**SWIFTFACTURE.COM**

**Contents**

[**1.** **SwiftFacture – Project Overview** 2](#_Toc210047449)

[**2. SwiftFacture Free Trial & Billing Setup** 5](#_Toc210047450)

[**3. MailerSend Integration Requirements for SwiftFacture** 9](#_Toc210047451)

[**4.** **SwiftFacture System connection** 11](#_Toc210047452)

[**5.** **Auth & Organization Onboarding System** 14](#_Toc210047453)

[**6. SwiftFacture Admin Page: Features & Capabilities** 17](#_Toc210047454)

[**7. SwiftFacture Admin Page Layout** 20](#_Toc210047455)

[8. **i18n.js Language Translation Setup for SwiftFacture** 25](#_Toc210047456)

[**9. GitHub + Netlify Deployment Workflow** 28](#_Toc210047457)

[**10. Summary feature setup for SwiftFacture** 30](#_Toc210047458)

# **SwiftFacture – Project Overview**

**SwiftFacture** is a **web-based billing and invoicing platform** designed to help businesses create, manage, and track **invoices, receipts, estimates, and clients** efficiently. The system is built around **organization-centric profiles**, ensuring that company information is stored once and automatically applied across all financial documents, reducing repetitive data entry.

**Key Features**

1. **Organization & User Profiles**
   * Each user belongs to an **organization** (company), which stores key details like name, address, tax/VAT number, logo, and currency.
   * Multiple users can join an organization, with **role-based permissions**: Owner, Admin, Member.
   * Organization data auto-populates financial documents to streamline workflow.
2. **Invoices, Receipts, & Estimates**
   * Create professional invoices, receipts, and estimates with **customizable templates**.
   * Link documents to **clients**, users, and organizations.
   * PDF generation and email delivery included.
3. **Client Management**
   * Maintain a centralized **client database** per organization.
   * Track invoices, estimates, and payment history for each client.
4. **Trial & Subscription System**
   * 30-day **free trial** per organization without requiring a credit card.
   * Subscription plans managed per organization, affecting all members.
   * Integration-ready for **Stripe payments** for upgrades and premium features.
5. **Authentication & Access Control**
   * Built on **Supabase Auth** for secure email/password login.
   * Role-based access ensures users only interact with resources within their organization.
   * Pre-login users cannot access protected pages; onboarding guides users to create or join an organization.
6. **Admin Dashboard**
   * Centralized control for monitoring **users, organizations, subscriptions, invoices, clients, and support tickets**.
   * System analytics, activity logs, and role management available for admin users.
7. **Notifications & Emails**
   * Integration with **MailerSend** for transactional emails:
     + Trial lifecycle notifications
     + Invoice/receipt delivery
     + Password resets and verification
     + Team invites and support updates
8. **i18n Multilingual Support**
   * French as **default language**, with English as an option.
   * Translations applied across UI, notifications, and emails.
   * Language preference stored per user for consistent experience.
9. **Security & Monitoring**
   * Audit logs for all major actions (document creation, subscription changes).
   * Role-based row-level security for data access.
   * Optional 2FA and monitoring for suspicious activity.
10. **Scalable & Extensible Architecture**
    * Built with **Supabase backend**, supporting future integrations like Stripe, additional languages, or advanced analytics.
    * API-centric design allows frontend frameworks or AI tools (like Gemini) to extend features quickly.

**How It Works**

1. Users sign up → create or join an **organization**.
2. Organization details are saved and auto-applied to all documents.
3. Users create **invoices, receipts, estimates**, and manage **clients** without re-entering company information.
4. **Trial management, notifications, and subscriptions** are organization-wide.
5. Admin users monitor system-wide activity, manage subscriptions, and handle support tickets.
6. Emails, PDFs, and UI are localized using **i18n**, with French as the default.

**Project Goals**

* Simplify invoicing and billing for small to medium businesses.
* Minimize repetitive data entry through centralized organization profiles.
* Provide secure, role-based access and administration.
* Offer a scalable architecture for future integrations and growth.

✅ **Summary Statement:**  
SwiftFacture is an **all-in-one billing and invoicing platform** built around organizations, enabling users to create and manage financial documents efficiently, while providing secure access, trial/subscription management, multilingual support, and admin oversight — all designed for simplicity, scalability, and automation.

**2. SwiftFacture Free Trial & Billing Setup**

**Goal**

Implement a **30-day free trial (no card required upfront)** for SwiftFacture (billing/invoicing app). Use **Supabase** for authentication and database, manage trial logic internally, and integrate **Stripe** later for payments.

