Development Blueprint: Space Billiards Automated Management System

Date: June 3, 2025

Overall Vision: To evolve the existing PWA into a fully automated, AI-driven billiards bar management system, enhancing operational efficiency, customer experience, and revenue opportunities. The `FunctionsDashboard` will serve as a new, central hub for advanced operational modules.

---

1. Objectives & Scope

- **Primary Goal:** Create a seamless, real-time, AI-driven system.
  - **Phase 1 (Staff-Facing):** Automate internal operations including advanced table management, device control via Home Assistant, order/status lookups (Square API), staff scheduling, notifications, and access to new functional modules via the `FunctionsDashboard`.
  -
  - **Phase 2 (Customer-Facing):** Introduce customer waitlist management, virtual server interactions, AI-powered menu recommendations, song requests, and feedback mechanisms.
  -
- **Key Benefits:**
  - **Efficiency/Staff Productivity:** Automated table assignments, call-server alerts, API-driven order/status lookups, push notifications, streamlined inventory and task management.
  -
  - **Customer Experience:** QR-code-driven waitlist, virtual menu with AI recommendations, song queue integration, SMS updates, post-session feedback.
  -
  - **Revenue Opportunities:** Upsells via AI-recommended menu items, streamlined service, upsell of extensions, promotional automations.
  -
  - **Scalability & Maintainability:** Modular architecture (Next.js PWA + Supabase + Home Assistant + AI Agent).
  -

---

2. System Architecture & Tech Stack

(This section remains largely the same as your "Blue Print First Draft", with clarifications for the existing PWA context)

- **Frontend Web App (PWA):**
  - Framework: Next.js (React) (as per billiards-timer (31)/package.json, billiards-timer (31)/app/layout.tsx)
  - Styling: Tailwind CSS (as per billiards-timer (31)/tailwind.config.ts, billiards-timer (31)/app/globals.css)
  - Service Worker (PWA): (as per billiards-timer (31)/public/sw.js, billiards-timer (31)/public/manifest.json, billiards-timer (31)/components/pwa-init.tsx)
  - Auth: Supabase Auth via custom login flow (as per billiards-timer (31)/app/api/auth/login/route.ts, billiards-timer (31)/contexts/auth-context.tsx, billiards-timer (31)/services/supabase-auth-service.ts).
- **Backend / Database:**
  - Supabase (Postgres + Realtime + Auth + RLS) (as per billiards-timer (31)/lib/supabase/client.ts, billiards-timer (31)/scripts/schema_core.sql, billiards-timer (31)/scripts/schema_staff.sql, billiards-timer (31)/scripts/schema_menu.sql).
- **Home Automation Hub:** Home Assistant (as per blueprint)
-
- **Notifications:**
  - Staff: Push Notifications (using VAPID keys as seen in billiards-timer (31)/app/api/notifications/send/route.ts and related files).
  - Customers: Twilio SMS (as per blueprint).
  -
- **External APIs / Integrations:**
  - Square API (Order lookups, bill views, appointments).
  -
  - Google Calendar & Gmail API (Staff schedules, reminders).
  -
  - OpenWeatherMap API, Sports API (as per blueprint).
  -
- **AI Agent:**
  - Hosted alongside Next.js server (as per billiards-timer (31)/app/api/ai/chat/route.ts).
  -
  - Model: Grok (from XAI, as per billiards-timer (31)/app/api/ai/chat/route.ts).
  - Responsibilities: Waitlist logic, table assignment, menu recommendations, simple customer chat, song request queuing (if implemented).
  -

---

3. Database Schema & Row-Level Security

(Refers to existing schema files: `billiards-timer (31)/scripts/schema_core.sql`, `billiards-timer (31)/scripts/schema_staff.sql`, `billiards-timer (31)/scripts/schema_menu.sql`, `billiards-timer (31)/scripts/push_notifications_schema.sql`. The blueprint should introduce new tables for Inventory, Tasks, and Internal Communications as needed by the Functions Dashboard modules.)

