🧭 PIKPAKGO

Technical Proposal for Next-Generation Travel Platform

Prepared by: Kaviska Dilshan
Date: 11/13/2025
Version: 1.0 (Comprehensive Edition)

Executive Summary

PikPakGo represents a paradigm shift in the travel booking industry. This comprehensive technical proposal outlines our approach to building a modern, AI-powered travel ecosystem that unifies Hotels, Vacation Rentals, and Experiences within a single, transparent platform.

Key Differentiators:

- **100% Price Transparency** – No hidden fees, ever
- **Unified Discovery** – One search for all accommodation types
- **AI-Powered Planning** – Smart trip recommendations
- **Rewards-First Philosophy** – Loyalty system built into every transaction
- **B2B-Ready Infrastructure** – White-label capability for travel agencies

Development Investment: 120,000 - 140,000 LKR (Without Including Hosting or Third-party Cost)
Timeline: 8 weeks to MVP launch
Technology: Modern, scalable, API-first architecture

---

1. Project Understanding & Vision

1.1 The Problem We're Solving

The travel booking industry is fragmented and opaque. Travelers face:

- **Hidden Fees:** Surprise charges at checkout erode trust
- **Fragmented Search:** Separate platforms for hotels vs. rentals vs. experiences
- **Poor Price Comparison:** Inconsistent pricing structures make comparison difficult
- **Limited Personalization:** Generic recommendations that don't understand traveler preferences
- **Weak Loyalty Programs:** Points systems that offer minimal value

1.2 The PikPakGo Solution

PikPakGo will revolutionize travel booking through:

Transparency First Every price shown includes all taxes, fees, and charges upfront. No surprises at checkout. This builds trust and reduces cart abandonment.

Unified Discovery Engine A single search interface that queries hotels, vacation rentals, and experiences simultaneously, presenting results in a unified, comparable format.

AI-Powered Personalization Machine learning algorithms that understand traveler preferences and create personalized trip itineraries, from accommodation to activities.

Meaningful Rewards A points system integrated into every transaction, redeemable immediately with transparent value (1 point = $0.01 USD).

B2B Enablement White-label infrastructure allowing travel agencies to offer PikPakGo's inventory under their own brand with customized markup structures.

1.3 Target Audience

Primary Users:

- **Leisure Travelers (65%):** Families, couples, solo travelers seeking vacation accommodations
- **Business Travelers (20%):** Corporate travelers needing flexible, reliable booking
- **Travel Agencies (15%):** B2B partners requiring white-label solutions

Geographic Focus (Phase 1):

- North America and Europe
- English language initially, with multi-language support in Phase 2

---

2. Market Analysis & Competitive Advantage

2.1 Market Opportunity

The global online travel booking market is valued at $817 billion (2024) and growing at 11.6% CAGR. Key trends:

- **Direct Booking Preference:** 43% of travelers prefer booking directly with providers
- **Mobile-First:** 72% of bookings initiated on mobile devices
- **Experience Economy:** 64% of millennials prioritize experiences over accommodations
- **Transparency Demand:** 89% of travelers abandon bookings due to unexpected fees

2.2 Competitive Landscape Analysis

Platform	Strengths	Weaknesses	PikPakGo Advantage
Booking.com	Massive inventory, brand trust	Hidden fees, complex UI	Transparent pricing, cleaner UX
Airbnb	Strong rentals, community	Limited hotels, service fees	Unified hotel+rental search
Expedia	Multi-service bundle	Cluttered interface	AI-powered simplicity
Vrbo	Vacation rental focus	No hotel integration	Complete accommodation spectrum

2.3 Our Competitive Advantages

1. **Total Price Transparency:** Only platform showing complete costs upfront
2. **AI Trip Planner:** Personalized itineraries combining lodging and experiences
3. **Unified Search:** Hotels + Rentals + Experiences in one result set
4. **Generous Rewards:** 3-5% back on every booking (vs. industry 1-2%)
5. **Modern Tech Stack:** Faster, more reliable, better mobile experience
6. **B2B-Ready:** White-label capability from day one