**1. Database Setup**

* Create tables in Supabase:
  + **organizations**: org metadata, owner.
  + **org\_members**: link users to organizations with roles.
  + **billing\_subscriptions**: track subscription status, trial dates, Stripe IDs.
  + **plans**: metadata for trial and paid plans.
  + **billing\_events**: record trial start, reminders, expiry, conversions.

**2. Security & RLS**

* Enable Row Level Security (RLS) on all billing-related tables.
* Policies:
  + Owners can insert their organizations.
  + Members can view their org and subscription data.
  + Writes (org creation, subscription creation) are done server-side using **service role key**.

**3. Signup Flow**

* After user signs up with Supabase Auth:
  + Create organization and link user as owner.
  + Add org\_member row for the owner.
  + Create billing\_subscriptions row:
    - plan\_id = trial-30
    - status = trialing
    - trial\_start = now()
    - trial\_end = now() + 30 days
  + Log trial\_started in billing\_events.

**4. Access Enforcement**

* Middleware/API checks:
  + Allow access if subscription is active.
  + Allow access if subscription is trialing and trial\_end not passed.
  + Deny/limit access if trial\_end has passed or status = expired.
* Block premium features after expiry and show upgrade CTA.

**5. Scheduled Jobs**

* Daily cron job or scheduled function to:
  + Find subscriptions where status = trialing and trial\_end < now().
  + Update status to expired.
  + Insert trial\_expired event into billing\_events.
  + Send corresponding notification emails.
* Additional cron logic for reminders (e.g., 7 days, 2 days, 1 day before expiry).

**6. Notifications**

* Use transactional email provider (SendGrid, Mailgun, Postmark).
* Email templates:
  + Welcome → “Your 30-day trial started, ends on X date.”
  + 7 days left → Reminder.
  + 2 days left → Urgent reminder.
  + 1 day left → Final reminder.
  + Trial expired → Notify and provide upgrade link.
* Track which reminders have been sent to avoid duplicates.

**7. Stripe Integration (Later Phase)**

* Add fields to billing\_subscriptions for stripe\_customer\_id and stripe\_subscription\_id.
* When upgrading:
  + Create Stripe Customer and Subscription.
  + Pass existing trial\_end to Stripe so trial periods match.
* Webhooks to implement:
  + invoice.payment\_succeeded → mark subscription active.
  + invoice.payment\_failed → mark subscription past\_due.
  + customer.subscription.deleted → mark subscription canceled.
* Upgrade page in UI with Stripe Checkout or Elements.

**8. UX Requirements**

* Account/Billing page: show current plan, trial days left, and upgrade options.
* Upgrade page: collect payment when trial ends.
* Expired trial screen: restrict functionality, show “Upgrade to continue.”
* Admin/internal dashboard: list active trials, expirations, and conversions.

**9. Testing**

* Staging environment in Supabase with test data.
* Test scenarios:
  + Signup → trial starts correctly.
  + Access allowed during trial.
  + Access blocked after expiry.
  + Reminder emails sent at correct times.
  + Expired trial flow works.
  + Stripe upgrade flow tested with test keys.
* Include unit tests for trial logic and e2e tests for the whole flow.

**10. Security & Config**

* Store environment secrets securely (service role key, Stripe key, email API key).
* Never expose SUPABASE\_SERVICE\_ROLE\_KEY to client.
* Use environment variables in server functions only.
* Log trial and subscription events for auditing.

**11. Metrics & Analytics**

* Track billing\_events:
  + trial\_started
  + reminder\_sent (7d, 2d, 1d)
  + trial\_expired
  + converted\_to\_paid
* Use events to measure trial-to-paid conversion rate.

**12. Data Retention & Policy**

* Decide what happens when trial expires:
  + Keep data in read-only mode for X days.
  + Delete data after Y days if no upgrade.
* Communicate retention policy in Terms & Privacy.

✅ Deliverable: A working **30-day trial system** with Supabase + email reminders, and a clear upgrade path to Stripe billing.

## **3. MailerSend Integration Requirements for SwiftFacture**

**1. Domain Setup**

* Verify sending domain (swiftfacture.com or chosen domain) in MailerSend.
* Configure SPF, DKIM, DMARC records in DNS.
* Use branded sender addresses: support@swiftfacture.com, billing@swiftfacture.com.

**2. User Authentication & Onboarding**

* Send **welcome email** after user signs up and account is created.
* Optional: replace Supabase default emails with MailerSend for:
  + **Email verification**
  + **Password reset**

**3. Trial Management (30 days free trial)**

* Send **trial start email** on sign-up: includes trial start and end date.
* Send **trial reminder emails**:
  + 7 days before expiry
  + 2 days before expiry
  + 1 day before expiry
* Send **trial expired email** when free plan ends.

