PROJECT PORTFOLIO – SONI SINGH

PROFILE SUMMARY

Results-driven Full Stack Developer & AI Automation Specialist with 14+ years of experience in building scalable web applications, enterprise WordPress systems, and AI-driven automation platforms. Expertise in WordPress, PHP, React, Angular, Python (Django), REST APIs, OCR pipelines, and performance optimization.

Project Name: Texas Brain Institute – Patient Registration, Clinical Forms & Automated Medical Reporting Platform

Website: https://patients.tbi.clinic/

Tech Stack:

Frontend: HTML5, CSS3, JavaScript, jQuery, AJAX
Backend: WordPress, PHP, Plugin/Module Development
Database: MySQL
Document & Report Generation: PHPWord, DOCX/PDF generation, custom HTML-to-Word formatting engine
OCR & AI Processing: Spatie PDF to Text, Smalot PDF Parser, AWS Textract, BastionGPT, Azure OpenAI/Translator API / OCR-ready integrations
APIs & Integrations: REST API, Twilio SMS OTP, Email Notifications, AWS S3, AWS SNS
Tools & Services: Git, cPanel, WordPress Transients, ACF, custom admin dashboards, secure AJAX upload handlers

Project Description:

Texas Brain Institute's patient portal was a highly customized medical intake, form management, and report automation platform built for patient registration, questionnaire handling, secure form submission, PDF generation, and medico-legal report preparation. The system was designed to manage the complete lifecycle of patient intake—from registration and OTP-based login to dynamic multilingual forms, digital signatures, backend editing, OCR-based medical record extraction, and generation of structured clinical reports such as Evaluation Reports, Correlation Reports, Darren's Summary, TMS Reports, and Outside Medical Records Summary.

The platform supported both patient-facing and admin-facing workflows. On the patient side, it enabled secure registration, login with mobile OTP, dynamic questionnaires, local save/resume functionality, bilingual support (English/Spanish), signature capture, conditional form logic, and prevention of duplicate or incomplete submissions. On the admin side, it provided dashboards for editing entries, tracking submissions, managing PDFs, reviewing extracted medical records, and generating highly structured Word/PDF reports used for clinical and legal purposes.

A major part of the project involved solving complex medical-document processing challenges. The system handled text-based PDFs, scanned/image-based reports, neuroimaging records, MMSE forms, ImPACT reports, and outside medical records by using a hybrid OCR architecture and AI-assisted extraction workflow. The final solution significantly reduced manual work for doctors and staff while improving consistency, compliance, and turnaround time.

Role:

Full Stack Developer / WordPress Developer / AI Automation & Medical Reporting Engineer

Key Responsibilities:

- Developed the complete patient registration and login workflow with backend verification and secure password-based authentication.
- Implemented mobile OTP verification during login using Twilio SMS integration for additional portal security.
- Built dynamic multi-step patient forms with conditional field visibility based on earlier answers.
- Added data persistence so partially filled forms could be resumed later from the patient side.
- Prevented empty submissions, duplicate form submissions, and unauthorized resubmissions after final sign-off.
- Enabled digital signature functionality with support for draw, type, and update options across multiple forms.
- Added disclosure and consent sections to signature pages and enforced signature handling rules during revisions.
- Developed bilingual workflow support, including Spanish form submission and English-ready output generation.
- Implemented auto-fill and pre-fill logic across Brain, Headache, Pain, Rivermead, QOL, and related questionnaires based on registration data.
- Built new diagram-based forms for Head Injury, Headache, and Body Pain questionnaires.
- Removed "Office Use Only" sections, corrected page numbering, updated date formatting, and restructured PDF field mappings as per medical/legal requirements.

- Updated login, forgot password, reset password, and verification screens to match project branding and usability needs.
- Reworked email templates and integrated outgoing communication through the Texas Brain Institute email address.
- Developed admin functionality to edit patient entries, highlight submitted vs pending forms, and download entries in PDF format.
- Built report-generation modules for Comprehensive Evaluation Report, Correlation Report, Darren's Summary, TMS Report, Treatment Plan workflows, and Outside Medical Records Summary.
- Implemented structured neuroimaging tables, ImPACT report support, Rivermead and QOL form integrations, and Accident Information form workflows.
- Created a custom DOCX generation engine using PHPWord to convert HTML medical narratives into professionally formatted Word documents.
- Developed custom formatting parsers to maintain Times New Roman styling, justified paragraphs, bullets, headings, and multi-paragraph medical narratives in final reports.
- Built secure AJAX-based PDF upload handlers for external medical records, MMSE, ImPACT, and neuroimaging reports.
- Designed and implemented a hybrid OCR pipeline using Spatie PDF to Text, Smalot PDF Parser, and AWS Textract for structured extraction from both text-based and scanned PDFs.
- Added asynchronous OCR handling with AWS Textract, S3 storage, SNS callbacks, polling logic, retry handling, and chunked page processing.
- Improved extraction reliability for handwritten and mixed-content medical documents, especially MMSE forms.
- Integrated BastionGPT-based structured summarization to convert OCR output into medically formatted narratives with strict factual rules.
- Applied medical-legal content rules such as positive-facts-only extraction, no unsupported inference, standardized pronouns, date formatting, and fallback handling for incomplete data.
- Developed multi-user locking systems using WordPress transients to prevent concurrent edits on Evaluation, Correlation, Darren, and TMS reports.
- Added heartbeat APIs, auto-lock release, token validation, and conflict messaging for safe report editing by multiple clinic users.
- Created report recreation, comparison, re-download, versioning, and change-tracking features for admin users.
- Improved performance, secure endpoint handling, memory usage, duplicate file prevention, and fallback logic for large report generation workflows.

My Contribution / How I Worked on This Project:

I worked on this project end-to-end as both the application developer and the workflow architect. I first analyzed the clinic's intake and reporting process in detail and translated it into a digital system that could support real-world medical operations without disrupting legal or clinical reliability. Instead of treating the project as only a form-building task, I approached it as a connected ecosystem involving patient onboarding, data capture, multilingual usability, document processing, report generation, and secure admin workflows.

