidebarContext = React.createContext<SidebarContextProps | null>(null)

function useSidebar() {
  const context = React.useContext(SidebarContext)
  if (!context) {
    throw new Error("useSidebar must be used within a SidebarProvider.")
  }

  return context
}

function SidebarProvider({
  defaultOpen = true,
  open: openProp,
  onOpenChange: setOpenProp,
  className,
  style,
  children,
  ...props
}: React.ComponentProps<div> & {
  defaultOpen?: boolean
  open?: boolean
  onOpenChange?: (open: boolean) => void
}) {
  const isMobile = useIsMobile()
  const [openMobile, setOpenMobile] = React.useState(false)

  // This is the internal state of the sidebar.
  // We use openProp and setOpenProp for control from outside the component.
  const [_open, _setOpen] = React.useState(defaultOpen)
  const open = openProp ?? _open
  const setOpen = React.useCallback(
    (value: boolean | ((value: boolean) => boolean)) => {
      const openState = typeof value === "function" ? value(open) : value
      if (setOpenProp) {
        setOpenProp(openState)
      } else {
        _setOpen(openState)
      }
    },
    [setOpenProp, open]
  )

  // This sets the cookie to keep the sidebar state.
  document.cookie = `${SIDEBAR_COOKIE_NAME}=${openState}; path=/; max-age=${SIDEBAR_COOKIE_MAX_AGE}`

  const toggleSidebar = React.useCallback(() => {
    return isMobile ? setOpenMobile((open) => !open) : setOpen((open) => !open)
  }, [isMobile, setOpen, setOpenMobile])

  // Adds a keyboard shortcut to toggle the sidebar.
  React.useEffect(() => {

```

```

const handleKeyDown = (event: KeyboardEvent) => {
  if (
    event.key === SIDEBAR_KEYBOARD_SHORTCUT &&
    (event.metaKey || event.ctrlKey)
  ) {
    event.preventDefault()
    toggleSidebar()
  }
}

window.addEventListener("keydown", handleKeyDown)
return () => window.removeEventListener("keydown", handleKeyDown)
}, [toggleSidebar])

// We add a state so that we can do data-state="expanded" or "collapsed".
// This makes it easier to style the sidebar with Tailwind classes.
const state = open ? "expanded" : "collapsed"

const contextValue = React.useMemo<SidebarContextProps>(() => ({
  state,
  open,
  setOpen,
  isMobile,
  openMobile,
  setOpenMobile,
  toggleSidebar,
}), [state, open, setOpen, isMobile, openMobile, setOpenMobile, toggleSidebar])

return (
  <SidebarContext.Provider value={contextValue}>
    <TooltipProvider delayDuration={0}>
      <div
        data-slot="sidebar-wrapper"
        style={{
          "--sidebar-width": SIDEBAR_WIDTH,
          "--sidebar-width-icon": SIDEBAR_WIDTH_ICON,
          ...style,
        }} as React.CSSProperties
        className={cn(
          "group/sidebar-wrapper has-data-[variant=inset]:bg-sidebar flex min-h-svh w-full",
          className
        )}
        {...props}
      >
        {children}
      </div>
    </TooltipProvider>
  </SidebarContext.Provider>
)
}

function Sidebar({
  side = "left",
  variant = "sidebar",
  collapsible = "offcanvas",
  className,
  children,
  ...props
}: React.ComponentProps<div> & {
  side?: "left" | "right"
  variant?: "sidebar" | "floating" | "inset"
  collapsible?: "offcanvas" | "icon" | "none"
}) {
  const { isMobile, state, openMobile, setOpenMobile } = useSidebar()

  if (collapsible === "none") {
    return (
      <div
        data-slot="sidebar"
        className={cn(
          "bg-sidebar text-sidebar-foreground flex h-full w-[var(--sidebar-width)] flex-col",
          className
        )}
        {...props}
      >
        {children}
      </div>
    )
  }

  if (isMobile) {
    return (
      <Sheet open={openMobile} onOpenChange={setOpenMobile} {...props}>
        <SheetContent
          data-sidebar="sidebar"
          data-slot="sidebar"
          data-mobile="true"
          className="bg-sidebar text-sidebar-foreground w-[var(--sidebar-width)] p-0 [&button]:hidden"
          style={{
            "--sidebar-width": SIDEBAR_WIDTH_MOBILE,
          }} as React.CSSProperties
        >
          <div>
            <div>
              {children}
            </div>
            <SheetHeader className="sr-only">

```

```

        <SheetTitle><Sidebar</SheetTitle>
        <SheetDescription>Displays the mobile sidebar.</SheetDescription>
        <SheetHeader>
        <div className="flex h-full w-full flex-col">{children}</div>
        <SheetContent>
        </Sheet>
    )
}

return (
    <div
        className="group peer text-sidebar-foreground hidden md:block"
        data-state={state}
        data-collapsible={state === "collapsed" ? collapsible : ""}
        data-variant={variant}
        data-side={side}
        data-slot="sidebar"
    >
        { /* This is what handles the sidebar gap on desktop */ }
        <div
            data-slot="sidebar-gap"
            className={cn(
                "relative w-[var(--sidebar-width)] bg-transparent transition-[width] duration-200 ease-linear",
                "group-data-[collapsible=offcanvas]:w-0",
                "group-data-[side=right]:rotate-180",
                variant === "floating" || variant === "inset"
                    ? "group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)+var(--spacing-4))]"
                    : "group-data-[collapsible=icon]:w-[var(--sidebar-width-icon)]"
            )}
        >
        </div>
        <div
            data-slot="sidebar-container"
            className={cn(
                "fixed inset-y-0 z-10 hidden h-svh w-[var(--sidebar-width)] transition-[left,right,width] duration-200 ease-linear md:flex",
                side === "left"
                    ? "left-0 group-data-[collapsible=offcanvas]:left-[calc(var(--sidebar-width)*-1)]"
                    : "right-0 group-data-[collapsible=offcanvas]:right-[calc(var(--sidebar-width)*-1)]",
                // Adjust the padding for floating and inset variants.
                variant === "floating" || variant === "inset"
                    ? "p-2 group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)+var(--spacing-4)+2px)]"
                    : "group-data-[collapsible=icon]:w-[var(--sidebar-width-icon)] group-data-[side=left]:border-r group-data-[side=right]:border-l",
            )}
            {...props}
        >
            <div
                data-sidebar="sidebar"
                data-slot="sidebar-inner"
                className="bg-sidebar group-data-[variant=floating]:border-sidebar-border flex h-full w-full flex-col group-data-[variant=floating]:rounded-lg group-
            >
                {children}
            </div>
        </div>
    </div>
)

function SidebarTrigger({
    className,
    onClick,
    ...props
}): React.ComponentProps<typeof Button> {
    const { toggleSidebar } = useSidebar()

    return (
        <Button
            data-sidebar="trigger"
            data-slot="sidebar-trigger"
            variant="ghost"
            size="icon"
            className={cn("h-7 w-7", className)}
            onClick={(event) => {
                onClick?.(event)
                toggleSidebar()
            }}
            {...props}
        >
            <PanelLeftIcon />
            <span className="sr-only">Toggle Sidebar</span>
            </Button>
        </div>
    )
}

function SidebarRail({ className, ...props }: React.ComponentProps<"button"> {
    const { toggleSidebar } = useSidebar()

    // Note: Tailwind v3.4 doesn't support "in-" selectors. So the rail won't work perfectly.
    return (
        <button
            data-sidebar="rail"
            data-slot="sidebar-rail"
            aria-label="Toggle Sidebar"
            tabIndex={-1}
            onClick={toggleSidebar}
            title="Toggle Sidebar"
            className={cn(
                "hover:after:bg-sidebar-border absolute inset-y-0 z-20 hidden w-4 -translate-x-1/2 transition-all ease-linear group-data-[side=left]:-right-4 group-da-
            "in-data-[side=left]:cursor-w-resize in-data-[side=right]:cursor-e-resize",
            "[[data-side=left][data-state=collapsed]&]:cursor-e-resize [[data-side=right][data-state=collapsed]&]:cursor-w-resize",
            "hover:group-data-[collapsible=offcanvas]:bg-sidebar group-data-[collapsible=offcanvas]:translate-x-0 group-data-[collapsible=offcanvas]:after:left-fu-
            "[[data-side=left][data-collapsible=offcanvas]&]:-right-2",
        )}
    )
}

```

```

        "[data-side=right][data-collapsible=offcanvas]&#x26;:-left-2",
        className
    )}
    {...props}
  /&gt;
)
}

function SidebarInset({ className, ...props }: React.ComponentProps&#x26;"main"&#x26;){
  return (
    &#x26;main
    data-slot="sidebar-inset"
    className={cn(
      "bg-background relative flex w-full flex-1 flex-col",
      "md:peer-data-[variant=inset]:m-2 md:peer-data-[variant=inset]:ml-0 md:peer-data-[variant=inset]:rounded-xl md:peer-data-[variant=inset]:shadow-sm md:peer-data-[variant=inset]:z-10",
      className
    )}
    {...props}
  /&gt;
)
}

function SidebarInput({
  className,
  ...props
}: React.ComponentProps&#x26;"input"&#x26;){
  return (
    &#x26;input
    data-slot="sidebar-input"
    data-sidebar="input"
    className={cn("bg-background h-8 w-full shadow-none", className)}
    {...props}
  /&gt;
)
}

function SidebarHeader({ className, ...props }: React.ComponentProps&#x26;"div"&#x26;){
  return (
    &#x26;div
    data-slot="sidebar-header"
    data-sidebar="header"
    className={cn("flex flex-col gap-2 p-2", className)}
    {...props}
  /&gt;
)
}

function SidebarFooter({ className, ...props }: React.ComponentProps&#x26;"div"&#x26;){
  return (
    &#x26;div
    data-slot="sidebar-footer"
    data-sidebar="footer"
    className={cn("flex flex-col gap-2 p-2", className)}
    {...props}
  /&gt;
)
}

function SidebarSeparator({
  className,
  ...props
}: React.ComponentProps&#x26;"div"&#x26;){
  return (
    &#x26;div
    data-slot="sidebar-separator"
    data-sidebar="separator"
    className={cn("bg-sidebar-border mx-2 w-auto", className)}
    {...props}
  /&gt;
)
}

function SidebarContent({ className, ...props }: React.ComponentProps&#x26;"div"&#x26;){
  return (
    &#x26;div
    data-slot="sidebar-content"
    data-sidebar="content"
    className={cn(
      "flex min-h-0 flex-1 flex-col gap-2 overflow-auto group-data-[collapsible=icon]:overflow-hidden",
      className
    )}
    {...props}
  /&gt;
)
}

function SidebarGroup({ className, ...props }: React.ComponentProps&#x26;"div"&#x26;){
  return (
    &#x26;div
    data-slot="sidebar-group"
    data-sidebar="group"
    className={cn("relative flex w-full min-w-0 flex-col p-2", className)}
    {...props}
  /&gt;
)
}

function SidebarGroupLabel({
  className,
  asChild = false,
  ...props
}) {
  return (
    &#x26;div
    data-slot="sidebar-group-label"
    data-sidebar="group-label"
    className={cn("flex flex-col gap-2 p-2", className)}
    {...props}
  /&gt;
)
}

```

```

}: React.ComponentProps<div> & { asChild?: boolean }) {
  const Comp = asChild ? Slot : "div"

  return (
    <Comp
      data-slot="sidebar-group-label"
      data-sidebar="group-label"
      className={cn(
        "text-sidebar-foreground/70 ring-sidebar-ring flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium outline-hidden transition-[margin,opa",
        "group-data-[collapsible=icon]:-mt-8 group-data-[collapsible=icon]:opacity-0",
        className
      )}
      {...props}
    />
  )
}

function SidebarGroupAction({
  className,
  asChild = false,
  ...props
}: React.ComponentProps<button> & { asChild?: boolean }) {
  const Comp = asChild ? Slot : "button"

  return (
    <Comp
      data-slot="sidebar-group-action"
      data-sidebar="group-action"
      className={cn(
        "text-sidebar-foreground ring-sidebar-ring hover:bg-sidebar-accent hover:text-sidebar-accent-foreground absolute top-3.5 right-3 flex aspect-square w-",
        // Increases the hit area of the button on mobile.
        "after:absolute after:-inset-2 md:after:hidden",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
}

function SidebarGroupContent({
  className,
  ...props
}: React.ComponentProps<div>) {
  return (
    <div
      data-slot="sidebar-group-content"
      data-sidebar="group-content"
      className={cn("w-full text-sm", className)}
      {...props}
    />
  )
}

function SidebarMenu({ className, ...props }: React.ComponentProps<ul>) {
  return (
    <ul
      data-slot="sidebar-menu"
      data-sidebar="menu"
      className={cn("flex w-full min-w-0 flex-col gap-1", className)}
      {...props}
    />
  )
}

function SidebarMenuItem({ className, ...props }: React.ComponentProps<li>) {
  return (
    <li
      data-slot="sidebar-menu-item"
      data-sidebar="menu-item"
      className={cn("group/menu-item relative", className)}
      {...props}
    />
  )
}

const sidebarMenuButtonVariants = cva(
  "peer/menu-button flex w-full items-center gap-2 overflow-hidden rounded-md p-2 text-left text-sm outline-hidden ring-sidebar-ring transition-[width,height,flex-b",
  {
    variants: {
      variant: {
        default: "hover:bg-sidebar-accent hover:text-sidebar-accent-foreground",
        outline: "bg-background shadow-[0_0_1px_hsl(var(--sidebar-border))] hover:bg-sidebar-accent hover:text-sidebar-accent-foreground hover:shadow-[0_0_1px_hsl",
      },
      size: {
        default: "h-8 text-sm",
        sm: "h-7 text-xs",
        lg: "h-12 text-sm group-data-[collapsible=icon]:p-0!",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
)

function SidebarMenuButton({
  asChild = false,
  isActive = false,

```

```

variant = "default",
size = "default",
tooltip,
className,
...props
}: React.ComponentProps<"button"> & {
  asChild?: boolean
  isActive?: boolean
  tooltip?: string | React.ComponentProps<typeof TooltipContent>
} & VariantProps<typeof sidebarMenuButtonVariants> {
  const Comp = asChild ? Slot : "button"
  const { isMobile, state } = useSidebar()

  const button = (
    <Comp
      data-slot="sidebar-menu-button"
      data-sidebar="menu-button"
      data-size={size}
      data-active={isActive}
      className={cn(sidebarMenuButtonVariants({ variant, size })), className}
      {...props}
    />
  )

  if (!tooltip) {
    return button
  }

  if (typeof tooltip === "string") {
    tooltip = {
      children: tooltip,
    }
  }

  return (
    <Tooltip>
      <TooltipTrigger asChild>{button}</TooltipTrigger>
      <TooltipContent
        side="right"
        align="center"
        hidden={state !== "collapsed" || isMobile}
        {...tooltip}
      />
    </Tooltip>
  )
}

function SidebarMenuAction({
  className,
  asChild = false,
  showOnHover = false,
  ...props
}: React.ComponentProps<"button"> & {
  asChild?: boolean
  showOnHover?: boolean
}) {
  const Comp = asChild ? Slot : "button"

  return (
    <Comp
      data-slot="sidebar-menu-action"
      data-sidebar="menu-action"
      className={cn(
        "text-sidebar-foreground ring-sidebar-ring hover:bg-sidebar-accent hover:text-sidebar-accent-foreground peer-hover/menu-button:text-sidebar-accent-foreground",
        // Increases the hit area of the button on mobile.
        "after:absolute after:-inset-2 md:after:hidden",
        "peer-data-[size=sm]/menu-button:top-1",
        "peer-data-[size=default]/menu-button:top-1.5",
        "peer-data-[size=lg]/menu-button:top-2.5",
        "group-data-[collapsible=icon]:hidden",
        showOnHover & "peer-data-[active=true]/menu-button:text-sidebar-accent-foreground group-focus-within/menu-item:opacity-100 group-hover/menu-item:opacity-100 data-",
        className
      )}
      {...props}
    />
  )
}

function SidebarMenuBadge({
  className,
  ...props
}: React.ComponentProps<"div">) {
  return (
    <div
      data-slot="sidebar-menu-badge"
      data-sidebar="menu-badge"
      className={cn(
        "text-sidebar-foreground pointer-events-none absolute right-1 flex h-5 min-w-5 items-center justify-center rounded-md px-1 text-xs font-medium tabular-nums",
        "peer-hover/menu-button:text-sidebar-accent-foreground peer-data-[active=true]/menu-button:text-sidebar-accent-foreground",
        "peer-data-[size=sm]/menu-button:top-1",
        "peer-data-[size=default]/menu-button:top-1.5",
        "peer-data-[size=lg]/menu-button:top-2.5",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
}

function SidebarMenuSkeleton({

```

```

    className,
    showIcon = false,
    ...props
  }: React.ComponentProps<div> & {
    showIcon?: boolean
  }) {
    // Random width between 50 to 90%.
    const width = React.useMemo(() => {
      return `${Math.floor(Math.random() * 40) + 50}%`
    }, [])

    return (
      <div
        data-slot="sidebar-menu-skeleton"
        data-sidebar="menu-skeleton"
        className={cn("flex h-8 items-center gap-2 rounded-md px-2", className)}
        {...props}
      >
        {showIcon & & {
          <Skeleton
            className="size-4 rounded-md"
            data-sidebar="menu-skeleton-icon"
          />
        }
        <Skeleton
          className="h-4 max-w-[var(--skeleton-width)] flex-1"
          data-sidebar="menu-skeleton-text"
          style={
            {
              "--skeleton-width": width,
            } as React.CSSProperties
          }
        />
      </div>
    )
  }

function SidebarMenuSub({ className, ...props }: React.ComponentProps<ul>) {
  return (
    <ul
      data-slot="sidebar-menu-sub"
      data-sidebar="menu-sub"
      className={cn(
        "border-sidebar-border mx-3.5 flex min-w-0 translate-x-px flex-col gap-1 border-l px-2.5 py-0.5",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
}

function SidebarMenuSubItem({
  className,
  ...props
}: React.ComponentProps<li>) {
  return (
    <li
      data-slot="sidebar-menu-sub-item"
      data-sidebar="menu-sub-item"
      className={cn("group/menu-sub-item relative", className)}
      {...props}
    />
  )
}

function SidebarMenuSubButton({
  asChild = false,
  size = "md",
  isActive = false,
  className,
  ...props
}: React.ComponentProps<a> & {
  asChild?: boolean
  size?: "sm" | "md"
  isActive?: boolean
}) {
  const Comp = asChild ? Slot : "a"

  return (
    <Comp
      data-slot="sidebar-menu-sub-button"
      data-sidebar="menu-sub-button"
      data-size={size}
      data-active={isActive}
      className={cn(
        "text-sidebar-foreground ring-sidebar-ring hover:bg-sidebar-accent hover:text-sidebar-accent-foreground active:bg-sidebar-accent active:text-sidebar-a",
        "data-[active=true]:bg-sidebar-accent data-[active=true]:text-sidebar-accent-foreground",
        size === "sm" & & "text-xs",
        size === "md" & & "text-sm",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
}

export {
  Sidebar,
  SidebarContent,

```

```
SidebarFooter,  
SidebarGroup,  
SidebarGroupAction,  
SidebarGroupContent,  
SidebarGroupLabel,  
SidebarHeader,  
SidebarInput,  
SidebarInset,  
SidebarMenu,  
SidebarMenuAction,  
SidebarMenuBadge,  
SidebarMenuButton,  
SidebarMenuItem,  
SidebarMenuSkeleton,  
SidebarMenuSub,  
SidebarMenuSubButton,  
SidebarMenuSubItem,  
SidebarProvider,  
SidebarRail,  
SidebarSeparator,  
SidebarTrigger,  
useSidebar,  
}
```

File: client/src/components/ui/skeleton.tsx

```
import { cn } from "@/lib/utils"

function Skeleton({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) {
  return (
    <div
      className={cn("animate-pulse rounded-md bg-muted", className)}
      {...props}
    />
  )
}

export { Skeleton }
```

File: client/src/components/ui/slider.tsx

```
import * as React from "react"
import * as SliderPrimitive from "@radix-ui/react-slider"

import { cn } from "@/lib/utils"

const Slider = React.forwardRef<
  React.ElementRef<typeof SliderPrimitive.Root>;
  React.ComponentPropsWithoutRef<typeof SliderPrimitive.Root>
>(({ className, ...props }, ref) => {
  <SliderPrimitive.Root
    ref={ref}
    className={cn(
      "relative flex w-full touch-none select-none items-center",
      className
    )}
    {...props}
  >
    <SliderPrimitive.Track className="relative h-2 w-full grow overflow-hidden rounded-full bg-secondary">
      <SliderPrimitive.Range className="absolute h-full bg-primary" />
    </SliderPrimitive.Track>
    <SliderPrimitive.Thumb className="block h-5 w-5 rounded-full border-2 border-primary bg-background ring-offset-background transition-colors focus-visible:outline focus-visible:outline-2 focus-visible:outline-offset-2" />
  </SliderPrimitive.Root>
</>
))
Slider.displayName = SliderPrimitive.Root.displayName

export { Slider }
```

File: client/src/components/ui/switch.tsx

```
export * as React from "react"
import * as SwitchPrimitives from "@radix-ui/react-switch"

import { cn } from "@/lib/utils"

const Switch = React.forwardRef<
  React.ElementRef<typeof SwitchPrimitives.Root>;,
  React.ComponentPropsWithoutRef<typeof SwitchPrimitives.Root>;>(
  &gt;({ className, ...props }, ref) => {
    <SwitchPrimitives.Root
      className={cn(
        "peer inline-flex h-6 w-11 shrink-0 cursor-pointer items-center rounded-full border-2 border-transparent transition-colors focus-visible:outline-none fo
        className
      )}
      {...props}
      ref={ref}
    </SwitchPrimitives.Thumb
      className={cn(
        "pointer-events-none block h-5 w-5 rounded-full bg-background shadow-lg ring-0 transition-transform data-[state=checked]:translate-x-5 data-[state=unch
      )}
    />
    </SwitchPrimitives.Root>
  )}
  Switch.displayName = SwitchPrimitives.Root.displayName

export { Switch }
```

File: client/src/components/ui/table.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Table = React.forwardRef<
  HTMLTableElement,
  React.HTMLAttributes<HTMLTableElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;div className="relative w-full overflow-auto"&gt;
    &lt;table
      ref={ref}
      className={cn("w-full caption-bottom text-sm", className)}
      {...props}
    /&gt;
    &lt;/div>
  &gt;
}

Table.displayName = "Table"

const TableHeader = React.forwardRef<
  HTMLTableSectionElement,
  React.HTMLAttributes<HTMLTableSectionElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;thead ref={ref} className={cn("&_tr:border-b", className)} {...props} /&gt;
}

TableHeader.displayName = "TableHeader"

const TableBody = React.forwardRef<
  HTMLTableSectionElement,
  React.HTMLAttributes<HTMLTableSectionElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;tbody
    ref={ref}
    className={cn("&_tr:last-child:border-0", className)}
    {...props}
  /&gt;
}

TableBody.displayName = "TableBody"

const TableFooter = React.forwardRef<
  HTMLTableSectionElement,
  React.HTMLAttributes<HTMLTableSectionElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;tfoot
    ref={ref}
    className={cn(
      "border-t bg-muted/50 font-medium [&_&tr]:last:border-b-0",
      className
    )}
    {...props}
  /&gt;
}

TableFooter.displayName = "TableFooter"

const TableRow = React.forwardRef<
  HTMLTableRowElement,
  React.HTMLAttributes<HTMLTableRowElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;tr
    ref={ref}
    className={cn(
      "border-b transition-colors hover:bg-muted/50 data-[state=selected]:bg-muted",
      className
    )}
    {...props}
  /&gt;
}

TableRow.displayName = "TableRow"

const TableHead = React.forwardRef<
  HTMLTableCellElement,
  React.ThHTMLAttributes<HTMLTableCellElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;th
    ref={ref}
    className={cn(
      "h-12 px-4 text-left align-middle font-medium text-muted-foreground [&:has([role=checkbox])]:pr-0",
      className
    )}
    {...props}
  /&gt;
}

TableHead.displayName = "TableHead"

const TableCell = React.forwardRef<
  HTMLTableCellElement,
  React.TdHTMLAttributes<HTMLTableCellElement> &
  &gt;({ className, ...props }, ref) => {
  &lt;td
    ref={ref}
    className={cn("p-4 align-middle [&:has([role=checkbox])]:pr-0", className)}
    {...props}
  /&gt;
}

TableCell.displayName = "TableCell"

const TableCaption = React.forwardRef<
  HTMLTableCaptionElement,
```

```
      React.HTMLAttributes<HTMLTableCaptionElement>;
    &gt;(({ className, ...props }, ref) => {
      <caption
        ref={ref}
        className={cn("mt-4 text-sm text-muted-foreground", className)}
        {...props}
      />
    })
    TableCaption.displayName = "TableCaption"

    export {
      Table,
      TableHeader,
      TableBody,
      TableFooter,
      TableHead,
      TableRow,
      TableCell,
      TableCaption,
    }
  }
}
```

File: client/src/components/ui/tabs.tsx

```
import * as React from "react"
import * as TabsPrimitive from "@radix-ui/react-tabs"

import { cn } from "@/lib/utils"

const Tabs = TabsPrimitive.Root

const TabsList = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.List>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.List>
>(({ className, ...props }, ref) => {
  <TabsPrimitive.List
    ref={ref}
    className={cn(
      "inline-flex h-10 items-center justify-center rounded-md bg-muted p-1 text-muted-foreground",
      className
    )}
    {...props}
  />
})
TabsList.displayName = TabsPrimitive.List.displayName

const TabsTrigger = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.Trigger>
>(({ className, ...props }, ref) => {
  <TabsPrimitive.Trigger
    ref={ref}
    className={cn(
      "inline-flex items-center justify-center whitespace-nowrap rounded-sm px-3 py-1.5 text-sm font-medium ring-offset-background transition-all focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2",
      className
    )}
    {...props}
  />
})
TabsTrigger.displayName = TabsPrimitive.Trigger.displayName

const TabsContent = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.Content>
>(({ className, ...props }, ref) => {
  <TabsPrimitive.Content
    ref={ref}
    className={cn(
      "mt-2 ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2",
      className
    )}
    {...props}
  />
})
TabsContent.displayName = TabsPrimitive.Content.displayName

export { Tabs, TabsList, TabsTrigger, TabsContent }
```

File: client/src/components/ui/textarea.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Textarea = React.forwardRef<
  HTMLTextAreaElement,
  React.ComponentProps<"textarea">
>(({ className, ...props }, ref) => {
  return (
    <textarea
      className={cn(
        "flex min-h-[80px] w-full rounded-md border border-input bg-background px-3 py-2 text-base ring-offset-background placeholder:text-muted-foreground fo
        className
      )}
      ref={ref}
      {...props}
    />
  )
})
Textarea.displayName = "Textarea"

export { Textarea }
```

File: client/src/components/ui/toast.tsx

```
import * as React from "react"
import * as ToastPrimitives from "@radix-ui/react-toast"
import { cva, type VariantProps } from "class-variance-authority"
import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const ToastProvider = ToastPrimitives.Provider

const ToastViewport = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Viewport>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Viewport> &
  &gt;({ className, ...props }, ref) => {
  <ToastPrimitives.Viewport
    ref={ref}
    className={cn(
      "fixed top-0 z-[100] flex max-h-screen w-full flex-col-reverse p-4 sm:bottom-0 sm:right-0 sm:top-auto sm:flex-col md:max-w-[420px]",
      className
    )}
    {...props}
  />
}

ToastViewport.displayName = ToastPrimitives.Viewport.displayName

const toastVariants = cva(
  "group pointer-events-auto relative flex w-full items-center justify-between space-x-4 overflow-hidden rounded-md border p-6 pr-8 shadow-lg transition-all duration-100",
  {
    variants: {
      variant: {
        default: "border bg-background text-foreground",
        destructive:
          "destructive group border-destructive bg-destructive text-destructive-foreground",
      },
    },
    defaultVariants: {
      variant: "default",
    },
  }
)

const Toast = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Root>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Root> &
  VariantProps<typeof toastVariants>
  &gt;({ className, variant, ...props }, ref) => {
  return (
    <ToastPrimitives.Root
      ref={ref}
      className={cn(toastVariants({ variant }), className)}
      {...props}
    />
  )
}

Toast.displayName = ToastPrimitives.Root.displayName

const ToastAction = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Action>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Action>
  &gt;({ className, ...props }, ref) => {
  <ToastPrimitives.Action
    ref={ref}
    className={cn(
      "inline-flex h-8 shrink-0 items-center justify-center rounded-md border bg-transparent px-3 text-sm font-medium ring-offset-background transition-colors focus-visible:outline focus-visible:outline-2 focus-visible:outline-offset-2",
      className
    )}
    {...props}
  />
}

ToastAction.displayName = ToastPrimitives.Action.displayName

const ToastClose = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Close>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Close>
  &gt;({ className, ...props }, ref) => {
  <ToastPrimitives.Close
    ref={ref}
    className={cn(
      "absolute right-2 top-2 rounded-md p-1 text-foreground/50 opacity-0 transition-opacity hover:text-foreground focus:opacity-100 focus:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2",
      className
    )}
    toast-close=""
    {...props}
  />
  <X className="h-4 w-4" />
  </ToastPrimitives.Close>
}

ToastClose.displayName = ToastPrimitives.Close.displayName

const ToastTitle = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Title>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Title>
  &gt;({ className, ...props }, ref) => {
  <ToastPrimitives.Title
    ref={ref}
    className={cn("text-sm font-semibold", className)}
    {...props}
  />
}
```

```

))
ToastTitle.displayName = ToastPrimitives.Title.displayName

const ToastDescription = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Description>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Description>
>(({ className, ...props }, ref) => {
  <ToastPrimitives.Description
    ref={ref}
    className={cn("text-sm opacity-90", className)}
    {...props}
  />
))
ToastDescription.displayName = ToastPrimitives.Description.displayName

type ToastProps = React.ComponentPropsWithoutRef<typeof Toast>;

type ToastActionElement = React.ReactElement<typeof ToastAction>;

export {
  type ToastProps,
  type ToastActionElement,
  ToastProvider,
  ToastViewport,
  Toast,
  ToastTitle,
  ToastDescription,
  ToastClose,
  ToastAction,
}

```

File: client/src/components/ui/toaster.tsx

