

# Can AI Build Your Next API Integration?

## Lessons from Pandium's Experimentation with LLMs

Presented by Shon Urbas



# Agenda

1. What is Pandium
2. Our AI Journey: The Early Days
3. The Smallest Problem: JSON Key Path Mapping
4. Building the Foundation
5. A Rekindling: AI in Production July, 2024
6. The Revival: Q4 2024 Success
7. Live Demo
8. Key Lessons Learned
9. The Future of AI in Integration Development
10. Closing: The Real AI Revolution



# Shon Urbas

*[linkedin.com/in/shon-urbas/](https://www.linkedin.com/in/shon-urbas/)*

- CTO & Co-Founder of **Pandium**
- Former Platform Engineering Manager at Handshake (Acquired by Shopify)

# What is Pandium?

# What is Pandium?

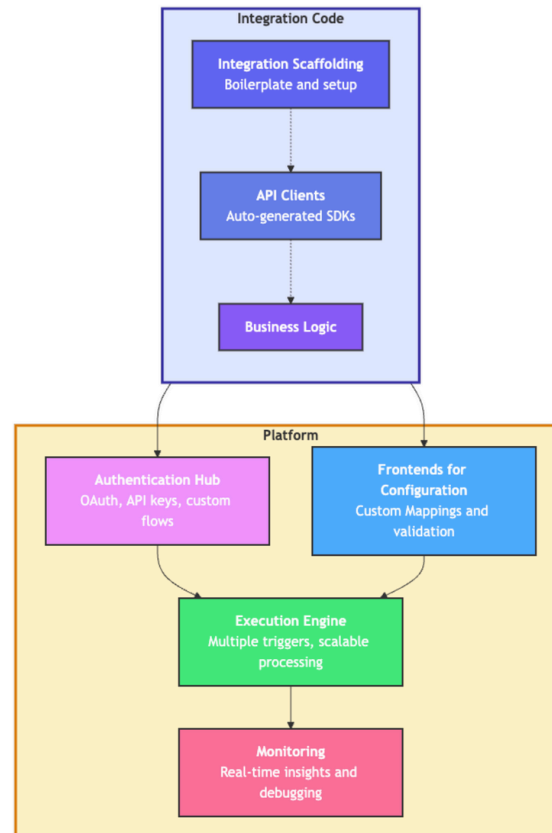
## The Integration Challenge

Our typical engineer users face:

- **Time sink:** 60-80% of development time on integrations, not core features
- **Complexity creep:** Each integration touches authentication, data mapping, error handling, rate limiting
- **Maintenance burden:** APIs change, breaking existing integrations

# What is Pandium?

## Pandium's Solution



# **Our AI Journey: The Early Days**

# Our AI Journey: The Early Days

## Phase 1: The "Whole Shabang" Approach (Birth of ChatGPT Era)

### Late 2022: Ambitious but Naive

- **Goal:** Generate entire integrations from natural language prompts
- **Reality:** Models produced syntactically correct but functionally broken code

**Example:** *"Connect Salesforce to Slack"* → 200 lines of code that didn't actually work, forget about at scale



# Our AI Journey: The Early Days

## The Expert Partnership Experiment

### Mid-2023: Bringing in the Consultants

- Partnered with AI consulting firms
- Promise: "We'll make this production-ready"
- **Results:**
  - 3x longer timelines than quoted
  - Couldn't even approach doing the whole thing
  - Drove us to think smaller so that's good, but even failed at that
  - **Key insight:** Hype was ahead of practicality

# Our AI Journey: The Early Days

## The Pivot: From Sky-High to Surgical

**Discovery:** Most integration work doesn't need AI

- **90%** can be handled deterministically
- **10%** requires intelligent transformation
- **Focus shift:** Target the 90% that we can, and give the AI stuff more time.