---

3. Technical Objectives & Success Criteria

3.1 Core Technical Objectives

Performance Objectives:

- Search results returned in <1000ms (95th percentile)
- Support 10,000 concurrent users at launch

Functional Objectives:

- Integrate minimum 3 hotel suppliers (e.g., Hotelbeds, Expedia API)
- Integrate minimum 2 vacation rental APIs (e.g., Guesty, Rentals United)
- Real-time availability checking (<5 seconds)
- Multi-currency support (USD, EUR, GBP initially)
- Mobile-responsive design (100% feature parity)

Security Objectives:

- PCI DSS Level 1 compliance via Stripe
- GDPR and CCPA compliant data handling
- Zero-knowledge architecture for payment data
- SOC 2 Type II certification path (post-launch)

---

4. Proposed Technology Stack

4.1 Frontend Technologies

Next.js 14 (App Router)

- **Why:** Server-side rendering for SEO, excellent performance, React ecosystem
- **Benefits:** Built-in routing, API routes, image optimization, automatic code splitting
- **Use Case:** Powers all user-facing pages and Progressive Web App shell

React 18

- **Why:** Component-based architecture, extensive ecosystem, concurrent rendering
- **Benefits:** Reusable UI components, efficient updates, strong developer community
- **Use Case:** All interactive UI elements and state management

TailwindCSS 3.x

- **Why:** Utility-first CSS, rapid development, consistent design system
- **Benefits:** Smaller CSS bundles, responsive design helpers, dark mode support
- **Use Case:** All styling and responsive layouts

TypeScript

- **Why:** Type safety, better IDE support, fewer runtime errors
- **Benefits:** Catch bugs at compile time, self-documenting code, refactoring confidence
- **Use Case:** All frontend and backend code

4.2 Backend Technologies

Node.js 20 LTS

- **Why:** JavaScript ecosystem, high concurrency, large package ecosystem
- **Benefits:** Non-blocking I/O, event-driven architecture, JSON native
- **Use Case:** API server, background jobs, supplier integrations

Express.js / Nest.js API Routes

- **Why:** Lightweight, flexible, well-documented routing framework
- **Benefits:** Middleware system, route handling, request parsing
- **Use Case:** RESTful API endpoints, webhook handlers

Prisma ORM

- **Why:** Type-safe database queries, automatic migrations, excellent DX
- **Benefits:** Schema-first design, query performance, relation handling
- **Use Case:** All database operations and schema management

4.3 Database & Storage

PostgreSQL 15

- **Why:** ACID compliance, complex queries, JSON support, proven reliability
- **Benefits:** Foreign keys, transactions, full-text search, extensions
- **Use Case:** Primary data store (users, bookings, listings, rewards)

Redis 7

- **Why:** In-memory speed, pub/sub capabilities, data structure support
- **Benefits:** <1ms latency, caching, session storage, rate limiting
- **Use Case:** API response caching, session management, job queues

AWS S3 / Cloudinary

- **Why:** Scalable object storage, CDN integration, image optimization
- **Benefits:** Automatic backups, image transformations, global distribution
- **Use Case:** Property images, user uploads, static assets

4.4 Third-Party Services

Search: Algolia

- **Purpose:** Ultra-fast, faceted search with geo-filtering
- **Features:** <100ms search, typo tolerance, filtering, ranking
- **Implementation:** Index all properties, sync via webhooks

Maps: Mapbox GL JS

- **Purpose:** Interactive map visualization with custom styling
- **Features:** Clustering, heat maps, custom markers, offline support
- **Implementation:** Property location display, area searches

Payments: Stripe

- **Purpose:** PCI-compliant payment processing
- **Features:** Multi-currency, saved cards, subscriptions, payouts
- **Implementation:** Checkout flow, reward redemption, host payouts