```
import { useToast } from "@hooks/use-toast"
import {
  Toast,
  ToastClose,
  ToastDescription,
  ToastProvider,
  ToastTitle,
  ToastViewport,
} from "@components/ui/toast"

export function Toaster() {
  const { toasts } = useToast()

  return (
    <ToastProvider>
      {toasts.map(function ({ id, title, description, action, ...props }) {
        return (
          <Toast key={id} {...props}>
            <div className="grid gap-1">
              {title && <ToastTitle>{title}</ToastTitle>}
              {description && (
                <ToastDescription>{description}</ToastDescription>
              )}
            </div>
            {action}
            <ToastClose />
          </Toast>
        )
      })}
    </ToastViewport />
    </ToastProvider>
  )
}
```

File: client/src/components/ui/toggle-group.tsx

```
"use client"

import * as React from "react"
import * as ToggleGroupPrimitive from "@radix-ui/react-toggle-group"
import { type VariantProps } from "class-variance-authority"

import { cn } from "@lib/utils"
import { toggleVariants } from "@components/ui/toggle"

const ToggleGroupContext = React.createContext<{
  VariantProps:<typeof toggleVariants>;
}>({
  size: "default",
  variant: "default",
})

const ToggleGroup = React.forwardRef<
  React.ElementRef<typeof ToggleGroupPrimitive.Root>;
  React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Root>; &
  VariantProps<typeof toggleVariants>;
>(({ className, variant, size, children, ...props }, ref) => {
  <ToggleGroupPrimitive.Root
    ref={ref}
    className={cn("flex items-center justify-center gap-1", className)}
    {...props}
  >
    <ToggleGroupContext.Provider value={{ variant, size }}>
      {children}
    </ToggleGroupContext.Provider>
  </ToggleGroupPrimitive.Root>
})

ToggleGroup.displayName = ToggleGroupPrimitive.Root.displayName

const ToggleGroupItem = React.forwardRef<
  React.ElementRef<typeof ToggleGroupPrimitive.Item>;
  React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Item>; &
  VariantProps<typeof toggleVariants>;
>(({ className, children, variant, size, ...props }, ref) => {
  const context = React.useContext(ToggleGroupContext)

  return (
    <ToggleGroupPrimitive.Item
      ref={ref}
      className={cn(
        toggleVariants({
          variant: context.variant || variant,
          size: context.size || size,
        }),
        className
      )}
      {...props}
    >
      {children}
    </ToggleGroupPrimitive.Item>
  )
})

ToggleGroupItem.displayName = ToggleGroupPrimitive.Item.displayName

export { ToggleGroup, ToggleGroupItem }
```

File: client/src/components/ui/toggle.tsx

```
import * as React from "react"
import * as TogglePrimitive from "@radix-ui/react-toggle"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const toggleVariants = cva(
  "inline-flex items-center justify-center rounded-md text-sm font-medium ring-offset-background transition-colors hover:bg-muted hover:text-muted-foreground",
  {
    variants: {
      variant: {
        default: "bg-transparent",
        outline:
          "border border-input bg-transparent hover:bg-accent hover:text-accent-foreground",
      },
      size: {
        default: "h-10 px-3 min-w-10",
        sm: "h-9 px-2.5 min-w-9",
        lg: "h-11 px-5 min-w-11",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
)

const Toggle = React.forwardRef<
  React.ElementRef<typeof TogglePrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof TogglePrimitive.Root> &
    VariantProps<typeof toggleVariants>
>(({ className, variant, size, ...props }, ref) => {
  <TogglePrimitive.Root
    ref={ref}
    className={cn(toggleVariants({ variant, size, className }))}
    {...props}
  />
))

Toggle.displayName = TogglePrimitive.Root.displayName

export { Toggle, toggleVariants }
```

File: client/src/components/ui/tooltip.tsx

```
"use client"

import * as React from "react"
import * as TooltipPrimitive from "@radix-ui/react-tooltip"

import { cn } from "@lib/Utils"

const TooltipProvider = TooltipPrimitive.Provider

const Tooltip = TooltipPrimitive.Root

const TooltipTrigger = TooltipPrimitive.Trigger

const TooltipContent = React.forwardRef<
  React.ElementRef<typeof TooltipPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content>
>(({ className, sideOffset = 4, ...props }, ref) => {
  <TooltipPrimitive.Content
    ref={ref}
    sideOffset={sideOffset}
    className={cn(
      "z-50 overflow-hidden rounded-md border bg-popover px-3 py-1.5 text-sm text-popover-foreground shadow-md animate-in fade-in-0 zoom-in-95 data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=closed]:zoom-out-95 data-[delay-duration=0]:data-[state=closed]:duration-0"
    )}
    {...props}
  />
})
TooltipContent.displayName = TooltipPrimitive.Content.displayName

export { Tooltip, TooltipTrigger, TooltipContent, TooltipProvider }
```

File: client/src/hooks/use-mobile.tsx

```
import * as React from "react"

const MOBILE_BREAKPOINT = 768

export function useIsMobile() {
  const [isMobile, setIsMobile] = React.useState<boolean | undefined>(undefined)

  React.useEffect(() => {
    const mql = window.matchMedia(`(max-width: ${MOBILE_BREAKPOINT - 1}px)`)
    const onChange = () => {
      setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    }
    mql.addEventListener("change", onChange)
    setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    return () => mql.removeEventListener("change", onChange)
  }, [])

  return !isMobile
}
```

File: client/src/hooks/use-toast.ts

```
import * as React from "react"

import type {
  ToastActionElement,
  ToastProps,
} from "@components/ui/toast"

const TOAST_LIMIT = 1
const TOAST_REMOVE_DELAY = 1000000

type ToasterToast = ToastProps & {
  id: string
  title?: React.ReactNode
  description?: React.ReactNode
  action?: ToastActionElement
}

const actionTypes = {
  ADD_TOAST: "ADD_TOAST",
  UPDATE_TOAST: "UPDATE_TOAST",
  DISMISS_TOAST: "DISMISS_TOAST",
  REMOVE_TOAST: "REMOVE_TOAST",
} as const

let count = 0

function genId() {
  count = (count + 1) % Number.MAX_SAFE_INTEGER
  return count.toString()
}

type ActionType = typeof actionTypes

type Action =
  | {
      type: ActionType["ADD_TOAST"]
      toast: ToasterToast
    }
  | {
      type: ActionType["UPDATE_TOAST"]
      toast: Partial<ToasterToast>
    }
  | {
      type: ActionType["DISMISS_TOAST"]
      toastId?: ToasterToast["id"]
    }
  | {
      type: ActionType["REMOVE_TOAST"]
      toastId?: ToasterToast["id"]
    }

interface State {
  toasts: ToasterToast[]
}

const toastTimeouts = new Map<string, ReturnType<typeof setTimeout>>>()

const addToRemoveQueue = (toastId: string) => {
  if (toastTimeouts.has(toastId)) {
    return
  }

  const timeout = setTimeout(() => {
    toastTimeouts.delete(toastId)
    dispatch({
      type: "REMOVE_TOAST",
      toastId: toastId,
    })
  }, TOAST_REMOVE_DELAY)

  toastTimeouts.set(toastId, timeout)
}

export const reducer = (state: State, action: Action): State => {
  switch (action.type) {
    case "ADD_TOAST":
      return {
        ...state,
        toasts: [action.toast, ...state.toasts].slice(0, TOAST_LIMIT),
      }

    case "UPDATE_TOAST":
      return {
        ...state,
        toasts: state.toasts.map((t) => {
          t.id === action.toast.id ? { ...t, ...action.toast } : t
        }),
      }

    case "DISMISS_TOAST": {
      const { toastId } = action

      // ! Side effects ! - This could be extracted into a dismissToast() action,
      // but I'll keep it here for simplicity
      if (toastId) {
        addToRemoveQueue(toastId)
      } else {

```

```

        state.toasts.forEach((toast) => {
            addToRemoveQueue(toast.id)
        })
    }

    return {
        ...state,
        toasts: state.toasts.map((t) => {
            t.id === toastId || toastId === undefined
                ? {
                    ...t,
                    open: false,
                }
                : t
            },
        ),
    }
}

case "REMOVE_TOAST":
    if (action.toastId === undefined) {
        return {
            ...state,
            toasts: [],
        }
    }
    return {
        ...state,
        toasts: state.toasts.filter((t) => t.id !== action.toastId),
    }
}

const listeners: Array<(state: State) => void> = []

let memoryState: State = { toasts: [] }

function dispatch(action: Action) {
    memoryState = reducer(memoryState, action)
    listeners.forEach((listener) => {
        listener(memoryState)
    })
}

type Toast = Omit<ToasterToast, "id">

function toast({ ...props }: Toast) {
    const id = genId()

    const update = (props: ToasterToast) => {
        dispatch({
            type: "UPDATE_TOAST",
            toast: { ...props, id },
        })
    }

    const dismiss = () => dispatch({ type: "DISMISS_TOAST", toastId: id })

    dispatch({
        type: "ADD_TOAST",
        toast: {
            ...props,
            id,
            open: true,
            onOpenChange: (open) => {
                if (!open) dismiss()
            },
        },
    })

    return {
        id: id,
        dismiss,
        update,
    }
}

function useToast() {
    const [state, setState] = React.useState<State>(memoryState)

    React.useEffect(() => {
        listeners.push(setState)
        return () => {
            const index = listeners.indexOf(setState)
            if (index > -1) {
                listeners.splice(index, 1)
            }
        }
    }, [state])

    return {
        ...state,
        toast,
        dismiss: (toastId?: string) => dispatch({ type: "DISMISS_TOAST", toastId }),
    }
}

export { useToast, toast }

```

File: client/src/hooks/useAuth.ts

```
import { useQuery } from "@tanstack/react-query";
import type { User } from "@shared/schema";

export function useAuth() {
  const { data: user, isLoading } = useQuery<User | null>({
    queryKey: ["/api/auth/user"],
    retry: false,
  });

  return {
    user,
    isLoading,
    isAuthenticated: !!user,
    isAdmin: !!user?.isAdmin,
  };
}
```

File: client/src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* LIGHT MODE */
:root {
  --button-outline: rgba(0,0,0, .10);
  --badge-outline: rgba(0,0,0, .05);
  --opaque-button-border-intensity: -8;
  --elevate-1: rgba(0,0,0, .03);
  --elevate-2: rgba(0,0,0, .08);
  --background: 210 5% 98%;
  --foreground: 210 6% 12%;
  --border: 210 5% 90%;
  --card: 210 5% 96%;
  --card-foreground: 210 6% 12%;
  --card-border: 210 5% 92%;
  --sidebar: 210 5% 94%;
  --sidebar-foreground: 210 6% 14%;
  --sidebar-border: 210 5% 88%;
  --sidebar-primary: 217 91% 45%;
  --sidebar-primary-foreground: 217 91% 98%;
  --sidebar-accent: 210 6% 88%;
  --sidebar-accent-foreground: 210 6% 14%;
  --sidebar-ring: 217 91% 45%;
  --popover: 210 5% 92%;
  --popover-foreground: 210 6% 12%;
  --popover-border: 210 5% 86%;
  --primary: 217 91% 45%;
  --primary-foreground: 217 91% 98%;
  --secondary: 210 6% 88%;
  --secondary-foreground: 210 6% 14%;
  --muted: 210 6% 90%;
  --muted-foreground: 210 6% 30%;
  --accent: 210 8% 92%;
  --accent-foreground: 210 8% 14%;
  --destructive: 0 84% 48%;
  --destructive-foreground: 0 84% 98%;
  --input: 210 6% 70%;
  --ring: 217 91% 45%;
  --chart-1: 217 91% 42%;
  --chart-2: 186 75% 38%;
  --chart-3: 280 70% 45%;
  --chart-4: 32 95% 44%;
  --chart-5: 142 76% 36%;
  --font-sans: Inter, system-ui, sans-serif;
  --font-serif: Georgia, serif;
  --font-mono: Menlo, monospace;
  --radius: .5rem;
  --shadow-2xs: 0px 2px 0px 0px hsl(210 5% 12% / 0.00);
  --shadow-xs: 0px 2px 0px 0px hsl(210 5% 12% / 0.00);
  --shadow-sm: 0px 2px 0px 0px hsl(210 5% 12% / 0.00), 0px 1px 2px -1px hsl(210 5% 12% / 0.00);
  --shadow: 0px 2px 0px 0px hsl(210 5% 12% / 0.00), 0px 1px 2px -1px hsl(210 5% 12% / 0.00);
  --shadow-md: 0px 2px 0px 0px hsl(210 5% 12% / 0.00), 0px 2px 4px -1px hsl(210 5% 12% / 0.00);
  --shadow-lg: 0px 2px 0px 0px hsl(210 5% 12% / 0.00), 0px 4px 6px -1px hsl(210 5% 12% / 0.00);
  --shadow-xl: 0px 2px 0px 0px hsl(210 5% 12% / 0.00), 0px 8px 10px -1px hsl(210 5% 12% / 0.00);
  --shadow-2xl: 0px 2px 0px 0px hsl(210 5% 12% / 0.00);
  --tracking-normal: 0em;
  --spacing: 0.25rem;
}

/* Fallback for older browsers */
--sidebar-primary-border: hsl(var(--sidebar-primary));
--sidebar-primary-border: hsl(from hsl(var(--sidebar-primary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--sidebar-accent-border: hsl(var(--sidebar-accent));
--sidebar-accent-border: hsl(from hsl(var(--sidebar-accent)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--primary-border: hsl(var(--primary));
--primary-border: hsl(from hsl(var(--primary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--secondary-border: hsl(var(--secondary));
--secondary-border: hsl(from hsl(var(--secondary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--muted-border: hsl(var(--muted));
--muted-border: hsl(from hsl(var(--muted)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--accent-border: hsl(var(--accent));
--accent-border: hsl(from hsl(var(--accent)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--destructive-border: hsl(var(--destructive));
--destructive-border: hsl(from hsl(var(--destructive)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);
}

.dark {
  --button-outline: rgba(255,255,255, .10);
  --badge-outline: rgba(255,255,255, .05);
  --opaque-button-border-intensity: 9;
  --elevate-1: rgba(255,255,255, .04);
  --elevate-2: rgba(255,255,255, .09);
  --background: 210 5% 8%;
  --foreground: 210 5% 96%;
}
```

```

--border: 210 5% 18%;
--card: 210 5% 10%;
--card-foreground: 210 5% 96%;
--card-border: 210 5% 14%;
--sidebar: 210 5% 12%;
--sidebar-foreground: 210 5% 94%;
--sidebar-border: 210 5% 16%;
--sidebar-primary: 217 91% 55%;
--sidebar-primary-foreground: 217 91% 98%;
--sidebar-accent: 210 6% 16%;
--sidebar-accent-foreground: 210 6% 94%;
--sidebar-ring: 217 91% 55%;
--popover: 210 5% 14%;
--popover-foreground: 210 5% 96%;
--popover-border: 210 5% 18%;
--primary: 217 91% 55%;
--primary-foreground: 217 91% 98%;
--secondary: 210 6% 18%;
--secondary-foreground: 210 6% 94%;
--muted: 210 6% 16%;
--muted-foreground: 210 6% 70%;
--accent: 210 8% 14%;
--accent-foreground: 210 8% 94%;
--destructive: 0 84% 52%;
--destructive-foreground: 0 84% 98%;
--input: 210 6% 30%;
--ring: 217 91% 55%;
--chart-1: 217 91% 65%;
--chart-2: 186 75% 62%;
--chart-3: 280 70% 68%;
--chart-4: 32 95% 68%;
--chart-5: 142 76% 64%;
--shadow-2xs: 0px 2px 0px 0px hsl(210 5% 96% / 0.00);
--shadow-xs: 0px 2px 0px 0px hsl(210 5% 96% / 0.00);
--shadow-sm: 0px 2px 0px 0px hsl(210 5% 96% / 0.00), 0px 1px 2px -1px hsl(210 5% 96% / 0.00);
--shadow: 0px 2px 0px 0px hsl(210 5% 96% / 0.00), 0px 1px 2px -1px hsl(210 5% 96% / 0.00);
--shadow-md: 0px 2px 0px 0px hsl(210 5% 96% / 0.00), 0px 2px 4px -1px hsl(210 5% 96% / 0.00);
--shadow-lg: 0px 2px 0px 0px hsl(210 5% 96% / 0.00), 0px 4px 6px -1px hsl(210 5% 96% / 0.00);
--shadow-xl: 0px 2px 0px 0px hsl(210 5% 96% / 0.00), 0px 8px 10px -1px hsl(210 5% 96% / 0.00);
--shadow-2xl: 0px 2px 0px 0px hsl(210 5% 96% / 0.00);

/* Fallback for older browsers */
--sidebar-primary-border: hsl(var(--sidebar-primary));
--sidebar-primary-border: hsl(from hsl(var(--sidebar-primary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--sidebar-accent-border: hsl(var(--sidebar-accent));
--sidebar-accent-border: hsl(from hsl(var(--sidebar-accent)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--primary-border: hsl(var(--primary));
--primary-border: hsl(from hsl(var(--primary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--secondary-border: hsl(var(--secondary));
--secondary-border: hsl(from hsl(var(--secondary)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--muted-border: hsl(var(--muted));
--muted-border: hsl(from hsl(var(--muted)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--accent-border: hsl(var(--accent));
--accent-border: hsl(from hsl(var(--accent)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);

/* Fallback for older browsers */
--destructive-border: hsl(var(--destructive));
--destructive-border: hsl(from hsl(var(--destructive)) h s calc(1 + var(--opaque-button-border-intensity)) / alpha);
}

@layer base {
  * {
    @apply border-border;
  }

  body {
    @apply font-sans antialiased bg-background text-foreground;
  }
}

/**
 * Using the elevate system.
 * Automatic contrast adjustment.
 *
 * <element className="hover-elevate" />
 * <element className="active-elevate-2" />
 *
 * // Using the tailwind utility when a data attribute is "on"
 * <element className="toggle-elevate data-[state=on]:toggle-elevated" />
 * // Or manually controlling the toggle state
 * <element className="toggle-elevate toggle-elevated" />
 *
 * Elevation systems have to handle many states.
 * - not-hovered, vs. hovered vs. active (three mutually exclusive states)
 * - toggled or not
 * - focused or not (this is not handled with these utilities)
 *
 * Even without handling focused or not, this is six possible combinations that
 * need to be distinguished from eachother visually.
 */
@layer utilities {

```

```

/* Hide ugly search cancel button in Chrome until we can style it properly */
input[type="search"]::-webkit-search-cancel-button {
  @apply hidden;
}

/* Placeholder styling for contentEditable div */
[contenteditable][data-placeholder]:empty::before {
  content: attr(data-placeholder);
  color: hsl(var(--muted-foreground));
  pointer-events: none;
}

/* .no-default-hover-elevate/no-default-active-elevate is an escape hatch so consumers of
 * buttons/badges can remove the automatic brightness adjustment on interactions
 * and program their own. */
.no-default-hover-elevate {}

.no-default-active-elevate {}

/**
 * Toggleable backgrounds go behind the content. Hoverable/active goes on top.
 * This way they can stack/compound. Both will overlap the parent's borders!
 * So borders will be automatically adjusted both on toggle, and hover/active,
 * and they will be compounded.
 */
.toggle-elevate::before,
.toggle-elevate-2::before {
  content: "";
  pointer-events: none;
  position: absolute;
  inset: 0px;
  /*border-radius: inherit; match rounded corners */
  border-radius: inherit;
  z-index: -1;
  /* sits behind content but above backdrop */
}

.toggle-elevate.toggle-elevated::before {
  background-color: var(--elevate-2);
}

/* If there's a 1px border, adjust the inset so that it covers that parent's border */
.border.toggle-elevate::before {
  inset: -1px;
}

/* Does not work on elements with overflow:hidden! */
.hover-elevate:not(.no-default-hover-elevate),
.active-elevate:not(.no-default-active-elevate),
.hover-elevate-2:not(.no-default-hover-elevate),
.active-elevate-2:not(.no-default-active-elevate) {
  position: relative;
  z-index: 0;
}

.hover-elevate:not(.no-default-hover-elevate)::after,
.active-elevate:not(.no-default-active-elevate)::after,
.hover-elevate-2:not(.no-default-hover-elevate)::after,
.active-elevate-2:not(.no-default-active-elevate)::after {
  content: "";
  pointer-events: none;
  position: absolute;
  inset: 0px;
  /*border-radius: inherit; match rounded corners */
  border-radius: inherit;
  z-index: 999;
  /* sits in front of content */
}

.hover-elevate:hover:not(.no-default-hover-elevate)::after,
.active-elevate:active:not(.no-default-active-elevate)::after {
  background-color: var(--elevate-1);
}

.hover-elevate-2:hover:not(.no-default-hover-elevate)::after,
.active-elevate-2:active:not(.no-default-active-elevate)::after {
  background-color: var(--elevate-2);
}

/* If there's a 1px border, adjust the inset so that it covers that parent's border */
.border.hover-elevate:not(.no-hover-interaction-elevate)::after,
.border.active-elevate:not(.no-active-interaction-elevate)::after,
.border.hover-elevate-2:not(.no-hover-interaction-elevate)::after,
.border.active-elevate-2:not(.no-active-interaction-elevate)::after,
.border.hover-elevate:not(.no-hover-interaction-elevate)::after {
  inset: -1px;
}

```

File: client/src/lib/authUtils.ts

```
export function isUnauthorizedError(error: Error): boolean {  
  return /^401: .*Unauthorized/.test(error.message);  
}
```

File: client/src/lib/queryClient.ts

```
import { QueryClient, QueryFunction } from "@tanstack/react-query";

async function throwIfResNotOk(res: Response) {
  if (!res.ok) {
    const text = (await res.text()) || res.statusText;
    throw new Error(`${res.status}: ${text}`);
  }
}

export async function apiRequest(
  method: string,
  url: string,
  data?: unknown | undefined,
): Promise<Response> {
  const res = await fetch(url, {
    method,
    headers: data ? { "Content-Type": "application/json" } : {},
    body: data ? JSON.stringify(data) : undefined,
    credentials: "include",
  });

  await throwIfResNotOk(res);
  return res;
}

type UnauthorizedBehavior = "returnNull" | "throw";
export const getQueryFn: <T>(options: {
  on401: UnauthorizedBehavior;
}) => QueryFunction<T> =
  ({ on401: unauthorizedBehavior }) => {
    async ({ queryKey }) => {
      const res = await fetch(queryKey.join("/") as string, {
        credentials: "include",
      });

      if (unauthorizedBehavior === "returnNull" && res.status === 401) {
        return null;
      }

      await throwIfResNotOk(res);
      return await res.json();
    };
  };

export const queryClient = new QueryClient({
  defaultOptions: {
    queries: {
      queryFn: getQueryFn({ on401: "throw" }),
      refetchInterval: false,
      refetchOnWindowFocus: false,
      staleTime: Infinity,
      retry: false,
    },
    mutations: {
      retry: false,
    },
  },
});
```

File: client/src/lib/utils.ts

```
import { clsx, type ClassValue } from "clsx"
import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}
```

File: client/src/main.tsx

```
import { createRoot } from "react-dom/client";
import App from "../App";
import "../index.css";

createRoot(document.getElementById("root")!).render(<App />);
```

File: client/src/pages/ClientDashboard.tsx

```
import { useState } from "react";
import { useQuery } from "@tanstack/react-query";
import { SidebarTrigger } from "@/components/ui/sidebar";
import { ThemeToggle } from "@/components/ThemeToggle";
import { OrdersTable } from "@/components/OrdersTable";
import { OrderDetailModal } from "@/components/OrderDetailModal";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import {
  FileBox,
  CheckCircle,
  Clock,
  Loader2,
  Download,
  ExternalLink,
} from "lucide-react";
import type { OrderWithFiles } from "@/shared/schema";

export default function ClientDashboard() {
  const [selectedOrder, setSelectedOrder] = useState<OrderWithFiles | null>(null);

  const { data: orders = [], isLoading } = useQuery<OrderWithFiles[]>({
    queryKey: ["/api/orders"],
  });

  const stats = {
    total: orders.length,
    pending: orders.filter((o) => o.status === "pending").length,
    inProgress: orders.filter((o) =>
      ["paid", "uploaded", "processing"].includes(o.status)
    ).length,
    completed: orders.filter((o) => o.status === "complete").length,
  };

  const handleDownloadOutputs = async () => {
    if (selectedOrder) {
      const outputFile = selectedOrder.files?.find((f) => f.fileType === "output");
      if (outputFile) {
        window.open(`/api/files/${outputFile.id}/download`, "_blank");
      }
    }
  };

  if (isLoading) {
    return (
      <div className="flex items-center justify-center h-full">
        <Loader2 className="h-8 w-8 animate-spin text-muted-foreground" />
      </div>
    );
  }

  return (
    <div className="flex flex-col h-full">
      <header className="flex items-center justify-between gap-4 border-b px-4 py-3 md:px-6">
        <div className="flex items-center gap-3">
          <SidebarTrigger data-testid="button-sidebar-toggle" />
          <div>
            <h1 className="text-lg font-semibold">My Orders</h1>
            <p className="text-sm text-muted-foreground">
              Track your LOD 400 upgrade orders
            </p>
          </div>
          <ThemeToggle />
        </div>
      </header>
      <main className="flex-1 overflow-auto p-4 md:p-6 space-y-6">
        <div className="grid gap-4 sm:grid-cols-3">
          <Card>
            <CardHeader className="flex flex-row items-center justify-between gap-2 space-y-0 pb-2">
              <CardTitle className="text-sm font-medium text-muted-foreground">
                Total Orders
              </CardTitle>
              <FileBox className="h-4 w-4 text-muted-foreground" />
            </CardHeader>
            <CardContent>
              <div className="text-2xl font-bold">{stats.total}</div>
            </CardContent>
          </Card>
          <Card>
            <CardHeader className="flex flex-row items-center justify-between gap-2 space-y-0 pb-2">
              <CardTitle className="text-sm font-medium text-muted-foreground">
                In Progress
              </CardTitle>
              <Clock className="h-4 w-4 text-muted-foreground" />
            </CardHeader>
            <CardContent>
              <div className="text-2xl font-bold">{stats.inProgress}</div>
            </CardContent>
          </Card>
          <Card>
            <CardHeader className="flex flex-row items-center justify-between gap-2 space-y-0 pb-2">
              <CardTitle className="text-sm font-medium text-muted-foreground">
                Completed
              </CardTitle>
              <CheckCircle className="h-4 w-4 text-muted-foreground" />
            </CardHeader>
          </Card>
        </div>
      </main>
    </div>
  );
}
```

```

        <CardContent>
          <div className="text-2xl font-bold">{stats.completed}</div>
        </CardContent>
      </Card>
    </div>

    {orders.length === 0 ? (
      <Card className="py-12">
        <CardContent className="flex flex-col items-center justify-center text-center">
          <div className="rounded-full bg-muted p-4 mb-4">
            <FileBox className="h-8 w-8 text-muted-foreground" />
          </div>
          <h3 className="text-lg font-semibold mb-2">No orders yet</h3>
          <p className="text-sm text-muted-foreground max-w-sm">
            Create your first order using the Revit add-in. Select your sheets,
            pay securely, and upload your model for LOD 400 upgrade.
          </p>
          <Button variant="outline" className="mt-4" asChild>
            <a
              href="https://github.com/LOD400/revit-addin/releases"
              target="_blank"
              rel="noopener noreferrer"
            >
              <Download className="h-4 w-4 mr-2" />
              Download Revit Add-in
            </a>
          </Button>
        </CardContent>
      </Card>
    ) : (
      <div className="space-y-4">
        <h2 className="text-lg font-semibold">Your Orders</h2>
        <OrdersTable
          orders={orders}
          isLoading={isLoading}
          onViewOrder={setSelectedOrder}
        />
      </div>
    )}
  </main>

  <OrderDetailModal
    order={selectedOrder}
    isOpen={!selectedOrder}
    onClose={() => setSelectedOrder(null)}
    onDownloadOutputs={handleDownloadOutputs}
  />
</div>
);
}

