

Big Fam App Home	2
Big Fam Festival App The Ultimate Festival Companion	5
Big Fam Festival App Comprehensive Design Document	7
Big Fam Festival App Comprehensive Jira Tasks Report (Refactored)	12
Big Fam Festival App Outstanding Technology & Architecture Decisions	30
Big Fam Festival App New Developer Onboarding	35
DOCS	39
API Documentation	40
Open API (Swagger) Docs	53
Infrastructure Setup to Deployment & Monitoring	58
UI/UX	62
COMP: Top Nav	63
COMP: Bottom Nav	64
PAGE: My Schedule	67
PAGE: Admin Panel	70
ADMIN: Artist Manager	73
ADMIN: Shift Manager (Admin / Director)	77
ADMIN: Vendor Manager	81
ADMIN: Ticket Manager	85
PAGE: Staff Panel	88
PAGE: Volunteer Panel	92
PAGE: Vendor Panel	95
PAGE: Artist Panel	99
PAGE: Interactive Map & Embedded Navigation	103
PAGE: Events	106
PAGE: Profile Editor	109
PAGE: Friends	113
COMP: Request Medical	116
PAGE: Tickets	121
DECISIONS	125
DECISION TEMPLATE	126
DECISION 1: Cloud Provider Selection	127
DECISION 2: Backend Framework Selection	129
DECISION 3: Frontend Framework Selection	131
DECISION 4: Database Selection	133
DECISION 5: Real-Time Notification	135
DECISION 6: CI/CD and DevOps Pipeline	137
DECISION 7: Embedded Navigation SDK Selection	139
Re-evaluating In-App Navigation vs. External Mapping Solutions	141
DECISION 8: File Upload & Storage Service	143

Big Fam App Home

Welcome to your new space! ☺

Spaces help your team structure, organize, and share work, so every team member has visibility into institutional knowledge and access to the information they need to do their best work.

Recent updates

-  [Big Fam App Home](#)
about 10 hours ago • contributed by [Robert Eriksen](#)

-  [Open API \(Swagger\) Docs](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [API Documentation](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [ADMIN: Ticket Manager](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [ADMIN: Vendor Manager](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [ADMIN: Shift Manager \(Admin / Director\)](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [ADMIN: Artist Manager](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Staff Panel](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Vendor Panel](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Volunteer Panel](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Artist Panel](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Admin Panel](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Friends](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [COMP: Request Medical](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [COMP: Bottom Nav](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Profile Editor](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Interactive Map & Embedded Navigation](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [COMP: Top Nav](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: My Schedule](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Events](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [PAGE: Tickets](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [UI/UX](#)
Feb 11, 2025 • contributed by [Robert Eriksen](#)

-  [DETAILED-API-FLOW.PNG](#)
Feb 11, 2025 • attached by [Robert Eriksen](#)

-  [DECISION 8: File Upload & Storage Service](#)

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 7: Embedded Navigation SDK Selection

Feb 11, 2025 • contributed by [Robert Eriksen](#)

Re-evaluating In-App Navigation vs. External Mapping Solutions

Feb 11, 2025 • contributed by [Robert Eriksen](#)

Infrastructure Setup to Deployment & Monitoring

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 1: Cloud Provider Selection

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 6: CI/CD and DevOps Pipeline

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 5: Real-Time Notification

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 4: Database Selection

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION TEMPLATE

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 3: Frontend Framework Selection

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISION 2: Backend Framework Selection

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DECISIONS

Feb 11, 2025 • contributed by [Robert Eriksen](#)

DOCS

Feb 11, 2025 • contributed by [Robert Eriksen](#)

BIGFAM

Feb 05, 2025 • attached by [Robert Eriksen](#)

Big Fam App

Feb 05, 2025 • contributed by [Robert Eriksen](#)

Big Fam Festival App | Comprehensive Jira Tasks Report (Refactored)

Feb 05, 2025 • contributed by [Robert Eriksen](#)

Big Fam Festival App | Comprehensive Design Document

Feb 05, 2025 • contributed by [Robert Eriksen](#)

Big Fam Festival App | Outstanding Technology & Architecture Decisions

Feb 05, 2025 • contributed by [Robert Eriksen](#)

Big Fam Festival App | New Developer Onboarding

Feb 05, 2025 • contributed by [Robert Eriksen](#)

Big Fam Festival App | The Ultimate Festival Companion

Feb 05, 2025 • contributed by [Robert Eriksen](#)

TLDRJwj047xVNp7oOEf33__0MC2laz2c5aqVZ35J6Diz1tQQtQscCVvtXribJU15kNsNPOSxvz7wOrsfvBUj26ORVBN6GJoWhZYOnCS EscyQZefywYeQai5HvDKmdYIfFnaD2xBiW5yAb0dkS5RxMqPuV32tIL3Mtxz4kfeY7gPJI4Y39ZB2722ar1DW1qa4MNCn4M_ya8nUelJ 9PSqfz_BfoZyr88KxfOH9VuiH5pWBr.png

Jan 31, 2025 • attached by [Robert Eriksen](#)

Get started with the basics ☀

Start editing this page:

- Click the pencil icon  or `e` on your keyboard to edit and start typing. You can edit anywhere.
- Hit `/` to see all the types of content you can add to your page. Try `/image` or `/table`
- Use the toolbar at the top to play around with `font`, `colors`, `formatting`, and more
- Click `close` to save your draft or `publish` when your page is ready to be shared

💡 Need some inspiration?

🔗 Check out our [Confluence best practices guide](#).

- Get a quick intro into what spaces are and how to best use them at [Set up your site and spaces](#).

- Check out our guide for ideas on how to [set up your space overview](#).
- If starting from a blank space is daunting, try using one of the [space templates](#) instead.

Big Fam Festival App | The Ultimate Festival Companion

Below is an updated, compelling pitch for the Big Fam Festival App that encapsulates our comprehensive design—from multi-stakeholder engagement and offline capabilities to interactive mapping, embedded navigation, and robust ticketing—all designed to elevate the festival experience.

Last updated: Feb 5, 2025 by [@Robert Eriksen](#)

Imagine a festival experience where every attendee, vendor, artist, and staff member is seamlessly connected, informed, and empowered—right from your pocket. The Big Fam Festival App is that solution.

What We Offer

- **All-in-One Platform for Every Festival Participant**

Our app isn't just for attendees. It's a comprehensive solution for everyone:

- **Attendees:** Create personalized schedules, access offline ticketing, receive real-time notifications, and even mark your campsite—all in one place.
- **Artists & Vendors:** Manage performance schedules and business profiles, update menus or bios, and communicate directly with festival staff.
- **Staff & Admin:** From volunteer coordinators to high-level admins, everyone gets a role-specific dashboard. Admins control stage schedules, monitor real-time attendee locations, and flag users for enhanced safety and security.

- **Seamless, In-App Experience with Embedded Navigation**

Whether you're navigating to a stage, a food vendor, a medical tent, or even your own campsite, our fully embedded, turn-by-turn navigation ensures you never have to leave the app. Our offline-capable routing means you're always guided—even in areas with spotty connectivity.

- **Interactive Map with Dynamic, Real-Time Features**

Explore the festival grounds through an interactive map that displays every point of interest with custom icons and live updates:

- **Stages, Vendors, and Facilities:** Easily find where performances and services are happening.
- **Campsite Sharing & Friend Locations:** See your friends' campsites (displayed with their profile pictures) and track their last reported positions.
- **Live Position Indicator:** A blinking icon shows your current location in real time, keeping you on track throughout the festival.

- **Robust Ticketing & Security**

Our ticketing management system, accessible by admin and staff, uses QR code scanning and offline data caching to validate tickets quickly and securely—even during peak loads. Every scan is logged for accountability, ensuring a smooth entry experience.

- **Social & Personalized Experience**

Share your festival journey effortlessly:

- **Schedule Sharing Snapshot:** Generate and share visually stunning snapshots of your personal festival schedule on social media.
- **Personalized Itinerary:** With our schedule builder (available to every user), curate your festival experience and get push notifications for upcoming events—so you never miss a beat.

- **Scalable & Reliable Infrastructure**

Built on a robust cloud-native foundation using AWS/GCP with auto-scaling, load balancing, and comprehensive real-time monitoring, our platform is engineered to handle over 4000 concurrent users. Extensive stress testing and offline capabilities ensure a flawless experience even under heavy load.

Why Big Fam Festival App? ☺

- **Unified Experience:** No more juggling multiple apps. One app for all your festival needs—from planning and navigation to ticketing and social sharing.
- **Empowered Communication:** Seamless communication channels let artists, vendors, and staff collaborate effortlessly. Admins get critical insights through advanced monitoring and flagging tools.
- **Innovative & Future-Ready:** With embedded offline navigation, real-time interactive mapping, and comprehensive social features, the Big Fam Festival App is built to scale and evolve as festivals grow.
- **User-Centric Design:** Our intuitive UI, dynamic navigation bar, and role-specific dashboards ensure that every user—whether a first-time attendee or a veteran festival organizer—finds exactly what they need, when they need it.

In a Nutshell ☺

The Big Fam Festival App transforms the chaotic festival experience into a harmonious, connected, and unforgettable journey. By bridging technology with the vibrant energy of a live festival, we empower every participant to be informed, engaged, and truly part of the Big Fam community.

Join us in redefining the festival experience—where every detail is at your fingertips, and every moment is seamlessly connected.

Let me know if you need any further adjustments or additional points to highlight for your stakeholders!

Big Fam Festival App | Comprehensive Design Document

Below is an updated Big Fam Festival App Design document formatted for Confluence. This document consolidates all our discussions—from initial requirements to the final prioritized implementation roadmap—into a single comprehensive design specification.

Last updated: Feb 5, 2025 by [@Robert Eriksen](#)

1. Overview

The Big Fam Festival App is a multi-stakeholder platform engineered to revolutionize the festival experience for every participant. Our solution serves:

- **Attendees:** Who can build personalized schedules, receive real-time notifications, access offline ticketing, and mark/share their campsites.
- **Vendors:** Who update business profiles, manage menus, set operating hours, and mark their location on an interactive map.
- **Artists:** Who view performance schedules, update their profiles, and directly communicate with festival staff.
- **Staff & Volunteers:** Who have access to role-specific dashboards for managing shifts and internal communications.
- **Admins:** Who possess comprehensive control over festival operations, including modifying stage schedules, monitoring attendee locations, flagging users, and managing ticketing.

The platform is built upon a secure, token-based authentication system with Role-Based Access Control (RBAC), ensuring that each user's experience and permissions are tailored according to their role. The app is further enhanced by offline capabilities, an interactive map with embedded turn-by-turn navigation, and robust infrastructure designed to handle 4000+ concurrent users.

2. Refactored Epics

Below is our reorganized roadmap, which has been refactored to follow a logical implementation route while minimizing redundancies and optimizing development time.

Epic A: Infrastructure, Cloud Setup & Stress Testing

- **Objective:**
Establish a robust, scalable cloud-native infrastructure to support high concurrency and ensure system reliability.
- **Key Areas:**
 - Cloud Provider & Deployment Model (AWS/GCP, Kubernetes vs. Serverless)
 - Database & Caching (Managed PostgreSQL, Redis, NoSQL for real-time/offline data)
 - CDN & Content Delivery (CloudFront/Cloud CDN)
 - Real-Time Monitoring, Logging & Stress Testing (Prometheus/Grafana, ELK, k6/JMeter)

Epic B: Profile & Authentication Enhancements and Ticket Type Assignment

- **Objective:**
Build a unified authentication and profile management system, ensuring every user has a persistent profile (accessed via the nav bar) and that ticket types are derived automatically from ticket records.
- **Key Areas:**

- Persistent profile icon in the nav bar for authenticated users; “Login” button for guests.
- Universal Profile Editor allowing updates to contact info, notification preferences, and profile images.
- Backend logic to derive ticket type (allowed values: "ga", "vip", "rv", defaulting to "need-ticket").

Epic C: Core Attendee Experience & Offline Functionality

- **Objective:**

Deliver a robust personal schedule builder with offline support and real-time notifications to empower users to manage their festival itinerary.

- **Key Areas:**

- Personalized schedule builder interface with local caching.
- Real-time push notifications for event reminders.
- Offline ticketing capabilities ensuring QR code and ticket data availability even during connectivity issues.

Epic D: Global Landing Page & Dynamic Navigation

- **Objective:**

Develop a unified landing page that displays live and upcoming events across all festival stages, and a dynamic navigation bar that adapts based on user authentication and role.

- **Key Areas:**

- Card-based landing page with stage filter buttons.
- Dynamic nav bar showing profile icon (or “Login” for unauthenticated users) and key links (My Schedule, Map, etc.).

Epic E: Interactive Map, Embedded Navigation & POI Enhancements

- **Objective:**

Create an interactive map that displays all Points of Interest (POIs) with custom icons, including stages, vendors, facilities, campsites, and friend locations, integrated with an embedded navigation system that supports offline routing.

- **Key Areas:**

- Interactive map with custom markers and detailed pop-ups.
- Embedded turn-by-turn navigation for every POI using a navigation SDK.
- Live position indicator (blinking icon) and custom map legend.

Epic F: Friend Management & Campsite Sharing

- **Objective:**

Enable users to manage friend relationships and share their campsites. Friends’ shared campsites and last reported locations are displayed on the map using profile pictures.

- **Key Areas:**

- Dedicated Friends tab for adding friends (by email/phone), and managing pending requests.
- “Mark My Campsite” feature with optional sharing.
- Map integration showing friend campsite markers (labeled, with “Take Me Here”) and last reported locations using profile pictures.

Epic G: Enhanced Social Features – Schedule Sharing Snapshot

- **Objective:**

Allow users to generate and share a visually appealing snapshot of their personal festival schedule on social media, all within the app.

- **Key Areas:**

- “Share My Schedule” button in the schedule builder.

- Generation of a branded snapshot (including event names, times, stages, imagery) optimized for social media.
- Offline snapshot generation support.

Epic H: Ticketing Management System

- **Objective:**

Provide a robust ticketing solution for Admin and Staff panels, enabling QR code scanning, ticket validation, and offline redemption with background synchronization.

- **Key Areas:**

- QR code scanning module integrated in Admin/Staff panels.
- Real-time and offline ticket validation.
- Local caching and background sync for ticket redemptions.
- Audit log for ticket scanning and redemption events.

Epic I: Admin Panel Enhancements & Flagged User Monitoring

- **Objective:**

Empower admins with advanced controls to modify stage schedules, monitor detailed attendee locations, view comprehensive campsite data, and flag users for monitoring.

- **Key Areas:**

- Admin-only interface for modifying stage schedules with audit logging.
- Detailed view of all marked campsites and attendee location monitoring.
- Functionality to flag users, triggering push notifications with navigation to updated locations.

Epic J: Vendor, Artist, and Staff Panels

- **Objective:**

Develop dedicated panels tailored to vendors, artists, and non-admin staff (volunteers, artist relations) with features specific to their roles.

- **Key Areas:**

- Vendor Panel: Business profile management, menu uploads, geolocation, and messaging.
- Artist Panel: Performance schedule display, profile management, and direct communication.
- Staff Panel: Viewing shift assignments, profile updates, and internal messaging.

Epic K: Embedded Navigation & Offline Routing

- **Objective:**

Implement an in-app, embedded navigation system that provides turn-by-turn directions using cached map data, ensuring full offline capability.

- **Key Areas:**

- Integration of an embedded navigation SDK.
- Local caching of map tiles, POI data, and routing information.
- Seamless in-app navigation with turn-by-turn instructions and route overlays.

Epic L: Testing & Quality Assurance

- **Objective:**

Ensure comprehensive testing across all modules, including unit, integration, regression, and RBAC tests.

- **Key Areas:**

- Extensive unit and integration tests for frontend, backend, and navigation.
- Role-specific testing to enforce RBAC.
- Automated regression testing using Cypress (web) and Detox (mobile).

3. Architecture & Implementation ☰

3.1. Infrastructure ☰

- **Cloud Environment:** AWS/GCP with Kubernetes or serverless options.
- **Database & Caching:** Managed PostgreSQL, Redis, Firestore/DynamoDB.
- **CDN:** CloudFront/Cloud CDN for global asset distribution.
- **Monitoring & Stress Testing:** Prometheus/Grafana and load testing with k6/JMeter.

3.2. Backend & API ☰

- **Authentication & RBAC:** JWT-based authentication with role claims.
- **API Endpoints:** Unified endpoints for profile, schedule, map POIs, friend management, ticketing, and admin-specific actions.
- **File Uploads & Mapping:** Shared services for profile images, menu uploads, and interactive map integration.
- **Offline Sync:** Local caching (IndexedDB/native storage) with background synchronization for schedule data, ticketing, and map assets.

3.3. Frontend ☰

- **UI Components:** Consistent, reusable components for forms, navigation, and notifications.
 - **Role-Specific Panels:** Dedicated dashboards for Admin, Vendor, Artist, and Staff.
 - **Dynamic Navigation:** Adaptive navigation bar showing a persistent profile icon (or 'Login' button) and key links.
 - **Embedded Navigation:** Fully in-app turn-by-turn routing with offline capability.
-

4. Database Schema & API Documentation ☰

Refer to the separate Database Schema and API Documentation documents in the `/docs` directory. These cover:

- **Users, Profiles, Tickets, Campsites, Flagged Users, and Facilities tables.**
 - **Endpoints for Authentication, Schedule Building, Interactive Map, Friend Management, Ticketing, Admin actions, and Role-Specific Panels.**
-

5. Testing & Quality Assurance ☰

- **Unit & Integration Testing:** Minimum 80% coverage.
 - **RBAC Testing:** Verify that restricted endpoints are accessible only by appropriate roles.
 - **Automated Regression Testing:** Using Cypress and Detox.
-

6. Roadmap & Next Steps ☰

- **Phase 1:** Infrastructure Setup & Stress Testing (Epic A)
- **Phase 2:** Profile & Authentication, Ticket Type Assignment (Epic B)
- **Phase 3:** Core Attendee Experience (Epic C)
- **Phase 4:** Global Landing Page & Navigation (Epic D)
- **Phase 5:** Interactive Map & Embedded Navigation (Epic E)
- **Phase 6:** Friend Management & Campsite Sharing (Epic F)
- **Phase 7:** Social Schedule Sharing (Epic G)

- **Phase 8:** Ticketing Management (Epic H)
- **Phase 9:** Admin Panel Enhancements & Flagged Monitoring (Epic I)
- **Phase 10:** Vendor, Artist, and Staff Panels (Epic J)
- **Phase 11:** Embedded Navigation Enhancements (Epic K)
- **Phase 12:** Testing & QA (Epic L)

Regularly update this roadmap checklist in Confluence to reflect progress and adjust priorities as needed.

7. Conclusion ☺

This updated design document reflects our refactored Epics and provides a comprehensive guide to the Big Fam Festival App project—from foundational infrastructure and authentication to core attendee functionalities, interactive mapping, social features, role-specific panels, and robust testing. This document serves as the single source of truth for our team and will be maintained in Confluence for ongoing reference.

Lead Developer: Robert Eriksen

For questions or clarifications, please contact Robert at robert.eriksen@bigfamfestival.com.

Let me know if any further modifications or additional sections are needed!

Big Fam Festival App | Comprehensive Jira Tasks Report (Refactored)

Last updated: Feb 5, 2025 by [@Robert Eriksen](#)

This document provides a detailed, high-level overview of our project tasks organized into Epics, with each Story containing verbose descriptions, acceptance criteria, and estimated story points. Use this as a reference for progress tracking and for Jira imports (after appropriate field mapping).

Epic A: Infrastructure, Cloud Setup & Stress Testing

Objective: Establish a robust, scalable cloud-native infrastructure to support 4000+ concurrent users. This epic covers cloud provider selection, compute, database, caching, CDN, monitoring, and stress testing.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic A: Infrastructure, Cloud Setup & Stress Testing	Establish a robust, scalable cloud-native infrastructure to support 4000+ concurrent users. Includes cloud deployment, auto-scaling, load balancing, databases, caching, CDN, monitoring, and stress testing.	<ul style="list-style-type: none">Infrastructure provisioned on the selected provider.- Auto-scaling and load balancing are operational.- Databases and caching layers are set up and tested.- CDN delivers static assets with low latency.- Stress tests simulate 4000+ users with acceptable performance.	--	
Story	A.1 Cloud Environment Setup	Deploy the application using AWS/GCP with Kubernetes (EKS/GKE) or serverless functions. Configure auto-scaling policies, load	1. Cloud resources are provisioned on the selected provider.2. Auto-scaling triggers based on load.3. Load balancers	8	Epic A: Infrastructure, Cloud Setup & Stress Testing

		balancers, and security groups to ensure high availability.	distribute traffic evenly.4. Security configurations are verified in staging.		
Story	A.2 Database Setup & Caching	Configure a managed PostgreSQL instance for structured data, set up Redis for caching, and integrate Firestore/Dynamo DB for real-time/offline support. Implement backup and replication strategies.	1. PostgreSQL is deployed with backups and replication.2. Redis caching is operational for key queries.3. Real-time/offline data store is integrated and tested.	8	Epic A: Infrastructure, Cloud Setup & Stress Testing
Story	A.3 CDN Configuration	Set up a CDN (CloudFront/Cloud CDN) to deliver static assets globally and reduce latency.	1. Static assets are served via the CDN with reduced latency.2. CDN configuration passes performance tests.3. Documentation on CDN integration is complete.	5	Epic A: Infrastructure, Cloud Setup & Stress Testing
Story	A.4 Real-Time Monitoring & Logging Setup	Implement real-time monitoring using Prometheus/Grafana (or CloudWatch/Stac kdriver) and centralized logging (ELK or similar) to track system health, resource usage, and errors.	1. Monitoring dashboards display key metrics (CPU, memory, API response times, error rates).2. Alerts are configured for critical thresholds.3. Centralized logging is set up and accessible for	5	Epic A: Infrastructure, Cloud Setup & Stress Testing

			troubleshooting.		
Story	A.5 Load & Stress Testing	Conduct load testing using k6/JMeter to simulate 4000+ concurrent users. Analyze performance data and optimize auto-scaling and caching configurations accordingly.	1. Load tests simulate at least 4000 concurrent users.2. Performance metrics meet defined benchmarks.3. Bottlenecks are identified and addressed.	5	Epic A: Infrastructure, Cloud Setup & Stress Testing

Epic B: Profile & Authentication Enhancements and Ticket Type Assignment ☀

Objective: Build a unified authentication and profile management system. Every user (attendee, vendor, artist, staff, admin) will have a persistent profile accessible via the navigation bar, and an attendee's ticket type will be derived automatically from ticket records.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment	Implement a unified, secure authentication system with global profile management. Derive ticket type automatically (defaulting to 'need-ticket' if no ticket exists).	<ul style="list-style-type: none"> Users authenticate via JWT with role claims.- Profile management is accessible through a persistent nav bar icon.- Ticket type is derived automatically from backend data. 	--	
Story	B.1 Persistent Profile Icon in Nav Bar	Display the user's profile picture (or default avatar) in the navigation bar for authenticated users; clicking it navigates to the Profile Editor.	1. Profile icon is visible on all screens for logged-in users.2. Clicking the icon navigates to the Profile Editor.3. Profile updates	5	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment

			are reflected in real time.		
Story	B.2 Universal Profile Editor	Develop a dedicated Profile Editor view where users can update their profile picture, contact info, and notification preferences. This view is separate from the schedule builder.	1. Users can update profile picture, contact details, and notification settings.2. Changes are saved and immediately reflected.3. The interface is responsive and accessible across all roles.	5	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment
Story	B.3 Nav Bar for Unauthenticated Users	Display a 'Login' button in the nav bar for users not logged in, which routes them to a login page with options for registration and password reset.	1. Unauthenticated users see a 'Login' button in the nav bar.2. Clicking 'Login' navigates to an authentication page with registration and password reset options.	3	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment
Story	B.4 Derive Ticket Type from Ticket Records	Implement backend logic to derive a user's ticket type from their ticket record; default to 'need-ticket' if no ticket exists. This field is read-only.	1. API responses include a derived 'ticket_type' field.2. Users with no ticket receive 'need-ticket'.3. The field is not modifiable by the user.	3	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment
Story	B.5 Update API Endpoints to Return Derived Ticket Type	Modify registration and profile update endpoints so that the 'ticket_type' is not user-editable, and is instead derived	1. Endpoints reject any 'ticket_type' input.2. Returned user details include the correct	2	Epic B: Profile & Authentication Enhancements and Ticket Type Assignment

		from existing ticket data.	derived ticket type.	
--	--	----------------------------	----------------------	--

Epic C: Core Attendee Experience & Offline Functionality ☀

Objective: Provide every user with a robust personal schedule builder, offline access to schedule and ticket data, and real-time notifications.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic C: Core Attendee Experience & Offline Functionality	Enable users to create and manage a personal schedule with offline support and real-time notifications for events and ticketing.	<ul style="list-style-type: none"> Users can add events to a personal schedule.- Offline caching is implemented.- Real-time notifications trigger for upcoming events. 	--	
Story	C.1 Personalized Schedule Builder	Develop an interface for users to browse festival events and add them to their personal itinerary with offline caching using Service Workers/IndexDB.	<ol style="list-style-type: none"> Users can browse and add events. Schedule data is cached and available offline. Push notifications trigger for events. 	5	Epic C: Core Attendee Experience & Offline Functionality
Story	C.2 Real-Time Notifications & Offline Ticketing	Implement push notifications for event reminders and ensure ticket information (including QR codes) is accessible offline for entry validation.	<ol style="list-style-type: none"> Notifications are delivered in real time. Ticket data is accessible offline and validated correctly. Offline functionality is verified. 	8	Epic C: Core Attendee Experience & Offline Functionality

Epic D: Global Landing Page & Dynamic Navigation ↗

Objective: Develop a unified landing page displaying live/upcoming events across all stages with stage filters, and a dynamic navigation bar that adapts based on authentication state.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic D: Global Landing Page & Dynamic Navigation	Develop a landing page with a card-based view of live/upcoming events and a dynamic navigation bar that adapts based on user authentication.	<ul style="list-style-type: none"> Landing page displays events in a clear, card-based layout.- Stage filter buttons work properly.- Navigation bar adapts: profile icon for authenticated users, 'Login' for unauthenticated users. 	--	
Story	D.1 Unified Landing Page with Live/Upcoming Events & Stage Filters	Create a landing page that shows live/upcoming events across all stages with filter buttons for each stage.	<ol style="list-style-type: none"> Events are displayed in a card layout. Stage filters update the view accurately. The landing page is responsive. 	8	Epic D: Global Landing Page & Dynamic Navigation
Story	D.2 Dynamic Navigation Bar	Implement a navigation bar that shows a persistent profile icon for logged-in users (navigating to Profile Editor) and a 'Login' button for guests, along with key feature links.	<ol style="list-style-type: none"> Nav bar displays profile icon for authenticated users and 'Login' for unauthenticated users. All key links (My Schedule, Map, etc.) are functional. Nav bar updates dynamically on login/logout. 	5	Epic D: Global Landing Page & Dynamic Navigation

