MVP PRODUCT LAUNCH PLAN

Online Tutoring Marketplace Platform

Preply-Inspired | Next.js | Multi-Language | Fully Configurable

Version 1.0 | February 2026

Table of Contents

1. Executive Summary & MVP Philosophy
2. Platform Architecture & Configurable Design
3. Phase 1: Foundation & Core Infrastructure (Weeks 1-3)
4. Phase 2: Core Marketplace Features (Weeks 4-6)
5. Phase 3: Booking, Payments & Communication (Weeks 7-9)
6. Phase 4: Polish, SEO & Launch Prep (Weeks 10-12)
7. Post-MVP Roadmap
8. Tech Stack & Dependencies
9. Configuration Architecture
10. Multi-Language (i18n) Strategy
11. SEO Strategy for Next.js
12. Database Schema (Core Entities)
13. API Structure
14. Risk Matrix & Mitigations
15. Success Metrics

1. Executive Summary & MVP Philosophy

This document outlines the MVP launch plan for an online tutoring marketplace platform inspired by Preply. The platform connects learners with tutors for live, one-on-one lessons across multiple subjects and languages. Built with Next.js for exceptional SEO performance, the platform is designed to be fully configurable with zero hardcoded values, supporting multiple languages from day one.

MVP Guiding Principles

Ship fast, iterate faster: 12-week timeline to a production-ready MVP. Every feature earns its place by directly enabling the core booking loop.

Configuration over code: Every label, rate, commission percentage, subject category, and UI string is stored in the database or environment config — never hardcoded.

Multi-language native: i18n is not a Phase 2 bolt-on. The system is built from the start to support

configurable locales, RTL layouts, and locale-aware SEO routes.

Minimalist & clean UI: Following Preply's design language — generous whitespace, clear typography hierarchy, trust signals (reviews, verified badges), and a frictionless search-to-book flow.

SEO as a growth engine: SSR/SSG via Next.js App Router, dynamic metadata, structured data (JSON-LD), hreflang tags, and programmatic landing pages for every subject + language + city combination.

Core Loop: Search tutor → View profile → Book trial lesson → Pay → Attend lesson → Subscribe for more. Every MVP feature must support this loop or be deferred.

2. Platform Architecture & Configurable Design

The platform follows a modern, decoupled architecture that separates concerns and ensures every user-facing element is driven by configuration rather than code changes.

Architecture Overview

Frontend: Next.js 14+ (App Router) with React Server Components for SEO-critical pages. Client components only where interactivity is needed (booking widget, chat, filters).

Backend API: Next.js API Routes + tRPC or REST endpoints. Separate service layer for business logic.

Database: PostgreSQL via Prisma ORM. All configurable values stored in dedicated config tables.

Auth: NextAuth.js (Auth.js) with configurable providers (Google, Email/Password, Facebook — all toggleable via config).

Payments: Stripe Connect for marketplace payments. Commission rates, trial pricing, and subscription tiers all configurable.

Video: Integration-ready for Daily.co, Twitch, or Zoom SDK — provider configurable via environment variable.

CMS/Config: Admin dashboard for all platform settings. No developer needed to change subjects, pricing, languages, or UI copy.

File Storage: S3-compatible (AWS S3, Cloudflare R2) for profile photos, intro videos, and lesson materials.

Search: PostgreSQL full-text search for MVP. Meilisearch/Algolia upgrade path configurable.

Configurable Design Principle

The following categories are fully configurable via the admin panel or environment variables. No code deployment is required for any of these changes:

Platform Identity	App name, logo, favicon, tagline, primary/secondary colors, social links	DB + .env
Subjects & Categories	Subject tree (languages, academics, skills), icons, slug, display order	DB (admin panel)
Pricing	Commission %, trial lesson price, min/max hourly rates, currency options	DB (admin panel)
Locales	Supported languages, default locale, RTL flag, date/number formats	DB + i18n files

Auth Providers	Enable/disable Google, Facebook, Apple, Email+Password sign-in	.env + DB toggle
SEO	Meta templates, OG image defaults, structured data schemas, sitemap rules	DB (admin panel)
Email Templates	Booking confirmations, reminders, review prompts — all templated	DB (admin panel)
Feature Flags	Video lessons, group classes, messaging, reviews — toggle on/off	DB (admin panel)
Lesson Settings	Duration options (25/50 min), buffer time, cancellation policy hours	DB (admin panel)
Landing Pages	Hero text, CTA labels, testimonials, stats counters	DB (admin panel)