```

File: client/src/pages/Clients.tsx

```
import { useState } from "react";
import { useQuery } from "@tanstack/react-query";
import { SidebarTrigger } from "@components/ui/sidebar";
import { ThemeToggle } from "@components/ThemeToggle";
import { Input } from "@components/ui/input";
import { Avatar, AvatarFallback, AvatarImage } from "@components/ui/avatar";
import { Card, CardContent } from "@components/ui/card";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@components/ui/table";
import { Skeleton } from "@components/ui/skeleton";
import { Search, Users, Loader2 } from "lucide-react";
import type { User } from "@shared/schema";
import { formatDistanceToNow } from "date-fns";

interface ClientWithStats extends User {
  orderCount: number;
  totalSpent: number;
}

function formatCurrency(amount: number): string {
  return new Intl.NumberFormat("en-SA", {
    style: "currency",
    currency: "SAR",
    minimumFractionDigits: 0,
  }).format(amount);
}

function formatDate(date: Date | string | null | undefined): string {
  if (!date) return "-";
  const d = typeof date === "string" ? new Date(date) : date;
  return formatDistanceToNow(d, { addSuffix: true });
}

export default function Clients() {
  const [searchQuery, setSearchQuery] = useState("");

  const { data: clients = [], isLoading } = useQuery<ClientWithStats[]>({
    queryKey: ["/api/admin/clients"],
  });

  const filteredClients = clients.filter((client) => {
    if (!searchQuery) return true;
    const search = searchQuery.toLowerCase();
    return (
      client.email?.toLowerCase().includes(search) ||
      client.firstName?.toLowerCase().includes(search) ||
      client.lastName?.toLowerCase().includes(search)
    );
  });

  const getUserInitials = (client: ClientWithStats) => {
    if (client.firstName && client.lastName) {
      return `${client.firstName[0]}${client.lastName[0]}`.toUpperCase();
    }
    if (client.email) {
      return client.email[0].toUpperCase();
    }
    return "U";
  };

  if (isLoading) {
    return (
      <div className="flex items-center justify-center h-full">
        <Loader2 className="h-8 w-8 animate-spin text-muted-foreground" />
      </div>
    );
  }

  return (
    <div className="flex flex-col h-full">
      <header className="flex items-center justify-between gap-4 border-b px-4 py-3 md:px-6">
        <div className="flex items-center gap-3">
          <SidebarTrigger data-testid="button-sidebar-toggle" />
        </div>
        <h1 className="text-lg font-semibold">Clients</h1>
        <p className="text-sm text-muted-foreground">View all registered clients</p>
      </header>
      <main className="flex-1 overflow-auto p-4 md:p-6 space-y-6">
        <div className="flex items-center justify-between gap-4">
          <div className="relative w-full md:w-80">
            <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted-foreground" />
            <Input
              placeholder="Search by name or email..."
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
            />
          </div>
        </div>
      </main>
    </div>
  );
}
```

```

        className="pl-9"
        data-testid="input-search-clients"
      />
    </div>
    <div className="text-sm text-muted-foreground">
      {filteredClients.length} client{filteredClients.length !== 1 ? "s" : ""}
    </div>
  </div>

  {clients.length === 0 ? (
    <Card className="py-12">
      <CardContent className="flex flex-col items-center justify-center text-center">
        <div className="rounded-full bg-muted p-4 mb-4">
          <Users className="h-8 w-8 text-muted-foreground" />
        </div>
        <h3 className="text-lg font-semibold mb-2">No clients yet</h3>
        <p className="text-sm text-muted-foreground max-w-sm">
          Clients will appear here once they sign up and create orders.
        </p>
      </CardContent>
    </Card>
  ) : (
    <div className="rounded-md border overflow-x-auto">
      <Table>
        <TableHeader>
          <TableRow>
            <TableHead>Client</TableHead>
            <TableHead>Email</TableHead>
            <TableHead className="text-center">Orders</TableHead>
            <TableHead className="text-right">Total Spent</TableHead>
            <TableHead>Joined</TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {filteredClients.map((client) => (
            <TableRow
              key={client.id}
              className="hover-elevate"
              data-testid={`row-client-${client.id}`}
            >
              <TableCell>
                <div className="flex items-center gap-3">
                  <Avatar className="h-9 w-9">
                    <AvatarImage
                      src={client.profileImageUrl || undefined}
                      alt={client.firstName || "Client"}
                      className="object-cover"
                    />
                    <AvatarFallback className="text-xs">
                      {getUserInitials(client)}
                    </AvatarFallback>
                  </Avatar>
                  <span className="font-medium">
                    {client.firstName} {client.lastName}
                  </span>
                </div>
              </TableCell>
              <TableCell className="text-muted-foreground">
                {client.email || ""}
              </TableCell>
              <TableCell className="text-center font-medium">
                {client.orderCount}
              </TableCell>
              <TableCell className="text-right font-medium">
                {formatCurrency(client.totalSpent)}
              </TableCell>
              <TableCell className="text-muted-foreground text-sm">
                {formatDate(client.createdAt)}
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </div>
  )}
</main>
</div>
);
}

```

File: client/src/pages/Dashboard.tsx

```
import { useState } from "react";
import { useQuery, useMutation } from "@tanstack/react-query";
import { SidebarTrigger } from "@components/ui/sidebar";
import { ThemeToggle } from "@components/ThemeToggle";
import { StatsCard } from "@components/StatsCard";
import { OrdersTable } from "@components/OrdersTable";
import { OrderDetailModal } from "@components/OrderDetailModal";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@components/ui/tabs";
import { useToast } from "@hooks/use-toast";
import { apiRequest, queryClient } from "@lib/queryClient";
import {
  FileBox,
  Clock,
  CheckCircle,
  DollarSign,
  Loader2,
} from "lucide-react";
import type { OrderWithFiles } from "@shared/schema";

type OrderStatus = "pending" | "paid" | "uploaded" | "processing" | "complete";

export default function Dashboard() {
  const { toast } = useToast();
  const [selectedOrder, setSelectedOrder] = useState<OrderWithFiles | null>(null);
  const [statusFilter, setStatusFilter] = useState<"all" | OrderStatus>("all");

  const { data: orders = [], isLoading } = useQuery<OrderWithFiles[]>({
    queryKey: ["/api/admin/orders"],
  });

  const markCompleteMutation = useMutation({
    mutationFn: async (orderId: string) => {
      await apiRequest("POST", `/api/admin/orders/${orderId}/complete`);
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/admin/orders"] });
      toast({
        title: "Order completed",
        description: "The client has been notified via email.",
      });
      setSelectedOrder(null);
    },
    onError: () => {
      toast({
        title: "Error",
        description: "Failed to mark order as complete.",
        variant: "destructive",
      });
    },
  });

  const filteredOrders = orders.filter((order) => {
    statusFilter === "all" ? true : order.status === statusFilter
  });

  const stats = {
    total: orders.length,
    pending: orders.filter((o) => o.status === "pending").length,
    inProgress: orders.filter((o) => ["paid", "uploaded", "processing"].includes(o.status)).length,
    completed: orders.filter((o) => o.status === "complete").length,
    revenue: orders
      .filter((o) => o.status !== "pending")
      .reduce((acc, o) => acc + o.totalPriceSar, 0),
  };

  const handleDownloadInputs = async (order: OrderWithFiles) => {
    const inputFile = order.files?.find((f) => f.fileType === "input");
    if (inputFile) {
      window.open(`/api/files/${inputFile.id}/download`, "_blank");
    }
  };

  const handleDownloadOutputs = async () => {
    if (selectedOrder) {
      const outputFile = selectedOrder.files?.find((f) => f.fileType === "output");
      if (outputFile) {
        window.open(`/api/files/${outputFile.id}/download`, "_blank");
      }
    }
  };

  if (isLoading) {
    return (
      <div className="flex items-center justify-center h-full">
        <Loader2 className="h-8 w-8 animate-spin text-muted-foreground" />
      </div>
    );
  }

  return (
    <div className="flex flex-col h-full">
      <header className="flex items-center justify-between gap-4 border-b px-4 py-3 md:px-6">
        <div className="flex items-center gap-3">
          <SidebarTrigger data-testid="button-sidebar-toggle" />
        </div>
      </header>
    </div>
  );
}
```

```

        <h1 className="text-lg font-semibold">Dashboard</h1>
        <p className="text-sm text-muted-foreground">
          Manage your LOD 400 orders
        </p>
      </div>
    </div>
    <ThemeToggle />
  </header>

  <main className="flex-1 overflow-auto p-4 md:p-6 space-y-6">
    <div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-4">
      <StatsCard
        title="Total Orders"
        value={stats.total}
        icon={FileBox}
        description="All time orders"
      </>
      <StatsCard
        title="Pending"
        value={stats.pending}
        icon={Clock}
        description="Awaiting payment or upload"
      </>
      <StatsCard
        title="In Progress"
        value={stats.inProgress}
        icon={FileBox}
        description="Being processed"
      </>
      <StatsCard
        title="Revenue"
        value={`$${stats.revenue.toLocaleString()} SAR`}
        icon={DollarSign}
        description="Total earnings"
      </>
    </div>

    <div className="space-y-4">
      <div className="flex flex-col sm:flex-row sm:items-center justify-between gap-4">
        <h2 className="text-lg font-semibold">Recent Orders</h2>
        <Tabs
          value={statusFilter}
          onValueChange={(v) => setStatusFilter(v as typeof statusFilter)}
        >
          <TabsList>
            <TabsTrigger value="all" data-testid="filter-all">
              All
            </TabsTrigger>
            <TabsTrigger value="pending" data-testid="filter-pending">
              Pending
            </TabsTrigger>
            <TabsTrigger value="uploaded" data-testid="filter-uploaded">
              Uploaded
            </TabsTrigger>
            <TabsTrigger value="processing" data-testid="filter-processing">
              Processing
            </TabsTrigger>
            <TabsTrigger value="complete" data-testid="filter-complete">
              Complete
            </TabsTrigger>
          </TabsList>
        </Tabs>
      </div>

      <OrdersTable
        orders={filteredOrders}
        isLoading={isLoading}
        isAdmin={true}
        onViewOrder={setSelectedOrder}
        onDownloadInputs={handleDownloadInputs}
        onUploadOutputs={(order) => {
          setSelectedOrder(order);
        }}
        onMarkComplete={(order) => {
          markCompleteMutation.mutate(order.id);
        }}
      </>
    </div>
  </main>

  <OrderDetailModal
    order={selectedOrder}
    isOpen={!selectedOrder}
    onClose={() => setSelectedOrder(null)}
    isAdmin={true}
    onDownloadInputs={() => {
      if (selectedOrder) handleDownloadInputs(selectedOrder);
    }}
    onDownloadOutputs={handleDownloadOutputs}
    onMarkComplete={() => {
      if (selectedOrder) {
        markCompleteMutation.mutate(selectedOrder.id);
      }
    }}
    isMarkingComplete={markCompleteMutation.isPending}
  </>
</div>
);
}

```

File: client/src/pages/Landing.tsx

```
import { Button } from "@/components/ui/button";
import { Card, CardContent } from "@/components/ui/card";
import { ThemeToggle } from "@/components/ThemeToggle";
import {
  Building2,
  CheckCircle,
  FileBox,
  Shield,
  Zap,
  ArrowRight,
  Layers,
} from "lucide-react";

const features = [
  {
    icon: Layers,
    title: "LOD 400 Precision",
    description:
      "Production-ready models with fabrication-level detail for accurate construction.",
  },
  {
    icon: Zap,
    title: "Fast Turnaround",
    description:
      "Efficient processing workflow ensures your deliverables are ready quickly.",
  },
  {
    icon: Shield,
    title: "Secure Transfer",
    description:
      "Your Revit models are encrypted and securely handled throughout the process.",
  },
  {
    icon: FileBox,
    title: "Easy Integration",
    description:
      "Seamless Revit add-in for direct model upload and delivery downloads.",
  },
];

const steps = [
  {
    number: "01",
    title: "Select Sheets",
    description: "Choose the sheets you need upgraded from your Revit model.",
  },
  {
    number: "02",
    title: "Pay Securely",
    description: "150 SAR per sheet. Secure payment through Stripe.",
  },
  {
    number: "03",
    title: "Upload Model",
    description: "Our add-in packages and uploads your model automatically.",
  },
  {
    number: "04",
    title: "Get Deliverables",
    description: "Receive your LOD 400 upgraded model within days.",
  },
];

export default function Landing() {
  return (
    <div className="min-h-screen bg-background">
      <header className="sticky top-0 z-50 w-full border-b bg-background/95 backdrop-blur supports-[backdrop-filter]:bg-background/60">
        <div className="container flex h-16 items-center justify-between gap-4 px-4 md:px-6">
          <div className="flex items-center gap-3">
            <div className="flex h-9 w-9 items-center justify-center rounded-md bg-primary">
              <Building2 className="h-5 w-5 text-primary-foreground" />
            </div>
            <span className="font-semibold text-lg hidden sm:inline-block">
              LOD 400 Delivery
            </span>
          </div>
          <div className="flex items-center gap-3">
            <ThemeToggle />
            <Button asChild data-testid="button-login">
              <a href="/api/login">Sign In</a>
            </Button>
          </div>
        </div>
      </header>
      <main>
        <section className="relative py-20 md:py-32 overflow-hidden">
          <div className="absolute inset-0 bg-gradient-to-b from-primary/5 to-transparent" />
          <div className="container px-4 md:px-6 relative">
            <div className="flex flex-col items-center text-center max-w-3xl mx-auto space-y-8">
              <div className="inline-flex items-center gap-2 rounded-full bg-primary/10 px-4 py-1.5 text-sm font-medium text-primary">
                <CheckCircle className="h-4 w-4" />
                Professional BIM Model Upgrades
              </div>
              <h1 className="text-4xl md:text-5xl lg:text-6xl font-bold tracking-tight">
                Transform Your Models to{" "}
                <span className="text-primary">LOD 400</span>
              </h1>
            </div>
          </div>
        </section>
      </main>
    </div>
  );
}
```

```

<!--/h1-->
<div className="text-lg md:text-xl text-muted-foreground max-w-2xl">
  Professional LOD 300 to LOD 400 model upgrades for construction
  documentation. Upload directly from Revit and receive
  production-ready deliverables.
</div>
<div className="flex flex-col sm:flex-row gap-4">
  <Button size="lg" asChild data-testid="button-get-started">
    <a href="/api/login">
      Get Started
    </a>
  </Button>
  <Button
    size="lg"
    variant="outline"
    asChild
    data-testid="button-learn-more">
    <a href="#how-it-works">Learn More</a>
  </Button>
</div>
<div className="flex items-center gap-6 pt-4 text-sm text-muted-foreground">
  <div className="flex items-center gap-2">
    <CheckCircle className="h-4 w-4 text-green-500" />
    150 SAR per sheet
  </div>
  <div className="flex items-center gap-2">
    <CheckCircle className="h-4 w-4 text-green-500" />
    Secure payment
  </div>
  <div className="flex items-center gap-2">
    <CheckCircle className="h-4 w-4 text-green-500" />
    Fast delivery
  </div>
</div>
</div>
</section>

<section className="py-20 border-t bg-muted/30" id="features">
  <div className="container px-4 md:px-6">
    <div className="text-center mb-12">
      <h2 className="text-3xl font-bold mb-4">Why Choose Us</h2>
      <p className="text-muted-foreground max-w-2xl mx-auto">
        We specialize in delivering high-quality LOD 400 upgrades that
        meet construction and fabrication requirements.
      </p>
    </div>
    <div className="grid sm:grid-cols-2 lg:grid-cols-4 gap-6">
      {features.map((feature) => {
        <Card key={feature.title} className="hover-elevate">
          <CardContent className="pt-6">
            <div className="flex h-12 w-12 items-center justify-center rounded-lg bg-primary/10 mb-4">
              <feature.icon className="h-6 w-6 text-primary" />
            </div>
            <h3 className="font-semibold mb-2">{feature.title}</h3>
            <p className="text-sm text-muted-foreground">
              {feature.description}
            </p>
          </CardContent>
        </Card>
      })}
    </div>
  </div>
</section>

<section className="py-20 border-t" id="how-it-works">
  <div className="container px-4 md:px-6">
    <div className="text-center mb-12">
      <h2 className="text-3xl font-bold mb-4">How It Works</h2>
      <p className="text-muted-foreground max-w-2xl mx-auto">
        A simple, streamlined process from order to delivery.
      </p>
    </div>
    <div className="grid sm:grid-cols-2 lg:grid-cols-4 gap-8">
      {steps.map((step, index) => {
        <div key={step.number} className="relative">
          <div className="text-5xl font-bold text-primary/10 mb-4">
            {step.number}
          </div>
          <h3 className="font-semibold mb-2">{step.title}</h3>
          <p className="text-sm text-muted-foreground">
            {step.description}
          </p>
          {index < steps.length - 1 & & (
            <ArrowRight className="hidden lg:block absolute top-8 -right-4 h-6 w-6 text-muted-foreground/30" />
          )}
        </div>
      })}
    </div>
  </div>
</section>

<section className="py-20 border-t bg-primary/5">
  <div className="container px-4 md:px-6">
    <div className="flex flex-col items-center text-center max-w-2xl mx-auto space-y-6">
      <h2 className="text-3xl font-bold">Ready to Get Started?</h2>
      <p className="text-sm text-muted-foreground">
        Sign in to access the platform and start upgrading your BIM
        models to LOD 400 today.
      </p>
    </div>
  </div>
</section>

```

```

        </p>
        <Button size="lg" asChild>
          <a href="/api/login">
            Sign In to Continue
          <ArrowRight className="ml-2 h-4 w-4" />
        </a>
      </Button>
    </div>
  </div>
</section>
</main>

<footer className="border-t py-8">
  <div className="container px-4 md:px-6">
    <div className="flex flex-col md:flex-row items-center justify-between gap-4">
      <div className="flex items-center gap-2">
        <Building2 className="h-5 w-5 text-primary" />
        <span className="font-medium">LOD 400 Delivery Platform</span>
      </div>
      <p className="text-sm text-muted-foreground">
        Professional BIM model upgrades for construction excellence.
      </p>
    </div>
  </div>
</footer>
</div>
);
}

```

File: client/src/pages/not-found.tsx

```
import { Button } from "@/components/ui/button";
import { Home, ArrowLeft } from "lucide-react";

export default function NotFound() {
  return (
    <div className="flex flex-col items-center justify-center min-h-screen p-4 text-center">
      <div className="space-y-6 max-w-md">
        <div className="space-y-2">
          <h1 className="text-6xl font-bold text-primary">404</h1>
          <h2 className="text-2xl font-semibold">Page Not Found</h2>
          <p className="text-muted-foreground">
            The page you're looking for doesn't exist or has been moved.
          </p>
        </div>
        <div className="flex flex-col sm:flex-row gap-3 justify-center">
          <Button variant="outline" onClick={() => window.history.back()} data-testid="button-go-back">
            <ArrowLeft className="h-4 w-4 mr-2" />
            Go Back
          </Button>
          <Button asChild data-testid="button-go-home">
            <a href="/">
              <Home className="h-4 w-4 mr-2" />
              Go Home
            </a>
          </Button>
        </div>
      </div>
    </div>
  );
}
```

File: client/src/pages/Orders.tsx

```
import { useState } from "react";
import { useQuery, useMutation } from "@tanstack/react-query";
import { SidebarTrigger } from "@/components/ui/sidebar";
import { ThemeToggle } from "@/components/ThemeToggle";
import { OrdersTable } from "@/components/OrdersTable";
import { OrderDetailModal } from "@/components/OrderDetailModal";
import { Input } from "@/components/ui/input";
import { Tabs, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { useToast } from "@/hooks/use-toast";
import { apiRequest, queryClient } from "@/lib/queryClient";
import { Search, Loader2 } from "lucide-react";
import type { OrderWithFiles } from "@/shared/schema";

type OrderStatus = "pending" | "paid" | "uploaded" | "processing" | "complete";

export default function Orders() {
  const { toast } = useToast();
  const [selectedOrder, setSelectedOrder] = useState<OrderWithFiles | null>(null);
  const [statusFilter, setStatusFilter] = useState<"all" | OrderStatus>("all");
  const [searchQuery, setSearchQuery] = useState("");

  const { data: orders = [], isLoading } = useQuery<OrderWithFiles[],>({
    queryKey: ["/api/admin/orders"],
  });

  const markCompleteMutation = useMutation({
    mutationFn: async (orderId: string) => {
      await apiRequest("POST", `/api/admin/orders/${orderId}/complete`);
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/admin/orders"] });
      toast({
        title: "Order completed",
        description: "The client has been notified via email.",
      });
      setSelectedOrder(null);
    },
    onError: () => {
      toast({
        title: "Error",
        description: "Failed to mark order as complete.",
        variant: "destructive",
      });
    },
  });

  const updateStatusMutation = useMutation({
    mutationFn: async ({ orderId, status }: { orderId: string; status: OrderStatus }) => {
      await apiRequest("PATCH", `/api/admin/orders/${orderId}/status`, { status });
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/admin/orders"] });
      toast({
        title: "Status updated",
        description: "Order status has been updated.",
      });
    },
    onError: () => {
      toast({
        title: "Error",
        description: "Failed to update order status.",
        variant: "destructive",
      });
    },
  });

  const filteredOrders = orders.filter((order) => {
    const matchesStatus = statusFilter === "all" || order.status === statusFilter;
    const matchesSearch =
      !searchQuery ||
      order.id.toLowerCase().includes(searchQuery.toLowerCase()) ||
      order.user?.email?.toLowerCase().includes(searchQuery.toLowerCase()) ||
      order.user?.firstName?.toLowerCase().includes(searchQuery.toLowerCase()) ||
      order.user?.lastName?.toLowerCase().includes(searchQuery.toLowerCase());
    return matchesStatus & matchesSearch;
  });

  const handleDownloadInputs = async (order: OrderWithFiles) => {
    const inputFile = order.files?.find((f) => f.fileType === "input");
    if (inputFile) {
      window.open(`/api/files/${inputFile.id}/download`, "_blank");
    }
  };

  const handleDownloadOutputs = async () => {
    if (selectedOrder) {
      const outputFile = selectedOrder.files?.find((f) => f.fileType === "output");
      if (outputFile) {
        window.open(`/api/files/${outputFile.id}/download`, "_blank");
      }
    }
  };

  if (isLoading) {
    return (
      <div className="flex items-center justify-center h-full">
        <Loader2 className="h-8 w-8 animate-spin text-muted-foreground" />
      </div>
    );
  }
}
```

```

    </div>>
  );
}

return (
  <div className="flex flex-col h-full">
    <header className="flex items-center justify-between gap-4 border-b px-4 py-3 md:px-6">
      <div className="flex items-center gap-3">
        <SidebarTrigger data-testid="button-sidebar-toggle" />
      </div>
      <h1 className="text-lg font-semibold">Orders</h1>
      <p className="text-sm text-muted-foreground">
        View and manage all orders
      </p>
    </div>
    <div>
      <ThemeToggle />
    </div>
  </header>

  <main className="flex-1 overflow-auto p-4 md:p-6 space-y-6">
    <div className="flex flex-col md:flex-row md:items-center justify-between gap-4">
      <div className="relative w-full md:w-80">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted-foreground" />
        <input
          placeholder="Search by order ID, client name or email..."
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)}
          className="pl-9"
          data-testid="input-search"
        />
      </div>
      <div>
        <Tabs
          value={statusFilter}
          onValueChange={(v) => setStatusFilter(v as typeof statusFilter)}
        >
          <TabsList className="flex-wrap">
            <TabsTrigger value="all">All ({orders.length})</TabsTrigger>
            <TabsTrigger value="pending">
              Pending ({orders.filter((o) => o.status === "pending").length})
            </TabsTrigger>
            <TabsTrigger value="paid">
              Paid ({orders.filter((o) => o.status === "paid").length})
            </TabsTrigger>
            <TabsTrigger value="uploaded">
              Uploaded ({orders.filter((o) => o.status === "uploaded").length})
            </TabsTrigger>
            <TabsTrigger value="processing">
              Processing ({orders.filter((o) => o.status === "processing").length})
            </TabsTrigger>
            <TabsTrigger value="complete">
              Complete ({orders.filter((o) => o.status === "complete").length})
            </TabsTrigger>
          </TabsList>
        </div>
      </div>

    <OrdersTable
      orders={filteredOrders}
      isLoading={isLoading}
      isAdmin={true}
      onViewOrder={setSelectedOrder}
      onDownloadInputs={handleDownloadInputs}
      onUploadOutputs={(order) => {
        setSelectedOrder(order);
      }}
      onMarkComplete={(order) => {
        markCompleteMutation.mutate(order.id);
      }}
    />
  </main>

  <OrderDetailModal
    order={selectedOrder}
    isOpen={!selectedOrder}
    onClose={() => setSelectedOrder(null)}
    isAdmin={true}
    onDownloadInputs={() => {
      if (selectedOrder) handleDownloadInputs(selectedOrder);
    }}
    onDownloadOutputs={handleDownloadOutputs}
    onMarkComplete={() => {
      if (selectedOrder) {
        markCompleteMutation.mutate(selectedOrder.id);
      }
    }}
    isMarkingComplete={markCompleteMutation.isPending}
  />
  </div>
);
}

```

File: client/src/pages/Settings.tsx

```
import { useState } from "react";
import { useQuery, useMutation } from "@tanstack/react-query";
import { SidebarTrigger } from "@components/ui/sidebar";
import { ThemeToggle } from "@components/ThemeToggle";
import { useTheme } from "@components/ThemeProvider";
import { useAuth } from "@hooks/useAuth";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@components/ui/card";
import { Button } from "@components/ui/button";
import { Switch } from "@components/ui/switch";
import { Label } from "@components/ui/label";
import { Input } from "@components/ui/input";
import { Avatar, AvatarFallback, AvatarImage } from "@components/ui/avatar";
import { Badge } from "@components/ui/badge";
import { Separator } from "@components/ui/separator";
import { useToast } from "@hooks/use-toast";
import { apiRequest, queryClient } from "@lib/queryClient";
import { User, Shield, Moon, Sun, LogOut, ExternalLink, Download, Key, Plus, Copy, Trash2, Eye, EyeOff, Check, Loader2 } from "lucide-react";
import type { ApiKey } from "@shared/schema";

export default function Settings() {
  const { user, isAdmin } = useAuth();
  const { theme, toggleTheme } = useTheme();
  const { toast } = useToast();
  const [newKeyName, setNewKeyName] = useState("");
  const [showNewKey, setShowNewKey] = useState<string | null>(null);
  const [copiedKey, setCopiedKey] = useState(false);

  const { data: apiKeys, isLoading: keysLoading } = useQuery<ApiKey[]>({
    queryKey: ["/api/user/api-keys"],
    enabled: !isAdmin,
  });

  const createKeyMutation = useMutation({
    mutationFn: async (name: string) => {
      const response = await apiRequest("POST", "/api/user/api-keys", { name });
      return response.json();
    },
    onSuccess: (data) => {
      setShowNewKey(data.rawKey);
      setNewKeyName("");
      queryClient.invalidateQueries({ queryKey: ["/api/user/api-keys"] });
      toast({
        title: "API Key Created",
        description: "Your new API key has been created. Copy it now - you won't be able to see it again!",
      });
    },
    onError: () => {
      toast({
        variant: "destructive",
        title: "Error",
        description: "Failed to create API key. Please try again.",
      });
    },
  });

  const deleteKeyMutation = useMutation({
    mutationFn: async (keyId: string) => {
      await apiRequest("DELETE", `/api/user/api-keys/${keyId}`);
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["/api/user/api-keys"] });
      toast({
        title: "API Key Deleted",
        description: "The API key has been revoked and can no longer be used.",
      });
    },
    onError: () => {
      toast({
        variant: "destructive",
        title: "Error",
        description: "Failed to delete API key. Please try again.",
      });
    },
  });

  const handleCopyKey = async (key: string) => {
    try {
      await navigator.clipboard.writeText(key);
      setCopiedKey(true);
      setTimeout(() => setCopiedKey(false), 2000);
    } catch {
      toast({
        variant: "destructive",
        title: "Copy Failed",
        description: "Could not copy to clipboard. Please select and copy manually.",
      });
    }
  };

  const handleCreateKey = () => {
    if (newKeyName.trim()) {
      setShowNewKey(null);
      setCopiedKey(false);
      createKeyMutation.mutate(newKeyName.trim());
    }
  };

  const getUserInitials = () => {
```

```

if (user?.firstName & user?.lastName) {
  return `${user.firstName[0]}${user.lastName[0]}`.toUpperCase();
}
if (user?.email) {
  return user.email[0].toUpperCase();
}
return "U";
};

return (
  <div className="flex flex-col h-full">
    <header className="flex items-center justify-between gap-4 border-b px-4 py-3 md:px-6">
      <div className="flex items-center gap-3">
        <SidebarTrigger data-testid="button-sidebar-toggle" />
        <div>
          <h1 className="text-lg font-semibold">Settings</h1>
          <p className="text-sm text-muted-foreground">
            Manage your account and preferences
          </p>
        </div>
      </div>
      <ThemeToggle />
    </header>

    <main className="flex-1 overflow-auto p-4 md:p-6">
      <div className="max-w-2xl space-y-6">
        <Card>
          <CardHeader>
            <CardTitle className="flex items-center gap-2">
              <User className="h-5 w-5" />
              Profile
            </CardTitle>
            <CardDescription>
              Your account information from your sign-in provider.
            </CardDescription>
          </CardHeader>
          <CardContent className="space-y-4">
            <div className="flex items-center gap-4">
              <Avatar className="h-16 w-16">
                <AvatarImage
                  src={user?.profileImageUrl || undefined}
                  alt={user?.firstName || "User"}
                  className="object-cover"
                />
                <AvatarFallback className="text-lg">
                  {getUserInitials()}
                </AvatarFallback>
              </Avatar>
              <div className="space-y-1">
                <div className="flex items-center gap-2">
                  <h3 className="font-semibold">
                    {user?.firstName} {user?.lastName}
                  </h3>
                  {isAdmin & (
                    <Badge variant="secondary" className="gap-1">
                      <Shield className="h-3 w-3" />
                      Admin
                    </Badge>
                  )}
                </div>
                <p className="text-sm text-muted-foreground">
                  {user?.email || "No email associated"}
                </p>
              </div>
            </div>
          </CardContent>
        </Card>

        <Card>
          <CardHeader>
            <CardTitle className="flex items-center gap-2">
              {theme === "dark" ? (
                <Moon className="h-5 w-5" />
              ) : (
                <Sun className="h-5 w-5" />
              )}
              Appearance
            </CardTitle>
            <CardDescription>
              Customize how the application looks.
            </CardDescription>
          </CardHeader>
          <CardContent>
            <div className="flex items-center justify-between">
              <div className="space-y-0.5">
                <Label htmlFor="dark-mode">Dark Mode</Label>
                <p className="text-sm text-muted-foreground">
                  Switch between light and dark themes.
                </p>
              </div>
              <Switch
                id="dark-mode"
                checked={theme === "dark"}
                onCheckedChange={toggleTheme}
                data-testid="switch-dark-mode"
              />
            </div>
          </CardContent>
        </Card>
      </div>
    </main>
  </div>
);

```

[illegible]

```

    })
    <button>
  </div>
})
</div>
) : (
  <p className="text-sm text-muted-foreground text-center py-4">
    No API keys yet. Create one to use with the Revit add-in.
  </p>
)
</CardContent>
</Card>

<Card>
  <CardHeader>
    <CardTitle className="flex items-center gap-2">
      <Download className="h-5 w-5" />
      Revit Add-in
    </CardTitle>
    <CardDescription>
      Download and install the add-in to submit orders directly from Revit.
    </CardDescription>
  </CardHeader>
  <CardContent>
    <Button variant="outline" asChild>
      <a
        href="https://github.com/LOD400/revit-addin/releases"
        target="_blank"
        rel="noopener noreferrer"
      >
        <Download className="h-4 w-4 mr-2" />
        Download Add-in
        <ExternalLink className="h-3 w-3 ml-2" />
      </a>
    </Button>
  </CardContent>
</Card>
</>
)

<Separator />

<Card className="border-destructive/50">
  <CardHeader>
    <CardTitle className="text-destructive">Sign Out</CardTitle>
    <CardDescription>
      Sign out of your account on this device.
    </CardDescription>
  </CardHeader>
  <CardContent>
    <Button variant="destructive" asChild data-testid="button-signout">
      <a href="/api/logout">
        <Logout className="h-4 w-4 mr-2" />
        Sign Out
      </a>
    </Button>
  </CardContent>
</Card>
</div>
</main>
</div>
);
}