I started by building the patient registration and authentication layer, including backend verification, secure login, and OTP-based access control. After that, I created the main questionnaire system with conditional logic, auto-populated fields, and save/resume functionality so patients could complete long medical forms without losing progress. Once the patient-side workflow was stable, I focused on PDF generation, signature workflows, and backend edit control to make sure submitted documents remained protected and traceable.

The next major phase was report automation. I built structured reporting workflows for Evaluation Reports, Correlation Reports, Darren's Summary, TMS reports, and Outside Medical Records Summary. These reports were not simple exports; they required deep logic for conditional medical statements, intake mapping, symptom correlations, neuroimaging integration, ICD handling, and narrative standardization. To support this, I designed modular PHP logic and reusable templates so different report types could share architecture while still remaining functionally separate.

A major technical area I handled was OCR and AI-driven extraction from medical records. Because the project involved text PDFs, scanned reports, handwritten MMSE forms, imaging reports, and mixed-layout clinical documents, I implemented a hybrid OCR approach. I used text extraction where possible for speed and lower cost, then routed difficult/scanned documents through OCR workflows. I also prepared structured prompts and downstream integrations so extracted content could be converted into medically usable summaries for the final reports.

Throughout the project, I worked iteratively with ongoing change requests from clinic stakeholders. Many updates affected field positions, PDF layouts, signature rules, report placeholders, and extraction logic, so I had to keep the codebase flexible, modular, and stable while ensuring that live clinic usage was not disrupted.

Challenges Faced in the Project:

1. Frequent Form and PDF Changes

One of the biggest challenges was that form fields, PDFs, signatures, and layouts were changing repeatedly during development. In a normal application, field changes are usually limited to frontend or backend updates, but in this project every field change also affected PDF mapping, generated documents, database structures, report templates, and sometimes legal wording. I handled this by creating reusable mapping logic, improving naming consistency, and carefully updating field IDs and output placement rules.

2. Medical-Legal Accuracy Requirements

This project required a high level of accuracy because the generated reports were used in a medical and legal context. A wrongly placed field, unsupported statement, or inconsistent summary could create serious issues. To solve this, I introduced strict narrative rules, standardized formatting, positive-only extraction logic, and controlled report generation paths to reduce the risk of incorrect output.

3. Processing Scanned and Complex PDFs

Many uploaded reports were scanned or image-based PDFs, which could not be handled reliably by simple text extraction tools. Some files also caused high server memory consumption. To address this, I implemented a hybrid OCR architecture with multiple fallback layers, async processing, chunk-based extraction, and external OCR options so the platform could support both normal and difficult documents more safely.

4. Handwritten Form Extraction

MMSE and similar forms included handwriting, table-like layouts, and mixed text areas. Standard extraction methods were not reliable enough for these files. I improved processing by using OCR pipelines that could identify structured rows/cells and then applied post-processing logic to interpret medical scoring correctly.

5. Multi-User Editing Conflicts

Doctors and staff members could access the same report simultaneously, which created a risk of overwriting each other's work. I solved this by developing a concurrency-safe locking system with tokens, transients, heartbeat APIs, and auto-release logic so only one active user could edit a report at a time.

6. Maintaining Exact Report Formatting in Word/PDF Output

Medical reports required professional formatting that matched clinic expectations. Standard HTML-to-DOCX conversion was not enough because it often broke alignment, bullets, fonts, spacing, and paragraph structure. I handled this by creating custom formatting and sanitization logic using PHPWord, so the final DOCX files remained consistent and usable.

7. Performance and Stability Under Heavy Workflows

The system had to support large PDF uploads, OCR processing, multi-step forms, auto-save, and complex report generation. This created challenges around PHP memory, AJAX reliability, retry logic, and duplicate handling. I improved stability through caching, duplicate prevention, polling systems, safe retries, optimized upload handling, and better isolation between report modules.

Tools & Services Used:

- **WordPress + PHP:** For building the main portal, admin modules, AJAX actions, custom workflows, and report engines.
- **MySQL:** For patient data storage, questionnaire responses, and report-linked metadata.
- **JavaScript / jQuery / AJAX:** For dynamic form logic, live validations, autosave-style interactions, and admin search/filter features.
- **Twilio:** For OTP-based mobile authentication.
- **PHPWord:** For DOCX generation and precise report formatting.
- **Spatie PDF to Text / Smalot PDF Parser:** For lightweight and text-based PDF extraction.
- **AWS Textract + S3 + SNS:** For asynchronous OCR and structured extraction from scanned/complex PDFs.
- **BastionGPT / AI Prompting:** For structured medical summarization and extraction logic.
- **Git / cPanel:** For version control, deployment, and server management.

Business / Clinical Impact Delivered:

- Reduced manual effort in patient intake and report preparation.
- Improved consistency of medical and legal reporting outputs.
- Enabled bilingual patient-facing workflows.
- Increased security with OTP login and controlled editing permissions.
- Improved handling of scanned, text-based, and handwritten medical records.
- Provided doctors and staff with faster access to structured reports and downloadable PDFs/DOCX files.
- Built a scalable foundation for additional report modules such as TMS, Treatment Plan, and Past/Outside Medical Records Summary.

Project Name: Multi-Module CMS Portal – Admin Panel & Frontend Platform

Website: https://www.africanporthub.com/

Project Type: Content Management, Publishing, Membership, Directory & Event Portal

Tech Stack:

Frontend: Angular (TypeScript), HTML5, CSS3, Bootstrap, JavaScript, jQuery, AJAX
Backend: Node.js (Express.js – REST API Development)
Database: MySQL
Integrations: Mailchimp, Gmail/SMTP, Google Analytics, Stripe, Razorpay, Google Maps
SEO & Marketing: On-page SEO, Meta Management, Emailer Integration, Ad Placement Management
Tools & Services: Git, cPanel, Admin Dashboard, Custom CRUD APIs, Role-based Access Control

Project Description:

This project was a large-scale content-driven portal built using Angular and Node.js, featuring a custom Admin Panel and dynamic Frontend interface. The system was designed to manage multiple business modules through a centralized dashboard with API-driven architecture.