Epic E: Interactive Map, Embedded Navigation & POI Enhancements

Objective: Create an interactive map that displays all POIs with custom icons, embedded in-app turn-by-turn navigation, and offline support.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic E: Interactive Map, Embedded Navigation & POI Enhancements	Develop an interactive map showing markers for stages, vendors, facilities, campsites, friend campsites, and friend last reported locations. Integrate embedded, in-app navigation and offline functionality.	<ul style="list-style-type: none"> Map displays all POIs with custom icons.- Each marker has a pop-up with details and a 'Take Me Here' button.- Live position indicator and offline caching are implemented. 	--	
Story	E.1 Interactive Map with POI Markers	Implement an interactive map that displays markers for all POIs with detailed pop-ups and a 'Take Me Here' navigation button.	<ol style="list-style-type: none"> All POIs are accurately displayed with custom icons. Pop-ups provide correct details and navigation options. Offline caching of map data is implemented. 	8	Epic E: Interactive Map, Embedded Navigation & POI Enhancements
Story	E.2 Embedded Navigation for All POIs	Integrate an embedded navigation SDK to provide in-app turn-by-turn navigation with route overlays. Each POI pop-up includes a 'Take Me Here' (or 'Take Me Home') button.	<ol style="list-style-type: none"> Navigation launches in-app without external redirection. Turn-by-turn instructions are accurate and clear. Offline functionality is supported. 	10	Epic E: Interactive Map, Embedded Navigation & POI Enhancements
Story	E.3 Live Position Indicator	Display the user's current location	<ol style="list-style-type: none"> Live position icon is clearly 	3	Epic E: Interactive Map,

		as a continuously updating, blinking icon on the map using real-time geolocation data.	visible on the map.2. It updates in real time.3. Blinking effect indicates active tracking.		Embedded Navigation & POI Enhancements
Story	E.4 Additional Facility POIs & Custom Icons	Add markers for additional facilities (medical, watering stations, bathrooms, staff buildings) using custom-designed icons. Each marker should be navigable with a 'Take Me Here' button.	1. Facility markers display with distinct custom icons.2. Pop-ups show facility details and a navigation button.3. Offline support is verified.	8	Epic E: Interactive Map, Embedded Navigation & POI Enhancements
Story	E.5 Custom Legend/Key Integration	Develop an overlay legend that clearly explains the custom icons used on the map, including those for friend campsites and friend last reported locations.	1. Legend overlay is accessible on the map.2. Each icon is clearly defined.3. Legend updates dynamically with changes.	3	Epic E: Interactive Map, Embedded Navigation & POI Enhancements

Epic F: Friend Management & Campsite Sharing ☺

Objective: Enable users to manage friend relationships and share their campsites, with friend campsites and last reported locations displayed on the map using profile pictures.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic F: Friend Management & Campsite Sharing	Enable users to add friends, manage pending requests, mark their campsite, and display friends' shared campsites and	<ul style="list-style-type: none"> Users can search for friends and send friend requests.- 'Mark My Campsite' feature 	--	

		last reported locations on the map.	captures and optionally shares location.- Friend campsites and last locations display on the map with navigation options.		
Story	F.1 Friend Management System	Build a dedicated Friends tab for users to add friends (via email or phone), view pending requests, and manage the friend list.	1. Users can search by email/phone.2. Pending and accepted requests display correctly.3. Friend list updates in real time.	8	Epic F: Friend Management & Campsite Sharing
Story	F.2 Mark My Campsite with Sharing Option	Implement a 'Mark My Campsite' feature that captures the user's geolocation with an option to share it with friends. The campsite marker displays with a 'Take Me Home' button.	1. User's campsite location is accurately captured.2. Marker is displayed on the map with correct coordinates.3. Sharing option updates user profile accordingly.	8	Epic F: Friend Management & Campsite Sharing

Story	F.3 Display Friend Campsites Using Profile Pictures	Render friend campsite markers on the map using a distinct home icon combined with the friend's profile picture and label (e.g., "Friend A's Campsite") along with a 'Take Me Here' button.	1. Friend campsite markers display correct profile picture and label. 2. Pop-ups include a functional 'Take Me Here' button. 3. Navigation from markers works correctly.	5	Epic F: Friend Management & Campsite Sharing
Story	F.4 Display Friends' Last Reported Locations as Profile Pictures	Show friends' last reported locations on the map using their profile pictures as markers; pop-ups display the friend's name, update timestamp, and a 'Take Me Here' button.	1. Friends' last reported locations are rendered using profile pictures. 2. Pop-ups show friend's name and update time. 3. Navigation works correctly.	5	Epic F: Friend Management & Campsite Sharing

Epic G: Enhanced Social Features – Schedule Sharing Snapshot ☀

Objective: Allow users to generate and share a visually appealing snapshot of their personal schedule on social media directly from the app.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic G: Enhanced Social Features – Schedule Sharing Snapshot	Enable users to share a branded snapshot of their personal schedule on social media, using native share functionality without leaving the app.	<ul style="list-style-type: none"> A snapshot of the user's schedule is generated from cached data.- The snapshot is styled with festival branding and optimized for social media. 	--	

Story	G.1 Implement 'Share My Schedule' Snapshot Generation	Add a 'Share My Schedule' button in the schedule builder interface that generates a snapshot image/card summarizing the user's schedule with event names, times, stages, and imagery.	1. The button is visible and functional.2. A snapshot image is generated with all key details.3. The snapshot meets branding guidelines.	5	Epic G: Enhanced Social Features - Schedule Sharing Snapshot
Story	G.2 Optimize Snapshot Design for Social Media	Refine the visual design of the generated schedule snapshot to ensure high resolution, responsiveness, and alignment with festival branding for social media sharing.	1. Snapshot design aligns with branding.2. Image is high resolution and responsive.3. User testing feedback is incorporated.	3	Epic G: Enhanced Social Features - Schedule Sharing Snapshot
Story	G.3 Offline Support for Snapshot Generation	Ensure the snapshot generation process uses locally cached schedule data, allowing offline snapshot creation and sharing.	1. Snapshot generation works offline using cached data.2. Users are notified when offline data is used.3. The snapshot accurately reflects the current schedule.	3	Epic G: Enhanced Social Features - Schedule Sharing Snapshot

Epic H: Ticketing Management System ☈

Objective: Implement a robust ticketing solution for Admin and Staff panels to scan, validate, and redeem tickets using QR codes, with offline support and background synchronization.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link

Epic	Epic H: Ticketing Management System	Develop a ticketing system that enables Admin/Staff to scan and validate tickets, mark them as redeemed, and maintain an audit log, with offline caching and sync.	<ul style="list-style-type: none"> Ticket scanning and validation work both online and offline.- Audit logs record all ticket redemptions. 	--		
Story	H.1 QR Code Ticket Scanning & Validation	Develop a module that uses the device camera to scan QR codes, decode the ticket identifier, and validate ticket existence, validity, and redemption status.	<ol style="list-style-type: none"> QR scanner accurately reads codes. Validation confirms ticket existence, validity, and non-redemption. Visual feedback is provided for successful or failed scans. 	8	Epic H: Ticketing Management System	
Story	H.2 Offline Ticket Data Caching & Sync	Implement local caching for ticket data and offline redemption attempts, with a background sync process to update the backend upon connectivity restoration.	<ol style="list-style-type: none"> Ticket data is cached locally. Offline redemption attempts are queued and synced when online. Duplicate redemptions are handled appropriately. 	5	Epic H: Ticketing Management System	
Story	H.3 Ticket Redemption Audit Log	Build an audit log view in the Admin panel that records all ticket scan events, including ticket ID, redemption timestamp, and operator details, with filtering options.	<ol style="list-style-type: none"> Audit log is accessible only to admins. All ticket scan events are recorded with complete details. The log supports filtering by 	5	Epic H: Ticketing Management System	

			operator and timestamp.	
--	--	--	-------------------------	--

Epic I: Admin Panel Enhancements & Flagged User Monitoring

Objective: Enhance the Admin panel with advanced management features such as modifying stage schedules, detailed attendee and campsite monitoring, and flagged user alerts with push notifications.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic I: Admin Panel Enhancements & Flagged User Monitoring	Provide admins with tools to modify stage schedules, view comprehensive attendee and campsite data, and flag users to trigger push notifications for location updates.	<ul style="list-style-type: none"> Admin-only functionalities are secured.- Stage schedules can be modified with audit logging.- Detailed views of attendee locations and campsites are available.- Flagging users triggers appropriate notifications. 	--	
Story	I.1 Admin View for Modifying Stage Schedules	Develop an interface for admins to modify stage schedules (event times, order, details) with conflict validation and audit logging.	1. Only admins can access the schedule modification interface.2. Schedule changes are validated and conflict-free.3. All changes are logged with admin ID, previous and new values, and change reason.	5	Epic I: Admin Panel Enhancements & Flagged User Monitoring
Story	I.2 Admin View of All Marked Campsites with Attendee Details	Create a dashboard view that lists all marked campsites along with detailed	1. The dashboard is restricted to admins.2. It displays comprehensive and up-to-	5	Epic I: Admin Panel Enhancements & Flagged User Monitoring

		attendee information (name, role, contact, account notes, geolocation, timestamp).	date attendee data for each campsite.3. Filtering and sorting options are available.		
Story	I.3 Display All Attendees' Last Reported Positions	Provide a real-time view in the Admin panel showing each attendee's last reported location with their profile picture, name, contact info, and update timestamp, including a 'Take Me Here' button.	1. View is accessible only to admins.2. Attendee locations are updated in real time.3. Navigation buttons function correctly.	5	Epic I: Admin Panel Enhancements & Flagged User Monitoring
Story	I.4 Flag a User for Monitoring with Push Notifications	Enable admins to flag a user so that when the flagged user's location updates, a push notification with a 'Take Me Here' button is sent to the flagging admin.	1. Only admins can flag users.2. A flag record is created with a flag reason.3. On location update, a push notification is sent with accurate details.4. The 'Take Me Here' button navigates correctly.	8	Epic I: Admin Panel Enhancements & Flagged User Monitoring

Epic J: Vendor, Artist, and Staff Panels ☀

Objective: Develop dedicated panels tailored to the needs of vendors, artists, and non-admin staff (volunteers, artist relations).

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic J: Vendor, Artist, and Staff Panels	Develop dedicated dashboards for vendors, artists, and non-admin	• Vendor Panel allows business profile updates,	--	

		staff, each with role-specific functionalities.	menu uploads, geolocation management, and messaging with staff.- Artist Panel enables performance schedule viewing, profile management, and direct communication.- Staff Panel displays shift assignments, profile updates, and internal messaging.		
Story	J.1 Vendor Panel for Business Profile Management	Develop a Vendor panel to update business profiles, upload menus, manage geolocation, and communicate with Big Fam staff.	1. Vendors can update business details (name, description, operating hours, contact info).2. Menu uploads are processed and securely stored.3. Geolocation data is accurate and reflected on the map.4. Messaging functions are operational.	8	Epic J: Vendor, Artist, and Staff Panels
Story	J.2 Artist Panel for Performance Schedule & Profile Management	Create an Artist panel for artists to view their performance schedules, update their profiles (picture, bio, social links), and contact Big	1. Artists can view an accurate performance schedule with stage, time, and location details.2. Profile updates	8	Epic J: Vendor, Artist, and Staff Panels

		Fam staff or assigned directors.	work reliably.3. Direct communication features are implemented and secure.		
Story	J.3 Staff Panel for Volunteers & Artist Relations	Develop a Staff panel for non-admin staff (volunteers, artist relations) to view shift assignments, update profiles, and access internal messaging functionalities.	1. The panel clearly displays shift assignments.2 . Profile updates are functioning correctly.3. Internal messaging is available and restricted to staff roles.	5	Epic J: Vendor, Artist, and Staff Panels

Epic K: Embedded Navigation & Offline Routing ☺

Objective: Implement a fully in-app, embedded navigation system that provides turn-by-turn directions using cached map data, ensuring offline capability.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic K: Embedded Navigation & Offline Routing	Develop an embedded navigation system that provides turn-by-turn directions entirely within the app and supports offline routing through local caching.	<ul style="list-style-type: none"> Navigation launches in-app with accurate turn-by-turn instructions.- Offline routing works seamlessly with cached map data. 	--	
Story	K.1 Integrate Embedded Navigation SDK	Integrate an in-app navigation SDK (e.g., Mapbox Navigation or HERE SDK) to provide turn-by-turn directions with	1. Navigation launches within the app without external redirection.2. Turn-by-turn directions are accurate and	10	Epic K: Embedded Navigation & Offline Routing

		route overlays and step-by-step instructions.	clear.3. Route overlays are correctly displayed.4. Offline navigation is functional.		
Story	K.2 Ensure Offline Routing with Cached Map Data	Implement local caching of map tiles, POI data, and routing information so that the navigation system operates reliably offline.	1. Map tiles and POI data are cached locally.2. Offline routing calculations produce accurate routes.3. The offline experience is smooth and reliable.	5	Epic K: Embedded Navigation & Offline Routing

Epic L: Testing & Quality Assurance ↴

Objective: Ensure comprehensive testing across all modules, including unit, integration, regression, and RBAC tests.

Issue Type	Summary	Description	Acceptance Criteria	Estimated SP	Epic Link
Epic	Epic L: Testing & Quality Assurance	Implement comprehensive testing for all modules, ensuring at least 80% coverage, proper RBAC enforcement, and automated regression tests.	<ul style="list-style-type: none"> • Minimum 80% test coverage across all modules.- Role-based access is strictly enforced.- Automated regression tests run successfully before merges. 	--	

Story	L.1 Unit & Integration Testing for All Modules	Develop comprehensive unit and integration tests for frontend, backend, and navigation modules.	1. Achieve at least 80% test coverage.2. Critical user flows are thoroughly tested.3. Tests pass successfully in the CI/CD pipeline.	5	Epic L: Testing & Quality Assurance
Story	L.2 Role-Specific RBAC Testing	Implement tests to ensure RBAC is enforced correctly, preventing unauthorized access to restricted endpoints.	1. Non-admin users cannot access admin-only endpoints.2. Admin endpoints function correctly for admin users.3. Automated RBAC tests are integrated into CI/CD.	3	Epic L: Testing & Quality Assurance
Story	L.3 Automated Regression Testing	Set up automated regression tests using Cypress (web) and Detox (mobile) to catch unintended side effects during updates.	1. Regression tests cover all core functionalities.2. Tests are automatically triggered during CI/CD.3. No regressions are detected before merging code.	3	Epic L: Testing & Quality Assurance

Summary ↴

This Markdown document provides a verbose backup of all Epics, Stories, and Features for the Big Fam Festival App project, including detailed descriptions, acceptance criteria, and estimated story points. Before importing into Jira, ensure that custom fields such as "Acceptance Criteria" and "Story Points" are set up and that you map the columns appropriately.

Let me know if you need any further modifications or additional details!

Big Fam Festival App | Outstanding Technology & Architecture Decisions

Below is the updated Confluence Decision Document outlining all key decisions that still need to be made for the Big Fam Festival App project. This document follows our standard Confluence Decision Documentation template and includes research details, pros and cons, and recommendations for each decision. Please review and provide feedback so we can finalize our choices.

Project: Big Fam Festival App

Lead Developer: [@Robert Eriksen](#)

Last Updated: Feb 5th, 2025

Decision 1: Cloud Provider Selection

Context

We must choose a primary cloud provider to host our infrastructure, which includes container orchestration, managed databases, caching, CDN, and real-time monitoring services. Our options are primarily AWS and GCP.

Options

- **AWS:**

- **Pros:** Mature services (EKS, RDS, ElastiCache, CloudFront); extensive documentation; broad community support; robust security features.
- **Cons:** Can be complex to manage and may have higher costs at scale depending on usage.

- **GCP:**

- **Pros:** Strong managed services (GKE, Cloud SQL, Memorystore, Cloud CDN); competitive pricing; streamlined integration with other Google services; ease of use.
- **Cons:** Smaller market share than AWS may mean less community support for some niche features; team expertise may vary.

Research Findings

- **Performance & Scalability:** Both providers offer auto-scaling and high-availability configurations.
- **Pricing:** Preliminary cost comparisons show that pricing differences are workload-dependent; a detailed cost model is required.
- **Team Expertise:** Our team currently has more experience with AWS tools, but GCP's streamlined UI could lower the learning curve.

Recommendation

Initial Recommendation: Lean toward AWS for its maturity and our existing team expertise, but conduct a detailed cost analysis and proof-of-concept (POC) to validate performance for our expected workload.

Next Steps

- Perform a cost analysis for anticipated traffic and storage needs.
- Develop a small POC to simulate our workload using AWS services.
- Schedule a team review meeting to finalize the decision.

Decision 2: Deployment Model – Kubernetes vs. Serverless ☰

Context ☰

We need to determine the optimal deployment model for our backend services to support 4000+ concurrent users with complex microservices.

Options ☰

- **Kubernetes (EKS/GKE):**
 - **Pros:** Full control over container orchestration; supports complex, long-running services; mature ecosystem.
 - **Cons:** Higher operational overhead; requires more in-house expertise for cluster management.
- **Serverless (AWS Lambda/GCP Cloud Functions):**
 - **Pros:** Simplifies operations; scales automatically; lower management overhead.
 - **Cons:** Potential cold start issues; less control over long-running processes; potential cost unpredictability under heavy load.

Research Findings ☰

- **Operational Overhead:** Kubernetes requires dedicated DevOps resources, while serverless can reduce operational burden.
- **Performance under Load:** For consistently high traffic and complex workflows, Kubernetes tends to offer more predictable performance.
- **Scalability:** Both models scale, but Kubernetes gives us granular control over resource allocation.

Recommendation ☰

Initial Recommendation: Use Kubernetes for production given the anticipated workload and complexity. Consider a hybrid approach for non-critical services if needed.

Next Steps ☰

- Evaluate team capacity for Kubernetes management.
- Create a pilot deployment to simulate stress conditions.
- Review potential hybrid strategies for microservices that are less latency-sensitive.

Decision 3: Embedded Navigation SDK Selection ☰

Context ☰

We require an in-app navigation solution that delivers full turn-by-turn directions and offline routing without exiting the app.

Options ☰

- **Mapbox Navigation SDK:**
 - **Pros:** Robust turn-by-turn guidance; excellent offline caching support; extensive documentation and community examples.
 - **Cons:** Can be cost-intensive at scale; pricing model needs evaluation.
- **HERE Navigation SDK:**
 - **Pros:** High accuracy; strong offline capabilities; competitive pricing.
 - **Cons:** Less widespread usage in our context; integration examples may be fewer.

- **Alternatives (OpenStreetMap-based solutions):**

- **Pros:** Open source and cost-effective.
- **Cons:** May require additional development effort for offline support and turn-by-turn guidance.

Research Findings

- **Offline Functionality:** Both Mapbox and HERE offer strong offline capabilities, with Mapbox often praised for its offline map caching.
- **Integration Complexity:** Mapbox tends to have more extensive developer resources and sample code, which can accelerate integration.
- **Cost:** HERE's pricing may be more attractive for high-volume scenarios; however, detailed pricing comparisons are necessary.

Recommendation

Initial Recommendation: Conduct a POC with both Mapbox Navigation SDK and HERE Navigation SDK. Based on performance, offline reliability, and integration ease, choose the one that best meets our requirements for seamless, in-app navigation.

Next Steps

- Develop small-scale POCs using both SDKs.
 - Measure performance under offline conditions and assess developer integration experience.
 - Collect cost estimates based on projected usage.
-

Decision 4: Mapping API for Interactive Map

Context

Our interactive map is a critical component, requiring robust POI display, offline functionality, and customization.

Options

- **Mapbox GL JS:**

- **Pros:** Highly customizable; strong offline caching; excellent visual design options; supports custom icons and legends.
- **Cons:** Requires a subscription for high usage; potential learning curve for custom styling.

- **Google Maps API:**

- **Pros:** Feature-rich and familiar to users; excellent documentation.
- **Cons:** More expensive at scale; licensing restrictions; less flexible with offline caching.

- **Leaflet with OpenStreetMap:**

- **Pros:** Open source and free; lightweight.
- **Cons:** May require extra work to support offline caching and advanced customization.

Research Findings

- **Customization & Offline:** Mapbox GL JS leads in terms of offline support and customizability, which is crucial for our festival app.
- **Cost Considerations:** Google Maps' pricing model can become expensive under heavy load, while Leaflet may lack some advanced features.
- **Developer Experience:** Mapbox provides comprehensive documentation and examples for offline maps.

Recommendation

Initial Recommendation: Mapbox GL JS appears to be the most fitting solution for its balance of customization, offline capabilities, and developer support. We will verify this choice with a brief pilot test.

Next Steps

- Validate Mapbox GL JS offline functionality with a prototype.
 - Confirm pricing details against our projected usage.
 - Finalize integration plan with mapping and offline caching strategies.
-

Decision 5: File Upload & Storage Service

Context

We need to manage file uploads for profile pictures and vendor menus securely and efficiently.

Options

- **AWS S3:**
 - **Pros:** Industry standard; seamless integration with AWS infrastructure (if AWS is chosen); robust and secure.
 - **Cons:** Pricing can vary based on usage; requires proper configuration for access control.
- **Google Cloud Storage:**
 - **Pros:** Competitive pricing; integration with GCP services; high scalability.
 - **Cons:** Best suited if GCP is chosen; similar functionality to S3.
- **Azure Blob Storage:**
 - **Pros:** Reliable but less likely to be used based on current provider preferences.
 - **Cons:** Not aligned with our current focus (AWS vs. GCP).

Research Findings

- **Integration:** Align file storage with the chosen cloud provider to simplify integration and management.
- **Security & Performance:** Both AWS S3 and Google Cloud Storage provide high levels of security, scalability, and performance.
- **Cost:** Cost is dependent on usage patterns; pricing models should be evaluated in the context of our overall cloud provider decision.

Recommendation

Initial Recommendation: Use the file storage service that aligns with our final cloud provider decision—if AWS is chosen, use S3; if GCP, use Google Cloud Storage.

Next Steps

- Finalize the cloud provider decision.
 - Perform a cost comparison based on our expected file upload and storage requirements.
 - Update integration documentation accordingly.
-

Summary & Next Steps

This document captures the remaining key decisions for the Big Fam Festival App project. The decisions are:

1. **Cloud Provider Selection:** AWS vs. GCP
2. **Deployment Model:** Kubernetes vs. Serverless
3. **Embedded Navigation SDK:** Mapbox Navigation SDK vs. HERE Navigation SDK
4. **Mapping API for Interactive Map:** Mapbox GL JS vs. Google Maps vs. Leaflet/OpenStreetMap
5. **File Upload & Storage Service:** AWS S3 vs. Google Cloud Storage (aligned with cloud provider)

Next Steps:

- Review and validate each decision with the development team.
- Conduct proof-of-concept tests for the Navigation SDK and Mapping API.
- Perform detailed cost analysis and team capacity review for the cloud provider and deployment model.
- Finalize decisions in a follow-up meeting with stakeholders and update this document with the outcomes.

Please review this document in Confluence and provide any additional input or further research requirements. This living document will be updated as new information is gathered and decisions are finalized.

Let me know if you need any further modifications or additional details for the decision-making process!

Big Fam Festival App | New Developer Onboarding

Below is an updated New Developer Onboarding document for Confluence. This guide is designed to bring new developers up to speed on the Big Fam Festival App project and outlines our technical architecture, development practices, tools, and key contacts. Robert Eriksen is the Lead Developer, and his guidance is available for any questions or assistance during onboarding.

Last Updated: Feb 5, 2025 by [@Robert Eriksen](#)

1. Welcome & Project Overview ☀

Welcome to the Big Fam Festival App development team! This project is a multi-stakeholder platform that enhances the festival experience for attendees, vendors, artists, staff, and admins. Our app integrates personalized schedule building, real-time notifications, embedded offline navigation, robust ticketing, and comprehensive role-based functionality (including Admin, Staff, Vendor, Artist, and Friend Management).

Key Features:

- **Schedule Builder:** Allows every user to create and manage their personal festival itinerary.
- **Interactive Map & Embedded Navigation:** Displays stages, vendors, facilities, campsites, and friend locations with in-app, turn-by-turn navigation.
- **Ticketing Management:** Embedded QR code scanning and validation with offline support.
- **Multi-Panel System:** Separate dashboards for Admin, Staff, Vendors, and Artists.
- **Social & Offline Capabilities:** Schedule sharing on social media, offline data caching, and real-time notifications.

Lead Developer:

Robert Eriksen

Email: robert.eriksen@bigfamfestival.com

Phone: [Insert Phone Number]

Feel free to reach out to Robert for any guidance or clarification during your onboarding.

2. Technical Architecture & Environment ☀

2.1. Cloud Infrastructure & Deployment ☀

• Cloud Provider:

Our application is deployed on AWS (or GCP) using Kubernetes (or serverless options where applicable) for container orchestration and auto-scaling.

• Key Services:

- **Compute:** Kubernetes clusters (EKS) or AWS Lambda for serverless functions.
- **Databases:** Managed PostgreSQL (AWS RDS/GCP Cloud SQL) for structured data; Redis for caching; Firestore/DynamoDB for real-time/offline data.
- **Content Delivery:** CloudFront for CDN distribution.
- **Monitoring:** Prometheus and Grafana (or CloudWatch) for real-time monitoring and logging.

2.2. Continuous Integration & Deployment (CI/CD) ↗

- **CI/CD Pipeline:**

We use GitHub Actions (or GitLab CI/CD) to automate builds, tests, and deployments. Our pipelines include linting, unit testing, integration testing, and automated regression testing.

- **Infrastructure as Code:**

We manage infrastructure with Terraform or Pulumi, ensuring consistency across environments.

3. Codebase & Repository Structure ↗

- **Main Repository:**

The core codebase for the Big Fam Festival App is hosted on GitHub (or Bitbucket). Please refer to the repository's README for a high-level overview.

- **Repository Structure:**

- `/frontend` : Contains the mobile (React Native) and web (Next.js) client applications.
- `/backend` : Contains our server-side code (e.g., Node.js with NestJS or Python with FastAPI).
- `/docs` : Documentation, including this onboarding guide, API documentation, and design documents.
- `/infra` : Infrastructure as Code scripts, deployment configuration, and CI/CD pipeline definitions.

- **Branching & Git Workflow:**

We follow a Git workflow where:

- `main` is the stable production branch.
 - **Feature branches** are created off of `main` for new features and bug fixes.
 - Pull requests (PRs) are reviewed by peers and approved by a senior developer (with Robert Eriksen providing final oversight where necessary).
-

4. Development Setup & Tools ↗

4.1. Local Development Environment ↗

- **Prerequisites:**

- Node.js (version X.X.X or later)
- Python (if using FastAPI)
- Docker (for containerized development)
- Git
- IDE/Text Editor of your choice (e.g., VS Code, IntelliJ)

- **Environment Setup:**

- a. **Clone the Repository:**

```
1 git clone https://github.com/bigfamfestival/app-backend.git
2 git clone https://github.com/bigfamfestival/app-frontend.git
3
```

- b. **Install Dependencies:**

For the backend:

```
1 cd app-backend
2 npm install # For Node.js, or pip install -r requirements.txt for FastAPI
3
```

For the frontend:

```
1 cd app-frontend  
2 npm install  
3
```

c. Environment Variables:

Create a `.env` file in the backend and frontend directories with the required configuration variables (refer to the sample `.env.example` provided in the repository).

d. Database Setup:

Ensure that your local PostgreSQL instance is running and configured. Use provided migration scripts (e.g., via Flyway or Alembic) to set up the schema.

4.2. Tools & Services ↗

- **Version Control:** Git
- **CI/CD:** GitHub Actions (review pipelines in the `.github/workflows` directory)
- **Containerization:** Docker (see Dockerfiles for backend and frontend)
- **Testing:**
 - **Backend:** Jest (for Node.js) or Pytest (for FastAPI)
 - **Frontend:** Cypress (for web), Detox (for mobile)
- **Communication:** Slack/Teams for daily standups and developer channels.
- **Documentation:** Confluence for project documentation and design documents.

5. Development Process & Best Practices ↗

- **Coding Standards:**

Follow our established style guides (provided in the `/docs` folder) for code consistency and quality.

- **Code Reviews:**

All PRs must be reviewed by at least one peer, with Robert Eriksen providing final sign-off on critical changes.

