**🚀 Next.js + Zoho One Integration Guide**

**Customer Experience–Driven Engineering Dashboard**

**1. Discovery & Customer Journey Mapping**

**Objectives**

* Define end-to-end lifecycle: **Inquiry → Order → Project → Billing → Delivery → Support**.
* Map customer touchpoints: dashboard login, project tracking, file exchange, billing transparency, support access.
* Identify service provider needs: automation, compliance, analytics, reduced manual tasks.

**Customer POV**

* Single login access.
* Transparent real-time project and billing status.
* Easy uploads/downloads of engineering files.
* Notifications for key milestones, invoices, support updates.

**Service Provider POV**

* All requests automatically stored in **Zoho CRM**.
* Real-time sync with **Zoho Projects, Books, WorkDrive, Desk**.
* Compliance with GDPR & IP protection.
* Centralized reporting and audit trails.

**2. Zoho One Setup & Configuration**

**Core Modules**

* **Zoho CRM** → customer onboarding, order management.
* **Zoho Projects** → milestones, deadlines, ownership visibility.
* **Zoho Books/Subscriptions** → invoices, recurring billing, payment tracking.
* **Zoho WorkDrive** → engineering files (CAD, blueprints, reports).
* **Zoho Forms** → onboarding forms, service requests.
* **Zoho Desk** → support ticketing & knowledge base.

**Best Practices**

* Enable **role-based access**: clients (read/project files), engineers (write/update).
* Configure **GDPR consent forms** in CRM & Forms.
* Map **file permissions** in WorkDrive by client/project folder.

**3. Data Modeling & Security Architecture**

**Data Model**

* **Customer Entity** → CRM as master record.
* **Orders** → linked with CRM + Books.
* **Projects** → linked with CRM + Projects.
* **Invoices & Payments** → Books + Subscriptions.
* **Files** → WorkDrive, linked to Project & Customer.
* **Support Tickets** → Desk, linked to Customer.

**Security**

* **OAuth 2.0 for Zoho APIs**.
* **JWT/Session-based auth for Next.js frontend**.
* **Audit logs**: file uploads, project updates, payments.
* **Encryption**: files at rest (Zoho WorkDrive) + TLS in transit.

**4. API & Auth Layer for Next.js**

**Setup**

* Create **API routes** in /pages/api/zoho/\*.
* Use **Zoho OAuth 2.0 tokens** with refresh token handling.
* Store credentials in **Next.js env variables**.

**Integration Strategy**

* **SSR (Server-Side Rendering)** → critical data (orders, billing).
* **CSR (Client-Side Rendering)** → real-time updates (notifications, project progress).
* **Hybrid** → preload essentials, lazy-load extras.

**5. Feature Implementation**

**5.1 Orders & Project Status**

* Fetch project milestones from **Zoho Projects API**.
* Show **Gantt chart or Kanban view** inside dashboard.

**5.2 Billing & Invoices**

* Integrate **Zoho Books API**.
* Show invoice history + payment status.
* Allow download of **PDF invoices**.

**5.3 Payments & Subscriptions**

* Use **Zoho Subscriptions API** for recurring billing.
* Show active subscriptions, renewal dates.

**5.4 File Uploads & Downloads**

* Connect with **Zoho WorkDrive API**.
* Create **project-specific folders**.
* Enable **secure uploads** (large CAD, blueprints).

**5.5 Contact Forms & Requests**

* Use **Zoho Forms** → push data to **CRM Leads**.
* Embed form inside dashboard with **iframe/SDK**.

**5.6 Notifications & Alerts**

* Use **Zoho Flow/Deluge scripts**:
  + Invoice overdue → email + dashboard alert.
  + Project milestone reached → push + email.
  + File uploaded → engineer notified.

**5.7 Role-Based Access**

* **Clients**: limited to their projects/files.
* **Engineers/Admins**: manage multiple clients/projects.

**5.8 Knowledge Base & Support**

* Embed **Zoho Desk knowledge base**.
* Ticketing system linked to customer account.