The platform included news publishing, magazine management, project and development listings, event management, job portal, directory submissions, editorial profiles, team pages, contributors, sponsor showcases, press releases, advertisements, static pages, homepage controls, SEO settings, user management, and admin role management.

The primary objective was to build a scalable and structured system where administrators could control all content, approvals, user data, and business workflows from one place. The frontend was designed as a fast and responsive Single Page Application (SPA) using Angular, ensuring seamless user experience with dynamic pages, filters, and real-time interactions.

The system also supported payment integration, digital magazine access, event ticketing, directory listings, newsletter subscriptions, region-based content display, and modular homepage configurations. The backend was designed using REST APIs to ensure flexibility, scalability, and smooth frontend-backend communication.

Role:

Full Stack Developer (Angular + Node.js) / CMS & Portal Development Engineer

Key Responsibilities:

Developed a modular Admin Panel using Angular integrated with Node.js REST APIs for centralized content management. Built scalable RESTful APIs using Node.js (Express) for all modules with full CRUD operations. Implemented publish/unpublish workflows with admin approval logic across all modules. Designed dynamic data structures including media uploads, categories, regions, SEO metadata, and payment links.

Developed Angular-based frontend with reusable components and modular architecture. Implemented dynamic routing and filter-based listing pages for all modules. Built secure user authentication system including registration, login, and subscription flows. Developed user dashboard for profile management, password updates, and account control. Integrated Stripe and Razorpay payment gateways for subscriptions and digital content. Configured Mailchimp integration for newsletter subscription and campaign sync.

Integrated Google Analytics and SEO settings for performance tracking and search optimization. Developed dynamic homepage controlled via backend APIs. Integrated Google Maps for location-based modules such as projects, jobs, events,

and directory. Implemented role-based access control (RBAC) for admin and sub-admin users. Built media-driven modules supporting images, videos, PDFs, and ad placements. Handled UI redesign including white-theme implementation and section restructuring. Planned advanced features such as secure downloads, order history, messaging system, and A–Z directory filtering.

Admin Panel Modules Delivered:

1. News Management

Built a complete news publishing system with title, images, category, region, author, content, slug, and approval workflow.

2. Magazine Management

Developed digital magazine system with PDF access, pricing, payment integration, and publishing controls.

3. Project & Development Listing

Created project module with media, location, map integration, status, region, industry, and publishing logic.

4. Industry Management

Built structured industry content publishing module.

5. Jobs Listing

Implemented job board with job details, salary, location, maps, and categories.

6. Directory Submission Management

Developed business directory with contact details, maps, industry classification, and approval workflow.

7. Event Management

Built event system with banners, speakers, sponsors, tickets, payment links, and gallery support.

8. Editorial Management

Created editorial board management with profiles and regional filtering.

9. Corporate Identity Management

Developed module for corporate branding and identity content.

10. Contributors Management

Built contributor profile management system.

11. Team Members Management

Developed team profile module with social links.

12. Testimonials Management

Implemented testimonial content module.

13. Gallery Management

Created image and video gallery system.

14. Sponsor Management

Built sponsor showcase module.

15. Press Release Management

Developed press release system with structured content and publishing workflow.

16. Speakers Management

Created speaker profiles for events and webinars.

17. Paid Advertise Management

Implemented advertisement management system with placement controls.

18. Other Pages Management

Built dynamic static page creation module.

19. Header Settings

Configured header branding, video, menu, and CTA settings.

20. Footer Settings

Developed footer content and navigation settings.

21. Social Media Settings

Managed social media links.

22. Home Page Settings

Built modular homepage configuration system.

23. Google Settings

Integrated analytics tracking.

24. Emailer Settings (Mailchimp)

Implemented email subscription and campaign integration.

25. On-page SEO Management

Developed SEO meta management system.

26. User Management

Managed users, subscribers, and membership users.

27. Admin Management

Implemented admin roles and access control.

28. Video Placeholders

Created configurable video slots across the platform.

Frontend Development Delivered:

Built Angular-based Single Page Application (SPA). Developed dynamic pages for News, Industry, Events, Magazines, Jobs, Directory, Projects, and more. Implemented filtering, pagination, and structured layouts. Built responsive UI for all devices. Developed authentication system and user dashboard. Integrated API-driven data rendering. Implemented contact forms and inquiry workflows with email alerts. Connected backend-controlled homepage modules with frontend display.

My Contribution / How I Worked on This Project:

I worked as a Full Stack Developer, handling both Angular frontend and Node.js backend development. I designed the backend API architecture first, ensuring all modules were modular and scalable. Each module was developed as an independent API service with proper validation and data handling.

On the frontend, I built reusable Angular components and implemented clean UI structures with proper routing and state management. I ensured smooth communication between frontend and backend using REST APIs.

As the project evolved, I handled redesign requirements, UI updates, and additional feature planning. I worked closely with stakeholders to convert business requirements into technical workflows and ensured the system remained stable and scalable.

Challenges Faced in the Project:

1. Large Multi-Module Architecture

Handled 25+ modules in a single platform, solved using modular API-based architecture.

2. Frequent Requirement Changes

Managed continuous UI redesign and feature expansion using reusable components.

3. Frontend-Backend Synchronization

Ensured smooth API integration and data consistency.

4. Performance Optimization

Handled large datasets using pagination and optimized queries.

5. Secure Payment & Content Access

Designed logic for secure digital content access and payment validation.

6. UI Redesign Without Breaking System

Handled redesign while keeping backend stable using separation of concerns.

Tools & Services Used:

Angular (TypeScript), Node.js (Express), MySQL, JavaScript, jQuery, Mailchimp, Google Analytics, Stripe, Razorpay, Google Maps, Git, cPanel

Business Impact Delivered:

Centralized content management system. Reduced dependency on developers for updates. Enabled scalable publishing and monetization. Improved user experience with dynamic UI. Built foundation for subscriptions and directory business model.