- **Testing:**

Write unit tests and integration tests for every new feature. Automated tests must pass in the CI/CD pipeline before merging.

- **Commit Messages:**

Use clear, descriptive commit messages that follow our conventional commit guidelines.

- **Issue Tracking:**

All work is tracked in Jira. Ensure your branch name and commit messages reference the relevant Jira task (e.g., “BF-123: Implement schedule builder feature”).

6. Deployment & Monitoring ↗

- **Deployment Process:**

- Code is automatically deployed to a staging environment via our CI/CD pipeline.
- Once approved, deployments follow a blue/green or canary strategy for production rollout.
- Post-deployment, real-time monitoring dashboards (Prometheus/Grafana) track system performance and health.

- **Monitoring & Logging:**

- Logs are aggregated centrally (via ELK Stack or cloud-native logging solutions) for troubleshooting.
- Alerts are configured for key performance indicators and system anomalies.

- **Stress Testing:**

Regular load and stress testing are conducted using k6 or JMeter to ensure the system can handle high user concurrency (4000+ users).

7. Key Contacts & Resources

- **Lead Developer:**

Robert Eriksen

- Email: robert.eriksen@bigfamfestival.com
- Slack/Teams: @RobertEriksen

- **Project Manager:**

[Insert Name & Contact]

- **Documentation:**

Refer to our Confluence space for detailed design documents, API documentation, and user flows.

- **Additional Resources:**

- GitHub Repositories: [Backend](#), [Frontend](#)
 - Confluence Space: [Big Fam Festival App Documentation](#)
-

8. Next Steps

1. **Review this Onboarding Document:**

Familiarize yourself with the project architecture, codebase structure, and development tools.

2. **Set Up Your Local Environment:**

Follow the steps in Section 4 to clone repositories, install dependencies, and configure environment variables.

3. **Start with a Starter Task:**

Begin with a small Jira task from one of our initial Epics (such as a bug fix or a simple feature) to get acquainted with the codebase.

4. **Engage with the Team:**

Join our daily standups and reach out via Slack/Teams for any questions. Robert Eriksen is available to help as your lead developer.

Welcome aboard! We're excited to have you on the Big Fam Festival App team and look forward to building a truly transformative festival experience together.

Let me know if you need any further clarifications or additional information.

DOCS

API Documentation

- Big Fam Festival App - Updated API Documentation

Last Updated: [Insert Date]

This documentation covers all major API endpoints for the Big Fam Festival App. Endpoints are designed with role-based access control (RBAC), real-time synchronization, and offline support in mind. The following sections include endpoints for:

- Authentication & User Profile Management
- Schedule Builder & Management
- Interactive Map & Embedded Navigation
- Friend Management & Campsite Sharing
- Ticketing Management
- Admin-Specific Endpoints (Stage Schedule, Marked Campsites, Attendee Locations, Flag Users)
- Vendor, Artist, and Staff Endpoints
- Request Medical & Follow-Up
- Shift Management (with new endpoints for multi-artist events and staff shift pickup/drop requests)
- Miscellaneous Endpoints (File Uploads)

1. Authentication & User Profile Management

1.1 Login Endpoint

POST /api/v1/users/login

Description: Authenticates a user and returns a JWT with user details, including their role (admin, staff, artist, vendor, volunteer, director).

Request Payload:

```
1 json
```

Copy

```
{ "email": "user@example.com", "password": "password123" }
```

Response Example:

```
1 json
```

Copy

```
{ "token": "jwt_token_here", "user": { "id": "uuid", "name": "User Name", "role": "admin" } }
```

1.2 Get User Profile

GET /api/v1/users/{userId}

Description: Retrieves the user profile, including sharing preferences (shareMyCampsite, shareMyLocation).

Response Example:

```
1 json
```

Copy

```
{ "id": "uuid", "name": "User Name", "email": "user@example.com", "phone": "123-456-7890", "shareMyCampsite": true, "shareMyLocation": false, "role": "artist" }
```

1.3 Update User Profile

PUT /api/v1/users/{userId}/profile

Description: Updates user profile details, including toggles for sharing sensitive location data.

Request Payload:

```
1 json
```

Copy

```
{ "name": "Updated Name", "email": "newemail@example.com", "phone": "123-456-7890", "shareMyCampsite": true, "shareMyLocation": true }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Profile updated successfully", "profile": { /* updated profile details */ } }
```

2. Schedule Builder & Management

2.1 Retrieve Personal Schedule

GET /schedule/{userId}

Description: Retrieves the personal schedule (list of events added) for the specified user.

Response Example:

```
1 json
```

Copy

```
{ "userId": "uuid", "schedule": [ { "event_id": "event_uuid", "event_name": "Main Stage Performance", "start_time": "2025-06-20T18:00:00Z", "end_time": "2025-06-20T20:00:00Z", "stage": "Main Stage" } ] }
```

2.2 Add Event to Personal Schedule

POST /schedule

Description: Adds an event to the user's personal schedule.

Request Payload Example:

```
1 json
```

Copy

```
{ "userId": "uuid", "event_id": "event_uuid" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Event added to schedule." }
```

3. Interactive Map & Embedded Navigation

3.1 Retrieve POIs

GET /map/pois

Description: Retrieves all Points of Interest (POIs) including stages, vendors, facilities, campsites, and friend locations.

Response Example:

```
1 json
```

Copy

```
{ "pois": [ { "type": "stage", "id": "stage_uuid", "name": "Main Stage", "location": { "lat": 42.3314, "long": -83.0458 }, "description": "North Field, near entrance" }, { "type": "vendor", "id": "vendor_uuid", "name": "Food Truck X", "location": { "lat": 42.3320, "long": -83.0460 }, "description": "Delicious street food" }, { "type": "facility", "facility_type": "medical", "id": "facility_uuid", "name": "First Aid Tent", "location": { "lat": 42.3310, "long": -83.0445 }, "description": "Medical assistance available 24/7" }, { "type": "campsite", "id": "campsite_uuid", "userId": "uuid", "location": { "lat": 42.3300, "long": -83.0430 } }, { "type": "friend_campsite", "id": "campsite_uuid", "userId": "friend_uuid", "name": "Friend A's Campsite", "location": { "lat": 42.3305, "long": -83.0425 } } ] // Additional POIs... ] }
```

3.2 Embedded Navigation

GET /navigation?destination=lat,long

Description: Returns route information (turn-by-turn directions, distance, estimated time) from the user's current location to the specified destination. Supports offline routing using cached map data.

Response Example:

```
1 json
```

Copy

```
{ "route": { "steps": [ { "instruction": "Head north on Main St.", "distance": "200m" }, { "instruction": "Turn right onto Festival Rd.", "distance": "150m" } ], "total_distance": "350m", "estimated_time": "5 minutes" } }
```

4. Friend Management & Campsite Sharing

4.1 Send Friend Request

POST /friend_requests

Description: Sends a friend request to another user based on email or phone number.

Request Payload Example:

```
1 json
```

Copy

```
{ "target": "friend@example.com" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Friend request sent.", "request_id": "friend_req_uuid" }
```

4.2 Retrieve Friend List

GET /friends

Description: Retrieves a list of all accepted friends for the authenticated user.

Response Example:

```
1 json
```

Copy

```
{ "friends": [ { "user_id": "friend_uuid", "name": "Friend A", "profile_picture_url": "https://images.bigfamfestival.com/profiles/friendA.jpg" } ] }
```

4.3 Mark My Campsite

POST /campsites

Description: Marks the user's current campsite location. Optionally, users can choose to share this location with

friends.

Request Payload Example:

```
1 json
```

Copy

```
{ "userId": "uuid", "location_lat": 42.3300, "location_long": -83.0430, "shared_with_friends": true }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Campsite marked successfully." }
```

4.4 Retrieve Friend Campsites

GET /friends/campsites

Description: Retrieves shared campsite locations of friends.

Response Example:

```
1 json
```

Copy

```
{ "campsites": [ { "user_id": "friend_uuid", "name": "Friend A's Campsite", "location": { "lat": 42.3305, "long": -83.0425 }, "profile_picture_url": "https://images.bigfamfestival.com/profiles/friendA.jpg" } ] }
```

4.5 Retrieve Friend Last Reported Locations

GET /friends/locations

Description: Retrieves the last reported locations of friends, using their profile pictures as markers.

Response Example:

```
1 json
```

Copy

```
{ "locations": [ { "user_id": "friend_uuid", "name": "Friend A", "location": { "lat": 42.3320, "long": -83.0450 }, "last_updated": "15 minutes ago", "profile_picture_url": "https://images.bigfamfestival.com/profiles/friendA.jpg" } ] }
```

5. Ticketing Management

5.1 Validate Ticket (QR Code Scanning)

POST /tickets/validate

Description: Validates a scanned ticket's QR code by checking if the ticket exists, is valid, and has not been redeemed.

Request Payload Example:

```
1 json
```

Copy

```
{ "ticket_code": "QR123ABC" }
```

Response Example:

```
1 json
```

Copy

```
{ "ticket_code": "QR123ABC", "valid": true, "is_redeemed": false }
```

5.2 Offline Ticket Sync

POST /tickets/sync

Description: Synchronizes offline ticket redemption data with the backend once connectivity is restored.

Request Payload Example:

```
1 json
```

Copy

```
{ "redemptions": [ { "ticket_code": "QR123ABC", "redeemed_at": "2025-06-01T12:00:00Z", "processed_by": "admin_uuid" } ] }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Ticket redemptions synchronized successfully." }
```

5.3 Retrieve Ticket Audit Log (Admin Only)

GET /admin/tickets/audit

Description: Retrieves the audit log of ticket redemptions.

Response Example:

```
1 json
```

Copy

```
{ "audit_log": [ { "ticket_id": "ticket_uuid", "ticket_code": "QR123ABC", "processed_by": "admin_uuid", "redeemed_at": "2025-06-01T12:00:00Z" } ] }
```

6. Admin-Specific Endpoints

6.1 Modify Stage Schedule (Admin Only)

PUT /stages/{stageId}/schedule

Description: Allows admins to update the schedule for a specific stage. Changes are logged in the schedule_change_log. **Request Payload Example:**

```
1 json
```

Copy

```
{ "event_id": "event_uuid", "new_start_time": "2025-06-20T18:30:00Z", "new_end_time": "2025-06-20T20:30:00Z", "change_reason": "Delayed due to technical issues" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Stage schedule updated successfully." }
```

6.2 Retrieve All Marked Campsites with Attendee Details (Admin Only)

GET /admin/campsites

Description: Retrieves a list of all marked campsites along with detailed attendee information (name, role, contact info, account notes, and location).

Response Example:

```
1 json
```

Copy

```
{ "campsites": [ { "user_id": "uuid", "name": "Jane Doe", "role": "attendee", "contact_info": "555-123-4567", "account_notes": "VIP guest", "location": { "lat": 42.3300, "long": -83.0430 }, "marked_at": "2025-06-01T12:00:00Z" } ] }
```

6.3 Retrieve All Attendees' Last Reported Positions (Admin Only)

GET /admin/attendee-locations

Description: Returns the latest reported locations of all attendees, including profile pictures, names, contact info, and timestamps. **Response Example:**

```
1 json
```

Copy

```
{ "attendee_locations": [ { "user_id": "uuid", "name": "Jane Doe", "location": { "lat": 42.3320, "long": -83.0440 }, "last_updated": "5 minutes ago", "profile_picture_url": "https://images.bigfamfestival.com/profiles/jane.jpg", "contact_info": "555-123-4567" } ] }
```

6.4 Flag a User for Monitoring (Admin Only)

POST /admin/flag_user

Description: Allows an admin to flag a user for monitoring. When the flagged user's location updates, a push notification with a "Take Me Here" option is sent to the admin. **Request Payload Example:**

```
1 json
```

Copy

```
{ "flagged_user_id": "user_uuid", "flag_reason": "Suspicious behavior observed" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "User flagged for monitoring successfully.", "flag_id": "flag_uuid" }
```

7. Vendor, Artist, and Staff Endpoints

7.1 Vendor Profile Management

PUT /vendor_profiles/{userId}

Description: Allows vendors to update their business profile details including business name, description, operating hours, contact info, menu URL, and geolocation. **Request Payload Example:**

```
1 json
```

Copy

```
{ "business_name": "Food Truck X", "description": "Delicious street food.", "operating_hours": "12:00 PM - 10:00 PM", "contact_info": "555-987-6543", "menu_url": "https://bigfamfestival.com/menus/foodtruckx.pdf", "location_lat": 42.3320, "location_long": -83.0450 }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Vendor profile updated successfully." }
```

7.2 Artist Profile Management

PUT /artist_profiles/{userId}

Description: Enables artists to update their performance profiles, including profile picture, bio, social media links, and view their performance schedule. **Request Payload Example:**

```
1 json
```

Copy

```
{ "profile_picture_url": "https://images.bigfamfestival.com/profiles/artist123.jpg", "bio": "Up-and-coming indie artist.", "social_links": { "instagram": "https://instagram.com/artist123", "twitter": "https://twitter.com/artist123" } }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Artist profile updated successfully." }
```

7.3 Staff Panel (For Volunteers & Artist Relations)

GET /staff_profiles/{userId}

Description: Retrieves profile and shift information for non-admin staff (e.g., volunteers, artist relations). **Response Example:**

```
1 json
```

Copy

```
{ "user_id": "user_uuid", "staff_role": "volunteer", "shifts": [ { "date": "2025-06-20", "start_time": "09:00", "end_time": "17:00" } ], "contact_info": "volunteer@example.com" }
```

8. Offline & Embedded Navigation Endpoints

8.1 Navigation (Embedded, Offline-Capable)

GET /navigation?destination=lat,long

Description: Returns an in-app route with turn-by-turn directions from the current location to the specified destination. Supports offline routing using cached map tiles. **Response Example:**

```
1 json
```

Copy

```
{ "route": { "steps": [ { "instruction": "Head north on Main St.", "distance": "200m" }, { "instruction": "Turn right onto Festival Rd.", "distance": "150m" } ], "total_distance": "350m", "estimated_time": "5 minutes" } }
```

9. Miscellaneous Endpoints

9.1 File Upload (Profile Picture, Menu Uploads)

POST /uploads

Description: Handles file uploads for profile pictures, menus, and other assets. **Request:** Multipart/form-data with file data. **Response Example:**

```
1 json
```

Copy

```
{ "file_url": "https://cdn.bigfamfestival.com/uploads/file123.jpg" }
```

10. Additional Endpoints for New Functionalities

10.1 Request Medical

POST /emergency

Description: Submits a medical request with an urgency level, optional description, and the user's current geolocation.

Request Payload Example:

```
1 json
```

Copy

```
{ "urgency": "IMMEDIATE", "description": "Severe injury near the stage", "location": { "latitude": 42.123456, "longitude": -83.123456 } }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Medical request sent successfully", "requestId": "uuid" }
```

10.2 Update Medical Request Follow-Up

POST /emergency/followup

Description: Processes follow-up actions for a medical request. The user chooses either "Take Me To Medical" or "Stay Put" (with nearby facilities displayed). **Request Payload Example:**

```
1 json
```

Copy

```
{ "requestId": "uuid", "action": "TAKE_ME_TO_MEDICAL", // or "STAY_PUT" "destination": { "latitude": 42.125000, "longitude": -83.125000 } }
```

Response Example (if TAKE_ME_TO_MEDICAL):

```
1 json
```

Copy

```
{ "route": { "steps": [ { "instruction": "Head north on Main St.", "distance": "200m" }, { "instruction": "Turn right onto Festival Rd.", "distance": "150m" } ], "total_distance": "350m", "estimated_time": "5 minutes" }, "message": "Navigation initiated to the nearest medical facility." }
```

Response Example (if STAY_PUT):

```
1 json
```

Copy

```
{ "nearby_facilities": [ { "facility_id": "facility_uuid", "name": "First Aid Tent", "location": { "lat": 42.3310, "long": -83.0445 }, "distance": "200m", "estimated_time": "3 minutes" }, { "facility_id": "facility_uuid2", "name": "Medical Station Bravo", "location": { "lat": 42.3300, "long": -83.0450 }, "distance": "350m", "estimated_time": "5 minutes" } ], "message": "Nearby medical facilities retrieved." }
```

10.3 Multi-Artist Event Support

POST /events

Description: Creates a new event that must have at least one artist assigned and can include multiple artists.

Request Payload Example:

```
1 json
```

Copy

```
{ "event_name": "Evening Concert", "stage": "Main Stage", "date": "2025-06-20", "start_time": "19:00", "end_time": "21:00", "artists": ["artist_uuid1", "artist_uuid2"] // Must include at least one artist }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Event created successfully", "event": { "event_id": "event_uuid", "event_name": "Evening Concert", "stage": "Main Stage", "date": "2025-06-20", "start_time": "19:00", "end_time": "21:00", "artists": ["artist_uuid1", "artist_uuid2"] } }
```

11. Shift Management Endpoints (Extended for Directors/Staff)

11.1 Retrieve Open Shifts for Staff

GET /shifts/open

Description: Retrieves a list of unassigned open shifts that do not conflict with the staff's current schedule.

Response Example:

```
1 json
```

Copy

```
{ "open_shifts": [ { "shift_id": "shift_uuid1", "what": "Setup Crew", "where": "Main Tent", "when": { "startTime": "2025-06-20T09:00:00Z", "duration": 240 } } ] }
```

11.2 Claim an Open Shift (Staff)

POST /shifts/{shiftId}/claim

Description: Allows a staff member to claim an open shift, provided it does not conflict with their existing shifts.

Request Payload Example:

```
1 json
```

Copy

```
{ "userId": "staff_uuid" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Shift claimed successfully", "shift": { /* updated shift details with assigned user */ } }
```

11.3 Request to Drop a Shift (Staff)

POST /shifts/{shiftId}/drop_request

Description: Allows a staff member to request dropping an assigned shift. This request is sent to the Director responsible for the staff member. **Request Payload Example:**

```
1 json
```

Copy

```
{ "userId": "staff_uuid", "reason": "Scheduling conflict" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Drop request submitted successfully", "requestId": "drop_req_uuid" }
```

11.4 Shift Management for Directors/Administrators (Extended)Add Shift Endpoint

POST /shifts

Description: Creates a new shift assignment. The "Who" field is optional; if provided, the shift is immediately assigned to that staff member. **Request Payload Example:**

```
1 json
```

Copy

```
{ "who": "staff_email@example.com", // Optional "what": "Setup Crew", "where": "Main Tent", "why": "Stage Setup", "when": { "startTime": "2025-06-20T09:00:00Z", "duration": 240 } }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Shift created successfully", "shift": { /* shift details */ } }
```

Update Shift Endpoint

PUT /shifts/{shiftId}

Description: Updates details of an existing shift. **Request Payload Example:**

```
1 json
```

Copy

```
{ "what": "Updated Setup Crew", "where": "Main Tent", "why": "Revised Stage Setup", "when": { "startTime": "2025-06-20T09:30:00Z", "duration": 180 } }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Shift updated successfully", "shift": { /* updated shift details */ } }
```

Delete Shift Endpoint

DELETE /shifts/{shiftId}

Description: Deletes a shift assignment. **Response Example:**

```
1 json
```

Copy

```
{ "message": "Shift deleted successfully" }
```

Transfer Shift Endpoint

POST /shifts/{shiftId}/transfer

Description: Transfers a shift to another staff member. **Request Payload Example:**

```
1 json
```

Copy

```
{ "newStaff": "new_staff@example.com" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Shift transferred successfully", "shift": { /* updated shift details */ } }
```

12. Volunteer Management Endpoints

12.1 Get Volunteer Shifts

GET /volunteers/shifts

Description: Retrieves a list of volunteer shifts assigned to the user. **Response Example:**

```
1 json
```

Copy

```
[ { "id": "shift_uuid", "role": "Food Stall Greeter", "date": "2025-06-20", "start_time": "10:00", "end_time": "14:00", "location": "Food Court", "status": "Upcoming" } ]
```

12.2 Update a Volunteer Shift

PUT /volunteers/shifts/{shiftId}

Description: Updates the status of a volunteer shift (e.g., mark as completed). **Request Payload Example:**

```
1 json
```

Copy

```
{ "status": "Completed" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Volunteer shift updated successfully" }
```

12.3 Get Volunteer Messages

GET /volunteers/messages

Description: Retrieves internal messages/announcements for volunteers. **Response Example:**

```
1 json
```

Copy

```
[ { "id": "msg_uuid", "sender": "Festival Organizer", "content": "Please report to the main tent for a briefing at 9:00 AM.", "timestamp": "2025-06-20T08:45:00Z" } ]
```

12.4 Update Volunteer Profile

PUT /volunteers/profile

Description: Updates the volunteer's personal information and availability. **Request Payload Example:**

```
1 json
```

Copy

```
{ "name": "Volunteer Name", "email": "volunteer@example.com", "phone": "123-456-7890", "availability": "Evenings" }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Volunteer profile updated successfully", "profile": { /* updated profile details */ } }
```

13. Conclusion

This updated API documentation now includes all our new decisions and wireframe details for the Big Fam Festival App framework. Key updates include:

- **Multi-Artist Association for Events:**

The event creation endpoint now accepts an array of artist IDs (with at least one required).

- **Enhanced Shift Management for Staff:**

New endpoints allow staff to retrieve unassigned open shifts, claim open shifts, and request to drop shifts (with drop requests sent to the assigned Director).

- **Expanded Functionalities:**

All other previously defined endpoints remain in place to support authentication, profile management, interactive navigation, friend management, ticketing, vendor/artist/staff operations, request medical, and volunteer management.

Each endpoint is designed with security (RBAC), real-time synchronization, and offline support in mind.

Please review this documentation and provide any feedback or additional endpoint requirements as necessary.

Let me know if you need further modifications or additional details!

- You said:

For volunteers add a value called remainingHours to each user and when a shift has changed its status to "Completed" that has been assigned to that volunteer, subtract the duration of that shift from their remaining hours

- Reasoned for 10 seconds

Below is an updated section of the API documentation that adds a new property, **remainingHours**, to each volunteer's profile. When a volunteer's assigned shift is marked as "Completed," the duration of that shift is subtracted from their remainingHours. This update affects the volunteer profile endpoints as well as the volunteer shift update endpoint.

Updated Volunteer Management Endpoints

12. Volunteer Profile Enhancements

Each volunteer now has a **remainingHours** field in their profile, which tracks the total number of hours remaining for the volunteer (for example, a weekly quota or total commitment hours).

12.4 Update Volunteer Profile

PUT /volunteers/profile

Description: Updates the volunteer's personal information, availability, and remainingHours.

Request Payload Example:

```
1 json
```

Copy

```
{ "name": "Volunteer Name", "email": "volunteer@example.com", "phone": "123-456-7890", "availability": "Evenings", "remainingHours": 20 // New field representing the volunteer's current remaining hours }
```

Response Example:

```
1 json
```

Copy

```
{ "message": "Volunteer profile updated successfully", "profile": { "user_id": "volunteer_uuid", "name": "Volunteer Name", "email": "volunteer@example.com", "phone": "123-456-7890", "availability": "Evenings", "remainingHours": 20 } }
```

12.2 Update a Volunteer Shift

PUT /volunteers/shifts/{shiftId}

Description: Updates the status of a volunteer shift (for example, marking it as "Completed"). When a shift is updated to "Completed," the duration of that shift is subtracted from the volunteer's remainingHours.

Request Payload Example:

```
1 json
```

Copy

```
{ "status": "Completed" }
```

Note: The backend calculates the shift duration (in hours) based on the shift's start and end times and automatically subtracts that value from the volunteer's current remainingHours.

Response Example:

```
1 json
```

Copy

```
{ "message": "Volunteer shift updated successfully", "shift": { "id": "shift_uuid", "role": "Food Stall Greeter", "date": "2025-06-20", "start_time": "10:00", "end_time": "14:00", "location": "Food Court", "status": "Completed", "duration": 4 // Duration in hours }, "updatedVolunteerProfile": { "user_id": "volunteer_uuid", "remainingHours": 16 // Updated remainingHours after subtracting shift duration } }
```

12.1 Get Volunteer Profile

GET /volunteers/profile/{userId}

Description: Retrieves the volunteer's profile information, now including the **remainingHours** value.

Response Example:

```
1 json
```

Copy

```
{ "user_id": "volunteer_uuid", "name": "Volunteer Name", "email": "volunteer@example.com", "phone": "123-456-7890", "availability": "Evenings", "remainingHours": 20 }
```

Summary

- **Volunteer Profiles:** Now include a new field, **remainingHours**, which tracks the number of volunteer hours remaining.
- **Shift Completion:** When a volunteer shift is updated to "Completed" via the **PUT /volunteers/shifts/{shiftId}** endpoint, the backend calculates the duration of that shift and subtracts it from the volunteer's remainingHours.
- **Profile Retrieval & Updates:** Endpoints for getting and updating volunteer profiles now incorporate this new field, allowing the application to display up-to-date remaining volunteer hours.

Open API (Swagger) Docs

Big Fam Festival API

1.0.0 OAS 3.0

API for the Big Fam Festival App covering authentication, user profile management, schedule builder and management, interactive map & embedded navigation, friend management & campsite sharing, ticketing management, admin-specific endpoints, vendor, artist, staff, volunteer management, request medical & follow-up, extended shift management (including multi-artist support), and additional functionalities such as vendor menu item management and event updates.

Servers

<https://api.bigfamfestival.com>

[Authorize](#)

User

POST [`/users/login`](#) User Login

GET [`/users/{user_id}`](#) Get User Profile

PUT [`/users/{user_id}/profile`](#) Update User Profile

PUT [`/users/{user_id}/campsite`](#) Update User Campsite

Schedule

GET [`/users/{user_id}/schedule`](#) Retrieve Personal Schedule

POST [`/users/{user_id}/schedule`](#) Add Event to Personal Schedule

DELETE [`/users/{user_id}/schedule`](#) Remove Event from Personal Schedule

Map

GET [`/map/pois`](#) Retrieve POIs

GET [`/navigation`](#) Embedded Navigation

Friends

POST [`/friend_requests`](#) Send Friend Request

GET [`/friends`](#) Retrieve Friend List

Ticketing

POST [`/tickets/validate`](#) Validate Ticket (QR Code Scanning)

POST [`/tickets-sync`](#) Offline Ticket Sync

GET /admin/tickets/audit Retrieve Ticket Audit Log (Admin Only) ▼

Admin

PUT /stages/{stageId}/schedule Modify Stage Schedule (Admin Only) 🔒 ▼

GET /admin/campsites Retrieve All Marked Campsites with Attendee Details (Admin Only) ▼

GET /admin/attendee-locations Retrieve All Attendees' Last Reported Positions (Admin Only) ▼

POST /admin/flag_user Flag a User for Monitoring (Admin Only) ▼

Vendor

PUT /vendor_profiles/{user_id} Update Vendor Profile ▼

GET /vendor_profiles/{user_id}/menu_items Retrieve Menu Items (Vendor Only) ▼

POST /vendor_profiles/{user_id}/menu_items Add Menu Item (Vendor Only) ▼

PUT /vendor_profiles/{user_id}/menu_items/{menu_item_id} Update Menu Item (Vendor Only) ▼

DELETE /vendor_profiles/{user_id}/menu_items/{menu_item_id} Delete Menu Item (Vendor Only) ▼

Artist

PUT /artist_profiles/{user_id} Artist Profile Management ▼

Events

GET /events Retrieve Events ▼

POST /events Create New Event (Admin Only) 🔒 ▼

PUT /events/{event_id} Update Event (Admin Only) 🔒 ▼

DELETE /events/{event_id} Delete Event (Admin Only) 🔒 ▼

Volunteer

GET /volunteers/shifts Get Volunteer Shifts ▼

PUT /volunteers/shifts/{shiftId} Update Volunteer Shift ▼

GET /volunteers/messages Get Volunteer Messages ▼

PUT /volunteers/profile Update Volunteer Profile ▼

Schemas

^

User

Event

Ticket

Vendor

MenuItem

Artist

Volunteer

Shift

Infrastructure Setup to Deployment & Monitoring

Below is the updated "Infrastructure Setup to Deployment & Monitoring" document for Confluence. This document covers our cloud architecture, database and caching setup, content delivery, deployment pipeline, and monitoring strategies. It is designed to be the single source of truth for our infrastructure and operations—from initial setup to post-launch monitoring.

