Miha Najvirt Horvat

Web Developer Portfolio

Professional Development Portfolio

Projects spanning web, mobile, and desktop applications

(Personal projects not included in this portfolio)

Stack:	JS/TS ecosystem (React, Next.js, NestJS, React Native, Electron, Remix, WXT),
Platforms:	Web, mobile, desktop, browser extensions
Infrastructure:	Docker, Coolify, Apache, SSL, deployment pipelines
Specialization:	Full-stack development, cross-platform & DevOps

Email:	mihanajvirthorvat@gmail.com
GitHub:	github.com/miha-yy
Location:	Nebova 24, 2229, Malečnik, Slovenia

Contents

1 Executive Summary

This portfolio presents a collection of major professional projects that showcase my expertise as a versatile full-stack developer. These projects represent my professional work across web, mobile, and desktop applications. My work demonstrates proficiency in modern JavaScript ecosystems, enterprise-level integrations, and cutting-edge AI implementations.

1.1 Core Technical Expertise

- **Full-Stack Development:** Extensive experience with React, Next.js 15, NestJS, and TypeScript across multiple production applications
- **Mobile Development:** Developed React Native applications with Expo for iOS, Android, and Web from a single codebase using monorepo architecture
- **Desktop Applications:** Built cross-platform desktop apps using Electron.js for both Windows and Mac environments
- **Browser Extensions:** Created production-ready Chrome extensions with WXT framework and InboxSDK integration
- **ERP Integrations:** Implemented ODBC connections for enterprise-level ERP systems with real-time synchronization
- **Backend as a Service:** Worked extensively with Supabase including Edge Functions, real-time subscriptions, Row Level Security, and storage
- **AI-Powered Solutions:** Integrated multiple AI providers (OpenAI, Anthropic, Google AI, Azure AI) with Vercel AI SDK for intelligent automation
- **Database Management:** Worked extensively with PostgreSQL using both TypeORM, Prisma ORM, and Supabase
- **Payment Integrations:** Successfully integrated Stripe, PayPal, and Lemon Squeezy across multiple e-commerce projects
- **Financial Tech:** Built investment platforms with KYC verification (Sumsub), wallet systems, and transaction management
- **Infrastructure:** Configured Docker, Apache, SSL certificates, and production deployment pipelines

1.2 Notable Achievements

- Architected and developed a complete monorepo structure for cross-platform applications (Bo)
- Built sophisticated email automation systems with AI-powered categorization and response generation (InboxZero)
- Created a Gmail browser extension integrating AI assistant directly into email workflows (InboxZero Extension)

- Implemented complex ERP integrations with ODBC protocol for real-time data synchronization (Maconomy)
- Developed multi-tenant e-commerce platforms supporting multiple countries and payment systems (Megabon)
- Built comprehensive CMS systems with AWS S3 integration and multilingual content management (ORS CMS)
- Designed XML processing systems for financial data extraction and management (Extra XML Parser)
- Developed AI-powered invoice processing system using Azure Document Intelligence (Mikrocop Expenses AI)
- Built complete investment mobile application with bonds trading, KYC verification, and portfolio tracking (Equito)
- Implemented full particle treatment plan report (PDF) for an FDA-approved radiation therapy planning system, including 20+ report sections and export pipeline (CTPS-GUI Particle Plan Report)
- Built UCB IOF 2026 congress microsite from start to finish for Madano/UCB, including event pages, calendar integration, Google Maps, survey, and pharma-compliant exit ramp (UCB IOF 2026 Microsite)

1.3 Professional Highlights

My projects demonstrate expertise in:

- Working independently and leading development of critical features across all project types
- Integrating complex external systems (ERPs, payment gateways, AI providers, email APIs)
- Optimizing performance for high-load applications with caching strategies and queue systems
- Ensuring code quality with TypeScript, comprehensive testing, and clean architecture patterns
- Adapting quickly to new technologies and frameworks in production environments

2 Education and Languages

2.1 Education

- **Bachelor's Degree in Computer Science**
  Faculty of Electrical Engineering and Computer Science (FERI)
  University of Maribor, Slovenia

2.2 Languages

- **Slovene:** Native
- **English:** Fluent
- **German:** Fluent

3 Work Experience

3.1 CLOUD9DEVS d.o.o.

Position: Lead/Senior Developer
Duration: May 2022 - Present (since company inception)
Location: Rimska ploščad 6, 2250 Ptuj, Slovenia
Website: `https://cloud9.studio/`

CLOUD9DEVS d.o.o. is a software development company (computer science and informatics) based in Ptuj, Slovenia. I have been freelancing for CLOUD9DEVS since their foundation (company entry: 30. 5. 2022) and continue to collaborate on projects in the same technology stack and quality standards as for EXTRA DEV—full-stack web, mobile, and desktop development, integrations, and deployment.

3.1.1 Key Responsibilities

- **Full-Stack Development:** Delivering end-to-end solutions for CLOUD9DEVS clients across web, mobile, and desktop
- **Architecture and Design:** Contributing to system design and technical decisions on freelance projects
- **Client Delivery:** Direct delivery for CLOUD9DEVS clients with same quality standards as in employed roles
- **Technology Stack:** React, Next.js, NestJS, TypeScript, Supabase, Docker, and related tooling
- **Integration and Deployment:** Backend integrations, APIs, and deployment pipelines as required by projects

3.1.2 Projects and Scope

- Freelance development and delivery for CLOUD9DEVS clients across multiple domains
- Full-stack web, mobile, and desktop applications; same stack and practices as EXTRA DEV projects
- Long-term collaboration since the company's inception (May 2022)

3.1.3 Professional Growth

- **Long-Term Collaboration:** Continuous engagement since CLOUD9DEVS foundation
- **Consistent Standards:** Same code quality, architecture, and delivery standards across all engagements
- **Cross-Company Experience:** Applying learnings from EXTRA DEV and own practice to CLOUD9DEVS work

3.2 EXTRA DEV d.o.o.

Position: Lead/Senior Developer
Duration: August 2023 - Present
Location: Trg svobode 14A, 2390 Ravne na Koroškem, Slovenia
Website: `https://extra.dev`

EXTRA DEV d.o.o. is a software development company specializing in computer programming and development of information solutions. I have been with the company since its inception (August 2023), taking on increasing responsibilities and leadership duties.

3.2.1 Key Responsibilities

- **Project Leadership:** Leading development of multiple simultaneous projects across web, mobile, and desktop platforms
- **Team Management:** Managing and mentoring development team members, coordinating work across projects
- **Full-Stack Development:** Developing end-to-end solutions from frontend to backend across various technology stacks
- **Architecture and Design:** Designing scalable system architectures and making critical technical decisions
- **Client Communication:** Direct communication with clients to understand requirements and present technical solutions
- **Code Quality:** Ensuring high code quality standards through code reviews and best practices implementation
- **Technology Evaluation:** Researching and evaluating new technologies and frameworks for project requirements

3.2.2 Projects and Scope

- Lead development on multiple projects across web, mobile, desktop, and browser extensions
- Full-stack delivery for EXTRA DEV clients; projects showcased in this portfolio
- Collaboration since company inception (August 2023)

3.2.3 Professional Growth

Since joining at inception, my contributions demonstrate:

- **Rapid Skill Development:** Quickly mastering new technologies and frameworks required for diverse projects
- **Leadership Abilities:** Successfully transitioning from individual contributor to team leader role
- **Project Management:** Managing multiple complex projects simultaneously with different technology stacks
- **Client Relations:** Building strong relationships with clients across various industries
- **Technical Excellence:** Delivering high-quality, production-ready applications across all platforms

3.3 Miha Najvirt Horvat, s.p.

Position: Freelance Developer / Sole Proprietor
Duration: December 2023 - Present
Location: Nebova 24, 2229 Malečnik, Slovenia
Contact: mihanajvirthorvat@gmail.com, +386 31 893 142

Miha Najvirt Horvat, s.p. is my own company (sole proprietorship) for freelance software development and IT consulting. The company is registered in Slovenia (SKIS S.14100 - Samozaposleni delodajalci, matična 9544275000) and focuses on computer science and informatics. Through it I deliver full-stack development, architecture, and technical consulting for clients alongside my work at EXTRA DEV and CLOUD9DEVS.