**5.9 Audit Logs & Compliance**

* Every action (upload, invoice, ticket) logged.
* Provide **downloadable activity report** for compliance.

**6. Customer Experience Enhancements**

* Real-time updates via **Webhooks** (Zoho → Next.js).
* **Proactive emails**: milestone progress, invoice reminders.
* **Personalized dashboard** (client-specific branding, project stats).

**7. Testing & QA**

* **Functional testing**: orders, files, invoices.
* **Security testing**: penetration tests, access roles.
* **UX validation**: feedback from pilot customers.
* **Compliance review**: GDPR, IP protection.

**8. Deployment & Monitoring**

* Deploy Next.js → **Vercel**.
* API proxy layer for Zoho → **serverless functions**.
* Monitoring → **Zoho Analytics + Sentry (errors)**.
* SLA metrics: uptime, response times, file delivery speed.

**9. Maintenance & Scaling**

* Automate onboarding via **Zoho Flow**.
* Periodic **audit log exports**.
* Onboard new clients with templated WorkDrive + Projects.
* Scale with caching (Redis/CDN for static files).

# ****Phase 1 – Foundations (Easiest)****

### Prompt 1.1 – Dashboard Skeleton

TASK: Scaffold a Next.js 14 TypeScript dashboard skeleton

FILES: /app/dashboard/page.tsx, /app/dashboard/layout.tsx

REQUIREMENTS:

- Pages: /dashboard, /projects, /billing, /files

- Use placeholder/mock JSON data for projects and invoices

- Add sidebar with navigation tabs: Projects, Billing, Files, Support

- Include simple header with logo and user name

- Use tailwind CSS for layout

### Prompt 1.2 – OAuth 2.0 Setup

TASK: Create Zoho OAuth 2.0 login flow

FILES: /app/api/auth/zoho/login/route.ts, /app/api/auth/zoho/callback/route.ts

REQUIREMENTS:

- Use Zoho OAuth 2.0 (India DC)

- Handle token exchange and refresh token storage

- Store refresh tokens encrypted in PostgreSQL

- Return session JWT for frontend use

- Include error handling for failed login

### Prompt 1.3 – Database Schema

TASK: Create PostgreSQL schema for users and mock cache

FILES: /db/schema.sql

REQUIREMENTS:

- Users table: id, contact\_id, account\_id, email, role, zoho\_refresh\_token, created\_at

- Projects\_cache table: id, zoho\_project\_id, account\_id, name, status, milestone\_json, updated\_at

- Invoices\_cache table: id, zoho\_invoice\_id, account\_id, status, amount\_due, due\_date, updated\_at

# ****Phase 2 – Core Integrations (Medium)****

### Prompt 2.1 – Zoho CRM Integration

TASK: Fetch Zoho CRM leads/accounts

FILES: /app/api/zoho/crm/accounts.ts

REQUIREMENTS:

- Use stored OAuth tokens

- Return account info for logged-in user

- Cache response in PostgreSQL for 5 minutes

- Handle token refresh automatically

### Prompt 2.2 – Zoho Projects Integration

TASK: Fetch Zoho Projects tasks/milestones

FILES: /app/api/zoho/projects.ts

REQUIREMENTS:

- Return list of projects for logged-in customer

- Include milestone JSON for progress bar

- Implement SSR for project page

- Include error handling and caching

### Prompt 2.3 – Zoho Books Integration

TASK: Fetch Zoho Books invoices

FILES: /app/api/zoho/invoices.ts

REQUIREMENTS:

- List invoices with status, amount\_due, due\_date

- Provide PDF download links

- Optionally provide Pay Now link (mock if payment not integrated yet)

### Prompt 2.4 – Role-Based Access

TASK: Implement role-based access middleware

FILES: /app/middleware/roles.ts

REQUIREMENTS:

- Roles: client, client-admin, engineer, pm, admin

- Block access to projects/invoices not owned by client

- Return 403 for unauthorized requests

# ****Phase 3 – File Handling & Communication (Medium-Hard)****

### Prompt 3.1 – Zoho WorkDrive Integration

TASK: Implement secure file upload/download

FILES: /app/api/files/upload-url.ts, /app/api/files/list.ts

REQUIREMENTS:

- Project-specific folders in WorkDrive

- Signed upload URL valid 15 min

- List files accessible by logged-in client

- Record each upload/download in audit\_log

### Prompt 3.2 – Contact Forms

TASK: Integrate Zoho Forms for service requests

FILES: /app/dashboard/components/contact-form.tsx

REQUIREMENTS:

- Embed Zoho Form iframe/SDK

- On submit, create CRM Lead and Desk ticket

- Display success/error message on dashboard

# ****Phase 4 – Automation & Workflows (Harder)****

### Prompt 4.1 – Zoho Flow Automation

TASK: Setup webhook handlers for Zoho Flow

FILES: /app/api/webhooks/projects.ts, /app/api/webhooks/books.ts

REQUIREMENTS:

- Projects: milestone complete → update local cache + notify client

- Books: invoice paid → update local cache + send confirmation email

- Handle retries for failed webhook calls

### Prompt 4.2 – Notifications

TASK: Implement notifications system

FILES: /app/api/notifications.ts

REQUIREMENTS:

- Send email + in-app push when invoice overdue, milestone reached, or file uploaded

- Use SWR to fetch unread notifications in dashboard

- Allow client to mark notifications as read

### Prompt 4.3 – Payment Integration

TASK: Integrate Stripe or Zoho Checkout (optional)

FILES: /app/api/payments/checkout.ts

REQUIREMENTS:

- Generate secure payment link

- Update invoice status after payment

- Handle failed payments

# ****Phase 5 – Advanced Features (Hardest)****

### Prompt 5.1 – Audit Logging

TASK: Implement detailed audit logs

FILES: /lib/audit.ts

REQUIREMENTS:

- Record user\_id, action, entity, entity\_id, meta, IP, user-agent, timestamp

- Log: file uploads/downloads, project milestone changes, invoice payments

- Enable querying by admin dashboard

### Prompt 5.2 – Analytics Dashboard

TASK: Display project & billing analytics

FILES: /app/dashboard/components/analytics.tsx

REQUIREMENTS:

- Charts: project completion %, invoice status, recurring revenue

- Use chart.js or similar

- Only visible to PM/admin roles

### Prompt 5.3 – Optional AI Chatbot

TASK: Integrate AI chatbot for customer support

FILES: /app/dashboard/components/chatbot.tsx

REQUIREMENTS:

- Connect to Zoho Desk knowledge base

- Provide suggested answers to client queries

- Allow ticket creation if unresolved

✅ **How to use this with Kiro.dev**

1. Paste each prompt one at a time.
2. Ask Kiro to scaffold **files + API routes + components**.
3. Test each phase in isolation before moving to the next.
4. After Phase 2, you’ll already have a functional dashboard for projects and invoices.

# 📄 spec.md — Ideinstein Customer Dashboard Integration

# Ideinstein Customer Dashboard — Zoho One + Next.js Integration Spec

## Meta

Platform: Next.js 14 (App Router, TypeScript)

Backend: Zoho One (CRM, Projects, Books, Subscriptions, WorkDrive, Desk, Forms)