Last updated: Feb 5, 2025 by [@Robert Eriksen](#)

1. Overview

The Big Fam Festival App is built on a robust, scalable, cloud-native architecture designed to support 4000+ concurrent users while providing seamless real-time and offline functionality. This document outlines our end-to-end infrastructure strategy—from cloud resource provisioning and database configuration through to deployment, continuous integration, and monitoring.

2. Cloud Environment & Infrastructure

2.1. Cloud Provider & Deployment Model

- **Platform:**

We deploy our app on a leading cloud provider (AWS or GCP), leveraging managed services and scalable architectures.

- **Compute Options:**

- **Kubernetes:**

Deploy containerized microservices using AWS EKS or GCP GKE for greater control and orchestration.

- **Serverless:**

Where appropriate, utilize serverless functions (e.g., AWS Lambda or GCP Cloud Functions) to optimize resource usage and simplify scaling.

- **Auto-Scaling & Load Balancing:**

- Configure auto-scaling policies to handle fluctuating load, ensuring the system can dynamically scale to support peak loads.

- Use load balancers (e.g., AWS ELB or GCP Load Balancing) to distribute traffic evenly across instances.

2.2. Database & Caching Infrastructure

- **Primary Database:**

- **Managed PostgreSQL:**

Use AWS RDS or GCP Cloud SQL for structured data storage (user data, schedules, ticketing, etc.). Ensure regular backups and high-availability configurations.

- **Caching Layer:**

- **Redis:**

Leverage AWS ElastiCache or GCP Memorystore for fast in-memory caching of frequently accessed data, reducing load on the primary database.

- **Real-Time & Offline Data:**

- **Firestore/DynamoDB:**

Implement NoSQL databases (Firestore for GCP or DynamoDB for AWS) to handle real-time data needs, such as notifications, friend locations, and offline synchronization.

3. Content Delivery & Edge Caching

- **CDN Setup:**

- Use CloudFront (AWS) or Cloud CDN (GCP) to serve static assets (images, JavaScript, CSS) globally.
 - Edge caching minimizes latency, ensuring fast asset delivery even during peak traffic periods.
-

4. Continuous Integration & Deployment (CI/CD)

4.1. CI/CD Pipeline

- **Tools:**

- Leverage GitHub Actions, GitLab CI/CD, or similar tools to automate build, test, and deployment processes.

- **Pipeline Stages:**

- a. **Code Linting & Unit Testing:**

Run static analysis and unit tests on every commit.

- b. **Integration & End-to-End Testing:**

Execute integration tests (using Jest, Cypress, Detox, etc.) to validate cross-module interactions.

- c. **Staging Deployment:**

Automatically deploy to a staging environment for QA and performance testing.

- d. **Production Deployment:**

Upon approval, deploy to the production environment using blue/green or canary strategies to minimize downtime.

4.2. Infrastructure as Code

- **Tools:**

- Use Terraform or Pulumi for provisioning and managing cloud resources, ensuring consistency across environments.
-

5. Monitoring, Logging & Stress Testing

5.1. Real-Time Monitoring

- **Monitoring Tools:**

- Deploy Prometheus and Grafana for real-time performance monitoring and alerting.
- Alternatively, use native solutions like AWS CloudWatch or GCP Stackdriver.

- **Key Metrics:**

- Monitor CPU, memory, and network usage.
- Track API response times, error rates, and database query performance.
- Monitor user load and concurrent connections during peak periods.

5.2. Logging & Alerting

- **Centralized Logging:**

- Implement centralized logging using ELK Stack (Elasticsearch, Logstash, Kibana) or cloud-native logging solutions to capture logs from all microservices.

- **Alerting:**

- Configure alerts for critical events (e.g., high error rates, resource exhaustion) to trigger automated notifications via email, SMS, or messaging platforms.

5.3. Load & Stress Testing

- **Testing Tools:**

- Use k6, Apache JMeter, or similar tools to simulate 4000+ concurrent users.

- **Objectives:**

- Validate system performance under load.
 - Identify bottlenecks in scaling, database queries, or API responses.
 - Optimize auto-scaling and caching configurations based on test results.
-

6. Deployment Strategy

6.1. Production Deployment

- **Deployment Models:**

- Use container orchestration (Kubernetes) or serverless deployments to ensure minimal downtime.

- **DNS & SSL:**

- Ensure proper DNS routing and SSL certificate management for secure communication.

- **Rollout Strategy:**

- Employ blue/green or canary deployments to gradually shift traffic and monitor system stability before full production rollout.

6.2. Post-Deployment Monitoring & Maintenance

- **Health Checks:**

- Continuous health checks on services and automated restarts if a service becomes unresponsive.

- **Feedback Loop:**

- Regular performance reviews and incident reports to refine auto-scaling, caching, and overall system performance.

- **User Feedback & Iteration:**

- Monitor user experience and application logs to prioritize iterative improvements.
-

7. Summary & Next Steps

This document outlines the complete infrastructure strategy for the Big Fam Festival App, from cloud setup and database configuration to CI/CD pipelines, monitoring, stress testing, and production deployment. The key points include:

- **Scalable Cloud Infrastructure:**

Auto-scaling, load balancing, and high-availability configurations on AWS/GCP.

- **Robust Data Management:**

Managed PostgreSQL, Redis caching, and NoSQL solutions for real-time data and offline support.

- **Optimized Content Delivery:**

Global distribution via CDN and edge caching.

- **Automated CI/CD & IaC:**

Streamlined development and deployment through continuous integration and infrastructure as code.

- **Comprehensive Monitoring & Testing:**

Real-time dashboards, centralized logging, and stress testing to ensure performance under peak loads.

- **Deployment & Post-Launch Management:**

Strategies to minimize downtime and ensure rapid response to any issues.

Next Steps:

1. Finalize and approve this document with all stakeholders.
2. Update Confluence with the latest version of this document.
3. Use this document to guide the implementation of Epics in Jira.
4. Begin provisioning infrastructure and setting up CI/CD pipelines as detailed.

This infrastructure setup to deployment and monitoring strategy is critical to ensuring that the Big Fam Festival App performs reliably and scales effectively during the festival. Let me know if you require any further adjustments or additional details.

UI/UX

COMP: Top Nav

Purpose:

Provide a clean, minimal top bar that reinforces brand identity, enables content search, and facilitates user authentication.

Layout Elements:

- **Left:** Big Fam Logo
- **Center:** Search Bar (for searching artists, vendors, stages, etc.)
- **Right:**
 - If authenticated: User Profile Picture (clickable, leads to Profile Editor)
 - If unauthenticated: LOGIN button (clickable, leads to Login/Registration page)

ASCII Diagram:

```
1 +---+ | [Big Fam Logo] [🔍 Search Artist, Vendor, Stage...] [Profile Icon] or [LOGIN] | +---+
```

Description:

- **Left:** The Big Fam Logo is always visible for brand recognition.
- **Center:** A prominent search bar that expands or provides auto-suggestions.
- **Right:** The user's profile picture (if logged in) or a LOGIN button (if not logged in), enabling quick access to account settings or authentication.

COMP: Bottom Nav

Purpose:

Offer quick access to the app's core features while keeping the interface uncluttered by consolidating overflow options under a "More" tab. We have now re-added the Notifications tab to ensure users can quickly view real-time alerts.

Primary Items:

- **Home:** Global Landing Page (overview of live/upcoming events)
- **My Schedule:** Personal event itinerary and schedule builder
- **Map:** Interactive map with embedded navigation
- **Notifications:** Real-time alerts and event updates
- **More:** Overflow menu containing additional functions

"More" Tab Contents:

- Friends
- Vendors
- Artists
- Stages

ASCII Diagram:

```
1 +---+ | [🏠 Home] | [❤️ My Schedule] | [📍 Map] | [🔔 Notifications] | [... More] | +---+
```

Description:

- **Home:** Provides immediate access to the Global Landing Page, showcasing live and upcoming events.
- **My Schedule:** Directs users to their personal schedule builder, where they can plan and manage their festival itinerary.
- **Map:** Opens the interactive map featuring all Points of Interest with embedded navigation.
- **Notifications:** Displays real-time notifications and alerts, ensuring users are promptly informed of updates and changes.
- **More:** Consolidates additional functions (Friends, Vendors, Artists, Stages and Tickets) to keep the main navigation uncluttered.

Within the "More" Menu – Dynamic Panel Option ↗

• Dynamic Panel Option:

- **The More menu will display a single role-specific panel link based on the current user's role. This link adapts as follows:**
 - **Admin:**
 - **Label:** "Admin Panel"
 - **Icon:** A gear, shield, or lock icon
 - **Link:** Navigates to /admin-panel
 - **Staff:**
 - **Label:** "Staff Panel"
 - **Icon:** A group or briefcase icon
 - **Link:** Navigates to /staff-panel

- **Artist:**
 - **Label:** “Artist Panel”
 - **Icon:** A microphone or palette icon
 - **Link:** Navigates to /artist-panel
- **Vendor:**
 - **Label:** “Vendor Panel”
 - **Icon:** A shopping bag or storefront icon
 - **Link:** Navigates to /vendor-panel
- **Request Medical:**
 - **Styling:**
 - Displayed in red text with an emergency icon (e.g., red cross) to indicate urgency.
 - **Action:**
 - Tapping this option opens the same Request Medical modal as described above.

- **Implementation (Pseudo-code Example in React):**

```

1 const RoleBasedPanelLink = ({ user }) => {
2   if (user.role === 'admin') {
3     return (
4       <NavLink to="/admin-panel">
5         <Icon name="gear" /> Admin Panel
6       </NavLink>
7     );
8   } else if (user.role === 'staff') {
9     return (
10       <NavLink to="/staff-panel">
11         <Icon name="group" /> Staff Panel
12       </NavLink>
13     );
14   } else if (user.role === 'artist') {
15     return (
16       <NavLink to="/artist-panel">
17         <Icon name="microphone" /> Artist Panel
18       </NavLink>
19     );
20   } else if (user.role === 'vendor') {
21     return (
22       <NavLink to="/vendor-panel">
23         <Icon name="store" /> Vendor Panel
24       </NavLink>
25     );
26   }
27   return null;
28 };
29 
```

ASCII Diagram Example:

```

1 +-----+
2 | More Options          |
3 |-----|
4 | • Friends            |
5 | • Vendors            |
6 | • Artists            |
7 | • Stages              |
8 | • [ ! Request Medical] |

```

```
9 | • [⚙️ Dynamic Panel Option]      | <-- shows "Admin Panel" for admins, "Staff Panel" for staff, etc.  
10 +-----+  
11
```

- **Description:**

- **Role-Based Visibility:**

The dynamic panel option is rendered conditionally based on the user's role (retrieved via JWT or profile data). Only the relevant panel link is displayed.

- **Seamless Navigation:**

Tapping the role-specific panel option navigates the user directly to their dedicated dashboard (e.g., Admin, Staff, Artist, or Vendor Panel), ensuring that each user accesses only the functions that are pertinent to their role.

- **Interface Cleanliness:**

This approach avoids clutter in the More menu by providing a single, dynamic option rather than multiple redundant links. Non-admin users will not see the admin panel link, for example.

Benefits ↗

- **Clarity:**

Users see only the panel option that is relevant to their role, reducing confusion.

- **Efficiency:**

A single dynamic link simplifies the More menu while providing quick access to role-specific functionality.

- **Security:**

Role-based rendering coupled with backend RBAC ensures that users can only access panels they are authorized to view.

This dynamic approach strikes a balance between accessibility for admins and a clean, focused interface for all users. Let me know if you need any further modifications or additional details!

PAGE: My Schedule

Last Updated: [@Robert Eriksen](#) Feb 11, 2025

1. Overview

The Personal Schedule Builder page allows users to manage their personal festival itinerary. Users can add events to their schedule from anywhere using the heart icon, and remove events by clicking the heart again in their My Schedule view. Instead of having a nested "Available Events" section, users will be directed to the global Events page via a link.

Key Features:

- **My Schedule View:** Displays events that the user has added.
 - **Event Addition/Removal via Heart Icon:** Users click the heart icon on event cards to add events to their schedule; clicking it again in My Schedule will remove the event.
 - **Link to Global Events Page:** A dedicated link/button allows users to browse all available events.
 - **Offline Support:** The schedule is cached locally for offline access and later synchronization.
 - **Clean Layout:** The search bar is omitted here since it is available in the persistent top navigation.
-

2. UI Components & Layout

A. Header (Global - always present)

• Elements:

- **Left:** Big Fam Logo
- **Center:** Search Bar (persistent across the app)
- **Right:** Profile Icon (or LOGIN button)

B. Personal Schedule Builder Page

1. Page Header

- **Title:** "My Schedule"
- **Subheader / Link:** A link/button labeled "Browse Events" that directs users to the global Events page.

2. Schedule Content Area

• Schedule List:

- A scrollable list (or calendar view) displaying all events that the user has added.
- Each event is represented by a card that includes:
 - **Event Image:** (e.g., artist image)
 - **Event Title:** (e.g., "DJ Nightwave")
 - **Details:** Stage, date, and time (e.g., "Stage A - 7:30 PM")
 - **Heart Icon:**
 - In the global Events page, tapping the heart icon adds the event to the schedule.
 - In the My Schedule view, the heart icon is active; tapping it again will remove the event from the schedule.

3. Floating Action Button (FAB)

• Share My Schedule:

- A floating button in the bottom-right corner that, when tapped, generates a snapshot of the current schedule for social sharing.

C. Interaction Flow ↴

1. Adding an Event:

- User browses events (using the global search or on the global Events page).
- When an event card is displayed, the heart icon is visible.
- Tapping the heart icon on an event card immediately adds the event to the user's schedule.

2. Removing an Event:

- In the My Schedule view, events appear with an active heart icon.
- Tapping the heart icon on a scheduled event removes it from the schedule.

3. Browsing Events:

- A "Browse Events" button/link is present on the My Schedule page header that navigates users to the global Events page.

4. Sharing the Schedule:

- Tapping the FAB labeled "Share My Schedule" generates a visual snapshot for social sharing.

3. Updated Wireframe Diagrams ↴

A. Personal Schedule Builder Page (My Schedule View) ↴

```

1 +-----+
2 | [←] My Schedule [X] |
3 +-----+
4 | [Day Filter][Stage Filter][Browse Events] |
5 +-----+
6 |
7 | [Scheduled Event Card] |
8 | +-----+ |
9 | | [Event Image] | |
10| | DJ Nightwave | [Heart Icon (Active)] | |
11| | Stage A - 7:30 PM | |
12| | (Tap Heart to Remove) | |
13| +-----+ |
14|
15| [Scheduled Event Card] |
16| +-----+ |
17| | [Event Image] | |
18| | Indie Sparks | [Heart Icon (Active)] | |
19| | Stage B - 9:00 PM | |
20| | (Tap Heart to Remove) | |
21| +-----+ |
22|
23+-----+
24| [Floating Action Button: Share My Schedule] |
25+-----+

```

4. Summary ↴

• Personal Schedule Page:

- Displays the user's selected events in a clear, scrollable list.
- Includes a "Browse Events" link to the global Events page.
- Features an active heart icon for each scheduled event, allowing users to remove events.

- Contains a floating action button for sharing the schedule.

This updated UI/UX design streamlines event management by separating the browsing and scheduling functions. It leverages a global search, a dedicated "Browse Events" link, and intuitive heart icon interactions to add or remove events from the schedule—all while ensuring offline functionality and easy sharing.

PAGE: Admin Panel

1. Overview ↗

Purpose:

The Admin Panel provides festival administrators with a centralized dashboard to manage key operational functions, including:

- **Stage Schedule Modifications:** Update event times, order, and details for each stage with real-time validation and audit logging.
- **Campsite & Attendee Monitoring:** View comprehensive details for marked campsites and real-time attendee locations, with navigation options.
- **Flagged User Management:** Flag users for monitoring and receive push notifications when flagged users update their location.
- **Internal Messaging:** Access secure messaging features to communicate with staff and other key stakeholders.

Access:

- This panel is accessible only to users with the admin role.
 - The link to the Admin Panel is dynamically rendered in both the “More” menu and the Profile Editor for admin users.
-

2. Key UI Components ↗

A. Header & Global Navigation ↗

- **Header Elements:**
 - **Left:** Big Fam Festival logo (consistent with the rest of the app).
 - **Center:** Page Title ("Admin Panel Dashboard").
 - **Right:** Profile icon (shows logged-in admin's profile) with an option for settings.

B. Navigation Sidebar (Left Panel) ↗

- **Sidebar Menu Options:**
 - **Dashboard Home:** Overview and key metrics.
 - **Stage Schedules:** Interface for modifying stage schedules.
 - **Campsite & Attendee Monitoring:** Detailed view of all marked campsites and attendee locations.
 - **Flagged Users:** List of flagged users with options to view details and navigate.
 - **Internal Messaging:** Access to messaging or announcements for staff communication.
 - (Optional) Additional links such as Reports or Analytics.

C. Main Content Area (Right Panel) ↗

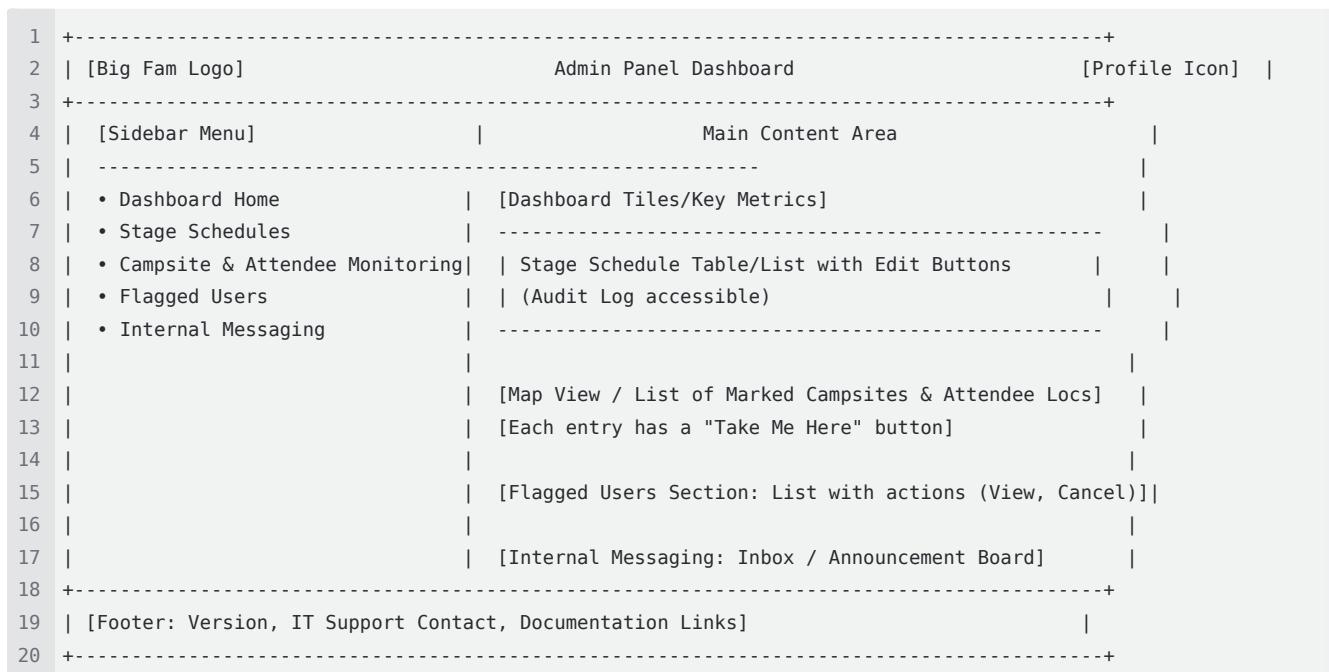
- **Dashboard Home View:**
 - High-level metrics (e.g., number of active users, flagged incidents, upcoming schedule changes).
 - Quick access tiles for key actions (e.g., “Modify Stage Schedule”, “View Campsites”, “Flag User”).
- **Stage Schedule Modification Section:**
 - **Editable Table/List:** Display current schedules for each stage with an “Edit” button beside each event.
 - **Audit Log Button:** A link or icon to view the log of all modifications.
- **Campsite & Attendee Monitoring Section:**

- **Map View or Table:** Shows a real-time map or list of all marked campsites and attendees' last reported locations.
- **Navigation Buttons:** "Take Me Here" buttons next to each entry for immediate navigation.
- **Flagged User Management Section:**
 - **Flagged List:** Displays users flagged for monitoring along with details (reason, timestamp).
 - **Action Buttons:** Options to review flag details, cancel flag, or navigate to the flagged user's location.
- **Internal Messaging Section:**
 - A simple inbox or announcement board for internal communication.

D. Footer (Optional) ↗

• Footer Elements:

- Information such as version number, contact information for IT support, or links to documentation.



4. Interaction & User Flow ↗

1. Dashboard Home:

- Upon logging in, the admin sees the dashboard with key metrics (active users, alerts, upcoming schedule changes) and quick action tiles.
- Clicking on a tile or button navigates to the corresponding section (e.g., Stage Schedules, Campsite Monitoring).

2. Modifying Stage Schedules:

- Admin selects "Stage Schedules" from the sidebar.
- A table or list of current schedules appears.
- The admin clicks an "Edit" button next to an event, updates details (time, order, etc.), and saves changes.
- Changes are logged in the audit log; a confirmation message appears.

3. Campsite & Attendee Monitoring:

- Admin selects "Campsite & Attendee Monitoring."
- A real-time map or list displays all marked campsites and attendee locations.
- Each entry includes a "Take Me Here" button to initiate navigation.
- Additional filters may allow the admin to search by location, time, or user.

4. Flagged User Management:

- Admin selects "Flagged Users."
- A list of flagged users is displayed with details (flag reason, timestamp).
- Action buttons allow the admin to review details, cancel a flag, or navigate to the user's current location.

5. Internal Messaging:

- Admin selects "Internal Messaging" to view an inbox of messages or announcements.
- The interface allows for quick reading and replying to messages from staff or other admins.

6. Responsive and Accessible Flow:

- The interface is designed to be responsive for use on tablets and desktops.
 - Accessible navigation, clear labels, and feedback messages are provided for all actions.
-

5. Additional Considerations ☰

- **Security & RBAC:**

Ensure that all admin functionalities are secured and only accessible to users with the admin role.

- **Performance:**

Real-time data (e.g., map view and attendee locations) should refresh at appropriate intervals without degrading performance.

- **User Feedback:**

Confirmation messages, loading indicators, and error notifications should be clear and informative.

- **Customization:**

The sidebar may be collapsible to maximize screen space for the main content on smaller displays.

6. Conclusion ☰

This Admin Panel UI/UX wireframe provides a comprehensive view of all the essential functionalities that admins require to manage festival operations effectively. With a clear sidebar for navigation, a detailed main content area for various operational tasks, and responsive design considerations, this panel ensures that administrators can quickly access and act on critical information.

ADMIN: Artist Manager

Last Updated: Feb 11, 2025

1. Overview ☰

Purpose:

The Artist Manager is designed to manage artist profiles for the festival. In addition to editing an artist's personal details (such as profile picture, bio, and social media links), the panel now supports the management of multiple performance events for each artist. This enables artists to be associated with more than one event—critical for cases where an artist is scheduled to perform multiple times. Admins can add, edit, or delete performance events, ensuring that all performance details are up to date.

Access:

- Accessible only to admin users via the unified dynamic panel.
- The Artist Manager is reached via a dynamic sidebar link (labeled "Artist Manager") when the logged-in user is an admin.
- The updated panel includes a "Performance Schedule" section integrated with the existing Artist Profile section.

2. Key UI Components ☰

A. Header & Global Navigation ☰

- **Header Elements:**

- **Left:** Big Fam Festival logo.
- **Center:** Page title, which dynamically displays "Admin Panel - Artist Manager."
- **Right:** Profile icon (showing the admin's profile) with a dropdown for account actions (e.g., Logout, Settings).

B. Sidebar Navigation (Within Admin Panel) ☰

- **Common Sidebar Items:**

- Home, Profile, Notifications, Stage Schedules, Shift Manager, Vendor Manager, etc.

- **Dynamic Role-Specific Link:**

- A link labeled "Artist Manager" is rendered for admin users.

C. Main Content Area – Artist Manager Dashboard ☰

1. Artist Profile Overview ☰

- **Editable Fields:**

- **Artist Name:** A text input field.
- **Bio:** A multi-line text area.
- **Profile Image:** With an option to update (upload new image).
- **Social Media Links:** Input fields or icon links (e.g., Instagram, Twitter, Facebook).

- **Action Button:**

- "Save Changes" for the artist's profile details.

2. Performance Schedule Section (New) ↗

- **Purpose:**

Display and manage all performance events associated with the artist.

- **Layout Elements:**

- **Performance Event List/Table:**

- Each performance event is shown as a card or row with the following details:

- **Stage:** The stage where the artist will perform.

- **Date:** The date of the performance.

- **Start Time:** The scheduled start time.

- **Duration:** How long the performance is scheduled to last.

- (Optional) **Additional Details:** E.g., a brief description or specific notes.

- Each event card includes action buttons:

- **Edit:** To modify the performance details.

- **Delete:** To remove the event.

- **Add Performance Event Button:**

- A clearly labeled button (“Add Performance Event”) that opens a modal for adding a new performance event.

3. Add/Edit Performance Event Modal ↗

- **Modal Header:** “Add New Performance” or “Edit Performance”

- **Form Fields:**

- **Stage (Required):** Text input or dropdown for stage name.

- **Date (Required):** Date picker for the performance date.

- **Start Time (Required):** Time picker for the start time.

- **Duration (Required):** Input for performance duration (or an end time picker).

- **Additional Details (Optional):** Multi-line text area for any extra notes.

- **Action Buttons:**

- **Save/Update:** Submits the performance event.

- **Cancel:** Closes the modal without saving.

D. Footer ↗

- **Footer Elements:**

- Displays version information, IT support contacts, and links to documentation/help resources.

3. Wireframe Diagram (ASCII Representation) ↗

A. Artist Manager – Main Dashboard (Multi-Event Support) ↗



```

12 |   • Artist Manager      | <- Selected
13 |   • More
14 |
15 |
16 | Main Content Area:
17 | +-----+
18 | | Artist Profile Overview
19 | |   - Artist Name, Bio, Profile Image, Social Media Links
20 | |   [Save Changes]
21 | +-----+
22 | +-----+
23 | | Performance Schedule
24 | |
25 | | | Event Card:
26 | | |   - Stage: Main Stage
27 | | |   - Date: June 20
28 | | |   - Start Time: 7:30 PM
29 | | |   - Duration: 90 min
30 | | |   [Edit]  [Delete]
31 | | |
32 | | [Add Performance Event] (Floating Button)
33 | +-----+
34 +-----+
35 | [Footer: Version, IT Support, Documentation Links]
36 +-----+
37

```

B. Add/Edit Performance Event Modal ☰

```

1 +-----+
2 |           {Add / Edit} Performance          | <- Modal Header
3 +-----+
4 | Stage:      [_____]                      | <- Required
5 | Date:       [Date Picker]                 | <- Required
6 | Start Time: [Time Picker]                | <- Required
7 | Duration:   [_____] (minutes or End Time) | <- Required
8 | Additional Info: [_____]                  | <- Optional
9 +-----+
10 | [Save Performance]  [Cancel]            | <- Action Buttons
11 +-----+
12

```

4. Interaction & User Flow ☰

1. Accessing the Artist Manager:

- An admin logs in and selects “Artist Manager” from the Admin Panel sidebar.
- The Artist Manager dashboard loads, showing the Artist Profile Overview at the top and the Performance Schedule section below.