Conceptual Table Schema Visualizer (Textual Representation):

- **Core (`schema_core.sql`):**
  - `billiard_tables`: id, name, is_active, start_time, remaining_time, initial_time, guest_count, server_id (FK to staff_members), group_id, has_notes, note_id (FK to note_templates), note_text, created_at, updated_at, updated_by_admin, updated_by.
  - `session_logs`: id, table_id, table_name, action, timestamp, details, created_at.
  - `system_settings`: id, day_started, group_counter, day_start_time, default_session_time, warning_threshold, critical_threshold, last_updated.
  - `servers`: (Consider renaming or clarifying if this is different from `staff_members` with role 'server'. Based on `schema_core.sql`, this seems to be a simpler list, while `staff_members` is more detailed for auth. For this blueprint, we'll assume `staff_members` is the primary source for staff, and `servers` might be a denormalized list or a specific role table if intended differently). Current `schema_core.sql` has `servers (id UUID, name TEXT, enabled BOOLEAN...)`.
  - `note_templates`: id, text, created_at, updated_at.
- **Staff (`schema_staff.sql`):**
  - `staff_roles`: id, role_name (admin, manager, server, etc.), description.
  - `staff_members`: id (UUID, links to auth.users.id), first_name, display_name, email, phone, native_language, pin_code, role (FK to staff_roles.role_name), auth_id.
  - `staff_permissions`: id, staff_id (FK to staff_members.id), (various boolean permission flags like can_manage_users, can_start_session).
- **Menu (`schema_menu.sql`):**
  - `menu_items`: id, name, category, price, description, popularity, pairings, image_url, is_available.
  - `combos`: id, name, price_weekday, price_weekend, billiard_time_minutes, max_guests, includes_billiard_time, image_url, description.
  - `sales_data`: id, item_id, item_type, date, quantity, revenue, time_of_day, day_of_week.

- **Push Notifications (push_notifications_schema.sql):**
  - push_subscriptions: user_id (FK to auth.users), endpoint, p256dh, auth.
  - notification_logs: title, body, user_id, table_id, sent_at.
- **New Tables for Functions Dashboard:**
  - inventory_items: id, name, category, current_stock, reorder_threshold, supplier_id (FK), unit_cost, expiry_date (nullable).
  - suppliers: id, name, contact_info.
  - purchase_orders: id, supplier_id (FK), order_date, expected_delivery_date, status, total_cost.
  - po_items: id, po_id (FK), item_id (FK), quantity, unit_price.
  - staff_schedules: id, staff_id (FK), start_time, end_time, shift_notes.
  - time_clock_entries: id, staff_id (FK), clock_in_time, clock_out_time.
  - orders (for Order Management, if not fully relying on Square): id, table_id (FK), staff_id (FK), order_time, status, total_amount, square_order_id (nullable).
  - order_items: id, order_id (FK), menu_item_id (FK to menu_items/combos), quantity, price_at_time_of_order.
  - reservations: id, customer_name, phone_number, email, table_id (FK), reservation_time, duration, guest_count, status (pending, confirmed, cancelled), square_appointment_id (nullable), google_calendar_event_id (nullable).
  - events: id, event_name, start_time, end_time, description, associated_tables (JSONB or text array).
  - blocked_tables: id, table_id (FK), reason, start_time, end_time.
  - internal_messages: id, thread_id, sender_id (FK to staff_members), receiver_id (FK to staff_members, nullable for group/announcement), content, timestamp, read_status.
  - message_threads: id, participants (JSONB array of staff_ids), last_message_timestamp.
  - announcements: id, author_id (FK to staff_members), title, content, publish_date, target_roles (JSONB array).
  - tasks: id, title, description, due_date, priority, status (todo, in_progress, completed), assigned_to_staff_id (FK), assigned_to_role (FK), recurring_rule (e.g., cron string), created_by_staff_id (FK).
  - task_checklists: id, task_id (FK), item_text, is_completed.