Database: PostgreSQL

Deployment: Vercel frontend + Neon/Railway Postgres

Security: GDPR compliance, role-based access, engineering IP protection

---

# Phase 1 – Foundations (Easiest)

## Task 1.1 – Dashboard Skeleton

```doc

FILES: /app/dashboard/page.tsx, /app/dashboard/layout.tsx

REQUIREMENTS:

- Pages: /dashboard, /projects, /billing, /files

- Use mock JSON data for projects and invoices

- Sidebar navigation: Projects, Billing, Files, Support

- Header with logo + user name

- Tailwind CSS layout

## Task 1.2 – OAuth 2.0 Setup

FILES: /app/api/auth/zoho/login/route.ts, /app/api/auth/zoho/callback/route.ts

REQUIREMENTS:

- Zoho OAuth 2.0 (India DC)

- Token exchange + refresh token storage

- Encrypted PostgreSQL storage

- Session JWT for frontend

- Error handling for failed login

## Task 1.3 – Database Schema

FILES: /db/schema.sql

REQUIREMENTS:

- Users table: id, contact\_id, account\_id, email, role, zoho\_refresh\_token, created\_at

- Projects\_cache table: id, zoho\_project\_id, account\_id, name, status, milestone\_json, updated\_at

- Invoices\_cache table: id, zoho\_invoice\_id, account\_id, status, amount\_due, due\_date, updated\_at

# Phase 2 – Core Integrations (Medium)

## Task 2.1 – Zoho CRM Integration

FILES: /app/api/zoho/crm/accounts.ts

REQUIREMENTS:

- Fetch accounts for logged-in user

- Use OAuth tokens with auto-refresh

- Cache response 5 min

- Error handling

## Task 2.2 – Zoho Projects Integration

FILES: /app/api/zoho/projects.ts

REQUIREMENTS:

- Fetch projects + milestones

- SSR for project page

- Include milestone JSON

- Caching + error handling

## Task 2.3 – Zoho Books Integration

FILES: /app/api/zoho/invoices.ts

REQUIREMENTS:

- List invoices: status, amount\_due, due\_date

- Provide PDF download

- Optional Pay Now link

## Task 2.4 – Role-Based Access Middleware

FILES: /app/middleware/roles.ts

REQUIREMENTS:

- Roles: client, client-admin, engineer, pm, admin

- Restrict access to unauthorized data

- Return 403 for blocked requests

# Phase 3 – File Handling & Communication (Medium-Hard)

## Task 3.1 – Zoho WorkDrive Integration

FILES: /app/api/files/upload-url.ts, /app/api/files/list.ts

REQUIREMENTS:

- Project-specific folders

- Signed upload URL (15 min)

- List accessible files for client

- Record upload/download in audit\_log

## Task 3.2 – Contact Forms Integration

FILES: /app/dashboard/components/contact-form.tsx

REQUIREMENTS:

- Embed Zoho Form iframe/SDK

- On submit: create CRM Lead + Desk ticket

- Show success/error message

# Phase 4 – Automation & Workflows (Harder)

## Task 4.1 – Zoho Flow Webhooks

FILES: /app/api/webhooks/projects.ts, /app/api/webhooks/books.ts

REQUIREMENTS:

- Projects milestone complete → update cache + notify client

- Books invoice paid → update cache + notify client

- Retry failed webhook calls

## Task 4.2 – Notifications System

FILES: /app/api/notifications.ts

REQUIREMENTS:

- Email + in-app push notifications

- Invoice overdue, milestone reached, file uploaded

- SWR for unread notifications

- Allow marking as read

## Task 4.3 – Payment Integration

FILES: /app/api/payments/checkout.ts

REQUIREMENTS:

- Secure payment link

- Update invoice after payment

- Handle failed payments

# Phase 5 – Advanced Features (Hardest)

## Task 5.1 – Audit Logging

FILES: /lib/audit.ts

REQUIREMENTS:

- Log: user\_id, action, entity, entity\_id, meta, IP, user-agent, timestamp

- Track file uploads/downloads, milestone changes, invoice payments

- Queryable by admin dashboard

## Task 5.2 – Analytics Dashboard

FILES: /app/dashboard/components/analytics.tsx

REQUIREMENTS:

- Charts: project completion %, invoice status, recurring revenue

- Use chart.js or similar

- Visible to PM/admin only

## Task 5.3 – Optional AI Chatbot

FILES: /app/dashboard/components/chatbot.tsx

REQUIREMENTS:

- Connect to Zoho Desk knowledge base

- Suggest answers to client queries

- Allow ticket creation if unresolved

# Usage Instructions

1. Paste this spec.md into **Kiro.dev**.
2. Kiro will automatically split **phases → tasks → files**.
3. Execute tasks phase by phase, starting with Phase 1.

