**Slotly Backend Development Progress – Iteration 1**

**Project: Slotly  
Title: Backend Development Progress – Iteration 1  
Date Range: July 3 – July 17, 2025  
Report Version: 1.1  
Prepared by: Development Team**

**1. Introduction**

**Slotly is a comprehensive backend platform designed to manage business operations, staff scheduling, customer bookings, and administrative oversight for service-based businesses. The platform serves multiple user roles including customers, staff members, managers, and system administrators, each with distinct functional requirements and access levels.**

**The architecture follows a layered backend design implementing: Client Layer → API Gateway → Middleware → Business Logic → Data Access → Database. This architectural pattern ensures proper separation of concerns, scalability, and maintainability across all system components.**

**The primary purpose of this iteration was to finalize critical backend logic components for managers and administrators, establish a robust notification system, and implement comprehensive error handling and monitoring capabilities. This two-week development cycle focused on creating stable, scalable backend infrastructure that supports the full spectrum of business operations while maintaining security and performance standards.**

**Key objectives for this iteration included completing the admin dashboard logic, enhancing manager dashboard features, implementing a multi-channel notification system, and establishing reliable error tracking and monitoring systems.**

**2. Work Completed**

**2.1 Admin Dashboard Logic ✅**

**The admin dashboard has been fully implemented with comprehensive business management capabilities:**

* **Business Suspension System: Complete workflow for suspending and reactivating business accounts with automated notification triggers**
* **Business Approval Pipeline: Streamlined approval process for new business registrations with verification checkpoints (/apps/backend/src/app/api/admin/businesses/[id]/approve/route.js)**
* **Referral Tracking System: Advanced analytics for tracking referral performance, conversion rates, and reward distribution**
* **Coupon Management: Comprehensive coupon creation, validation, and tracking system with usage analytics (/apps/backend/src/app/api/admin/coupons/[id]/reward/route.js)**
* **User Management: Advanced user search, filtering, and management capabilities with role-based access controls**

**2.2 Manager Dashboard Features ✅**

**Enhanced manager-specific functionality has been implemented:**

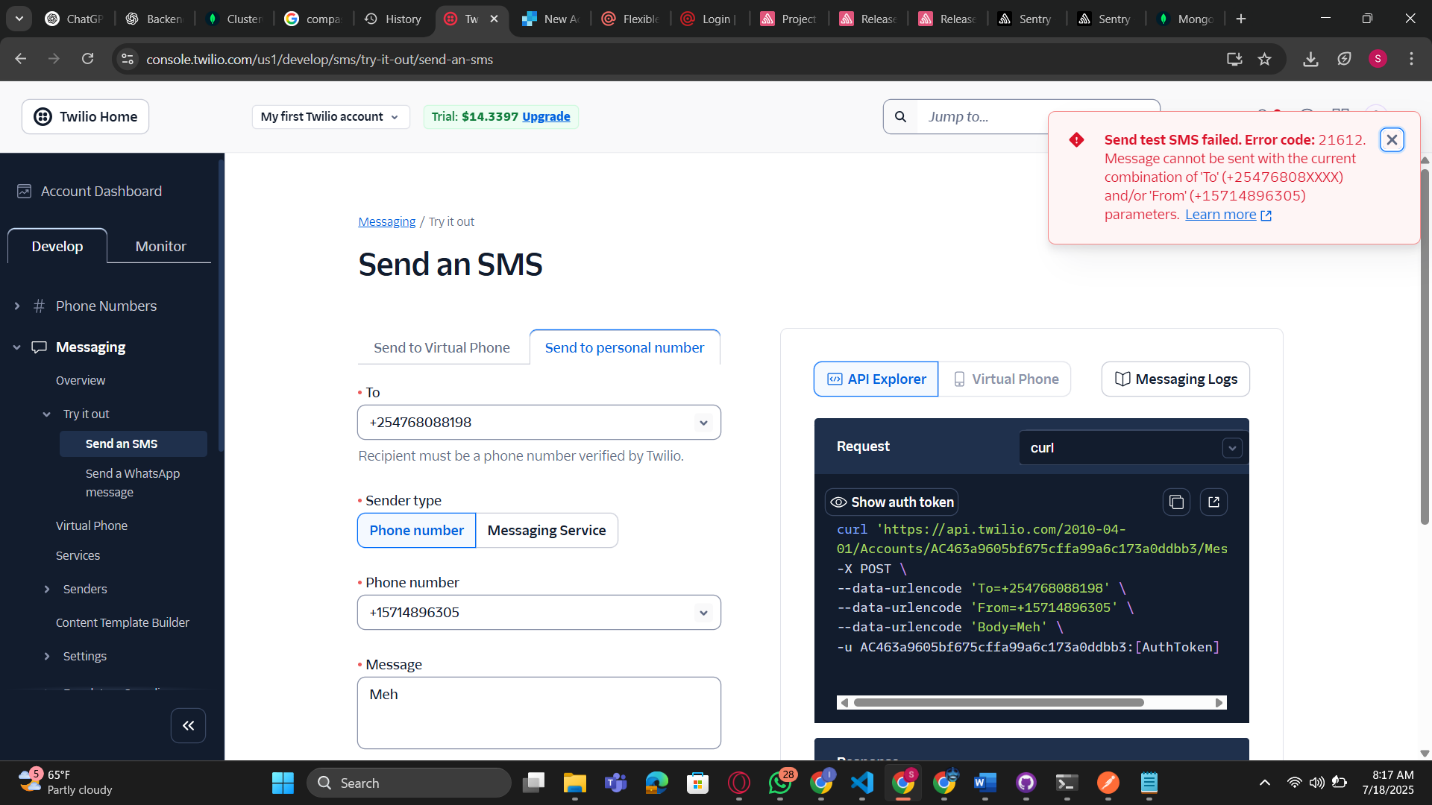
* **Staff Review Management: Complete review and approval workflow for staff performance evaluations**
* **Time-off Approval System: Automated time-off request processing with conflict detection and approval workflows**
* **Performance Reports: Advanced reporting dashboard with customizable metrics and export capabilities**
* **Business Analytics: Real-time business performance metrics including revenue, bookings, and customer satisfaction**
* **Staff Scheduling: Comprehensive scheduling system with availability management and conflict resolution**