Project Name: Cetas Insights – Client Account Portal & Charting Workflow Platform

Website: https://spdpcharting.sonisingh.online/ (http://spdpcharting.com/)

Project Type: Client Dashboard, Subscription Portal, Project Workflow Management & File-Based Delivery System

Tech Stack:

Frontend: HTML5, CSS3, JavaScript, jQuery, AJAX
Backend: WordPress, PHP, Custom Theme/Child Theme Development
Database: MySQL
Theme & Plugins: Hello Elementor Child Theme, Elementor, Elementor Pro, Advanced Custom Fields (ACF), ARForms, ARMember / Membership-related workflow, Dynamic Custom Content for Elementor
Integrations: AWS Server Setup, Gmail/SMTP Email Flow, Login Email Verification, Google Maps, Custom Audio Recording Upload
Tools & Services: Git, cPanel, WordPress AJAX, Custom Shortcodes, User Meta / ACF User Fields, Admin Customization

Project Description:

Cetas Insights was a custom client account portal and project workflow system built on WordPress to manage subscriptions, projects, invoices, support tickets, profile settings, and delivery resources for different service segments such as Survey Programming, Data Charting, Data Processing, and Open-End Coding.

The portal was designed as a secure logged-in user platform where clients could access their dashboard, manage subscription plans, review invoices, raise support requests, update profile settings, and track project progress. In addition to the standard account workflow, the project also included custom business logic for subscription purchase flows, invoice generation, project upload enhancements, audio recording uploads, email verification during registration, AWS environment setup, and additional report fields across multiple project segments.

A key part of the project was extending the initial charting project into a broader delivery platform with a more structured account area. This included building multiple frontend dashboard pages, integrating custom user subscription logic, handling invoice records dynamically, and creating segment-specific upload/reporting enhancements.

Role:

Full Stack WordPress Developer / Custom Portal Development Engineer

Key Responsibilities:

- Developed and customized the logged-in client account portal for profile, projects, subscriptions, invoices, support, settings, and upgrade plan flows.
- Built custom frontend account pages using WordPress, Elementor, ACF, ARForms, JavaScript, jQuery, and AJAX.

- Extended the existing charting workflow into four service segments: Survey Programming, Data Charting, Data Processing, and Open-End Coding.
- Added additional fields in project forms and upload report forms for each segment.
- Implemented custom subscription selection logic with dynamic price table rendering using AJAX.
- Built invoice creation workflow on subscription purchase using WordPress custom posts, ACF fields, and user meta updates.
- Developed login email verification flow for newly registered users with custom activation links and restricted login until activation.
- Customized user registration email templates with branded HTML email layout.
- Built first-time login redirection logic for subscription onboarding.
- Created custom shortcodes for reusable dynamic data such as product URLs, plan URLs, file metadata, assigned users, payment status, and email verification handling.
- Integrated user-specific subscription details, usage counts, and plan allocation values through ACF user fields.
- Developed support for audio recording upload in the project upload form using Recorder.js, browser microphone access, and WordPress AJAX upload handler.
- Implemented file upload and resource display features for project detail pages and data delivery sections.
- Set up and supported AWS server-related environment requirements for the platform.
- Customized WordPress admin experience for non-admin users by hiding admin bar elements, screen options, contextual help, and application passwords.
- Added project filtering, status display, subscription upgrade flow, invoice display, and support listing workflows on the frontend.
- Worked on styling and UI consistency for the account dashboard pages such as dashboard overview, profile, projects, subscription, invoices, support, settings, and upgrade subscription screens.

Main Modules Delivered:

1. Account Dashboard

Built the main client dashboard showing account summary, usage counters, subscription overview, and recent projects.

2. Profile Management

Created profile page displaying personal info, company info, address details, and user-related metadata.

3. Projects Module

Developed project listing page with search, date filtering, project type filtering, priority, and status-based display.

4. Subscription Module

Built subscription details page showing active plan, limits, plan features, and renewal workflow.

5. Upgrade Subscription Flow

Created plan comparison and plan purchase/upgrade screens with dynamic pricing and feature rendering.

6. Invoice Management

Developed invoice listing and invoice creation workflow tied to subscription purchases.

7. Support Module

Built support request listing page with filter options and add-support workflow.

8. Settings Module

Created account settings page with editable personal details, company details, address details, and password update tabs.

9. Project Delivery / Resource Section

Built structured file/resource display area for project outputs such as Excel, PDF, PowerPoint, custom files, and audio assets.

10. Audio Recording Upload Feature

Added microphone-based audio capture and upload functionality directly in project submission flow.

11. Registration & Email Verification

Implemented user email verification workflow with activation link and restricted login for inactive accounts.

12. Custom Subscription & Billing Logic

Built dynamic plan selection, pricing updates, invoice generation, and subscription metadata updates using AJAX and ACF.

My Contribution / How I Worked on This Project:

I worked on this project as the main WordPress full stack developer, primarily focused on extending an existing charting-related system into a more complete client account portal. My work started from understanding the business workflow across the four service segments and then converting those operational requirements into a structured logged-in dashboard experience.

I first worked on the backend logic and data structure. Since the portal relied heavily on custom user data, subscription metadata, invoice records, plan allocation counts, and project-linked resources, I used ACF fields, custom post types, user meta, shortcodes, and AJAX handlers to keep the system dynamic and manageable. I also customized the Hello Elementor child theme and wrote custom PHP modules for reusable portal functions.

On the frontend side, I built and connected multiple account pages such as dashboard, projects, subscriptions, invoices, support, settings, and plan upgrades. I used JavaScript, jQuery, AJAX, and form integrations to make the user experience smoother, especially for dynamic plan loading, invoice creation, auto-filled profile fields, and subscription purchase actions.

A major part of the work also included custom workflow enhancements beyond the usual portal pages. For example, I implemented login email verification, custom branded registration email templates, first-time login redirection, audio recording uploads for project forms, and segment-specific field extensions for project/report uploads. I also handled admin-side simplification by restricting certain WordPress backend elements for non-admin users.

Challenges Faced in the Project:

1. Extending an Existing Project Without Breaking Current Workflow

The project was not built entirely from scratch. It involved adding new functionality on top of an existing charting project. This meant I had to carefully extend forms, account logic, and project segments without disrupting the already working parts of the system.