**4. Billing & Payments (Stripe, later stage)**

* Send **payment success email** with receipt.
* Send **payment failure email** requesting updated payment method.
* Send **subscription canceled email** with confirmation details.

**5. Invoicing & Core Features**

* Allow users to **send invoices and estimates** to their clients via MailerSend:
  + Email includes PDF attachment of invoice/estimate.
  + Subject line includes invoice/estimate number.
* Send **invoice payment reminder emails** to clients.
* Send **confirmation emails to user** when invoice/estimate is sent successfully.

**6. Team & Account Notifications**

* Send **team invitation email** when owner invites a member to the workspace.
* Send **account activity alert emails** (login from new device, unusual activity).

**7. Support & Communication**

* Send **support ticket confirmation email** when a user submits a support request.
* Send **support response email** when a team member replies.

**8. MailerSend Features to Use**

* Use **transactional templates** for all emails with branding and dynamic placeholders (e.g., {{name}}, {{trial\_end}}, {{invoice\_number}}).
* Use **attachments** for invoices/receipts in PDF.
* Use **webhooks** to track delivery, opens, bounces.
* Use **analytics** to measure open/click rates on trial and invoice reminders.
* Support multiple sender domains if needed (e.g., support@swiftfacture.com, billing@swiftfacture.com).

## **SwiftFacture System connection**

**Task:**  
Set up a data and feature connection system in **SwiftFacture** so that every feature (invoicing, receipts, estimates, clients, billing, notifications, support, trial management) is connected to both the **user** and their **organization profile**.

**1. Core Structure**

* Create a **Users table** linked to Supabase Auth.
* Create an **Organizations table** storing company details.
* Ensure each user belongs to an organization (org\_id).
* Allow organizations to have multiple users (team members).

**2. Invoicing & Financial Records**

* Connect **Invoices, Receipts, Estimates** to both user\_id and org\_id.
* Auto-fill organization details (name, address, tax number, logo, currency) from the organizations table when creating invoices/receipts/estimates.
* Store which **user created** the document for audit purposes.

**3. Clients**

* Create a **Clients table** tied to organizations (org\_id).
* Ensure each org manages its own clients list.
* Connect invoices/estimates to clients via client\_id.

**4. Subscriptions & Premium Upgrades**

* Create a **Subscriptions table** tied to organizations (org\_id).
* Store subscription plan, status, trial start/end, and Stripe subscription ID (future).
* Premium upgrades apply to the entire organization, not just one user.
* All team members inherit the organization’s subscription level.

**5. Trials**

* On signup, assign a **30-day trial** to the organization.
* Connect trial lifecycle (start, reminders, expiration) to the organization’s subscription record.
* Link trial-related emails to both org and owner user.

**6. Notifications & Emails (via MailerSend)**

* All system emails should include organization context:
  + Welcome & trial start (with org details)
  + Trial reminders (7, 2, 1 day left)
  + Trial expired
  + Payment success/failure
  + Invoice/estimate sent (with org branding)
  + Team invites
  + Support ticket updates
* Use org\_id to pull company name/logo in emails.

**7. Support Tickets**

* Create a **SupportTickets table** tied to both user\_id and org\_id.
* Ensure support requests are connected to the organization profile.
* Use MailerSend to notify users and org admins about ticket updates.

**8. Security & Access Control**

* Users can only access invoices, receipts, estimates, clients, and subscriptions belonging to their own organization.
* Admin users can manage subscription, billing, and invite team members.
* Regular users can create invoices/estimates/receipts but not modify subscription.

**Deliverable:**  
Generate schema connections, API routes, and logic so that **all SwiftFacture features are tied to both the user and their organization profile**, ensuring no duplicate entry of company details, seamless premium upgrade handling, and consistent email/notification context.

## **Auth & Organization Onboarding System**

**Task:**  
Design and implement a **complete authentication and onboarding system** for SwiftFacture, including user login/signup, organization creation, trial setup, and access control.

**1. User Authentication**

* Use **Supabase Auth** (email/password, optionally social logins).
* Ensure users can **sign up**, **verify email**, **log in**, **reset password**, and **log out**.
* Store basic user info:
  + id (Supabase Auth UUID)
  + name
  + email
  + role (owner, admin, member)
  + org\_id (link to organization, null until org created)

**2. Pre-Login Restrictions**

* Users **cannot access** any protected routes: invoices, receipts, estimates, clients, billing, subscriptions, team management, or support.
* Public pages allowed:
  + Landing page / marketing info
  + Sign up / login forms
  + Password reset / email verification

**3. Post-Login Flow**

After logging in, users can:

1. **Create a new organization/company** (if they are the first user or owner).
2. **Join an existing organization** via invitation (if invited).
3. **Set up profile details** for their organization:
   * Organization name
   * Business type (LLC, Sole Trader, etc.)
   * Address
   * Tax/VAT number
   * Logo / branding
   * Default currency
   * Billing email
4. **Access trial or subscription info**:
   * Start 30-day trial automatically upon organization creation.
   * View current plan and trial end date.
5. **Access core features** linked to their organization:
   * Create invoices, receipts, estimates
   * Manage clients
   * Send emails/notifications
   * Manage team members

**4. Organization & User Roles**

* **Owner**: full control, can manage subscription, team members, org settings.
* **Admin**: can create/edit invoices/receipts/estimates, manage clients, access reporting.
* **Member**: can create/edit invoices/estimates/receipts but cannot manage subscription or invite users.
* Each organization can have multiple users with roles assigned.

**5. Security & Access Control**

* Users **cannot access any organization data** until they are logged in and either:
  + Created a new organization (owner)
  + Joined an existing organization (member/admin)
* Implement middleware/API checks to ensure **user’s org\_id matches the requested resource**.
* Only owners can access subscription/payment features.

**6. Trial & Subscription Integration**

* On org creation, automatically create **billing\_subscriptions** with a 30-day trial.
* Track trial start/end dates.
* Connect trial status to features and MailerSend notifications.

**7. Optional Enhancements**

* Email verification after signup.
* Password reset flow using Supabase + MailerSend.
* First-time onboarding wizard to guide user through:
  + Organization creation
  + Adding team members
  + Setting company logo and basic info
* Show trial countdown in dashboard.
* Prevent duplicate organizations (check org name/email).

**8. Deliverables**

* Supabase tables for users, organizations, org\_members.
* Auth API routes for signup, login, logout, password reset.
* Middleware to restrict pre-login access.
* Post-login flow to create org and assign user roles.
* Integration with trial/subscription system.
* Clear access control rules for all roles.

# **6. SwiftFacture Admin Page: Features & Capabilities**

**1. Dashboard Overview**

* **High-level metrics:**
  + Number of active users and organizations
  + Number of trials started/active/expired
  + Subscription breakdown (free, trial, premium)
  + Revenue from paid subscriptions (once Stripe integrated)
* **Recent activity feed:**
  + New signups
  + Trial starts/expirations
  + Invoices/receipts created
  + Support tickets submitted

**2. User Management**

* View all registered users across organizations.
* Filter users by: role (owner, admin, member), organization, subscription status.
* Actions per user:
  + Edit profile (name, email)
  + Change role within organization
  + Suspend or deactivate user
  + Reset password

**3. Organization Management**

* View all organizations and their metadata:
  + Organization name, business type, address, VAT/tax ID
  + Subscription plan and trial status
  + Number of members
  + Creation date
* Actions per organization:
  + Upgrade/downgrade subscription manually
  + Suspend/reactivate organization access
  + Delete organization (with warnings)
  + Send admin notifications or emails to org owners

**4. Subscription & Billing Oversight**

* Monitor all subscriptions (trial, active, expired, canceled).
* View payment history, invoices, and failed payment attempts (once Stripe integrated).
* Trigger manual subscription changes (upgrade, downgrade, trial extension).
* Generate reports on revenue and trial-to-paid conversion rates.

**5. Invoices, Receipts, & Estimates**

* View documents across all organizations.
* Search/filter by organization, user, client, date, or status.
* Download or resend invoices/receipts.
* Ensure data integrity and monitor unusual activity.

**6. Client Management**

* View all clients across organizations (linked to respective org).
* Edit or remove client info.
* Monitor client activity (invoices, payments, communication history).

**7. Support & Communication**

* Access all support tickets submitted by users.
* Respond directly to tickets, assign them to team members.
* Filter tickets by status: open, pending, closed.
* Send system-wide notifications or emails to users/orgs.

**8. System Monitoring & Logs**

* View login activity, failed auth attempts, and suspicious actions.
* Track API usage, MailerSend delivery stats, and trial notifications sent.
* Export logs for audit or compliance purposes.

**9. Settings & Configuration**

* Manage system-wide settings (branding, email templates, feature flags).
* Set default trial period duration, default currency, or document templates.
* Integrate/manage third-party services (Stripe, MailerSend, Twilio).
* Configure roles and permissions for future admin or moderator accounts.

**10. Security & Access**

* Only admin users can access this page.
* Sensitive actions (delete org, suspend user) require confirmation modal or 2FA.
* All actions are logged for auditing.

✅ **Summary:**  
The admin page acts as the **central hub for monitoring, managing, and controlling** every aspect of SwiftFacture. From users and organizations to billing, invoices, support, and system settings — everything is visible and actionable in one interface.