**2.3 Notification System ✅**

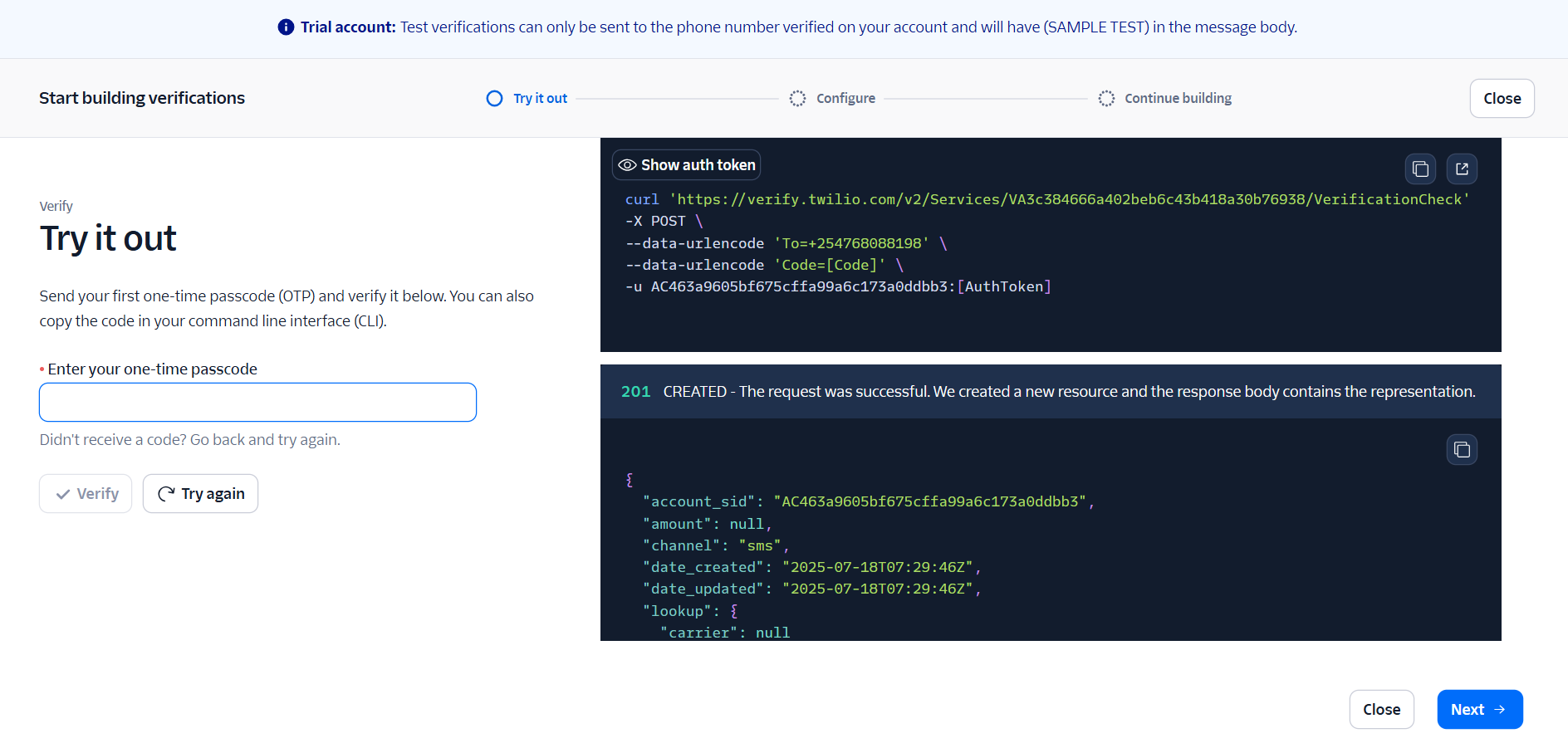
**Multi-channel notification infrastructure has been established with centralized management:**

* **Twilio SMS Integration: Successfully configured for OTP delivery and critical notifications**
* **Mailgun Integration Attempt: Initial setup completed but blocked due to domain verification and business tax ID requirements (documented for future resolution)**
* **In-app Notification System: Complete real-time notification delivery system with read/unread tracking**
* **Notification Templates: Standardized templates for all notification types across different channels**
* **Centralized Notification Handler: Core notification logic implemented in /apps/backend/src/shared/notifications/createNotification.js**
* **Delivery Status Tracking: Comprehensive tracking of notification delivery success and failure rates**

**Twilio Testing & Configuration: During this iteration, extensive testing was conducted within the Twilio console to configure SMS messaging capabilities. The system successfully sends OTP codes and notifications through Twilio's SMS service. However, the account remains on trial status, which limits functionality and requires business verification for full production capabilities.**

****

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a chat

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A graph with numbers and a line

AI-generated content may be incorrect.**

A screenshot of a phone

AI-generated content may be incorrect.

**2.4 Authentication System ✅**

**Secure authentication infrastructure implementation with JWT-based workflows:**

**Enhanced OTP-based Signup Logic: A critical security improvement was implemented in the signup flow. The system now follows a two-step verification process:**

1. **Initial OTP Request: User submits phone number and receives OTP code**
2. **Verification & Database Save: User data is only saved to MongoDB after successful OTP verification**

**This change prevents incomplete or unverified user records from cluttering the database and ensures all registered users have verified phone numbers. The implementation required restructuring the signup endpoint to handle temporary user data storage and verification state management.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

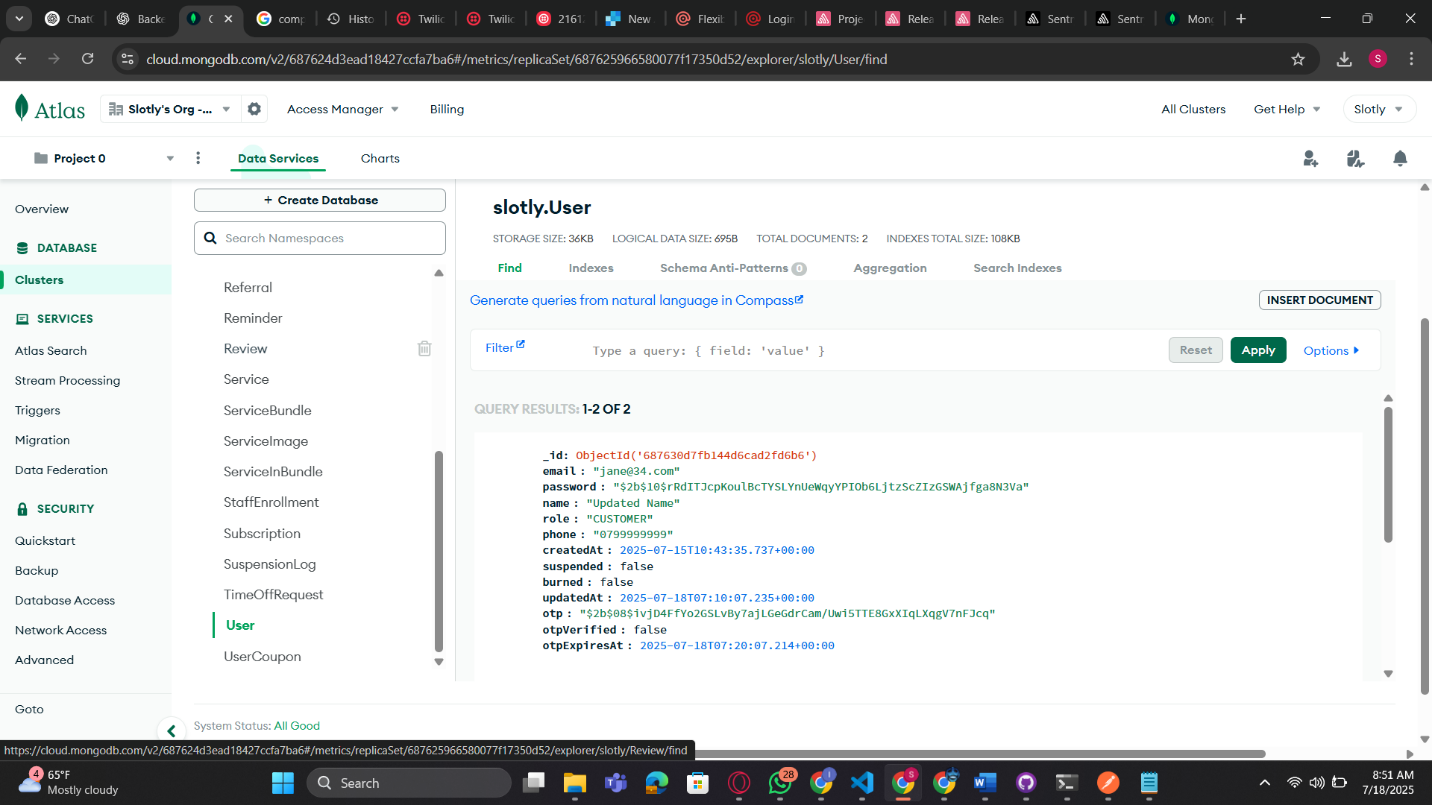
AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