3. Phase 1: Foundation & Core Infrastructure

Weeks 1–3 | Goal: Solid technical foundation with i18n and config system ready

Week 1: Project Setup & Config System

- Initialize Next.js 14+ project with App Router, TypeScript, and Tailwind CSS
- Set up PostgreSQL database with Prisma ORM and seed scripts
- Create the configuration system: platform_config table with key-value pairs, typed config loader, admin API endpoints for CRUD
- Implement environment variable schema validation (using zod or similar)
- Set up next-intl or similar for i18n routing: /en/tutors, /es/tutores, /fr/tuteurs
- Create translation file structure with namespace separation (common, auth, tutor, booking)
- Configure ESLint, Prettier, Husky pre-commit hooks
- Set up CI/CD pipeline (GitHub Actions → Vercel or similar)
- Create base layout with configurable header, footer, and language switcher

Week 2: Authentication & User Roles

- Implement NextAuth.js with configurable providers (credentials + OAuth)
- Create role-based access: Learner, Tutor, Admin — roles stored in DB, not code
- Build registration flows for both learner and tutor (separate onboarding)
- Tutor onboarding: profile details, subject selection (from configurable list), intro video upload, rate setting, availability calendar
- Learner onboarding: learning goals, preferred schedule, budget range
- Email verification with configurable email templates (using Resend, SendGrid, or similar — provider configurable)
- Password reset flow with rate limiting
- Session management and JWT token configuration

Week 3: Database Schema & Admin Foundation

- Complete database schema for all core entities (see Section 12)
- Build admin dashboard shell with sidebar navigation
- Admin: Platform settings page (name, logo, colors, feature flags)
- Admin: Subject/category management (CRUD with drag-and-drop ordering)
- Admin: Language/locale management (add locales, manage translation keys)

- Admin: User management (list, search, suspend, role change)
- Set up database migrations strategy and seed data for development
- Create reusable UI component library: Button, Input, Select, Modal, Card, Avatar (all respecting theme config)

Phase 1 Deliverable: Working app shell with authentication, admin panel, configuration system, i18n routing, and reusable components. No marketplace features yet, but the infrastructure for everything configurable is in place.

4. Phase 2: Core Marketplace Features

Weeks 4–6 | Goal: Tutors can list themselves, learners can discover and filter tutors

Week 4: Tutor Profiles & Search

- Tutor profile page (SSR for SEO): photo, intro video player, bio, subjects taught, languages spoken, hourly rate, availability summary, reviews section
- Tutor search/listing page with configurable filters: subject, price range (min/max from config), language, native speaker toggle, availability, rating
- Full-text search with PostgreSQL (upgrade path to Meilisearch configured via env)
- Tutor card component: avatar, name, rating stars, subjects, price, "Book Trial" CTA
- Pagination and infinite scroll (configurable: pagination vs infinite)
- Sort options: relevance, price low-high, price high-low, rating, number of reviews
- Responsive grid layout: 1/2/3 columns based on viewport

Week 5: Programmatic SEO Pages

- Auto-generated landing pages for every configurable subject: /en/online/english-tutors, /es/online/profesores-de-ingles
- Auto-generated city + subject pages (if city data configured): /en/english-tutors-in-melbourne
- Dynamic metadata generation: title, description, OG tags — all from configurable templates
- JSON-LD structured data for tutor profiles (Person schema), course listings (Course schema), and organization (Organization schema)
- Hreflang tags on all pages linking equivalent content across configured locales
- XML sitemap generation (dynamic, includes all tutor profiles, subject pages, city pages)
- Robots.txt configuration via admin panel
- Breadcrumb navigation with structured data markup

Week 6: Tutor Dashboard & Availability

- Tutor dashboard: overview stats (views, bookings, earnings), upcoming lessons, recent reviews
- Availability calendar: weekly recurring slots + specific date overrides
- Timezone handling: all times stored in UTC, displayed in user's local timezone (configurable default timezone)
- Tutor profile editor: update bio, subjects, rate, video, teaching style, certifications
- Subject expertise levels (beginner, intermediate, advanced — configurable labels)
- Profile completion progress indicator with configurable required fields
- Tutor verification system (admin-controlled badges: verified, top-rated, certified)

Phase 2 Deliverable: Fully functional tutor marketplace. Tutors can create rich profiles, learners can search and

filter, and every page is SEO-optimized with dynamic metadata and structured data. Programmatic landing pages are live.