## **7. SwiftFacture Admin Page Layout**

**1. Top Navigation Bar (Header)**

* **Logo / Branding** (top-left)
* **Global Search Bar** (search users, organizations, invoices)
* **Notifications Icon** (for system alerts, emails, trial expirations)
* **Admin Profile Dropdown** (settings, logout)

**2. Side Navigation Panel (Sidebar)**

* Dashboard (overview metrics)
* Users
* Organizations
* Subscriptions / Billing
* Invoices / Receipts / Estimates
* Clients
* Support Tickets
* System Logs / Monitoring
* Settings

**3. Main Content Area (Dynamic Panels)**

**A. Dashboard Panel (Default Landing)**

* **Metrics Cards:**
  + Total Users
  + Total Organizations
  + Active Trials
  + Active Paid Subscriptions
  + Revenue Overview
* **Recent Activity Feed:**
  + New signups
  + Trial started/expired
  + Invoices created
  + Support tickets submitted
* **Charts:**
  + Trial → Paid conversion over time
  + Monthly revenue graph
  + Active vs inactive users

**B. Users Panel**

* **Table View:**
  + Columns: Name, Email, Role, Organization, Status, Last Login
  + Filters: Role, Organization, Status
* **Actions:**
  + Edit profile / role
  + Suspend / deactivate
  + Reset password
* Optional: Quick search per organization or email

**C. Organizations Panel**

* **Table View:**
  + Columns: Org Name, Owner, Members Count, Plan, Trial Status, Creation Date
* **Actions:**
  + View organization details
  + Upgrade/Downgrade subscription
  + Suspend / Reactivate organization
  + Delete organization
* Optional: Quick access to invoices and clients linked to the org

**D. Subscriptions / Billing Panel**

* **Table View:**
  + Org Name, Plan, Status, Trial End, Stripe ID
* **Actions:**
  + Manually upgrade/downgrade
  + Extend trial
  + View payment history
  + Resend receipts

**E. Invoices / Receipts / Estimates Panel**

* **Table View:**
  + Columns: Document Number, Org, User, Client, Status, Date, Total Amount
* **Actions:**
  + View/Download PDF
  + Resend email to client
  + Search by org, client, or status

**F. Clients Panel**

* **Table View:**
  + Columns: Client Name, Org, Email, Tax ID, Number of Documents
* **Actions:**
  + Edit client info
  + Delete client
  + View linked invoices/estimates

**G. Support Tickets Panel**

* **Table View:**
  + Columns: Ticket ID, Org, User, Subject, Status, Date
* **Actions:**
  + Respond / Assign / Close
  + Filter by status (open/pending/closed)
* Optional: Search tickets by user or org

**H. System Logs / Monitoring Panel**

* **Logs Table:**
  + Event, User, Org, Timestamp, Status
* **Filters:** Event type, date, organization, user
* **Charts:** Login attempts, failed auth, email delivery stats

**I. Settings Panel**

* **Sections:**
  + System-wide configurations (default trial days, currency, document templates)
  + MailerSend integration & templates
  + Stripe keys & billing settings
  + Roles & permissions for admins/moderators

**4. Footer / Extra**

* Version info
* Contact / support link

**✅ Notes for Implementation**

* **Responsive layout:** Sidebar collapses on mobile.
* **Dynamic content:** Panels update based on sidebar selection.
* **Search & filter:** Everywhere tables are displayed.
* **Action buttons:** Always confirm destructive actions (delete, suspend).
* **Analytics/Charts:** Use Chart.js or Recharts for revenue/trial analytics.

## 8. **i18n.js Language Translation Setup for SwiftFacture**

**1. Install i18n Library**

* Use [i18next](https://www.i18next.com/) or i18n.js depending on your framework (React, Vue, etc.).
* Install necessary packages:
* npm install i18next react-i18next

or appropriate for your stack.

**2. Default Language**

* **French** is the default language (fr).
* English (en) is the secondary language.
* Store user language preference in:
  + Local storage (localStorage)
  + Or Supabase user metadata (for logged-in users).

**3. Folder & File Structure**

/locales

/fr

translation.json

/en

translation.json

* Each JSON contains key-value pairs for all UI text.
* Example:

// fr/translation.json

{

"welcome": "Bienvenue sur SwiftFacture",

"login": "Connexion",

"signup": "Inscription"

}

// en/translation.json

{

"welcome": "Welcome to SwiftFacture",

"login": "Login",

"signup": "Sign Up"

}

**4. Initialize i18n**

* Configure i18n at app startup:
  + Set default language to French (fr).
  + Detect user language from browser or stored preference.
  + Enable fallback to French if translation missing.