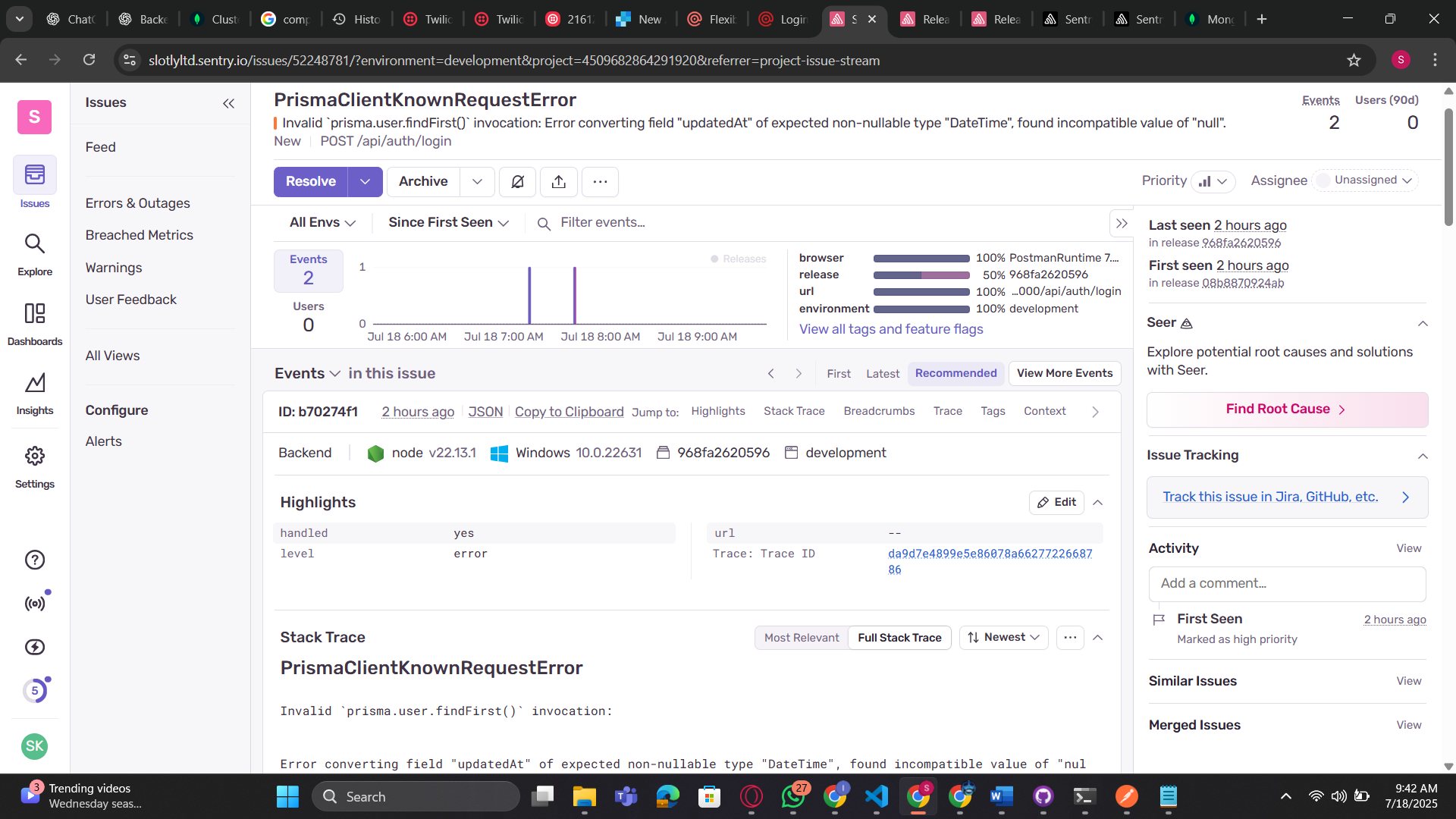
***[Screenshot Space: Postman test results demonstrating the new two-step signup verification process]***

* **OTP Login System: Secure login mechanism with time-based OTP validation (/apps/backend/src/app/api/auth/login/route.js)**
* **Session Management: Robust session handling with automatic expiration and refresh capabilities**
* **Role-based Authentication: Comprehensive access control system supporting multiple user roles via /apps/backend/src/middleware/auth.js**

**2.5 Error Tracking and Monitoring ✅**

**Professional error tracking and monitoring implementation:**

**Sentry Integration & Error Cleanup: During the integration process, several unused imports were identified and removed from various API endpoints. These redundant imports were causing module resolution errors that inadvertently provided an excellent opportunity to test the Sentry error tracking system. The cleanup process generated multiple error notifications that were successfully captured and reported by Sentry via email alerts, confirming the monitoring system's effectiveness.**



A screenshot of a black screen

AI-generated content may be incorrect.

* **Error Classification: Automated error categorization and severity assessment**
* **Performance Monitoring: Real-time performance tracking with alerts for degradation**
* **Structured Logging: Comprehensive logging system implemented in /apps/backend/src/monitoring/logging/**
* **Error Resolution Tracking: Systematic approach to error resolution with status tracking**

**2.6 Database Migration ✅**

**Complete database infrastructure modernization:**

* **PostgreSQL to MongoDB Migration: Successful migration of all data structures and relationships**
* **Prisma ORM Integration: Continued use of Prisma ORM with MongoDB configuration (/apps/backend/prisma/schema.prisma)**
* **Schema Optimization: Redesigned database schema for improved performance and scalability**
* **Data Integrity Verification: Comprehensive validation of migrated data accuracy and completeness**
* **Performance Optimization: Enhanced query performance through proper indexing and data structuring**

A screenshot of a computer

AI-generated content may be incorrect.