```

File: components.json

```
{
  "$schema": "https://ui.shadcn.com/schema.json",
  "style": "new-york",
  "rsc": false,
  "tsx": true,
  "tailwind": {
    "config": "tailwind.config.ts",
    "css": "client/src/index.css",
    "baseColor": "neutral",
    "cssVariables": true,
    "prefix": ""
  },
  "aliases": {
    "components": "@components",
    "utils": "@lib/utils",
    "ui": "@components/ui",
    "lib": "@lib",
    "hooks": "@hooks"
  }
}
```

File: design_guidelines.md

Design Guidelines: LOD 400 Delivery Platform

Design Approach

****System Selection**:** Modern SaaS Dashboard Pattern (inspired by Linear, Vercel, Stripe)

****Rationale**:** This is a professional B2B utility application requiring clarity, efficiency, and trust. The design prioritizes information hierarchy, workflow

Core Design Elements

A. Typography

****Font Stack**:** Inter (via Google Fonts CDN)

- ****Headings**:** 600 weight, tight letter-spacing (-0.02em)
 - Page titles: text-2xl to text-3xl
 - Section headers: text-lg to text-xl
- ****Body**:** 400 weight, comfortable line-height (1.6)
 - Primary text: text-base
 - Secondary/meta text: text-sm
 - Small labels: text-xs
- ****Data/Numbers**:** 500 weight (for order IDs, prices, counts)
- ****Buttons/CTAs**:** 500 weight, uppercase for primary actions

B. Layout System

****Spacing Primitives**:** Consistent use of Tailwind units: 2, 4, 6, 8, 12, 16, 24

- Component padding: p-4 to p-6
- Section spacing: space-y-6 to space-y-8
- Card internal spacing: p-6
- Form field gaps: space-y-4
- Button padding: px-4 py-2 (standard), px-6 py-3 (large)

****Container Strategy**:**

- Admin dashboard: Full-width with sidebar (fixed 64 width on desktop)
- Content area: max-w-7xl with px-4 to px-8
- Forms/modals: max-w-2xl centered
- Tables: Full container width with horizontal scroll on mobile

C. Component Library

****Navigation**:**

- Fixed sidebar (desktop): Logo at top, navigation items with icons, admin info at bottom
- Mobile: Collapsible hamburger menu with overlay
- Top bar: Breadcrumbs, search (if needed), notifications icon, profile dropdown

****Dashboard Cards**:**

- Stats cards: Grid layout (grid-cols-1 md:grid-cols-3), rounded-lg borders
- Each card: Icon (top-left), metric (large number), label (small text), trend indicator
- Subtle hover elevation (shadow-sm to shadow-md transition)

****Data Tables**:**

- Striped rows for readability (alternating subtle backgrounds)
- Sticky header on scroll
- Row hover state (subtle background change)
- Status badges: Pill-shaped, color-coded (Pending: amber, Paid: blue, Complete: green)
- Actions column: Icon buttons (download, view, edit, delete)
- Pagination: Bottom-right, showing "1-10 of 234 results"

****Forms**:**

- Full-width inputs with consistent height (h-10)
- Labels above inputs (text-sm, font-medium)
- Input states: default, focus (ring-2), error (red border), disabled (opacity-50)
- File upload: Drag-and-drop zone with dashed border, icon, and instructions
- Submit buttons: Right-aligned, primary style

****Modals/Overlays**:**

- Centered overlay with backdrop (backdrop-blur-sm, bg-black/50)
- Card style: rounded-lg, max-w-lg to max-w-2xl
- Header with title and close button (X icon top-right)
- Content area with appropriate padding
- Footer with action buttons (Cancel left, Primary right)

****Buttons**:**

- Primary: Solid fill, rounded-md, shadow-sm
- Secondary: Border style with transparent background
- Danger: Red variant for destructive actions
- Icon-only: Square aspect, centered icon
- Loading state: Spinner icon, disabled opacity

****Status Indicators**:**

- Progress bars: Rounded-full, gradient fill showing percentage
- Loading spinners: Centered with optional text below
- Toast notifications: Top-right corner, slide-in animation, auto-dismiss

****Empty States**:**

- Centered icon (large, muted)
- Heading and description text
- Primary action button below

D. Application Structure

****Admin Dashboard Layout**:**

1. ****Sidebar**** (fixed left, 64 units wide):
 - Logo/branding at top
 - Navigation links: Dashboard, Orders, Settings
 - Admin profile card at bottom
2. ****Main Content Area**:**

- ****Header Section****: Page title, optional action button (e.g., "New Order" - if manual creation needed)
- ****Stats Overview****: 4-column grid showing: Total Orders, Pending, In Progress, Completed
- ****Orders Table****: Filterable/sortable table with columns: Order ID, Client Email, Sheet Count, Price, Status, Upload Date, Actions
- Each row action: View details, Download inputs, Upload outputs, Mark complete

3. ****Order Detail View**** (modal or dedicated page):

- Order summary card (ID, date, status, price)
- Client information section
- Files section: Input files (download button), Output files (upload interface)
- Status timeline showing: Created → Paid → Uploaded → Processing → Complete
- Action buttons: Download all, Upload deliverables, Mark complete, Send notification

****File Management Interface****:

- Upload zone: Large, centered, with progress bar during upload
- File list: Name, size, upload date, download/delete actions
- Chunked upload progress: Real-time percentage indicator

****Payment/Order Flow**** (what client sees - can be simple):

- Order confirmation page showing: Sheet count, total price, "Pay with Stripe" button
- Post-payment success page with order ID and status check link
- Order status page: Timeline view, file download when complete

E. Trust & Professional Elements

- Security badges: "Secure Payment via Stripe" near checkout
- Professional email templates: Branded header, clear order details, prominent CTA
- Error handling: Friendly but informative messages, never raw errors
- Loading states: Always show progress, never leave users uncertain

F. Responsive Behavior

****Desktop (lg:)****: Full sidebar, multi-column tables, 3-4 column stat grids

****Tablet (md:)****: Collapsible sidebar, 2-column grids, scrollable tables

****Mobile (base)****: Hidden sidebar (hamburger menu), single-column layouts, stacked cards, simplified table views with expandable rows

Images

****No hero images required**** - this is a functional dashboard application, not a marketing site. Use icons throughout for visual interest:

- ****Dashboard icons****: Chart/graph icons for stats cards (from Heroicons)
- ****File type icons****: Document/folder icons in file lists
- ****Empty state illustrations****: Simple line-art style icons (large, centered)
- ****Logo placeholder****: Top-left of sidebar (can be text-based initially)

Focus is on clarity and efficiency, not visual storytelling.

File: drizzle.config.ts

```
import { defineConfig } from "drizzle-kit";

if (!process.env.DATABASE_URL) {
  throw new Error("DATABASE_URL, ensure the database is provisioned");
}

export default defineConfig({
  out: "./migrations",
  schema: "./shared/schema.ts",
  dialect: "postgresql",
  dbCredentials: {
    url: process.env.DATABASE_URL,
  },
});
```

File: package.json

```
{
  "name": "rest-express",
  "version": "1.0.0",
  "type": "module",
  "license": "MIT",
  "scripts": {
    "dev": "NODE_ENV=development tsx server/index.ts",
    "build": "tsx script/build.ts",
    "start": "NODE_ENV=production node dist/index.cjs",
    "check": "tsc",
    "db:push": "drizzle-kit push"
  },
  "dependencies": {
    "@google-cloud/storage": "^7.17.3",
    "@hookform/resolvers": "^3.10.0",
    "@jridgewell/trace-mapping": "^0.3.25",
    "@neondatabase/serverless": "^0.10.4",
    "@radix-ui/react-accordion": "^1.2.4",
    "@radix-ui/react-alert-dialog": "^1.1.7",
    "@radix-ui/react-aspect-ratio": "^1.1.3",
    "@radix-ui/react-avatar": "^1.1.4",
    "@radix-ui/react-checkbox": "^1.1.5",
    "@radix-ui/react-collapsible": "^1.1.4",
    "@radix-ui/react-context-menu": "^2.2.7",
    "@radix-ui/react-dialog": "^1.1.7",
    "@radix-ui/react-dropdown-menu": "^2.1.7",
    "@radix-ui/react-hover-card": "^1.1.7",
    "@radix-ui/react-label": "^2.1.3",
    "@radix-ui/react-menubar": "^1.1.7",
    "@radix-ui/react-navigation-menu": "^1.2.6",
    "@radix-ui/react-popover": "^1.1.7",
    "@radix-ui/react-progress": "^1.1.3",
    "@radix-ui/react-radio-group": "^1.2.4",
    "@radix-ui/react-scroll-area": "^1.2.4",
    "@radix-ui/react-select": "^2.1.7",
    "@radix-ui/react-separator": "^1.1.3",
    "@radix-ui/react-slider": "^1.2.4",
    "@radix-ui/react-slot": "^1.2.0",
    "@radix-ui/react-switch": "^1.1.4",
    "@radix-ui/react-tabs": "^1.1.4",
    "@radix-ui/react-toast": "^1.2.7",
    "@radix-ui/react-toggle": "^1.1.3",
    "@radix-ui/react-toggle-group": "^1.1.3",
    "@radix-ui/react-tooltip": "^1.2.0",
    "@tanstack/react-query": "^5.60.5",
    "@types/memoizee": "^0.4.12",
    "@uppy/aws-s3": "^5.0.2",
    "@uppy/core": "^5.1.1",
    "@uppy/dashboard": "^5.0.4",
    "@uppy/react": "^5.1.1",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "cmdk": "^1.1.1",
    "connect-pg-simple": "^10.0.0",
    "date-fns": "^3.6.0",
    "drizzle-orm": "^0.39.1",
    "drizzle-zod": "^0.7.0",
    "embla-carousel-react": "^8.6.0",
    "express": "^4.21.2",
    "express-session": "^1.18.1",
    "framer-motion": "^11.13.1",
    "input-otp": "^1.4.2",
    "lucide-react": "^0.453.0",
    "memoizee": "^0.4.17",
    "memorystore": "^1.6.7",
    "next-themes": "^0.4.6",
    "openid-client": "^6.8.1",
    "passport": "^0.7.0",
    "passport-local": "^1.0.0",
    "react": "^18.3.1",
    "react-day-picker": "^8.10.1",
    "react-dom": "^18.3.1",
    "react-hook-form": "^7.55.0",
    "react-icons": "^5.4.0",
    "react-resizable-panels": "^2.1.7",
    "recharts": "^2.15.2",
    "stripe": "^18.5.0",
    "stripe-replit-sync": "^0.0.12",
    "tailwind-merge": "^2.6.0",
    "tailwindcss-animate": "^1.0.7",
    "tw-animate-css": "^1.2.5",
    "vaul": "^1.1.2",
    "wouter": "^3.3.5",
    "ws": "^8.18.0",
    "zod": "^3.24.2",
    "zod-validation-error": "^3.4.0"
  },
  "devDependencies": {
    "@replit/vite-plugin-cartographer": "^0.4.4",
    "@replit/vite-plugin-dev-banner": "^0.1.1",
    "@replit/vite-plugin-runtime-error-modal": "^0.0.3",
    "@tailwindcss/typography": "^0.5.15",
    "@tailwindcss/vite": "^4.1.3",
    "@types/connect-pg-simple": "^7.0.3",
    "@types/express": "4.17.21",
    "@types/express-session": "^1.18.0",
    "@types/node": "20.16.11",
  }
}
```

```
"@types/passport": "^1.0.16",
"@types/passport-local": "^1.0.38",
"@types/react": "^18.3.11",
"@types/react-dom": "^18.3.1",
"@types/ws": "^8.5.13",
"@vitejs/plugin-react": "^4.7.0",
"autoprefixer": "^10.4.20",
"drizzle-kit": "^0.31.4",
"esbuild": "^0.25.0",
"postcss": "^8.4.47",
"tailwindcss": "^3.4.17",
"tsx": "^4.20.5",
"typescript": "5.6.3",
"vite": "^5.4.20"
},
"optionalDependencies": {
  "bufferutil": "^4.0.8"
}
}
```

File: postcss.config.js

```
export default {  
  plugins: {  
    tailwindcss: {},  
    autoprefixer: {},  
  },  
}
```

File: README.md

LOD 400 Delivery Platform

A professional B2B SaaS platform for upgrading Building Information Modeling (BIM) models from LOD 300 to LOD 400 specification. The platform provides a complete solution for the construction industry.

Features

Web Dashboard

- **Client Portal**: Create orders, track progress, download deliverables
- **Admin Dashboard**: Manage orders, upload completed files, view analytics
- **Secure Authentication**: Login via Replit Auth (OpenID Connect)
- **Payment Processing**: Stripe integration with 150 SAR (~\$40) per sheet pricing
- **Dark/Light Theme**: Modern UI with theme switching
- **API Key Management**: Generate keys for Revit add-in authentication

Revit Add-in

- **Sheet Selection**: Browse and select specific sheets from your Revit model
- **Real-time Pricing**: See pricing before payment
- **Secure Upload**: Automatic model packaging with workshared support
- **Order Tracking**: Check status and download deliverables directly from Revit
- **API Key Auth**: Secure connection to platform using generated API keys

Tech Stack

Frontend

- **React** with TypeScript
- **Vite** for build tooling
- **Tailwind CSS** with shadcn/ui components
- **TanStack Query** for data fetching
- **wouter** for routing

Backend

- **Node.js** with Express
- **TypeScript** (ES modules)
- **Drizzle ORM** with PostgreSQL
- **Passport.js** for authentication

Infrastructure

- **PostgreSQL** (Neon Serverless) - Database
- **Google Cloud Storage** - File storage
- **Stripe** - Payment processing
- **Replit Auth** - User authentication

Revit Add-in

- **C#** with .NET Framework 4.8
- **WPF** for user interface
- **Revit API** (2020-2025 compatible)

Project Structure

```
...
├── client/
│   ├── src/
│   │   ├── components/
│   │   │   └── # Reusable UI components
│   │   ├── pages/
│   │   │   └── # Page components
│   │   ├── hooks/
│   │   │   └── # Custom React hooks
│   │   ├── lib/
│   │   │   └── # Utility functions
│   └── index.html
├── server/
│   ├── index.ts
│   │   └── # Server entry point
│   ├── routes.ts
│   │   └── # API routes
│   ├── storage.ts
│   │   └── # Database operations
│   ├── auth.ts
│   │   └── # Authentication logic
│   └── shared/
│       ├── schema.ts
│       │   └── # Database schema & types
│       └── revit-addin/
│           └── # Revit add-in source
│               ├── LOD400Uploader/
│               │   └── # C# project
│               └── README.md
...
```

Getting Started

Prerequisites

- Node.js 18+
- PostgreSQL database
- Stripe account
- (For add-in) Visual Studio 2022 + Revit 2020-2025

Installation

- Clone the repository**

```
``bash
git clone https://github.com/yourusername/lo400-platform.git
cd lo400-platform
``
```
- Install dependencies**

```
``bash
npm install
``
```
- Set up environment variables**

Create a `.env` file with:

```
``env
DATABASE_URL=postgresql://user:password@host:5432/database
SESSION_SECRET=your-secure-session-secret
``
```

```
STRIPE_SECRET_KEY=sk_live_or_test_key
STRIPE_PUBLISHABLE_KEY=pk_live_or_test_key
```
```

```
4. **Push database schema**
```bash
npm run db:push
```
```

```
5. **Start the development server**
```bash
npm run dev
```
```

The app will be available at `http://localhost:5000`

### ### Production Deployment

For production deployment on platforms like Replit, Railway, or Render:

1. Set all environment variables in your platform's dashboard
2. Build the application:  
```bash  
npm run build
```
3. Start the production server:  
```bash  
npm start
```

### ## Configuration

#### ### Environment Variables

Variable	Description	Required
DATABASE_URL	PostgreSQL connection string	Yes
SESSION_SECRET	Secret for session encryption	Yes
STRIPE_SECRET_KEY	Stripe secret API key	Yes
STRIPE_PUBLISHABLE_KEY	Stripe publishable key	Yes
DEFAULT_OBJECT_STORAGE_BUCKET_ID	GCS bucket ID for file storage	For file uploads

#### ### Pricing Configuration

The price per sheet is configured in `shared/schema.ts`:

```
```typescript
export const PRICE_PER_SHEET_SAR = 150; // 150 SAR per sheet
```
```

### ## API Documentation

#### ### Authentication

The platform uses two authentication methods:

1. \*\*Web Dashboard\*\*: Replit Auth (OIDC) with session cookies
2. \*\*Revit Add-in\*\*: API key authentication via Bearer token

#### ### Client Endpoints

| Method | Endpoint                 | Description            |
|--------|--------------------------|------------------------|
| GET    | /api/orders              | List user's orders     |
| POST   | /api/orders              | Create new order       |
| GET    | /api/orders/:id          | Get order details      |
| POST   | /api/orders/:id/checkout | Create Stripe checkout |

#### ### Add-in Endpoints

| Method | Endpoint                              | Description              |
|--------|---------------------------------------|--------------------------|
| GET    | /api/addin/validate                   | Validate API key         |
| POST   | /api/addin/create-order               | Create order with sheets |
| GET    | /api/addin/orders                     | List user orders         |
| GET    | /api/addin/orders/:id/status          | Get order status         |
| POST   | /api/addin/orders/:id/upload-url      | Get upload URL           |
| POST   | /api/addin/orders/:id/upload-complete | Mark upload complete     |
| GET    | /api/addin/orders/:id/download-url    | Get download URL         |

#### ### Admin Endpoints

| Method | Endpoint                     | Description         |
|--------|------------------------------|---------------------|
| GET    | /api/admin/orders            | List all orders     |
| PATCH  | /api/admin/orders/:id        | Update order status |
| GET    | /api/admin/clients           | List all clients    |
| POST   | /api/admin/orders/:id/upload | Upload deliverable  |

### ## Revit Add-in Setup

#### ### Step 1: Generate API Key

1. Log in to the web dashboard
2. Go to \*\*Settings\*\* → \*\*API Keys\*\*
3. Click the \*\*\*\* button to create a new key
4. Copy and save the key (shown only once)

#### ### Step 2: Configure the Add-in

1. Open `revit-addin/LOD400Uploader/App.cs`

```

2. Update the API URL:
```csharp
ApiBaseUrl = Environment.GetEnvironmentVariable("LOD400_API_URL")
?? "https://your-deployed-url.replit.app";
```

Step 3: Update Revit References

In `LOD400Uploader.csproj`, update paths to match your Revit installation:

```xml
<Reference Include="RevitAPI">
  <HintPath>C:\Program Files\Autodesk\Revit 2024\RevitAPI.dll</HintPath>
</Reference>
<Reference Include="RevitAPIUI">
  <HintPath>C:\Program Files\Autodesk\Revit 2024\RevitAPIUI.dll</HintPath>
</Reference>
```

Step 4: Build

1. Open `LOD400Uploader.csproj` in Visual Studio 2022
2. Build in **Release** mode
3. Output will be in `bin\Release\net48\`

Step 5: Install

Copy these files to your Revit add-ins folder:
- `LOD400Uploader.dll`
- `Newtonsoft.Json.dll`
- `LOD400Uploader.addin`

Add-ins folder location:
```
%APPDATA%\Autodesk\Revit\Addins\2024\
```

Step 6: Use

1. Open Revit → find the **LOD 400** tab
2. Click **Upload Sheets**
3. Enter your API key when prompted
4. Select sheets, pay, and upload

User Workflow

Client Workflow

1. **Login**: Access web dashboard via Replit Auth
2. **Generate API Key**: Create key in Settings for add-in
3. **Open Revit**: Launch Revit with the add-in installed
4. **Enter API Key**: First-time login in add-in
5. **Select Sheets**: Choose sheets to upgrade
6. **Pay**: Complete Stripe checkout in browser
7. **Upload**: Add-in uploads model automatically
8. **Wait**: Admin processes the order
9. **Download**: Get completed deliverables

Admin Workflow

1. **View Orders**: See incoming paid orders
2. **Download Input**: Get client's uploaded model
3. **Process**: Upgrade model to LOD 400
4. **Upload Output**: Upload completed files
5. **Mark Complete**: Update order status

Database Schema

```typescript
// Users
// Users
users: {
  id: string, // Replit user ID
  email: string,
  firstName: string,
  lastName: string,
  isAdmin: boolean
}

// Orders
orders: {
  id: serial,
  userId: string,
  sheetNames: string[],
  sheetCount: number,
  totalPrice: number,
  status: 'pending' | 'paid' | 'uploaded' | 'processing' | 'complete',
  stripeSessionId: string,
  createdAt: timestamp
}

// Files
files: {
  id: serial,
  orderId: number,
  filename: string,
  storagePath: string,
  type: 'input' | 'output',
  uploadedAt: timestamp
}

// API Keys

```

```
apiKeys: {
  id: serial,
  userId: string,
  name: string,
  keyHash: string,    // SHA-256 hash
  keyPrefix: string,  // First 8 chars for display
  lastUsedAt: timestamp,
  createdAt: timestamp
}
``,`
```

Migrating to Other Platforms

This project uses standard technologies and can be deployed elsewhere:

Database

- Export schema from `shared/schema.ts`
- Works with any PostgreSQL provider (Supabase, Railway, Render, AWS RDS)

File Storage

- Replace GCS with any S3-compatible storage
- Update `server/storage.ts` with new provider

Authentication

- Replace Replit Auth with:
 - Supabase Auth
 - Auth0
 - NextAuth.js
 - Custom JWT implementation

Hosting

Compatible with:

- Vercel (serverless functions for API)
- Railway
- Render
- DigitalOcean App Platform
- AWS/GCP/Azure

Scripts

Script	Description
`npm run dev`	Start development server
`npm run build`	Build for production
`npm start`	Start production server
`npm run db:push`	Push schema to database
`npm run db:studio`	Open Drizzle Studio

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

License

This project is proprietary software. All rights reserved.

Support

For technical support or questions, please contact the development team.

Built with Replit Agent

File: replit.md

LOD 400 Delivery Platform

Overview

The LOD 400 Delivery Platform is a professional B2B SaaS application for upgrading BIM (Building Information Modeling) models from LOD 300 to LOD 400 specifically.

User Preferences

Preferred communication style: Simple, everyday language.

System Architecture

Frontend Architecture

****Framework**:** React with TypeScript, using Vite as the build tool and bundler.

****UI Component Library**:** shadcn/ui (Radix UI primitives) with Tailwind CSS for styling. The design follows a modern SaaS dashboard pattern inspired by Linear.

****Routing**:** wouter for client-side routing with role-based layouts (admin vs. client dashboards).

****State Management**:** TanStack Query (React Query) for server state management, with custom query client configuration for authentication handling.

****Theme System**:** Dark/light mode theming using CSS variables and a custom ThemeProvider context.

****Design Tokens**:**

- Typography: Inter font family via Google Fonts CDN
- Spacing: Tailwind's consistent spacing scale (2, 4, 6, 8, 12, 16, 24)
- Colors: HSL-based color system with CSS variables for theme flexibility
- Components: Standardized card layouts, badges, buttons with hover elevation effects

Backend Architecture

****Runtime**:** Node.js with Express.js framework, built using esbuild for production.

****Language**:** TypeScript (ES modules) for type safety across the stack.

****API Design**:** RESTful HTTP endpoints with role-based access control:

- `/api/orders` - Client order management
- `/api/admin/*` - Administrative functions (orders, clients, file uploads)
- `/api/stripe/webhook` - Payment processing webhooks
- `/api/files/:id/download` - Secure file downloads

****Authentication**:** Replit Auth (OpenID Connect) with Passport.js strategy. Session-based authentication using express-session with PostgreSQL session store.

****Authorization**:** Role-based access control with `isAdmin` flag. Middleware functions (`isAuthenticated`, `isAdmin`) protect routes.

****File Uploads**:** Uppy file uploader with AWS S3-compatible storage through Google Cloud Storage (via Replit's Object Storage sidecar). Large file support (up to 5GB).

Data Storage Architecture

****Primary Database**:** PostgreSQL (via Neon serverless with WebSocket connection pooling).

****ORM**:** Drizzle ORM with schema-first design approach. Schema co-located in `shared/schema.ts` for type sharing between client and server.

****Database Schema Design**:**

- `users` - User profiles with Replit Auth integration (id, email, firstName, lastName, isAdmin)
- `orders` - Order records with status tracking (pending → paid → uploaded → processing → complete)
- `files` - File metadata and storage paths (type: input/output, orderId foreign key)
- `api_keys` - API authentication for Revit add-in (hashed keys, last used tracking)
- `sessions` - Session persistence for Replit Auth
- Stripe schema managed by stripe-replit-sync package

****Session Storage**:** PostgreSQL-backed sessions using connect-pg-simple for reliable session persistence.

****Object Storage**:** Google Cloud Storage for file uploads/downloads, accessed through Replit's sidecar authentication mechanism. Organized by environment-specific buckets.

Authentication & Authorization

****User Authentication**:** Replit Auth (OIDC) for web dashboard login with automatic user provisioning on first login.

****API Authentication**:** Custom API key system for Revit add-in integration:

- API keys generated via web dashboard
- SHA-256 hashing for secure storage
- Bearer token validation on API endpoints
- Last-used timestamp tracking

****Session Management**:** 7-day session TTL with httpOnly, secure cookies. PostgreSQL-backed session store for scalability.

****Authorization Levels**:**

- ****Client**:** Can create orders, upload files, view own orders
- ****Admin**:** Full access to all orders, client management, file operations, order status updates

Payment Processing

****Payment Provider**:** Stripe with stripe-replit-sync for managed webhook handling.

****Payment Flow**:**

1. Client selects sheets → price calculation (150 SAR per sheet)
2. Stripe Checkout Session creation with order metadata
3. Payment confirmation via webhook → order status update to "paid"
4. Automatic webhook endpoint creation and management

****Webhook Handling**:** Managed webhooks through stripe-replit-sync package with automatic endpoint registration and signature verification. Custom handlers in `shared/webhooks`.