AI: OpenAI GPT-4

- **Purpose:** Natural language trip planning and recommendations
- **Features:** Context understanding, itinerary generation, Q&A
- **Implementation:** Trip planner, search assistance, content generation

Email: SendGrid / AWS SES

- **Purpose:** Transactional and marketing email delivery
- **Features:** Templates, tracking, scheduling, deliverability

- **Implementation:** Booking confirmations, password resets, marketing

Analytics: Google Analytics 4 + Mixpanel

- **Purpose:** User behavior tracking and conversion analytics
- **Features:** Event tracking, funnel analysis, cohort analysis
- **Implementation:** Page views, booking funnel, feature usage

4.5 DevOps & Infrastructure

Hosting: Vercel (Frontend) + AWS (Backend)

- **Vercel:** Next.js optimization, edge functions, automatic scaling
- **AWS:** EC2/ECS for API servers, RDS for PostgreSQL, ElastiCache for Redis

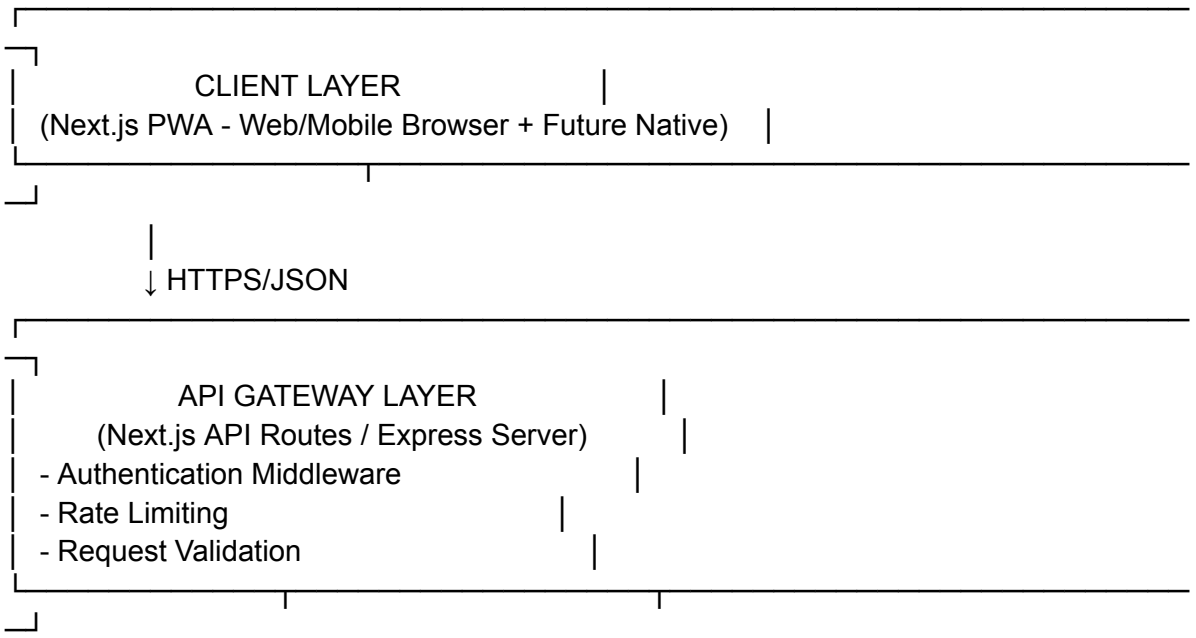
CI/CD: GitHub Actions

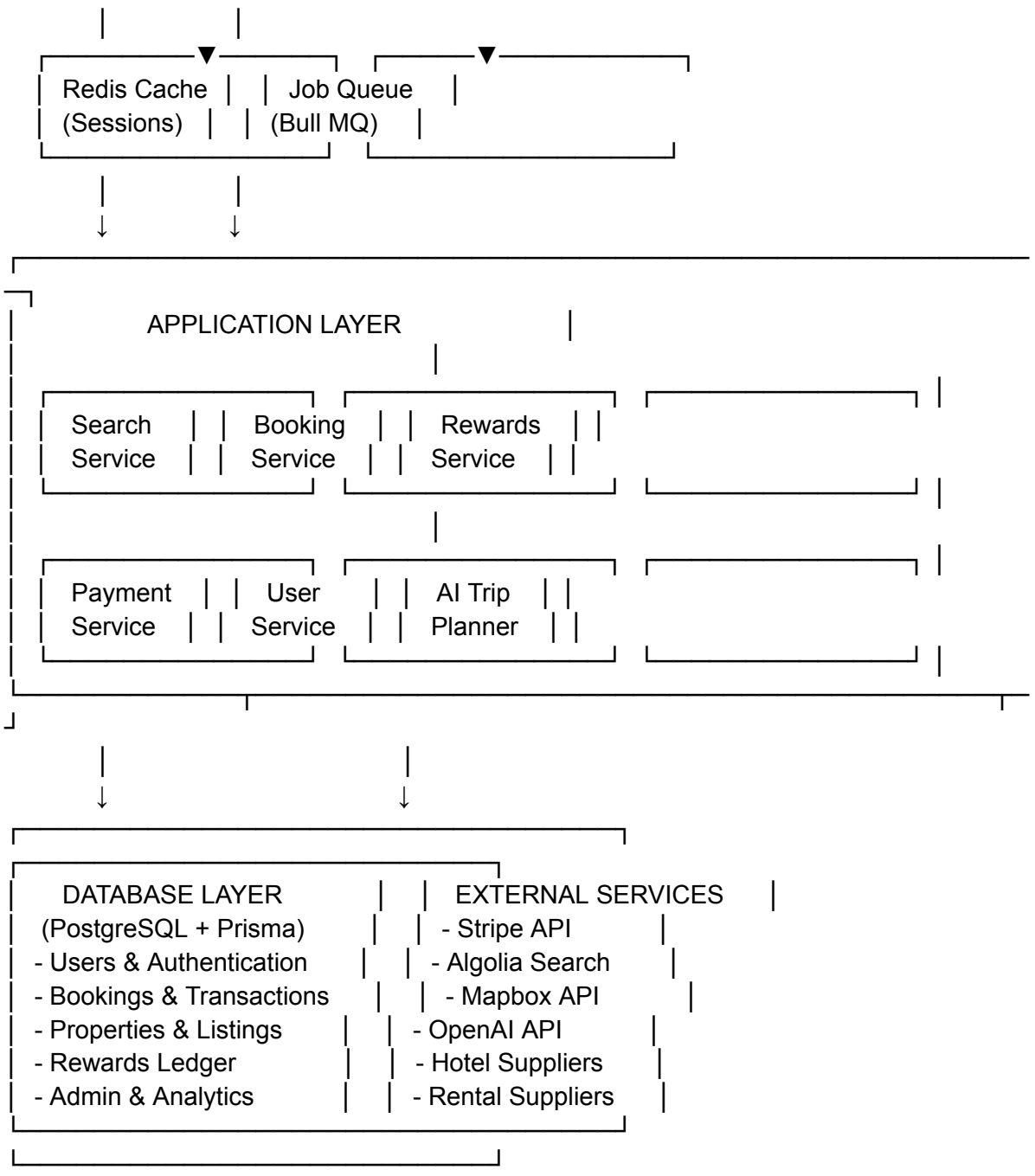
- **Purpose:** Automated testing and deployment pipelines
- **Workflow:** Lint → Test → Build → Deploy to staging → Deploy to production

---

5. System Architecture & Design Patterns

5.1 High-Level Architecture





5.2 Architecture Principles

1. API-First Design All functionality exposed via REST APIs, enabling future mobile apps and third-party integrations.

2. Separation of Concerns Each service handles a specific domain (Search, Booking, Payments) with clear interfaces.

3. Caching Strategy

- **L1 Cache:** In-memory (Node.js) for frequently accessed data

- **L2 Cache:** Redis for shared state across instances
- **L3 Cache:** CDN for static assets