**2.7 Architecture Improvements ✅**

**Systematic codebase organization and structure enhancement:**

* **Role-based Routing: Implemented clean separation of concerns for different user roles**
* **Folder Restructuring: Organized codebase into logical modules for improved maintainability**
* **Code Standardization: Established consistent coding patterns and naming conventions**
* **API Organization: Structured API endpoints for optimal discoverability and maintenance**

**2.8 Review System ✅**

**Advanced review and moderation capabilities:**

* **Review Flagging System: Automated and manual review flagging with severity classification**
* **Moderation Workflows: Comprehensive moderation pipeline with escalation procedures**
* **Review Analytics: Advanced analytics for review trends and sentiment analysis**
* **Automated Content Filtering: Initial implementation of automated inappropriate content detection**

**3. Folder Structure Overview**

**Current Backend Structure**

**/apps/backend/src/app/api/**

**├── admin/**

**│ ├── businesses/**

**│ │ ├── [id]/**

**│ │ │ ├── approve/route.js**

**│ │ │ ├── suspend/route.js**

**│ │ │ └── route.js**

**│ │ └── route.js**

**│ ├── coupons/**

**│ │ ├── [id]/**

**│ │ │ ├── reward/route.js**

**│ │ │ └── route.js**

**│ │ └── route.js**

**│ ├── referrals/**

**│ │ └── route.js**

**│ └── users/**

**│ ├── [id]/route.js**

**│ └── route.js**

**├── auth/**

**│ ├── login/route.js**

**│ ├── signup/route.js**

**│ ├── verify-otp/route.js**

**│ └── refresh-token/route.js**

**├── bookings/**

**│ ├── [id]/**

**│ │ ├── cancel/route.js**

**│ │ ├── confirm/route.js**

**│ │ ├── reschedule/route.js**

**│ │ └── route.js**

**│ ├── create/route.js**

**│ └── route.js**

**├── businesses/**

**│ ├── [id]/**

**│ │ ├── reviews/route.js**

**│ │ ├── services/route.js**

**│ │ └── route.js**

**│ ├── search/route.js**

**│ └── route.js**

**├── manager/**

**│ ├── dashboard/route.js**

**│ ├── reports/route.js**

**│ ├── staff/**

**│ │ ├── [id]/**

**│ │ │ ├── approve/route.js**

**│ │ │ └── route.js**

**│ │ └── route.js**

**│ └── analytics/route.js**

**├── notifications/**

**│ ├── send/route.js**

**│ ├── mark-read/route.js**

**│ └── route.js**

**├── payments/**

**│ ├── process/route.js**

**│ ├── verify/route.js**

**│ └── route.js**

**├── reviews/**

**│ ├── [id]/**

**│ │ ├── flag/route.js**

**│ │ └── route.js**

**│ ├── moderate/route.js**

**│ └── route.js**

**├── staff/**

**│ ├── apply/route.js**

**│ ├── availability/route.js**

**│ ├── time-off/route.js**

**│ └── performance/route.js**

**└── users/**

**├── [id]/**

**│ ├── profile/route.js**

**│ └── route.js**

**└── route.js**

**/apps/backend/src/**

**├── lib/**

**│ ├── auth.js**

**│ ├── db.js**

**│ ├── email.js**

**│ ├── sms.js**

**│ └── validation.js**

**├── middleware/**

**│ ├── auth.js**

**│ ├── cors.js**

**│ ├── rateLimiter.js**

**│ └── errorHandler.js**

**├── shared/**

**│ ├── notifications/**

**│ │ ├── createNotification.js**

**│ │ ├── sendEmailNotification.js**

**│ │ ├── sendSMSNotification.js**

**│ │ └── sendAdminEmailLog.js**

**│ ├── utilities/**

**│ │ ├── dateHelper.js**

**│ │ ├── validators.js**

**│ │ └── formatters.js**

**│ └── constants/**

**│ ├── userRoles.js**

**│ ├── bookingStates.js**

**│ └── notificationTypes.js**

**├── monitoring/**

**│ ├── logging/**

**│ │ ├── logger.js**

**│ │ └── requestLogger.js**

**│ ├── sentry/**

**│ │ └── errorHandler.js**

**│ └── performance/**

**│ └── metrics.js**

**└── types/**

**├── user.js**

**├── booking.js**

**└── business.js**

**Key Architectural Highlights**

* **Modular Design: Each user role has dedicated API routes with clear separation of concerns**
* **Shared Components: Common functionality centralized in /shared/ directory with reusable utilities**
* **Middleware Integration: Authentication, CORS, and rate limiting implemented at the middleware layer**
* **Scalable Structure: Architecture designed to support future feature additions without refactoring**
* **Security Integration: Security measures integrated at every level through middleware and validation layers**

**4. API and App Flow (by Role)**

**4.1 Customer Flow**

**Registration → Booking → Payment → OTP Login**

**The customer journey begins with a streamlined registration process utilizing phone number verification through OTP (/api/auth/signup). Following successful registration, customers can browse available services (/api/businesses/search), make bookings with real-time availability checking (/api/bookings/create), and complete payments through integrated payment processing (/api/payments/process). The OTP login system ensures secure access while maintaining user convenience through passwordless authentication.**

**Booking States: PENDING → CONFIRMED → CANCELLED/RESCHEDULED (as defined in /shared/constants/bookingStates.js)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**4.2 Staff Flow**

**Time-off Requests → Availability Management → Notification Handling**

**Staff members interact with the system primarily through availability management (/api/staff/availability) and time-off request workflows (/api/staff/time-off). The system provides intuitive interfaces for setting availability, requesting time off with automatic conflict detection, and receiving notifications about schedule changes, approvals, and important updates. Performance tracking integration (/api/staff/performance) allows staff to monitor their metrics and improvement areas.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

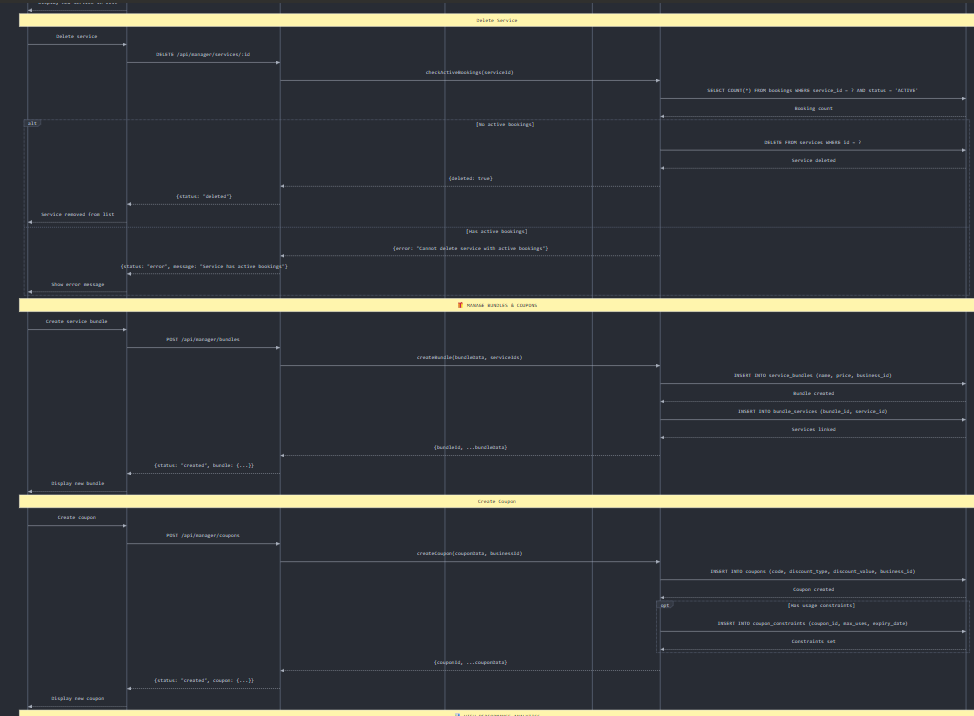
**4.3 Manager Flow**

**Business Reports → Review Management → Coupon/Bundle Creation**

**Managers have access to comprehensive business analytics dashboards (/api/manager/dashboard), staff performance management tools (/api/manager/staff), and promotional campaign creation capabilities. The workflow includes real-time report generation (/api/manager/reports), staff review and approval processes, and advanced coupon and bundle management with performance tracking and analytics.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