2. Subscription and Invoice Workflow Customization

The billing workflow was more custom than a normal e-commerce setup. Plan selection, invoice generation, subscriber IDs, user-specific unit counts, and account subscription updates all needed to work together. I handled this by creating custom AJAX handlers and ACF-based update logic.

3. Managing User-Based Dynamic Content

A large portion of the portal content changed depending on the logged-in user, their plan, their usage counts, and their project records. I used user meta, ACF user fields, and shortcode-based rendering to make the account experience dynamic.

4. Email Verification and Restricted Login Flow

The registration process required account verification before login. This meant building a complete activation flow with email templates, activation hashes, restricted authentication checks, and helpful failure messaging.

5. Audio Recording in Browser and Upload to WordPress

Adding microphone-based audio recording and upload inside a WordPress portal was a technical challenge because it involved browser media APIs, Recorder.js, temporary blobs, AJAX upload handlers, and storing the final file path back into form fields.

6. Keeping WordPress Admin Clean for Non-Admin Users

The portal had users with different access levels. I customized the admin experience by hiding admin bar elements and backend options for non-admin roles to make the system safer and less confusing.

7. Frontend Dashboard Consistency Across Multiple Pages

The project included many account pages like dashboard, profile, projects, subscriptions, invoices, support, settings, and upgrade plan. Maintaining UI consistency while keeping each module functional required careful structuring of templates and reusable styling.

Tools & Services Used:

- **WordPress + PHP:** For portal development, custom business logic, user authentication flow, AJAX handlers, and backend customization.
- **MySQL:** For storing user data, project records, invoices, subscription details, and content metadata.
- **Elementor / Elementor Pro:** For building and managing frontend account layouts.
- **Advanced Custom Fields (ACF):** For custom user fields, plan details, invoice metadata, project counts, and file/resource mapping.
- **ARForms:** For registration, settings, subscription, and other interactive form workflows.
- **JavaScript / jQuery / AJAX:** For dynamic plan loading, price updates, invoice processing, auto-fill logic, and asynchronous interactions.
- **Recorder.js / Browser Media APIs:** For audio recording and upload functionality.
- **AWS Server Setup:** For infrastructure/environment support tied to the project needs.
- **Gmail/SMTP / Custom HTML Emails:** For registration email verification and account communication.
- **Git / cPanel:** For deployment and server-side project management.

Business Impact Delivered:

- Converted a segment-based charting workflow into a structured self-service client account portal.
- Improved visibility for clients across subscriptions, invoices, projects, and support requests.
- Reduced manual subscription and invoice handling through automated logic.
- Added account verification and onboarding flow for better security and user control.
- Extended project submission capabilities with audio recording and additional segment-wise fields.
- Created a scalable base for future account, billing, and project delivery enhancements.

Project: Service Management Dashboard & Reporting System (WordPress + ACF)

Role:

Full Stack WordPress Developer

Project Description:

Developed a dynamic Service Management Dashboard system using WordPress that enables technicians and administrators to create, manage, and update service reports directly from the frontend without accessing the admin panel. The system supports multiple post types such as write-ups and completed reports, integrates Advanced Custom Fields (ACF) for structured data handling, and provides a real-time dashboard view with role-based access control. It also includes frontend editable forms, file upload functionality, and reusable short code components to enhance usability and scalability.

Business Use / Application Usage:

This application is designed for service-based organizations such as automobile service centers, equipment maintenance companies, and field service teams where technicians need to record work orders, diagnostics, and repair details. It allows non-technical users to log in and manage service reports from a simple frontend interface, reducing dependency on backend/admin panels. Administrators can monitor all service activities in a centralized dashboard, track work orders, verify technician submissions, and ensure proper documentation with file uploads (images, PDFs). The system improves operational efficiency, ensures data consistency, and provides a structured workflow for service reporting and audit purposes.

Roles & Responsibilities:

Designed and developed a fully dynamic frontend-driven dashboard system using WordPress by implementing custom shortcodes and Advanced Custom Fields (ACF) for structured data management. Built reusable shortcode components such as dashboard listing, user welcome message, logout functionality, and dynamic field rendering to create a seamless user experience. Developed a frontend editable form system that allows users to update service reports, including technician details, work orders, customer information, and repair summaries, along with file upload capabilities for images and documents. Implemented role-based access control to ensure secure editing and viewing permissions for administrators, editors, and respective authors. Optimized data retrieval and display using WP_Query for handling multiple post types efficiently. Ensured proper data validation, sanitization, and performance optimization to maintain scalability and security of the application.

System Workflow:

1. User (Technician/Admin) logs into the system
2. Dashboard displays list of service reports (write-up / complete)
3. User can view or edit reports based on role permissions
4. Frontend form allows updating of:
    - Technician details
    - Work order
    - Customer name
    - Condition found
    - Probable cause
    - Correction steps
    - Shop supplies used
5. User uploads supporting files (images/PDF)
6. Data is saved using ACF fields and reflected instantly in dashboard

Key Features:

- Frontend Dashboard (No WP Admin dependency)
- Custom Shortcodes for dynamic UI
- ACF-based structured data system
- Role-based access control
- Editable frontend forms
- File upload (JPG, PNG, PDF)
- Multi post-type support (write-up, complete)
- Dynamic table listing with actions (View/Edit)

Technical Highlights:

- Custom Shortcodes for modular architecture
- WP_Query optimization for performance
- ACF integration for structured and scalable data
- Secure data handling using sanitization and validation
- Role-based conditional rendering for security
- Frontend-first architecture for better usability

Technologies Used:

WordPress, PHP, Advanced Custom Fields (ACF), WP_Query, HTML, CSS, JavaScript

Performance Tracking & Bonus System:

Integrated technician performance tracking within the dashboard to monitor productivity, work quality, and report completion metrics. Implemented logic to evaluate technician performance on a monthly, quarterly (Q1, Q2, Q3, Q4), and yearly basis. Enabled calculation of incentives/bonuses based on predefined KPIs such as number of completed work orders, accuracy of reports, and timely submissions. This feature allows management to generate performance insights, reward top-performing technicians, and maintain accountability across the service team.