4. Scalability Patterns

- Horizontal scaling via containerization (Docker)
- Database read replicas for analytics queries
- Asynchronous job processing for non-critical tasks

5. Fault Tolerance

- Circuit breaker pattern for external APIs
- Graceful degradation (show cached results if supplier API fails)
- Retry logic with exponential backoff

5.3 Data Flow Example: Hotel Search

1. User enters "Paris hotels, July 1-5"
2. Frontend sends request to /api/search
3. API layer:
   a. Validates input
   b. Checks Redis cache (key: "search:paris:july1-5")
   c. If cache miss, query supplier APIs in parallel
   d. Normalize results to common format
   e. Store in cache (TTL: 15 minutes)
   f. Index in Algolia for faceted search
4. Return results to frontend (JSON)
5. Frontend renders cards + map markers
6. User applies filters → client-side Algolia query

5.4 Database Schema Design (Simplified)

Key Tables:

- `users` - Authentication and profile data
- `properties` - Hotel and rental listings
- `bookings` - Reservation records
- `rewards` - User reward balances and transactions
- `payments` - Payment records (non-sensitive)
- `reviews` - User-generated content
- `admin_logs` - Audit trail

Relationships:

- One user → Many bookings
- One property → Many bookings
- One booking → One payment

- One user → One rewards account → Many reward transactions

---

6. Core Modules & Feature Specifications

6.1 Traveler Portal

6.1.1 Smart Search Interface

Features:

- **Destination Input:** Autocomplete with city, region, landmark support
- **Date Picker:** Calendar with flexible dates, weekend/weekday highlighting
- **Guest Configuration:** Adults, children, infants with age specifications
- **Property Type Filter:** Hotels, apartments, houses, unique stays
- **Advanced Filters:**
    - Price range (slider with histogram)
    - Amenities (WiFi, parking, pool, kitchen, etc.)
    - Guest ratings (4+ stars, etc.)
    - Cancellation policy (free cancellation, flexible, etc.)
    - Property features (beachfront, city center, etc.)

Technical Implementation:

```
// Search API Endpoint
POST /api/search
Request Body:
{
  "destination": "Paris, France",
  "checkIn": "2026-07-01",
  "checkOut": "2026-07-05",
  "guests": {
    "adults": 2,
    "children": 1,
    "infants": 0
  },
  "filters": {
    "propertyTypes": ["hotel", "apartment"],
    "priceRange": [50, 300],
    "amenities": ["wifi", "parking"],
    "minRating": 4.0
  }
}

Response:
{
  "results": [
```

```
    {
      "id": "prop_abc123",
      "name": "Hotel Example",
      "type": "hotel",
      "location": {
        "lat": 48.8566,
        "lon": 2.3522,
        "address": "123 Rue Example"
      },
      "pricing": {
        "basePrice": 150,
        "taxes": 22.50,
        "fees": 10,
        "total": 182.50,
        "currency": "USD",
        "breakdown": {
          "nightlyRate": 150,
          "nights": 4,
          "subtotal": 600,
          "totalWithTaxes": 730
        }
      },
      "amenities": ["wifi", "pool", "parking"],
      "rating": 4.5,
      "reviewCount": 324,
      "images": ["url1", "url2"]
    }
  ],
  "total": 156,
  "page": 1,
  "perPage": 20
}
```


6.1.2 Interactive Map View

Features:

- **Cluster Markers:** Groups of properties shown as numbered clusters
- **Price Overlay:** Hover shows starting price
- **Draw Search Area:** Users can draw custom search boundaries
- **Subway/Transit Layer:** Public transport overlay
- **Neighborhood Boundaries:** Visual district separations
- **Sync with List:** Clicking marker highlights list item

Technical Stack:

- Mapbox GL JS for rendering

- GeoJSON for property coordinates
- Supercluster for efficient marker clustering
- Custom marker icons for property types

6.1.3 Property Detail Page

Layout Components:

1. **Photo Gallery:** Full-screen carousel with thumbnails
2. **Key Information Card:**
    - Property name and type
    - Location (address + map pin)
    - Rating and review count
    - Host information
3. **Pricing Widget (Sticky):**
    - Nightly rate
    - Number of nights
    - Subtotal
    - Taxes (itemized)
    - Fees (itemized)
    - **Total Price (bold)**
    - Reward points earned
    - "Reserve" button
4. **Amenities Grid:** Icons + labels for all features
5. **Description:** Host-written content with "Read more"
6. **Location Map:** Embedded map with nearby attractions
7. **Reviews Section:** Filterable, sortable user reviews
8. **Cancellation Policy:** Clear terms and deadlines
9. **House Rules:** Check-in/out times, policies
10. **Similar Properties:** Recommendations carousel

6.1.4 Checkout Flow

Step 1: Trip Summary

- Selected dates and guest count
- Property thumbnail and name
- Price breakdown (read-only)

Step 2: Guest Information

- Full name (required)
- Email (required, verified)
- Phone number (required)
- Special requests (optional textarea)

Step 3: Payment

- Stripe embedded checkout

- Saved cards (if logged in)
- New card form
- Apply rewards toggle (shows discount)
- Final total

Step 4: Confirmation

- Booking reference number
- Email confirmation sent
- Add to calendar link
- Host contact information
- Directions to property

Security Measures:

- PCI compliance via Stripe
- No card data touches our servers
- 3D Secure authentication
- Fraud detection (Stripe Radar)

6.1.5 Trip Dashboard

Upcoming Trips:

- Card view with property photo
- Check-in countdown
- Quick actions: Cancel, modify, contact host
- Add to calendar button

Past Trips:

- Booking history
- Leave review prompt
- Rebook option
- Download receipt

Saved Properties:

- Wishlist with price tracking
- Price drop notifications
- Quick booking from saves

6.1.6 Rewards Center

Rewards Balance:

- Current points balance (prominent)
- Points value in USD (1 point = $0.01)
- Tier status (Bronze, Silver, Gold, Platinum)

Transaction History:

- Earned from bookings
- Redeemed on bookings
- Expiration dates
- Bonus points events

Redemption:

- Apply at checkout
- Minimum redemption (500 points = $5)
- Cannot exceed 50% of booking value

Earning Rules:

- Hotels: 5% back
- Rentals: 3% back
- Experiences: 4% back
- Bonus: Invite friends (500 points per signup)

---

7. User Experience & Interface Design

7.1 Design Philosophy

Principles:

1. **Clarity Over Complexity:** Every element serves a purpose
2. **Speed is a Feature:** Optimize perceived and actual performance
3. **Mobile-First:** Design for small screens, enhance for large
4. **Accessible by Default:** WCAG 2.1 AA compliance minimum
5. **Consistent Patterns:** Reusable components across all pages

7.2 Visual Design System

Color Palette:

- Primary: Blue (#0066FF) - Trust, reliability
- Secondary: Orange (#FF6B35) - Energy, rewards

- Success: Green (#00C853)
- Warning: Amber (#FFC107)
- Error: Red (#F44336)
- Neutral: Grays (#F5F5F5 to #212121)

Typography:

- Headings: Inter (Sans-serif)
- Body: Inter (Regular, Medium)
- Monospace: JetBrains Mono (for codes)

Spacing System:

- Base unit: 8px
- Scale: 8, 16, 24, 32, 48, 64, 96px

Component Library:

- Buttons (Primary, Secondary, Tertiary, Ghost)
- Form inputs (Text, Select, Date, Checkbox, Radio)
- Cards (Property, Feature, Info)
- Modals and drawers
- Navigation (Header, Footer, Breadcrumbs)
- Alerts and notifications