****Price Configuration**:** Centralized in `shared/schema.ts` (PRICE_PER_SHEET_SAR = 150).

External Dependencies

Third-Party Services

Replit Infrastructure:

- Replit Auth (OIDC) - Authentication provider
- Replit Object Storage - File storage via Google Cloud Storage sidecar
- Replit Connectors - Stripe credential management

Payment Processing:

- Stripe - Payment gateway and checkout
- stripe-replit-sync - Webhook management and Stripe data synchronization

Database:

- Neon Serverless PostgreSQL - Primary database with WebSocket support
- connect-pg-simple - PostgreSQL session store adapter

File Storage:

- Google Cloud Storage - Object storage backend (via Replit sidecar)
- @uppy/aws-s3 - S3-compatible upload library

UI Component Libraries

Radix UI Primitives (shadcn/ui foundation):

- Dialog, Dropdown Menu, Popover, Toast, Tabs
- Accordion, Alert Dialog, Checkbox, Radio Group
- Navigation Menu, Sidebar, Command Menu
- All primitives used with Tailwind CSS styling

Utility Libraries:

- Tailwind CSS - Utility-first styling
- class-variance-authority - Component variant management
- clsx + tailwind-merge - Conditional class merging
- date-fns - Date formatting and manipulation

Development Tools

Build Pipeline:

- Vite - Frontend build tool and dev server
- esbuild - Server-side bundling for production
- TypeScript - Type checking across full stack
- Drizzle Kit - Database migrations and schema push

Code Quality:

- PostCSS with Autoprefixer - CSS processing
- Replit-specific plugins for dev environment integration

Revit Integration

Revit Add-in (C#/.NET Framework 4.8):

- Located in `./revit-addin` directory
- Communicates with platform via REST API using API keys
- Handles sheet selection, model packaging, upload progress
- Supports Revit 2024 (adaptable for 2020-2025)

Recent Changes

December 1, 2025:

- Added complete API Keys management UI to Settings page
- Users can create, view, and delete API keys for Revit add-in authentication
- Keys are SHA-256 hashed in database with only prefix visible after creation
- Fixed Uppy CSS imports for v5.x compatibility (using subpath imports)
- State management improved for API key creation (proper reset before new creation)
- Clipboard error handling added for copy functionality

MVP Completion Status

The LOD 400 Delivery Platform MVP is now complete with:

Frontend:

- Landing page with pricing and feature highlights
- User dashboard with order overview
- Order management with status tracking
- Admin dashboard for order processing
- API Keys management UI in Settings
- Dark/light theme support
- Responsive design

Backend:

- RESTful API with role-based access control
- Stripe payment integration with managed webhooks
- Object storage for file uploads/downloads
- API key authentication for Revit add-in
- Order lifecycle management

Revit Add-in (Source code):

- Complete C# source code for Visual Studio compilation
- WPF dialogs for login, sheet selection, and status
- API key authentication
- Model packaging with workshared support
- Upload with progress tracking
- Payment flow via browser redirect

Tested Flows:

- User registration and authentication (Replit OIDC)
- API key creation and validation
- Order creation and status tracking
- Payment processing via Stripe
- Admin file upload and order management

File: revit-addin/LOD400Uploader/App.cs

```
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using System;
using System.Reflection;
using System.Windows.Media.Imaging;

namespace LOD400Uploader
{
    /// <summary>
    /// Revit External Application - Entry point for the add-in
    /// </summary>
    public class App : IExternalApplication
    {
        public static string ApiBaseUrl { get; private set; }
        public static string AuthToken { get; set; }

        public Result OnStartup(UIControlledApplication application)
        {
            try
            {
                // Set API base URL from environment or default
                ApiBaseUrl = Environment.GetEnvironmentVariable("LOD400_API_URL")
                    ?? "https://YOUR-REPLIT-URL.replit.app";

                // Create ribbon tab
                string tabName = "LOD 400";
                application.CreateRibbonTab(tabName);

                // Create ribbon panel
                RibbonPanel ribbonPanel = application.CreateRibbonPanel(tabName, "Sheet Upgrade");

                // Get assembly path for button
                string assemblyPath = Assembly.GetExecutingAssembly().Location;

                // Create push button for upload command
                PushButtonData uploadButtonData = new PushButtonData(
                    "UploadSheets",
                    "Upload\nSheets",
                    assemblyPath,
                    "LOD400Uploader.Commands.UploadSheetsCommand"
                );
                uploadButtonData.ToolTip = "Select sheets and upload for LOD 400 upgrade";
                uploadButtonData.LongDescription = "Opens the LOD 400 Upload dialog where you can select sheets from your model, " +
                    "review pricing (150 SAR per sheet), pay securely via Stripe, and upload your model for professional LOD 400 upgrade.";

                PushButton uploadButton = ribbonPanel.AddItem(uploadButtonData) as PushButton;

                // Create push button for check status
                PushButtonData statusButtonData = new PushButtonData(
                    "CheckStatus",
                    "Check\nStatus",
                    assemblyPath,
                    "LOD400Uploader.Commands.CheckStatusCommand"
                );
                statusButtonData.ToolTip = "Check order status and download deliverables";
                statusButtonData.LongDescription = "View your existing orders, check processing status, and download completed LOD 400 deliverables.";

                PushButton statusButton = ribbonPanel.AddItem(statusButtonData) as PushButton;

                return Result.Succeeded;
            }
            catch (Exception ex)
            {
                TaskDialog.Show("Error", $"Failed to initialize LOD 400 Add-in: {ex.Message}");
                return Result.Failed;
            }
        }

        public Result OnShutdown(UIControlledApplication application)
        {
            return Result.Succeeded;
        }
    }
}
```

File: revit-addin/LOD400Uploader/Commands/CheckStatusCommand.cs

```
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using LOD400Uploader.Views;
using System;

namespace LOD400Uploader.Commands
{
    /// <summary>
    /// Command to check order status and download deliverables
    /// </summary>
    [Transaction(TransactionMode.Manual)]
    public class CheckStatusCommand : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            try
            {
                // Open the status dialog
                var dialog = new StatusDialog();
                dialog.ShowDialog();

                return Result.Succeeded;
            }
            catch (Exception ex)
            {
                message = ex.Message;
                TaskDialog.Show("Error", $"An error occurred: {ex.Message}");
                return Result.Failed;
            }
        }
    }
}
```

File: revit-addin/LOD400Uploader/Commands/UploadSheetsCommand.cs

```
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using LOD400Uploader.Views;
using System;

namespace LOD400Uploader.Commands
{
    /// <summary>
    /// Command to open the sheet selection and upload dialog
    /// </summary>
    [Transaction(TransactionMode.Manual)]
    public class UploadSheetsCommand : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            try
            {
                UIApplication uiApp = commandData.Application;
                UIDocument uiDoc = uiApp.ActiveUIDocument;
                Document doc = uiDoc.Document;

                // Check if document is saved
                if (string.IsNullOrEmpty(doc.PathName))
                {
                    TaskDialog.Show("Save Required",
                        "Please save your Revit model before uploading.\n\n" +
                        "Go to File > Save As to save your model first.");
                    return Result.Cancelled;
                }

                // Open the main upload dialog
                var dialog = new UploadDialog(doc);
                dialog.ShowDialog();

                return Result.Succeeded;
            }
            catch (Exception ex)
            {
                message = ex.Message;
                TaskDialog.Show("Error", $"An error occurred: {ex.Message}");
                return Result.Failed;
            }
        }
    }
}
```

File: revit-addin/LOD400Uploader/LOD400Uploader.addin

```
<?xml version="1.0" encoding="utf-8"?>
<RevitAddIns>
  <AddIn Type="Application">
    <Name>LOD 400 Uploader</Name>
    <Assembly>LOD400Uploader.dll</Assembly>
    <FullClassName>LOD400Uploader.App</FullClassName>
    <AddInId>A1B2C3D4-E5F6-4A5B-8C9D-0E1F2A3B4C5D</AddInId>
    <VendorId>LOD400</VendorId>
    <VendorDescription>LOD 400 Delivery Platform</VendorDescription>
  </AddIn>
</RevitAddIns>
```

File: revit-addin/LOD400Uploader/LOD400Uploader.csproj

```
<!-- Project Sdk="Microsoft.NET.Sdk" -->

<!-- PropertyGroup -->
  <!-- TargetFramework --> net48<!-- /TargetFramework -->
  <!-- LangVersion --> latest<!-- /LangVersion -->
  <!-- UseWPF --> true<!-- /UseWPF -->
  <!-- AssemblyTitle --> LOD 400 Uploader<!-- /AssemblyTitle -->
  <!-- Product --> LOD 400 Delivery Add-in<!-- /Product -->
  <!-- Description --> Revit add-in for uploading BIM models for LOD 400 upgrade service<!-- /Description -->
  <!-- Company --> LOD 400 Delivery<!-- /Company -->
  <!-- Copyright --> Copyright © 2024<!-- /Copyright -->
  <!-- OutputType --> Library<!-- /OutputType -->
  <!-- RootNamespace --> LOD400Uploader<!-- /RootNamespace -->
  <!-- AssemblyName --> LOD400Uploader<!-- /AssemblyName -->
<!-- /PropertyGroup -->

<!-- ItemGroup -->
  <!-- Revit API references - Update paths for your Revit version -->
  <!-- Reference Include="RevitAPI" -->
    <!-- HintPath --> C:\Program Files\Autodesk\Revit 2024\RevitAPI.dll<!-- /HintPath -->
    <!-- Private --> False<!-- /Private -->
  <!-- /Reference -->
  <!-- Reference Include="RevitAPIUI" -->
    <!-- HintPath --> C:\Program Files\Autodesk\Revit 2024\RevitAPIUI.dll<!-- /HintPath -->
    <!-- Private --> False<!-- /Private -->
  <!-- /Reference -->
<!-- /ItemGroup -->

<!-- ItemGroup -->
  <!-- PackageReference Include="Newtonsoft.Json" Version="13.0.3" -->
<!-- /ItemGroup -->

<!-- ItemGroup -->
  <!-- Resource Include="Resources\icon.png" -->
<!-- /ItemGroup -->

<!-- /Project -->
```

File: revit-addin/LOD400Uploader/Models/Order.cs

```
using System;
using System.Collections.Generic;
using Newtonsoft.Json;

namespace LOD400Uploader.Models
{
    public class Order
    {
        [JsonProperty("id")]
        public string Id { get; set; }

        [JsonProperty("userId")]
        public string UserId { get; set; }

        [JsonProperty("sheetCount")]
        public int SheetCount { get; set; }

        [JsonProperty("totalPriceSar")]
        public int TotalPriceSar { get; set; }

        [JsonProperty("status")]
        public string Status { get; set; }

        [JsonProperty("stripeSessionId")]
        public string StripeSessionId { get; set; }

        [JsonProperty("stripePaymentIntentId")]
        public string StripePaymentIntentId { get; set; }

        [JsonProperty("notes")]
        public string Notes { get; set; }

        [JsonProperty("createdAt")]
        public DateTime? CreatedAt { get; set; }

        [JsonProperty("updatedAt")]
        public DateTime? UpdatedAt { get; set; }

        [JsonProperty("paidAt")]
        public DateTime? PaidAt { get; set; }

        [JsonProperty("uploadedAt")]
        public DateTime? UploadedAt { get; set; }

        [JsonProperty("completedAt")]
        public DateTime? CompletedAt { get; set; }

        [JsonProperty("files")]
        public List<OrderFile> Files { get; set; }
    }

    public class OrderFile
    {
        [JsonProperty("id")]
        public string Id { get; set; }

        [JsonProperty("orderId")]
        public string OrderId { get; set; }

        [JsonProperty("fileType")]
        public string FileType { get; set; }

        [JsonProperty("fileName")]
        public string FileName { get; set; }

        [JsonProperty("fileSize")]
        public long? FileSize { get; set; }

        [JsonProperty("storageKey")]
        public string StorageKey { get; set; }

        [JsonProperty("mimeType")]
        public string MimeType { get; set; }

        [JsonProperty("createdAt")]
        public DateTime? CreatedAt { get; set; }
    }

    public class CreateOrderRequest
    {
        [JsonProperty("sheetCount")]
        public int SheetCount { get; set; }
    }

    public class CreateOrderResponse
    {
        [JsonProperty("order")]
        public Order Order { get; set; }

        [JsonProperty("checkoutUrl")]
        public string CheckoutUrl { get; set; }
    }

    public class UploadUrlResponse
    {
        [JsonProperty("uploadURL")]
        public string UploadURL { get; set; }
    }
}
```

```
}

public class DownloadUrlResponse
{
    [JsonProperty("downloadURL")]
    public string DownloadURL { get; set; }

    [JsonProperty("fileName")]
    public string FileName { get; set; }
}

public class UploadCompleteRequest
{
    [JsonProperty("fileName")]
    public string FileName { get; set; }

    [JsonProperty("fileSize")]
    public long FileSize { get; set; }

    [JsonProperty("uploadURL")]
    public string UploadURL { get; set; }
}
}
```

File: revit-addin/LOD400Uploader/Services/ApiService.cs

```
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;
using LOD400Uploader.Models;
using System.Collections.Generic;
using System.Net.Http.Headers;

namespace LOD400Uploader.Services
{
    public class ApiService
    {
        private readonly HttpClient _httpClient;
        private readonly string _baseUrl;
        private string _apiKey;

        public ApiService()
        {
            _httpClient = new HttpClient();
            _httpClient.Timeout = TimeSpan.FromMinutes(10);
            _baseUrl = App.ApiBaseUrl;
        }

        public void SetApiKey(string apiKey)
        {
            _apiKey = apiKey;
            _httpClient.DefaultRequestHeaders.Clear();
            _httpClient.DefaultRequestHeaders.Add("X-API-Key", apiKey);
        }

        public bool HasApiKey => !string.IsNullOrEmpty(_apiKey);

        public async Task<bool> ValidateApiKeyAsync(string apiKey)
        {
            try
            {
                using (var client = new HttpClient())
                {
                    client.DefaultRequestHeaders.Add("X-API-Key", apiKey);
                    var response = await client.GetAsync($"{_baseUrl}/api/addin/validate");
                    return response.IsSuccessStatusCode;
                }
            }
            catch
            {
                return false;
            }
        }

        public async Task<CreateOrderResponse> CreateOrderAsync(int sheetCount)
        {
            EnsureApiKey();
            var request = new CreateOrderRequest { SheetCount = sheetCount };
            var json = JsonConvert.SerializeObject(request);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync($"{_baseUrl}/api/addin/create-order", content);
            response.EnsureSuccessStatusCode();

            var responseJson = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<CreateOrderResponse>(responseJson);
        }

        public async Task<Order> PollOrderStatusAsync(string orderId, int maxAttempts = 60, int delayMs = 2000)
        {
            EnsureApiKey();
            for (int i = 0; i < maxAttempts; i++)
            {
                var order = await GetOrderStatusAsync(orderId);
                if (order.Status == "paid" || order.Status == "uploaded" ||
                    order.Status == "processing" || order.Status == "complete")
                {
                    return order;
                }
                await Task.Delay(delayMs);
            }
            throw new TimeoutException("Payment verification timed out. Please check your order status manually.");
        }

        public async Task<string> GetUploadUrlAsync(string orderId, string fileName)
        {
            EnsureApiKey();
            var request = new { fileName = fileName };
            var json = JsonConvert.SerializeObject(request);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync($"{_baseUrl}/api/addin/orders/{orderId}/upload-url", content);
            response.EnsureSuccessStatusCode();

            var responseJson = await response.Content.ReadAsStringAsync();
            var result = JsonConvert.DeserializeObject<UploadUrlResponse>(responseJson);
            return result.UploadURL;
        }

        public async Task MarkUploadCompleteAsync(string orderId, string fileName, long fileSize, string uploadUrl)
        {
            EnsureApiKey();
            var request = new { orderId = orderId, fileName = fileName, fileSize = fileSize, uploadUrl = uploadUrl };
            var json = JsonConvert.SerializeObject(request);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync($"{_baseUrl}/api/addin/orders/{orderId}/mark-complete", content);
            response.EnsureSuccessStatusCode();

            var responseJson = await response.Content.ReadAsStringAsync();
            var result = JsonConvert.DeserializeObject<MarkUploadCompleteResponse>(responseJson);
            return result.UploadURL;
        }
    }
}
```

```

        EnsureApiKey();
        var request = new UploadCompleteRequest
        {
            FileName = fileName,
            FileSize = fileSize,
            UploadURL = uploadUrl
        };
        var json = JsonConvert.SerializeObject(request);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync($"({_baseUrl})/api/addin/orders/{orderId}/upload-complete", content);
        response.EnsureSuccessStatusCode();
    }

    public async Task<Order> GetOrderStatusAsync(string orderId)
    {
        EnsureApiKey();
        var response = await _httpClient.GetAsync($"({_baseUrl})/api/addin/orders/{orderId}/status");
        response.EnsureSuccessStatusCode();

        var responseJson = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject<Order>(responseJson);
    }

    public async Task<List<Order>> GetOrdersAsync()
    {
        EnsureApiKey();
        var response = await _httpClient.GetAsync($"({_baseUrl})/api/addin/orders");
        response.EnsureSuccessStatusCode();

        var responseJson = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject<List<Order>>(responseJson);
    }

    public async Task<DownloadUrlResponse> GetDownloadUrlAsync(string orderId)
    {
        EnsureApiKey();
        var response = await _httpClient.GetAsync($"({_baseUrl})/api/addin/orders/{orderId}/download-url");
        response.EnsureSuccessStatusCode();

        var responseJson = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject<DownloadUrlResponse>(responseJson);
    }

    public async Task UploadFileAsync(string uploadUrl, byte[] fileData, Action<int> progressCallback)
    {
        using (var client = new HttpClient())
        {
            client.Timeout = TimeSpan.FromHours(2);

            var content = new ByteArrayContent(fileData);
            content.Headers.ContentType = new MediaTypeHeaderValue("application/zip");

            progressCallback?.Invoke(0);
            var response = await client.PutAsync(uploadUrl, content);
            response.EnsureSuccessStatusCode();
            progressCallback?.Invoke(100);
        }
    }

    private void EnsureApiKey()
    {
        if (string.IsNullOrEmpty(_apiKey))
        {
            throw new InvalidOperationException("API key not set. Please log in first.");
        }
    }
}

```

File: revit-addin/LOD400Uploader/Services/PackagingService.cs

```
using System;
using System.IO;
using System.IO.Compression;
using System.Collections.Generic;
using Autodesk.Revit.DB;

namespace LOD400Uploader.Services
{
    public class PackagingService
    {
        private string _tempDir;
        private string _zipPath;

        public string PackageModel(Document document, List<ElementId> selectedSheetIds, Action<int, string> progressCallback)
        {
            _tempDir = null;
            _zipPath = null;

            try
            {
                progressCallback?.Invoke(5, "Validating model...");

                string originalPath = document.PathName;
                if (string.IsNullOrEmpty(originalPath) || !File.Exists(originalPath))
                {
                    throw new InvalidOperationException("Please save your Revit model before uploading.");
                }

                if (document.IsModified)
                {
                    throw new InvalidOperationException("Please save your changes before uploading. The model has unsaved modifications.");
                }

                _tempDir = Path.Combine(Path.GetTempPath(), "LOD400Upload_" + Guid.NewGuid().ToString("N"));
                Directory.CreateDirectory(_tempDir);

                progressCallback?.Invoke(15, "Preparing model copy...");

                string fileName = Path.GetFileName(originalPath);
                string modelCopyPath = Path.Combine(_tempDir, fileName);

                bool isWorkshared = document.IsWorkshared;

                if (isWorkshared)
                {
                    progressCallback?.Invoke(25, "Detaching workshared model...");
                    var detachOption = new SaveAsOptions
                    {
                        OverwriteExistingFile = true,
                        MaximumBackups = 1
                    };

                    var worksharingOptions = new WorksharingSaveAsOptions();
                    worksharingOptions.SaveAsCentral = false;
                    detachOption.SetWorksharingOptions(worksharingOptions);

                    var tempSavePath = Path.Combine(_tempDir, "temp_" + fileName);
                    document.SaveAs(tempSavePath, detachOption);

                    if (File.Exists(tempSavePath))
                    {
                        File.Move(tempSavePath, modelCopyPath);
                    }
                }
                else
                {
                    progressCallback?.Invoke(25, "Copying model file...");
                    File.Copy(originalPath, modelCopyPath, true);
                }

                progressCallback?.Invoke(50, "Creating sheet manifest...");
                string manifestPath = Path.Combine(_tempDir, "sheets.json");
                CreateSheetManifest(document, selectedSheetIds, manifestPath);

                progressCallback?.Invoke(70, "Creating ZIP package...");
                _zipPath = Path.Combine(Path.GetTempPath(), $"LOD400_Upload_{DateTime.Now:yyyyMMdd_HH:mm:ss}.zip");

                if (File.Exists(_zipPath))
                {
                    File.Delete(_zipPath);
                }

                ZipFile.CreateFromDirectory(_tempDir, _zipPath, CompressionLevel.Optimal, false);

                progressCallback?.Invoke(90, "Cleaning up temporary files...");
                CleanupTempDirectory();

                progressCallback?.Invoke(100, "Package created successfully");

                return _zipPath;
            }
            catch (Exception)
            {
                CleanupTempDirectory();
                CleanupZipFile();
                throw;
            }
        }
    }
}
```

```

}

private void CreateSheetManifest(Document document, List<ElementId> selectedSheetIds, string manifestPath)
{
    var sheets = new List<object>();

    foreach (var sheetId in selectedSheetIds)
    {
        var sheet = document.GetElement(sheetId) as ViewSheet;
        if (sheet != null)
        {
            sheets.Add(new
            {
                id = sheetId.IntegerValue,
                number = sheet.SheetNumber ?? "",
                name = sheet.Name ?? "",
                revisionNumber = GetParameterValue(sheet, BuiltInParameter.SHEET_CURRENT_REVISION),
                revisionDate = GetParameterValue(sheet, BuiltInParameter.SHEET_CURRENT_REVISION_DATE),
                drawnBy = GetParameterValue(sheet, BuiltInParameter.SHEET_DRAWN_BY),
                checkedBy = GetParameterValue(sheet, BuiltInParameter.SHEET_CHECKED_BY)
            });
        }
    }

    var manifest = new
    {
        projectName = document.Title ?? "Untitled",
        projectNumber = GetProjectInfo(document, BuiltInParameter.PROJECT_NUMBER),
        clientName = GetProjectInfo(document, BuiltInParameter.CLIENT_NAME),
        exportDate = DateTime.UtcNow.ToString("o"),
        revitVersion = document.Application?.VersionNumber ?? "Unknown",
        isWorkshared = document.IsWorkshared,
        sheetCount = sheets.Count,
        sheets = sheets
    };

    string json = Newtonsoft.Json.JsonConvert.SerializeObject(manifest, Newtonsoft.Json.Formatting.Indented);
    File.WriteAllText(manifestPath, json);
}

private string GetParameterValue(Element element, BuiltInParameter param)
{
    try
    {
        var p = element?.get_Parameter(param);
        return p?.AsString() ?? "";
    }
    catch
    {
        return "";
    }
}

private string GetProjectInfo(Document document, BuiltInParameter param)
{
    try
    {
        var projectInfo = document?.ProjectInformation;
        var p = projectInfo?.get_Parameter(param);
        return p?.AsString() ?? "";
    }
    catch
    {
        return "";
    }
}

public long GetFileSize(string filePath)
{
    if (string.IsNullOrEmpty(filePath) || !File.Exists(filePath))
    {
        throw new FileNotFoundException("Package file not found.", filePath);
    }
    return new FileInfo(filePath).Length;
}

public byte[] ReadFileBytes(string filePath)
{
    if (string.IsNullOrEmpty(filePath) || !File.Exists(filePath))
    {
        throw new FileNotFoundException("Package file not found.", filePath);
    }
    return File.ReadAllBytes(filePath);
}

public void Cleanup(string filePath)
{
    _zipPath = filePath;
    CleanupZipFile();
}

private void CleanupTempDirectory()
{
    if (!string.IsNullOrEmpty(_tempDir) && Directory.Exists(_tempDir))
    {
        try
        {
            Directory.Delete(_tempDir, true);
        }
        catch
        {
        }
    }
}

```

```
        }
        _tempDir = null;
    }
}

private void CleanupZipFile()
{
    if (!string.IsNullOrEmpty(_zipPath) && File.Exists(_zipPath))
    {
        try
        {
            File.Delete(_zipPath);
        }
        catch
        {
        }
        _zipPath = null;
    }
}
}
```

File: revit-addin/LOD400Uploader/Views/LoginDialog.xaml

```
<Window x:Class="LOD400Uploader.Views.LoginDialog"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="LOD 400 - Login"
        Height="300" Width="450"
        WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        Background="#F5F5F5">

    <Window.Resources>

        <Style TargetType="Button" x:Key="PrimaryButton">
            <Setter Property="Background" Value="#2563EB"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Padding" Value="24,12"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="FontWeight" Value="SemiBold"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                CornerRadius="6"
                                Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style.Triggers>
            <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="Background" Value="#1D4ED8"/>
            </Trigger>
            <Trigger Property="IsEnabled" Value="False">
                <Setter Property="Background" Value="#94A3B8"/>
            </Trigger>
        </Style.Triggers>
    </Window.Resources>

    <Grid Margin="32">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <StackPanel Grid.Row="0">
            <TextBlock Text="LOD 400 Delivery" FontSize="24" FontWeight="Bold" Foreground="#111827"/>
            <TextBlock Text="Enter your API key to connect" FontSize="14" Foreground="#6B7280" Margin="0,8,0,0"/>
        </StackPanel>

        <StackPanel Grid.Row="1" Margin="0,24,0,0">
            <TextBlock Text="API Key" FontSize="13" FontWeight="SemiBold" Foreground="#374151" Margin="0,0,0,8"/>
            <TextBox x:Name="ApiKeyTextBox" FontSize="14" Padding="12,10"
                    Background="White" BorderBrush="#D1D5DB" BorderThickness="1"/>
            <TextBlock x:Name="ErrorText" Text="" FontSize="12" Foreground="#DC2626" Margin="0,8,0,0"
                    Visibility="Collapsed"/>
        </StackPanel>

        <StackPanel Grid.Row="2" Margin="0,16,0,0">
            <TextBlock Text="Get your API key from your account dashboard at:"
                    FontSize="12" Foreground="#6B7280" TextWrapping="Wrap"/>
            <TextBlock x:Name="UrlText" Text="https://your-app.replit.app/settings"
                    FontSize="12" Foreground="#2563EB" Cursor="Hand" Margin="0,4,0,0"
                    MouseDown="UrlText_MouseDown" TextDecorations="Underline"/>
        </StackPanel>

        <StackPanel Grid.Row="3" Orientation="Horizontal" HorizontalAlignment="Right">
            <Button Content="Cancel" Padding="16,10" Background="White" Foreground="#374151"
                    BorderBrush="#D1D5DB" BorderThickness="1" Margin="0,0,12,0" Click="CancelButton_Click"/>
            <Button x:Name="LoginButton" Content="Connect" Style="{StaticResource PrimaryButton}"
                    Click="LoginButton_Click"/>
        </StackPanel>
    </Grid>
</Window>
```

File: revit-addin/LOD400Uploader/Views/LoginDialog.xaml.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Windows;
using System.Windows.Input;
using LOD400Uploader.Services;

namespace LOD400Uploader.Views
{
    public partial class LoginDialog : Window
    {
        private readonly ApiService _apiService;
        private static readonly string ConfigPath = Path.Combine(
            Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),
            "LOD400Uploader",
            "config.json"
        );

        public bool IsAuthenticated { get; private set; }
        public ApiService AuthenticatedApiService => _apiService;

        public LoginDialog()
        {
            InitializeComponent();
            _apiService = new ApiService();

            UrlText.Text = $"{App.ApiBaseUrl}/settings";

            LoadSavedApiKey();
        }

        private void LoadSavedApiKey()
        {
            try
            {
                if (File.Exists(ConfigPath))
                {
                    var json = File.ReadAllText(ConfigPath);
                    var config = Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(json);
                    string savedKey = config?.apiKey;
                    if (!string.IsNullOrEmpty(savedKey))
                    {
                        ApiKeyTextBox.Text = savedKey;
                    }
                }
            }
            catch
            {
            }
        }

        private void SaveApiKey(string apiKey)
        {
            try
            {
                var dir = Path.GetDirectoryName(ConfigPath);
                if (!string.IsNullOrEmpty(dir) && !Directory.Exists(dir))
                {
                    Directory.CreateDirectory(dir);
                }

                var config = new { apiKey = apiKey };
                var json = Newtonsoft.Json.JsonConvert.SerializeObject(config);
                File.WriteAllText(ConfigPath, json);
            }
            catch
            {
            }
        }

        private async void LoginButton_Click(object sender, RoutedEventArgs e)
        {
            var apiKey = ApiKeyTextBox.Text?.Trim();

            if (string.IsNullOrEmpty(apiKey))
            {
                ShowError("Please enter your API key.");
                return;
            }

            LoginButton.IsEnabled = false;
            LoginButton.Content = "Connecting...";
            ErrorText.Visibility = Visibility.Collapsed;

            try
            {
                bool isValid = await _apiService.ValidateApiKeyAsync(apiKey);

                if (isValid)
                {
                    _apiService.SetApiKey(apiKey);
                    SaveApiKey(apiKey);
                    IsAuthenticated = true;
                    DialogResult = true;
                    Close();
                }
                else
                {
                }
            }
        }
    }
}
```

```

        {
            ShowError("Invalid API key. Please check and try again.");
        }
    }
    catch (Exception ex)
    {
        ShowError($"Connection failed: {ex.Message}");
    }
    finally
    {
        LoginButton.IsEnabled = true;
        LoginButton.Content = "Connect";
    }
}

private void ShowError(string message)
{
    ErrorText.Text = message;
    ErrorText.Visibility = Visibility.Visible;
}

private void CancelButton_Click(object sender, RoutedEventArgs e)
{
    DialogResult = false;
    Close();
}

private void UrlText_MouseDown(object sender, MouseButtonEventArgs e)
{
    try
    {
        Process.Start(new ProcessStartInfo
        {
            FileName = UrlText.Text,
            UseShellExecute = true
        });
    }
    catch
    {
    }
}
}
}

```

File: revit-addin/LOD400Uploader/Views/StatusDialog.xaml

```
<!-- Window -->
<Window x:Class="LOD400Uploader.Views.StatusDialog"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="My Orders - LOD 400"
        Height="500" Width="700"
        WindowStartupLocation="CenterScreen"
        Background="#F5F5F5">

    <!-- Window.Resources -->
    <Window.Resources>
        <Style TargetType="Button" x:Key="PrimaryButton">
            <Setter Property="Background" Value="#2563EB"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Padding" Value="16,8"/>
            <Setter Property="FontSize" Value="13"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                CornerRadius="4"
                                Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="Button" x:Key="SecondaryButton">
            <Setter Property="Background" Value="White"/>
            <Setter Property="Foreground" Value="#374151"/>
            <Setter Property="Padding" Value="16,8"/>
            <Setter Property="FontSize" Value="13"/>
            <Setter Property="BorderBrush" Value="#D1D5DB"/>
            <Setter Property="BorderThickness" Value="1"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="4"
                                Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="TextBlock">
            <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="Background" Value="#F3F4F6"/>
            </Trigger>
        </Style>
    </Window.Resources>

    <!-- Grid -->
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <!-- Header -->
        <Border Grid.Row="0" Background="White" BorderBrush="#E5E7EB" BorderThickness="0,0,0,1" Padding="20,16">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <StackPanel Grid.Column="0">
                    <TextBlock Text="My Orders" FontSize="20" FontWeight="Bold" Foreground="#111827"/>
                    <TextBlock Text="View your order history and download deliverables"
                             FontSize="13" Foreground="#6B7280" Margin="0,4,0,0"/>
                </StackPanel>

                <Button Grid.Column="1" Content="Refresh" Style="{StaticResource SecondaryButton}"
                        Click="RefreshButton_Click"/>
            </Grid>
        </Border>

        <!-- Orders List -->
        <Grid Grid.Row="1">
            <!-- Loading indicator -->
            <StackPanel x:Name="LoadingPanel" VerticalAlignment="Center" HorizontalAlignment="Center">
                <TextBlock Text="Loading orders..." FontSize="14" Foreground="#6B7280"/>
            </StackPanel>
        </Grid>
    </Grid>
</Window>
```

```

</StackPanel>>

<!-- Empty state -->
<StackPanel x:Name="EmptyPanel" VerticalAlignment="Center" HorizontalAlignment="Center" Visibility="Collapsed">
    <TextBlock Text="No orders yet" FontSize="16" FontWeight="SemiBold" Foreground="#6B7280" HorizontalAlignment="Center"/>
    <TextBlock Text="Create your first order using the Upload Sheets command"
        FontSize="13" Foreground="#9CA3AF" HorizontalAlignment="Center" Margin="0,8,0,0"/>
</StackPanel>>

<!-- Orders list -->
<ListView x:Name="OrdersListView" Margin="16" Visibility="Collapsed"
    SelectionChanged="OrdersListView_SelectionChanged">
    <ListView.View>
        <GridView>
            <GridViewColumn Header="Order ID" Width="200">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding Id}" FontFamily="Consolas" FontSize="11"/>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn Header="Sheets" Width="80" DisplayMemberBinding="{Binding SheetCount}"/>
            <GridViewColumn Header="Price" Width="100">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBlock>
                            <Run Text="{Binding TotalPriceSar, StringFormat=N0}"/>
                            <Run Text=" SAR"/>
                        </TextBlock>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn Header="Status" Width="120">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <Border Padding="8,4" CornerRadius="4"
                            Background="{Binding StatusBackground}">
                            <TextBlock Text="{Binding StatusDisplay}"
                                Foreground="{Binding StatusForeground}"
                                FontSize="12" FontWeight="SemiBold"/>
                        </Border>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn Header="Created" Width="150">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding CreatedAt, StringFormat='{0:MMM dd, yyyy HH:mm}'}"/>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
        </ListView.View>
    </ListView>
</Grid>