2. Viewing and Managing Performance Events:

- The Performance Schedule section displays a list of events associated with the artist.
- Each event card shows details such as stage, date, start time, and duration.
- Action buttons (“Edit” and “Delete”) allow the admin to modify or remove events.
- Tapping “Edit” opens the Add/Edit Performance Event modal pre-populated with the event’s details.

3. Adding a New Performance Event:

- The admin clicks the “Add Performance Event” floating button.

- The modal appears prompting for required inputs: Stage, Date, Start Time, Duration, and an optional Additional Info field.
- After entering the information, tapping “Save Performance” adds the new event to the Performance Schedule list.
- A confirmation message is displayed, and the updated schedule is immediately visible.

4. Real-Time Updates & Data Synchronization:

- All changes (additions, edits, deletions) are synchronized with the backend.
 - Confirmation messages and error handling ensure a smooth user experience.
-

5. Additional Considerations

• Responsive Design:

The layout and modals should adjust gracefully on mobile, tablet, and desktop devices.

• Accessibility:

Ensure that all input fields, buttons, and modals have proper labels, keyboard accessibility, and sufficient color contrast.

• User Feedback:

Provide immediate visual feedback (toast notifications, confirmation dialogs) after any update.

• Security:

Validate all input fields before submission and enforce proper access controls on the backend.

• Offline Support:

Cache performance data locally when possible, ensuring updates are queued and synchronized when connectivity is restored.

6. Conclusion

This updated Artist Manager wireframe now supports multiple performance events per artist, allowing for comprehensive management of an artist’s festival schedule. With a dedicated Performance Schedule section and a user-friendly modal for adding and editing events, admins can efficiently manage artists who are performing more than once. This design provides a consistent and dynamic interface that meets the needs of the festival’s diverse operational requirements.

ADMIN: Shift Manager (Admin / Director)

1. Overview ☰

Purpose:

This interface is designed for directors (a specialized staff role) to manage shift assignments during the festival. Directors can add new shifts, assign staff members, edit or delete existing shifts, and transfer shifts to other staff. When creating or editing a shift, the form requires:

- **Who:** (Optional) The assigned staff member (by email or phone).
- **What:** (Required) Description/Type of the shift.
- **Where:** (Required) Location for the shift.
- **Why:** (Required) Purpose of the shift.
- **When:** (Required) Start time and duration.

Directors can also filter shifts based on:

- Shift Type (What)
- Shift Location (Where)
- Shift Start Time
- Shift Duration (When)

2. Key UI Components ☰

A. Header & Global Navigation ☰

• Header Elements:

- **Left:** Big Fam Festival logo.
- **Center:** Page Title (“Director Shift Management”).
- **Right:** Profile icon (with dropdown for account actions).

B. Sidebar Navigation (Dynamic) ☰

• Common Items:

- Home, Profile, Notifications

• Dynamic Role-Specific Link:

- “Staff Panel” is displayed for directors (along with the additional director shift management functions).

C. Main Content Area – Shift Management ☰

1. Filters Bar (at the top of the content area) ☰

• Filter Fields/Controls:

- **Shift Type (What):** Dropdown or input to filter by shift description/type.
- **Shift Location (Where):** Dropdown or input for location.
- **Shift Start Time:** Date/time picker or range slider.
- **Shift Duration (When):** Input or dropdown to filter by duration.

2. Shift List / Cards ☰

• Each Shift Card Displays:

- **Shift Details:**

- What: Shift description/type
- Where: Location
- Why: Brief purpose
- When: Start time and duration
- Who: Assigned staff member (if provided, showing their email/phone)
- **Action Buttons:**
 - **Edit:** Opens the edit shift modal.
 - **Delete:** Prompts for confirmation and deletes the shift.
 - **Transfer:** Opens a modal to transfer the shift to another staff member.

3. Add New Shift Button

- **Floating Action Button (FAB):**
 - A prominent button labeled “Add New Shift” located in a fixed position (e.g., bottom-right).
 - Tapping it opens the “Add Shift” modal.

D. Add/Edit Shift Modal

- **Modal Header:** “Add New Shift” or “Edit Shift”
- **Form Fields:**
 - **Who (Optional):** Input field for staff email/phone (if assigning immediately).
 - **What (Required):** Input field for shift description/type.
 - **Where (Required):** Input field for shift location.
 - **Why (Required):** Text area for the shift’s purpose.
 - **When (Required):**
 - **Start Time:** Time/date picker.
 - **Duration:** Input field (or end time picker).
- **Action Buttons:**
 - **Save/Update:** To submit the shift details.
 - **Cancel:** To close the modal without saving.

E. Transfer Shift Modal

- **Modal Header:** “Transfer Shift”
- **Content:**
 - **Shift Summary:** Display current shift details.
 - **New Staff Assignment:** Input field for new staff email/phone.
- **Action Buttons:**
 - **Transfer Shift:** To confirm the transfer.
 - **Cancel:** To close without action.

F. Footer (Optional)

- **Footer Elements:**
 - Version information, IT support contacts, help/documentation links.

3. Wireframe Diagrams (ASCII Representation)

A. Director Shift Management – Main Panel

```

1 +-----+
2 | [Big Fam Logo]          Director Shift Management           [Profile Icon ▾] |
3 +-----+
4 | Sidebar:                | Main Content Area:                         |
5 | +-----+ | +-----+ | +-----+ | +-----+ |
6 | | • Home                 | | | [Filters:                                |
7 | | • Profile               | | | - Shift Type: [All ▾] - Location: [All ▾] | |
8 | | • Notifications        | | | - Start Time: [_____] - Duration: [_____] | |
9 | | • Staff Panel          | | +-----+ | +-----+ |
10| | (Director Options)    | | | | | | | | |
11| | • More                  | | | [Shift Card] | | |
12| +-----+ | +-----+ | +-----+ | +-----+ |
13| | | | What: Setup Crew | | |
14| | | | Where: Main Tent | | |
15| | | | Why: Stage Setup | | |
16| | | | When: June 20, 9:00 AM - 1:00 PM | | |
17| | | | Who: John Doe (jdoe@example.com) | | |
18| | | | [Edit] [Delete] [Transfer] | | |
19| | | +-----+ | +-----+ |
20| | | [Shift Card] (Additional shift cards...) | | |
21| | | | | | | | |
22| | | [Add New Shift] (Floating Action Button) | | |
23+-----+
24| [Footer: Version, IT Support, Documentation] |
25+-----+
26

```

B. Add New Shift

```

1 +-----+
2 |          Add New Shift          | <-- Modal Header
3 +-----+
4 | Who (Optional): [Enter Email/Phone] (Leave blank if none) |
5 | What (Required): [_____] | <-- Dropdown of depts i.e stagehand,
  ticketing etc.
6 | Where (Required): [_____] | <-- Dropdown of locations possible
7 | Why (Required): [_____] | <-- Quick description of the job details
8 | When:
9 |   - Start Time: [_____] - Duration: [_____] |
10+-----+
11| [Save Shift]      [Cancel] |
12+-----+
13

```

C. Transfer Shift

```

1 +-----+
2 |          Transfer Shift          | <-- Modal Header
3 +-----+
4 | Shift: Setup Crew | |
5 | Current Assigned: John Doe (jdoe@example.com) |
6 | Enter New Staff Email/Phone: [_____] |
7 +-----+
8 | [Transfer Shift]  [Cancel] |
9 +-----+
10

```

4. Interaction & User Flow ↴

1. Accessing the Director Panel:

- A director logs in and the unified dynamic panel loads the Director Shift Management view.
- The sidebar displays director-specific navigation, including a link to the Staff Panel with director options.

2. Filtering Shifts:

- The director uses the filters at the top of the main content area to narrow down shifts by type (What), location (Where), start time, or duration.
- The shift list updates dynamically based on selected filters.

3. Adding a New Shift:

- The director clicks the “Add New Shift” FAB.
- The Add New Shift modal appears, requiring input for What, Where, Why, and When, with an optional Who field.
- The director can input a staff member’s email/phone to assign the shift immediately.
- After filling the form, clicking “Save Shift” adds the new shift to the shift list.

4. Editing or Deleting a Shift:

- Each shift card has Edit and Delete buttons.
- Tapping “Edit” opens the modal pre-populated with the current shift details for modification.
- Tapping “Delete” prompts a confirmation before removing the shift.

5. Transferring a Shift:

- Tapping “Transfer” on a shift card opens the Transfer Shift modal.
- The director inputs the new staff member’s email/phone.
- After confirming, the shift is reassigned, and the change reflects in the respective staff member’s shift list.

6. Feedback:

- Confirmation messages and visual cues (such as success notifications or error prompts) appear for each action (add, edit, delete, transfer).

5. Additional Considerations ↴

• Responsive Design:

The layout should adapt to various screen sizes. On smaller screens, the sidebar might collapse into a hamburger menu.

• Accessibility:

Ensure that all buttons, input fields, and modals have accessible labels and support keyboard navigation.

• Real-Time Updates:

Use real-time data synchronization (via websockets or similar technology) to update shift assignments across the system immediately.

• Security:

Validate all inputs and enforce backend RBAC so that only directors can perform these shift management actions.

6. Conclusion ↴

The Director Shift Management interface within the unified dynamic panel is designed to provide directors with robust tools to manage shift assignments. With capabilities to add, edit, delete, and transfer shifts, along with comprehensive filtering options, this interface streamlines operational management during the festival. The modular design and responsive layout ensure a consistent, efficient, and user-friendly experience across devices.

ADMIN: Vendor Manager

Last Updated: Feb 11, 2025

1. Overview ☰

Purpose:

The Vendor Manager is an administrative tool integrated into the Admin Panel that allows festival administrators to manage vendor information. Administrators can:

- **Add Vendors:** Create new vendor entries with details such as business name, description, contact info, and location.
- **Edit Vendors:** Update existing vendor details.
- **Delete Vendors:** Remove vendor entries as needed.

This tool ensures that vendor data remains accurate and up to date throughout the festival.

Access:

- The Vendor Manager is accessible only to users with the admin role.
- It is integrated into the Admin Panel sidebar as a dedicated section.

2. Key UI Components ☰

A. Header & Global Navigation ☰

- **Header:**
 - **Left:** Big Fam Festival logo.
 - **Center:** Page Title (e.g., "Admin Panel – Vendor Manager").
 - **Right:** Profile icon (displaying admin's profile) with account actions (e.g., Logout, Settings).

B. Sidebar Navigation (Within Admin Panel) ☰

- **Common Sidebar Items:**
 - Home, Profile, Notifications, Stage Schedules, Shift Manager, Campsite & Attendee Monitoring, Flagged Users, Internal Messaging.
- **New Item - Vendor Manager:**
 - A dedicated link labeled "**Vendor Manager**" that, when clicked, loads the Vendor Manager dashboard in the main content area.

C. Main Content Area – Vendor Manager Dashboard ☰

1. Vendor List/Table ☰

- **Display:**
 - A scrollable table or list showing all vendors.
- **Each Vendor Entry Displays:**
 - **Vendor ID/Logo:** A small image or placeholder.
 - **Business Name:** The vendor's name.
 - **Description:** A brief summary of the vendor's offerings.
 - **Contact Info:** (Optional) Email and phone.
 - **Location:** (Optional) Address or map pin.

- **Action Buttons for Each Entry:**

- **Edit:** Opens a modal or inline form to update vendor details.
- **Delete:** Triggers a confirmation prompt before deletion.

2. Add New Vendor Section ☰

- **Add Vendor Button:**

- A prominent “Add Vendor” button (could be a Floating Action Button or positioned at the top of the Vendor Manager dashboard).

- **Add Vendor Modal/Form:**

- **Form Fields:**

- **Business Name:** (Required)
- **Description:** (Required)
- **Contact Information:** (Email, Phone)
- **Location:** (Optional; text input or interactive map picker)
- **Vendor Logo/Image Upload:** (Optional)

- **Action Buttons:**

- **Save/Submit:** To add the new vendor.
- **Cancel:** To close the modal without saving.

3. Wireframe Diagram (ASCII Representation) ☰

A. Admin Panel – Vendor Manager Main Dashboard ☰

```

1 +-----+
2 | [Big Fam Logo]          Admin Panel – Vendor Manager           [Profile Icon ▾] |
3 +-----+
4 | Sidebar (Admin Navigation):
5 | +-----+
6 | | • Home
7 | | • Profile
8 | | • Notifications
9 | | • Stage Schedules
10 | | • Shift Manager
11 | | • Campsite & Attendee
12 | | Monitoring
13 | | • Flagged Users
14 | | • Internal Messaging
15 | | • Vendor Manager      | <-- Selected
16 | +-----+
17 |
18 | Main Content Area:
19 | +-----+
20 | | [Add Vendor] (Button)
21 | +-----+
22 | | Vendor List / Table:
23 | | -----
24 | | | Vendor Logo | Business Name | Description | Actions | |
25 | | | ----- | ----- | ----- | ----- | |
26 | | | [Image]     | Vendor A     | Food & Drinks | [Edit] | |
27 | | |             |             |             | [Delete] | |
28 | | | [Image]     | Vendor B     | Merchandise | [Edit] | |
29 | | |             |             |             | [Delete] | |

```

```

30 |   |   | ...
31 |   | -----
32 |   +-----+
33 +-----+
34 | [Footer: Version, IT Support, Documentation Links]
35 +-----+
36

```

B. Add/Edit Vendor Modal

```

1 +-----+
2 |           Add New Vendor           | <-- Modal Header
3 +-----+
4 | Business Name: [_____]
5 | Description: [_____]
6 | Contact Email: [_____]
7 | Contact Phone: [_____]
8 | Location: [_____]
9 | Director: [_____]
10 | Upload Logo: [Choose File]
11 +-----+
12 | [Save Vendor] [Cancel]
13 +-----+

```

4. Interaction & User Flow

1. Accessing Vendor Manager:

- An admin logs in and selects "Vendor Manager" from the Admin Panel sidebar.
- The Vendor Manager dashboard loads, showing the list of vendors and an "Add Vendor" button.

2. Viewing the Vendor List:

- The admin sees a table or list of all vendors with basic details (logo, business name, description) and action buttons (Edit, Delete).
- The list is scrollable with possible pagination or infinite scroll if needed.

3. Adding a New Vendor:

- The admin taps the "Add Vendor" button.
- The Add New Vendor modal appears with fields for business name, description, contact information, location, and an option to upload a logo.
- The admin fills in the details and clicks "Save Vendor."
- The new vendor appears in the vendor list, and a confirmation message is displayed.

4. Editing an Existing Vendor:

- The admin clicks the "Edit" button on a vendor entry.
- A similar modal appears, pre-populated with the vendor's current information.
- The admin makes the necessary changes and saves them.
- The vendor list updates immediately with the revised information.

5. Deleting a Vendor:

- The admin taps the "Delete" button on a vendor entry.
- A confirmation prompt appears to ensure the admin wants to remove the vendor.
- Upon confirmation, the vendor is removed from the list and a success message is displayed.

6. Real-Time Feedback & Synchronization:

- All actions provide immediate visual feedback (toast notifications, modals, etc.).

- Changes are synchronized with the backend to ensure that updated vendor information is available throughout the app.
-

5. Additional Considerations

- **Responsive Design:**

The layout should adjust smoothly for various devices. On smaller screens, the sidebar may collapse into a hamburger menu.

- **Accessibility:**

All interactive elements (buttons, input fields, file upload controls) must include proper labels and keyboard navigation support.

- **Data Validation:**

Ensure that all vendor details are validated (e.g., correct email format, phone number validation) before submission.

- **Security:**

Sensitive data (e.g., contact information and file uploads) should be securely transmitted and stored.

- **User Feedback:**

Use clear confirmation messages for add, edit, and delete actions.

6. Conclusion

The Vendor Manager within the Admin Panel is now fully integrated into the unified dynamic panel. This section enables administrators to add, edit, and delete vendor information seamlessly. With a clear, modular design and responsive layout, the Vendor Manager provides an efficient tool for keeping vendor data current and accurate throughout the festival.

ADMIN: Ticket Manager

Last Updated: Feb 11, 2025

1. Overview

Purpose:

The Ticket Manager enables designated users (Staff, Directors, and Admins) to use their mobile devices to scan attendee tickets (via QR codes), validate them, and process the results with the backend. For admin users, an additional option to manually add tickets is provided, allowing for situations where a ticket needs to be created or adjusted manually.

Access:

- **Available To:** Staff, Directors, and Admins.
 - **Admin-Only Function:** The "Add Ticket" functionality is exclusively visible to admin users.
-

2. Key UI Components

A. Header & Global Navigation

• Header Elements:

- **Left:** Big Fam Festival Logo.
- **Center:** Page Title ("Ticket Manager").
- **Right:** Profile icon (with a dropdown for account actions like Logout).

B. Main Content Area

1. Ticket Scanning & Validation

• QR Code Scanner Interface:

- A full-screen camera view with an overlay guiding the user to align the ticket's QR code within a scanning frame.
- A button labeled "Scan Ticket" (or an auto-scan that triggers when a QR code is detected).

• Ticket Details Preview:

- After scanning, display a summary of the ticket details retrieved from the backend.
- Information may include Ticket ID, Ticket Type (e.g., GA, VIP, RV), and Validation Status (e.g., Valid, Redeemed, Pending).

• Action Buttons:

- **Validate/Process Ticket:** A button to confirm validation and process the ticket (which may trigger updating its status).
- **Cancel/Rescan:** An option to cancel and return to the scanning interface.

2. Admin-Only: Manual Ticket Addition

• "Add Ticket" Option (Admin Only):

- A clearly labeled button ("Add Ticket") visible only to admin users.
- When tapped, it opens a modal or form allowing admins to manually input ticket details.

3. Add Ticket Modal / Form (Admin Only)

• Form Fields:

- **Ticket ID:** A unique identifier (auto-generated or manually entered).

- **Ticket Type:** Dropdown (e.g., GA, VIP, RV, need-ticket).
 - **Attendee Information:** Optional fields such as attendee name, email, or phone.
 - **Validation Status:** Initially set to “Pending” or “Valid” as required.
- **Action Buttons:**
- **Save Ticket:** Submits the manual ticket entry to the backend.
 - **Cancel:** Closes the modal without saving.

C. Footer ↗

- **Footer Elements:**
- Version number, support/IT contact details, and links to documentation.

3. Wireframe Diagram (ASCII Representation) ↗

A. Ticket Manager Main View ↗

```

1 +-----+
2 | [Big Fam Logo]           Ticket Manager           [Profile Icon ▾] |
3 +-----+
4 |
5 |   [QR Code Scanner Area]
6 |   -----
7 |   |
8 |   |   [Scanning Frame Overlay]
9 |   |   (Align ticket QR code here)
10|  |
11|  -----
12|
13|   [Scan Ticket]  [Cancel/Rescan]
14|
15|   --- Ticket Details Preview (after scan) ---
16|   Ticket ID: QR123ABC      Ticket Type: VIP      Status: Valid
17|   [Validate/Process Ticket]
18|
19|   (For Admins Only: Manual Ticket Addition)
20|   [Add Ticket]
21+-----+
22|   [Footer: Version, IT Support, Documentation Links]
23+-----+
24

```

B. Add Ticket Modal (Admin Only) ↗

```

1 +-----+
2 |           Add Ticket           | <- Modal Header
3 +-----+
4 | Ticket ID:      [_____]       | <- Input Field
5 | Ticket Type:    [Dropdown: GA, VIP, RV, need-ticket] | <- Required
6 | Attendee Name:  [_____]       | <- Optional
7 | Attendee Email: [_____]       | <- Optional
8 | Attendee Phone: [_____]       | <- Optional
9 | Validation Status: [Pending / Valid] (auto-set)     | <- Read-only/auto
10+-----+
11|   [Save Ticket]      [Cancel]    | <- Action Buttons

```

4. Interaction & User Flow ↗

1. Ticket Scanning & Validation (For Staff, Directors, Admins):

- The user navigates to the Ticket Manager.
- The QR Code Scanner interface loads and activates the device camera.
- The user aligns the attendee ticket's QR code within the scanning frame.
- The system auto-detects the QR code and displays ticket details (Ticket ID, Type, Validation Status).
- The user reviews the details and taps "Validate/Process Ticket" to finalize the validation.
- A confirmation message appears (e.g., "Ticket validated successfully") and the backend updates the ticket status.
- Alternatively, if the scan is incorrect, the user can tap "Cancel/Rescan" to try again.

2. Manual Ticket Addition (For Admins Only):

- An admin sees the "Add Ticket" button on the Ticket Manager screen.
- Tapping "Add Ticket" opens the Add Ticket modal.
- The admin enters the required information (Ticket ID, Ticket Type, optional attendee details) and submits the form.
- The new ticket is added to the system, and a confirmation message is displayed.
- The manually added ticket appears in the Ticket Manager list.

3. Feedback & Error Handling:

- Visual feedback (e.g., toast notifications) confirms successful validation, addition, or errors.
- The system handles errors (e.g., invalid QR code, missing required fields) and prompts the user accordingly.

5. Additional Considerations ↗

• Responsive Design:

Ensure the QR scanning interface, ticket details preview, and modals adapt well to mobile devices.

• Accessibility:

Include accessible labels for all buttons, input fields, and camera overlays. Provide sufficient contrast and keyboard navigability.

• Security:

Validate all scanned and manually entered ticket data on the backend. Enforce role-based access control for manual ticket addition.

• Offline Support:

Cache essential ticket data locally to allow staff to operate in low-connectivity environments, synchronizing when online.

6. Conclusion ↗

The Ticket Manager provides an essential tool for managing attendee ticket validations via QR code scanning, while also enabling admin users to manually add tickets when necessary. This unified interface is designed to be user-friendly, secure, and responsive, ensuring that all designated staff members can efficiently process and validate tickets during the festival.

PAGE: Staff Panel

Last Updated Feb 11, 2025

1. Overview ↗

Purpose:

The Staff Panel is designed to provide non-admin staff members (e.g., volunteer coordinators, artist relations) with a dedicated dashboard to manage their daily tasks during the festival. In this panel, staff can:

- **View Shift Assignments:**

See a list or calendar view of their scheduled shifts and tasks.

- **Access Internal Messaging:**

Read announcements and messages from festival organizers or coordinate with other staff members.

- **Manage Personal Profile:**

Update their contact details, availability, and other staff-specific information.

- **Quick Actions/Task List:**

Access a to-do list or quick links for common staff functions.

Access:

- This panel is accessible only to users with the staff role.
- It is reached via dynamic role-based navigation (from the "More" menu or Profile Editor).

2. Key UI Components ↗

A. Header & Global Navigation ↗

- **Header Elements:**

- **Left:** Big Fam Festival Logo.
- **Center:** Page Title ("Staff Panel").
- **Right:** Profile icon (shows staff member's profile) with a dropdown for account actions (Logout, Settings).

B. Navigation Sidebar (or Top Tab Navigation) ↗

- **Sidebar Menu Options:**

- **My Shifts:**

View and manage scheduled shifts and tasks.

- **Internal Messaging:**

Access an inbox or announcements board for communications from festival organizers.

- **Profile:**

Edit personal staff details and availability.

- **(Optional) Task List/Quick Actions:**

A section for to-do items or quick links to common staff tasks.

C. Main Content Area ↗

1. My Shifts Section ↗

- **Shift List or Calendar View:**

- Displays assigned shifts with key details:

- **Shift Title/Role:** (e.g., "Volunteer Coordinator" or "Artist Relations Support")
- **Date & Time:** e.g., "June 20, 10:00 AM - 4:00 PM."
- **Location:** e.g., "Main Tent" or "Backstage."
- **Status:** Indicators such as "Upcoming," "In Progress," or "Completed."
- **Action Buttons:**
 - "View Details" to open a detailed view for each shift.
 - (Optional) "Mark as Completed" if staff need to self-update their shift status.

2. Internal Messaging/Announcements Section ☀

- **Inbox or Announcement Board:**
 - A scrollable list showing recent messages/announcements.
 - Each message displays:
 - Sender name and/or profile picture.
 - A brief preview of the message.
 - Timestamp.
 - Tapping a message opens the full content with reply options.

3. Profile Management Section ☀

- **Editable Staff Profile:**
 - **Profile Picture:** With an option to update.
 - **Contact Details:** Name, email, phone, and (if applicable) emergency contact.
 - **Availability/Shift Preferences:** Optional fields to update availability.
 - A "Save Changes" button to commit updates.

4. Task List / Quick Actions (Optional) ☀

- **Purpose:**
Provide a quick overview of pending tasks or actions (e.g., "Review Shift Changes," "Update Availability").
- **Layout:**
 - A list or grid of task cards with short titles and status indicators.
 - Each card links to the detailed task or function.

D. Footer (Optional) ☀

- **Footer Elements:**
 - Version number, IT support contact information, and links to help/documentation.

3. Wireframe Diagram (ASCII Representation) ☀

```

1 +-----+
2 | [Big Fam Logo]           Staff Panel Dashboard           [Profile Icon] |
3 +-----+
4 | Sidebar (Navigation)      Main Content Area           |
5 | -----|-----|-----|
6 | • My Shifts              | [My Shifts Section]          |
7 | • Internal Messaging     | +-----+-----+-----+
8 | • Profile                 | | Shift Card:             |
9 | • (Optional) Task List    | | | - Role: Volunteer Coordinator |
10 |                         | | | - Date/Time: June 20, 10:00 AM - 4:00 PM |
11 |                         | | | - Location: Main Tent   |
12 |                         | | | - Status: Upcoming      |

```

```

13 |           | [View Details] [Mark as Completed] | |
14 |           | +-----+ | |
15 |           | +-----+ | |
16 |           | | ... (Additional Shift Cards) | |
17 |           | +-----+ | |
18 |           | | | |
19 |           | | [Internal Messaging Section] | |
20 |           | | +-----+ | |
21 |           | | | Message: "Staff meeting at 3:00 PM" | |
22 |           | | | Sender: Festival Organizer | |
23 |           | | | Timestamp: 9:30 AM | |
24 |           | | +-----+ | |
25 |           | | | |
26 |           | | [Profile Management Section] | |
27 |           | | +-----+ | |
28 |           | | | Profile Picture: [Editable] | |
29 |           | | | Name: [_____] | |
30 |           | | | Email: [_____] | |
31 |           | | | Phone: [_____] | |
32 |           | | | Availability: [_____] | |
33 |           | | | [Save Changes] | |
34 |           | | +-----+ | |
35 |           | | | |
36 |           | | [Optional: Task List / Quick Actions Section] | |
37 |           | | +-----+ | |
38 |           | | | Task: Update Shift Preferences | |
39 |           | | | Status: Pending | |
40 |           | | +-----+ | |
41 | +-----+ | |
42 | | [Footer: Version, IT Support, Documentation Links] | |
43 | +-----+ | |

```

4. Interaction & User Flow

1. Accessing the Staff Panel:

- A staff member logs in and selects the “Staff Panel” link from the dynamic navigation.
- The Staff Panel loads with the header displaying “Staff Panel Dashboard” and a sidebar with navigation options.

2. Managing Shifts:

- The default view in the Main Content Area is the My Shifts section.
- Staff view their assigned shifts as cards or list items.
- Tapping “View Details” opens additional shift information; if available, they can mark a shift as completed.