RLS Policies: (Refer to existing RLS policies in schema files and extend for new tables, generally restricting access based on staff roles and ownership, with admins having broader access).

4. Frontend (Next.js PWA) Structure & Components

(Incorporating the Functions Dashboard and its modules)

Conceptual App Structure Visualizer (Textual Representation - Key Areas):

- `/app`
  - `layout.tsx` (Root layout)
  - `page.tsx` (Main dashboard page, likely `BilliardsTimerDashboard`)
  - `/admin`
    - `/users/page.tsx` (User management UI, links to SupabaseUserManagement)
  - `/api` (Backend routes)
    - `/ai/chat/route.ts`
    - `/auth/login/route.ts`
    - `/notifications/*` (public-key, send, subscribe, unsubscribe routes)
    - `/staff/*` (members, permissions, roles routes)/route.ts, file:billiards-timer (31)/app/api/staff/members/route.ts, file:billiards-timer (31)/app/api/staff/permissions/[id]/route.ts, file:billiards-timer (31)/app/api/staff/permissions/route.ts, file:billiards-timer (31)/app/api/staff/roles/route.ts]
    - `/square/*` (New: orders, bills, appointments)
    - `/google/*` (New: calendar/events, gmail/send)
    - `/twilio/*` (New: send-sms)
    - (New API routes for inventory, tasks, internal_communications as needed)
  - `/diagnostics/page.tsx`
  - `/debug/page.tsx`
- `/components`
  - `/system`
    - `BilliardsTimerDashboard.tsx` (Main operational view)
    - `FunctionsDashboard.tsx` (New central hub for modules)
      - `/Inventory` (New Folder)
        - `InventoryDashboard.tsx`
        - `StockLevelTable.tsx`
        - `LowStockAlerts.tsx`
        - `AddStockItemForm.tsx`
      - `/Staff` (New Folder)
        - `StaffScheduleView.tsx`
        - `StaffPerformanceDashboard.tsx`

                - (User/Role Management may link to existing `/admin/users` or `SupabaseUserManagement` component)
            - `/Reports` (New Folder)
                - `ReportsDashboard.tsx`
                - `OccupancyHeatmap.tsx`
                - `SalesSummaryChart.tsx`
            - `/Orders` (New Folder)
                - `OrderEntryInterface.tsx`
                - `KDSView.tsx`
                - `OpenOrdersList.tsx`
            - `/Reservations` (New Folder)
                - `ReservationsCalendar.tsx`
                - `NewReservationForm.tsx`
                - `TableBlockingTool.tsx`
            - `/Settings` (New Folder - for advanced settings beyond main settings dialog)
                - `SystemParametersForm.tsx`
                - `PricingRulesEditor.tsx`
                - `ApiIntegrationSettings.tsx`
            - `/Communications` (New Folder)
                - `MessagingInterface.tsx`
                - `AnnouncementsBoard.tsx`
            - `/Tasks` (New Folder)
                - `TaskManagerDashboard.tsx`
                - `CreateTaskForm.tsx`
                - `TaskCard.tsx`
            - `/AI` (Utilizes existing AI components)
                - (Entry point to `AiAnalyticsDashboard` or a more interactive AI chat if needed)
        - `AiAnalyticsDashboard.tsx` (Possibly accessed via Functions Dashboard or PullUp Panel)
        - `PullUpInsightsPanel.tsx`
    - `/tables` (Table cards, dialogs)
    - `/dialogs` (Settings, Confirm, Day Report, AI Assistant etc.)
    - `/admin` (User management components)
    - `/auth` (Login components)
    - `/mobile` (Specific mobile view components)
    - `/ui` (Shadcn UI components)
- `/services` (Data interaction logic)
    - `ai-service.ts`

