

BillBuddy – WindSurf-AI Task Prompts (v2)

Updated: May 29, 2025 – Onboarding TOS & Streaming Services

Month 1 – Planning & Design

1.1 Initialize Git Repository & Workspace Rules

- # WindSurf-AI Flow: init_repo
1. Create a new **private** GitHub repo called `billbuddy`.
 2. Add a `.gitignore` suitable for Flutter (Dart) and Node/TypeScript.
 3. Add a `README.md` that includes a **project overview**, tech stack, and development workflow summary.
 4. Create a `.windsurfrules` file that sets:
 - Flutter project structure and naming conventions.
 - NestJS backend structure conventions.
 - Coding standards (lint rules, comments required).
 - Secure coding rules (no **hard-coded** secrets; use ENV vars).
 - Testing requirement: every feature must ship with unit + integration tests.
 5. Commit and push to `main`; set branch protection to require PR + CI pass.

1.2 Create Figma Wireframes & Style Guide

- # WindSurf-AI Flow: design_wireframes
1. Create a new Figma project named **BillBuddy v1**.
 2. Generate **low-fidelity** wireframes for:
 - Onboarding wizard (4–5 screens).
 - Main dashboard (mobile + web).
 - Bill detail & payment screens.
 - Admin dashboard overview.
 3. Produce a **style guide** frame with:
 - Primary/secondary colors (professional fintech blue/green palette), HEX values.
 - Typography: headings, body, caption.
 - UI components: buttons, **text** fields, cards, icons.
 4. Export a share link and write it to a file `DESIGN_LINK.md`.

1.3 Define Database Schema

- # WindSurf-AI Flow: db_schema
1. Using the ERD description, generate SQL migration scripts for PostgreSQL that create tables: `users`, `accounts`, `bills`, `payments`, `documents`, `subscriptions`, `offers`, `user_offers`, `negotiations`, `admins`, `audit_logs`.
 2. Place scripts in `backend/migrations/2025_01_initial.sql`.
 3. Generate ORM entities (TypeORM) under `backend/src/entities/`.
 4. Add unit tests that spin up a dockerized Postgres in CI and run migrations.

1.4 Bootstrap CI/CD Pipeline

- # WindSurf-AI Flow: setup_ci
1. Add GitHub Actions workflow `.github/workflows/ci.yml`.
 2. Steps:
 - Setup Node and install backend deps.
 - Setup Flutter (`flutter-action`) and run `flutter pub get`.
 - Run `npm run test` for backend.
 - Run `flutter test` and `flutter analyze`.
 - Cache dependencies.
 3. On success of `main` branch, deploy backend to staging (use AWS EB CLI placeholder) and deploy Flutter web to staging bucket.

Month 2 – Core Development

2.1 Generate Flutter App Skeleton

- # WindSurf-AI Flow: flutter_skeleton
1. Use `flutter create billbuddy` with `--platforms ios,android,web`.
 2. Add Riverpod package for state management.
 3. Scaffold navigation: `SplashScreen` → `Onboarding` → `Dashboard`.
 4. Implement adaptive layout breakpoints for web vs mobile.

2.2 Build Onboarding Wizard with OCR

WindSurf-AI Flow: onboarding_wizard (UPDATED)

****Wizard Page Sequence****

- 1) Welcome & value prop splash
- 2) ****Terms of Service / Privacy Agreements****
 - Display embedded TOS/Privacy HTML (loaded from `/assets/legal/tos.html`).
 - Disable "Continue" until user scrolls to bottom of the document.
 - Require checkbox ****"I agree"**: on tap store `tosAcceptedAt` UTC in `users` table.**
- 3) Sign up / Login (email + password + Google/Apple)
- 4) Bill Capture – Scan first bill (camera / file)
- 5) Confirm extracted bill info & category
- 6) ****Add Streaming Services****
 - Auto-suggest common services (Netflix, Hulu, Disney+, Prime, Max, Peacock).
 - Allow one tap "Add" → opens provider picker; user selects and enters email/password OR manually enters monthly amount + due day.
 - Save each streaming bill with category `subscription_streaming`.
- 7) Link Bank / Select Payment Method (optional for now)
- 8) Success screen → Go to Dashboard

****Technical Tasks****

- Create `TermsOfServicePage` widget with `ScrollablePositionedList` listener; enable `Continue` only when `scrollController.offset == maxScrollExtent`.
- Add `LegalService` to fetch latest TOS/Privacy from backend `/legal/tos` (so we can update without app release).
- Extend `OnboardingStepController` enum with `terms`, `streaming`.
- Add `StreamingServicePicker` component; pre-populate list via `static/services/streaming.json`.
- Unit test: verify TOS must be scrolled before proceeding.
- Integration test: complete full onboarding flow including adding a streaming service stub.

2.3 Backend API Skeleton

WindSurf-AI Flow: backend_api

1. Use NestJS CLI to create project in `backend`.
2. Implement modules: Auth, Users, Bills, Payments.
3. Auth: email/password + Google OAuth (passport.js strategies).
4. BillsController CRUD endpoints with DTO validation.
5. Auto-generate OpenAPI (Swagger) docs at `/api/docs`.

2.4 Authentication & Session

WindSurf-AI Flow: auth_sessions

1. Add JWT auth (access + refresh tokens) with 15m/30d expiry.
2. Secure cookies for web; secure storage (flutter_secure_storage) for mobile.
3. Enable MFA toggle in user table, placeholder field.
4. Tests: verify login, token refresh, protected route.