7.3 Responsive Breakpoints

- **Mobile:** 320px - 767px (Single column)
- **Tablet:** 768px - 1023px (Flexible grid)
- **Desktop:** 1024px - 1439px (Multi-column)
- **Wide:** 1440px+ (Max-width container)

7.4 Accessibility Features

- **Keyboard Navigation:** Full site navigable without mouse
- **Screen Reader Support:** Semantic HTML, ARIA labels
- **Focus Indicators:** Clear visual focus states
- **Color Contrast:** Minimum 4.5:1 ratio for text
- **Alt Text:** All images have descriptive alternatives
- **Form Labels:** Associated labels for all inputs
- **Error Messages:** Clear, actionable error descriptions

---

8. API Integration Strategy

8.1 Supplier API Architecture

Adapter Pattern: Each supplier has a dedicated adapter that:

1. Translates our internal request format to supplier format
2. Handles authentication (API keys, OAuth)
3. Normalizes responses to our standard format
4. Implements retry logic and error handling
5. Caches results appropriately

Example: Hotel Supplier Adapter

```
interface HotelSearchParams {
  destination: string;
  checkIn: string;
  checkOut: string;
  guests: GuestConfig;
}

interface NormalizedProperty {
  id: string;
  name: string;
  type: 'hotel' | 'apartment' | 'house';
  location: Location;
  pricing: PricingBreakdown;
  amenities: string[];
  images: string[];
  rating: number;
  reviewCount: number;
}

class HotelbedsAdapter {
  async search(params: HotelSearchParams): Promise<NormalizedProperty[]> {
    // 1. Transform to Hotelbeds format
    const hotelbedsRequest = this.transformRequest(params);

    // 2. Make API call
    const response = await this.client.post('/hotels', hotelbedsRequest);

    // 3. Normalize response
    return response.data.hotels.map(this.normalizeProperty);
  }

  private normalizeProperty(raw: any): NormalizedProperty {
    return {
      id: `hotelbeds_${raw.code}`,
      name: raw.name,
      type: 'hotel',
      location: {
        lat: raw.latitude,
```

```
        lon: raw.longitude,
        address: raw.address
      },
      pricing: this.calculateTotalPrice(raw.rooms[0]),
      amenities: this.mapAmenities(raw.facilities),
      images: raw.images.map(img => img.url),
      rating: raw.categoryCode / 2, // Convert 5-star to 5.0
      reviewCount: 0 // Hotelbeds doesn't provide this
    };
  }
}
```

8.2 Planned Supplier Integrations

Hotels:

1. **Hotelbeds** - 180,000+ hotels worldwide
2. **Expedia RAPID API** - 700,000+ properties
3. **Amadeus** - Global distribution system

Vacation Rentals:

1. **Guesty** - Property management system integration
2. **Rentals United** - 20+ channel manager connections
3. **Direct Host API** - For hosts managing via our platform

Experiences:

1. **GetYourGuide** - Tours and activities
2. **Viator** - 300,000+ experiences worldwide

8.3 Real-Time Availability Checking

Challenge: Suppliers don't always have real-time inventory

Solution:

1. **Search Phase:** Show cached availability (5-15 min old)
2. **Pre-Booking:** Real-time check before showing pricing
3. **Booking Phase:** Final availability verification
4. **Inventory Webhook:** Update cache when suppliers notify changes

Flow:

User Searches → Check Cache (Fast)
User Clicks Property → Real-time API Call (Accurate)
User Clicks Book → Final Verification → Hold Inventory

8.4 Error Handling & Fallbacks

Supplier API Down:

- Show cached results with disclaimer
- Hide "Book Now" for affected properties
- Log incident for monitoring

Partial Failure:

- If 1 of 3 suppliers fails, show results from 2
- Display "Some results may be missing" notice

Timeout Handling:

- Set 5-second timeout per supplier
- Cancel slow requests, proceed with fast ones

---

9. AI Trip Planner Implementation

9.1 Feature Overview

The AI Trip Planner transforms natural language travel desires into structured, bookable itineraries.