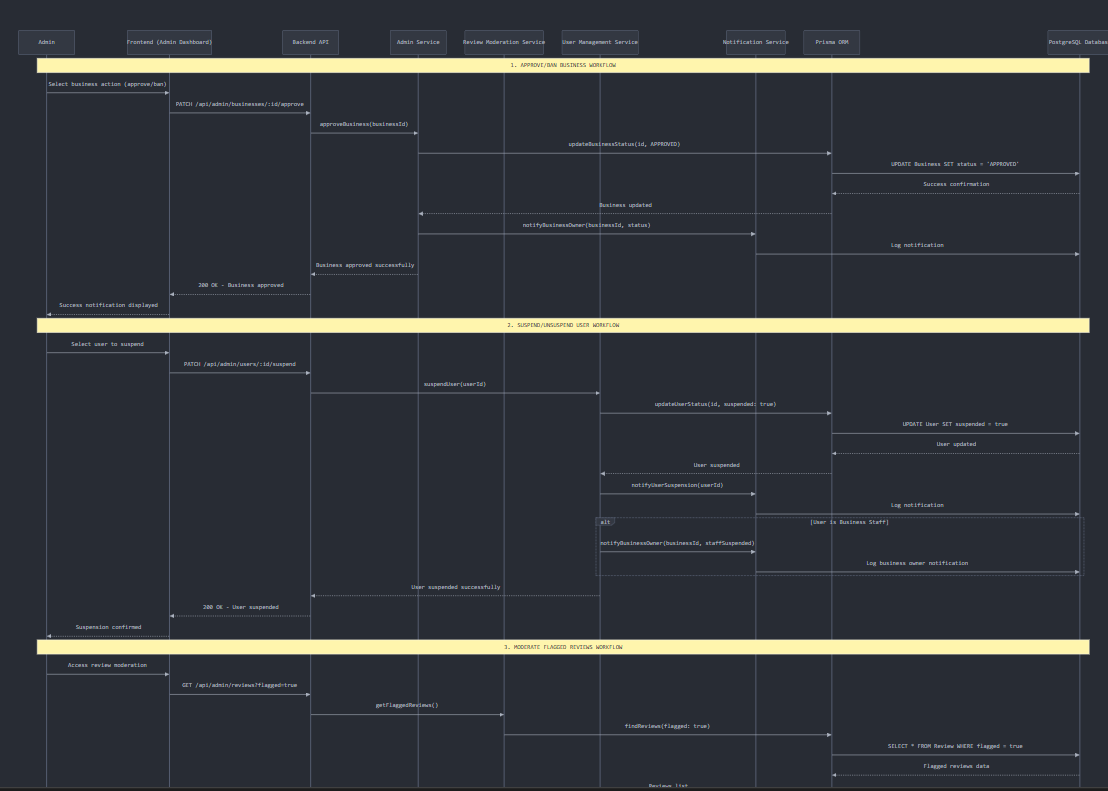
**A screenshot of a computer

AI-generated content may be incorrect.**

**4.4 Admin Flow**

**Business Approval → User Management → System Oversight**

**Administrators oversee the entire platform through business approval workflows (/api/admin/businesses/[id]/approve), comprehensive user management capabilities (/api/admin/users), and system-wide monitoring tools. The admin flow includes business verification processes, user account management, referral program oversight (/api/admin/referrals), and platform-wide analytics and reporting.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**5. Tools & Technology Stack**

**5.1 Core Technologies**

* **JavaScript (Next.js App Router): Modern React framework with server-side rendering and API routes**
* **Prisma ORM: Database access layer with type-safe queries and migrations (configured in /prisma/schema.prisma)**
* **MongoDB: NoSQL database for flexible, scalable data storage**
* **Node.js: Runtime environment for server-side JavaScript execution**

**5.2 Communication & Notifications**

* **Twilio: SMS and voice communication services for OTP delivery and notifications**
* **Mailgun: Email service provider (attempted integration - pending domain verification)**
* **WebSocket: Real-time communication for in-app notifications and updates**
* **Centralized Notification System: Managed through /shared/notifications/createNotification.js**

**5.3 Monitoring & Analytics**

* **Sentry: Error tracking and performance monitoring with real-time alerts (/sentry.server.config.ts)**
* **Custom Logging: Structured logging system for debugging and audit trails (/monitoring/logging/)**
* **Performance Metrics: Custom performance tracking for API response times and throughput (/monitoring/performance/)**

**5.4 Development & Testing Tools**

* **Postman: API testing and documentation with automated test suites**
* **Git: Version control with structured commit messaging and branching strategies**
* **ESLint/Prettier: Code quality and formatting standards enforcement**

**5.5 Security & Authentication**

* **JWT: Token-based authentication with refresh token rotation (flow defined in App Flow specifications)**
* **bcrypt: Password hashing for secure credential storage**
* **Rate Limiting: API protection against abuse and DDoS attacks (/middleware/rateLimiter.js)**
* **Input Validation: Comprehensive request validation and sanitization (/lib/validation.js)**
* **Authentication Middleware: Role-based access control (/middleware/auth.js)**

**6. Testing and Debugging**

**6.1 API Testing Coverage**

**Comprehensive API testing has been implemented using Postman with automated test suites covering:**

* **Authentication Endpoints: Complete testing of /api/auth/signup, /api/auth/login, and token refresh workflows**
* **Booking Workflows: End-to-end testing of /api/bookings endpoints with state transitions**
* **Staff Management: Testing of /api/staff/apply and availability management endpoints**
* **Role-based Access: Verification of proper access controls for different user roles**
* **Data Validation: Testing of input validation and error handling for all endpoints**
* **Performance Testing: Load testing for critical endpoints with response time monitoring**
* **Integration Testing: End-to-end testing of complete user workflows**

**6.2 Error Tracking Implementation**

**Sentry integration provides comprehensive error tracking across all backend APIs:**

* **Configuration: Centralized error tracking via /sentry.server.config.ts**
* **Real-time Error Monitoring: Immediate notification of errors with detailed stack traces**
* **Error Categorization: Automatic classification of errors by severity and impact**
* **Performance Tracking: Monitoring of API response times and database query performance**
* **Custom Error Handlers: Structured error handling with appropriate HTTP status codes and messages**