3.3.1 Key Responsibilities

- **Full-Stack Development:** Delivering end-to-end solutions under own brand for direct clients
- **Architecture and Consulting:** Technical leadership, system design, and consulting on a contract basis
- **Client Communication:** Direct client contact, requirements, and delivery for freelance engagements
- **Technology Stack:** Same stack as in employed roles (React, Next.js, NestJS, TypeScript, Supabase, Docker, etc.)
- **Code Quality:** Same delivery standards and practices as at EXTRA DEV and CLOUD9DEVS

3.3.2 Projects and Scope

- Freelance development and IT consulting under Miha Najvirt Horvat, s.p.
- Client projects, consulting, and technical leadership on a contract basis
- Operation alongside ongoing work at EXTRA DEV and CLOUD9DEVS (since December 2023)

3.3.3 Professional Growth

- **Own Brand:** Building and maintaining freelance practice under registered company
- **Multi-Engagement Balance:** Delivering for own clients while contributing to EXTRA DEV and CLOUD9DEVS
- **Consistent Standards:** Same quality and technical standards across all three engagements

4 Acquired Knowledge

4.1 Technical Skills

- Development of desktop applications with Electron.js and Capacitor
- Development of browser extensions with WXT framework
- Work with Chrome Extension API and Manifest V3
- Integration with Gmail via InboxSDK
- Development of PHP applications with PDO for data management
- Work with XML structures and SimpleXML for parsing bank statements
- Development with modern Remix framework
- Development of full-stack applications with Next.js 15 (App Router)
- Development of mobile applications with React Native and Expo
- File-based routing with Expo Router for React Native applications
- Mobile navigation with React Navigation (Tab and Stack navigators)
- Advanced use of TypeScript and Prisma ORM
- Development of backend applications with NestJS and TypeORM
- Type-safe API development with tRPC
- Database management with Drizzle ORM
- API documentation with Swagger/OpenAPI

- Integration of external APIs and managing complex data structures
- ERP integrations via ODBC protocol
- Backend as a Service integration with Supabase (Auth, Storage, Edge Functions, Real-time)
- AI integrations with multiple providers (OpenAI, Anthropic, Google AI, Bedrock, Groq, Azure Document Intelligence)
- Development of AI-powered applications with Vercel AI SDK
- Work with queue systems for processing larger amounts of data (p-queue)
- Browser automation with Playwright
- Web framework development with Fastify for standalone services
- Monorepo architecture management with Lerna for shared packages
- Implementation of multilingual support in web applications
- Work with content management systems (Builder.io, Sanity.io)
- UI component libraries (Ant Design, Radix UI, shadcn/ui)
- Client-side routing with React Router
- List virtualization with React Window for performance optimization
- Data visualization with Chart.js and React Native Skia
- Performance optimization of web applications
- Modern CSS practices with Tailwind
- Work with Redis/Upstash for cache and state management
- State management libraries (Jotai for global state, SWR for data fetching)
- Authentication libraries (NextAuth.js, JWT with social logins)
- Cloud storage integration with AWS S3
- Analytics platforms (PostHog, Tinybird, Vercel Analytics, Google Analytics, Facebook Pixel, TikTok Pixel)
- Email services (Resend, React Email, Mailjet, Mailchimp)
- Subscription management systems (RevenueCat, Lemon Squeezy)
- Video conferencing integration with Dyte SDK
- Structured logging with Winston
- Systems for automated data synchronization with cron jobs

- Work with PostgreSQL database
- Docker and Docker Compose for containerization of applications
- Apache configuration for production deployment
- SSL certificate management with Let's Encrypt
- Secure secret management with Azure Key Vault
- Integration with Google Gmail API and Microsoft Graph API
- Real-time webhook systems for event processing
- Azure AI Document Intelligence integration for invoice processing
- KYC verification integration with Sumsub
- Row Level Security (RLS) implementation for database security
- Push notification systems with Expo
- Digital signature systems for document signing
- Investment and portfolio management systems

4.2 Professional Skills

- **Agile Development Environment:**
  - Active participation in Scrum ceremonies (daily standup, sprint planning, retrospective)
  - Effective management and estimation of user stories
  - Adapting to changing requirements and priorities during development
  - Participation in sprint planning and scope determination
- **Communication and Collaboration:**
  - Effective communication with clients from various industries (tourism, e-commerce, IT)
  - Presenting technical solutions to non-technical audiences
  - Active participation in progress review meetings
  - Mentoring and helping other team members
- **Documentation and Knowledge Transfer:**
  - Writing technical documentation for different user levels
  - Maintaining API and integration documentation
  - Preparing user manuals and instructions
  - Documenting architectural decisions and technical solutions

- **Code Management:**
  - Advanced use of Git for version control
  - Execution and review of pull requests
  - Branch management and conflict resolution
  - Maintaining clean and readable code
- **Problem Solving:**
  - Analytical approach to solving complex technical challenges
  - Debugging and performance optimization
  - Proactively identifying potential issues
  - Researching and implementing best practices
- **Project Management:**
  - Setting priorities and managing deadlines
  - Coordination between different project stakeholders
  - Progress monitoring and status reporting
  - Resource and time management
- **Business Understanding:**
  - Understanding business processes in various industries
  - Identifying business needs and converting them to technical requirements
  - Optimizing solutions based on business goals
  - Understanding impacts of technical decisions on business
- **Code Quality:**
  - Implementation and maintenance of tests
  - Ensuring compliance with coding standards
  - Conducting code reviews
  - Continuous integration and delivery (CI/CD)

5 ORS CMS

ORS CMS is an advanced content management system, specifically designed for the tourism industry. The system enables management of hotel offers, vacation packages, and travel through integration with the ORS API. The application is built using the latest web technologies and follows best practices in web application development. The special feature of the system is its multilingual support and adaptability to different tourism providers.

5.1 Key Features of ORS CMS

- **Content Management:** Enables editing and publishing of tourist offers, blogs, and news
- **Integration with ORS API:** Automatic retrieval and updating of data about hotels and vacation packages
- **Multi-language Support:** Full support for multiple languages (SI, HR, EN, DE, RS, SK)
- **Newsletter Management System:** Tool for customer communication and sending promotional messages
- **Advanced Search:** Powerful search for hotels and vacation packages with various filters
- **Reservation Management:** System for tracking and managing reservations

5.2 My Implementations in the ORS CMS Project

- **CMS Feature Development:**
  - Implementation of an advanced content editor with rich text editor
  - Development of a system for managing multilingual content and translations
  - Implementation of a system for managing media files with AWS S3
  - Development of an advanced system for editing metadata
- **Integrations:**
  - Development of robust integration with ORS API for data synchronization
  - Implementation of a system for sending email notifications with Mailjet
  - Integration with analytics system and user tracking
  - Development of API endpoints for external integrations
- **Optimizations:**
  - Implementation of advanced caching for faster operation
  - Query and data structure optimization
  - Development of a system for batch processing of large amounts of data
  - Performance improvements when working with large datasets

5.3 Technical Implementation of ORS CMS

5.3.1 Frontend Technology Stack

- **Frontend Framework:** React with TypeScript and Remix
- **API Integration:** Axios for communication with ORS API

- **Form Management:** React Hook Form with Zod validation
- **Multi-language Support:** i18n system for multiple language support
- **File Management:** AWS S3 for storing images and documents
- **Email Services:** Mailjet and React Email for customer communication

5.3.2 Backend Technology Stack

- **Database:** Prisma ORM with support for different databases

5.3.3 Architectural Features

The application follows modern architecture with clear separation of responsibilities:

- **Multi-tenant Architecture:** Support for multiple different tourism agencies
- **Modularity:** Separate modules for different functionalities (blogs, newsletters, reservations)
- **Security:** Advanced systems for authentication and authorization
- **API Integration:** Robust integration with ORS API for offer data
- **Type Safety:** Extensive use of TypeScript for better code reliability
- **Performance Optimization:** Efficient caching and data management

5.4 My Contribution to the Project

In the ORS CMS project, I focused on the following areas:

- Implementation of an advanced content editor with rich text editor
- Development of a system for managing multilingual content and translations
- Implementation of a system for managing media files with AWS S3
- Development of an advanced system for editing metadata
- Development of robust integration with ORS API for data synchronization
- Implementation of a system for sending email notifications with Mailjet
- Integration with analytics system and user tracking
- Development of API endpoints for external integrations
- Implementation of advanced caching for faster operation
- Query and data structure optimization
- Development of a system for batch processing of large amounts of data
- Performance improvements when working with large datasets

6 Megabon

Megabon is a modern web platform for group buying and offers, connecting merchants with potential customers. The project is built using the latest web technologies and follows best practices in web application development. The special feature of the platform is its multilingual support and support for multiple different stores in the region (Slovenia, Croatia, Bosnia and Herzegovina, Serbia, Montenegro).