3. Internal Messaging:

- The staff member selects the “Internal Messaging” option from the sidebar.
- The messaging section displays recent announcements and messages from festival organizers.
- Tapping a message opens the full content for review and response.

4. Updating Personal Profile:

- The staff member navigates to the “Profile” section.
- They update their contact details, availability, and other relevant information.
- Upon clicking “Save Changes,” a confirmation is displayed.

5. Optional Task List / Quick Actions:

- The optional Task List provides quick links to pending tasks (e.g., updating shift preferences or reviewing new announcements).

6. Footer & Global Navigation:

- The footer remains visible, providing version info, IT support contacts, and help/documentation links.
 - Global navigation elements (top nav) remain accessible for quick access to other parts of the app.
-

5. Additional Considerations ☰

- **Responsive Design:**

Ensure the layout adjusts smoothly on mobile, tablet, and desktop devices.

- **Accessibility:**

Provide accessible labels for all buttons, text fields, and interactive elements; ensure sufficient contrast.

- **User Feedback:**

Visual confirmation messages (e.g., "Profile Updated Successfully," "Shift Marked as Completed") should be shown after actions.

- **Offline Functionality:**

Cache essential data (shift schedules, messages, profile details) locally to maintain usability in low-connectivity environments.

- **Security:**

Ensure that sensitive staff data and messaging are protected with appropriate access controls.

6. Conclusion ☰

The Staff Panel UI/UX wireframe offers a comprehensive and user-friendly interface for festival staff to manage their shifts, internal communications, and personal profile details. With clearly defined sections for shifts, messaging, profile management, and optional task management, the panel supports efficient and responsive staff operations during the festival.

PAGE: Volunteer Panel

Last Updated: Feb 11, 2025

1. Overview

Purpose:

The Volunteer Panel is designed specifically for festival volunteers to manage their commitments during the event. In this panel, volunteers can:

- View their assigned shifts in a clear list or calendar format.
- Update their availability (if applicable) or mark shifts as completed.
- Access messages and announcements from volunteer coordinators.
- Update and review their personal profile details relevant to volunteering.

Access:

- This panel is accessible only to users with the volunteer role.
 - It is reached via dynamic role-based navigation (e.g., through the “More” menu or Profile Editor).
-

2. Key UI Components

A. Header & Global Navigation

• Header Elements:

- **Left:** Big Fam Festival Logo.
- **Center:** Title “Volunteer Panel.”
- **Right:** Profile icon (displays volunteer’s picture) with a dropdown for account actions (such as Logout).

B. Navigation Sidebar (or Tabbed Navigation)

• Sidebar Menu Options:

- **My Shifts:** View and manage upcoming and past volunteer shifts.
- **Profile:** View and update personal volunteer details (e.g., contact info, emergency contacts).
- **Messages:** Access internal messaging or announcements from volunteer coordinators.
- **(Optional) Availability:** If volunteers need to update their availability, this can be a separate option or included in the Profile section.

C. Main Content Area

1. My Shifts Section

• Shift Cards/List:

- Each card or list item represents an assigned volunteer shift.
- **Details Include:**
 - **Shift Title/Role:** For example, “Food Stall Greeter” or “Setup Crew.”
 - **Date & Time:** Clearly display the scheduled date and time (e.g., “June 20, 10:00 AM – 4:00 PM”).
 - **Location:** The specific area or tent (e.g., “Main Tent”).
 - **Status:** An indicator (e.g., “Upcoming,” “In Progress,” “Completed”).
- **Action Buttons:**

- A “View Details” button to see more shift information.
- (Optional) A “Mark as Completed” button if volunteers can self-update shift status.

2. Profile Section ☰

• Volunteer Profile Information:

- Profile picture, name, and contact information.
- Emergency contact or additional volunteer-specific details.
- Option to update availability (if applicable).

3. Messages/Announcements Section ☰

• Inbox or Announcement Board:

- A list view showing recent messages or announcements from volunteer coordinators.
- Each message includes the sender, a short preview, and a timestamp.
- Option to tap and open a detailed view or reply.

D. Footer (Optional) ☰

• Footer Elements:

- Basic festival info, version number, support/contact information, and help links.

3. Wireframe Diagram (ASCII Representation) ☰

```

1 +-----+
2 | [Big Fam Logo]           Volunteer Panel           [Profile Icon] |
3 +-----+
4 | Sidebar:                  Main Content Area:          |
5 | +-----+     +-----+     +-----+     |
6 | | • My Shifts           | | [Volunteer Shift Card]   | |
7 | | • Profile              | | -----|     | |
8 | | • Messages             | | Shift: Food Stall Greeter | |
9 | | (• Availability, optional) | | Date: 2025-06-20 | |
10 | |                         | | Time: 10:00 AM - 4:00 PM | |
11 | |                         | | Location: Main Tent | |
12 | +-----+     | | Status: Upcoming | |
13 | |                         | | [View Details] [Mark as Completed] | |
14 | |                         | +-----+     | |
15 | |                         | +-----+     | |
16 | |                         | | [Volunteer Shift Card] | |
17 | |                         | | ... | |
18 | |                         | +-----+     | |
19 +-----+
20 | [Footer: Version, Support, Help Links]           |
21 +-----+
22

```

4. Interaction & User Flow ☰

1. Accessing the Volunteer Panel:

- The volunteer logs in and selects the Volunteer Panel from the dynamic navigation (e.g., via the “More” menu or their Profile Editor).
- The header displays “Volunteer Panel” with the volunteer’s profile icon.

2. Viewing Assigned Shifts (My Shifts Section):

- The default view shows a list or grid of volunteer shift cards.

- Each card displays the shift role, date/time, location, and current status.
- Tapping “View Details” on a shift card opens a more detailed view of that shift, which may include instructions or links to relevant resources.
- If applicable, the volunteer can mark a shift as completed using an action button.

3. Managing Personal Profile:

- Switching to the “Profile” section (via the sidebar/tab) shows the volunteer’s current profile information.
- The volunteer can update contact details, emergency contact information, and availability.
- Changes are saved and reflected immediately.

4. Reviewing Messages/Announcements:

- In the “Messages” section, the volunteer sees a list of communications from volunteer coordinators.
- Tapping a message opens the full message and provides options to reply or mark as read.

5. Footer and Global Navigation:

- The footer displays version information and support links.
- Global navigation elements remain accessible for quick access to the Home screen, Map, etc.

5. Additional Considerations ☰

- **Responsive Design:**

The Volunteer Panel layout should adapt smoothly across mobile, tablet, and desktop devices.

- **Accessibility:**

Ensure that all interactive elements (buttons, text fields, icons) include proper labels, contrast, and keyboard navigation support.

- **User Feedback:**

Provide immediate visual confirmation for actions such as saving profile updates, marking shifts as completed, and sending messages.

- **Offline Functionality:**

If needed, cache volunteer shift and profile data locally to ensure availability during connectivity issues.

6. Conclusion ☰

The Volunteer Panel UI/UX wireframe provides a clear and intuitive interface for festival volunteers to manage their shifts, update their profiles, and access internal communications. By focusing on key functionalities such as shift management, profile updates, and messaging, the panel ensures that volunteers have all the information and tools they need at their fingertips.

Please review the wireframe and user flow above, and let me know if you require any additional modifications or further details.

PAGE: Vendor Panel

Last Updated: Feb 11, 2025

1. Overview ☰

Purpose:

The Vendor Panel provides vendors with a dedicated space to manage their festival business details. In addition to updating their business profile, contact information, and location, vendors can now:

- **Manually manage their menu:**

Add, edit, or remove menu items by entering a Title, Description, and Cost. These items will be displayed under the vendor's profile throughout the app.

- **Manually update operating hours:**

Enter or update operating hours for each day of the festival. These hours will be shown under the vendor's profile.

Access:

- This panel is accessible only to users with the vendor role.

- It is reached via a dynamic role-based navigation link (e.g., from the "More" menu or Profile Editor).

2. Key UI Components ☰

A. Header & Global Navigation ☰

- **Header Elements:**

- **Left:** Big Fam Festival logo.
- **Center:** Page title ("Vendor Panel").
- **Right:** Profile icon (showing the vendor's profile) with a dropdown for account actions (e.g., settings, logout).

B. Navigation Sidebar (or Top Tab Navigation) ☰

- **Sidebar Menu Options:**

- **Profile Overview:** Displays current vendor business details and contact info.
- **Menu Management:**
 - **Manual Menu Editor:** Add, edit, and remove individual menu items (Title, Description, Cost).
- **Operating Hours & Location:**
 - **Operating Hours:** Manually update operating hours for each day.
 - **Location:** Set or update the vendor's location on an interactive map.
- **Communication:** Access messaging tools for direct communication with Big Fam staff.

C. Main Content Area ☰

1. Vendor Profile Overview ☰

- **Editable Fields:**

- **Business Name:** Text input for vendor's business name.
- **Description:** Multi-line text area for describing the business and offerings.
- **Contact Information:** Fields for email, phone, and possibly social media links.

- **Save/Update Button:** A prominent button to commit changes.

2. Manual Menu Editor (New/Updated Section) ☀

- **Purpose:** Allow vendors to manage their menu items manually.

- **Layout Elements:**

- **Current Menu List:**

- A scrollable list or table showing current menu items.
 - Each menu item displays:
 - **Title**
 - **Description**
 - **Cost**
 - Action buttons: "Edit" and "Delete."

- **Add New Menu Item Form:**

- Input fields for:
 - **Title:** Single-line text.
 - **Description:** Multi-line text area.
 - **Cost:** Numeric input (with currency symbol if desired).
 - An "Add Item" button to append the new menu item.

- **Dynamic Updates:**

- Menu items can be updated or removed throughout the weekend, with immediate visual feedback.

3. Operating Hours Management (Updated Section) ☀

- **Purpose:** Allow vendors to manually set or update operating hours.

- **Layout Elements:**

- **Hours Table/Editor:**

- A table or list with rows for each day (e.g., Monday through Sunday or specific festival days).
 - Each row includes:
 - **Day Label:** (e.g., "Day 1", "Day 2", or specific weekday names)
 - **Opening Time:** Editable time picker.
 - **Closing Time:** Editable time picker.
 - An "Update Hours" button to save changes.

- **Visual Feedback:**

- Changes are immediately reflected in the vendor profile view.

4. Location Section (Existing) ☀

- **Interactive Map:**

- Displays the vendor's current location.
 - A "Set/Update Location" button to adjust the vendor location via map interaction.

- **Confirmation:**

- Location updates are synchronized with the backend for display throughout the app.

5. Communication Section (Existing) ☀

- **Messaging Interface:**

- A button labeled "Message Big Fam Staff" to open a messaging modal or dedicated chat interface.
 - Enables direct communication with event organizers or support.

D. Footer (Optional) ☀

- **Footer Elements:**

- Version number, support/contact information, and links to documentation.

3. Wireframe Diagram (ASCII Representation)

Overall Layout

```

1 +-----+-----+
2 | [Big Fam Logo]           Vendor Panel           [Profile Icon] |
3 +-----+-----+
4 | Sidebar:                 | Main Content Area          |
5 | +-----+-----+-----+-----+
6 | | • Profile Overview    | | | Vendor Profile Overview   |
7 | | • Menu Management     | | | - Business Name: [_____] |
8 | | • Operating Hours & Location | | | - Description: [_____] |
9 | | • Communication       | | | - Contact Info: [Email] [Phone] |
10| +-----+-----+-----+-----+
11| | | | [Save Changes]      | | |
12| | +-----+-----+-----+-----+
13| | | Manual Menu Editor   | | |
14| | | [Current Menu Items: | | |
15| | | Title | Description | Cost | [Edit][Del] | | |
16| | | ...                   | | |
17| | | [Add New Item]        | | |
18| | | Title: [_____]         | | |
19| | | Description: [_____]   | | |
20| | | Cost: [_____]          | | |
21| | | [Add Item]             | | |
22| | +-----+-----+-----+-----+
23| | +-----+-----+-----+-----+
24| | | Operating Hours & Location | | |
25| | | - [Operating Hours Table with Time Pickers] | | |
26| | | - [Set/Update Location] | | |
27| | +-----+-----+-----+-----+
28| | +-----+-----+-----+-----+
29| | | Communication Section | | |
30| | | [Message Big Fam Staff] | | |
31| | +-----+-----+-----+-----+
32| +-----+-----+-----+-----+
33+-----+-----+
34| [Footer: Version, Support, Documentation Links] | |
35+-----+-----+

```

4. Interaction & User Flow

1. Accessing the Vendor Panel:

- A vendor logs in and selects the "Vendor Panel" from the dynamic navigation.
- The Vendor Panel loads with the header and sidebar showing available sections (Profile Overview, Menu Management, Operating Hours & Location, Communication).

2. Editing Vendor Profile:

- The vendor updates business details (name, description, contact info) and saves changes.
- Confirmation messages are shown upon successful update.

3. Managing the Menu:

- In the Manual Menu Editor section, current menu items are listed.
- The vendor can:

- **Add a New Menu Item:**

Enter Title, Description, and Cost, then click “Add Item.” The new item appears in the list immediately.

- **Edit/Delete an Existing Menu Item:**

Use the “Edit” and “Delete” buttons next to each menu item to update or remove it.

- Updates are synchronized and reflected in the vendor profile view across the app.

4. Updating Operating Hours:

- The vendor accesses the Operating Hours section.
- They update times using time pickers for each day (or festival-specific day).
- Tapping “Update Hours” saves the changes and displays a confirmation.

5. Setting/Updating Location:

- The vendor taps the “Set/Update Location” button in the Location section.
- An interactive map allows the vendor to update their geolocation.
- Once updated, the new location is saved and synchronized with the backend.

6. Accessing Communication:

- The vendor taps “Message Big Fam Staff” to open the messaging interface.
- They can send and receive messages regarding vendor-related inquiries or support.

7. Footer & Global Navigation:

- The footer remains available with support links and version information.
-

5. Additional Considerations

- **Responsive Design:**

Ensure that the layout and interactive elements (input fields, time pickers, map) adapt smoothly across various devices.

- **Accessibility:**

All form elements, buttons, and navigation items must have accessible labels and sufficient contrast.

- **User Feedback:**

Provide clear visual feedback (toast messages, confirmation dialogs) when menu items and operating hours are updated.

- **Security:**

Validate and securely transmit any data (especially file uploads or location data) to the backend.

- **Offline Support:**

Where possible, cache vendor data locally to allow updates when connectivity is limited, synchronizing changes once online.

6. Conclusion

This updated Vendor Panel wireframe now includes comprehensive functionality for manual menu management and operating hours updates. Vendors can easily add, edit, and remove menu items with a Title, Description, and Cost, and they can update their operating hours through an intuitive time picker interface. These changes are integrated seamlessly into the Vendor Panel, ensuring that updated business information is visible throughout the app.

PAGE: Artist Panel

1. Overview

Purpose:

The Artist Panel provides a dedicated space for artists to manage their festival presence. This panel enables artists to:

- View their performance schedule (with details like stage, time, and venue).
- Update their personal profile (profile picture, bio, and social media links).
- Directly contact festival staff or their assigned director for inquiries or coordination.

Access:

- This panel is accessible only to users with the artist role.
 - It is reached via a dynamic role-based navigation link (visible in the “More” menu or Profile Editor).
-

2. Key UI Components

A. Header & Global Navigation

- **Header Elements:**
 - **Left:** Big Fam Festival logo (consistent with the overall app branding).
 - **Center:** Page Title (e.g., “Artist Panel”).
 - **Right:** Profile icon (shows the logged-in artist’s picture) with a dropdown for quick account actions.

B. Navigation Sidebar (Optional Tabbed Navigation)

- **Purpose:**

Provide quick access to the major functions within the Artist Panel.
- **Sidebar / Tab Options:**
 - **Performance Schedule:** View a list or calendar of scheduled performances.
 - **Profile:** Edit personal profile details, including profile picture, bio, and social media links.
 - **Contact:** Access direct communication tools to reach festival staff or the assigned director.

C. Main Content Area

1. Performance Schedule Section

- **Layout:**
 - A list or calendar view displaying upcoming and past performance events.
 - Each event card includes:
 - **Event Image:** A visual representation or artist photo.
 - **Event Title:** (e.g., “Live at Stage A”).
 - **Details:** Stage name, date, and time.
 - **Action:** A “View Details” button for additional event information.

- **Purpose:**

Ensure artists can quickly see where and when they are scheduled to perform.

2. Profile Management Section

- **Layout:**

- Similar to a simplified profile editor for artists.
- **Editable Fields:**
 - **Profile Picture:** With an edit icon to change the image.
 - **Bio:** A text area for the artist's description.
 - **Social Media Links:** Input fields or icons linking to platforms like Instagram, Twitter, etc.

- **Purpose:**

Allow artists to update and maintain their public profile for festival promotion.

3. Contact Section ☰

- **Layout:**

- A simple interface that displays:
 - **Assigned Director/Staff Contact:** Name, photo, and contact method (e.g., email or chat).
 - **Message Form:** A text area with a "Send Message" button to quickly contact the assigned director or festival staff.

- **Purpose:**

Provide a streamlined way for artists to reach out with queries or concerns.

D. Footer (Optional) ☰

- **Elements:**

- Basic festival info, version number, and links to help or documentation.

3. Wireframe Diagrams (ASCII Representation) ☰

A. Overall Layout ☰

```

1  +-----+
2  | [Big Fam Logo]           Artist Panel           [Profile Icon] |
3  +-----+
4  | Sidebar / Tab Navigation:
5  |   • Performance Schedule  • Profile   • Contact
6  +-----+
7  |                         Main Content Area
8  |
9  |   [Performance Schedule Section]
10 |
11 |   | [Event Card]
12 |   |   - Event Image: [Thumbnail]
13 |   |   - Event Title: "Live at Stage A"
14 |   |   - Stage: Stage A, Date, Time
15 |   |   - [View Details]
16 |
17 |
18 |   [Profile Management Section]
19 |
20 |   | Profile Picture  [Edit Icon]
21 |   | Name:    [Artist Name]
22 |   | Bio:     [Editable Text Area]
23 |   | Social:  [Instagram] [Twitter] [Facebook]
24 |
25 |
26 |   [Contact Section]
27 |

```

```

28 | | Assigned Director: [Director Name & Photo] | |
29 | | Message: [_____] [Send Message] | |
30 | -----+-----+-----+-----+
31 +-----+-----+-----+-----+
32 | [Footer: Festival Info, Version, Help Links] | |
33 +-----+-----+-----+-----+

```

B. Sidebar / Tab Navigation Detail ↗

```

1 +-----+-----+
2 | • Performance Schedule | |
3 | • Profile | |
4 | • Contact | |
5 +-----+-----+

```

4. Interaction & User Flow ↗

1. Accessing the Artist Panel:

- An artist logs in and selects the "Artist Panel" link from the dynamic navigation (e.g., via the More menu or Profile Editor).
- The Artist Panel loads with the header displaying "Artist Panel" and the artist's profile icon.

2. Navigating Between Sections:

- The sidebar (or tab navigation) allows the artist to switch between:
 - **Performance Schedule:** Quickly view upcoming and past performance details.
 - **Profile:** Edit their profile picture, bio, and social media links.
 - **Contact:** Access contact information for their assigned director and send messages.

3. Performance Schedule Interaction:

- Artists review their scheduled events. Tapping "View Details" on an event card opens additional event information.

4. Profile Management Interaction:

- Artists update their profile details using in-place editing. Changes are saved and immediately reflected.
- Social media links are updated with clickable icons or input fields.

5. Contact Interaction:

- The Contact section displays the assigned director's information.
- The artist can type a message in the provided text area and tap "Send Message" to initiate direct communication.

6. Feedback & Navigation:

- Visual cues (such as confirmation messages or loaders) are provided during updates.
- The layout is fully responsive and accessible.

5. Additional Considerations ↗

• Responsive Design:

Ensure that the Artist Panel adapts smoothly to mobile, tablet, and desktop devices.

• Accessibility:

All interactive elements must have accessible labels, appropriate contrast, and keyboard navigation support.

• User Feedback:

Provide confirmation messages for updates (e.g., "Profile updated successfully") and error handling for failed actions.

• Integration:

The Artist Panel should integrate seamlessly with other parts of the Big Fam Festival App, including shared authentication and navigation components.

6. Conclusion ☀

The Artist Panel is designed to empower artists with an intuitive, all-in-one interface where they can view their performance schedule, update their profile, and directly contact festival staff or their assigned director. This wireframe provides a clear blueprint for the development and design teams to create high-fidelity prototypes in tools like Figma or Sketch.

PAGE: Interactive Map & Embedded Navigation

1. Overview

The Interactive Map page allows users to navigate the festival grounds with real-time location tracking and interactive Points of Interest (POIs). This update focuses on enhancing the campsite management feature. Once a user marks their campsite using the "Mark My Campsite" button, the following changes occur:

- The floating action button in the bottom-right changes to a Home icon labeled "Take Me Home." Tapping this button launches the embedded navigation to guide the user back to their marked campsite.
 - Additionally, a text button labeled "Remove My Campsite" is displayed at the bottom of the map. Tapping this button will remove the user's campsite marker from the map and update their profile accordingly.
-

2. Key UI Components (Updated for Campsite Management)

A. Full-Screen Interactive Map

- **Map Canvas:**
Displays an interactive, full-screen map with standard zoom/pan functionality.
- **Live Position Indicator:**
A blinking icon shows the user's current location, updated in real time.

B. "Mark My Campsite" Button (Pre-Marking)

- **Before Marking:**
 - A floating action button (FAB) in the bottom-right corner labeled "Mark My Campsite."
 - Tapping the button triggers the process to obtain the user's location, convert it to a three-word address via the WhatThreeWords API, and then mark the campsite.

C. Post-Marking Updates

- **Floating Action Button Change:**
 - After the campsite is marked, the original "Mark My Campsite" FAB is replaced with a new button.
- **New FAB:**
 - Displays a Home icon.
 - Label: "Take Me Home."
 - Tapping this button launches the embedded navigation overlay directing the user back to their campsite.
- **Remove Campsite Button:**
 - A text button is added at the bottom of the map.
 - **Label:** "Remove My Campsite"
 - This button allows the user to remove their marked campsite. Tapping it sends an update to the backend to delete the campsite record and removes the campsite marker from the map.

D. Transfer & Sync with Backend

- When a user marks or removes their campsite, the updated information is sent to the backend for storage.
 - Admins can view all marked campsites via a dedicated admin view.
-

3. Updated Wireframe Diagrams ↗

A. Pre-Marking State (Before Campsite is Marked) ↗

```
1 +-----+
2 | [Big Fam Logo]      [🔍 Search Bar]      [Profile Icon/LOGIN] |
3 +-----+
4 |
5 | +-----+
6 | |           Interactive Map           |
7 | |                                     |
8 | |   [Live Position Indicator]        |
9 | |   (Blinking dot)                 |
10| |
11| |   [Custom POI Markers for stages, vendors,    |
12| |   facilities, campsites, friend locations]   |
13| |
14| +-----+
15|
16|   [Floating Action Button: Mark My Campsite]   |
17+-----+
18| [Legend: Stage | Vendor | Facility | Campsite | Friend]  [Offline Mode Indicator] |
19+-----+
20|
```

B. "Mark My Campsite" Confirmation Modal ↗

```
1 +-----+
2 |           Mark My Campsite           | <- Modal Header
3 +-----+
4 | Your current location:             |
5 | "apple.banana.cherry"    <- (WhatThreeWords address) |
6 |
7 | Would you like to mark this as your campsite? |
8 |
9 | [Confirm]           [Cancel]           | <- Action Buttons
10+-----+
11|
```

C. Post-Marking State (After Campsite is Marked) ↗

```
1 +-----+
2 | [Big Fam Logo]      [🔍 Search Bar]      [Profile Icon/LOGIN] |
3 +-----+
4 |
5 | +-----+
6 | |           Interactive Map           |
7 | |                                     |
8 | |   [Live Position Indicator]        |
9 | |   (Blinking dot)                 |
10| |
11| |   [Custom POI Markers for stages, vendors,    |
12| |   facilities, campsites (includes my campsite) |
13| |   & friend locations]               |
14| +-----+
15|
16|   [Floating Action Button: Take Me Home]   |
17|
```

```

17 +-----+
18 | [Remove My Campsite] (Text Button at bottom of map) |
19 | [Legend: Stage | Vendor | Facility | Campsite | Friend] [Offline Mode Indicator] |
20 +-----+
21 | [Embedded Navigation Overlay (activated when 'Take Me Home' is tapped)] |
22 +-----+
23

```

4. Interaction & User Flow

1. Initial State (Pre-Marking):

- User is on the Interactive Map page.
- The floating action button labeled “Mark My Campsite” is visible in the bottom-right corner.
- User taps the “Mark My Campsite” button.
- The app retrieves the current geolocation and uses WhatThreeWords to generate a three-word address.
- A confirmation modal appears with the three-word address for verification.
- User confirms; the campsite is marked and sent to the backend.

2. Post-Marking State:

- The floating action button changes to a Home icon with the label “Take Me Home.”
- A text button labeled “Remove My Campsite” appears at the bottom of the map.
- Tapping “Take Me Home” launches the embedded navigation overlay, guiding the user back to their marked campsite.
- Tapping “Remove My Campsite” removes the campsite marker and sends an update to the backend to delete the campsite record.

3. Feedback & Offline Sync:

- Visual feedback (toast messages) confirms the campsite has been marked, transferred, or removed.
- If offline, actions are queued and synchronized when connectivity is restored.

5. Additional Considerations

• Responsive Design:

Ensure the floating button and text button adapt gracefully to different screen sizes.

• Accessibility:

Provide accessible labels for both the “Take Me Home” button and “Remove My Campsite” text button.

• User Confirmation:

Include confirmation dialogs and visual feedback to prevent accidental removal of a marked campsite.

• Backend Integration:

Ensure seamless synchronization with the backend for marking and unmarking campsites, including error handling and data validation.

6. Conclusion

This updated wireframe for the Interactive Map & Embedded Navigation page now includes enhanced campsite management. Once a user marks their campsite, the floating action button is replaced with a “Take Me Home” button for easy navigation, and a text button is added at the bottom to remove the campsite if needed. This design provides a clear, intuitive, and efficient interface for managing campsite locations.

PAGE: Events

Last Updated: [@Robert Eriksen](#) Feb 11, 2025

1. Overview

The Global Events Page enables users to browse all available festival events. It incorporates robust filtering options to refine the list by day, stage, and other useful criteria, ensuring users can quickly find events that match their interests. The page leverages our persistent top navigation (with the search bar) and provides a clear, card-based layout for events.

2. Key UI Components

A. Header (Persistent - Already Defined)

- **Elements:**
 - **Left:** Big Fam Logo
 - **Center:** Global Search Bar (always available)
 - **Right:** Profile Icon (or LOGIN button if unauthenticated)

B. Events Filter Section

- **Purpose:**

Allow users to narrow down the events list using multiple criteria.
- **Filters Include:**
 - **Day Filter:**
 - A horizontal scrollable list of days (e.g., "Day 1", "Day 2", "Day 3") allowing the user to select a specific festival day.
 - **Stage Filter:**
 - A set of buttons or tabs labeled "All Stages", "Stage A", "Stage B", "Stage C", "Stage D".
 - When a stage is selected, only events on that stage (and day, if also filtered) are displayed.
 - **Additional Filters (Optional):**
 - **Time Slot:** For example, Morning, Afternoon, Evening (if applicable).
 - **Genre or Type:** If events can be categorized by music genre or event type, a dropdown or tag filter can be added.
 - **Sort Order:** Optionally, a sort by "Time", "Popularity", or "Alphabetical" might be useful.
- **UI Pattern:**
 - Use a combination of horizontal scrollable filter chips and dropdowns for more detailed criteria.
 - The filter section should be collapsible on smaller screens if necessary, to maximize space for event cards.