<!-- Footer -->
<Border Grid.Row="2" Background="White" BorderBrush="#E5E7EB" BorderThickness="0,1,0,0" Padding="20,16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="Auto" />
        </Grid.ColumnDefinitions>

        <TextBlock x:Name="SelectedOrderText" Grid.Column="0"
            Text="Select an order to view details"
            FontSize="13" Foreground="#6B7280" VerticalAlignment="Center"/>

        <StackPanel Grid.Column="1" Orientation="Horizontal">
            <Button x:Name="DownloadButton" Content="Download Deliverables"
                Style="{StaticResource PrimaryButton}"
                Click="DownloadButton_Click" IsEnabled="False" Margin="0,0,12,0"/>
            <Button Content="Close" Style="{StaticResource SecondaryButton}" Click="CloseButton_Click"/>
        </StackPanel>
    </Grid>
</Border>
</Grid>
</Window>

```

File: revit-addin/LOD400Uploader/Views/StatusDialog.xaml.cs

```
using System;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using LOD400Uploader.Models;
using LOD400Uploader.Services;

namespace LOD400Uploader.Views
{
    public partial class StatusDialog : Window
    {
        private readonly ApiService _apiService;
        private readonly ObservableCollection<OrderViewModel> _orders;
        private bool _isAuthenticated;

        public StatusDialog() : this(null) { }

        public StatusDialog(ApiService apiService)
        {
            InitializeComponent();
            _apiService = apiService ?? new ApiService();
            _orders = new ObservableCollection<OrderViewModel>();
            OrdersListView.ItemsSource = _orders;

            Loaded += async (s, e) => await InitializeAsync();
        }

        private async System.Threading.Tasks.Task InitializeAsync()
        {
            if (!_apiService.HasApiKey)
            {
                var loginDialog = new LoginDialog();
                if (loginDialog.ShowDialog() != true || !loginDialog.IsAuthenticated)
                {
                    Close();
                    return;
                }
                _isAuthenticated = true;
            }
            else
            {
                _isAuthenticated = true;
            }

            await LoadOrdersAsync();
        }

        private async System.Threading.Tasks.Task LoadOrdersAsync()
        {
            if (!_isAuthenticated) return;

            try
            {
                LoadingPanel.Visibility = Visibility.Visible;
                EmptyPanel.Visibility = Visibility.Collapsed;
                OrdersListView.Visibility = Visibility.Collapsed;

                var orders = await _apiService.GetOrdersAsync();

                _orders.Clear();

                if (orders != null)
                {
                    foreach (var order in orders.OrderByDescending(o => o.CreatedAt ?? DateTime.MinValue))
                    {
                        _orders.Add(new OrderViewModel(order));
                    }
                }

                LoadingPanel.Visibility = Visibility.Collapsed;

                if (_orders.Count == 0)
                {
                    EmptyPanel.Visibility = Visibility.Visible;
                }
                else
                {
                    OrdersListView.Visibility = Visibility.Visible;
                }
            }
            catch (Exception ex)
            {
                LoadingPanel.Visibility = Visibility.Collapsed;
                EmptyPanel.Visibility = Visibility.Visible;

                MessageBox.Show(
                    $"Failed to load orders:\n\n{ex.Message}\n\n" +
                    "Please check your connection and try again.",
                    "Error",
                    MessageBoxButton.OK,
                    MessageBoxImage.Error);
            }
        }

        private async void RefreshButton_Click(object sender, RoutedEventArgs e)
        {
            await LoadOrdersAsync();
        }
    }
}
```

```

{
    await LoadOrdersAsync();
}

private void OrdersListView_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    var selectedOrder = OrdersListView.SelectedItem as OrderViewModel;

    if (selectedOrder != null)
    {
        string shortId = selectedOrder.Id?.Length > 8
            ? selectedOrder.Id.Substring(0, 8) + "..."
            : selectedOrder.Id ?? "";

        SelectedOrderText.Text = $"Order: {shortId} | Status: {selectedOrder.StatusDisplay}";
        DownloadButton.IsEnabled = selectedOrder.Status == "complete";
    }
    else
    {
        SelectedOrderText.Text = "Select an order to view details";
        DownloadButton.IsEnabled = false;
    }
}

private async void DownloadButton_Click(object sender, RoutedEventArgs e)
{
    var selectedOrder = OrdersListView.SelectedItem as OrderViewModel;
    if (selectedOrder == null)
    {
        MessageBox.Show("Please select an order first.", "No Order Selected",
            MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    if (selectedOrder.Status != "complete")
    {
        MessageBox.Show("This order is not yet complete. Please wait for processing to finish.",
            "Order Not Ready", MessageBoxButton.OK, MessageBoxImage.Information);
        return;
    }

    try
    {
        DownloadButton.IsEnabled = false;
        DownloadButton.Content = "Downloading...";

        var downloadInfo = await _apiService.GetDownloadUrlAsync(selectedOrder.Id);

        if (downloadInfo == null || string.IsNullOrEmpty(downloadInfo.DownloadURL))
        {
            throw new InvalidOperationException("No download URL available for this order.");
        }

        Process.Start(new ProcessStartInfo
        {
            FileName = downloadInfo.DownloadURL,
            UseShellExecute = true
        });

        MessageBox.Show(
            $"Download started for: {downloadInfo.FileName ?? "deliverables"}\n\n" +
            "The file will be saved to your default downloads folder.",
            "Download Started",
            MessageBoxButton.OK,
            MessageBoxImage.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(
            $"Failed to start download:\n\n{ex.Message}",
            "Download Error",
            MessageBoxButton.OK,
            MessageBoxImage.Error);
    }
    finally
    {
        DownloadButton.Content = "Download Deliverables";
        DownloadButton.IsEnabled = OrdersListView.SelectedItem != null &&&
            ((OrderViewModel)OrdersListView.SelectedItem).Status == "complete";
    }
}

private void CloseButton_Click(object sender, RoutedEventArgs e)
{
    Close();
}

}

public class OrderViewModel
{
    public string Id { get; }
    public int SheetCount { get; }
    public int TotalPriceSar { get; }
    public string Status { get; }
    public DateTime? CreatedAt { get; }

    public string StatusDisplay => Status switch
    {
        "pending" => "Pending Payment",
        "paid" => "Paid",
        "uploaded" => "Uploaded",
    }
}

```

```

        "processing" => "Processing",
        "complete" => "Complete",
        _ => Status ?? "Unknown"
    };

    public Brush StatusBackground => Status switch
    {
        "pending" => new SolidColorBrush(Color.FromRgb(254, 243, 199)),
        "paid" => new SolidColorBrush(Color.FromRgb(219, 234, 254)),
        "uploaded" => new SolidColorBrush(Color.FromRgb(219, 234, 254)),
        "processing" => new SolidColorBrush(Color.FromRgb(233, 213, 255)),
        "complete" => new SolidColorBrush(Color.FromRgb(209, 250, 229)),
        _ => new SolidColorBrush(Color.FromRgb(229, 231, 235))
    };

    public Brush StatusForeground => Status switch
    {
        "pending" => new SolidColorBrush(Color.FromRgb(146, 64, 14)),
        "paid" => new SolidColorBrush(Color.FromRgb(30, 64, 175)),
        "uploaded" => new SolidColorBrush(Color.FromRgb(30, 64, 175)),
        "processing" => new SolidColorBrush(Color.FromRgb(107, 33, 168)),
        "complete" => new SolidColorBrush(Color.FromRgb(22, 101, 52)),
        _ => new SolidColorBrush(Color.FromRgb(75, 85, 99))
    };

    public OrderViewModel(Order order)
    {
        Id = order?.Id ?? "";
        SheetCount = order?.SheetCount ?? 0;
        TotalPriceSar = order?.TotalPriceSar ?? 0;
        Status = order?.Status ?? "";
        CreatedAt = order?.CreatedAt;
    }
}

```

File: revit-addin/LOD400Uploader/Views/UploadDialog.xaml

```
<Window x:Class="LOD400Uploader.Views.UploadDialog"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="LOD 400 Sheet Upgrade"
        Height="600" Width="800"
        WindowStartupLocation="CenterScreen"
        ResizeMode="CanResizeWithGrip"
        Background="#F5F5F5">

    <Window.Resources>

        <Style TargetType="Button" x:Key="PrimaryButton">
            <Setter Property="Background" Value="#2563EB"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Padding" Value="20,10"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="FontWeight" Value="SemiBold"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                CornerRadius="6"
                                Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="Button" x:Key="SecondaryButton">
            <Setter Property="Background" Value="White"/>
            <Setter Property="Foreground" Value="#374151"/>
            <Setter Property="Padding" Value="20,10"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="BorderBrush" Value="#D1D5DB"/>
            <Setter Property="BorderThickness" Value="1"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="6"
                                Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <!-- Header -->
        <Border Grid.Row="0" Background="White" BorderBrush="#E5E7EB" BorderThickness="0,0,0,1" Padding="24,20">
            <StackPanel>
                <TextBlock Text="LOD 400 Sheet Upgrade" FontSize="24" FontWeight="Bold" Foreground="#111827"/>
                <TextBlock Text="Select the sheets you want to upgrade to LOD 400 level of detail"
                           FontSize="14" Foreground="#6B7280" Margin="0,8,0,0"/>
            </StackPanel>
        </Border>

        <!-- Sheet List -->
        <Grid Grid.Row="1" Margin="24,16">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="*/>
            </Grid.RowDefinitions>

            <StackPanel Orientation="Horizontal" Margin="0,0,0,12">
                <CheckBox x:Name="SelectAllCheckBox" Content="Select All" FontSize="14"
                        VerticalAlignment="Center" Click="SelectAllCheckBox_Click"/>
                <TextBlock x:Name="SelectedCountText" Text="0 sheets selected"
                        FontSize="14" Foreground="#6B7280" Margin="20,0,0,0" VerticalAlignment="Center"/>
            </StackPanel>
        </Grid>
    </Grid>
</Window>
```

```

</StackPanel>>

<Border Grid.Row="1" Background="White" BorderBrush="#E5E7EB" BorderThickness="1" CornerRadius="8">
    <ListView x:Name="SheetListView" BorderThickness="0"
        SelectionMode="Multiple" SelectionChanged="SheetListView_SelectionChanged">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="" Width="40">
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <CheckBox IsChecked="{Binding IsSelected, Mode=TwoWay}"
                                Click="SheetCheckBox_Click"/>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
                <GridViewColumn Header="Sheet Number" Width="120" DisplayMemberBinding="{Binding SheetNumber}">
                </GridViewColumn>
                <GridViewColumn Header="Sheet Name" Width="350" DisplayMemberBinding="{Binding SheetName}">
                </GridViewColumn>
                <GridViewColumn Header="Revision" Width="100" DisplayMemberBinding="{Binding Revision}">
                </GridViewColumn>
            </GridView>
        </ListView.View>
    </ListView>
</Border>
</Grid>

<!-- Pricing Summary -->
<Border Grid.Row="2" Background="White" BorderBrush="#E5E7EB" BorderThickness="0,1,0,0" Padding="24,16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="Auto" />
        </Grid.ColumnDefinitions>

        <StackPanel Grid.Column="0">
            <TextBlock Text="Pricing" FontSize="16" FontWeight="SemiBold" Foreground="#111827">
            <TextBlock Text="150 SAR per sheet (~$40 USD)" FontSize="14" Foreground="#6B7280" Margin="0,4,0,0">
        </StackPanel>

        <StackPanel Grid.Column="1" HorizontalAlignment="Right">
            <TextBlock FontSize="14" Foreground="#6B7280">
                <Run x:Name="SheetCountRun" Text="0"/> sheets selected
            </TextBlock>
            <TextBlock FontSize="24" FontWeight="Bold" Foreground="#2563EB" HorizontalAlignment="Right" Margin="0,4,0,0">
                <Run x:Name="TotalPriceRun" Text="0"/> SAR
            </TextBlock>
        </StackPanel>
    </Grid>
</Border>

<!-- Footer Actions -->
<Border Grid.Row="3" Background="White" BorderBrush="#E5E7EB" BorderThickness="0,1,0,0" Padding="24,16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="Auto" />
        </Grid.ColumnDefinitions>

        <!-- Progress -->
        <StackPanel x:Name="ProgressPanel" Grid.Column="0" Visibility="Collapsed">
            <TextBlock x:Name="ProgressText" Text="Preparing..." FontSize="14" Foreground="#6B7280">
            <ProgressBar x:Name="ProgressBar" Height="8" Margin="0,8,0,0"
                Background="#E5E7EB" Foreground="#2563EB">
        </StackPanel>

        <Button Grid.Column="1" Content="Cancel" Style="{StaticResource SecondaryButton}"
            Margin="0,0,12,0" Click="CancelButton_Click">
        <Button x:Name="UploadButton" Grid.Column="2" Content="Pay & Upload"
            Style="{StaticResource PrimaryButton}" Click="UploadButton_Click"
            IsEnabled="False">
        </Grid>
    </Border>
</Grid>
</Window>

```

File: revit-addin/LOD400Uploader/Views/UploadDialog.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using Autodesk.Revit.DB;
using LOD400Uploader.Services;

namespace LOD400Uploader.Views
{
    public partial class UploadDialog : Window
    {
        private readonly Document _document;
        private readonly ObservableCollection<SheetItem> _sheets;
        private readonly ApiService _apiService;
        private readonly PackagingService _packagingService;
        private const int PRICE_PER_SHEET = 150;

        public UploadDialog(Document document) : this(document, null) { }

        public UploadDialog(Document document, ApiService apiService)
        {
            InitializeComponent();
            _document = document;
            _sheets = new ObservableCollection<SheetItem>();
            _apiService = apiService ?? new ApiService();
            _packagingService = new PackagingService();

            LoadSheets();

            private void LoadSheets()
            {
                try
                {
                    var collector = new FilteredElementCollector(_document)
                        .OfClass(typeof(ViewSheet))
                        .Cast<ViewSheet>()
                        .Where(s => s != null && !s.IsPlaceholder)
                        .OrderBy(s => s.SheetNumber ?? "");

                    foreach (var sheet in collector)
                    {
                        _sheets.Add(new SheetItem
                        {
                            ElementId = sheet.Id,
                            SheetNumber = sheet.SheetNumber ?? "",
                            SheetName = sheet.Name ?? "",
                            Revision = GetRevision(sheet),
                            IsSelected = false
                        });
                    }

                    SheetListView.ItemsSource = _sheets;
                    UpdateSummary();
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Error loading sheets: {ex.Message}", "Error",
                        MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }

            private string GetRevision(ViewSheet sheet)
            {
                try
                {
                    var param = sheet?.get_Parameter(BuiltInParameter.SHEET_CURRENT_REVISION);
                    return param?.AsString() ?? "";
                }
                catch
                {
                    return "";
                }
            }

            private void UpdateSummary()
            {
                int selectedCount = _sheets.Count(s => s.IsSelected);
                int totalPrice = selectedCount * PRICE_PER_SHEET;

                SelectedCountText.Text = $"{selectedCount} sheets selected";
                SheetCountRun.Text = selectedCount.ToString();
                TotalPriceRun.Text = totalPrice.ToString("N0");

                UploadButton.IsEnabled = selectedCount > 0;
            }

            private void SelectAllCheckBox_Click(object sender, RoutedEventArgs e)
            {
                bool isChecked = SelectAllCheckBox.IsChecked ?? false;
                foreach (var sheet in _sheets)
                {

```

```

        sheet.IsSelected = isChecked;
    }
    SheetListView.Items.Refresh();
    UpdateSummary();
}

private void SheetCheckBox_Click(object sender, RoutedEventArgs e)
{
    UpdateSummary();

    bool allSelected = _sheets.All(s => s.IsSelected);
    bool noneSelected = !_sheets.Any(s => s.IsSelected);

    if (allSelected)
        SelectAllCheckBox.IsChecked = true;
    else if (noneSelected)
        SelectAllCheckBox.IsChecked = false;
    else
        SelectAllCheckBox.IsChecked = null;
}

private void SheetListView_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateSummary();
}

private void CancelButton_Click(object sender, RoutedEventArgs e)
{
    Close();
}

private async void UploadButton_Click(object sender, RoutedEventArgs e)
{
    var selectedSheets = _sheets.Where(s => s.IsSelected).ToList();
    if (selectedSheets.Count == 0)
    {
        MessageBox.Show("Please select at least one sheet.", "No Sheets Selected",
            MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    if (!_apiService.HasApiKey)
    {
        var loginDialog = new LoginDialog();
        if (loginDialog.ShowDialog() != true || !loginDialog.IsAuthenticated)
        {
            return;
        }
    }

    UploadButton.IsEnabled = false;
    ProgressPanel.Visibility = Visibility.Visible;

    string packagePath = null;

    try
    {
        ProgressText.Text = "Creating order...";
        ProgressBar.Value = 5;

        var orderResponse = await _apiService.CreateOrderAsync(selectedSheets.Count);
        var order = orderResponse.Order;

        if (!string.IsNullOrEmpty(orderResponse.CheckoutUrl))
        {
            ProgressText.Text = "Opening payment page...";
            ProgressBar.Value = 10;

            Process.Start(new ProcessStartInfo
            {
                FileName = orderResponse.CheckoutUrl,
                UseShellExecute = true
            });

            MessageBox.Show(
                "A payment page has been opened in your browser.\n\n" +
                "Please complete the payment. The upload will begin automatically once payment is confirmed.\n\n" +
                "Do not close this window.",
                "Complete Payment",
                MessageBoxButton.OK,
                MessageBoxImage.Information);

            ProgressText.Text = "Waiting for payment confirmation...";
            ProgressBar.Value = 15;
            ProgressBar.IsIndeterminate = true;

            try
            {
                order = await _apiService.PollOrderStatusAsync(order.Id, maxAttempts: 90, delayMs: 2000);
                ProgressBar.IsIndeterminate = false;
            }
            catch (TimeoutException)
            {
                ProgressBar.IsIndeterminate = false;
                var result = MessageBox.Show(
                    "Payment confirmation is taking longer than expected.\n\n" +
                    "Would you like to continue waiting?",
                    "Payment Pending",
                    MessageBoxButton.YesNo,
                    MessageBoxImage.Question);
                if (result == MessageBoxResult.Yes)

```

```

        {
            ProgressBar.IsIndeterminate = true;
            order = await _apiService.PollOrderStatusAsync(order.Id, maxAttempts: 180, delayMs: 2000);
            ProgressBar.IsIndeterminate = false;
        }
        else
        {
            ProgressPanel.Visibility = Visibility.Collapsed;
            UploadButton.IsEnabled = true;
            MessageBox.Show(
                $"Order {order.Id} has been created but not paid.\n\n" +
                "You can complete payment later and upload your model using the 'Check Status' command.",
                "Order Pending",
                MessageBoxButton.OK,
                MessageBoxImage.Information);
            return;
        }
    }
}

if (order.Status != "paid" &&& order.Status != "uploaded" &&&
    order.Status != "processing" &&& order.Status != "complete")
{
    throw new InvalidOperationException($"Order status is '{order.Status}'. Expected 'paid' or later status.");
}

ProgressText.Text = "Packaging model...";
ProgressBar.Value = 20;

var selectedIds = selectedSheets.Select(s => s.ElementId).ToList();

await Task.Run(() =>
{
    packagePath = _packagingService.PackageModel(_document, selectedIds, (progress, message) =>
    {
        {
            Dispatcher.Invoke(() =>
            {
                ProgressText.Text = message;
                ProgressBar.Value = 20 + (progress * 0.4);
            });
        });
    });

    ProgressText.Text = "Preparing upload...";
    ProgressBar.Value = 65;

    string fileName = System.IO.Path.GetFileName(packagePath);
    string uploadUrl = await _apiService.GetUploadUrlAsync(order.Id, fileName);

    ProgressText.Text = "Uploading model...";
    ProgressBar.Value = 70;

    long fileSize = _packagingService.GetFileSize(packagePath);
    byte[] fileData = await Task.Run(() => _packagingService.ReadFileBytes(packagePath));

    await _apiService.UploadFileAsync(uploadUrl, fileData, (progress) =>
    {
        Dispatcher.Invoke(() =>
        {
            {
                ProgressBar.Value = 70 + (progress * 0.25);
            });
        });
    });

    ProgressText.Text = "Finalizing...";
    ProgressBar.Value = 95;

    await _apiService.MarkUploadCompleteAsync(order.Id, fileName, fileSize, uploadUrl);

    _packagingService.Cleanup(packagePath);
    packagePath = null;

    ProgressBar.Value = 100;
    ProgressText.Text = "Upload complete!";

    MessageBox.Show(
        $"Your model has been uploaded successfully!\n\n" +
        $"Order ID: {order.Id}\n" +
        $"Sheets: {selectedSheets.Count}\n" +
        $"Total: {order.TotalPriceSar} SAR\n\n" +
        "You will be notified when your LOD 400 deliverables are ready.",
        "Upload Complete",
        MessageBoxButton.OK,
        MessageBoxImage.Information);

    Close();
}
catch (Exception ex)
{
    if (!string.IsNullOrEmpty(packagePath))
    {
        _packagingService.Cleanup(packagePath);
    }

    MessageBox.Show(
        $"An error occurred during the upload process:\n\n{ex.Message}\n\n" +
        "Please try again or contact support if the issue persists.",
        "Upload Error",
        MessageBoxButton.OK,
        MessageBoxImage.Error);

    ProgressPanel.Visibility = Visibility.Collapsed;
}

```

```

        ProgressBar.IsIndeterminate = false;
        UploadButton.IsEnabled = true;
    }
}

public class SheetItem : INotifyPropertyChanged
{
    private bool _isSelected;

    public ElementId ElementId { get; set; }
    public string SheetNumber { get; set; }
    public string SheetName { get; set; }
    public string Revision { get; set; }

    public bool IsSelected
    {
        get => _isSelected;
        set
        {
            _isSelected = value;
            OnPropertyChanged(nameof(IsSelected));
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

File: revit-addin/README.md

LOD 400 Uploader - Revit Add-in

A Revit add-in for uploading BIM models to the LOD 400 Delivery Platform for professional LOD 300 to LOD 400 upgrades.

Features

- ****Sheet Selection****: Browse and select specific sheets from your Revit model
- ****Pricing Preview****: See real-time pricing (150 SAR per sheet) before payment
- ****Secure Payment****: Integrated Stripe checkout for secure transactions
- ****Model Packaging****: Automatically packages your model with sheet manifest
- ****Workshared Support****: Safely handles workshared/central models
- ****Upload Progress****: Track upload progress with real-time feedback
- ****Order Tracking****: Check order status and download completed deliverables

Requirements

- ****Revit Version****: 2024 (can be adapted for 2020-2025)
- ****NET Framework****: 4.8
- ****Visual Studio****: 2022 or later (for compilation)
- ****Internet Connection****: Required for API communication

Getting Started

Step 1: Generate Your API Key

1. Log in to the LOD 400 Delivery web dashboard
2. Go to Settings > API Keys
3. Click "Create New API Key"
4. Copy and save the key - it will only be shown once

Step 2: Configure API URL

Before building, update the API URL in `App.cs`:

```
```csharp
// In App.cs, update this line with your actual Replit URL:
ApiBaseUrl = Environment.GetEnvironmentVariable("LOD400_API_URL")
?? "https://YOUR-REPLIT-URL.replit.app";
```
```

Step 3: Update Revit References

Update the Revit API references in `LOD400Uploader.csproj` to match your Revit installation:

```
```xml
<Reference Include="RevitAPI">
 <HintPath>C:\Program Files\Autodesk\Revit 2024\RevitAPI.dll</HintPath>
 <Private>False</Private>
</Reference>
<Reference Include="RevitAPIUI">
 <HintPath>C:\Program Files\Autodesk\Revit 2024\RevitAPIUI.dll</HintPath>
 <Private>False</Private>
</Reference>
```
```

For different Revit versions, update the path (e.g., `Revit 2023`, `Revit 2022`, etc.)

Step 4: Build the Project

1. Open `LOD400Uploader.csproj` in Visual Studio 2022
2. Restore NuGet packages (Newtonsoft.Json)
3. Build the solution in Release mode
4. The output will be in `bin\Release\net48\`

Step 5: Install the Add-in

1. Copy the following files to your Revit add-ins folder:
 - `LOD400Uploader.dll`
 - `Newtonsoft.Json.dll`
 - `LOD400Uploader.addin`
2. The add-ins folder is typically located at:
 - ****Current User****: `%APPDATA%\Autodesk\Revit\Addins\2024\`
 - ****All Users****: `C:\ProgramData\Autodesk\Revit\Addins\2024\`
3. Restart Revit

Usage

First-Time Login

1. Go to the ****LOD 400**** tab in the ribbon
2. Click ****Upload Sheets**** or ****Check Status****
3. Enter your API key when prompted
4. The key will be saved for future sessions

Uploading Sheets

1. Open your Revit model
2. Save the model (required before upload)
3. Go to the ****LOD 400**** tab in the ribbon
4. Click ****Upload Sheets****
5. Select the sheets you want to upgrade
6. Review the pricing summary
7. Click ****Pay & Upload****
8. Complete payment in your browser
9. Wait for payment confirmation (automatic polling)

10. Upload begins automatically after payment

Checking Order Status

1. Go to the ****LOD 400**** tab
2. Click ****Check Status****
3. View your order history
4. Select a completed order
5. Click ****Download Deliverables**** to get your upgraded model

Project Structure

```
...
LOD400Uploader/
■■■ App.cs                # Main application entry point
■■■ LOD400Uploader.csproj  # Project file
■■■ LOD400Uploader.addin   # Revit add-in manifest
■■■ Commands/
■   ■■■ UploadSheetsCommand.cs # Upload command
■   ■■■ CheckStatusCommand.cs  # Status check command
■■■ Models/
■   ■■■ Order.cs               # API data models
■■■ Services/
■   ■■■ ApiService.cs         # API communication (with API key auth)
■   ■■■ PackagingService.cs     # Model packaging (workshared safe)
■■■ Views/
■   ■■■ LoginDialog.xaml       # API key login UI
■   ■■■ LoginDialog.xaml.cs
■   ■■■ UploadDialog.xaml      # Upload UI
■   ■■■ UploadDialog.xaml.cs
■   ■■■ StatusDialog.xaml      # Status UI
■■■ ■■■ StatusDialog.xaml.cs
...
```

Authentication

The add-in uses API key authentication:

- Generate API keys from the web dashboard
- Keys are stored securely in `%APPDATA%\LOD400Uploader\config.json`
- Keys can be revoked from the web dashboard at any time

Workshared Models

The add-in safely handles workshared (central) models:

- Detaches a copy for upload without affecting the central model
- Preserves all model data and links
- No changes are made to the original central model

Troubleshooting

Add-in Not Loading

1. Check that all files are in the correct add-ins folder
2. Verify the `.addin` manifest has the correct assembly name
3. Check Revit's add-in manager for loading errors

API Key Invalid

1. Verify you copied the complete key
2. Generate a new key from the web dashboard
3. Delete `%APPDATA%\LOD400Uploader\config.json` and re-enter the key

Connection Errors

1. Verify your internet connection
2. Check that the API URL is correct in `App.cs`
3. Ensure the server is running

Upload Failures

1. Save your model before uploading
2. Ensure unsaved changes are saved
3. For workshared models, ensure you have proper access
4. Check file size (large models may take longer)
5. Ensure stable internet connection during upload

Payment Timeout

1. Complete payment in the browser promptly
2. If timeout occurs, you can continue waiting
3. Check order status later if needed

API Endpoints Used

The add-in communicates with these API endpoints:

- `GET /api/addin/validate` - Validate API key
- `POST /api/addin/create-order` - Create order and get payment URL
- `GET /api/addin/orders` - List user orders
- `GET /api/addin/orders/:id/status` - Get order status
- `POST /api/addin/orders/:id/upload-url` - Get file upload URL
- `POST /api/addin/orders/:id/upload-complete` - Mark upload complete
- `GET /api/addin/orders/:id/download-url` - Get deliverable download URL

Support

For technical support or questions, please contact the LOD 400 Delivery Platform team.

License

This add-in is provided as part of the LOD 400 Delivery Platform service.

File: script/build.ts

```
import { build as esbuild } from "esbuild";
import { build as viteBuild } from "vite";
import { rm, readFile } from "fs/promises";

// server deps to bundle to reduce openat(2) syscalls
// which helps cold start times
const allowlist = [
  "@google/generative-ai",
  "@neondatabase/serverless",
  "axios",
  "connect-pg-simple",
  "cors",
  "date-fns",
  "drizzle-orm",
  "drizzle-zod",
  "express",
  "express-rate-limit",
  "express-session",
  "jsonwebtoken",
  "memorystore",
  "multer",
  "nanoid",
  "nodemailer",
  "openai",
  "passport",
  "passport-local",
  "stripe",
  "uuid",
  "ws",
  "xlsx",
  "zod",
  "zod-validation-error",
];

async function buildAll() {
  await rm("dist", { recursive: true, force: true });

  console.log("building client...");
  await viteBuild();

  console.log("building server...");
  const pkg = JSON.parse(await readFile("package.json", "utf-8"));
  const allDeps = [
    ...Object.keys(pkg.dependencies || {}),
    ...Object.keys(pkg.devDependencies || {}),
  ];
  const externals = allDeps.filter((dep) => !allowlist.includes(dep));

  await esbuild({
    entryPoints: ["server/index.ts"],
    platform: "node",
    bundle: true,
    format: "cjs",
    outfile: "dist/index.cjs",
    define: {
      "process.env.NODE_ENV": "production",
    },
    minify: true,
    external: externals,
    logLevel: "info",
  });
}

buildAll().catch((err) => {
  console.error(err);
  process.exit(1);
});
```

File: server/db.ts

```
import { Pool, neonConfig } from '@neondatabase/serverless';
import { drizzle } from 'drizzle-orm/neon-serverless';
import ws from "ws";
import * as schema from "@shared/schema";

neonConfig.webSocketConstructor = ws;

if (!process.env.DATABASE_URL) {
  throw new Error(
    "DATABASE_URL must be set. Did you forget to provision a database?",
  );
}

export const pool = new Pool({ connectionString: process.env.DATABASE_URL });
export const db = drizzle({ client: pool, schema });
```

File: server/index.ts