5. Phase 3: Booking, Payments & Communication

Weeks 7–9 | Goal: Complete booking loop — learners can book, pay, and attend lessons

Week 7: Booking System

- Booking widget on tutor profile: select date → see available slots → choose duration (from config) → confirm
- Trial lesson flow: configurable trial price (can be $0), limited to one trial per tutor-learner pair
- Lesson duration options from config (e.g., 25 min, 50 min, 75 min)
- Booking confirmation with email notification (template from config)
- Calendar sync: iCal feed export for tutor and learner
- Cancellation policy enforcement (configurable: free cancellation up to X hours before, defined in config)
- Rescheduling flow with mutual agreement
- Booking conflict detection and slot locking during checkout

Week 8: Payments & Subscription

- Stripe Connect integration: tutors onboard as connected accounts
- Configurable commission model: platform takes X% (stored in config, changeable per subject if needed)
- Payment flow: learner pays → platform holds → lesson completed → tutor gets paid (minus commission)
- Subscription/package model: configurable lesson bundles (e.g., 4 lessons/month, 8 lessons/month)
- Trial lesson pricing: configurable per-subject or platform-wide
- Multiple currency support (currencies enabled via config, exchange rates from API or manual config)
- Refund policy implementation (configurable refund rules)
- Payout schedule for tutors (configurable: weekly, bi-weekly, monthly)
- Payment history and invoicing for both learner and tutor dashboards

Week 9: Messaging & Video Integration

- In-app messaging between learner and tutor (pre-booking inquiries + post-booking coordination)
- Message notifications (in-app + email, frequency configurable)
- Video lesson integration: configurable provider (Daily.co, Zoom, Jitsi — switchable via env)
- Virtual classroom with: video call, screen sharing, text chat, shared whiteboard (feature set depends on provider)
- Lesson join page with pre-call device check
- Automated lesson reminders: configurable timing (e.g., 24h before, 1h before)
- Post-lesson review prompt: configurable delay (e.g., 30 min after lesson ends)
- Review system: 1–5 star rating + text review, displayed on tutor profile

Phase 3 Deliverable: The core loop is complete. A learner can discover a tutor, book a trial, pay, join a video lesson, and leave a review. The platform is now a functioning marketplace.

6. Phase 4: Polish, SEO & Launch Prep

Weeks 10–12 | Goal: Production-ready polish, performance optimization, and launch

Week 10: UI/UX Polish & Preply-Inspired Design

- Design audit against Preply's principles: generous whitespace, clear visual hierarchy, trust signals prominent
- Homepage: configurable hero section, "How It Works" steps, featured tutors (algorithmically or manually curated via config), testimonials (from config), subject category grid
- Mobile responsiveness pass on all pages (mobile-first refinement)
- Loading states, skeleton screens, and empty states for all data-driven pages
- Error pages (404, 500) with configurable messaging and helpful navigation
- Accessibility audit: ARIA labels, keyboard navigation, color contrast (WCAG 2.1 AA)
- Micro-interactions: smooth transitions, hover effects, toast notifications
- Image optimization: Next.js Image component, WebP serving, lazy loading with blur placeholders

Week 11: SEO & Performance

- Core Web Vitals optimization: target LCP < 2.5s, FID < 100ms, CLS < 0.1
- Next.js ISR (Incremental Static Regeneration) for tutor profiles and landing pages
- Edge caching strategy for static assets and API responses
- Google Search Console setup and XML sitemap submission
- Canonical URL strategy for duplicate content prevention
- Open Graph and Twitter Card meta for social sharing (templates configurable)
- Blog/content section for organic traffic (optional, CMS-driven if enabled via config)
- Analytics integration: configurable provider (Google Analytics, Plausible, PostHog — via env)
- Performance monitoring: configurable (Vercel Analytics, Sentry, New Relic)

Week 12: Testing, Security & Launch

- End-to-end testing of the complete booking flow (Playwright or Cypress)
- Security audit: CSRF protection, rate limiting, input sanitization, SQL injection prevention (Prisma handles most)
- GDPR/privacy compliance: cookie consent (configurable banner), data export, account deletion
- Terms of Service and Privacy Policy pages (content from config/CMS)
- Payment security review: PCI compliance via Stripe (no card data touches your server)
- Load testing: simulate concurrent bookings and search queries
- Staging environment final review with real-world test scenarios
- DNS, SSL, and production deployment configuration
- Launch checklist: monitoring dashboards, error alerting, backup strategy
- Soft launch to beta testers, gather feedback, iterate for 1–2 days

> **Phase 4 Deliverable:** Production-ready MVP. Polished, performant, secure, and SEO-optimized. Ready for public launch with monitoring and analytics in place.