Impact:

- Reduced manual backend dependency by 80%
- Improved workflow efficiency for service teams
- Enabled real-time report updates from frontend
- Built scalable and reusable short code-based architecture
- Simplified reporting process for non-technical users

Project Name: The Last Mile NYC – Interactive Knowledge Hub & Workforce Strategy Platform

Website: https://thelastmilenyc.com/

Project Type: Interactive Content Platform, Category-Driven Knowledge Hub, Media-Rich Thought Leadership Website

Tech Stack:

Frontend: HTML5, CSS3, JavaScript, jQuery, Elementor, Custom WordPress Templates
Backend: WordPress, PHP, Custom Theme/Child Theme Development
Database: MySQL
Integrations: Azure API Integration, Dynamic Video API, Social Share Links, Custom Category Meta, WordPress Cron
Tools & Services: Hello Elementor Child Theme, Elementor, WordPress Taxonomy Meta, WP Cron, Custom Shortcodes, Custom Query Hooks, Media/Video Embeds

Project Description:

The Last Mile NYC project was a highly customized interactive content platform built on WordPress for presenting workforce strategy, operational expertise, execution insights, and thought leadership in a visually engaging and non-traditional format. Instead of using a standard blog or article-based layout, the website was designed as a category-driven knowledge hub where users could explore topics like Experience, Expertise, Execution, Employees, Process, Technology, Communication, Scheduling, Accountability, and other operational subjects through interactive visual navigation.

The platform combined structured content architecture with immersive media presentation. It included animated and visually rich category pages, keyword-based navigation, custom word-cloud style exploration, post ordering logic based on category weight and percentage, dynamic video retrieval from external APIs, social sharing links for multiple platforms, and automated synchronization of remote content into WordPress.

A core objective of the project was to transform regular editorial content into a more engaging digital storytelling experience. The site allows users to browse strategic themes visually, drill down into subtopics and keywords, and access supporting content such as videos, external platform links, and categorized insights in a branded, modern interface.

Role:

Full Stack WordPress Developer / Interactive Experience Developer / Custom Content Architecture Engineer

Key Responsibilities:

- Developed the custom WordPress child theme and overall frontend/backend logic for a highly interactive content experience platform.
- Built media-rich category and subcategory navigation flows to replace a standard blog-style browsing experience.
- Created custom taxonomy-driven architecture using WordPress categories as a structured hierarchy for topics, subtopics, and keyword exploration.
- Added custom taxonomy meta fields such as category type, category weight, and category percentage to control visual importance and ordering logic.
- Developed custom sorting logic to prioritize category and post display using taxonomy weight and URL-based hierarchy rules.
- Built custom Elementor query hooks to dynamically control post ordering based on category relationships and category weights.
- Implemented shortcode-based dynamic rendering for archive titles, primary category labels, featured images, share/play blocks, and hierarchical category pages.
- Developed an interactive word-cloud style category exploration section where topics are displayed visually based on assigned weights and percentages.
- Integrated Azure-hosted remote APIs for pulling website content lists and website video links into WordPress.
- Implemented automatic content synchronization using WP Cron to periodically fetch remote data and create/update WordPress posts and categories.
- Built logic to map remote API category structures into recursive WordPress categories with parent-child hierarchy support.
- Created logic to import and update remote content, assign post meta, set featured images, and connect content with proper category trees.
- Built a custom dynamic video shortcode to fetch video URLs from external APIs and render either direct video or Google Drive preview embeds based on content source.
- Added custom post meta boxes for platform-specific social share links such as Apple, Instagram, LinkedIn, YouTube, X, Spotify, Facebook, and Medium.
- Developed a shortcode-driven share and play UI that displays either direct play/share buttons or “coming soon” messaging based on content availability and publication date.
- Added custom frontend logic for rendering archive pages and topic pages with visual hierarchy and interactive engagement.

- Enqueued and managed additional frontend dependencies such as Waypoints and ensured Elementor frontend script compatibility where needed.
- Built a reusable and scalable content model so future topics, categories, keywords, and remote content entries could be added without major structural changes.

Main Features Delivered:

1. Interactive Topic Navigation

Built a category-based exploration system where users can move through high-level topics such as Experience, Expertise, and Execution, then drill down into more specific subjects and keywords.

2. Visual Word Cloud / Topic Cloud Interface

Developed a weighted word cloud interface where categories and keywords appear based on assigned term meta values such as weight and percent. This allowed important topics to visually stand out.

3. Recursive Category Architecture

Used WordPress categories as a structured hierarchy to represent topic trees, including parent topics and children/subtopics, while preserving navigation and content relationships.

4. Custom Post Ordering by Category Weight

Built custom query logic to order posts dynamically based on category weight and URL path hierarchy so the most relevant strategic content could appear first.

5. Remote Content Sync from API

Integrated external Azure APIs to fetch structured website content and automatically create/update WordPress posts and taxonomy structures.

6. Remote Video Fetching System

Created a shortcode-based video retrieval mechanism that dynamically resolves video content from an external API and renders it as embedded media.

7. Social Share Link Management

Added a custom post meta box for editors/admins to store and manage social sharing and media links for different platforms per content item.

8. Dynamic Share / Play UI

Built a smart share/play component that shows active share and play buttons when a media URL exists, and fallback “coming soon” messaging when publication dates are defined but media is not yet available.

9. Featured Media & Branded Visual Presentation

Integrated branded images, interactive visual scenes, and topic-specific graphics to support a rich storytelling experience instead of a plain article listing interface.

10. Automated Featured Image Import

Implemented logic to side-load remote background images, attach them to imported posts, and reuse existing media when available.

My Contribution / How I Worked on This Project:

I worked on this project as a full stack WordPress developer with a strong focus on custom content architecture and interactive frontend experience. The first step was to understand that this was not a normal corporate website or blog. The goal was to create a strategic storytelling platform where content could be explored visually, thematically, and hierarchically. Because of that, I designed the site around category relationships, weighted taxonomy metadata, and dynamic content rendering rather than relying only on standard post lists.