6.1 Key Features of Megabon

- **Offer Management:** Comprehensive system for publishing and managing merchant offers
- **Advanced Purchase System:** Integration with various payment systems (Paypal, Valu)
- **Multi-language Support:** Full support for multiple languages and localization (SI, HR, BA, RS, ME, EN)
- **Coupon System:** Advanced management of coupons and special offers
- **Partner Integrations:** Connection with various external systems (CashbackWorld, Dinopark, Online Media)
- **Analytics and Tracking:** Integration with Google Analytics, Facebook Pixel, TikTok Pixel
- **Email Marketing:** Advanced systems for customer communication (Klaviyo, Campaign Monitor)

6.2 My Implementations in the Megabon Project

- **E-commerce Functionality:**
  - Development of an advanced system for managing offers and coupons
  - Implementation of a complex order processing system
  - Development of a system for managing discounts and promotions
  - Implementation of an advanced search with multiple filters and sorting
- **Payment Systems:**
  - Integration of various payment systems (Paypal, Valu, Nuvei)
  - Implementation of secure payment processing
  - Development of a system for managing transactions and refunds
  - Implementation of payment analytics and reports
- **Marketing and Analytics:**
  - Integration with Google Analytics and Facebook Pixel
  - Development of a system for A/B testing
  - Implementation of an advanced conversion tracking system
  - Development of a reporting system for marketing

6.3 Technical Implementation of Megabon

6.3.1 Frontend Technology Stack

- **Frontend Framework:** Next.js with TypeScript
- **API Architecture:** tRPC for type-safe API calls
- **Authentication:** NextAuth.js with support for multiple providers
- **Form Management:** React Hook Form with Zod validation
- **File Management:** AWS S3 for storing media
- **Error Tracking:** Sentry for tracking errors in production
- **CMS Integration:** Builder.io for content management

6.3.2 Backend Technology Stack

- **Database:** Drizzle ORM with PostgreSQL

6.3.3 Architectural Features

The application follows modern architecture with clear separation of responsibilities:

- **Multi-tenant Architecture:** Support for multiple different stores and languages
- **Modularity:** Clear separation between different parts of the application (offers, coupons, orders)
- **Security:** Advanced systems for fraud prevention and user protection
- **API Integrations:** Robust integrations with various external systems
- **Performance Optimization:** Efficient caching and data management
- **Scalability:** Architecture designed for high load and growth

6.4 My Contribution to the Project

While working on the Megabon project, I focused on several key areas:

- Implementation of new features for managing offers and coupons
- Integration with various payment systems and external partners
- Optimization of performance and user experience
- Development of multilingual support and localization
- Implementation of analytics and tracking systems
- Security and user data protection improvements

7 Tweeds

Tweeds is an advanced web platform for managing and analyzing plant data and their sales. The project is designed as a modern administrative dashboard that enables comprehensive overview of plant sales, order management, and sales performance tracking. The platform is developed using the latest web technologies and follows best practices in web application development.

7.1 Key Features of Tweeds

- **Plant Management:** Comprehensive system for tracking and managing plant data
- **Analytics Dashboard:** Advanced graphical displays and sales statistics
- **Order Management:** System for tracking and processing plant orders
- **Multi-language Support:** Support for multiple languages and localization
- **Customer Management:** System for managing customer data and their orders
- **Reporting System:** Advanced tools for generating reports and analyses
- **Data Export:** Ability to export data to various formats (Excel, CSV)

7.2 My Implementations in the Tweeds Project

- **Data Management:**
  - Development of a system for managing complex data structures
  - Implementation of an advanced system for importing and exporting data
  - Development of a system for data validation and cleaning
  - Implementation of a system for data versioning
- **Analytics Functionality:**
  - Development of interactive dashboards with Chart.js
  - Implementation of a system for generating reports
  - Development of advanced filters and data aggregations
  - Implementation of real-time analytics
- **User Interface:**
  - Development of advanced data tables with multiple functionalities
  - Implementation of a system for bulk data editing
  - Development of advanced forms for data entry
  - Implementation of a system for previews and preview functionality

7.3 Technical Implementation of Tweeds

7.3.1 Frontend Technology Stack

- **Frontend Framework:** React with TypeScript
- **UI Components:** Ant Design and custom components
- **State Management:** React Query
- **Graphical Displays:** Chart.js for data visualization
- **Multi-language Support:** i18next for multiple language support
- **API Integrations:** Axios for communication with backend system
- **Authentication:** JWT with secure session management

7.3.2 Backend Technology Stack

- **Backend API:** RESTful API for data management

7.3.3 Architectural Features

The application follows modern architecture with the following characteristics:

- **Modular Design:** Clear separation between different functional modules
- **Atomic Design Components:** Hierarchical organization of components
- **Type Safety:** Strict use of TypeScript for better code reliability
- **Performance Optimization:** Efficient caching and data management
- **Adaptive Design:** Responsive user interface for different screen sizes
- **Centralized State Management:** Efficient management of application data

7.4 My Contribution to the Project

In the Tweeds project, I focused on the following areas:

- Development of new components for the analytics dashboard
- Implementation of a system for filtering and searching plants
- Performance optimization when displaying large amounts of data
- Development of data export functionality
- User experience and interface improvements
- Integration with backend system and data processing

8 Equito

Equito is a comprehensive investment mobile application for bonds and fixed-income securities, designed to democratize access to structured investment products. The application enables users to invest in bonds, track their portfolio performance, interact with financial advisors, and manage their investments through an intuitive mobile interface. Built with React Native and Expo, Equito provides a seamless cross-platform experience for iOS, Android, and Web. The special feature of the application is its complete investment lifecycle management, including KYC verification, digital contract signing, portfolio tracking, and real-time transaction processing through Supabase Edge Functions.

8.1 Key Features of Equito

- **Bond Investment Platform:** Browse and invest in fixed-income securities with detailed investment information
- **Digital Wallet Management:** Deposit and withdraw funds with transaction history and balance tracking
- **Portfolio Tracking:** Real-time portfolio valuation with interactive graphs showing investment performance over time
- **Advisor Matching:** Connect with financial advisors through personalized matching and direct communication
- **KYC Verification:** Integrated Sumsub verification system for compliance and identity verification
- **Digital Contract Signing:** Electronic signature system for investment agreements with dynamic content replacement
- **Comprehensive Onboarding:** Multi-step questionnaire to assess investment knowledge, risk tolerance, and financial goals
- **Transaction Management:** Track all investment transactions, payouts, and wallet operations with detailed history
- **Portfolio Analytics:** Historical portfolio value tracking with percentage change calculations
- **Multi-language Support:** Full internationalization with i18next
- **Push Notifications:** Real-time notifications for important investment events

8.2 My Implementations in the Equito Project

- **Investment System Development:**
  - Implementation of complete investment workflow from bond browsing to contract signing
  - Development of investment transaction system with Supabase database integration

  - Implementation of portfolio value calculation system with real-time updates
  - Development of investment status tracking (pending, paid, settled, withdrawn)

- **Wallet and Transaction Management:**
  - Development of digital wallet system with balance tracking
  - Implementation of deposit and withdrawal workflows
  - Development of transaction filtering and sorting system
  - Implementation of wallet-to-wallet transfer functionality

- **Backend Integrations:**
  - Development of Supabase Edge Functions for server-side operations
  - Implementation of Portfolio Snapshot system for historical tracking
  - Development of Sumsub KYC integration via Edge Functions
  - Implementation of push notification system with Expo Notifications
  - Development of email notification system via Supabase Functions

- **Authentication and Security:**
  - Implementation of Supabase authentication with PKCE flow
  - Development of role-based access control (RLS policies)
  - Implementation of secure session management
  - Development of deep linking for authentication callbacks

- **UI/UX Development:**
  - Development of custom design system with reusable components
  - Implementation of responsive layouts for different screen sizes
  - Development of interactive graphs using Skia for performance
  - Implementation of dark/light theme system
  - Development of onboarding flow with multi-step questionnaire

- **Data Management:**
  - Implementation of React Query for efficient data fetching and caching
  - Development of transaction entity system for complex relationships
  - Implementation of soft delete patterns for data integrity
  - Development of database migration system from MySQL to PostgreSQL

- **Monitoring and Analytics:**
  - Integration with Sentry for error tracking and session replay
  - Implementation of PostHog analytics for user behavior tracking
  - Development of logging system for debugging and monitoring
  - Implementation of error boundaries and error handling throughout the app