7. Post-MVP Roadmap

These features are explicitly deferred from the MVP to maintain focus on the core loop. Each is

designed to be enabled via the existing feature flag system when ready.

P1	AI Tutor Matching	Algorithm recommending tutors based on learner goals, budget, schedule, and past behavior	2–3 weeks
P1	Mobile App (React Native)	Native mobile experience sharing business logic with the web platform	6–8 weeks
P1	Group Classes	One tutor, multiple learners. Configurable max size, different pricing model	3–4 weeks
P2	Preply Business (B2B)	Corporate training: company accounts, team management, bulk billing, progress reporting	4–6 weeks
P2	AI Study Tools	AI-powered vocabulary flashcards, grammar exercises, lesson summaries	3–4 weeks
P2	Advanced Analytics	Tutor earnings analytics, learner progress tracking, platform-wide metrics	2–3 weeks
P3	Affiliate Program	Configurable referral rewards for both tutors and learners	2 weeks
P3	Certifications	Learners earn configurable certificates upon completing lesson milestones	2 weeks
P3	Homework/Materials	Tutors assign homework, share materials, track completion between lessons	2–3 weeks

8. Tech Stack & Dependencies

Framework	Next.js 14+ (App Router)	SSR/SSG/ISR for SEO, React Server Components, API routes	N/A
Language	TypeScript	Type safety, better DX, fewer runtime errors	N/A
Styling	Tailwind CSS + shadcn/ui	Utility-first, customizable theme tokens, accessible components	Theme via config
Database	PostgreSQL	Robust, relational, excellent full-text search, JSON support	Connection via env
ORM	Prisma	Type-safe queries, migrations, seeding, multi-DB support	DB provider via env

Auth	NextAuth.js (Auth.js)	Multiple providers, JWT/ session, role-based access	Providers via env
Payments	Stripe Connect	Marketplace payments, subscriptions, global payouts	Keys via env
Video	Daily.co (default)	WebRTC, recording, screen share. Swappable provider	Provider via env
i18n	next-intl	App Router compatible, SSR-safe, namespace support	Locales via config
Email	Resend + React Email	Transactional emails with React templates. Provider swappable	Provider via env
File Storage	S3 / Cloudflare R2	Object storage for media. Provider-agnostic via SDK	Provider via env
Search	PostgreSQL FTS (MVP)	Built-in, no extra service. Upgrade to Meilisearch later	Engine via env
Monitoring	Sentry + Vercel Analytics	Error tracking + performance metrics	Provider via env
Hosting	Vercel (default)	Optimized for Next.js. AWS/Docker alternative path	Platform via config

9. Configuration Architecture

The configuration system is the backbone of the platform. It ensures that no code deployment is needed for business-level changes. Configuration is layered in three tiers:

Tier 1: Environment Variables (.env)

Infrastructure-level settings that require a restart or redeploy: database URL, API keys for Stripe/ video/email/storage, auth provider credentials, Node environment, and base URL.

Tier 2: Database Configuration (Admin Panel)

Business-level settings changeable in real-time: platform name and branding, commission rates, subject categories, pricing rules, feature flags, email templates, SEO templates, and lesson settings.

Tier 3: Translation Files (i18n)

All user-facing strings organized by namespace and locale. Loaded at build time with ISR revalidation. Admin panel provides a translation editor for non-technical team members.

Config Loading Pattern

Server components fetch config at render time from the database (cached with ISR). Client components receive config via React context provider hydrated from the server. Config changes propagate within the ISR revalidation window (configurable, default 60 seconds).

> **Key Rule:** If a value might ever need to change without a code deploy, it must be in Tier 2 (database) or Tier 3 (translations). Only true infrastructure secrets belong in Tier 1.

10. Multi-Language (i18n) Strategy

URL Structure

Locale-prefixed URLs using Next.js middleware and next-intl. The default locale can be configured to either show or hide the prefix.