C. Event Cards

- **Event Card Elements:**
 - **Event Image:** A representative image (artist, performance, or stage image).
 - **Event Title:** Artist name and/or event title.
 - **Event Details:**
 - Stage name and time (e.g., "Stage A - 7:30 PM").

- A brief description or tagline.
- **Action Button (Heart Icon):**
 - An icon that, when tapped, adds the event to the user's personal schedule (turns active once added).
- **Layout:**
 - Cards arranged in a grid or list format.
 - Clear visual hierarchy with high-quality images and legible text.

D. Pagination / Infinite Scrolling ↗

- **Purpose:**
Ensure a smooth experience if there are many events.
 - **Implementation:**
 - Use infinite scrolling or pagination controls at the bottom to load more events as the user scrolls.
-

3. Wireframe Diagrams ↗

A. Filter Section & Event Cards (Desktop/Mobile Version) ↗

```

1 +-----+
2 | [Big Fam Logo]           [🔍 Search]   [Profile Icon / LOGIN] |
3 +-----+
4 | [Day Filter: Day 1] [Day 2] [Day 3]           |
5 | [Stage Filter: All Stages] [Stage A] [Stage B] [Stage C] [Stage D] |
6 | [Optional: Time Slot ▼] [Optional: Genre ▼] [Sort By: Time ▼] |
7 +-----+
8 | +-----+   +-----+   +-----+   +-----+ |
9 | | [Event Image]   | | [Event Image]   | | [Event Image]   | |
10 | | DJ Nightwave  | | Indie Sparks   | | Rock Revolution | |
11 | | Stage A - 7:30 PM | | Stage B - 9:00 PM | | Stage C - 10:00 PM | |
12 | | "❤️ Add to Schedule" | | "❤️ Add to Schedule" | | "❤️ Add to Schedule" | |
13 | +-----+   +-----+   +-----+   +-----+ |
14 | |
15 | +-----+   +-----+ |
16 | | [Event Image]   | | [Event Image]   | |
17 | | Drum Circle    | | Pop Sensation | |
18 | | Stage D - 8:45 PM | | Stage A - 11:00 AM | |
19 | | "❤️ Add to Schedule" | | "❤️ Add to Schedule" | |
20 | +-----+   +-----+ |
21 +-----+
22 | [Load More Events] (Pagination or Infinite Scroll) |
23 +-----+
24

```

B. Global Events Page (Mobile-Focused) ↗

```

1 +-----+
2 | [Big Fam Logo]   [🔍 Search]   [Profile/LGIN] |
3 +-----+
4 | [Day: Day 1] [Day: Day 2] [Day: Day 3]           |
5 +-----+
6 | [Stage: All]  [Stage A]  [Stage B]  [Stage C]  [Stage D] |
7 +-----+
8 | [Optional Filters: Time ▼ | Genre ▼ | Sort By: Time ▼] |

```

```

9 +-----+
10 | +-----+ +-----+ |
11 | | [Event Card] | | [Event Card] | |
12 | | DJ Nightwave | | Indie Sparks | |
13 | | Stage A - 7:30 PM | | Stage B - 9:00 PM | |
14 | | "❤️ Add to Schedule" | | "❤️ Add to Schedule" | |
15 | +-----+ +-----+ |
16 | +-----+ +-----+ |
17 | | [Event Card] | | [Event Card] | |
18 | | Rock Revolution | | Drum Circle | |
19 | | Stage C - 10:00 PM | | Stage D - 8:45 PM | |
20 | | "❤️ Add to Schedule" | | "❤️ Add to Schedule" | |
21 | +-----+ +-----+ |
22 |-----+ |
23 | [Load More Events] | |
24 +-----+
25

```

4. Interaction & User Flow

1. Filtering Events:

- The user can select a day filter (e.g., Day 1, Day 2, etc.).
- The user then selects a stage filter from the provided buttons.
- Additional optional filters (time slot, genre, sort order) further refine the event list.
- The filter selections update the event cards displayed below in real time.

2. Adding/Removing Events:

- Each event card displays a heart icon.
- Tapping the heart icon on an event card (from the Global Events page) adds the event to the user's personal schedule.
- In the personal schedule view (accessed via the bottom nav), tapping the active heart icon removes the event.

3. Search Functionality:

- The search bar is always available in the top navigation, allowing users to search for events at any time.
- Search results update dynamically and integrate with the filter selections if applicable.

4. Pagination/Infinite Scroll:

- As the user scrolls, more events are loaded either via infinite scrolling or a "Load More" button at the bottom of the page.

5. Summary

The updated Global Events Page provides a streamlined experience where users can easily filter events by day, stage, and other criteria. The interface leverages a dedicated filter section above a clear, card-based event listing, ensuring that users can quickly add events to their schedule using the heart icon. With a persistent top navigation bar for search and a bottom navigation bar for core actions, the Events page is designed to be both intuitive and efficient, supporting the festival experience seamlessly—even on mobile devices.

Please review the wireframes and interaction flow above and provide feedback for any adjustments or further details you'd like to include.

PAGE: Profile Editor

1. Overview ↗

Purpose:

The Profile Editor provides a dedicated settings page where all users (attendees, vendors, artists, staff, admins) can update their personal information. In addition to editing the profile picture, contact details, notification preferences, and device permissions, this page now includes options to control the sharing of sensitive location data. Two new toggle switches are added:

- **Share My Campsite**
- **Share My Location**

When enabled, an informational pop-up will explain that only users in the person's friend list can see these details.

Access:

- The Profile Editor is accessed via the profile icon in the top navigation bar.
 - It is separate from the schedule builder and provides a full account settings experience.
-

2. Key Layout Elements ↗

A. Profile Picture Section ↗

- **Editable Profile Picture:**
 - A circular image displays the current profile picture or a default avatar.
 - An "Edit" icon (pencil overlay) allows users to update the image (with cropping/preview functionality).

B. Contact Information ↗

- **Form Fields:**
 - **Name:** Editable text field.
 - **Email:** Editable text field with validation.
 - **Phone:** Editable text field.

C. Notification Preferences ↗

- **Toggle Switches:**
 - Options to enable/disable Push, Email, and SMS notifications.
 - Clear visual states (ON/OFF) for each.

D. Device Permissions ↗

- **Permission List:**
 - Display current status (Allowed/Not Allowed) for Location, Camera, Notifications, and Microphone.
 - Each permission includes a "Change" or "Request/Review" button.
 - Brief instructions explain the importance of each permission.

E. Sharing Preferences (New Section) ↗

- **Purpose:**

Allow users to control the visibility of their campsite and live location data.

- **Toggle Switches:**

- **Share My Campsite:**

- A toggle that, when enabled, allows the user's marked campsite location to be shared.

- **Share My Location:**

- A toggle that, when enabled, allows the user's live location to be shared with added friends.

- **Info Pop-Up/Confirmation Screen:**

- Upon toggling either option ON, a modal appears with a brief explanation:

- **Example Message:** "By enabling this option, only the friends you have added will be able to see your campsite/live location. Your privacy is protected, and only approved contacts can view this information."

- The user must confirm their selection to enable the sharing option.

F. Ticket Information

- **View My Tickets Button:**

- A button labeled "View My Tickets" that navigates the user to a Ticket Details page or opens a modal listing all tickets and their validation statuses.

G. Role-Specific Panel Access

- **Dynamic Panel Link:**

- A section that conditionally displays a role-specific panel link (e.g., Admin Panel, Staff Panel, Artist Panel, or Vendor Panel) based on the user's role.

H. Account Actions

- **Buttons/Links:**

- **Logout:** Logout button.
 - **Language Preferences:** Option to change the app language.
 - **Privacy Settings:** Link to adjust privacy/security settings.

3. Updated Wireframe Diagram (ASCII)

```
1 +-----+
2 |          Profile Editor          |
3 |-----|
4 | [Profile Picture]  [Edit Icon]   |
5 |-----|
6 | Name:  [_____]                 |
7 | Email: [_____]                 |
8 | Phone: [_____]                 |
9 |-----|
10| Notification Preferences:      |
11| - Push:  [ON/OFF]              |
12| - Email: [ON/OFF]              |
13| - SMS:   [ON/OFF]              |
14|-----|
15| Device Permissions:           |
16| - Location:  [Allowed]  [Change] |
17| - Camera:    [Not Allowed] [Request/Review] |
18| - Notifications: [Allowed] [Review] |
19| - Microphone: [Allowed/Not Allowed] [Change] |
20| (Adjust these permissions in your device settings) |
21|-----|
```

```

22 | Sharing Preferences:
23 |   - Share My Campsite: [ON/OFF] (Info icon)
24 |   - Share My Location: [ON/OFF] (Info icon)
25 |   [Info Popup: "By enabling this option, only your added
26 |     friends will be able to view your campsite/live location."]
27 |
28 | Ticket Information:
29 |   [View My Tickets]
30 |
31 | Role-Specific Panel Access:
32 |   {if user.role === 'admin'}
33 |     [Admin Panel]
34 |   {else if user.role === 'staff'}
35 |     [Staff Panel]
36 |   {else if user.role === 'artist'}
37 |     [Artist Panel]
38 |   {else if user.role === 'vendor'}
39 |     [Vendor Panel]
40 |
41 | Account Actions:
42 |   [Logout] [Language Preferences] [Privacy Settings]
43 ++
44

```

4. Interaction & User Flow ↗

1. Accessing the Profile Editor:

- The user taps the profile icon in the top navigation bar.
- The Profile Editor page loads displaying personal details, notification preferences, device permissions, sharing preferences, and account actions.

2. Updating Contact Information & Notification Preferences:

- The user edits name, email, phone, and toggles notification options.
- Changes are validated and saved immediately.

3. Reviewing/Adjusting Device Permissions:

- The user reviews current permission statuses.
- Tapping the "Change" or "Request/Review" buttons opens native dialogs to modify permissions.

4. Enabling Sharing Preferences:

- The user toggles "Share My Campsite" or "Share My Location" ON.
- An informational pop-up appears explaining:
"By enabling this option, only the friends you have added will be able to see your campsite/live location. Your privacy is protected, and only approved contacts can view this information."
- The user confirms the selection, and the toggle remains enabled.
- If the user cancels, the toggle reverts to OFF.

5. Viewing Ticket Information:

- Tapping the "View My Tickets" button navigates the user to the Ticket Details page.

6. Accessing Role-Specific Panels:

- Based on the user's role, the dynamic panel link is displayed.
- Tapping this link navigates the user to the corresponding panel (e.g., Admin Panel).

7. Logging Out or Accessing Additional Settings:

- The user can tap Logout or adjust language/privacy settings as needed.

5. Additional Considerations

- **Responsive Design:**

Ensure all elements, especially the new sharing toggles and info pop-up, render correctly on various devices.

- **Accessibility:**

All toggles, buttons, and pop-ups must have proper labels and keyboard navigation support.

- **User Feedback:**

Provide clear visual confirmation when sharing preferences are enabled or disabled.

- **Offline Support:**

Profile changes made offline are queued and synchronized when connectivity is restored.

6. Conclusion

This updated Profile Editor design now includes two new toggle switches for "Share My Campsite" and "Share My Location" along with an informational pop-up explaining that only added friends will have access to this information. This enhancement gives users precise control over their privacy while ensuring that all account settings are accessible in one comprehensive interface.

PAGE: Friends

1. Overview ↗

The Friends page is designed to allow users to easily manage their social connections within the Big Fam Festival App. Instead of a dynamic search bar, users now have a simple input field where they can enter an email address or phone number to send a friend request. The page still displays pending friend requests and a list of confirmed friends for easy management.

Key Features:

- **Input Field for Friend Addition:**

A text input for entering an email or phone number, accompanied by an "Add Friend" button.

- **Pending Friend Requests:**

A section that lists incoming friend requests with options to Accept or Decline.

- **Confirmed Friends List:**

A list of friends with their profile pictures and names.

- **User Feedback:**

Immediate confirmation when a friend request is sent or an action is completed.

2. Key UI Components ↗

A. Header ↗

- **Title:** "Friends"

- **Navigation:**

- A back arrow or hamburger menu for global navigation.
- Consistent with the global top navigation (Big Fam logo, Search, Profile/LOGIN).

B. Add Friend Section ↗

- **Input Field:**

- A text input labeled "Enter email or phone number" where users can type the contact information.

- **Add Friend Button:**

- A button labeled "Add Friend" that, when tapped, sends a friend request to the entered contact information.

- **Feedback:**

- A confirmation message is displayed once the request is sent.

C. Friend Requests Section ↗

- **Pending Requests List:**

- A separate sub-section listing incoming friend requests.
- Each request includes:
 - The requester's profile picture.
 - Name (and possibly a mutual friends count).
 - Action buttons: "Accept" and "Decline."

D. Confirmed Friends List ↗

- **Friend List Items:**

- A scrollable list showing confirmed friends.
- Each friend is represented by:
 - Profile picture.
 - Name.
 - (Optional) Online status indicator.
- Tapping on a friend can open their detailed profile or start a chat.

3. Wireframe Diagram (ASCII Representation)

```

1 +-----+
2 | [←] Friends
3 +-----+
4 | Add Friend:
5 | [Enter email or phone number _____] [Add Friend] |
6 +-----+
7 | Pending Requests
8 | -----
9 | | [Profile Pic] Friend B [Accept] [Decline] |
10 | -----
11 | | [Profile Pic] Friend C [Accept] [Decline] |
12 | -----
13 | Confirmed Friends
14 | -----
15 | | [Profile Pic] Friend A | |
16 | -----
17 | | [Profile Pic] Friend D | |
18 | -----
19 | | [Profile Pic] Friend E | |
20 | -----
21 +-----+

```

4. Interaction & User Flow

1. After Submitting a Medical Request:

- The "Request Medical" modal closes, and a post-submission modal appears with the message "Request Submitted" and the two options: "Take Me To Medical" and "Stay Put."

2. If the User Selects "Take Me To Medical":

- The app immediately launches the embedded navigation overlay.
- Turn-by-turn directions guide the user to the nearest medical facility.
- Route information (ETA, distance) is displayed on the overlay.
- The user can cancel navigation at any time using the "Cancel Navigation" button.

3. If the User Selects "Stay Put":

- The app displays an overlay or updates the map view to show nearby medical facilities.
- Each facility is listed with distance and estimated arrival time.
- The user can tap "Navigate" for a specific facility to receive turn-by-turn directions.
- Alternatively, the user can simply review the options and remain at their current location.

4. Confirmation & Feedback:

- Upon selection and processing, appropriate confirmation messages are displayed (e.g., "Navigation Started" or "Nearby medical facilities updated").
- If the user is offline, actions are queued and synchronization is attempted once connectivity is restored, with clear notifications.

5. Additional Considerations ☀

- **Urgency & Clarity:**

Ensure that the modals and overlays use clear, high-contrast colors (with red accents for urgency) so users immediately understand the context.

- **Accessibility:**

All buttons and text elements must include accessible labels and meet contrast guidelines.

- **Offline Handling:**

The system should gracefully handle offline scenarios by queuing requests and providing fallback messaging.

- **User Confirmation:**

Include confirmation messages and the option to review the selected medical facility before finalizing navigation, ensuring that users are making informed choices.

6. Conclusion ☀

This updated wireframe integrates a post-submission flow for the "Request Medical" feature. Users now have the option to either "Take Me To Medical" with immediate embedded navigation or "Stay Put" to view nearby medical facilities with navigation details. This design ensures that users can quickly and clearly act on their medical needs in an emergency while receiving relevant route and facility information.

COMP: Request Medical

1. "Request Medical" in the Profile Editor ☰

Overview ☰

Within the Profile Editor, in the Account Actions section (located at the very bottom), a new button labeled "**Request Medical**" is added. When tapped, this button launches a modal that allows the user to select an urgency level and optionally add a brief description. The modal then sends the user's current location and details to the backend for immediate review by festival staff.

Key Layout Elements ☰

- **Account Actions Section (Profile Editor):**
 - Existing Buttons: Logout, Language Preferences, Privacy Settings.
 - **New Button:** "Request Medical" (styled in red with an alert icon, e.g., a red cross or exclamation mark).
- **"Request Medical" Modal:**
 - **Modal Header:** "Request Medical"
 - **Urgency Dropdown:**
 - Options include: IMMEDIATE, URGENT, STANDARD, NO RUSH (or other intermediate options if desired).
 - **Optional Description Field:**
 - A text input where the user may briefly describe the issue.
 - **Action Buttons:**
 - **Send Request:** Confirms and sends the emergency alert with the current location.
 - **Cancel:** Closes the modal without sending any data.
 - **Information Note:**
 - A message (displayed in the modal or as a pop-up) explaining:
"By sending this request, your current location and details will be shared only with festival staff. Only friends you have added can view your live location if sharing is enabled."

"Request Medical" Modal Diagram (Profile Editor) ☰

```
1 +-----+
2 |           Request Medical          | <-> Modal Header (Red Accent)
3 +-----+
4 | Select Urgency Level:             |
5 |   [ IMMEDIATE ▾ ]                | <-> Dropdown (Options: IMMEDIATE, URGENT,
6 |   STANDARD, NO RUSH)              |
7 | Additional Information (Optional):|
8 |   [ _____ ]                      |
9 |                                     |
10| (Your current location will be sent to staff along with these details.) |
11|                                     |
12| [Send Request]  [X Cancel]        | <-> Action Buttons
13+-----+
```

2. Interaction & User Flow (for Request Medical) ☰

1. Access from Profile Editor:

- User is in the Profile Editor.
- They tap the "Request Medical" button located in the Account Actions section.
- The Request Medical modal appears.
- User selects the urgency level from the dropdown and optionally enters a brief description.
- Tapping "Send Request" sends the current location along with the selected urgency and description to the backend.
- A confirmation message appears (e.g., "Medical request sent. Help is on the way!").

2. Access from More Menu:

- User opens the More menu from the Bottom Navigation.
- They tap the "Request Medical" option (displayed in red).
- The same Request Medical modal appears.
- The process proceeds as above.

3. User Feedback:

- The system confirms successful submission.
 - If the action is canceled, the modal closes without sending any data.
-

3. Additional Considerations ☰

• Security & Permissions:

Ensure the app has appropriate location permissions before allowing a medical request.

• Offline Handling:

If the user is offline, notify them that the request will be queued and sent when connectivity is restored.

• Accessibility:

All interactive elements (dropdown, buttons, toggles) must have proper labels and support screen readers.

• Urgency Options:

The urgency dropdown should provide clear, descriptive labels (e.g., IMMEDIATE, URGENT, STANDARD, NO RUSH) to help users accurately convey the severity of their situation.

4. Conclusion ☰

The updated design integrates a new "Request Medical" feature accessible both from the Profile Editor and the Bottom Navigation's More menu. This feature allows users to quickly send a medical request along with an urgency level and optional details, ensuring that festival staff can respond promptly and appropriately. The design is both user-friendly and sensitive to the need for clear, immediate communication in emergency situations.

Post-Submission Options UI/UX Wireframe ☰

Last Updated: [Insert Date]

1. Overview ☰

After a user submits a medical request via the "Request Medical" modal, the system will present two options:

- **Stay Put:** The user remains at their current location. In this mode, the app shows the nearest medics or medical stations on the map with navigation options and estimated time of arrival (ETA) for each.
- **Take Me To Medical:** The user opts for immediate navigation. The app then launches an embedded navigation overlay that guides them to the nearest medical station.

This updated flow ensures that users have clear choices during a medical emergency and are provided with relevant information based on their decision.

2. Key UI Components ☰

A. Post-Submission Options Modal ☰

- **Modal Header:**

- "Request Submitted" with a confirmation message (e.g., "Your medical request has been sent to festival staff. What would you like to do next?").

- **Option Buttons:**

- **Stay Put:**

- Label: "Stay Put"
 - When tapped, the app remains at the user's current location and displays a list/overlay of nearby medical locations with navigation between them and an estimated time to arrival.

- **Take Me To Medical:**

- Label: "Take Me To Medical"
 - When tapped, the app launches the embedded navigation overlay to guide the user directly to the nearest medical station.

B. Nearest Medics Overlay (For "Stay Put" Option) ☰

- **Overlay Content:**

- **List or Map Markers:**

- A list of the nearest medical facilities is displayed, or the map view is updated to highlight nearby medics.

- **Navigation Information:**

- Each medical location displays navigation details (e.g., distance, ETA).

- **Action Button:**

- Optionally, each entry may have a "Navigate" button that launches navigation specifically for that location.

C. Embedded Navigation (For "Take Me To Medical") ☰

- **Navigation Overlay:**

- Provides turn-by-turn directions to the nearest medical facility.
 - Displays route overlays, estimated time, and distance.

3. Wireframe Diagrams ☰

A. Post-Submission Options Modal ☰

```
1 +-----+
2 |           Request Submitted          | <-- Modal Header
3 +-----+
4 | Your medical request has been sent to festival staff.      |
5 | What would you like to do next?          |
6 |                                         |
7 | [Take Me To Medical]   [Stay Put]        | <-- Option Buttons
8 +-----+
```

B. Nearest Medics Overlay (If "Stay Put" is Selected) ↗

```
1 +-----+
2 |           Nearest Medical Facilities      | <-- Overlay Header
3 +-----+
4 | 1. Medic Station Alpha                   |
5 |   - Distance: 200m    |   ETA: 3 min   |
6 |   [Navigate]                           |
7 |-----|
8 | 2. Medic Station Bravo                 |
9 |   - Distance: 350m    |   ETA: 5 min   |
10|  [Navigate]                          |
11|-----|
12| 3. First Aid Tent                     |
13|   - Distance: 500m    |   ETA: 7 min   |
14|  [Navigate]                          |
15|-----|
16| [Close Overlay]                      |
17+-----+
18
```

C. Embedded Navigation Overlay (If "Take Me To Medical" is Selected) ↗

```
1 +-----+
2 |           Navigation: Nearest Medical Facility | <-- Overlay Header
3 +-----+
4 |  [Route Overview]                         |
5 |   - Turn-by-turn directions are displayed here |
6 |   - Estimated Time: 3 minutes, Distance: 200m   |
7 |                                                 |
8 |  [Cancel Navigation]                      |
9 +-----+
```

4. Interaction & User Flow ↗

1. After Submitting a Medical Request:

- The "Request Medical" modal closes, and a post-submission modal appears with the message "Request Submitted" and the two options: "Take Me To Medical" and "Stay Put."

2. If the User Selects "Take Me To Medical":

- The app immediately launches the embedded navigation overlay.
- Turn-by-turn directions guide the user to the nearest medical facility.
- Route information (ETA, distance) is displayed on the overlay.
- The user can cancel navigation at any time using the "Cancel Navigation" button.

3. If the User Selects "Stay Put":

- The app displays an overlay or updates the map view to show nearby medical facilities.
- Each facility is listed with distance and estimated arrival time.
- The user can tap "Navigate" for a specific facility to receive turn-by-turn directions.
- Alternatively, the user can simply review the options and remain at their current location.

4. Confirmation & Feedback:

- Upon selection and processing, appropriate confirmation messages are displayed (e.g., "Navigation Started" or "Nearby medical facilities updated").
- If the user is offline, actions are queued and synchronization is attempted once connectivity is restored, with clear notifications.

5. Additional Considerations ☰

- **Urgency & Clarity:**

Ensure that the modals and overlays use clear, high-contrast colors (with red accents for urgency) so users immediately understand the context.

- **Accessibility:**

All buttons and text elements must include accessible labels and meet contrast guidelines.

- **Offline Handling:**

The system should gracefully handle offline scenarios by queuing requests and providing fallback messaging.

- **User Confirmation:**

Include confirmation messages and the option to review the selected medical facility before finalizing navigation, ensuring that users are making informed choices.

6. Conclusion ☰

This updated wireframe integrates a post-submission flow for the "Request Medical" feature. Users now have the option to either "Take Me To Medical" with immediate embedded navigation or "Stay Put" to view nearby medical facilities with navigation details. This design ensures that users can quickly and clearly act on their medical needs in an emergency while receiving relevant route and facility information.

PAGE: Tickets

Last Updated: [@Robert Eriksen](#) Feb 11, 2025

1. Overview

The Global Tickets page provides users with a complete view of all tickets associated with their account. This includes:

- A list of tickets showing details such as Ticket ID, Ticket Type, and Validation Status.
- An option to transfer a ticket to another email address.

• New Section: Transfer Requests

- Displays any pending or completed ticket transfers (both incoming and outgoing).
- Offers interactive buttons to accept or cancel a pending transfer.

This page ensures that users can easily manage their tickets and transfer requests from one unified location.

2. Key UI Components

A. Page Header

- **Title:** "My Tickets"
- **Subheader:** Brief description such as "View and manage your tickets and transfer requests."

B. Tickets List Section

- **Ticket Cards/Rows:** Each ticket card includes:
 - **Ticket Identifier/Code**
 - **Ticket Type:** e.g., ga, vip, rv, or need-ticket.
 - **Validation Status:** e.g., Valid, Redeemed, or Pending.
 - **Associated Event:** (if applicable)
 - **Action Button:** "Transfer Ticket" to initiate a ticket transfer.

C. Transfer Ticket Modal/Dialog (Triggered via "Transfer Ticket" button)

- **Modal Header:** "Transfer Ticket"
- **Content:**
 - **Ticket Details:** Read-only summary (Ticket ID, Type, Status).
 - **New Email Address Field:** Input field for the recipient's email.
- **Actions:**
 - **Transfer:** Button to submit the transfer.
 - **Cancel:** Button to close the modal without action.

D. Transfer Requests Section

- **Purpose:** Display all ticket transfer requests involving the user.
- **Layout Elements:**
 - **Transfer Request List/Table:** Each row or card displays:
 - **Transfer ID / Ticket ID**

- **Direction:** Indicate whether the request is incoming (a ticket transfer to this account) or outgoing (a transfer request initiated by this account).
- **Ticket Details:** Ticket Type, Validation Status, and Associated Event (if applicable).
- **Status:** Whether the transfer is Pending, Accepted, or Completed.
- **Action Buttons (for Pending Requests):**
 - **Accept:** For incoming pending transfers.
 - **Cancel:** For outgoing pending transfers, or cancel an incoming request if desired.
- **No Transfer Requests Message:**
 - If no transfers exist, display a friendly message such as "No transfer requests at this time."

E. Additional Elements ↗

- **Pagination/Infinite Scrolling:**
 - If there are many tickets or transfer requests, use pagination or infinite scrolling.
- **Consistent Styling:**
 - Use the same card/list style for both the Tickets and Transfer Requests sections for consistency.