8.3 Technical Implementation of Equito

8.3.1 Frontend Technology Stack

- **Mobile Framework:** React Native with Expo
- **Language:** TypeScript
- **Navigation:** Expo Router (file-based routing)
- **State Management:** TanStack React Query
- **Forms:** React Hook Form with Zod validation
- **Authentication:** Supabase Auth with PKCE flow
- **Database:** Supabase (PostgreSQL)
- **Styling:** StyleSheet with custom theme system
- **Internationalization:** i18next with React i18next
- **Error Tracking:** Sentry React Native
- **Analytics:** PostHog React Native
- **Push Notifications:** Expo Notifications
- **KYC Integration:** Sumsub SDK
- **Charts:** React Native Skia for performance

8.3.2 Backend Technology Stack

- **Database:** PostgreSQL with Supabase
- **Edge Functions:** Deno runtime for serverless functions
- **Storage:** Supabase Storage for document management
- **Authentication:** Supabase Auth with JWT
- **KYC Service:** Sumsub API integration
- **Email Service:** Email integration via Edge Functions
- **Real-time:** Supabase real-time subscriptions

8.3.3 Architectural Features

The application follows modern mobile architecture with the following characteristics:

- **Cross-platform:** Single codebase for iOS, Android, and Web
- **File-based Routing:** Expo Router for intuitive navigation structure
- **Type Safety:** Comprehensive TypeScript types generated from database
- **Modular Design:** Clear separation between public, private, and onboarding flows
- **RLS Security:** Row Level Security policies for data access control
- **Serverless Backend:** Supabase Edge Functions for backend operations
- **Real-time Updates:** Supabase real-time for live data synchronization
- **Soft Deletes:** Data preservation with delete flags
- **Performance:** Optimized Skia charts and efficient data caching

8.4 My Contribution to the Project

In the Equito project, I focused on the following areas:

- Complete development of the investment and portfolio tracking system
- Implementation of wallet and transaction management with comprehensive filtering
- Development of Supabase Edge Functions for server-side operations
- Integration of KYC verification system with Sumsub
- Implementation of digital contract signing with signature capture
- Development of advisor matching and communication features
- Creation of comprehensive onboarding flow with multi-step questionnaire
- Database migration from MySQL to PostgreSQL with schema transformation
- Implementation of monitoring and analytics with Sentry and PostHog
- Development of push notification system for real-time user engagement
- Performance optimization with efficient data fetching and caching strategies
- User experience improvements with modern UI components and interactions

9 Mikrocop Expenses AI

Mikrocop Expenses AI is an intelligent web application for automated invoice analysis and expense management. The application uses Azure Document Intelligence to extract structured data from invoices (PDF, JPG, PNG), enabling businesses to automatically process and analyze financial documents. The project leverages cutting-edge AI technology to convert unstructured invoice data into organized, searchable information with confidence scoring for each extracted field.

9.1 Key Features of Mikrocop Expenses AI

- **Intelligent Invoice Processing:** Automated extraction of structured data from PDF, JPG, and PNG invoice files using Azure Document Intelligence
- **Comprehensive Data Extraction:** Extraction of vendor and customer information, invoice numbers, dates, financial data, and line items
- **Confidence Scoring:** Display of confidence scores for each extracted field to ensure data accuracy
- **Advanced Field Support:** Support for vendor/customer details, addresses, payment terms, IBAN/SWIFT codes, tax information, and purchase orders
- **Line Item Analysis:** Detailed extraction and visualization of invoice line items with product codes, descriptions, quantities, and pricing
- **Multi-currency Support:** Recognition and display of currency codes for international invoices
- **Service Period Tracking:** Extraction of service start and end dates for subscription-based invoices
- **Dark Mode Support:** Modern UI with dark mode for comfortable document viewing
- **Responsive Design:** Adaptive interface for various screen sizes and devices

9.2 My Implementations in the Mikrocop Expenses AI Project

- **AI Integration:**
  - Implementation of Azure Document Intelligence integration using the prebuilt-invoice model
  - Development of asynchronous document processing with long-running pollers
  - Implementation of error handling and timeout management for API calls
  - Development of data transformation system from Azure response to structured TypeScript types
- **Data Extraction and Processing:**
  - Implementation of comprehensive field extraction (vendor/customer data, financial data, line items)
  - Development of complex address parsing and formatting system
  - Implementation of payment detail extraction with IBAN and SWIFT code support
  - Development of currency code detection and financial formatting
  - Implementation of confidence scoring display for all extracted fields
- **User Interface Development:**

  - Development of modern, responsive UI with Tailwind CSS
  - Implementation of file upload system with drag-and-drop support
  - Development of dynamic data display cards with conditional rendering
  - Implementation of detailed invoice line items table with proper formatting
  - Development of loading states and error handling UI
- **Backend Architecture:**
  - Development of Remix action handlers for server-side processing
  - Implementation of secure file upload and buffer handling
  - Development of comprehensive TypeScript interfaces for type-safe data
  - Implementation of server-side validation and error handling
- **Data Display and UX:**
  - Development of conditional rendering based on data availability
  - Implementation of formatted display for financial values with currency codes
  - Development of address formatting with proper label handling
  - Implementation of confidence score indicators for data quality assessment

9.3 Technical Implementation of Mikrocop Expenses AI

9.3.1 Frontend Technology Stack

- **Web Framework:** Remix with React and TypeScript
- **Styling:** Tailwind CSS with responsive design and dark mode support
- **Form Handling:** HTML5 file input with multipart form data
- **UI Components:** Custom React components with reusable InfoCard system
- **Build System:** Vite for fast development and optimized production builds

9.3.2 Backend Technology Stack

- **AI Service:** Azure Document Intelligence (prebuilt-invoice model)
- **Type Safety:** Comprehensive TypeScript interfaces for API responses
- **API Client:** Azure REST API client with long-running poller support

9.3.3 Architectural Features

The application follows modern web architecture with the following characteristics:

- **Server-Side Processing:** All AI processing happens on the server via Remix actions
- **Type-Safe Development:** Extensive TypeScript interfaces for all data structures
- **Modular Components:** Reusable InfoCard and InfoField components for structured data display
- **Asynchronous Processing:** Long-running API calls with polling for document analysis
- **Error Handling:** Comprehensive error handling with user-friendly error messages
- **Data Validation:** Server-side validation of uploaded files and processing results
- **Responsive Design:** Mobile-first approach with adaptive layout for all screen sizes
- **Dark Mode:** Modern UI with dark mode support for document viewing

9.4 My Contribution to the Project

In the Mikrocop Expenses AI project, I focused on the following areas:

- Complete development of the AI-powered invoice processing system
- Implementation of Azure Document Intelligence integration with comprehensive field extraction
- Development of user interface with modern design patterns and responsive layout
- Implementation of complex data transformation from unstructured to structured format
- Development of confidence scoring system for data quality assurance
- Optimization of file processing and API call handling
- User experience improvements with loading states and error handling
- Documentation of data structures and API integration patterns

10 Maconomy

Maconomy is a comprehensive desktop application for work time and absence management, designed for integration with the Maconomy ERP system. The application enables employees to record work time, manage absences, track expenses, and access key information about projects and tasks. The special feature of the application is its robust integration with the existing Maconomy ERP system via ODBC protocol, which enables data synchronization in real time. The application is built using modern web technologies and available as a desktop application using Electron and Capacitor.

10.1 Key Features of Maconomy

- **Work Time Management:** Comprehensive system for recording work time by projects, tasks, and days
- **Absence Management:** Complex system for managing absences with support for different types of absences (vacation, sick leave, training)
- **Expense Management:** System for recording and tracking expenses related to projects
- **Favorites System:** Storage of frequently used project and task combinations for quick access
- **Interactive Calendar:** Visual display of time entries, absences, and expenses by day
- **Detailed Statistics:** Time and absence statistics for better resource management
- **System Synchronization:** Automatic data synchronization with Maconomy ERP system via ODBC
- **Project and Task Management:** Access to information about projects, tasks, and access rights
- **Multi-language Support:** Support for multiple languages with easy switching

10.2 My Implementations in the Maconomy Project

- **Development of Key Features:**
  - Implementation of a complex time tracking system with validation logic
  - Development of an advanced absence management system with support for different types of absences
  - Implementation of an expense tracking system with document attachment capability
  - Development of a system for storing favorite project and task combinations
- **Backend Integrations:**
  - Development of robust ODBC integration for communication with Maconomy ERP system
  - Implementation of data synchronization system using cron jobs
  - Development of REST APIs for all application functionality
  - Implementation of session management and user authorization system
- **Architecture and Optimization:**
  - Implementation of modular architecture with clear separation of responsibilities
  - Development of an efficient data caching system