/en/online/english-tutors	English	English tutors listing (English UI)
/es/online/profesores-de-ingles	Spanish	English tutors listing (Spanish UI)
/fr/en-ligne/tuteurs-danglais	French	English tutors listing (French UI)
/ar/online/english-tutors	Arabic	English tutors listing (Arabic UI, RTL)
/en/tutor/john-doe	English	Tutor profile (English UI)
/es/tutor/john-doe	Spanish	Same tutor profile (Spanish UI)

Translation Architecture

- JSON translation files organized by namespace: common.json, auth.json, tutor.json, booking.json, admin.json, seo.json
- Interpolation support for dynamic values: "{{count}} tutors available in {{subject}}"
- Pluralization rules per locale: "1 lesson" vs "5 lessons" vs Arabic plural forms
- Configurable slug translations per locale (stored in DB): "tutors" → "tutores" → "tuteurs"
- Fallback chain: requested locale → default locale → translation key as fallback
- Admin translation editor: non-technical team can update any string without code access

SEO-Specific i18n

- Hreflang tags on every page linking all locale variants
- Locale-specific metadata templates: configurable per-locale title and description patterns
- Alternate XML sitemaps per locale
- RTL stylesheet loading for Arabic, Hebrew, and other RTL locales (configurable flag per locale)
- Locale-specific structured data (JSON-LD) with inLanguage property

11. SEO Strategy for Next.js

SEO is a first-class citizen of this platform. Every architectural decision is made with crawlability, indexability, and ranking in mind.

Rendering Strategy by Page Type

Homepage	ISR	Every 60s (configurable)	Dynamic content (featured tutors, stats) but needs fast load
Subject landing pages	ISR	Every 5 min (configurable)	Programmatic SEO pages, content changes slowly
Tutor profile	ISR	Every 60s (configurable)	Reviews and availability update, but profile is mostly static
Tutor search/listing	SSR	Per request	Filters create unique URL combinations, needs fresh data
Blog/content pages	SSG	On rebuild	Static content, no dynamic data
Dashboard (learner/tutor)	CSR	N/A	Authenticated, no SEO value, fully interactive

Booking/checkout	CSR	N/A	Authenticated, no SEO value, real-time slots

Metadata Template System

All page titles and descriptions are generated from configurable templates with variable substitution. For example, the subject listing page template might be: "Best {{subject}} Tutors Online | Learn {{subject}} 1-on-1 | {{platformName}}" where {{subject}} is replaced dynamically and {{platformName}} comes from the platform config.

Structured Data (JSON-LD)

- Organization schema on the homepage with configurable name, logo, and social profiles
- Person schema on tutor profiles with name, image, description, and teaches relationship
- Course schema on subject pages listing available tutors as offers
- BreadcrumbList schema on all pages for enhanced SERP display
- FAQPage schema on relevant landing pages (FAQ content from config)
- Review/AggregateRating schema on tutor profiles showing rating data

Programmatic Landing Pages

Following Preply's strategy, the platform auto-generates SEO landing pages for every combination of configurable dimensions. These pages are generated from configurable templates and populated with live tutor data. Examples include subject pages (/en/online/english-tutors), city pages (/en/english-tutors-in-melbourne), and level pages (/en/online/english-tutors-for-beginners). All slugs, titles, and descriptions are configurable per locale.

12. Database Schema (Core Entities)

The following shows the core database entities for the MVP. All enum values, labels, and configurable fields are stored in their own tables rather than as code-level constants.

User	id, email, name, role (LEARNER/ TUTOR/ADMIN), avatar_url, locale, timezone, email_verified, created_at	Single user table with role differentiation
TutorProfile	user_id (FK), bio, intro_video_url, hourly_rate, currency, is_verified, is_active, total_lessons, avg_rating, response_time	Extended profile for tutor role
Subject	id, slug, name_translations (JSON), parent_id (self-ref), icon, display_order, is_active	Hierarchical, fully configurable
TutorSubject	tutor_id, subject_id, expertise_level, is_primary	Many-to-many with metadata
Availability	tutor_id, day_of_week, start_time, end_time, is_recurring, specific_date, timezone	Recurring + override slots
Booking	id, learner_id, tutor_id, subject_id, start_time, end_time, duration_minutes, status (PENDING/ CONFIRMED/COMPLETED/ CANCELLED), type (TRIAL/ REGULAR)	Core booking record

Payment	id, booking_id, amount, currency, platform_fee, tutor_payout, stripe_payment_id, status, refund_amount	Financial record per booking
Subscription	id, learner_id, tutor_id, lessons_per_period, period (WEEKLY/MONTHLY), price, status, stripe_subscription_id	Recurring lesson packages
Review	id, booking_id, learner_id, tutor_id, rating (1-5), comment, is_visible, created_at	Post-lesson reviews
Message	id, sender_id, receiver_id, booking_id (optional), content, read_at, created_at	In-app messaging
PlatformConfig	key (unique), value (JSON), category, description, updated_by, updated_at	All configurable settings
Translation	id, locale, namespace, key, value, updated_at	Dynamic translations manageable via admin
LandingPage	id, slug_translations (JSON), subject_id, city, meta_title_template, meta_desc_template, content (JSON), is_active	Programmatic SEO pages

13. API Structure

RESTful API routes organized by resource. All responses follow a consistent format with pagination, error codes, and locale-aware data.