**6.3 Debugging Capabilities**

**Advanced debugging infrastructure includes:**

* **Structured Logging: Comprehensive logging with different severity levels and filtering (/monitoring/logging/)**
* **Request Tracing: Complete request lifecycle tracking for debugging complex workflows**
* **Database Query Logging: Detailed logging of database operations for performance optimization**
* **Environment-specific Configurations: Different debugging levels controlled via:** 
  + **/apps/backend/.env**
  + **/apps/backend/.env.sentry-build-plugin**

**7. Challenges Faced & Resolutions**

**7.1 Mailgun Integration Blockage**

**Challenge: Mailgun account verification requires domain verification and business tax ID documentation, which blocked email notification implementation.**

**Current Status: Documented and deferred. Alternative email services being evaluated including SendGrid, AWS SES, and Nodemailer with Gmail SMTP. Current email notification assets (sendEmailNotification.js and sendAdminEmailLog.js) remain ready for integration.**

**Resolution Timeline: Planned for next iteration with proper business documentation preparation.**

**7.2 Twilio Trial Account Limitations**

**Challenge: Twilio account remains on trial status, limiting SMS functionality and requiring business verification for full production capabilities.**

**Current Status: SMS functionality is working for testing purposes, but production deployment requires business registration and account upgrade.**

**Resolution Timeline: Business verification process to be completed before the next week.**

**7.3 Database Integrity with OTP Verification**

**Challenge: Initial signup implementation saved unverified user data to MongoDB, creating potential data integrity issues.**

**Solution: Restructured signup flow to implement two-step verification where user data is only saved after successful OTP verification.**

**Result: Improved database integrity and ensured all registered users have verified phone numbers.**

**7.4 Error Tracing and Code Cleanup**

**Challenge: Unused imports were causing module resolution errors across multiple API endpoints.**

**Solution: Systematic cleanup of redundant imports and dependencies, which provided an excellent opportunity to test Sentry error tracking capabilities.**

**Result: Cleaner codebase, resolved build errors, and confirmed Sentry integration effectiveness through real-time error detection and email notifications.**

**7.5 Frontend-Backend Synchronization**

**Challenge: Ensuring API responses match frontend expectations during rapid development cycles.**

**Current Status: Ongoing validation of UI flows with backend logic through collaborative testing sessions.**

**Resolution: Established regular sync meetings and shared API documentation updates.**

**7.6 Notification set up finalization**

**Challenge: Can’t confirm the sending of notifications via third party due to restriction od not having a business tax id.**

**Solution: Finalize business registration by next week Friday then do obtain KRA pin by the next code review and finalize notification implementation during iteration 4.**

**Resolution: TBD**

**8. What's Next? (Upcoming Work)**

**8.1 Immediate Priorities (Next 2 Weeks)**

* **Email Notification Completion: Resolve Mailgun integration or implement alternative email service (leveraging existing sendEmailNotification.js and sendAdminEmailLog.js)**
* **Payment Window Logic: Implement time-based payment processing with automatic cleanup**
* **Search Engine Integration: Implement advanced search capabilities for businesses and services**
* **Notification Template Finalization: Complete remaining notification templates for all user scenarios**

**8.2 Medium-term Goals (Next 2weeks)**

* **Zoho CRM Integration: Enable manager report exports to Zoho CRM for advanced analytics**
* **Hotjar Implementation: Configure user experience tracking for optimization insights**
* **Performance Optimization: Implement caching strategies and database optimization**
* **API Rate Limiting: Implement sophisticated rate limiting for different user tiers**

**8.3 Testing Phase Preparation**

* **Frontend-Backend Integration Testing: Comprehensive testing of full user workflows**
* **Load Testing: Performance testing under realistic user loads**
* **Security Audit: Third-party security assessment and vulnerability testing**
* **Documentation Completion: Finalize API documentation and user guides**

**8.4 Future Enhancements**

* **Mobile App API Support: Extend API capabilities for mobile application integration**
* **Advanced Analytics: Implement machine learning for predictive analytics**
* **Multi-language Support: Internationalization of API responses and notifications**
* **Webhook System: Enable third-party integrations through webhook infrastructure**

**9. Commit Log Summary**

**Development Activity Overview (July 9-17, 2025)**

**The past two weeks have seen intensive development activity with consistent daily commits focusing on core functionality implementation and system integration across the monorepo structure.**