  - ODBC query optimization for better performance
  - Implementation of error handling and user notification system
  - Implementation of advanced filtering and search system
  - Application performance optimization with efficient caching
- **Electron Integration:**
  - Configuration of Electron environment for desktop application
  - Implementation of application update system
  - Development of local caching and offline functionality system
  - Interface optimization for different operating systems
  - Security improvements with local data encryption
- **User Experience:**
  - Development of adaptive user interface for different screen sizes
  - Addition of advanced shortcuts for faster work

10.3 Technical Implementation of Maconomy

10.3.1 Frontend Technology Stack

- **UI Framework:** React with TypeScript
- **Desktop Integration:** Electron with Capacitor
- **Styling:** Tailwind CSS with Ant Design components
- **Form Management:** React Hook Form
- **API Communication:** Axios with JWT authentication
- **Multi-language Support:** i18next for multiple language support
- **Routing:** React Router for navigation
- **Error Tracking:** Sentry for debugging in production
- **Optimization:** React Window for virtualization of large lists
- **Development Tools:** Modern development environment with automatic refresh

10.3.2 Backend Technology Stack

- **Backend Framework:** NestJS with TypeScript
- **Database:** PostgreSQL with TypeORM
- **ERP Integration:** ODBC for connecting to Maconomy ERP
- **Authentication:** JWT with persistent sessions

- **API Documentation:** Swagger for REST API documentation
- **Monitoring:** Health checks for monitoring system status
- **Cron Jobs:** Cron jobs for automatic data synchronization
- **Security:** Azure Key Vault for storing sensitive data

10.3.3 Architectural Features

The application follows modern architecture with the following characteristics:

- **Modular Design:** Clear separation between different functional modules (work time, absences, expenses)
- **Reusable Components:** Common components are shared between modules
- **Service Layer:** Dedicated services for API communication
- **ERP Integration:** Robust connection with Maconomy ERP system via ODBC protocol
- **Type Safety:** Strict use of TypeScript on both frontend and backend
- **Automated Synchronization:** Cron jobs for regular data synchronization
- **Desktop Approach:** Desktop application with Electron enables offline operation
- **Electron Integration:** Efficient communication between web and desktop parts of the application
- **Scalability:** Architecture designed for working with large amounts of data
- **Real-time Updates:** Automatic adjustment of data upon changes in ERP system

10.4 My Contribution to the Project

In the Maconomy project, I focused on the following areas:

- Development of key features for work time, absence, and expense management
- Implementation of ODBC integration for communication with Maconomy ERP system
- Development of backend APIs and CRUD operations for all entities
- Optimization of systems for synchronization and data processing
- Configuration of Electron environment for desktop application
- Implementation of error handling and monitoring system
- Development of user interface with modern UI components
- Security and session management improvements

- Implementation of advanced filtering and search system
- Addition of advanced shortcuts for faster work
- Security improvements with local data encryption
- Development of adaptive user interface for different screen sizes
- Application performance optimization with efficient caching

11 Extra XML Parser

Extra XML Parser is a web application for managing and assigning business expenses from bank statements. The application enables processing of CAMT.053 XML bank statements and tracking and assigning expenses to team members. The project is designed as a simple and efficient application for expense management in small organizations. The special feature of the application is its ability to process structured XML bank statements and extract expense data with manual assignment to team members.

11.1 Key Features of Extra XML Parser

- **Bank Statement Processing:** Automatic processing of CAMT.053 XML bank statements with transaction extraction
- **Expense Assignment:** System for manual assignment of expenses to different team members
- **Filtering and Search:** Advanced filters for searching expenses by assignment status, date periods, and team members
- **Sorting:** Ability to sort by any column (date, amount, description, recipient)
- **Pagination:** Display optimization with ability to choose number of displayed items
- **Statistics:** Display of total expenses and expenses without tax
- **Upload Details:** System for managing file uploads and tracking processing status
- **Docker Integration:** Development and production environment with Docker Compose

11.2 My Implementations in the Extra XML Parser Project

- **XML Processing Development:**
  - Implementation of a system for processing CAMT.053 XML bank statements
  - Development of extraction of only DEBIT transactions (expenses)
  - Implementation of a system for managing XML namespaces
  - Development of a system for validation and processing of structured XML data
- **Data Management:**

  - Development of queries and filtering for various criteria (year, quarter, month)
  - Implementation of a system for sorting by any column
  - Development of pagination with ability to choose number of displayed items
  - Implementation of a system for aggregation and statistics display
- **User Interface:**
  - Development of user interface for uploading multiple files at once
  - Implementation of a system for manual editing of amounts without tax
  - Development of bulk selection functionality for mass operations
  - Implementation of visual status indicators (processing, success, failed)
- **Backend Infrastructure:**
  - Development of PostgreSQL database structure with tables for uploads and items
  - Implementation of a system for tracking user uploaded files
  - Development of validation and error handling for processing errors
  - Implementation of a system for preventing duplicate file uploads
- **Docker and Production Deployment:**
  - Configuration of Docker environment with Apache for production use
  - Implementation of SSL certificates via Let's Encrypt
  - Development of a system for secure password management with .htpasswd
  - Configuration of Docker Compose profiles for development and production environments

11.3 Technical Implementation of Extra XML Parser

11.3.1 Technology Stack

- **Backend Framework:** PHP 8 with PDO for database
- **Database:** PostgreSQL with structured queries
- **Tracking:** Database singleton pattern for managing connections
- **Web Server:** Apache with PHP module
- **Containerization:** Docker and Docker Compose for managing environments
- **SSL:** Let's Encrypt for secure HTTPS communication
- **Authentication:** .htpasswd for basic access protection
- **XML Processing:** PHP SimpleXML for parsing bank statements

11.3.2 Architectural Features

The application follows a simple, yet effective architecture with the following characteristics:

- **OIIS Architecture:** Organized in a simplified structure (Processing - Selection - Interface - Storage)
- **Modular Structure:** Separate files for different functionalities (upload, display, processing)
- **Database Singleton:** Centralized management of database connections
- **Error Handling:** Extensive error handling with user messages
- **Security:** Preventing duplicate uploads and data validation
- **Docker Support:** Container context for different environments
- **Responsive Design:** Adaptive user interface for different screens

11.4 My Contribution to the Project

In the Extra XML Parser project, I focused on the following areas:

- Development of the complete application architecture with PHP and PostgreSQL
- Implementation of a system for processing CAMT.053 XML bank statements
- Development of user interface for uploading and reviewing expenses
- Implementation of a system for filtering, sorting, and paginating data
- Development of functionality for manual editing of amounts without tax and expense assignment
- Configuration of Docker environment for development and production use
- Implementation of SSL certificates and security practices for production environment
- User experience and visual indicator improvements

12 Bo

Bo is a comprehensive mobile and web application for mental health and wellness, designed to help users manage their mental well-being, connect with therapists, and access various exercises for improving mental health. The application combines modern web technologies with monorepo architecture, enabling efficient operation on mobile devices (iOS/Android) and the web with the same code base. The special feature of the application is its flexible architecture, which enables operation across multiple platforms with optimization for each.

12.1 Key Features of Bo

- **Daily Mood Tracking:** System for recording daily mood and analyzing mental well-being patterns
- **Video Therapy:** Integration with Dyte platform for direct video sessions with therapists
- **Multi-language Support:** Support for multiple languages with easy switching
- **Mental Health Exercises:** Access to breathing exercises, meditation exercises, and mindfulness exercises
- **Tasks and Reminders:** System for managing therapeutic tasks and reminders
- **Write Journal:** Personal journaling system for recording thoughts and feelings
- **Therapist Selection:** Personalized system for choosing the most suitable therapist based on issues and preferences
- **Completed and Scheduled Sessions:** Complex system for managing therapy appointments
- **Payments:** Integration with Stripe for secure and reliable payment transactions
- **Subscription Plans:** System for managing subscriptions via RevenueCat
- **Monitoring:** Integration with analytics platforms (Sentry, PostHog)

12.2 My Implementations in the Bo Project

- **Development of Key Features:**
  - Implementation of a system for daily mood tracking with visualizations
  - Development of a complex system for managing therapy bookings
  - Implementation of a system for therapist selection with matching algorithm
  - Development of integration with Dyte platform for video calls
  - Implementation of a system for tasks and reminders with different types of exercises
- **Backend Integrations:**
  - Development of robust NestJS REST API architecture
  - Implementation of authorization system with JWT and social logins (Google, Apple, Facebook)
  - Development of integration with Stripe for payment transactions
  - Implementation of a system for managing push notifications
  - Development of a system for email communication with Mailchimp
- **Monorepo Architecture:**