**5. Wrap UI Components**

* Use t() function to translate text.
* Example in React:

import { useTranslation } from 'react-i18next';

const { t } = useTranslation();

<h1>{t('welcome')}</h1>

**6. Language Switcher**

* Provide toggle in UI (e.g., top-right corner or user profile menu).
* On language change:
  + Update i18n current language.
  + Save preference in localStorage or Supabase metadata.

**7. Dynamic Text / Variables**

* Support placeholders in translations:

"trial\_ends\_in": "Votre essai se termine dans {{days}} jours"

* Usage:

t('trial\_ends\_in', { days: 7 })

**8. Scope of Translation**

* All static UI text:
  + Buttons, labels, headings
  + Forms, placeholders, validation messages
  + Notifications & alerts (trial reminders, errors)
  + Admin page, dashboards, tables
* Emails can also be translated if MailerSend templates are created with **French default** and optional English versions.

**9. Fallback Strategy**

* French is default → if English translation missing, show French text.
* Keep JSON keys consistent between languages for maintainability.

**10. Optional Enhancements**

* Detect browser language and default to it if available.
* Allow organization-wide default language for multi-user accounts.
* Store language preference in Supabase users table for consistent experience across devices.

✅ **Summary:**  
Using **i18n.js**, French will be the default language for all users, with English as a selectable alternative. All UI components, notifications, and admin interfaces can be dynamically translated while storing user preference for future sessions.

## **9. GitHub + Netlify Deployment Workflow**

**1. GitHub Branch Strategy**

* **main** → Production branch
  + Netlify automatically deploys this branch to swiftfacture.com.
  + Only stable, tested features should go here.
* **develop** → Dev / Staging branch
  + Netlify deploys this branch to dev.swiftfacture.site.
  + Team tests new features here before merging into main.
* **feature branches** → Local development or isolated features
  + Example: feature/invoice-templates, feature/trial-updates
  + Developers work locally → push to GitHub → deploy preview URL on Netlify.

**2. Netlify Branch Deploys**

* Netlify can **automatically detect branches** and deploy them to separate URLs.
* Setup:
  1. In Netlify dashboard → **Deploy settings → Branch deploys**
  2. Map develop → dev.swiftfacture.site
  3. main → swiftfacture.com
  4. Feature branches → Netlify **preview URLs** (auto-generated)

**3. Environment Variables**

* Each branch can have **its own environment variables** in Netlify:
  + SUPABASE\_URL → dev vs prod
  + SUPABASE\_ANON\_KEY → dev vs prod
  + MAILERSEND\_API\_KEY → sandbox vs prod
  + STRIPE\_KEY → test vs live
* Keep sensitive keys **branch-specific** to avoid test data leaking into production.

**4. Workflow Summary**

1. **Local development** → feature branch → test locally.
2. **Push to GitHub** → feature branch → Netlify generates preview deploy URL for team review.
3. **Merge feature branch into develop** → auto-deploy to dev.swiftfacture.site → team tests integrated features.
4. **Merge develop into main** → auto-deploy to swiftfacture.com → production live.

**5. Benefits**

* Clear separation between **development, staging, and production**.
* Team can review new features before production release.
* Safe testing with **sandbox/test keys**.
* Automatic deployments reduce manual work.

## **10. Summary feature setup for SwiftFacture**

**1. Database & Supabase Tables**

* **Users** → connected to Supabase Auth, roles, org\_id.
* **Organizations** → org metadata, branding, subscription plan.
* **Org Members** → role-based access control.
* **Clients** → linked to organization.
* **Invoices / Receipts / Estimates** → linked to org + user + client.
* **Products / Services** → optional, to populate invoice line items.
* **Subscriptions / Billing** → trial start/end, plan, Stripe IDs.
* **Billing Events** → track trial reminders, upgrades, expirations.
* **Support Tickets** → org/user-linked, status tracking.
* **Logs** → authentication events, failed logins, system actions.

**2. Authentication & Roles**

* Supabase Auth integration.
* Roles: Owner, Admin, Member.
* Role-based Row Level Security (RLS) policies.
* Pre-login restriction middleware for protected routes.
* Post-login onboarding flow: create org, assign roles.

**3. Trial & Subscription System**

* 30-day free trial per organization.
* Trial reminders and expiration emails.
* Upgrade/downgrade plan workflow (Stripe integration later).
* Subscription status enforcement across app features.

**4. Email / Notification System**

* MailerSend integration for:
  + Auth verification & password reset
  + Trial lifecycle emails
  + Invoice/receipt notifications
  + Team invites
  + Support updates
* Template scaffolding with dynamic placeholders.

**5. Admin Page**

* Full dashboard for system overview, user/org management, subscriptions, invoices, clients, and logs.
* Ability to manage roles, subscriptions, and access.
* Analytics / charts for trials, conversions, and revenue.