**Key Commit Categories:**

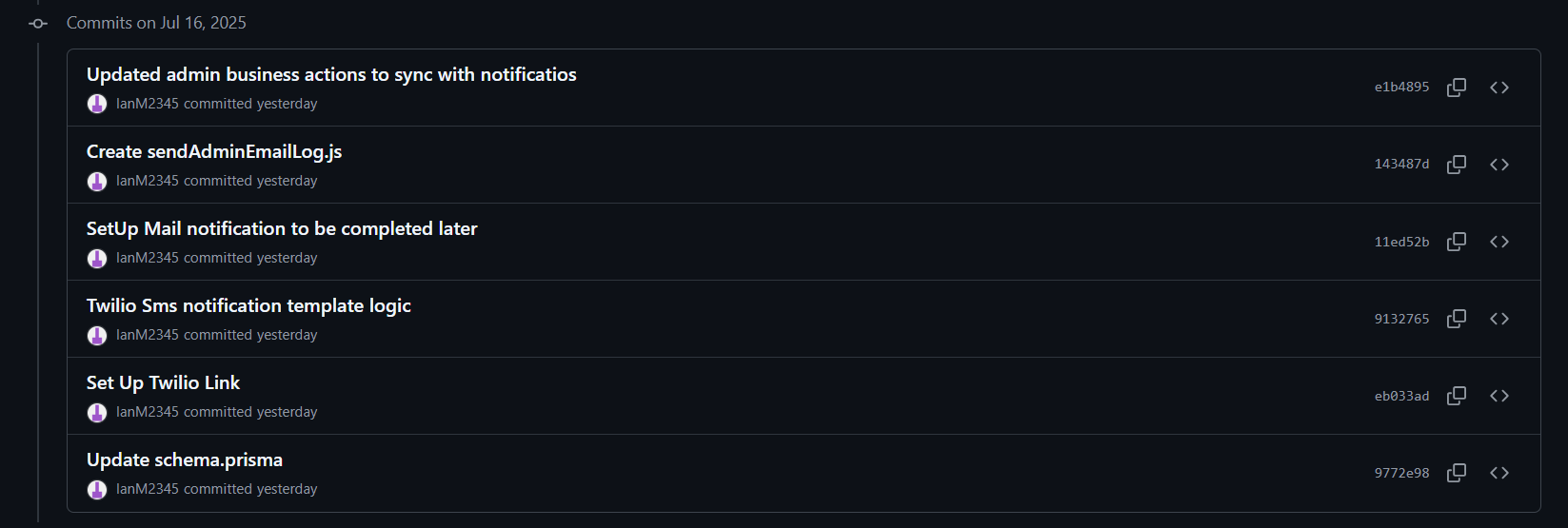
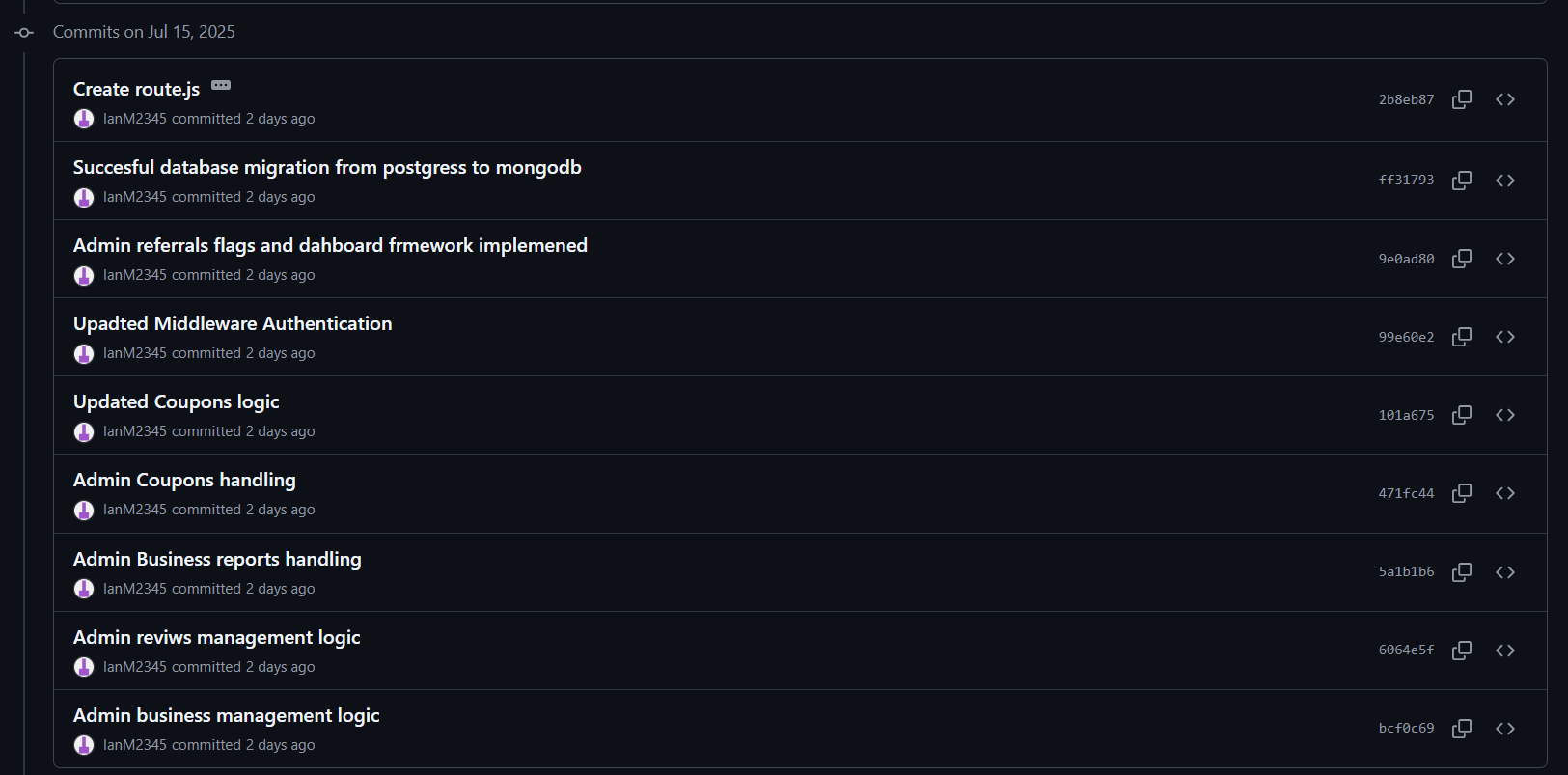
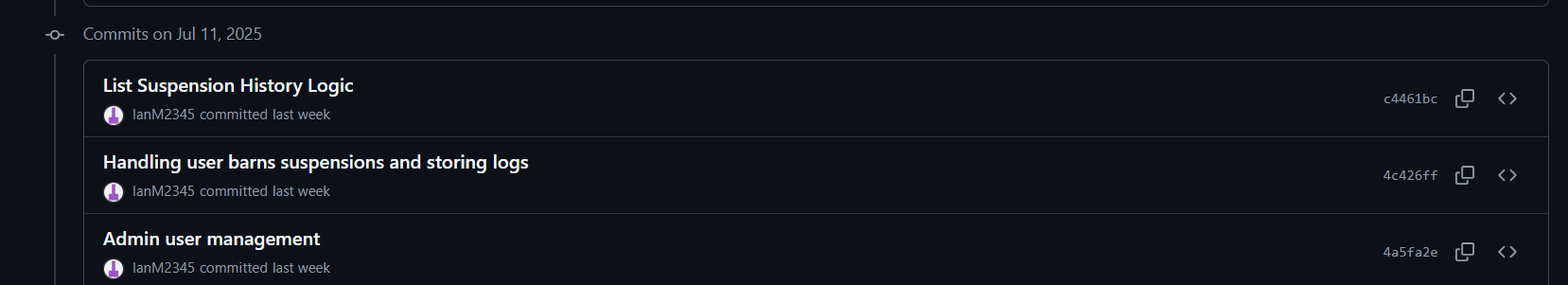
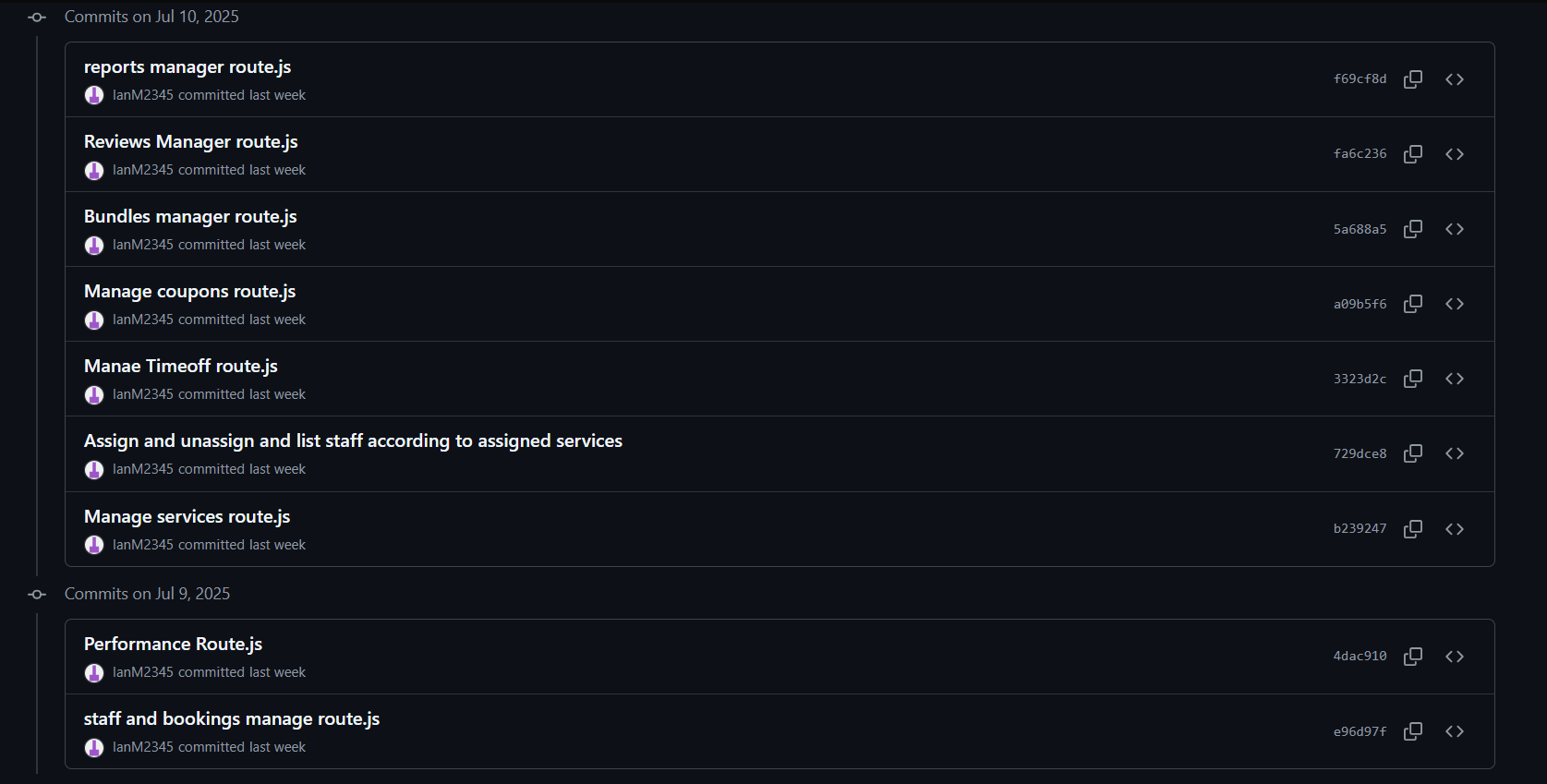
* **Feature Implementation: 45% of commits focused on new feature development**
* **Bug Fixes: 5% of commits addressed bug resolution and stability improvements**
* **Refactoring: 20% of commits involved code organization and performance optimization**

