PROPOSAL 2: Design + Development Proposal

1. Executive Summary

We propose building a modern, scalable real estate platform using:

- **Frontend:** Next.js
- **Backend:** Laravel (API + CMS)
- **Database:** MySQL
- **Hosting:** AWS (Current Server)
- **CI/CD:** GitHub Actions

This approach ensures:

- High performance
- SEO optimization
- Scalability for large property datasets

Note: We will use Next.js for the frontend and Laravel for the backend. Other packages will be selected by the development team based on requirements.

2. Architecture Overview

Frontend (Next.js)

- Server-Side Rendering (SSR) for property pages
- Static generation for content pages
- SEO optimization (meta tags, structured data)

Backend (Laravel)

- REST API for frontend
- Admin CMS (separate instance)
- Business logic & integrations

Database

- MySQL

CMS Strategy

- Separate Laravel CMS instance
- Website consumes data via API
- Secure admin environment

3. Property Data Integration (XML Feed)

Instead of manual entry:

- Properties will be fetched from XML feed
- Manual property add option will be there.

Cron job will:

- Fetch XML data
- Parse & transform
- Store/update in database

Benefits

- Automated updates
- No manual duplication
- Real-time listings sync

4. Key Features

Public Website
(Aligned with RFP)

- Home page
- Properties Page (sale, rent) for commercial and residential
- Property search (filters, sorting)
- Property detail pages
- Services page
- Services detail page
- Agents
- Areas
- Blog
- Blog Detail page
- Mortgage calculator
- Lead capture system

Admin CMS (Laravel)

- Property management (List of available properties from the XML link)
- Agents management
- Blog & content management
- Leads tracking

Note: Design will only be provided for the public page. For the admin side, we will use a modern template.

5. SEO & Performance Strategy

SEO Implementation

- SSR for all property pages
- Dynamic meta tags
- Structured data (JSON-LD)
- Sitemap & robots.txt

Core Web Vitals (Feasibility)
Based on architecture:

- SSR
- Image optimization
- Efficient APIs

We will aim towards maximum possible realistic scores against performance matrices of google, considering that this is dynamic website with dynamic content.

6. Deployment Strategy

Infrastructure

- AWS EC2 (application servers)
- AWS S3 (media storage)

CI/CD

- GitHub Actions:
    - Auto build & deploy
    - Environment-based deployments

Support Period for 30 days

- Continuous improvements
- Bug fixing
- Performance monitoring

7. Performance Optimization

- Image optimization (WebP)
- Lazy loading
- API caching
- Code splitting

8. Security

- Secure APIs (Laravel)
- Authenticated admin panel
- Input validation
- Environment-based secrets
- HTTPS across all services

9. Important Note (Content Management)

The current website is built on WordPress where:Almost all pages are editable

In the proposed system:

- Only structured content (blog, agents. etc.) will be editable
- Core pages will be component-driven (not fully free-edit)

This tradeoff ensures:

- Better performance
- SEO consistency
- Scalable architecture

10. Timeline

Phase	Duration
Design(Up to 3 revisions)	2–3 weeks
Development	4–5 weeks
Integration (XML + CMS)	2 weeks
Testing & Deployment	1–2 weeks

We need Approx 10-12 weeks for completion (Excluding Public Holidays According to the Pakistan Calendar)

Total Cost:

USD 7000 (Exclusive of VAT)

1.4 Payment Terms

We propose a milestone-based payment structure:

- **20%** – Project initiation
- **20%** – Design approval
- **30%** – Development completion
- **20%** – Testing & staging approval
- **10%** – Final delivery & go-live

11. Conclusion

This approach delivers:

- A modern, scalable property platform
- Strong SEO foundation
- Automated data handling via XML
- Clean separation of frontend and CMS