**6. Organization-Centric Feature Flow**

* All user actions tied to **organization profile**:
  + Auto-fill invoices/receipts/estimates with org details.
  + Shared clients across org members.
  + Subscription upgrades applied to all org users.

**7. i18n & Multilingual Support**

* French default, English alternative.
* Translate all UI text, notifications, and email templates.
* Language preference stored in Supabase user metadata.

**8. Front-End Scaffolding**

* **Pages/Routes:**
  + Auth: Login, Signup, Password Reset
  + Dashboard (user & admin)
  + Invoices / Receipts / Estimates
  + Clients management
  + Subscription & Billing
  + Admin panel
  + Support / Contact page
* **Components:**
  + Tables, forms, modals, notifications, charts, language switcher, file uploads (PDF).

**9. API & Server Functions**

* Supabase RPCs or serverless functions for:
  + Billing logic (trial start/end, subscription upgrade)
  + Invoice PDF generation
  + Email triggers via MailerSend
  + Role validation & access control
  + Support ticket management

**10. Analytics & Monitoring**

* Track user activity: logins, trial usage, document creation.
* Monitor failed logins, email delivery, payment events.
* Optional: integrate PostHog, Sentry, or Supabase analytics.

**11. Document Generation**

* PDFs for invoices, receipts, and estimates.
* Templates with placeholders for org, client, and item info.
* Store generated files linked to org and user.

**12. Optional Extras to Scaffold Early**

* **Audit logs** for all important actions (invoice creation, subscription changes).
* **Theme / branding support** per organization (logo, colors).
* **Search & filter system** across invoices, clients, users.
* **Multi-currency & tax handling** (optional for future expansion).

**SwiftFacture AI Build Prompt**

**Task:** Build a **full-featured, organization-centric billing and invoicing web application** called **SwiftFacture**. The app should enable businesses to efficiently create, manage, and track **invoices, receipts, estimates, and clients** within a secure, multi-user environment. Core requirements include:

* **Authentication & Role-Based Access:** Secure login with Supabase Auth, onboarding workflow for new users, and roles (Owner, Admin, Member) to control access and permissions.
* **Organization-Centric Workflow:** Store company details once and auto-populate across all financial documents, allowing users to work efficiently without repetitive data entry.
* **Billing & Invoicing Features:** Create, edit, view, and manage invoices, receipts, and estimates linked to clients and organizations. Include PDF generation and email delivery.
* **Trial & Subscription Management:** Implement a 30-day free trial per organization, with notifications, plan upgrades, and subscription tracking. Prepare for future Stripe integration.
* **Admin Panel & Analytics:** Centralized dashboard to monitor users, organizations, subscriptions, invoices, clients, and support tickets. Include system logs, role management, and actionable insights.
* **Notifications & Emails:** Integrate transactional email functionality (MailerSend) for account verification, trial lifecycle, document delivery, team invites, and support updates.
* **Multilingual Support (i18n):** French as default language, English as secondary, with translations for all UI elements, notifications, and emails.
* **Multi-Environment Deployment:** Full support for local development, dev/staging deployment (swiftfacture.site), and production deployment (swiftfacture.com) on Netlify, with environment-specific configuration and secrets management.
* **Scalable, Extensible Architecture:** Build on Supabase backend and API-centric design to allow future enhancements like Stripe payments, analytics dashboards, feature flags, and multi-currency support.

**Objective:** Deliver a production-ready, maintainable, and fully functional web application that streamlines billing workflows, ensures data integrity, supports team collaboration, and allows safe iteration via multi-environment deployment.

**1. Project Overview**

* SwiftFacture is a **web-based invoicing and billing platform**.
* Users belong to **organizations**, which store company info (name, address, VAT/tax, logo, currency).
* Users can create/manage invoices, receipts, estimates, and clients.
* App should have **trial system, subscription plans, admin dashboard, notifications, and multi-language support (French default, English optional)**.

**2. Authentication & Onboarding**

* Use **Supabase Auth** (email/password login).
* Role-based access: **Owner, Admin, Member**.
* First-time users create an **organization** or join via invite.
* Pre-login users **cannot access protected routes**.
* Onboarding flow: create org, add optional team members.
* Store user/org metadata in Supabase.

**3. Billing Features**

**3.1 Organization-Centric Document Workflow**

* All invoices, receipts, and estimates are **linked to the user and organization**.
* Organization details (name, address, VAT/tax, logo, currency) automatically **populate documents**, reducing repetitive data entry.
* Users can associate documents with **clients** within their organization.

