GigsTM - Project Workflow

This document outlines the step-by-step workflow for developing the GigsTM platform, a gig-based/task-based marketplace. The workflow is derived from the detailed analysis provided in the project's Jupyter notebooks and is tailored for a tech stack of **React.js** for the frontend and **Supabase** for the backend.

Phase 0: Project Setup

1. Frontend Setup (React.js)

- **Initialize React App:** Use `create-react-app` or Vite to set up the basic React project structure.
- **Install Core Libraries:**
  - `react-router-dom` for navigation.
  - `axios` or `fetch` for API calls.
  - A UI library like `Material-UI` or `Ant Design` for a consistent design system.
  - A state management library like `Redux Toolkit` or `Zustand`.
- **Project Structure:** Organize the project into folders for `components`, `pages`, `services`, `utils`, `hooks`, and `styles`.

2. Backend Setup (Supabase)

- **Create a New Supabase Project:** Set up a new project in the Supabase dashboard.
- **Database Schema:** Use the Supabase SQL editor to create the tables based on the modular architecture from the notebooks. The schema is detailed below.
- **Enable Authentication:** Configure Supabase Auth to handle user sign-up, sign-in, and role-based access control.
- **Set up Storage:** Configure Supabase Storage for handling file uploads (e.g., KYC documents, task proofs).

3. Database Schema (Supabase - PostgreSQL)

The following tables should be created in Supabase, based on the modular breakdown in the notebooks.

Module 1: User & Worker Management - `users` - `user_profiles` - `kyc_documents` - `roles_permissions` - `user_sessions` - `user_activity_logs` (can be a table with JSONB columns) - `referrals` - `user_experiences`

Module 2: Gig Listing & Discovery - `gigs` - `gig_steps` - `gig_categories` - `gig_qualifications` - `gig_tags` - `gig_reviews` (can be a table or a JSONB column in `gigs`) - `gig_media` - `gig_bookmarks`

Module 3: Application & Assignment - `applications` - `mcq_questions` - `mcq_results` - `trainings` - `training_modules` - `work_orders` - `application_feedback`

Module 4: Task Submission & Verification - `claims` - `claim_verifications` - `disputes` - `claim_history` - `claim_media`

Module 5: Payment & Rewards - `transactions` - `wallets` - `withdraw_requests` - `finance_audit_logs` - `reconciliations` - `tax_deductions` - `bonus_rewards` - `invoice_records` - `payout_batches`

Module 8: Communication & Support - `messages` - `support_tickets` - `announcements` - `faq_articles`

(Note: The other modules, 6 and 7, are more about backend logic and aggregation, and might not have dedicated tables initially, but will use the existing ones.)

Phase 1: Implementing Core Modules (MVP)

Module 1: User & Worker Management

- **User Authentication:** Implement sign-up and sign-in forms that interact with Supabase Auth.
- **Profile Management:** Create pages for users to view and edit their profiles (`user_profiles` and `user_experiences` tables).
- **KYC Submission:** Build a form for users to upload their KYC documents (`kyc_documents` table and Supabase Storage).
- **Role-Based Access Control (RBAC):** Set up roles in Supabase and protect routes in the React app based on user roles.

Module 2: Gig Listing & Discovery

- **Gig Creation (for Clients/Admins):** A form for clients or admins to create new gigs (`gigs` and `gig_steps` tables).
- **Gig Discovery (for Workers):** A main page that lists all active gigs. Implement search and filtering based on categories, location, and pay.
- **Gig Detail Page:** A page that shows the full details of a gig, including its steps, requirements, and pay structure.

Module 3: Application & Assignment

- **Apply to Gig:** An "Apply" button on the gig detail page that creates an entry in the `applications` table.
- **MCQ Test:** If a gig has a qualification test, present the MCQ test to the user.
- **Application Tracking:** A dashboard for workers to see the status of their applications.
- **Manager/Admin View:** A dashboard for managers to review applications, see test scores, and assign trainings.

Module 4: Task Submission & Verification

- **Task Submission:** A form for workers to submit proof of task completion for each step of a gig (`claims` and `claim_media` tables).
- **Claim Verification (for Managers):** A dashboard for managers to review submitted claims, approve or reject them, and add remarks.
- **Dispute Resolution:** A basic interface for workers to raise disputes on rejected claims (`disputes` table).

Module 5: Payment & Rewards

- **Wallet System:** A virtual wallet for each worker (`wallets` table).
- **Payout Generation:** When a claim is approved, automatically create a `transactions` entry and credit the worker's wallet.
- **Withdrawal Request:** A form for workers to request a withdrawal of their wallet balance (`withdraw_requests` table).
- **Finance Dashboard:** A view for the finance team to process withdrawal requests.

Phase 2: Implementing Supporting Modules

Module 6: Reports & Analytics

- **Admin Dashboard:** Create a dashboard for admins with key performance indicators (KPIs) like:
  - Total active users
  - Gigs created vs. completed
  - Total payouts
- **Client Reports:** Generate summary reports for clients on the performance of their gigs.
- **Worker Performance:** A section in the worker's dashboard showing their earnings, completed tasks, and ratings.

Module 7: System Administration, Security & Monitoring

- **Admin Panel:** An admin-only section to manage users, roles, and system configurations.
- **Security:**
  - Implement row-level security in Supabase to ensure users can only access their own data.
  - Add data validation and sanitization on both frontend and backend.
- **Monitoring:** Set up logging to monitor API calls, errors, and system health.

Module 8: Communication, Support & Engagement

- **Notifications:** A real-time notification system (using Supabase Realtime) for events like claim approval, new messages, etc.

- **In-App Messaging:** A simple chat interface for communication between workers and managers.
- **Support Tickets:** A helpdesk system for users to raise support tickets (`support_tickets` table).
- **FAQs:** A page with frequently asked questions to reduce support load.

Phase 3: Deployment & Scaling

1. Deployment

- **Frontend:** Deploy the React app to a platform like Vercel, Netlify, or AWS Amplify.
- **Backend:** Supabase is already cloud-based, so it's a matter of managing production environments.
- **CI/CD:** Set up a CI/CD pipeline (e.g., using GitHub Actions) to automate testing and deployment.

2. Scaling

- **Database Optimization:** Add indexes to frequently queried columns in Supabase to improve performance.
- **Caching:** Implement caching for frequently accessed data that doesn't change often.
- **Load Testing:** Use tools like k6 or JMeter to test the application's performance under load.
- **Scalable Architecture:** As the application grows, consider moving some of the backend logic from Supabase (if it becomes a bottleneck) to serverless functions or a dedicated backend server.