  - Setup and management of monorepo structure with Lerna
  - Development of shared packages for components, constants, translations, and type safety
  - Optimization of data sharing between mobile and web applications
  - Implementation of efficient build process for multiple platforms
- **Cross-platform Development:**
  - Development of React Native application with Expo for iOS, Android, and Web
  - Optimization of requirements for different platforms
  - Implementation of adaptive design for different screen sizes
  - Integration of native functionalities (image capture, notifications, location)

12.3 Technical Implementation of Bo

12.3.1 Frontend Technology Stack

- **Monorepo:** Lerna for managing multiple projects with shared packages
- **UI Framework:** React Native with Expo for mobile and web
- **Cross-platform:** For iOS, Android, and Web
- **Navigation:** React Navigation with Tab and Stack navigations
- **Video Conferencing:** Dyte SDK for video calls
- **Payments:** Stripe SDK and RevenueCat for subscriptions
- **Authentication:** JWT with social logins (Google, Apple, Facebook)
- **Monitoring:** Sentry and PostHog for analytics

12.3.2 Backend Technology Stack

- **Backend Framework:** NestJS with TypeScript
- **Database:** PostgreSQL with Prisma ORM
- **Email Services:** Mailchimp for marketing campaigns
- **API Documentation:** Swagger for REST API documentation
- **Logging:** Winston for structured log messages
- **Automation:** Cron jobs for scheduled tasks

12.3.3 Architectural Features

The application follows modern architecture with the following characteristics:

- **Monorepo Union:** Organized with Lerna for managing multiple projects
- **Modular Design:** Clear separation between different functional modules (Today, Therapy, Center)
- **Type Safety:** Strict use of TypeScript at all levels
- **Cross-platform Compatibility:** One code base for multiple platforms
- **Real-time Updates:** Integration with push notifications
- **Scalability:** Architecture designed for future growth and development

12.4 My Contribution to the Project

In the Bo project, I focused on the following areas:

- Development of monorepo architecture with shared packages
- Implementation of a complex daily mood tracking system
- Development of non-standard navigation structure with Tab and Stack
- Implementation of integration with video teleconferencing platforms
- Implementation of payment integrations
- Optimization of the application for different platforms and screen sizes
- Development of user interface with mobile-first approach
- Security and user session management improvements

13 InboxZero

InboxZero is an advanced AI-powered web platform for email management, helping users achieve and maintain an "inbox zero" state with automated email management. The platform uses AI for categorization, archiving, responding, and managing email, saving users time and reducing stress when dealing with large volumes of messages. The project is built using the most modern web technologies and follows best practices in web application development. The special feature of the platform is its integration with multiple AI providers and the ability to customize operating rules.

13.1 Key Features of InboxZero

- **AI Personal Assistant:** Advanced email automation system with ability to set rules for archiving, labeling, and responding
- **Bulk Unsubscribe:** Automated system for unsubscribing from numerous email lists and newsletters
- **Cold Email Blocker:** AI system for recognizing and filtering unwanted promotional emails
- **Reply Tracker:** System for tracking emails that require a response
- **Smart Categories:** AI categorization of email into different categories
- **Knowledge Base:** Contextual knowledge base for improving accuracy of AI decisions
- **Email Analytics:** Detailed statistics on email usage with Tinybird integration
- **Premium Subscription:** Integration with Lemon Squeezy for managing subscriptions
- **Multi-provider Support:** Support for Gmail and Outlook with unified interface

13.2 My Implementations in the InboxZero Project

- **AI Feature Development:**
  - Implementation of a system for selecting and executing AI rules with multiple AI providers
  - Development of a complex system for generating responses with AI
  - Implementation of a system for sender categorization with AI
  - Development of a system for recognizing unwanted promotional emails
  - Implementation of contextual knowledge base for improving AI decisions
- **Email Integrations:**
  - Development of robust integration with Gmail API for email management
  - Implementation of integration with Microsoft Outlook API
  - Development of a system for webhook management for real-time notifications
  - Implementation of a system for batch operations to work with multiple messages
  - Development of a system for archiving, labeling, and filtering email
- **Automation and Processing:**
  - Implementation of systems for automated unsubscribing with Playwright
  - Development of queue systems for processing larger amounts of data
  - Implementation of Redis cache systems for performance optimization

  - Development of a system for bulk archiving email
  - Implementation of a system for tracking responses with AI analysis
- **Analytics and Reports:**
  - Integration with Tinybird for real-time analytics
  - Development of systems for visualizing usage statistics
  - Implementation of systems for tracking AI usage and costs
  - Development of reporting systems for users
- **Security and Payments:**
  - Implementation of security mechanisms for API keys and access
  - Integration with Lemon Squeezy for managing subscriptions
  - Development of systems for encrypting sensitive data
  - Implementation of referral system

13.3 Technical Implementation of InboxZero

13.3.1 Frontend Technology Stack

- **UI Framework:** Next.js 15 with App Router and React 19
- **Programming Language:** TypeScript
- **Styling:** Tailwind CSS with shadcn/ui components (Radix UI)
- **Authentication:** NextAuth.js with JWT
- **State Management:** Jotai for global state, SWR for data
- **Form Handling:** React Hook Form with Zod validation
- **AI Integration:** Vercel AI SDK with multiple AI providers (OpenAI, Anthropic, Google, Bedrock, Groq, Ollama)
- **Analytics:** PostHog, Tinybird, Vercel Analytics
- **Error Tracking:** Sentry
- **Cache:** Redis/Upstash

13.3.2 Backend Technology Stack

- **Server Actions:** Next.js Server Actions for backend logic
- **Database:** PostgreSQL with Prisma ORM
- **API Integrations:** Google Gmail API, Microsoft Graph API
- **Automation:** Playwright for browser automation

- **Browser Automation:** Fastify application for unsubscriber service
- **Queue Systems:** P-Queue for task management
- **Email Systems:** Resend for transactional email
- **Payment System:** Lemon Squeezy integration
- **CMS:** Sanity.io for blog
- **API Documentation:** Swagger/OpenAPI

13.3.3 Architectural Features

The application follows modern monorepo architecture with the following characteristics:

- **Monorepo Structure:** Workspace with multiple applications and packages
- **AI-powered Architecture:** Integration of multiple AI providers with ability to customize models
- **Modular Design:** Clear separation between different functional modules (assistant, unsubscriber, analytics)
- **Real-time Processing:** Webhook system for real-time email processing
- **Type Safety:** Strict use of TypeScript throughout the application
- **Scalability:** Architecture designed for high load with queue systems
- **Multi-tenant:** Support for multiple user email accounts

13.4 My Contribution to the Project

In the InboxZero project, I focused on the following areas:

- Development of key AI features for email automation
- Implementation of integrations with Gmail and Outlook APIs
- Development of a system for AI-powered categorization and responding
- Implementation of bulk unsubscribe system with browser automation
- Development of real-time analytics systems with Tinybird integration
- Integration of multiple AI providers with unified interface
- Performance optimization with Redis cache and queue systems
- Development of security mechanisms for API keys and session management
- Implementation of premium functionality with Lemon Squeezy integration

14 InboxZero Extension

InboxZero Extension is an advanced browser extension for Gmail that enables integration of an AI assistant directly into the Gmail interface. The extension uses InboxSDK to integrate with Gmail and enables users to quickly access the AI assistant while reading and responding to email messages. The special feature of the extension is its seamless integration with Gmail, where a button is added to each email thread to open the AI assistant in a side panel.