##### You said:

Great and also provide as an requirement files which I can share with some freelancer also to get quite as backup

###### ChatGPT said:

Absolutely ✅ — I can convert the full integration spec into a **freelancer-friendly requirement package**. This will be in the form of **requirements files** that clearly describe **features, modules, deliverables, and expected outcomes** without needing the freelancer to decipher internal Kiro prompts.

Here’s how I would structure it:

# ****Ideinstein Customer Dashboard – Freelancer Requirements****

## 1. Project Overview

* **Project Name:** Ideinstein Engineering Dashboard
* **Platform:** Next.js 14 (TypeScript, App Router)
* **Backend Services:** Zoho One (CRM, Projects, Books, Subscriptions, WorkDrive, Desk, Forms)
* **Database:** PostgreSQL
* **Deployment:** Vercel + Postgres hosting
* **Goal:** Build a customer-facing dashboard that allows clients to track projects, billing, files, and support requests.

## 2. Functional Requirements

### 2.1 Dashboard & Navigation

* Pages: /dashboard, /projects, /billing, /files, /support
* Sidebar navigation with clear labels
* Header with logo, username, and notifications icon
* Responsive design using Tailwind CSS

### 2.2 Authentication

* Zoho OAuth 2.0 login (India DC)
* Secure token storage (refresh token + session JWT)
* Role-based access (client vs engineer/admin)

### 2.3 Projects

* Fetch project and milestone data from Zoho Projects
* Display Gantt/Kanban style progress bars
* Show milestone status updates and deadlines

### 2.4 Billing & Payments

* Fetch invoices from Zoho Books/Subscriptions
* Show invoice status, amount due, due dates
* PDF download of invoices
* Optional payment integration (Stripe/Zoho Checkout)

### 2.5 File Handling

* Upload/download project-specific files (CAD, blueprints) via Zoho WorkDrive
* Role-based access (clients: view/upload their files; engineers: full access)
* Audit logs for uploads/downloads

### 2.6 Contact & Support

* Integrate Zoho Forms for customer requests
* Automatic creation of CRM leads and Desk tickets
* Knowledge base access via Zoho Desk

### 2.7 Notifications

* Email, SMS, or in-app push notifications for:
  + Invoice overdue
  + Project milestone reached
  + File uploaded
* Mark notifications as read in dashboard

### 2.8 Audit & Compliance

* Track actions: file uploads/downloads, milestone changes, invoice payments
* Logs accessible by admin/PM for compliance

### 2.9 Analytics (Optional)

* Project completion charts
* Invoice status summary
* Recurring revenue dashboard
* Role-based visibility (admin/PM only)

## 3. Non-Functional Requirements

* **Security:** GDPR compliant, encryption at rest and in transit
* **Scalability:** Multi-client support, modular API structure, caching for APIs
* **Performance:** Fast loading dashboard, SSR for sensitive data
* **Maintainability:** Modular code, well-documented, phase-by-phase deliverables

## 4. Deliverables for Freelancer

1. **Next.js frontend pages** with React components
2. **API layer** for Zoho integrations (CRM, Projects, Books, WorkDrive, Desk, Forms)
3. **Database schema** (PostgreSQL)
4. **Authentication & role-based access control**
5. **Notifications & automation** via Zoho Flow / webhooks
6. **Audit logging system**
7. **Deployment guide** for Vercel + Postgres
8. Optional: **Analytics dashboard and AI chatbot**

## 5. Phase-wise Milestones for Quote

| **Phase** | **Description** | **Expected Deliverables** |
| --- | --- | --- |
| 1 | Dashboard skeleton + OAuth | Pages, layout, mock data, secure login |
| 2 | Core integrations | CRM, Projects, Books APIs, role-based access |
| 3 | File handling & forms | WorkDrive integration, contact forms, support ticket sync |
| 4 | Automation & notifications | Zoho Flow/webhooks, email/in-app notifications, payment links |
| 5 | Advanced features | Audit logs, analytics dashboard, optional chatbot |

## 6. Additional Notes

* Freelancer should provide **clear documentation** of each API endpoint and component.
* Phase-by-phase delivery is preferred for easier testing.
* Must ensure **role-based visibility** and **data security** at all times.
* Freelancer should be able to **hand over a working deployment** (Vercel + DB) at end of project.