2.5 Streaming Service Endpoints

WindSurf-AI Flow: streaming_endpoints

1. Backend: create `StreamingService` entity (id, name, logo_url) seeded with top 20 providers.
2. Add `/streaming/list` (public) to fetch providers for picker.
3. Extend `Bill` entity schema with `is_streaming` boolean.
4. Endpoint `/bills/streaming/add` – save subscription amount, renewal day.
5. Tests: create streaming bill via API, ensure category set correctly.

Month 3 – Integrations & Payments

3.1 Plaid Integration

WindSurf-AI Flow: plaid_link

1. Add Plaid Link SDK to Flutter (mobile + web).
2. On success, send `public_token` to backend `/plaid/exchange`.
3. Backend exchanges token, saves `access_token` encrypted.
4. Fetch accounts; return list to client for user selection.

3.2 ACH Payments via Stripe

WindSurf-AI Flow: stripe_ach

1. Backend: integrate Stripe SDK; create Customer per user.
2. Attach bank account using Plaid processor token.
3. Implement `/payments/schedule` that creates `PaymentIntent` with ACH debit (amount, description).
4. Webhook `/stripe/webhook` to update payment status in DB.

3.3 Headless Scraping Service

WindSurf-AI Flow: bill_scraper

1. Create `scraper` service (Node) using Playwright.
2. Accept job (biller_login, url, creds) from Redis queue.
3. Launch headless chromium, login, navigate to bill page, extract amount & due date, push update via gRPC to backend.
4. Add fallback/error handling; retry 3 times; log to audit.

3.4 OCR Provider Switch

WindSurf-AI Flow: ocr_switch

1. Abstract OCR service interface `OCRProvider`.
2. Implement `TesseractProvider` (default) and `GoogleVisionProvider`.
3. Admin API `/admin/ocr/provider` to toggle.
4. When premium user, call premium provider automatically.

Month 4 – Premium Features & Refinement

4.1 Negotiation Tool MVP

WindSurf-AI Flow: negotiate_tool

1. Create `NegotiationRequest` model/table.
2. Add UI: user selects bill and clicks 'Negotiate'.
3. Generate support email template with LLM (OpenAI) summarizing request.
4. Store status; allow admin to mark completed and input new amount.

4.2 Analytics Dashboard

WindSurf-AI Flow: analytics_dash

1. Backend: aggregate spend per category and month.
2. Flutter: implement `charts_flutter` line & pie charts under `Insights` tab.
3. Add endpoint `/analytics/summary` returning JSON for charts.
4. Unit tests validate aggregation correctness.

4.3 Affiliate Offers Engine

WindSurf-AI Flow: offers_engine

1. Create `Offer` entity and admin CRUD UI.
2. Dashboard: context card shows top 3 offers relevant by bill category.
3. Track clicks via `/offers/{id}/click`; store in `user_offers`.
4. Include UTM params for affiliate tracking.

4.4 Premium Subscription IAP

WindSurf-AI Flow: premium_iap

1. Flutter: integrate `in_app_purchase` (Apple/Google).
2. Define product id `billbuddy_premium_monthly`.
3. On purchase success, verify receipt with backend `/iap/verify`.
4. Backend sets `premium_status=true`, returns JWT with claim.

Month 5 – Testing, Compliance & Launch

5.1 Comprehensive Test Suite

WindSurf-AI Flow: test_suite

1. Achieve $\geq 80\%$ code coverage:
 - Flutter: widget & integration tests.
 - Backend: jest unit + supertest API integration tests.
2. Add `coverage-badge` action to README.

5.2 Security Hardening

WindSurf-AI Flow: security_hardening

1. Run Snyk and OWASP dependency check in CI.
2. Implement rate limiting middleware (e.g., 100 req/min).
3. Enforce HTTPS redirects, HSTS headers.
4. Perform script to rotate all API keys in staging.

5.3 Compliance Automation

WindSurf-AI Flow: compliance_check

1. Add GDPR export/delete endpoints under `/privacy`.
2. Integrate Stripe Radar or similar fraud detection.
3. Automatic sanctions screening via ComplyAdvantage API at user signup.
4. Generate SOC2 readiness checklist markdown.

5.4 App Store Submission

WindSurf-AI Flow: store_release

1. Generate production iOS (IPA) & Android (AAB) builds via fastlane.
2. Upload screenshots from Figma export.
3. Populate app metadata JSON (`metadata/en-US.json`) from marketing copy file.
4. Submit for review; wait status webhook to Slack.

AI Enhancements (Post-MVP)

A1 Intelligent Bill Categorization

WindSurf-AI Flow: ml_categorize

1. Export anonymized bill dataset CSV.
2. Train Scikit-Learn TF-IDF + SVM model to classify biller into categories (utilities, loans, etc.).
3. Serialize model as `bill_classifier.pkl`; deploy behind `/ml/categorize`.
4. Add inference microservice; fallback to rule-based if confidence < 0.75.

A2 Personalized Insights Generator

WindSurf-AI Flow: insights_nlg

1. Define Jinja template prompts for spending insight generation.
2. Use OpenAI GPT-4 API; feed aggregated spend metrics as system context.
3. Return 3 actionable insights JSON; surface in 'Insights' tab weekly.
4. Schedule Cloud Task weekly per user.

A3 Chatbot Assistant

WindSurf-AI Flow: chatbot

1. Integrate OpenAI Assistants API with function calling to backend for data.
2. Define commands: `next_bill`, `total_spend`, `enable_autopay`.
3. Chat UI widget added to dashboard; context limited to user data.
4. Add moderation check for user queries.

A4 Fraud Detection Rules

WindSurf-AI Flow: ml_fraud

1. Build a light gradient-boosted tree model on payment features (amount, frequency, device fingerprint).
2. Trigger `review_required` flag if risk score > 0.9.
3. Admin dashboard shows flagged payments for manual action.
4. Log model predictions for future retraining.