I used WordPress as the content engine, but extended it significantly with custom PHP logic, theme-level enhancements, Elementor-based rendering, shortcode systems, and taxonomy meta controls. I created structures where categories could represent topic groups and subtopics, then used custom query hooks and weighting logic to control how content appears across pages.

A major part of my work also involved integrating the site with remote APIs. I implemented automated content synchronization so structured content and video links from an external source could be brought into WordPress, organized into hierarchical categories, and then displayed through the custom frontend experience. This allowed the platform to stay connected to an external content source while still presenting everything inside a branded website experience.

I also built dynamic media rendering and social share logic so each content entry could support multiple external channels such as LinkedIn, YouTube, Medium, X, Spotify, and others. Overall, my role was to bridge content strategy, visual interaction, and WordPress engineering into one scalable platform.

Business Use / Application Usage:

This application is used as an interactive knowledge and brand storytelling platform for The Last Mile NYC. Instead of presenting information in a static corporate site format, it helps communicate operational expertise, workforce strategy, execution models, organizational challenges, and leadership insights in a more engaging and memorable way. The system is useful for thought leadership publishing, audience education, internal/external communication strategy, and digital brand positioning. It also allows the business to continuously expand topic clusters and publish new strategic insights through a scalable category and media framework.

Challenges Faced in the Project:

1. Converting Standard Content into an Interactive Experience

One of the biggest challenges was moving beyond a traditional article or blog layout. The platform needed to feel exploratory and visual. I solved this by creating custom archive experiences, weighted category displays, and shortcode-based visual modules.

2. Managing Hierarchical Topic Architecture in WordPress

The project required parent-child category structures with meaningful business relationships between topics, subtopics, and keywords. I used custom category metadata and recursive category creation logic to make the structure scalable.

3. Dynamic Content Ordering Based on Business Logic

Content could not simply be ordered by date. It needed to appear based on strategic importance and category relationship. I implemented custom query filters and sorting mechanisms using category weight.

4. Syncing External Content Reliably into WordPress

The project relied on Azure-hosted content and video APIs. I built automated sync logic using WP Cron, remote fetches, recursive category creation, content imports, and media sideloading to keep the site updated.

5. Supporting Multiple External Media and Share Platforms

Different content items required different external links and media sources such as YouTube, Medium, Spotify, Google Drive, and direct video URLs. I solved this by creating flexible post meta fields and a shortcode-based renderer that adapts based on available data.

6. Preserving Brand Experience While Keeping the System Editable

The frontend needed to look custom and immersive, but the backend still needed to remain manageable for ongoing content updates. I handled this by combining Elementor flexibility with custom-coded logic and reusable shortcodes.

Technologies Used:

WordPress, PHP, MySQL, Elementor, JavaScript, jQuery, HTML5, CSS3, WP Cron, WordPress Taxonomy Meta, Custom Shortcodes, Custom Query Hooks, Azure API Integration

Impact:

- Transformed standard WordPress content into an interactive visual strategy platform
- Enabled scalable content publishing through category hierarchy and remote sync
- Improved brand storytelling through media-rich and topic-driven presentation
- Reduced manual content duplication by syncing structured remote content into WordPress
- Created a reusable framework for expanding topics, subtopics, and associated media in the future

Project Name: AstraQuant AI – Market Intelligence, Trading Analytics & Predictive Insights Platform

Project Type: AI-Powered Financial Analytics Platform, Trading Dashboard, Market Intelligence & Prediction Support System

Tech Stack:

Frontend: React.js, TypeScript, Tailwind CSS, HTML5, JavaScript
Backend: Python, Django, Django REST Framework
Database: PostgreSQL / SQL-based structured data storage
APIs & Data Sources: Stock Market APIs, Broker APIs, News Feed APIs, REST APIs, WebSocket-ready live data integrations
AI / Analytics Layer: Predictive analytics logic, technical indicator engine, signal processing workflows, news impact analysis, rule-based + AI-assisted market insights
Tools & Services: Git, Postman, JSON mock APIs, environment-based configs, modular component architecture, responsive dashboard UI

Project Description:

AstraQuant AI is a full-stack financial analytics and market intelligence platform designed to help traders and market participants monitor, analyze, and interpret live and historical market data across multiple asset classes. The platform was planned as a scalable AI-assisted system that combines structured financial data, technical analysis, trading signals, macro/news-based market impact, and dashboard-driven visualization into a single interface.

The system was intended to support indices such as Nifty, Sensex, Bank Nifty, and Gift Nifty, while also being extensible to equities, commodities, currencies, mutual funds, futures, and options. The frontend was designed as a professional React-based dashboard with multiple pages, reusable components, services, and scalable UI architecture. The backend was planned using Python and Django to handle APIs, market data ingestion, processing logic, authentication, and predictive analytics workflows.

A key goal of the project was not just to display market prices, but to create a decision-support platform where technical structure, sentiment, live news, historical behavior, and derived indicators could be brought together to assist with forecasting, screening, and strategy evaluation. The product direction focused on building a clean, modern, and reliable analytics experience rather than a basic stock tracker.

Role:

Full Stack Developer / Product Architect / AI Market Analytics Platform Designer

Frontend Development:

Frontend Scope:

Designed the frontend as a scalable React.js + TypeScript application with a professional dashboard-style architecture. The UI was planned to provide an organized trading workspace where users could access watchlists, market snapshots, technical signals, sector movement, sentiment summaries, news impact, instrument-wise analytics, and prediction support in one place.

Frontend Responsibilities:

- Planned and structured the frontend shell, navigation system, and scalable page architecture.
- Designed dashboard-oriented UI flows for indices, sectors, watchlists, stock details, market overview, prediction boards, and analytics pages.
- Used React.js with TypeScript for maintainable, component-driven development.
- Planned Tailwind CSS-based design system for responsive and modern UI implementation.
- Designed reusable components for cards, widgets, filters, tables, tabs, chart containers, screener panels, and status summaries.
- Structured frontend services for API calls, modular state updates, and future broker/news integration.
- Planned UI sections for daily, weekly, and monthly views of market signals and forecasts.
- Considered separate views for Nifty, Sensex, Gift Nifty, Bank Nifty, commodities, and broader markets.
- Designed the frontend to support both live market data and delayed/mock API workflows during development.
- Planned responsive and scalable code organization for components, layouts, pages, hooks, services, and utility layers.
- Created a product-style UX direction aimed at making financial information easy to consume and compare visually.