14.1 Key Features of InboxZero Extension

- **Gmail Integration:** Seamless integration with Gmail interface using InboxSDK
- **AI Assistant in Side Panel:** Opening AI assistant directly from email thread
- **Authentication System:** Secure system for authentication using JWT tokens
- **Session Management:** Automatic management of user sessions with session age management
- **Side Panel:** Chrome Extension API for opening side panel
- **Seamless UX:** Transparent integration into user workflow

14.2 My Implementations in the InboxZero Extension Project

- **Extension Development:**
  - Implementation of background script for managing communication between components
  - Development of content script for integrating with Gmail interface via InboxSDK
  - Implementation of side panel with React application
  - Development of a system for handling different types of messages between components
- **Authentication and Security:**
  - Development of authentication system with external window opening
  - Implementation of JWT session management with support for 24-hour persistence
  - Development of automatic session token renewal with API calls
  - Implementation of protection for sensitive data in local storage
- **Integrations:**
  - Integration with InboxSDK for adding buttons to Gmail thread
  - Implementation of dynamic iframe for embedding external application
  - Integration with external API for AI assistant

- **Optimization and User Experience:**
  - Implementation of asynchronous communication between components
  - Development of a system for timed opening of side panel
  - Behavior optimization with delay for reliable operation
  - Implementation of error handling and user notifications

14.3 Technical Implementation of InboxZero Extension

14.3.1 Technology Stack

- **Extension Framework:** WXT (Web Extension Toolkit) for developing browser extensions
- **UI Framework:** React with TypeScript for side panel
- **Gmail Integration:** InboxSDK for integrating with Gmail interface
- **APIs:** Chrome Extension API for managing tabs, storage, and side panel
- **Authentication:** JWT system with session management
- **Manifest:** Manifest V3 for modern browser extensions
- **CSP:** Content Security Policy for secure script execution
- **Styling:** CSS for adaptive user interface

14.3.2 Architectural Features

The extension follows a modular architecture with the following characteristics:

- **Modular Design:** Separation into Background Script, Content Script, and Side Panel
- **Asynchronous Communication:** Messaging system for communication between components
- **Session Management:** Automatic management of user sessions with timeout function
- **Security:** Secure storage of sensitive data in local storage
- **Error Handling:** Robust error handling with user notifications
- **Cross-browser Support:** Support for Chrome and Firefox via WXT

14.4 My Contribution to the Project

In the InboxZero Extension project, I focused on the following areas:

- Development of the complete extension architecture with WXT framework
- Implementation of integration with Gmail via InboxSDK
- Development of authentication system with session management
- Implementation of communication system between Background, Content, and Side Panel
- Development of side panel with React application
- Optimization of user experience and integrating with Gmail workflow
- Implementation of security measures and error handling
- Configuration of CSP rules for secure extension execution

15 Extra Website

Extra Website is a modern, high-performance company website for EXTRA DEV, a software development agency. The website showcases the company's services, case studies, blog content, and career opportunities through an exceptional user experience built with cutting-edge web technologies. The special feature of the website is its automated job application workflow that integrates Slack, Notion, and Google Drive for seamless application processing, along with sophisticated animations and comprehensive content management through MDX for blog posts.

15.1 Key Features of Extra Website

- **Animated Homepage:** Immersive hero section with gradient backgrounds, floating particles, geometric shapes, and parallax effects
- **Job Application Workflow:** Automated application processing system integrating Slack notifications, Notion database entries, and Google Drive file management
- **Case Studies Portfolio:** Dynamic case study display with video backgrounds and detailed project information
- **Blog System:** Full-featured blog with MDX support for rich content authoring and rendering
- **Careers Section:** Job listings with detailed descriptions and integrated application forms
- **Contact Form:** Animated contact form with validation, rate limiting, and email integration
- **SEO Optimization:** Comprehensive metadata, Open Graph, and Twitter Card support

- **Responsive Design:** Fully adaptive layout for all screen sizes with mobile-first approach
- **Dark Mode:** Modern dark theme with gradient accents throughout

15.2 My Implementations in the Extra Website Project

- **Frontend Development:**
  - Development of animated homepage with Framer Motion including floating particles, geometric shapes, and parallax effects
  - Implementation of interactive service cards with hover animations and gradient effects
  - Creation of responsive navigation header with mobile menu and smooth transitions
  - Development of footer component with animated social media links
  - Implementation of breadcrumb navigation for better user experience
- **Job Application Workflow Integration:**
  - Development of automated job application processing system with multi-platform integration
  - Implementation of Google Drive API integration for CV storage with job-specific folder organization
  - Creation of Notion database integration using Notion API for structured application data storage
  - Development of Slack webhook integration for real-time team notifications with formatted messages
  - Implementation of end-to-end workflow: CV upload to Google Drive, application data to Notion, notification to Slack
  - Creation of property mapping system for Notion database fields with type-safe data conversion
  - Development of error handling and validation for file uploads and API integrations
  - Implementation of character limit validation for Notion API compliance
- **Content Management:**
  - Development of MDX blog system with gray-matter for frontmatter parsing
  - Implementation of blog listing page with card-based layout
  - Creation of dynamic blog post pages with MDX content compilation
  - Development of case studies data structure and dynamic routing
  - Implementation of job listings system with detailed pages
- **API and Backend:**

- Development of contact form API route with Next.js Server Actions
- Implementation of rate limiting middleware for contact form protection
- Creation of email service integration with Nodemailer
- Development of form validation utilities for client and server-side validation

15.3 Technical Implementation of Extra Website

15.3.1 Frontend Technology Stack

- **Web Framework:** Next.js 15 with App Router
- **UI Library:** React 19
- **Language:** TypeScript
- **Styling:** Tailwind CSS with custom dark theme
- **Animations:** Framer Motion for complex animations and transitions
- **Icons:** Lucide React for modern icon set
- **Typography:** Inter font from Google Fonts
- **Content:** MDX with next-mdx-remote for blog posts
- **Markdown Processing:** gray-matter for frontmatter parsing

15.3.2 Backend Technology Stack

- **API Routes:** Next.js API routes for server-side logic
- **Email Service:** Nodemailer for sending contact form emails
- **Notion Integration:** Notion API client for creating and managing application database entries
- **Google Drive Integration:** Google Drive API for file uploads and folder organization
- **Slack Integration:** Slack webhook API for sending formatted notifications
- **Validation:** Custom validation utilities for form data
- **Middleware:** Next.js middleware for rate limiting
- **Environment:** Environment variables for configuration

15.3.3 Architectural Features

The application follows modern Next.js architecture with the following characteristics:

- **App Router:** Latest Next.js App Router for optimal performance and developer experience
- **Server Components:** Extensive use of Server Components for better performance
- **Client Components:** Strategic use of Client Components only where interactivity is needed
- **Component Modularity:** Reusable components with clear separation of concerns
- **Type Safety:** Comprehensive TypeScript coverage throughout the application
- **Performance:** Optimized images, code splitting, and lazy loading
- **Responsive Design:** Mobile-first approach with Tailwind CSS breakpoints
- **Accessibility:** Semantic HTML and proper ARIA labels
- **SEO:** Server-side rendering with comprehensive metadata

15.4 My Contribution to the Project

In the Extra Website project, I focused on the following areas:

- Complete development of the company website from design to deployment
- Implementation of sophisticated animations and visual effects with Framer Motion
- Development of automated job application workflow integrating Slack, Notion, and Google Drive
- Creation of Google Drive API integration with job-specific folder organization for CV storage
- Implementation of Notion database integration for structured application data management
- Development of Slack webhook system for real-time team notifications with formatted messages
- Creation of MDX blog system for content management
- Implementation of interactive contact form with validation and email integration
- Development of rate limiting middleware for API protection
- Creation of case studies and careers sections with dynamic routing
- Implementation of comprehensive SEO metadata system
- Optimization of performance with Server Components and code splitting

16 UCB IOF 2026 Microsite

UCB IOF 2026 Microsite is a congress microsite built from start to finish for Madano (client: UCB) to support UCB's presence at the WCO-IOF-ESCEO 2026 congress in Prague. The site promotes EVENITY (romosozumab), UCB-sponsored events (satellite symposium, evening event, interactive booth), and provides congress details, resources, and a pre-congress survey. The application is built with Next.js 16, React 19, and Tailwind CSS 4 and follows pharmaceutical industry compliance patterns including an exit-ramp flow for external links and HCP-only disclaimers.