3. Wireframe Diagram (ASCII) ↗

Global Tickets Page (Combined View) ↗

```

1  +-----+
2  | [Big Fam Logo]      [🔍 Search]      [Profile Icon / LOGIN] |
3  +-----+
4  |                   My Tickets           |
5  | "View and manage your tickets and transfer requests." |
6  +-----+
7  | [Ticket Card / Row]                |
8  | -----|
9  | Ticket ID: QR123ABC | Ticket Type: VIP | Status: Valid   |
10 | Event: DJ Nightwave on Stage A - 7:30 PM |
11 | [Transfer Ticket]                |
12 | -----|
13 | [Ticket Card / Row]                |
14 | -----|
15 | Ticket ID: QR456DEF | Ticket Type: GA | Status: Redeemed |
16 | Event: Indie Sparks on Stage B - 9:00 PM |
17 | [Transfer Ticket]                |
18 | -----|
19 |             Transfer Requests Section |
20 | -----|
21 | [Transfer Request Row]            |
22 | -----|
23 | Transfer ID: TR789XYZ | Direction: Incoming | Ticket: VIP | Status: Pending |
24 | From: friend@example.com        |
25 | [Accept] [Cancel]               |
26 | -----|
27 | [Transfer Request Row]            |
28 | -----|
29 | Transfer ID: TR012ABC | Direction: Outgoing | Ticket: GA | Status: Pending |
30 | To: recipient@example.com       |
31 | [Cancel]                         |
32 | -----|
33 | [Load More Tickets / Transfer Requests] |
34 +-----+

```

Transfer Ticket Modal (When "Transfer Ticket" is clicked)

```

1 +-----+
2 |           Transfer Ticket           | <-- Modal Header
3 +-----+
4 | Ticket ID: QR123ABC | Ticket Type: VIP | Status: Valid | <-- Read-only summary
5 +-----+
6 | Enter recipient's email: [_____] | <-- Input field
7 +-----+
8 | [Transfer]          [Cancel]       | <-- Action buttons
9 +-----+
10

```

4. Interaction & User Flow ⚡

1. Viewing Tickets:

- The user navigates to the Global Tickets page.
- The page displays a list of tickets along with their validation statuses.
- Each ticket card includes a "Transfer Ticket" button to initiate a ticket transfer.

2. Transferring a Ticket:

- Upon clicking "Transfer Ticket," the user sees a modal prompting them to enter the recipient's email.
- The user submits the form; the system processes the transfer.
- The ticket transfer request is recorded, and its status appears in the Transfer Requests section.

3. Viewing Transfer Requests:

- The Transfer Requests section lists all pending and completed transfers to/from the user's account.
- For incoming pending transfers, the user sees an "Accept" button.
- For outgoing pending transfers, the user sees a "Cancel" button.
- Once a transfer is accepted or completed, the status updates accordingly.

4. Managing Transfer Requests:

- The user can accept an incoming transfer to add the ticket to their account.
- Alternatively, they can cancel a pending transfer request if no longer desired.
- Feedback is provided for each action (e.g., confirmation messages).

5. Additional Considerations ⚡

• Responsive Design:

Ensure the layout adjusts gracefully for mobile, tablet, and desktop devices.

• User Feedback:

Provide clear visual feedback for successful transfers, errors, and status updates.

• Security:

The transfer functionality should verify the user's identity before processing any transfer requests.

• Offline Support:

Ticket information is cached locally, and transfer actions initiated offline are queued for sync when connectivity is restored.

6. Conclusion

The updated Global Tickets page now includes a new Transfer Requests section. This section displays all pending and completed ticket transfers and allows users to accept or cancel pending transfers. This design ensures users have a comprehensive view of their tickets and related transfer activities, all within an intuitive and responsive interface.

DECISIONS

DECISION TEMPLATE

Status	NOT STARTED / IN PROGRESS / COMPLETE
Impact	HIGH / MEDIUM / LOW
Driver	
Approver	
Contributors	
Informed	
Due date	
Resources	

Background ☕

Relevant data ☕

Options considered ☕

	Option 1	Option 2
Description		
Pros and cons	⊕ ⊖	⊕ ⊖
Estimated cost	LOW / MEDIUM / LARGE	LOW / MEDIUM / LARGE

Action items ☕

Outcome ☕

✓ DECISION 1: Cloud Provider Selection

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	@Robert Eriksen
Due date	
Resources	

📘 Background 🌐

Big Fam Festival requires a **highly scalable and resilient** cloud infrastructure capable of handling **4000+ concurrent users** with real-time updates, offline access, and emergency notifications.

📊 Relevant data 🌐

- Past festival user load:** Peak of 3500+ concurrent users in 2024.
- Expected Growth:** 15% increase in user base for 2025.
- Team Expertise:** Mixed experience with AWS and GCP.
- Budget Constraints:** Need cost-effective scaling. **AWS (Kubernetes - EKS)**

🌈 Options considered 🌐

Option 1: AWS (Kubernetes - EKS)		Option 2: GCP (Cloud Run - Serverless)
Description		
Description <ul style="list-style-type: none"> Uses Elastic Beanstalk, EKS (Kubernetes) for backend services. PostgreSQL (RDS), Redis (ElastiCache), CloudFront (CDN) for caching and content delivery. Full control over infrastructure, manual scaling possible. 	<ul style="list-style-type: none"> High customizability & strong ecosystem. Best for persistent services (long-running backend processes). 	<ul style="list-style-type: none"> Uses Cloud Run (Serverless) for backend microservices. Cloud SQL (PostgreSQL), Memorystore (Redis), Cloud CDN for content caching. Fully managed, auto-scaling infrastructure with a pay-as-you-go model.
Pros and cons		<ul style="list-style-type: none"> No server management required, reducing DevOps overhead. Auto-scales seamlessly to handle traffic spikes.

	<ul style="list-style-type: none"> • Industry-leading security & compliance. <p></p> <ul style="list-style-type: none"> • Requires more setup & management. • Higher baseline cost due to always-running infrastructure. 	<ul style="list-style-type: none"> • Lower operational costs as unused instances scale to zero. <p></p> <ul style="list-style-type: none"> • Cold starts may cause slight delays when traffic surges. • Less fine-grained control over infrastructure
Estimated cost	MEDIUM / LARGE	LOW / MEDIUM

✓ Action items ☕

- Proceed with provisioning GCP Cloud Run infrastructure.
- Set up Cloud SQL, Memcache, and Cloud CDN for backend services.
- Begin backend and API development on GCP platform.

★ Outcome ☕

GCP (Cloud Run) selected for its **cost-effectiveness, ease of setup, and auto-scaling capabilities.**

✓ DECISION 2: Backend Framework Selection

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	@Robert Eriksen
Due date	
Resources	<ul style="list-style-type: none"> [Link to API architecture diagrams] [Comparison of backend frameworks] [Performance benchmarks]

📘 Background ☕

The backend framework will define the core API structure for the **Big Fam Festival App**. It needs to be **fast, scalable, and compatible** with **Google Cloud Run** while supporting **real-time updates, offline access, and event-driven workflows**.

📊 Relevant data ☕

- Cloud Provider:** GCP (Cloud Run) selected.
- Performance Needs:** Handle **4000+ concurrent users** with efficient database queries.
- Team Expertise:** Experience with both TypeScript (NestJS) and Python (FastAPI).
- Development Speed:** Need for rapid iteration and developer-friendly environment.

🌈 Options considered ☕

	Option 1: NestJS (TypeScript - Node.js)	Option 2: FastAPI (Python - ASGI)
Description	<ul style="list-style-type: none"> Uses TypeScript, built on Node.js, and follows an opinionated modular architecture. Supports GraphQL (Apollo Federation) and REST APIs. Native WebSockets support for real-time notifications. 	<ul style="list-style-type: none"> Uses Python and ASGI, designed for high-performance API development. Async-first framework that supports WebSockets and background tasks. Strong compatibility with machine learning models for future AI-driven features.
Pros and cons	<p>+</p> <ul style="list-style-type: none"> Highly structured & scalable, great for large teams. TypeScript ensures strong typing and maintainability. 	<p>+</p> <ul style="list-style-type: none"> Blazing fast and lightweight (async-based execution). Python is widely used in AI & data science, making it easier to integrate ML features.

	<ul style="list-style-type: none"> Large ecosystem with many pre-built integrations. Better support for event-driven workflows with Pub/Sub in GCP. <p>-</p> <ul style="list-style-type: none"> Slightly higher memory usage compared to FastAPI. More boilerplate code can slow down early development. 	<ul style="list-style-type: none"> Less boilerplate code, faster development. Lower memory usage than Node.js, optimized for Cloud Run auto-scaling. <p>-</p> <ul style="list-style-type: none"> Smaller ecosystem compared to Node.js. Limited TypeScript-like static typing, which can lead to runtime issues. WebSocket support requires additional configurations.
Estimated cost	MEDIUM	LOW / MEDIUM

✓ Action items

- Proceed with NestJS development.
- Set up project structure using NestJS on GCP Cloud Run.
- Implement API Gateway and real-time features using WebSockets.

★ Outcome

NestJS selected for its **structured approach, scalability, TypeScript support, and ecosystem consistency with frontend development**.

✓ DECISION 3: Frontend Framework Selection

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	
Due date	
Resources	<ul style="list-style-type: none"> [Link to UI/UX wireframes] [Performance benchmarks for mobile frameworks] [Comparison of React Native vs. Flutter]

📘 Background ☕

The frontend framework will define the user experience of the **Big Fam Festival App**. It needs to be **cross-platform**, **efficient**, and **capable of offline functionality** while ensuring seamless integration with the NestJS backend on **Google Cloud Run**.

📊 Relevant data ☕

- Backend Framework:** NestJS (TypeScript) selected.
- Performance Needs:** Smooth UI, fast load times, and ability to handle offline features.
- Team Expertise:** Experience with JavaScript (React Native) and some exposure to Dart (Flutter).
- Development Speed:** Need for rapid UI iteration and reusable components.

🌈 Options considered ☕

	Option 1: React Native (JavaScript - TypeScript) - ✓ Selected	Option 2: Flutter (Dart - Skia Engine) (Not Selected)
Description	<ul style="list-style-type: none"> Uses JavaScript/TypeScript, providing seamless integration with the existing NestJS backend. Large ecosystem with Expo CLI for fast development and React Native CLI for customization. Strong support for WebSockets, Offline Storage, and Push Notifications. 	<ul style="list-style-type: none"> Uses Dart with Skia rendering engine, providing near-native performance. Pre-built UI components for a highly polished user interface. Great for animations and high-performance mobile applications.

Pros and cons	+	<ul style="list-style-type: none"> • Unified stack with NestJS, ensuring smoother backend-frontend integration. • Massive community support and mature ecosystem. • Faster development cycle due to reusable web-like components. • Performance optimizations available with Hermes engine. 	-	<ul style="list-style-type: none"> • Superior performance, especially for animations and graphics. • Single codebase truly optimized for both iOS and Android. • Well-suited for visually rich applications.
		<ul style="list-style-type: none"> • Native performance is slightly lower than Flutter. • Requires additional tuning for smooth animations. 		<ul style="list-style-type: none"> • Dart learning curve for developers already familiar with JavaScript. • Less mature ecosystem compared to React Native. • Larger app size due to Skia rendering engine
Estimated cost	MEDIUM		MEDIUM / LARGE	

✓ Action items ↗

- Proceed with React Native development.
- Set up the project with Expo for faster iteration.
- Implement UI wireframes and offline storage capabilities.

★ Outcome ↗

React Native selected for its **compatibility with the existing JavaScript stack, fast iteration speed, and strong ecosystem support**.

✓ DECISION 4: Database Selection

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	@Robert Eriksen
Due date	
Resources	<ul style="list-style-type: none"> • [Link to database schema designs] • [Comparison of relational vs. NoSQL databases] • [Performance benchmarks for PostgreSQL and Firestore]

📘 Background ☕

The database selection is critical for ensuring **fast queries, scalability, and efficient data storage** in the **Big Fam Festival App**. The database needs to support **user schedules, ticketing, real-time updates, and offline synchronization** efficiently while integrating with **Google Cloud Run**.

📑 Relevant data ☕

- **Cloud Provider:** GCP (Cloud Run) selected.
- **Backend Framework:** NestJS selected.
- **Data Needs:** Structured relational data for tickets, users, and event schedules; potential for real-time and document-based data.
- **Performance Needs:** Efficient reads/writes under **4000+ concurrent users**.
- **Scalability Requirements:** Ability to handle festival peaks without bottlenecks.

🌈 Options considered ☕

	PostgreSQL (Relational - SQL) - (Not Selected)	Firestore (NoSQL - Document Database) - ✓ Selected
Description	<ul style="list-style-type: none"> • A structured relational database that ensures consistency, reliability, and complex querying capabilities. • Deployed via Cloud SQL on GCP, allowing managed database services with auto-scaling. • ACID-compliant transactions ensure data integrity across multiple festival operations. 	<ul style="list-style-type: none"> • A serverless NoSQL document database optimized for real-time syncing and high-speed lookups. • Integrates seamlessly with Firebase services and Cloud Run. • Ideal for real-time chat, notifications, and user-generated content.

Pros and cons	<p>+</p> <ul style="list-style-type: none"> • Well-suited for structured data like ticketing, user management, and schedules. • Supports complex queries for analytics and reporting. • Mature ecosystem with strong community support. • Easier integration with GraphQL & REST APIs. <p>-</p> <ul style="list-style-type: none"> • Higher write latency compared to NoSQL. • Less optimized for real-time updates and high-speed document storage. 	<p>+</p> <ul style="list-style-type: none"> • Real-time data syncing is built-in. • Auto-scaling and fully serverless (no management overhead). • Optimized for frequent reads/writes at high scale. • Native offline support for mobile users. • Cost-effective pay-as-you-go pricing (cheaper at low usage, scales well with demand). <p>-</p> <ul style="list-style-type: none"> • Not well-suited for structured relational data. • Less powerful querying capabilities compared to SQL. • Limited transactional support across multiple documents.
Estimated cost	MEDIUM	LOW / MEDIUM

✓ Action items ☀

- Proceed with Firestore setup on GCP.
- Define Firestore document schema for users, events, schedules, and ticketing.
- Leverage Firestore's built-in real-time capabilities for user schedules and notifications.

★ Outcome ☀

- Firestore selected** for its **real-time data syncing, offline-first capabilities, and seamless auto-scaling on Google Cloud**.

✓ DECISION 5: Real-Time Notification

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	
Due date	
Resources	<ul style="list-style-type: none"> [Link to real-time architecture design] [Comparison of WebSockets vs. Firebase Notifications] [Performance benchmarks for push notifications]

📘 Background ☕

The real-time notification architecture is critical for keeping users updated on **event schedule changes, emergency alerts, and personal reminders**. The solution must integrate seamlessly with **Firebase (selected database)** and support **offline-first capabilities**.

📑 Relevant data ☕

- Cloud Provider:** GCP (Cloud Run) selected.
- Backend Framework:** NestJS selected.
- Database:** Firestore selected for real-time data sync.
- Performance Needs:** Low latency notifications for schedule changes and emergency alerts.
- Scalability Requirements:** Ability to handle festival-wide notifications for **4000+ concurrent users**.

🌈 Options considered ☕

	Option 1: WebSockets (NestJS + Socket.IO) (Not Selected)	Option 2: Firebase Cloud Messaging (FCM) - <input checked="" type="checkbox"/> Selected
Description	<ul style="list-style-type: none"> Uses WebSockets to create a persistent connection between the client and server. Allows real-time bi-directional communication for instant updates. Requires users to be online to receive notifications. 	<ul style="list-style-type: none"> Uses Firebase Cloud Messaging (FCM) to deliver push notifications to users' devices. Works even when users are offline. Integrates natively with Firestore and React Native.
Pros and cons	<p>➕</p> <ul style="list-style-type: none"> Ultra-low latency, ideal for real-time features. Fully controlled within our backend, no reliance on third-party services. Efficient for active users who are online. 	<p>➕</p> <ul style="list-style-type: none"> Supports offline notifications, ensuring users never miss an update. Handles large-scale notifications efficiently with Firebase infrastructure.

	<ul style="list-style-type: none"> • No offline support, users will miss messages when disconnected. • More infrastructure overhead, requiring dedicated WebSocket management. • Scaling WebSockets for thousands of concurrent users requires additional configuration (e.g., Redis Pub/Sub). 	<ul style="list-style-type: none"> • Low operational overhead, no need to manage WebSocket connections. • Built-in analytics and segmentation for better targeting.
Estimated cost	MEDIUM / LARGE	LOW / MEDIUM

✓ Action items ↗

- Proceed with Firebase Cloud Messaging (FCM) integration.
- Configure notification types for event reminders, alerts, and schedule changes.
- Ensure push notifications work with offline mode for Firestore users.

★ Outcome ↗

Firebase Cloud Messaging (FCM) selected for its **offline support, scalability, and seamless Firestore integration**.

✓ DECISION 6: CI/CD and DevOps Pipeline

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	
Due date	
Resources	<ul style="list-style-type: none">[Link to CI/CD pipeline architecture][Comparison of GitHub Actions vs. GitLab CI/CD][Performance benchmarks for automated deployment systems]

📘 Background ☕

The CI/CD and DevOps pipeline is critical for automating **builds, testing, and deployments** in the **Big Fam Festival App**. The solution must integrate seamlessly with **Google Cloud Run, Firestore**, and **NestJS** while supporting **automated testing, security scanning, and rollback mechanisms**.

📑 Relevant data ☕

- Cloud Provider:** GCP (Cloud Run) selected.
- Backend Framework:** NestJS selected.
- Database:** Firestore selected for real-time data sync.
- Notification System:** Firebase Cloud Messaging selected.
- Scalability Requirements:** Seamless automated deployment with rollback capabilities.
- Team Workflow:** Developers need easy branch-based CI/CD deployment with staging and production environments.

🌈 Options considered ☕

	Option 1: GitHub Actions - <input checked="" type="checkbox"/> Selected	Option 2: GitLab CI/CD (Not Selected)
--	--	---------------------------------------

Description	<ul style="list-style-type: none"> Uses GitHub Actions to automate CI/CD workflows, directly integrated with GitHub repositories. Supports multi-stage workflows for testing, security scans, and deployments to Cloud Run. Native integration with Google Cloud via GitHub Actions Marketplace. 	<ul style="list-style-type: none"> Uses GitLab's integrated CI/CD pipelines to automate deployments with advanced configuration options. Provides built-in container registry and detailed pipeline visualization. Offers advanced security scanning features.
Pros and cons	<p>⊕</p> <ul style="list-style-type: none"> Seamless GitHub repository integration. First-party support for Google Cloud deployments. Parallelized testing and deployment workflows. Highly customizable workflow automation. <p>⊖</p> <ul style="list-style-type: none"> Can become complex for multi-cloud setups. Limited UI compared to GitLab CI/CD. 	<p>⊕</p> <ul style="list-style-type: none"> More advanced visualization and pipeline monitoring. Integrated container registry (reducing dependencies on external services). More flexibility for multi-cloud deployments. <p>⊖</p> <ul style="list-style-type: none"> More complex setup and learning curve. Requires additional integration with GitHub repositories. Higher cost for advanced features.
Estimated cost	LOW / MEDIUM	MEDIUM / LARGE

✓ Action items ☀

- Proceed with GitHub Actions for CI/CD automation.
- Set up automated deployment pipelines for staging and production.
- Implement automated testing, security scanning, and rollback mechanisms.

★ Outcome ☀

- GitHub Actions selected for its **seamless GitHub integration, scalability, and direct support for Google Cloud deployments**.

✓ DECISION 7: Embedded Navigation SDK Selection

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	
Due date	
Resources	<p>↳ Re-evaluating In-App Navigation vs. External Mapping Solutions</p> <ul style="list-style-type: none"> • [Link to SDK documentation] • [Comparison of Mapbox, HERE, and What3Words navigation solutions] • [Performance benchmarks for offline routing]

📘 Background ☕

We require an **offline-capable, in-app navigation solution** that delivers **full turn-by-turn directions** without requiring users to exit the app. Additionally, we see value in integrating **What3Words for campsite marking**, allowing users to **pin and share exact locations** within the festival grounds.

📑 Relevant data ☕

- **Cloud Provider:** GCP (Cloud Run) selected.
- **Backend Framework:** NestJS selected.
- **Database:** Firestore selected for real-time data sync.
- **Navigation Needs:** Offline routing, venue-specific turn-by-turn guidance, and user-friendly integration.
- **Scalability Requirements:** Efficient handling of in-app location services for thousands of concurrent users.
- **Enhanced Features:** Ability for users to mark and share campsite locations using **What3Words**.

🌈 Options considered ☕

	Option 1: Hybrid Approach (Mapbox or HERE + What3Words) - <input checked="" type="checkbox"/> Selected	Option 2: Mapbox Navigation SDK (Standalone)	Option 3: HERE Navigation SDK (Standalone)	Option 4: What3Words Only (Not Selected)
Description	<ul style="list-style-type: none"> • Mapbox or HERE will be used for turn-by-turn navigation inside the app. 	<ul style="list-style-type: none"> • A leading navigation SDK offering robust turn-by-turn guidance and excellent offline caching support. 	<ul style="list-style-type: none"> • Offers highly accurate turn-by-turn navigation and strong offline capabilities. 	<ul style="list-style-type: none"> • Uses What3Words to allow users to share their exact location using a 3-word address.

	<ul style="list-style-type: none"> • What3Words will be used for pinning and sharing precise campsite locations. • Users can save and share their What3Words locations for easy navigation back to their campsite. 	<ul style="list-style-type: none"> • Provides rich developer documentation and pre-built UI components for faster integration. 	<ul style="list-style-type: none"> • Competitive pricing model with flexible options for large-scale use. 	<ul style="list-style-type: none"> • Simplifies navigation for finding campsites and meeting points.
Pros and cons	<p>+</p> <ul style="list-style-type: none"> • Best of both worlds: Full offline navigation + precise location pinning. • What3Words simplifies campsite marking and location sharing. • Ensures seamless in-app navigation without switching to external maps. <p>-</p> <ul style="list-style-type: none"> • Requires dual integration effort (Mapbox/HERE + What3Words). • What3Words does not provide full routing—used only for marking locations. 	<p>+</p> <ul style="list-style-type: none"> • High-quality offline map caching and routing. • Extensive community support and documentation. • Highly customizable UI and map styling. <p>-</p> <ul style="list-style-type: none"> • Cost-intensive at scale; pricing model needs evaluation. • Does not support precise location marking like What3Words. 	<p>+</p> <ul style="list-style-type: none"> • Strong offline routing and accuracy. • Competitive pricing for high-volume scenarios. • Reliable navigation performance in low-connectivity environments. <p>-</p> <ul style="list-style-type: none"> • Less widespread usage in our context. • Fewer integration examples compared to Mapbox. • No native What3Words support for campsite marking. 	<p>+</p> <ul style="list-style-type: none"> • Highly accurate and intuitive location system. • Offline-friendly and easy to communicate precise locations. • Perfect for campsite marking and festival navigation. <p>-</p> <ul style="list-style-type: none"> • Not a full-fledged navigation SDK (no turn-by-turn routing). • Requires additional integration with a mapping service for directions.
Estimated cost	MEDIUM	MEDIUM / LARGE	MEDIUM	LOW / MEDIUM

✓ Action items ☀

- Proceed with a hybrid approach: Use Mapbox or HERE for navigation and What3Words for campsite, festival, vendor etc. marking.
- Develop a POC to compare Mapbox vs. HERE for offline routing.
- Integrate What3Words for campsite location sharing.
- Finalize the selected navigation SDK based on offline performance and ease of integration.

★ Outcome ☀

- Hybrid approach selected: Mapbox or HERE for turn-by-turn navigation + What3Words for campsite marking and location sharing.

Re-evaluating In-App Navigation vs. External Mapping Solutions

Now that we are deep into the decision-making process, let's step back and ask:

Do we need in-app navigation, or should we let users use their preferred mapping apps?

Key Considerations ☀

1 Offline Functionality ☀

✓ In-App Navigation (Mapbox, HERE, What3Words)

- **Supports offline routing** (if preloaded).
- Users can still get directions **without mobile data**.

✗ External Mapping Apps (Google Maps, Apple Maps, Waze, etc.)

- **Google Maps** allows offline maps but requires **manual downloads**.
 - **Apple Maps & Waze** do not have full offline navigation.
 - Users may **lose connectivity in remote festival areas** (affecting GPS-based routing).
-

2 User Experience & Integration ☀

✓ In-App Navigation

- Provides **turn-by-turn directions inside the festival app**—no need to switch apps.
- Can integrate **festival-specific routes** (e.g., stage entrances, vendor areas).
- Supports **custom markers & safety routes** for emergency planning.

✗ External Mapping Apps

- Users **must manually switch apps** to get directions.
 - **No direct integration** with festival updates or venue-specific routes.
 - Can be **confusing if different apps have different paths** or missing walkable routes.
-

3 Cost & Development Effort ☀

✓ External Mapping Apps

- **Zero development effort**—just provide venue links in Google Maps/Apple Maps.
- No SDK licensing fees.

✗ In-App Navigation

- Requires **SDK licensing** (Mapbox, HERE) → **Potential cost at scale**.
 - More **engineering time** for integration and testing.
-

🚀 Recommendation: Hybrid Approach? ☀

Option 1: Keep In-App Navigation (If Offline is a Priority) ☀

- Use **Mapbox or HERE for full offline functionality** inside the app.
- Ensure maps & routes are **preloaded** for festival grounds.

- Best for **providing emergency routing, custom paths, and full venue navigation.**

Option 2: Use External Apps for Simplicity (Cheaper & Faster) ↗

- Drop in-app navigation and instead **link users to Google Maps, Apple Maps, or What3Words for directions.**
- Warn users to **download offline maps in advance.**
- Best for **reducing costs & development complexity.**

✓ DECISION 8: File Upload & Storage Service

Status	COMPLETE
Impact	HIGH
Driver	@Robert Eriksen
Approver	@Robert Eriksen
Contributors	@Robert Eriksen
Informed	
Due date	
Resources	<ul style="list-style-type: none"> [Link to cloud storage pricing comparison] [Security and access control documentation] [Performance benchmarks for file uploads]

📘 Background ☕

We need to manage **file uploads for profile pictures and vendor menus** securely and efficiently. The storage service must provide **scalability, security, and seamless integration** with our cloud provider.

📑 Relevant data ☕

- Cloud Provider:** GCP (Cloud Run) selected.
- Backend Framework:** NestJS selected.
- Database:** Firestore selected for real-time data sync.
- File Types:** User profile images, vendor menu PDFs/images.
- Security Requirements:** Access control for vendors and users, encryption at rest.
- Scalability Needs:** Must handle thousands of file uploads efficiently.

🌈 Options considered ☕

	Option 1: AWS S3 (Not Selected)	Option 2: Google Cloud Storage - ✓ Selected	Option 3: Azure Blob Storage (Not Selected)
Description	<ul style="list-style-type: none"> Industry-leading object storage with seamless AWS integration. Provides fine-grained access control, versioning, and scalable storage. 	<ul style="list-style-type: none"> Fully managed object storage service integrated into Google Cloud. Provides multi-regional storage, strong security, and scalability. Supports Firestore and Firebase for seamless data retrieval. 	<ul style="list-style-type: none"> Cloud-based object storage with secure file management. Less commonly used in GCP or AWS-based projects.

Pros and cons	+ <ul style="list-style-type: none">• Highly reliable and widely used.• Strong security and compliance features.• Supports lifecycle management and CDN integration. - <ul style="list-style-type: none">• Not aligned with our GCP cloud provider choice.• Pricing can vary significantly based on usage.	+ <ul style="list-style-type: none">• Aligns with our existing GCP infrastructure.• Competitive pricing and auto-scaling capabilities.• Strong integration with Firestore and Firebase Authentication. - <ul style="list-style-type: none">• Similar functionality to AWS S3, but limited multi-cloud flexibility.• Requires careful configuration for role-based access control.	+ <ul style="list-style-type: none">• Reliable storage solution with high availability.• Good integration with Microsoft-based cloud ecosystems. - <ul style="list-style-type: none">• Not aligned with our current focus (AWS vs. GCP).• Less relevant given our selection of GCP services.
Estimated cost	MEDIUM	LOW / MEDIUM	MEDIUM

✓ Action items ☀

- Proceed with Google Cloud Storage as our primary file storage solution.
- Configure storage buckets for user profiles and vendor menus.
- Implement role-based access control and encryption policies.
- Perform cost comparisons based on projected usage.

★ Outcome ☀

✓ Google Cloud Storage selected for its alignment with GCP, cost efficiency, and seamless integration with our backend services.