```
import express, { type Request, Response, NextFunction } from "express";
import { registerRoutes } from "../routes";
import { serveStatic } from "../static";
import { createServer } from "http";
import { runMigrations } from 'stripe-replit-sync';
import { getStripeSync } from "../stripeClient";
import { WebhookHandlers } from "../webhookHandlers";

const app = express();
const httpServer = createServer(app);

declare module "http" {
  interface IncomingMessage {
    rawBody: unknown;
  }
}

export function log(message: string, source = "express") {
  const formattedTime = new Date().toLocaleTimeString("en-US", {
    hour: "numeric",
    minute: "2-digit",
    second: "2-digit",
    hour12: true,
  });
  console.log(`${formattedTime} [${source}] ${message}`);
}

async function initStripe() {
  const databaseUrl = process.env.DATABASE_URL;

  if (!databaseUrl) {
    log('DATABASE_URL not set, skipping Stripe initialization', 'stripe');
    return;
  }

  try {
    log('Initializing Stripe schema...', 'stripe');
    await runMigrations({
      databaseUrl,
      schema: 'stripe'
    });
    log('Stripe schema ready', 'stripe');

    const stripeSync = await getStripeSync();

    log('Setting up managed webhook...', 'stripe');
    const webhookBaseUrl = `https://${process.env.REPLIT_DOMAINS?.split(',')[0]}`;
    const { webhook, uuid } = await stripeSync.findOrCreateManagedWebhook(
      `${webhookBaseUrl}/api/stripe/webhook`,
      {
        enabled_events: ['checkout.session.completed', 'checkout.session.expired'],
        description: 'LOD 400 Delivery Platform webhook',
      }
    );
    log(`Webhook configured: ${webhook.url}`, 'stripe');

    log('Syncing Stripe data...', 'stripe');
    stripeSync.syncBackfill()
      .then(() => {
        log('Stripe data synced', 'stripe');
      })
      .catch((err: Error) => {
        log(`Error syncing Stripe data: ${err.message}`, 'stripe');
      });
  } catch (error: any) {
    log(`Failed to initialize Stripe: ${error.message}`, 'stripe');
  }
}

(async () => {
  try {
    // Register Stripe webhook route FIRST (before json middleware)
    app.post(
      '/api/stripe/webhook/:uuid',
      express.raw({ type: 'application/json' }),
      async (req, res) => {
        const signature = req.headers['stripe-signature'];

        if (!signature) {
          return res.status(400).json({ error: 'Missing stripe-signature' });
        }

        try {
          const sig = Array.isArray(signature) ? signature[0] : signature;

          if (!Buffer.isBuffer(req.body)) {
            log('Webhook error: req.body is not a Buffer', 'stripe');
            return res.status(500).json({ error: 'Webhook processing error' });
          }

          const { uuid } = req.params;
          await WebhookHandlers.processWebhook(req.body as Buffer, sig, uuid);

          res.status(200).json({ received: true });
        } catch (error: any) {
          log(`Webhook error: ${error.message}`, 'stripe');
        }
      }
    );
  }
});
```

```

        res.status(400).json({ error: 'Webhook processing error' });
    }
}
);

app.use(
    express.json({
        verify: (req, _res, buf) => {
            req.rawBody = buf;
        },
    }),
);

app.use(express.urlencoded({ extended: false }));

app.use((req, res, next) => {
    const start = Date.now();
    const path = req.path;
    let capturedJsonResponse: Record<string, any> | undefined = undefined;

    const originalResJson = res.json;
    res.json = function (bodyJson, ...args) {
        capturedJsonResponse = bodyJson;
        return originalResJson.apply(res, [bodyJson, ...args]);
    };

    res.on("finish", () => {
        const duration = Date.now() - start;
        if (path.startsWith("/api")) {
            let logLine = `${req.method} ${path} ${res.statusCode} in ${duration}ms`;
            if (capturedJsonResponse) {
                logLine += ` :: ${JSON.stringify(capturedJsonResponse)}`;
            }

            log(logLine);
        }
    });

    next();
});

await registerRoutes(httpServer, app);

app.use((err: any, _req: Request, res: Response, _next: NextFunction) => {
    const status = err.status || err.statusCode || 500;
    const message = err.message || "Internal Server Error";

    res.status(status).json({ message });
    console.error(err);
});

if (process.env.NODE_ENV === "production") {
    serveStatic(app);
} else {
    const { setupVite } = await import("./vite");
    await setupVite(httpServer, app);
}

const port = parseInt(process.env.PORT || "5000", 10);

// Start listening FIRST, then initialize external services
httpServer.listen(
    {
        port,
        host: "0.0.0.0",
        reusePort: true,
    },
    () => {
        log(`serving on port ${port}`);

        // Initialize Stripe in the background AFTER server is listening
        initStripe().catch((err) => {
            log(`Stripe initialization error: ${err.message}`, 'stripe');
        });
    },
);

} catch (error: any) {
    console.error('Failed to start server:', error);
    process.exit(1);
}
})();

```

File: server/objectStorage.ts

```
import { Storage, File } from "@google-cloud/storage";
import { Response } from "express";
import { randomUUID } from "crypto";

const REPLIT_SIDE CAR_ENDPOINT = "http://127.0.0.1:1106";

export const objectStorageClient = new Storage({
  credentials: {
    audience: "replit",
    subject_token_type: "access_token",
    token_url: `${REPLIT_SIDE CAR_ENDPOINT}/token`,
    type: "external_account",
    credential_source: {
      url: `${REPLIT_SIDE CAR_ENDPOINT}/credential`,
      format: {
        type: "json",
        subject_token_field_name: "access_token",
      },
    },
  },
  universe_domain: "googleapis.com",
},
{
  projectId: "",
});

export class ObjectNotFoundError extends Error {
  constructor() {
    super("Object not found");
    this.name = "ObjectNotFoundError";
    Object.setPrototypeOf(this, ObjectNotFoundError.prototype);
  }
}

export class ObjectStorageService {
  constructor() {}

  getPublicObjectSearchPaths(): Array<string> {
    const pathsStr = process.env.PUBLIC_OBJECT_SEARCH_PATHS || "";
    const paths = Array.from(
      new Set(
        pathsStr
          .split(",")
          .map((path) => path.trim())
          .filter((path) => path.length > 0)
      )
    );
    if (paths.length === 0) {
      throw new Error(
        "PUBLIC_OBJECT_SEARCH_PATHS not set. Create a bucket in 'Object Storage' +
        "tool and set PUBLIC_OBJECT_SEARCH_PATHS env var (comma-separated paths)."
      );
    }
    return paths;
  }

  getPrivateObjectDir(): string {
    const dir = process.env.PRIVATE_OBJECT_DIR || "";
    if (!dir) {
      throw new Error(
        "PRIVATE_OBJECT_DIR not set. Create a bucket in 'Object Storage' +
        "tool and set PRIVATE_OBJECT_DIR env var."
      );
    }
    return dir;
  }

  async searchPublicObject(filePath: string): Promise<File | null> {
    for (const searchPath of this.getPublicObjectSearchPaths()) {
      const fullPath = `${searchPath}/${filePath}`;
      const { bucketName, objectName } = this.parseObjectPath(fullPath);
      const bucket = objectStorageClient.bucket(bucketName);
      const file = bucket.file(objectName);
      const [exists] = await file.exists();
      if (exists) {
        return file;
      }
    }
    return null;
  }

  async downloadObject(file: File, res: Response, cacheTtlSec: number = 3600) {
    try {
      const [metadata] = await file.getMetadata();
      res.set({
        "Content-Type": metadata.contentType || "application/octet-stream",
        "Content-Length": metadata.size,
        "Cache-Control": `private, max-age=${cacheTtlSec}`,
      });
      const stream = file.createReadStream();
      stream.on("error", (err) => {
        console.error("Stream error:", err);
        if (!res.headersSent) {
          res.status(500).json({ error: "Error streaming file" });
        }
      });
      stream.pipe(res);
    } catch (error) {
      console.error("Error downloading file:", error);
    }
  }
}
```

```

        if (!res.headersSent) {
            res.status(500).json({ error: "Error downloading file" });
        }
    }
}

async getUploadURL(orderId: string, fileName: string): Promise<string> {
    const privateObjectDir = this.getPrivateObjectDir();
    if (!privateObjectDir) {
        throw new Error("PRIVATE_OBJECT_DIR not set.");
    }

    const objectId = randomUUID();
    const fullPath = `${privateObjectDir}/orders/${orderId}/${objectId}_${fileName}`;
    const { bucketName, objectName } = this.parseObjectPath(fullPath);

    return this.signObjectURL({
        bucketName,
        objectName,
        method: "PUT",
        ttlSec: 3600, // 1 hour
    });
}

async getDownloadURL(storageKey: string): Promise<string> {
    const { bucketName, objectName } = this.parseObjectPath(storageKey);

    return this.signObjectURL({
        bucketName,
        objectName,
        method: "GET",
        ttlSec: 3600, // 1 hour
    });
}

async getFile(storageKey: string): Promise<File> {
    const { bucketName, objectName } = this.parseObjectPath(storageKey);
    const bucket = objectStorageClient.bucket(bucketName);
    const file = bucket.file(objectName);
    const [exists] = await file.exists();
    if (!exists) {
        throw new ObjectNotFoundError();
    }
    return file;
}

normalizeStorageKey(uploadURL: string): string {
    if (!uploadURL.startsWith("https://storage.googleapis.com/")) {
        return uploadURL;
    }
    const url = new URL(uploadURL);
    return url.pathname.slice(1); // Remove leading /
}

parseObjectPath(path: string): { bucketName: string; objectName: string } {
    if (!path.startsWith("/")) {
        path = `/${path}`;
    }
    const pathParts = path.split("/");
    if (pathParts.length < 3) {
        throw new Error("Invalid path: must contain at least a bucket name");
    }
    const bucketName = pathParts[1];
    const objectName = pathParts.slice(2).join("/");
    return { bucketName, objectName };
}

async signObjectURL({
    bucketName,
    objectName,
    method,
    ttlSec,
}: {
    bucketName: string;
    objectName: string;
    method: "GET" | "PUT" | "DELETE" | "HEAD";
    ttlSec: number;
}): Promise<string> {
    const request = {
        bucket_name: bucketName,
        object_name: objectName,
        method,
        expires_at: new Date(Date.now() + ttlSec * 1000).toISOString(),
    };
    const response = await fetch(
        `${REPLIT_SIDECAR_ENDPOINT}/object-storage/signed-object-url`,
        {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(request),
        }
    );
    if (!response.ok) {
        throw new Error(
            `Failed to sign object URL, errorcode: ${response.status}`
        );
    }
    const { signed_url: signedURL } = await response.json();
    return signedURL;
}

```


File: server/replitAuth.ts

```
import * as client from "openid-client";
import { Strategy, type VerifyFunction } from "openid-client/passport";

import passport from "passport";
import session from "express-session";
import type { Express, RequestHandler } from "express";
import memoize from "memoizee";
import connectPg from "connect-pg-simple";
import { storage } from "../storage";

const getOidcConfig = memoize(
  async () => {
    return await client.discovery(
      new URL(process.env.ISSUER_URL ?? "https://replit.com/oidc"),
      process.env.REPL_ID!
    );
  },
  { maxAge: 3600 * 1000 }
);

export function getSession() {
  const sessionTtl = 7 * 24 * 60 * 1000; // 1 week
  const pgStore = connectPg(session);
  const sessionStore = new pgStore({
    connectionString: process.env.DATABASE_URL,
    createTableIfMissing: false,
    ttl: sessionTtl,
    tableName: "sessions",
  });
  return session({
    secret: process.env.SESSION_SECRET!,
    store: sessionStore,
    resave: false,
    saveUninitialized: false,
    cookie: {
      httpOnly: true,
      secure: true,
      maxAge: sessionTtl,
    },
  });
}

function updateUserSession(
  user: any,
  tokens: client.TokenEndpointResponse & client.TokenEndpointResponseHelpers
) {
  user.claims = tokens.claims();
  user.access_token = tokens.access_token;
  user.refresh_token = tokens.refresh_token;
  user.expires_at = user.claims?.exp;
}

async function upsertUser(claims: any) {
  await storage.upsertUser({
    id: claims["sub"],
    email: claims["email"],
    firstName: claims["first_name"],
    lastName: claims["last_name"],
    profileImageUrl: claims["profile_image_url"],
  });
}

export async function setupAuth(app: Express) {
  app.set("trust proxy", 1);
  app.use(getSession());
  app.use(passport.initialize());
  app.use(passport.session());

  const config = await getOidcConfig();

  const verify: VerifyFunction = async (
    tokens: client.TokenEndpointResponse & client.TokenEndpointResponseHelpers,
    verified: passport.AuthenticateCallback
  ) => {
    const user = {};
    updateUserSession(user, tokens);
    await upsertUser(tokens.claims());
    verified(null, user);
  };

  const registeredStrategies = new Set<string>();

  const ensureStrategy = (domain: string) => {
    const strategyName = `replitauth:${domain}`;
    if (!registeredStrategies.has(strategyName)) {
      const strategy = new Strategy(
        {
          name: strategyName,
          config,
          scope: "openid email profile offline_access",
          callbackURL: `https://${domain}/api/callback`,
        },
        verify,
      );
      passport.use(strategy);
      registeredStrategies.add(strategyName);
    }
  };
}
```

```

};

passport.serializeUser((user: Express.User, cb) => cb(null, user));
passport.deserializeUser((user: Express.User, cb) => cb(null, user));

app.get("/api/login", (req, res, next) => {
  ensureStrategy(req.hostname);
  passport.authenticate(`replitauth:${req.hostname}`, {
    prompt: "login consent",
    scope: ["openid", "email", "profile", "offline_access"],
  })(req, res, next);
});

app.get("/api/callback", (req, res, next) => {
  ensureStrategy(req.hostname);
  passport.authenticate(`replitauth:${req.hostname}`, {
    successReturnToOrRedirect: "/",
    failureRedirect: "/api/login",
  })(req, res, next);
});

app.get("/api/logout", (req, res) => {
  req.logout() => {
    res.redirect(
      client.buildEndSessionUrl(config, {
        client_id: process.env.REPL_ID!,
        post_logout_redirect_uri: `${req.protocol}://${req.hostname}`,
      }).href
    );
  });
});
}

export const isAuthenticated: RequestHandler = async (req, res, next) => {
  const user = req.user as any;

  if (!req.isAuthenticated() || !user.expires_at) {
    return res.status(401).json({ message: "Unauthorized" });
  }

  const now = Math.floor(Date.now() / 1000);
  if (now <= user.expires_at) {
    return next();
  }

  const refreshToken = user.refresh_token;
  if (!refreshToken) {
    res.status(401).json({ message: "Unauthorized" });
    return;
  }

  try {
    const config = await getOidcConfig();
    const tokenResponse = await client.refreshTokenGrant(config, refreshToken);
    updateUserSession(user, tokenResponse);
    return next();
  } catch (error) {
    res.status(401).json({ message: "Unauthorized" });
    return;
  }
};

export const isAdmin: RequestHandler = async (req, res, next) => {
  const user = req.user as any;
  const userId = user?.claims?.sub;

  if (!userId) {
    return res.status(401).json({ message: "Unauthorized" });
  }

  const dbUser = await storage.getUser(userId);
  if (!dbUser?.isAdmin) {
    return res.status(403).json({ message: "Forbidden - Admin access required" });
  }

  return next();
};

```

File: server/routes.ts

```
import type { Express } from "express";
import { createServer, type Server } from "http";
import { storage } from "../storage";
import { setupAuth, isAuthenticated, isAdmin } from "../replitAuth";
import { ObjectStorageService, ObjectNotFoundError } from "../objectStorage";
import { createOrderRequestSchema, PRICE_PER_SHEET_SAR } from "@shared/schema";
import { getUncachableStripeClient } from "../stripeClient";

const objectStorage = new ObjectStorageService();

export async function registerRoutes(
  httpServer: Server,
  app: Express
): Promise<Server> {
  await setupAuth(app);

  app.get("/api/auth/user", isAuthenticated, async (req: any, res) => {
    try {
      const userId = req.user.claims.sub;
      const user = await storage.getUser(userId);
      res.json(user);
    } catch (error) {
      console.error("Error fetching user:", error);
      res.status(500).json({ message: "Failed to fetch user" });
    }
  });

  // =====
  // CLIENT API ROUTES
  // =====

  app.get("/api/orders", isAuthenticated, async (req: any, res) => {
    try {
      const userId = req.user.claims.sub;
      const orders = await storage.getOrdersByUserId(userId);
      res.json(orders);
    } catch (error) {
      console.error("Error fetching orders:", error);
      res.status(500).json({ message: "Failed to fetch orders" });
    }
  });

  app.post("/api/orders", isAuthenticated, async (req: any, res) => {
    try {
      const userId = req.user.claims.sub;
      const parsed = createOrderRequestSchema.safeParse(req.body);

      if (!parsed.success) {
        return res.status(400).json({ message: "Invalid request", errors: parsed.error.errors });
      }

      const { sheetCount } = parsed.data;
      const totalPriceSar = sheetCount * PRICE_PER_SHEET_SAR;

      const order = await storage.createOrder({
        userId,
        sheetCount,
        totalPriceSar,
        status: "pending",
      });

      res.status(201).json(order);
    } catch (error) {
      console.error("Error creating order:", error);
      res.status(500).json({ message: "Failed to create order" });
    }
  });

  app.get("/api/orders/:orderId/checkout", isAuthenticated, async (req: any, res) => {
    try {
      const userId = req.user.claims.sub;
      const { orderId } = req.params;

      const order = await storage.getOrder(orderId);
      if (!order) {
        return res.status(404).json({ message: "Order not found" });
      }

      if (order.userId !== userId) {
        return res.status(403).json({ message: "Forbidden" });
      }

      if (order.status !== "pending") {
        return res.status(400).json({ message: "Order is not pending payment" });
      }

      const stripe = await getUncachableStripeClient();

      const session = await stripe.checkout.sessions.create({
        payment_method_types: ["card"],
        line_items: [
          {
            price_data: {
              currency: "sar",
              product_data: {
                name: `LOD 400 Sheet Upgrade (${order.sheetCount} sheets)`,
                description: `Professional LOD 300 to LOD 400 model upgrade for ${order.sheetCount} sheets`,
              },
            },
            quantity: 1,
          },
        ],
      });

      res.json({ url: session.url });
    } catch (error) {
      console.error("Error creating checkout session:", error);
      res.status(500).json({ message: "Failed to create checkout session" });
    }
  });
}
```

```

        },
        unit_amount: order.totalPriceSar * 100,
      },
      quantity: 1,
    },
  ],
  mode: "payment",
  success_url: `${req.protocol}://${req.get('host')}/?payment=success&order=${orderId}`,
  cancel_url: `${req.protocol}://${req.get('host')}/?payment=cancelled&order=${orderId}`,
  metadata: {
    orderId,
    userId,
  },
});

await storage.updateOrder(orderId, { stripeSessionId: session.id });

res.redirect(session.url!);
} catch (error) {
  console.error("Error creating checkout:", error);
  res.status(500).json({ message: "Failed to create checkout session" });
}
});

app.post("/api/orders/:orderId/upload-url", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const { orderId } = req.params;
    const { fileName } = req.body;

    if (!fileName) {
      return res.status(400).json({ message: "fileName is required" });
    }

    const order = await storage.getOrder(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.userId !== userId) {
      return res.status(403).json({ message: "Forbidden" });
    }

    if (order.status !== "paid") {
      return res.status(400).json({ message: "Order must be paid before uploading files" });
    }

    const uploadURL = await objectStorage.getUploadURL(orderId, fileName);

    res.json({ uploadURL });
  } catch (error) {
    console.error("Error getting upload URL:", error);
    res.status(500).json({ message: "Failed to get upload URL" });
  }
});

app.post("/api/orders/:orderId/upload-complete", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const { orderId } = req.params;
    const { fileName, fileSize, uploadURL } = req.body;

    if (!fileName || !uploadURL) {
      return res.status(400).json({ message: "fileName and uploadURL are required" });
    }

    const order = await storage.getOrder(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.userId !== userId) {
      return res.status(403).json({ message: "Forbidden" });
    }

    const storageKey = objectStorage.normalizeStorageKey(uploadURL);

    await storage.createFile({
      orderId,
      fileType: "input",
      fileName,
      fileSize: fileSize || null,
      storageKey,
      mimeType: "application/zip",
    });

    await storage.updateOrderStatus(orderId, "uploaded");

    res.json({ success: true });
  } catch (error) {
    console.error("Error completing upload:", error);
    res.status(500).json({ message: "Failed to complete upload" });
  }
});

app.get("/api/orders/:orderId/status", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const { orderId } = req.params;

    const order = await storage.getOrderWithFiles(orderId);
  }
});

```

```

    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    const user = await storage.getUser(userId);
    if (order.userId !== userId && !user?.isAdmin) {
      return res.status(403).json({ message: "Forbidden" });
    }

    res.json(order);
  } catch (error) {
    console.error("Error checking order status:", error);
    res.status(500).json({ message: "Failed to check order status" });
  }
});

app.get("/api/files/:fileId/download", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const { fileId } = req.params;

    const file = await storage.getFile(fileId);
    if (!file) {
      return res.status(404).json({ message: "File not found" });
    }

    const order = await storage.getOrder(file.orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    const user = await storage.getUser(userId);
    if (order.userId !== userId && !user?.isAdmin) {
      return res.status(403).json({ message: "Forbidden" });
    }

    const downloadURL = await objectStorage.getDownloadURL(file.storageKey);
    res.redirect(downloadURL);
  } catch (error) {
    console.error("Error downloading file:", error);
    if (error instanceof ObjectNotFoundError) {
      return res.status(404).json({ message: "File not found in storage" });
    }
    res.status(500).json({ message: "Failed to download file" });
  }
});

// =====
// ADMIN API ROUTES
// =====

app.get("/api/admin/orders", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const orders = await storage.getAllOrders();
    res.json(orders);
  } catch (error) {
    console.error("Error fetching orders:", error);
    res.status(500).json({ message: "Failed to fetch orders" });
  }
});

app.get("/api/admin/clients", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const clients = await storage.getUsersWithOrderStats();
    res.json(clients);
  } catch (error) {
    console.error("Error fetching clients:", error);
    res.status(500).json({ message: "Failed to fetch clients" });
  }
});

app.patch("/api/admin/orders/:orderId/status", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const { orderId } = req.params;
    const { status } = req.body;

    if (!["pending", "paid", "uploaded", "processing", "complete"].includes(status)) {
      return res.status(400).json({ message: "Invalid status" });
    }

    const order = await storage.updateOrderStatus(orderId, status);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    res.json(order);
  } catch (error) {
    console.error("Error updating order status:", error);
    res.status(500).json({ message: "Failed to update order status" });
  }
});

app.post("/api/admin/orders/:orderId/upload-url", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const { orderId } = req.params;
    const { fileName } = req.body;

    if (!fileName) {
      return res.status(400).json({ message: "fileName is required" });
    }

    const order = await storage.getOrder(orderId);
  }
});

```

```

    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.status !== "uploaded" && order.status !== "processing") {
      return res.status(400).json({ message: "Order is not ready for deliverables" });
    }

    const uploadURL = await objectStorage.getUploadURL(orderId, fileName);

    res.json({ uploadURL });
  } catch (error) {
    console.error("Error getting upload URL:", error);
    res.status(500).json({ message: "Failed to get upload URL" });
  }
});

app.post("/api/admin/orders/:orderId/upload-complete", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const { orderId } = req.params;
    const { fileName, fileSize, uploadURL } = req.body;

    if (!fileName || !uploadURL) {
      return res.status(400).json({ message: "fileName and uploadURL are required" });
    }

    const order = await storage.getOrder(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    const storageKey = objectStorage.normalizeStorageKey(uploadURL);

    await storage.createFile({
      orderId,
      fileType: "output",
      fileName,
      fileSize: fileSize || null,
      storageKey,
      mimeType: "application/zip",
    });

    await storage.updateOrderStatus(orderId, "processing");

    res.json({ success: true });
  } catch (error) {
    console.error("Error completing upload:", error);
    res.status(500).json({ message: "Failed to complete upload" });
  }
});

app.post("/api/admin/orders/:orderId/complete", isAuthenticated, isAdmin, async (req: any, res) => {
  try {
    const { orderId } = req.params;

    const order = await storage.getOrderWithFiles(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    const hasOutputFiles = order.files?.some(f => f.fileType === "output");
    if (!hasOutputFiles) {
      return res.status(400).json({ message: "Must upload deliverables before completing" });
    }

    await storage.updateOrderStatus(orderId, "complete");

    console.log(`Order ${orderId} marked complete. Client email: ${order.user?.email}`);

    res.json({ success: true });
  } catch (error) {
    console.error("Error completing order:", error);
    res.status(500).json({ message: "Failed to complete order" });
  }
});

// =====
// API KEY MANAGEMENT (for Revit add-in users)
// =====

app.get("/api/user/api-keys", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const keys = await storage.getApiKeysByUserId(userId);
    const safeKeys = keys.map(k => ({
      id: k.id,
      name: k.name,
      lastUsed: k.lastUsed,
      createdAt: k.createdAt,
    }));
    res.json(safeKeys);
  } catch (error) {
    console.error("Error fetching API keys:", error);
    res.status(500).json({ message: "Failed to fetch API keys" });
  }
});

app.post("/api/user/api-keys", isAuthenticated, async (req: any, res) => {
  try {
    const userId = req.user.claims.sub;
    const { name } = req.body;

```

```

    if (!name || typeof name !== "string") {
      return res.status(400).json({ message: "Name is required" });
    }

    const { apiKey, rawKey } = await storage.createApiKey(userId, name);
    res.status(201).json({
      id: apiKey.id,
      name: apiKey.name,
      key: rawKey,
      createdAt: apiKey.createdAt,
    });
  } catch (error) {
    console.error("Error creating API key:", error);
    res.status(500).json({ message: "Failed to create API key" });
  }
});

app.delete("/api/user/api-keys/:keyId", isAuthenticated, async (req, res) => {
  try {
    const userId = req.user.claims.sub;
    const { keyId } = req.params;

    const deleted = await storage.deleteApiKey(keyId, userId);
    if (!deleted) {
      return res.status(404).json({ message: "API key not found" });
    }
    res.json({ success: true });
  } catch (error) {
    console.error("Error deleting API key:", error);
    res.status(500).json({ message: "Failed to delete API key" });
  }
});

// =====
// API ROUTES FOR REVIT ADD-IN (API Key Auth)
// =====

const isApiKeyAuthenticated = async (req, res, next) => {
  const apiKey = req.headers["x-api-key"];

  if (!apiKey || typeof apiKey !== "string") {
    return res.status(401).json({ message: "API key required" });
  }

  const user = await storage.validateApiKey(apiKey);
  if (!user) {
    return res.status(401).json({ message: "Invalid API key" });
  }

  req.apiUser = user;
  next();
};

app.get("/api/addin/validate", isApiKeyAuthenticated, async (req, res) => {
  res.json({ valid: true, userId: req.apiUser.id });
});

app.post("/api/addin/create-order", isApiKeyAuthenticated, async (req, res) => {
  try {
    const userId = req.apiUser.id;
    const parsed = createOrderRequestSchema.safeParse(req.body);

    if (!parsed.success) {
      return res.status(400).json({ message: "Invalid request", errors: parsed.error.errors });
    }

    const { sheetCount } = parsed.data;
    const totalPriceSar = sheetCount * PRICE_PER_SHEET_SAR;

    const order = await storage.createOrder({
      userId,
      sheetCount,
      totalPriceSar,
      status: "pending",
    });

    const stripe = await getUncachableStripeClient();

    const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
      line_items: [
        {
          price_data: {
            currency: "sar",
            product_data: {
              name: `LOD 400 Sheet Upgrade (${order.sheetCount} sheets)`,
              description: `Professional LOD 300 to LOD 400 model upgrade for ${order.sheetCount} sheets`,
            },
            unit_amount: order.totalPriceSar * 100,
          },
          quantity: 1,
        },
      ],
      mode: "payment",
      success_url: `https://${process.env.REPLIT_DOMAINS?.split(',')[0]}?payment=success&order=${order.id}`,
      cancel_url: `https://${process.env.REPLIT_DOMAINS?.split(',')[0]}?payment=cancelled&order=${order.id}`,
      metadata: {
        orderId: order.id,
        userId,
      },
    });
  }
});

```

```

    await storage.updateOrder(order.id, { stripeSessionId: session.id });

    res.status(201).json({
      order,
      checkoutUrl: session.url
    });
  } catch (error) {
    console.error("Error creating order:", error);
    res.status(500).json({ message: "Failed to create order" });
  }
});

app.get("/api/addin/orders", isApiKeyAuthenticated, async (req: any, res) => {
  try {
    const userId = req.apiUser.id;
    const orders = await storage.getOrdersByUserId(userId);
    res.json(orders);
  } catch (error) {
    console.error("Error fetching orders:", error);
    res.status(500).json({ message: "Failed to fetch orders" });
  }
});

app.get("/api/addin/orders/:orderId/status", isApiKeyAuthenticated, async (req: any, res) => {
  try {
    const userId = req.apiUser.id;
    const { orderId } = req.params;

    const order = await storage.getOrderWithFiles(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.userId !== userId) {
      return res.status(403).json({ message: "Forbidden" });
    }

    res.json(order);
  } catch (error) {
    console.error("Error checking order status:", error);
    res.status(500).json({ message: "Failed to check order status" });
  }
});

app.post("/api/addin/orders/:orderId/upload-url", isApiKeyAuthenticated, async (req: any, res) => {
  try {
    const userId = req.apiUser.id;
    const { orderId } = req.params;
    const { fileName } = req.body;

    if (!fileName) {
      return res.status(400).json({ message: "fileName is required" });
    }

    const order = await storage.getOrder(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.userId !== userId) {
      return res.status(403).json({ message: "Forbidden" });
    }

    if (order.status !== "paid") {
      return res.status(400).json({ message: "Order must be paid before uploading files" });
    }

    const uploadURL = await objectStorage.getUploadURL(orderId, fileName);
    res.json({ uploadURL });
  } catch (error) {
    console.error("Error getting upload URL:", error);
    res.status(500).json({ message: "Failed to get upload URL" });
  }
});

app.post("/api/addin/orders/:orderId/upload-complete", isApiKeyAuthenticated, async (req: any, res) => {
  try {
    const userId = req.apiUser.id;
    const { orderId } = req.params;
    const { fileName, fileSize, uploadURL } = req.body;

    if (!fileName || !uploadURL) {
      return res.status(400).json({ message: "fileName and uploadURL are required" });
    }

    const order = await storage.getOrder(orderId);
    if (!order) {
      return res.status(404).json({ message: "Order not found" });
    }

    if (order.userId !== userId) {
      return res.status(403).json({ message: "Forbidden" });
    }

    const storageKey = objectStorage.normalizeStorageKey(uploadURL);

    await storage.createFile({
      orderId,
      fileType: "input",
      fileName,
      fileSize: fileSize || null,
    });
  }
});

```