User Input: "I want a romantic weekend in Paris for 2 adults, budget $2000, interested in art and food"

AI Output:

- Recommended neighborhood (Le Marais)
- 3 hotel options matching budget and preferences
- Daily itinerary with activities (Louvre, cooking class, Seine cruise)
- Restaurant recommendations
- Transportation tips

9.2 Technical Architecture

Components:

1. **Input Parser:** Extracts structured data from natural language
2. **Recommendation Engine:** Queries our API for matching properties
3. **Itinerary Generator:** Creates day-by-day plans
4. **Refinement Loop:** User can ask for adjustments

Implementation:

```
async function generateTripPlan(userInput: string) {
  // Step 1: Extract trip parameters using GPT-4
  const extractedParams = await openai.chat.completions.create({
    model: "gpt-4",
    messages: [
      {
        role: "system",
        content: `Extract trip details from user input. Return JSON:
        {
          "destination": "city, country",
          "duration": "number of nights",
          "travelers": {"adults": 2, "children": 0},
          "budget": "total in USD",
          "interests": ["tag1", "tag2"],
          "tripType": "romantic|family|adventure|business"
        }`
      },
      { role: "user", content: userInput }
    ],
    response_format: { type: "json_object" }
  });

  const tripParams = JSON.parse(extractedParams.choices[0].message.content);

  // Step 2: Search our database for matching properties
  const properties = await searchProperties({
    destination: tripParams.destination,
    checkIn: calculateCheckIn(tripParams.duration),
    checkOut: calculateCheckOut(tripParams.duration),
    guests: tripParams.travelers,
    budget: tripParams.budget
  });

  // Step 3: Generate itinerary
  const itinerary = await openai.chat.completions.create({
    model: "gpt-4",
    messages: [
      {
        role: "system",
        content: `You are a travel expert. Create a detailed daily itinerary
        for ${tripParams.destination} focusing on ${tripParams.interests.join(', ')}.
        Include morning, afternoon, and evening activities. Suggest specific
        restaurants, attractions, and experiences.`
      },
      {
        role: "user",
        content: `Create a ${tripParams.duration}-night itinerary for a
        ${tripParams.tripType} trip. Here are available accommodations:
```

```
        ${JSON.stringify(properties.slice(0, 5))}`
      }
    ]
  });

  return {
    params: tripParams,
    accommodations: properties.slice(0, 5),
    itinerary: itinerary.choices[0].message.content,
    estimatedCost: calculateTotalCost(properties[0], tripParams.duration)
  };
}
```

9.3 User Interface

Layout:

1. **Chat Interface:** Natural conversation flow
2. **Itinerary View:** Day-by-day cards with activities
3. **Property Carousel:** Top 3-5 recommended accommodations
4. **Budget Tracker:** Shows estimated vs. budget
5. **Book Button:** One-click to book entire trip

Iterative Refinement:

- "Make it cheaper" → Adjusts property selection
- "Add more museums" → Regenerates itinerary
- "Change hotel to Montmartre" → Updates location

---

10. Budget Breakdown (in LKR)

Category	Details	Estimated Cost (LKR)
System Design & Planning	Architecture diagrams, flow design	10,000

UI/UX Design	Component library, responsive layout	20,000
Frontend Development	Next.js, Mapbox, Algolia integration	40,000
Backend Development	API routes, booking, rewards, suppliers	40,000
AI Trip Planner	OpenAI integration & itinerary builder	20,000
QA & Optimization	Testing, SEO, performance	10,000

Total 140,000 LKR

11. Closing Statement

Our engineering team is confident that PikPakGo will set a new standard in the travel booking space — one that is transparent, scalable, and powered by a modern technical foundation.

We look forward to collaborating with your team to bring this platform to life.