GET	/api/config/:key	Get platform configuration value	Public (cached)
PUT	/api/admin/config/:key	Update configuration value	Admin
GET	/api/subjects	List all active subjects (localized)	Public
GET	/api/tutors	Search tutors with filters (paginated)	Public
GET	/api/tutors/:id	Get tutor profile details	Public
POST	/api/tutors/profile	Create/update tutor profile	Tutor
GET	/api/tutors/:id/availability	Get tutor available slots	Public
PUT	/api/tutors/availability	Update tutor availability	Tutor
POST	/api/bookings	Create a new booking	Learner
PUT	/api/bookings/:id/cancel	Cancel a booking	Learner/Tutor
POST	/api/payments/checkout	Create Stripe checkout session	Learner
POST	/api/payments/webhook	Stripe webhook handler	Stripe (verified)
GET	/api/ messages/:conversationId	Get conversation messages	Auth
POST	/api/messages	Send a message	Auth
POST	/api/reviews	Submit a lesson review	Learner

GET	/api/translations/:locale/:namespace	Get translations for locale/namespace	Public (cached)

14. Risk Matrix & Mitigations

Scope creep beyond 12 weeks	High	High	Strict phase gates. Features not in the core loop are deferred. Weekly scope reviews.
Video integration complexity	High	Medium	Use Daily.co managed service (no self-hosted WebRTC). Provider is swappable via env if issues arise.
Stripe Connect onboarding friction	Medium	Medium	Provide clear tutor onboarding guide. Use Stripe Express for simplest flow. Manual payout fallback via config.
SEO pages not indexed quickly	Medium	Medium	Submit sitemaps proactively. Use Google Indexing API. Start with high-value pages first.
Performance bottlenecks at scale	High	Low (MVP)	PostgreSQL indexes, ISR caching, CDN for static assets. Monitoring alerts for slow queries.
i18n translation quality	Medium	Medium	Start with 2–3 core locales. Use professional translators. Admin panel for quick fixes.
Payment disputes/fraud	High	Low	Stripe Radar for fraud detection. Clear refund policy (configurable). Escrow-style hold until lesson completion.
Tutor supply (cold start)	High	High	Pre-launch tutor recruitment campaign. Reduced commission for early tutors (configurable). Seed with test tutors for demo.

15. Success Metrics

MVP success is measured against these KPIs, all tracked via the configurable analytics provider.

Launch Metrics (First 30 Days)

Tutor sign-ups	50+ tutors with complete profiles	DB query: tutor profiles with completion > 80%

Learner sign-ups	200+ registered learners	Auth system user count
Trial bookings	50+ trial lessons booked	Booking table: type = TRIAL
Conversion: trial to paid	> 20% of trials convert to subscription	Subscription table after trial
Average page load (LCP)	< 2.5 seconds	Vercel Analytics / Lighthouse
Pages indexed by Google	> 80% of generated pages	Google Search Console
Uptime	> 99.5%	Monitoring service (configurable)

Growth Metrics (Months 2–3)

Organic traffic	1,000+ monthly visits from search	Analytics: organic channel
Repeat booking rate	> 40% of learners book again after trial	Booking table analysis
Tutor retention	> 70% of tutors active after 60 days	Last lesson date analysis
Platform revenue	Break-even on hosting/infrastructure costs	Stripe dashboard
Review completion	> 50% of completed lessons receive a review	Review vs. completed booking ratio
Average tutor rating	> 4.5 stars	Review aggregate query

Timeline Summary

Phase 1	1–3	Foundation & Infrastructure	App shell, auth, admin panel, config system, i18n routing
Phase 2	4–6	Core Marketplace	Tutor profiles, search, SEO landing pages, tutor dashboard
Phase 3	7–9	Booking & Payments	Booking flow, Stripe payments, messaging, video lessons, reviews
Phase 4	10–12	Polish & Launch	UI polish, performance, security, testing, production deployment

End of Document