**3.2 Currency Support:**

* Set **Euro (€)** as the **default currency** for all organizations and financial documents.
* Provide **optional additional currencies** for selection: USD (US Dollar), USD (US Dollar $), UGX (Ugandan Shilling), NOK (Norwegian Krone), and GBP (British Pound).
* Ensure that **all invoices, receipts, and estimates** reflect the selected currency correctly in totals, taxes, and PDF exports.
* Users can **change currency per organization**, and it should persist across all documents for that org.

**3.3 CRUD Operations**

* Support **Create, Read, Update, Delete** for invoices, receipts, and estimates.
* Documents are stored in the database and retrievable for editing, previewing, or sending.
* Draft functionality: users can **save documents for later editing or review**.

**3.4 Dynamic Document Templates**

* Provide **multiple pre-designed templates** for invoices, receipts, and estimates.
* Each template supports **placeholders** for dynamic content, including:
  + Organization information
  + Client information
  + Line items, quantities, prices, totals, and taxes
  + Dates, notes, and logos
* Users can **select a template** and view a **real-time preview** before saving or sending.
* PDF generation should reflect the chosen template and populate all placeholders correctly.
* Optional: allow **per-organization customization** of template colors, logos, and layout.

**3.5 Demo / Sample Data**

* Implement a **“Load Sample Data”** button for invoices, receipts, and estimates.
* Clicking the button populates the document forms with **pre-filled example data**, including clients, items, totals, and organization info.
* Allows users to **preview templates, test PDF generation, and practice workflows** without creating real entries.
* Sample data should integrate seamlessly with **all templates**, support **draft saving**, and respect **preview functionality**.
* Optional: disable or hide in **production environment** so real users do not see demo data.

**3.6 Integration with Workflow**

* When a user creates a document:
  1. Select a **template**.
  2. Optionally click **“Load Sample Data”** to populate form.
  3. Fill in client or item details (or edit demo data).
  4. Preview dynamically populated document in real-time.
  5. Save as draft or send directly via email.
* Ensure **all templates, previews, and PDF generation** work with **live data or demo data**.

**4. Admin Panel**

* Centralized dashboard for system overview.
* Metrics: total users, orgs, trials, revenue, document activity.
* Tables for users, orgs, subscriptions, invoices, clients, support tickets.
* Actions: suspend/reactivate orgs/users, manage roles, upgrade/downgrade subscriptions.
* Logs & audit trails for critical actions.

**5. Notifications & Emails**

* Integrate **MailerSend** for transactional emails:
  + Account verification & password reset
  + Trial lifecycle notifications
  + Invoice/receipt delivery
  + Team invites & support updates
* Support dynamic placeholders and multiple languages.

**6. Multilingual Support (i18n)**

* French as **default**, English as secondary.
* Translate all UI elements, notifications, and email templates.
* Language switcher available in UI.
* Store user language preference in Supabase.

**7. Multi-Environment Deployment**

* Local development environment for feature testing (localhost).
* **Dev/staging branch**: develop branch deployed to dev.swiftfacture.site via Netlify branch deploys.
* **Production branch**: main branch deployed to swiftfacture.com.
* Separate environment variables for **local / dev / production** (Supabase, MailerSend, Stripe).

**8. Database & Backend**

* Supabase tables: users, organizations, org\_members, clients, invoices, receipts, estimates, subscriptions, billing\_events, support\_tickets, logs.
* Row-Level Security (RLS) to ensure **role-based access**.
* Backend APIs or serverless functions for:
  + Billing & trial logic
  + Document generation & retrieval
  + Notifications & email triggers
  + Role and access validation

**9. Frontend**

* Pages & Routes:
  + Auth (login, signup, password reset)
  + Dashboard (user & admin)
  + Invoices / Receipts / Estimates
  + Clients management
  + Subscription & billing
  + Admin panel
  + Support / contact page
* Components: tables, forms, modals, notifications, charts, language switcher, PDF viewer, file uploads.
* Responsive design for desktop and mobile.

**10. Workflow**

* Feature development in **feature branches** → merge into develop → deploy to dev/staging → test → merge to main → deploy to production.
* Use preview URLs for feature branches on Netlify.
* Ensure trial, email, and Stripe sandbox/test keys are isolated from production.

**11. Optional / Future Enhancements**

* Stripe integration for subscriptions & payments.
* Feature flags for experimental features.
* Multi-currency & tax handling.
* Analytics & reporting dashboards.
* Backup & restore strategy for Supabase.

**Deliverables**

* Full-stack web app with frontend, dynamic templates with preview, backend, database, and hosting integration.
* Authentication, onboarding, org-centric workflow, trial system, and role-based access fully functional.
* Admin dashboard with complete system overview and management capabilities.
* Email notifications & multilingual support implemented.
* Multi-environment workflow (local, dev/staging, production) working on Netlify.