```

        storageKey,
        mimeType: "application/zip",
    });

    await storage.updateOrderStatus(orderId, "uploaded");
    res.json({ success: true });
} catch (error) {
    console.error("Error completing upload:", error);
    res.status(500).json({ message: "Failed to complete upload" });
}
});

app.get("/api/addin/orders/:orderId/download-url", isApiKeyAuthenticated, async (req: any, res) => {
    try {
        const userId = req.apiUser.id;
        const { orderId } = req.params;

        const order = await storage.getOrderWithFiles(orderId);
        if (!order) {
            return res.status(404).json({ message: "Order not found" });
        }

        if (order.userId !== userId) {
            return res.status(403).json({ message: "Forbidden" });
        }

        if (order.status !== "complete") {
            return res.status(400).json({ message: "Order is not complete" });
        }

        const outputFile = order.files?.find(f => f.fileType === "output");
        if (!outputFile) {
            return res.status(404).json({ message: "No deliverables found" });
        }

        const downloadURL = await objectStorage.getDownloadURL(outputFile.storageKey);
        res.json({ downloadURL, fileName: outputFile.fileName });
    } catch (error) {
        console.error("Error getting download URL:", error);
        res.status(500).json({ message: "Failed to get download URL" });
    }
});

app.get("/api/orders/:orderId/download-url", isAuthenticated, async (req: any, res) => {
    try {
        const userId = req.user.claims.sub;
        const { orderId } = req.params;

        const order = await storage.getOrderWithFiles(orderId);
        if (!order) {
            return res.status(404).json({ message: "Order not found" });
        }

        if (order.userId !== userId) {
            return res.status(403).json({ message: "Forbidden" });
        }

        if (order.status !== "complete") {
            return res.status(400).json({ message: "Order is not complete" });
        }

        const outputFile = order.files?.find(f => f.fileType === "output");
        if (!outputFile) {
            return res.status(404).json({ message: "No deliverables found" });
        }

        const downloadURL = await objectStorage.getDownloadURL(outputFile.storageKey);
        res.json({ downloadURL, fileName: outputFile.fileName });
    } catch (error) {
        console.error("Error getting download URL:", error);
        res.status(500).json({ message: "Failed to get download URL" });
    }
});

// Get Stripe publishable key for frontend
app.get("/api/stripe/config", async (req, res) => {
    try {
        const { getStripePublishableKey } = await import("./stripeClient");
        const publishableKey = await getStripePublishableKey();
        res.json({ publishableKey });
    } catch (error) {
        console.error("Error getting Stripe config:", error);
        res.status(500).json({ message: "Failed to get Stripe configuration" });
    }
});

return httpServer;
}

```

File: server/static.ts

```
import express, { type Express } from "express";
import fs from "fs";
import path from "path";

export function serveStatic(app: Express) {
  const distPath = path.resolve(__dirname, "public");
  if (!fs.existsSync(distPath)) {
    throw new Error(
      `Could not find the build directory: ${distPath}, make sure to build the client first`,
    );
  }

  app.use(express.static(distPath));

  // fall through to index.html if the file doesn't exist
  app.use("*", (_req, res) => {
    res.sendFile(path.resolve(distPath, "index.html"));
  });
}
```

File: server/storage.ts

```
import {
  users,
  orders,
  files,
  apiKeys,
  type User,
  type UpsertUser,
  type Order,
  type InsertOrder,
  type File as FileRecord,
  type InsertFile,
  type OrderWithFiles,
  type ApiKey,
  type InsertApiKey,
} from "@shared/schema";
import { db } from "../db";
import { eq, desc, and, sql } from "drizzle-orm";
import crypto from "crypto";

export interface IStorage {
  // User operations
  getUser(id: string): Promise<User | undefined>;
  upsertUser(user: UpsertUser): Promise<User>;
  getAllUsers(): Promise<User[]>;
  getUsersWithOrderStats(): Promise<Array<User & { orderCount: number; totalSpent: number }>>;

  // Order operations
  createOrder(order: InsertOrder): Promise<Order>;
  getOrder(id: string): Promise<Order | undefined>;
  getOrderWithFiles(id: string): Promise<OrderWithFiles | undefined>;
  getOrdersByUserId(userId: string): Promise<OrderWithFiles[]>;
  getAllOrders(): Promise<OrderWithFiles[]>;
  updateOrder(id: string, data: Partial<Order>): Promise<Order | undefined>;
  updateOrderStatus(id: string, status: Order["status"]): Promise<Order | undefined>;

  // File operations
  createFile(file: InsertFile): Promise<FileRecord>;
  getFile(id: string): Promise<FileRecord | undefined>;
  getFilesByOrderId(orderId: string): Promise<FileRecord[]>;

  // API Key operations
  createApiKey(userId: string, name: string): Promise<{ apiKey: ApiKey; rawKey: string }>;
  getApiKeysByUserId(userId: string): Promise<ApiKey[]>;
  validateApiKey(rawKey: string): Promise<User | null>;
  deleteApiKey(id: string, userId: string): Promise<boolean>;
  updateApiKeyLastUsed(id: string): Promise<void>;
}

export class DatabaseStorage implements IStorage {
  // User operations
  async getUser(id: string): Promise<User | undefined> {
    const [user] = await db.select().from(users).where(eq(users.id, id));
    return user;
  }

  async upsertUser(userData: UpsertUser): Promise<User> {
    const [user] = await db
      .insert(users)
      .values(userData)
      .onConflictDoUpdate({
        target: users.id,
        set: {
          ...userData,
          updatedAt: new Date(),
        },
      })
      .returning();
    return user;
  }

  async getAllUsers(): Promise<User[]> {
    return await db.select().from(users).orderBy(desc(users.createdAt));
  }

  async getUsersWithOrderStats(): Promise<Array<User & { orderCount: number; totalSpent: number }>> {
    const result = await db
      .select({
        id: users.id,
        email: users.email,
        firstName: users.firstName,
        lastName: users.lastName,
        profileImageUrl: users.profileImageUrl,
        isAdmin: users.isAdmin,
        createdAt: users.createdAt,
        updatedAt: users.updatedAt,
        orderCount: sql<number>`COALESCE(COUNT(${orders.id}), 0)::int`,
        totalSpent: sql<number>`COALESCE(SUM(CASE WHEN ${orders.status} != 'pending' THEN ${orders.totalPriceSar} ELSE 0 END), 0)::int`,
      })
      .from(users)
      .leftJoin(orders, eq(users.id, orders.userId))
      .where(eq(users.isAdmin, 0))
      .groupBy(users.id)
      .orderBy(desc(users.createdAt));

    return result;
  }

  // Order operations
```

```

async createOrder(order: InsertOrder): Promise<Order> {
  const [newOrder] = await db.insert(orders).values(order).returning();
  return newOrder;
}

async getOrder(id: string): Promise<Order | undefined> {
  const [order] = await db.select().from(orders).where(eq(orders.id, id));
  return order;
}

async getOrderWithFiles(id: string): Promise<OrderWithFiles | undefined> {
  const order = await this.getOrder(id);
  if (!order) return undefined;

  const orderFiles = await this.GetFilesByOrderId(id);
  const user = await this.getUser(order.userId);

  return {
    ...order,
    files: orderFiles,
    user: user ? {
      id: user.id,
      email: user.email,
      firstName: user.firstName,
      lastName: user.lastName,
      profileImageUrl: user.profileImageUrl,
    } : undefined,
  };
}

async getOrdersByUserId(userId: string): Promise<OrderWithFiles[]> {
  const userOrders = await db
    .select()
    .from(orders)
    .where(eq(orders.userId, userId))
    .orderBy(desc(orders.createdAt));

  const ordersWithFiles: OrderWithFiles[] = [];
  for (const order of userOrders) {
    const orderFiles = await this.GetFilesByOrderId(order.id);
    ordersWithFiles.push({
      ...order,
      files: orderFiles,
    });
  }

  return ordersWithFiles;
}

async getAllOrders(): Promise<OrderWithFiles[]> {
  const allOrders = await db
    .select()
    .from(orders)
    .orderBy(desc(orders.createdAt));

  const ordersWithFiles: OrderWithFiles[] = [];
  for (const order of allOrders) {
    const orderFiles = await this.GetFilesByOrderId(order.id);
    const user = await this.getUser(order.userId);
    ordersWithFiles.push({
      ...order,
      files: orderFiles,
      user: user ? {
        id: user.id,
        email: user.email,
        firstName: user.firstName,
        lastName: user.lastName,
        profileImageUrl: user.profileImageUrl,
      } : undefined,
    });
  }

  return ordersWithFiles;
}

async updateOrder(id: string, data: Partial<Order>): Promise<Order | undefined> {
  const [updated] = await db
    .update(orders)
    .set({ ...data, updatedAt: new Date() })
    .where(eq(orders.id, id))
    .returning();
  return updated;
}

async updateOrderStatus(id: string, status: Order["status"]): Promise<Order | undefined> {
  const now = new Date();
  const updateData: Partial<Order> = { status, updatedAt: now };

  if (status === "paid") {
    updateData.paidAt = now;
  } else if (status === "uploaded") {
    updateData.uploadedAt = now;
  } else if (status === "complete") {
    updateData.completedAt = now;
  }

  const [updated] = await db
    .update(orders)
    .set(updateData)
    .where(eq(orders.id, id))
    .returning();
}

```

```

    return updated;
  }

  // File operations
  async createFile(file: InsertFile): Promise<FileRecord> {
    const [newFile] = await db.insert(files).values(file).returning();
    return newFile;
  }

  async getFile(id: string): Promise<FileRecord | undefined> {
    const [file] = await db.select().from(files).where(eq(files.id, id));
    return file;
  }

  async getFilesByOrderId(orderId: string): Promise<FileRecord[]> {
    return await db.select().from(files).where(eq(files.orderId, orderId));
  }

  // API Key operations
  private hashApiKey(key: string): string {
    return crypto.createHash('sha256').update(key).digest('hex');
  }

  private generateApiKey(): string {
    return `lod400_${crypto.randomBytes(32).toString('hex')}`;
  }

  async createApiKey(userId: string, name: string): Promise<{ apiKey: ApiKey; rawKey: string }> {
    const rawKey = this.generateApiKey();
    const keyHash = this.hashApiKey(rawKey);

    const [apiKey] = await db.insert(apiKeys).values({
      userId,
      name,
      keyHash,
    }).returning();

    return { apiKey, rawKey };
  }

  async getApiKeysByUserId(userId: string): Promise<ApiKey[]> {
    return await db.select().from(apiKeys)
      .where(eq(apiKeys.userId, userId))
      .orderBy(desc(apiKeys.createdAt));
  }

  async validateApiKey(rawKey: string): Promise<User | null> {
    const keyHash = this.hashApiKey(rawKey);

    const [result] = await db.select()
      .from(apiKeys)
      .where(eq(apiKeys.keyHash, keyHash));

    if (!result) return null;

    await this.updateApiKeyLastUsed(result.id);

    const user = await this.getUser(result.userId);
    return user || null;
  }

  async deleteApiKey(id: string, userId: string): Promise<boolean> {
    const result = await db.delete(apiKeys)
      .where(and(eq(apiKeys.id, id), eq(apiKeys.userId, userId)))
      .returning();
    return result.length > 0;
  }

  async updateApiKeyLastUsed(id: string): Promise<void> {
    await db.update(apiKeys)
      .set({ lastUsed: new Date() })
      .where(eq(apiKeys.id, id));
  }
}

export const storage = new DatabaseStorage();

```

File: server/stripeClient.ts

```
import Stripe from 'stripe';

let connectionSettings: any;

async function getCredentials() {
  const hostname = process.env.REPLIT_CONNECTORS_HOSTNAME;
  const xReplitToken = process.env.REPL_IDENTITY
    ? 'repl ' + process.env.REPL_IDENTITY
    : process.env.WEB_REPL_RENEWAL
    ? 'depl ' + process.env.WEB_REPL_RENEWAL
    : null;

  if (!xReplitToken) {
    throw new Error('X_REPLIT_TOKEN not found for repl/depl');
  }

  const connectorName = 'stripe';
  const isProduction = process.env.REPLIT_DEPLOYMENT === '1';
  const targetEnvironment = isProduction ? 'production' : 'development';

  const url = new URL(`https://${hostname}/api/v2/connection`);
  url.searchParams.set('include_secrets', 'true');
  url.searchParams.set('connector_names', connectorName);
  url.searchParams.set('environment', targetEnvironment);

  const response = await fetch(url.toString(), {
    headers: {
      'Accept': 'application/json',
      'X_REPLIT_TOKEN': xReplitToken
    }
  });

  const data = await response.json();

  connectionSettings = data.items?.[0];

  if (!connectionSettings || (!connectionSettings.settings.publishable || !connectionSettings.settings.secret)) {
    throw new Error(`Stripe ${targetEnvironment} connection not found`);
  }

  return {
    publishableKey: connectionSettings.settings.publishable,
    secretKey: connectionSettings.settings.secret,
  };
}

export async function getUncachableStripeClient() {
  const { secretKey } = await getCredentials();

  return new Stripe(secretKey, {
    apiVersion: '2025-08-27.basil' as any,
  });
}

export async function getStripePublishableKey() {
  const { publishableKey } = await getCredentials();
  return publishableKey;
}

export async function getStripeSecretKey() {
  const { secretKey } = await getCredentials();
  return secretKey;
}

let stripeSync: any = null;

export async function getStripeSync() {
  if (!stripeSync) {
    const { StripeSync } = await import('stripe-replit-sync');
    const secretKey = await getStripeSecretKey();

    stripeSync = new StripeSync({
      poolConfig: {
        connectionString: process.env.DATABASE_URL!,
        max: 2,
      },
      stripeSecretKey: secretKey,
    });
  }
  return stripeSync;
}
```

File: server/vite.ts

```
import { type Express } from "express";
import { createServer as createViteServer, createLogger } from "vite";
import { type Server } from "http";
import viteConfig from "../vite.config";
import fs from "fs";
import path from "path";
import { nanoid } from "nanoid";

const viteLogger = createLogger();

export async function setupVite(server: Server, app: Express) {
  const serverOptions = {
    middlewareMode: true,
    hmr: { server, path: "/vite-hmr" },
    allowedHosts: true as const,
  };

  const vite = await createViteServer({
    ...viteConfig,
    configFile: false,
    customLogger: {
      ...viteLogger,
      error: (msg, options) => {
        viteLogger.error(msg, options);
        process.exit(1);
      },
    },
    server: serverOptions,
    appType: "custom",
  });

  app.use(vite.middlewares);

  app.use("", async (req, res, next) => {
    const url = req.originalUrl;

    try {
      const clientTemplate = path.resolve(
        import.meta.dirname,
        "..",
        "client",
        "index.html",
      );

      // always reload the index.html file from disk incase it changes
      let template = await fs.promises.readFile(clientTemplate, "utf-8");
      template = template.replace(
        `src="/src/main.tsx"`,
        `src="/src/main.tsx?v=${nanoid()}"`,
      );

      const page = await vite.transformIndexHtml(url, template);
      res.status(200).set({ "Content-Type": "text/html" }).end(page);
    } catch (e) {
      vite.ssrFixStacktrace(e as Error);
      next(e);
    }
  });
}
```

File: server/webhookHandlers.ts

```
import { getStripeSync, getUncachableStripeClient } from './stripeClient';
import { storage } from './storage';

export class WebhookHandlers {
  static async processWebhook(payload: Buffer, signature: string, uuid: string): Promise<void> {
    if (!Buffer.isBuffer(payload)) {
      throw new Error(
        'STRIPE WEBHOOK ERROR: Payload must be a Buffer. ' +
        'Received type: ' + typeof payload + '. ' +
        'This usually means express.json() parsed the body before reaching this handler. ' +
        'FIX: Ensure webhook route is registered BEFORE app.use(express.json()).'
      );
    }

    const sync = await getStripeSync();

    const stripe = await getUncachableStripeClient();
    const webhooks = await stripe.webhookEndpoints.list({ limit: 10 });
    const webhookEndpoint = webhooks.data.find(w => w.url?.includes(uuid));

    if (!webhookEndpoint?.secret) {
      throw new Error('Webhook endpoint secret not found');
    }

    const event = stripe.webhooks.constructEvent(payload, signature, webhookEndpoint.secret);

    switch (event.type) {
      case 'checkout.session.completed': {
        const session = event.data.object;
        await WebhookHandlers.handleCheckoutCompleted(session);
        break;
      }
      case 'checkout.session.expired': {
        const session = event.data.object;
        console.log(`Checkout session expired for order ${session.metadata?.orderId}`);
        break;
      }
    }

    await sync.processWebhook(payload, signature, uuid);
  }

  static async handleCheckoutCompleted(session: any): Promise<void> {
    const orderId = session.metadata?.orderId;
    if (orderId) {
      await storage.updateOrder(orderId, {
        stripePaymentIntentId: session.payment_intent as string,
      });
      await storage.updateOrderStatus(orderId, "paid");
      console.log(`Order ${orderId} payment completed via webhook`);
    }
  }
}
```

File: shared/schema.ts

```
import { sql, relations } from "drizzle-orm";
import {
  index,
  jsonb,
  pgTable,
  timestamp,
  varchar,
  integer,
  text,
  pgEnum,
} from "drizzle-orm/pg-core";
import { createInsertSchema } from "drizzle-zod";
import { z } from "zod";

// Order status enum
export const orderStatusEnum = pgEnum("order_status", [
  "pending",
  "paid",
  "uploaded",
  "processing",
  "complete",
]);

// File type enum
export const fileTypeEnum = pgEnum("file_type", ["input", "output"]);

// Session storage table for Replit Auth
export const sessions = pgTable(
  "sessions",
  {
    sid: varchar("sid").primaryKey(),
    sess: jsonb("sess").notNull(),
    expire: timestamp("expire").notNull(),
  },
  (table) => [index("IDX_session_expire").on(table.expire)],
);

// Users table for Replit Auth
export const users = pgTable("users", {
  id: varchar("id").primaryKey().default(sql`gen_random_uuid()`),
  email: varchar("email").unique(),
  firstName: varchar("first_name"),
  lastName: varchar("last_name"),
  profileImageUrl: varchar("profile_image_url"),
  isAdmin: integer("is_admin").default(0),
  createdAt: timestamp("created_at").defaultNow(),
  updatedAt: timestamp("updated_at").defaultNow(),
});

// Orders table
export const orders = pgTable("orders", {
  id: varchar("id").primaryKey().default(sql`gen_random_uuid()`),
  userId: varchar("user_id").notNull().references(() => users.id),
  sheetCount: integer("sheet_count").notNull(),
  totalPriceSar: integer("total_price_sar").notNull(),
  status: orderStatusEnum("status").notNull().default("pending"),
  stripeSessionId: varchar("stripe_session_id"),
  stripePaymentIntentId: varchar("stripe_payment_intent_id"),
  notes: text("notes"),
  createdAt: timestamp("created_at").defaultNow(),
  updatedAt: timestamp("updated_at").defaultNow(),
  paidAt: timestamp("paid_at"),
  uploadedAt: timestamp("uploaded_at"),
  completedAt: timestamp("completed_at"),
});

// Files table
export const files = pgTable("files", {
  id: varchar("id").primaryKey().default(sql`gen_random_uuid()`),
  orderId: varchar("order_id").notNull().references(() => orders.id),
  fileType: fileTypeEnum("file_type").notNull(),
  fileName: varchar("file_name").notNull(),
  fileSize: integer("file_size"),
  storageKey: varchar("storage_key").notNull(),
  mimeType: varchar("mime_type"),
  createdAt: timestamp("created_at").defaultNow(),
});

// API Keys table for Replit add-in authentication
export const apiKeys = pgTable("api_keys", {
  id: varchar("id").primaryKey().default(sql`gen_random_uuid()`),
  userId: varchar("user_id").notNull().references(() => users.id),
  name: varchar("name").notNull(),
  keyHash: varchar("key_hash").notNull(),
  lastUsed: timestamp("last_used"),
  createdAt: timestamp("created_at").defaultNow(),
});

// Relations
export const usersRelations = relations(users, ({ many }) => ({
  orders: many(orders),
  apiKeys: many(apiKeys),
}));

export const apiKeysRelations = relations(apiKeys, ({ one }) => ({
  user: one(users, {
    fields: [apiKeys.userId],
  }),
}));
```

```

    references: [users.id],
  }},
});

export const ordersRelations = relations(orders, ({ one, many }) => ({
  user: one(users, {
    fields: [orders.userId],
    references: [users.id],
  }),
  files: many(files),
}));

export const filesRelations = relations(files, ({ one }) => ({
  order: one(orders, {
    fields: [files.orderId],
    references: [orders.id],
  }),
}));

// Insert schemas
export const insertUserSchema = createInsertSchema(users).omit({
  id: true,
  createdAt: true,
  updatedAt: true,
});

export const insertOrderSchema = createInsertSchema(orders).omit({
  id: true,
  createdAt: true,
  updatedAt: true,
  paidAt: true,
  uploadedAt: true,
  completedAt: true,
});

export const insertFileSchema = createInsertSchema(files).omit({
  id: true,
  createdAt: true,
});

export const insertApiKeySchema = createInsertSchema(apiKeys).omit({
  id: true,
  createdAt: true,
  lastUsed: true,
});

// Types
export type UpsertUser = typeof users.$inferInsert;
export type User = typeof users.$inferSelect;
export type InsertOrder = z.infer<typeof insertOrderSchema>;
export type Order = typeof orders.$inferSelect;
export type InsertFile = z.infer<typeof insertFileSchema>;
export type File = typeof files.$inferSelect;
export type InsertApiKey = z.infer<typeof insertApiKeySchema>;
export type ApiKey = typeof apiKeys.$inferSelect;

// API request/response types
export const createOrderRequestSchema = z.object({
  sheetCount: z.number().min(1).max(1000),
});

export type CreateOrderRequest = z.infer<typeof createOrderRequestSchema>;

export const orderWithFilesSchema = z.object({
  id: z.string(),
  userId: z.string(),
  sheetCount: z.number(),
  totalPriceSar: z.number(),
  status: z.enum(["pending", "paid", "uploaded", "processing", "complete"]),
  stripeSessionId: z.string().nullable(),
  stripePaymentIntentId: z.string().nullable(),
  notes: z.string().nullable(),
  createdAt: z.date().nullable(),
  updatedAt: z.date().nullable(),
  paidAt: z.date().nullable(),
  uploadedAt: z.date().nullable(),
  completedAt: z.date().nullable(),
  user: z.object({
    id: z.string(),
    email: z.string().nullable(),
    firstName: z.string().nullable(),
    lastName: z.string().nullable(),
    profileImageUrl: z.string().nullable(),
  }).optional(),
  files: z.array(z.object({
    id: z.string(),
    orderId: z.string(),
    fileType: z.enum(["input", "output"]),
    fileName: z.string(),
    fileSize: z.number().nullable(),
    storageKey: z.string(),
    mimeType: z.string().nullable(),
    createdAt: z.date().nullable(),
  })).optional(),
});

export type OrderWithFiles = z.infer<typeof orderWithFilesSchema>;

// Price per sheet in SAR
export const PRICE_PER_SHEET_SAR = 150;

```

File: tailwind.config.ts

```
import type { Config } from "tailwindcss";

export default {
  darkMode: ["class"],
  content: ["/client/index.html", "/client/src/**/*.{js,jsx,ts,tsx}"],
  theme: {
    extend: {
      borderRadius: {
        lg: ".5625rem", /* 9px */
        md: ".375rem", /* 6px */
        sm: ".1875rem", /* 3px */
      },
      colors: {
        // Flat / base colors (regular buttons)
        background: "hsl(var(--background) / <alpha-value>)",
        foreground: "hsl(var(--foreground) / <alpha-value>)",
        border: "hsl(var(--border) / <alpha-value>)",
        input: "hsl(var(--input) / <alpha-value>)",
        card: {
          DEFAULT: "hsl(var(--card) / <alpha-value>)",
          foreground: "hsl(var(--card-foreground) / <alpha-value>)",
          border: "hsl(var(--card-border) / <alpha-value>)",
        },
        popover: {
          DEFAULT: "hsl(var(--popover) / <alpha-value>)",
          foreground: "hsl(var(--popover-foreground) / <alpha-value>)",
          border: "hsl(var(--popover-border) / <alpha-value>)",
        },
        primary: {
          DEFAULT: "hsl(var(--primary) / <alpha-value>)",
          foreground: "hsl(var(--primary-foreground) / <alpha-value>)",
          border: "var(--primary-border)",
        },
        secondary: {
          DEFAULT: "hsl(var(--secondary) / <alpha-value>)",
          foreground: "hsl(var(--secondary-foreground) / <alpha-value>)",
          border: "var(--secondary-border)",
        },
        muted: {
          DEFAULT: "hsl(var(--muted) / <alpha-value>)",
          foreground: "hsl(var(--muted-foreground) / <alpha-value>)",
          border: "var(--muted-border)",
        },
        accent: {
          DEFAULT: "hsl(var(--accent) / <alpha-value>)",
          foreground: "hsl(var(--accent-foreground) / <alpha-value>)",
          border: "var(--accent-border)",
        },
        destructive: {
          DEFAULT: "hsl(var(--destructive) / <alpha-value>)",
          foreground: "hsl(var(--destructive-foreground) / <alpha-value>)",
          border: "var(--destructive-border)",
        },
        ring: "hsl(var(--ring) / <alpha-value>)",
        chart: {
          "1": "hsl(var(--chart-1) / <alpha-value>)",
          "2": "hsl(var(--chart-2) / <alpha-value>)",
          "3": "hsl(var(--chart-3) / <alpha-value>)",
          "4": "hsl(var(--chart-4) / <alpha-value>)",
          "5": "hsl(var(--chart-5) / <alpha-value>)",
        },
        sidebar: {
          ring: "hsl(var(--sidebar-ring) / <alpha-value>)",
          DEFAULT: "hsl(var(--sidebar) / <alpha-value>)",
          foreground: "hsl(var(--sidebar-foreground) / <alpha-value>)",
          border: "hsl(var(--sidebar-border) / <alpha-value>)",
        },
        "sidebar-primary": {
          DEFAULT: "hsl(var(--sidebar-primary) / <alpha-value>)",
          foreground: "hsl(var(--sidebar-primary-foreground) / <alpha-value>)",
          border: "var(--sidebar-primary-border)",
        },
        "sidebar-accent": {
          DEFAULT: "hsl(var(--sidebar-accent) / <alpha-value>)",
          foreground: "hsl(var(--sidebar-accent-foreground) / <alpha-value>)",
          border: "var(--sidebar-accent-border)"
        },
        status: {
          online: "rgb(34 197 94)",
          away: "rgb(245 158 11)",
          busy: "rgb(239 68 68)",
          offline: "rgb(156 163 175)",
        },
      },
    },
  },
  fontFamily: {
    sans: ["var(--font-sans)"],
    serif: ["var(--font-serif)"],
    mono: ["var(--font-mono)"],
  },
  keyframes: {
    "accordion-down": {
      from: { height: "0" },
      to: { height: "var(--radix-accordion-content-height)" },
    },
    "accordion-up": {
      from: { height: "var(--radix-accordion-content-height)" },
      to: { height: "0" },
    },
  },
}
```

```
    },
  },
  animation: {
    "accordion-down": "accordion-down 0.2s ease-out",
    "accordion-up": "accordion-up 0.2s ease-out",
  },
},
},
plugins: [require("tailwindcss-animate"), require("@tailwindcss/typography")],
} satisfies Config;
```

File: tsconfig.json

```
{
  "include": ["client/src/**/*", "shared/**/*", "server/**/*"],
  "exclude": ["node_modules", "build", "dist", "**/*.test.ts"],
  "compilerOptions": {
    "incremental": true,
    "tsBuildInfoFile": "./node_modules/typescript/tsbuildinfo",
    "noEmit": true,
    "module": "ESNext",
    "strict": true,
    "lib": ["esnext", "dom", "dom.iterable"],
    "jsx": "preserve",
    "esModuleInterop": true,
    "skipLibCheck": true,
    "allowImportingTsExtensions": true,
    "moduleResolution": "bundler",
    "baseUrl": ".",
    "types": ["node", "vite/client"],
    "paths": {
      "@/*": ["/client/src/*"],
      "@shared/*": ["/shared/*"]
    }
  }
}
```

File: vite.config.ts

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import path from "path";
import runtimeErrorOverlay from "@replit/vite-plugin-runtime-error-modal";

export default defineConfig({
  plugins: [
    react(),
    runtimeErrorOverlay(),
    ...(process.env.NODE_ENV !== "production" & process.env.REPL_ID !== undefined
      ? [
          await import("@replit/vite-plugin-cartographer").then((m) => {
            m.cartographer(),
          }),
          await import("@replit/vite-plugin-dev-banner").then((m) => {
            m.devBanner(),
          }),
        ]
      : []),
  ],
  resolve: {
    alias: {
      "@": path.resolve(import.meta.dirname, "client", "src"),
      "@shared": path.resolve(import.meta.dirname, "shared"),
      "@assets": path.resolve(import.meta.dirname, "attached_assets"),
    },
  },
  root: path.resolve(import.meta.dirname, "client"),
  build: {
    outDir: path.resolve(import.meta.dirname, "dist/public"),
    emptyOutDir: true,
  },
  server: {
    fs: {
      strict: true,
      deny: ["**/*.**"],
    },
  },
});
```