    - `menu-data-service.ts`
    - `supabase-auth-service.ts`
    - (And other Supabase interaction services)
- `/contexts` (Global state management)
    - `AuthContext.tsx`
- `/hooks` (Custom React hooks)
    - `useAiAssistant.ts`
    - `useSupabaseData.ts`
    - `useTableTimer.ts`
    - `useMobile.ts`
- `/lib` (Utility functions, Supabase client setup)
    - `utils.ts`
    - `/supabase/client.ts`

Functions Dashboard UI (`billiards-timer (31)/components/system/functions-dashboard.tsx`):

- The main view of this component will be a tabbed interface or a grid of "Function Cards" as described in its current implementation.
- Each card/tab will navigate to or render the respective module's main component (e.g., `InventoryDashboard.tsx`, `StaffScheduleView.tsx`).

---

5. Home Assistant Automation Design

(As per your "Blue Print First Draft"). This section is well-defined and can be implemented based on those details.)

---

6. API Integrations

(As per your "Blue Print First Draft," with emphasis on Square API for Order Management and Square Appointments/Google Calendar for Reservation Management).

---

7. Central AI Agent

(As per your "Blue Print First Draft" and the AI Assistant features in the Functions Dashboard). The `AiService` and `AiAnalyticsDashboard` are key components here.

---

8. Development Steps & Suggested Timeline

(To be adjusted based on the new scope of the Functions Dashboard modules. Each module within the Functions Dashboard can be treated as a sub-project within the overall phases.)

Phase 1 (Staff-Facing MVP + Core Functions Dashboard Modules):

- Weeks 1-2: Infrastructure, Basic Schema, Auth (as per blueprint).
-
- Weeks 3-4: Core Table Management, Realtime Updates, Basic Device Control via HA (as per blueprint).
-
- Weeks 5-8: Develop core functionalities for **Inventory Management**, **Staff Management (Roles & Permissions UI, Scheduling Tool)**, and **Basic Reports & Analytics** from the Functions Dashboard. Integrate basic Square order lookup.
- Weeks 9-10: Develop **Task Manager (Core)** and **Internal Communications (Basic)**. Refine existing integrations (Google Calendar basic sync, FCM for calls).

Phase 1.5 (Expansion of Functions & Automation):

- Weeks 11-14: Enhance **Inventory Management** (low stock alerts, consumption analytics). Flesh out **Staff Performance Metrics**. Develop **Order Management** (KDS/BDS mockups, manual order linking if Square API direct order entry is complex for MVP). Implement **Event & Reservation Management** (manual entry, basic table blocking). Expand **System Settings** UI.
- Weeks 15-16: AI Assistant MVP integration (natural language queries for existing data, menu recommendations based on `MenuDataService`).

Phase 2 (Customer-Facing Features & Advanced Automation):

- Weeks 17-20: Customer Waitlist, Seated Experience (Virtual Server, Interactive Menu read-only from Square, Song Request Form).
- Weeks 21-24: Full Square API integration for orders (if not earlier), Twilio SMS notifications, advanced AI features (proactive insights, automated reporting summaries), advanced Home Assistant automations.
- Weeks 25-28: Advanced **Task Manager** features (recurring, automated generation). Advanced **Inventory** (expiry, reordering drafts). Full **Event & Reservation** with Square/Google Calendar API write access.
- Weeks 29-32: Testing, Deployment, Handover.

(Timeline is indicative and needs adjustment based on team size and parallel work.)

---

9. Hardware & Software Requirements

(As per your "Blue Print First Draft". No changes needed here based on the Functions Dashboard additions, as they are primarily software modules.)

---

10. Staff & Customer Workflows

(To be updated to include interactions with the new Functions Dashboard modules for staff, e.g., "Staff checks inventory levels via Functions Dashboard," "Manager assigns tasks using Task Manager module.")

Example additions to Staff Workflow:

- **Inventory Check:** Staff uses Functions Dashboard -> Inventory Management to view stock or log received goods.
- **Shift Review:** Staff uses Functions Dashboard -> Staff Management to view their schedule.
- **Task Update:** Staff uses Functions Dashboard -> Task Manager to view and update assigned tasks.