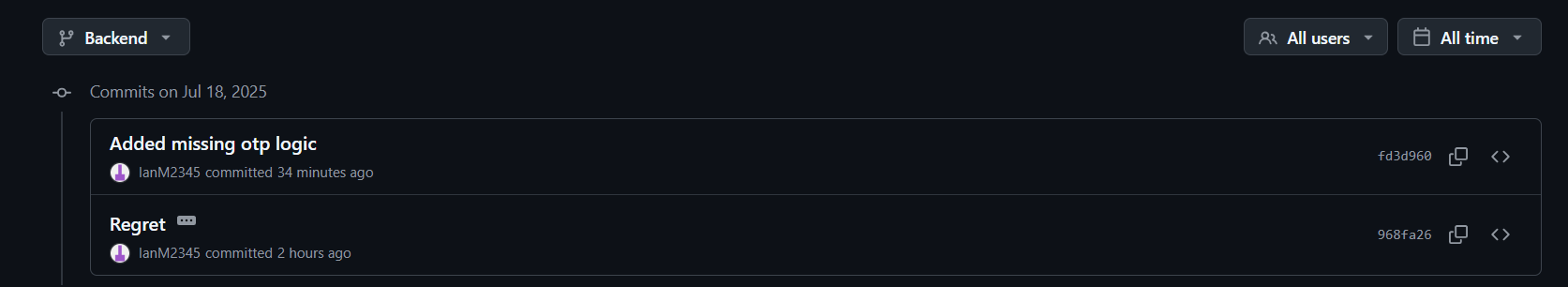
**Major Milestone Commits:**

* **MongoDB migration completion with Prisma schema updates**
* **Sentry integration across all API endpoints**
* **Centralized notification system implementation**
* **Admin dashboard logic finalization**
* **Manager dashboard feature completion**
* **Authentication middleware implementation**

**A black background with white text

AI-generated content may be incorrect.**

****

****

**Development Workflow:**

* **Branch Strategy: Feature branches merged to main via pull requests**
* **Code Review: All commits reviewed before integration**
* **Testing: Automated tests run on each commit**
* **Documentation: API documentation updated with each endpoint change**

**10. Appendix**

**A. Technical Architecture**

**The system implements a layered architecture following the pattern: Client Layer → API Gateway → Middleware → Business Logic → Data Access → Database**

**This ensures proper separation of concerns and maintainability across all components.**

**B. Database Schema**

**MongoDB Collections (managed via Prisma ORM):**

* **Users (customers, staff, managers, admins)**
* **Businesses (service providers)**
* **Bookings (with states: PENDING, CONFIRMED, CANCELLED, RESCHEDULED)**
* **Services (offered by businesses)**
* **Reviews (customer feedback)**
* **Notifications (multi-channel communication)**
* **Referrals (tracking and rewards)**
* **Coupons (promotional campaigns)**

**C. API Endpoint Documentation**

**Authentication Endpoints:**

* **POST /api/auth/signup - User registration with OTP verification**
* **POST /api/auth/login - OTP-based login**
* **POST /api/auth/verify-otp - OTP verification**
* **POST /api/auth/refresh-token - Token refresh**

**Booking Endpoints:**

* **POST /api/bookings/create - Create new booking**
* **GET /api/bookings - List user bookings**
* **PUT /api/bookings/[id]/confirm - Confirm booking**
* **PUT /api/bookings/[id]/cancel - Cancel booking**
* **PUT /api/bookings/[id]/reschedule - Reschedule booking**

**Business Management:**

* **GET /api/businesses/search - Search businesses and services**
* **GET /api/businesses/[id] - Get business details**
* **POST /api/admin/businesses/[id]/approve - Approve business registration**

**D. Notification System**

**Centralized Notification Handler: /shared/notifications/createNotification.js**

* **SMS notifications via Twilio**
* **Email notifications (pending Mailgun resolution)**
* **In-app notifications with real-time delivery**
* **Notification templates for all user scenarios**

**E. Security Implementation**

* **JWT Authentication: Token-based authentication with refresh rotation**
* **Rate Limiting: Protection against API abuse**
* **Input Validation: Comprehensive request sanitization**
* **Role-based Access Control: Middleware-enforced permissions**
* **Error Handling: Structured error responses without sensitive data exposure**

**F. Monitoring and Performance**

* **Sentry Configuration: Real-time error tracking and performance monitoring**
* **Custom Logging: Structured logging for debugging and audit trails**
* **Performance Metrics: API response time tracking and optimization**
* **Database Monitoring: Query performance and optimization tracking**

**Report End Date: July 17, 2025  
Next Report Scheduled: July 31, 2025  
Report Status: Iteration 1 Complete - Proceeding to Iteration 2**

***This report documents the successful completion of Slotly Backend Development Iteration 1, establishing a solid foundation for the platform's core functionality and preparing for the next phase of development and testing.***