Frontend Features Planned / Delivered in Architecture:

- Market overview dashboard
- Index cards for major benchmarks
- Sector and instrument tracking views
- Technical indicator panels
- News impact cards
- Forecast / signal blocks
- Watchlist and screening layouts
- Strategy and sentiment summary widgets
- Responsive dashboard shell for desktop-focused trading workflow

Backend Development:

Backend Scope:

Designed the backend using Python and Django as the core engine for data handling, analytics processing, API orchestration, authentication, and future predictive workflows. The backend was intended to act as the intelligence layer of the platform by aggregating market feeds, running calculations, storing structured financial information, and exposing clean APIs to the React frontend.

Backend Responsibilities:

- Planned backend architecture in Python and Django for scalable market analytics workflows.
- Designed REST API structure for delivering market data, analytics results, and dashboard summaries.
- Planned integration with stock/broker APIs for prices, instruments, and market snapshots.
- Prepared backend direction for live news feed ingestion and news-to-market impact mapping.
- Designed logic flow for combining technical indicators with structured news analysis.
- Planned support for multiple asset classes including indices, stocks, futures, options, commodities, and currency instruments.
- Structured backend modules for authentication, data ingestion, indicator processing, prediction support, and historical records.
- Designed database logic to store symbol data, candles, signals, categories, and derived market insights.
- Planned rule-based and AI-assisted signal generation workflows for forecasting support.
- Considered future support for scheduled jobs, batch analytics, and periodic market summary generation.
- Designed backend to remain extensible for future model experimentation and strategy modules.

Backend Features Planned / Delivered in Architecture:

- Django REST API structure
- Market data ingestion pipeline
- Historical + current data handling
- Technical indicator calculation workflow
- Signal/prediction support endpoints
- News feed ingestion planning
- Broker/data provider integration planning
- Authentication and user-based dashboard access
- Database-ready structure for multi-instrument analytics

Business Use / Application Usage:

This application is intended for traders, analysts, market learners, and financial product teams who want a single platform for market monitoring and analysis. Instead of checking separate tools for charts, indicators, news, and market sentiment, users can access organized financial intelligence from one dashboard. The system is useful for understanding market direction, screening instruments, tracking live developments, comparing assets, and supporting trading or investment decisions with structured data. It also provides a strong foundation for future subscription-based analytics or broker-integrated trading products.

Key Responsibilities:

- Defined the full-stack product architecture for an AI-assisted trading analytics platform.
- Planned both frontend and backend modules to support scalable market intelligence workflows.

- Designed React + TypeScript frontend structure for professional dashboard UX.
- Designed Django backend APIs and analytics architecture for market data processing.
- Structured the application to support multiple markets, timeframes, and asset categories.
- Planned integration points for broker APIs, market data providers, and news feeds.
- Focused on modular development so the platform could grow into a larger financial intelligence system.
- Designed the product around real user workflows such as monitoring, screening, comparing, forecasting, and reviewing signals.

My Contribution / How I Worked on This Project:

I worked on AstraQuant AI as both a product architect and full stack developer, with a focus on turning a broad vision for a market intelligence platform into a structured technical plan. I first broke the product into logical layers: frontend dashboard experience, backend analytics engine, market data ingestion, signal generation, and future AI/prediction support. This helped me separate display concerns from data processing concerns and made the platform easier to plan in a scalable way.

On the frontend side, I focused on designing a professional trading dashboard experience with reusable components and page-level modularity. On the backend side, I structured how Django APIs and market analytics workflows would interact so the system could support different financial instruments and future expansion. I also kept in mind that live market systems require careful architecture, so I approached the product as a long-term, extensible platform rather than a one-page app.

Challenges Faced in the Project:

1. Broad Product Scope Across Multiple Market Types

The platform was intended to handle indices, equities, commodities, futures, options, and currencies. This created a challenge in designing the architecture so it could remain organized and extensible. I solved this by planning modular frontend and backend layers.

2. Combining Technical Analysis with News and Sentiment

A major challenge was designing a system that could bring together technical indicators, price action, and external news impact. I handled this by separating the ingestion layer, processing layer, and presentation layer.

3. Live Data vs Development Workflow

Financial products often need live data, but development is easier with mock or staged APIs. I planned the platform to support both mock development flows and future live API integration.

4. Building for Scale from the Beginning

Even at planning stage, the platform needed to support future features like strategy modules, alerts, deeper analytics, and expanded asset support. I addressed this by keeping both frontend and backend architecture modular.

5. User Experience for Dense Financial Information

Market data can become visually overwhelming very quickly. I focused on a dashboard-based layout strategy that groups information into manageable, decision-friendly components.

Technologies Used:

React.js, TypeScript, Tailwind CSS, JavaScript, Python, Django, Django REST Framework, PostgreSQL, REST APIs, JSON mock data, market data integrations, news feed integration planning, Git

Impact:

- Created the full-stack foundation for a professional AI-assisted trading analytics platform
- Designed a scalable frontend and backend architecture for future market intelligence expansion
- Structured the product to support multiple asset classes and timeframes
- Built a strong base for integrating data, signals, dashboards, and prediction workflows into one system

KEY STRENGTHS

- Full Stack Development (WordPress, React, Angular, Django)
- AI & OCR Automation (AWS Textract, GPT-based systems)
- Scalable Architecture & API Design
- Performance Optimization & SEO

- Complex Workflow Automation & Dashboard Systems

NOTE FOR HR

All projects listed above are built end-to-end by me, including architecture, development, integration, optimization, and deployment. These projects demonstrate my ability to handle real-world, large-scale, and high-complexity systems independently.