16.1 Key Features of UCB IOF 2026 Microsite

- **Event-Centric Homepage:** Hero with key message, welcome section, and three event cards (Satellite Symposium, Standalone Evening Event, Interactive Booth Experience) with distinct branding and call-to-action links
- **Add to Calendar:** Calendar integration with Google Calendar, Microsoft Outlook, Yahoo Calendar, and Apple Calendar (.ics download), respecting external-link compliance via exit ramp
- **Event Detail Pages:** Dedicated pages for sponsored symposium, evening event, and interactive booth with headings, event info, faculty sections with bios and modals, agenda sections, and cross-links to other events
- **Congress Details:** Congress details page with calendar overview, Google Maps integration (custom-styled map and satellite toggle), key dates, programme schedule, and login information
- **Resources Page:** Curated EVENITY resources for healthcare professionals ahead of the congress
- **Survey:** Pre-congress survey form to gather HCP insights, with CTA to resources
- **Exit Ramp (Pharma Compliance):** All external links open via a modal (exit ramp) before leaving the site, with iframe postMessage handling for tracking; internal links and API routes (e.g. SmPC download) remain direct
- **Regulatory Content:** Footer and page-level regulatory text, prescribing information links, SmPC download via API route (proxy to EMA PDF), and HCP-only disclaimer in header
- **Responsive Design:** Sticky header with desktop dropdown and mobile hamburger menu, responsive layouts and typography (Nunito Sans, Bebas Neue, Geist)
- **Analytics:** PostHog integration for product analytics
- **SEO and Access:** Metadata per page, noindex/nofollow for closed HCP audience

16.2 My Implementations in the UCB IOF 2026 Microsite Project

- **Full-Stack Implementation:**
  - Development of the complete microsite from design to deployment (single developer, start to finish)
  - Next.js App Router structure with layout, global providers (PostHog, ExitRamp), and reusable sections and components
  - API route for EVENITY SmPC PDF download (server-side fetch from EMA, streamed response with correct Content-Disposition and caching)
- **Pharma Compliance and UX:**
  - Exit ramp context and modal: external links open in a modal with "Continue" to open in new tab; SafeLink component for automatic exit-ramp handling; integration with Add to Calendar so external calendar URLs use exit ramp
  - Centralized footer and page-level regulatory content with SafeLink for external prescribing links and internal link for SmPC download
- **Event and Calendar Features:**
  - EventCard component with configurable colors, calendar button, and link to event detail page
  - AddToCalendarButton with dropdown (Google, Outlook, Yahoo, .ics) and client-side ICS generation; all external calendar URLs gated by exit ramp
- **Event Detail and Content:**
  - EventHeading, EventInfoSection, AgendaSection, FacultySection with FacultyCard and BioModal for speaker bios
  - Congress details, calendar, map (Google Maps with custom Snazzy-style map and map/satellite toggle), dates, programme, and login information sections
  - Resources (desk-resources) and Survey page with SurveyForm component
- **UI and Layout:**
  - Header with EVENITY and WCO-IOF-ESCEO logos, desktop nav (UCB Events dropdown, Survey, Resources, Congress Details), mobile hamburger and slide-down menu
  - Hero with HeroImageFrame, WelcomeSection, EventsSection, Footing with animated border image
  - Footer component accepting rich content (regulatory paragraphs, links, references) per page

16.3 Technical Implementation of UCB IOF 2026 Microsite

16.3.1 Frontend Technology Stack

- **Web Framework:** Next.js 16 with App Router

- **UI:** React 19 with TypeScript
- **Styling:** Tailwind CSS 4 with custom theme (primary, secondary, hero background, grey-medium, etc.)
- **Fonts:** Next.js Google Fonts (Geist, Geist Mono, Nunito Sans, Bebas Neue)
- **Images:** next/image for logos and artwork with priority and sizes
- **Maps:** Google Maps JavaScript API with custom styling and marker for congress venue
- **Analytics:** PostHog (PostHogProvider client component)

16.3.2 Backend and API

- **API Routes:** Next.js Route Handlers for proxying EMA SmPC PDF with caching headers
- **Compliance:** Exit ramp implemented via React context and modal; no backend session required

16.3.3 Architectural Features

The application follows a modern Next.js architecture with the following characteristics:

- **App Router:** File-based routing with layout, page-level metadata (title, description, keywords, robots)
- **Server and Client Split:** Server Components for static content; Client Components for header (dropdown, mobile menu), AddToCalendarButton, CongressMapSection, ExitRampProvider, PostHogProvider
- **Pharma Compliance:** Exit ramp for external links, HCP-only disclaimer, noindex/nofollow, consistent regulatory footers
- **Modular Components:** Reusable sections (EventCard, EventHeading, EventInfoSection, AgendaSection, FacultySection, Footer, etc.) composed per page
- **Type Safety:** TypeScript throughout; CalendarEvent interface shared for calendar and event data

16.4 My Contribution to the Project

In the UCB IOF 2026 Microsite project, I was responsible for the entire project from start to finish:

- Full development of the microsite: architecture, routing, layout, and all pages (home, sponsored-symposium, evening-event, interactive-booth-experience, desk-congress-details, desk-resources, survey, exit)
- Implementation of pharma-compliant exit ramp (context, modal, SafeLink) and integration with Add to Calendar and all external links

- Development of Add to Calendar (Google, Outlook, Yahoo, .ics) with client-side ICS generation and exit-ramp handling for external URLs
- Integration of Google Maps with custom styling and map/satellite toggle for congress venue
- Implementation of API route for SmPC PDF download with caching and correct headers
- Responsive header with dropdown navigation and mobile menu; footer with configurable regulatory content
- PostHog analytics integration and page-level metadata for SEO and access control

17 CTPS-GUI Particle Plan Report

CTPS-GUI (Cosylab Treatment Planning System) is an FDA-approved medical radiation therapy planning system supporting photon and particle therapy. I implemented the complete **particle treatment plan report** feature: a PDF export that documents particle (proton/carbon) plans for clinical use. The work mirrors the existing photon plan report; the system already had common report infrastructure and data models-I added the particle-specific reporting pipeline and all particle-only sections.

17.1 Key Features of the Particle Plan Report

- **Full report pipeline:** Export from UI to PDF with validation (unsaved plan, comparison mode, dose selection), in-memory report build, preview dialog, and save to filesystem
- **Reuse of common sections:** Patient data, warnings/errors, anatomy model, structures, plan overview, intent, prescription (Gy (RBE)), setup position, couch shifts, dose reference points, normalization, clinical goals, dose-volume statistics, DVH, dose distribution screenshots, setup beams
- **Particle-specific sections:** Dose calculation settings, motion synchronization (conditional), distributions (primary and others, RBE/Physical/LET/Multi-RBE), RBE models (only used ones), Multi-RBE models (conditional), optimization settings, robust optimization/evaluation scenarios, bolus, STV node generation, node placement, expose/avoid/shield rules, STV structure generation, treatment beams table and per-beam properties (snout-mounted devices, energy layers)
- **Conditional logic:** Sections shown only when relevant (e.g. motion sync, Multi-RBE, robust scenarios)
- **Warning/error processor:** Particle-specific checks (e.g. unapproved plan, obsolete optimization, beam limits) integrated into the report preview

17.2 My Implementations in the Particle Plan Report

- **Core export utility:**
  - Implemented the particle plan report export utility in the Particle planning module: save/validation flow, comparison-mode and dose-distribution checks, long-running prepare/export actions, preview dialog, and PDF save to filesystem
  - Implemented section composition to build the ordered list of report sections from the common planning module and particle-specific sections
  - Integrated with the base plan report export for FlowDocument creation and persistence
- **Particle-specific report sections:**
  - Sections for dose calculation settings, prescription (Gy (RBE)), optimization settings, motion synchronization, distributions, RBE and Multi-RBE models, robustness scenarios, STV node and structure generation, treatment beams and per-beam properties, snout-mounted devices, energy layers, clinical goals, dose-volume statistics, and optimization objectives
- **Warning/error handling:**
  - Implemented the particle plan report warning/error processor extending the existing plan review processor, with particle-specific warnings (e.g. plan not approved, optimization obsolete, beam limits exceeded, scan reference point)
- **Data and patterns:**
  - All sections use the working particle treatment plan and related view models; conditional sections use plan state (e.g. motion sync, robust settings, Multi-RBE usage)
  - Followed the existing report section pattern: heading level, page break, data checks, content creation with WPF FlowDocument blocks and shared styles
  - Used shared string resources for headings and labels

17.3 Technical Implementation

17.3.1 Technology Stack

- **Platform:** .NET 8, WPF (Particle and common planning modules)
- **Report output:** FlowDocument-based report generation, PDF export via the base plan report export
- **Data:** Working particle treatment plan, beam and view model hierarchies, RBE and robust scenario definitions
- **UI integration:** Plan report menu and export command, preview dialog

17.3.2 Architectural Notes

- **Modularity:** Particle report lives in the Particle module; common sections and base export/print logic stay in the common planning module
- **Consistency:** Section ordering and conditional visibility match the photon report structure where applicable
- **Validation:** Export blocked when dose is missing/obsolete, comparison mode is on, or non-primary/robust dose is selected

17.4 My Contribution to the Project

In the CTPS-GUI Particle Plan Report project, I focused on the following areas:

- Design and implementation of the full particle plan report feature
- Main export utility and all particle-specific report section implementations
- Warning/error processor integrated into the report preview
- Reuse of existing common sections and base report infrastructure
- Validation and conditional section logic for export and section visibility
- Correct ordering and formatting so the generated PDF meets clinical documentation requirements