# **The Smallest Problem: JSON Key Path Mapping**

# The Smallest Problem: JSON Key Path Mapping

## The "Aha!" Moment

**Problem:** Mapping arbitrary JSON structures

```
// Source API
{
  "customer_info": {
    "full_name": "John Doe",
    "contact": {"email": "john@example.com"}
  }
}
// Target API expects
{
  "name": "John Doe",
  "email_address": "john@example.com"
}
```

```
$.customer_info.full_name -> $.name
$.customer_info.contact.email -> $.email_address
```

# Building the Foundation

# Building the Foundation

## What We Automated Deterministically

**Q4 2024: Focus on the 90%**

### API Client Generation

- Parse OpenAPI specs
- Generate type-safe SDKs
- Handle authentication patterns

### Project Scaffolding

- Standard project structure
- Configuration templates
- Testing frameworks

### Documentation

- Auto-generated API docs
- Integration guides
- Specification

# **A Rekindling: AI in Production July, 2024**

# A Rekindling: AI in Production July, 2024

**Fresh off our JSON mapping struggles, I was convinced AI was all hype**

- **My brilliant idea:** Submit talk proposal "AI Integration Hype vs Reality: Why It Doesn't Work"
- **Conference committee's response:** ❌ REJECTED ❌

**What I Learned:**

- Peoples enthusiasm in exploring the space was infectious
- Not all the things shown were production ready, but that wasn't really the point
- Lots of clever design patterns.
- Early glimpses of what we'd later call "agentic programming"

**Key insight:** Reinforced the thought that by working on the deterministic parts first, we could build a foundation for a successful AI model



# **The Revival: Q4 2024 Success**

# The Revival: Q4 2024 Success

## What Changed?

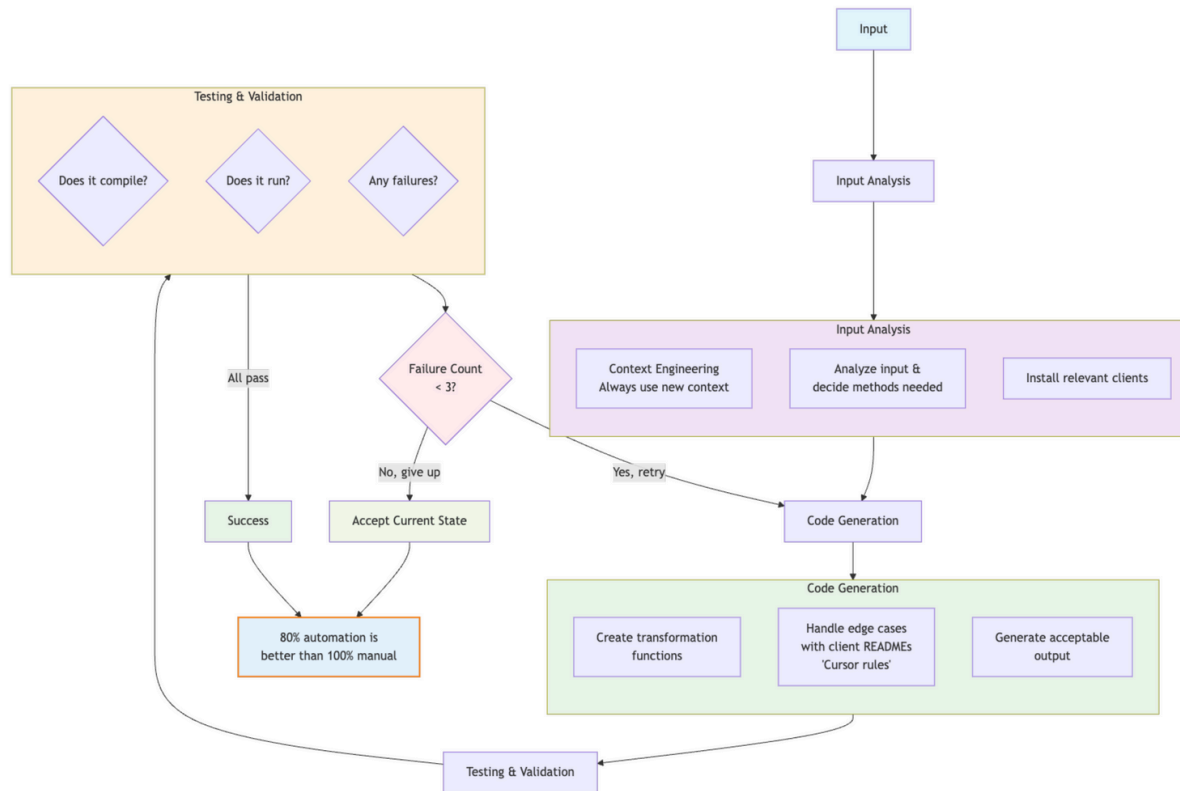
### Technology maturation:

- Much larger contexts, but still has constraints.
- “Context Engineering:” i.e. making sure only needed info is in prompt
- More reliable code generation

### Our refined “Agentic” approach:

- Narrow AI scope to mapping functions only (a huge upgrade from just JSON PATHs)
- Robust testing and validation
- Iterative failure handling

# The Revival: Q4 2024 Success



# The Revival: Q4 2024 Success

## The "Give Up Early" Principle

**Key insight:** Not everything needs to be perfect. i.e. We only try 3 times, for a successful flow before moving on.

- **Deterministic parts:** Keep them deterministic
- **AI parts:** Allow controlled failure
- **Success criteria:** 90% automation is better than 100% manual

# Example

# Example: Inputs

The screenshot shows a 'Generate Starter Integration' dialog box. At the top, there are two toggle switches: 'Include API Clients' (which is turned on) and 'Include CodeGen' (which is turned off). Below these is the 'Configure Flows:' section. It contains three rows of flow configuration. Each row has a dropdown menu on the left labeled 'Your App Object', a direction arrow in the middle, and another dropdown menu on the right labeled 'Service Object'. The first row shows 'Transactions' connected to 'Orders' with a left-pointing arrow. The second row shows 'Inventory' connected to 'Inventory Levels' with a left-pointing arrow. The third row shows 'Shipment' connected to 'Fulfillments' with a right-pointing arrow. Each row also has a trash icon to its right. At the bottom left of the dialog is a '+ ADD FLOW' button. At the bottom right are three buttons: 'BACK', 'CANCEL', and 'START'.

**The “Prompt” for user:**

**Connector A: ActiveCampaign**

**Resource A: Sales**

**Connector B: MindBody**

**Resource B: Order**

# Example: Method Defs

## Example of a “Product”

```
+ //
+ "listProducts": {
+   "requestType": "ProductsApiListProductsRequest",
+   "responseType": "Promise<AsyncGenerator<ShipbobProductsApiViewModelsPublicProductViewModel, any, unknown>>",
+   "supportingTypes": {
+     "ProductsApiListProductsRequest": "interface ProductsApiListProductsRequest { readonly referenceIds?: Array<string>; readonly page?: number; readonly limit?: number; readonly ids?: Array<number>; readonly search?: string; readonly
+     activeStatus?: ShipbobProductsCommonModelsProductActiveStatus; readonly bundleStatus?: ShipbobProductsCommonModelsProductBundleStatus; readonly shipbobChannelId?: number; }",
+     "ShipbobProductsCommonModelsProductActiveStatus": "const ShipbobProductsCommonModelsProductActiveStatus { readonly Any: \"Any\"; readonly Active: \"Active\"; readonly Inactive: \"Inactive\"; }\n\nexport type
+     ShipbobProductsCommonModelsProductActiveStatus = typeof ShipbobProductsCommonModelsProductActiveStatus[keyof typeof ShipbobProductsCommonModelsProductActiveStatus];",
+     "ShipbobProductsCommonModelsProductBundleStatus": "const ShipbobProductsCommonModelsProductBundleStatus { readonly Any: \"Any\"; readonly Bundle: \"Bundle\"; readonly NotBundle: \"NotBundle\"; }\n\nexport type
+     ShipbobProductsCommonModelsProductBundleStatus = typeof ShipbobProductsCommonModelsProductBundleStatus[keyof typeof ShipbobProductsCommonModelsProductBundleStatus];",
+     "ShipbobProductsApiViewModelsPublicProductViewModel": "interface ShipbobProductsApiViewModelsPublicProductViewModel { id?: number; reference_id?: string | null; bundle_root_information?:
+     ShipbobProductsApiViewModelsPublicBundleRootInformationViewModel; created_date?: string; channel?: ShipbobProductsApiViewModelsPublicChannelViewModel; sku?: string | null; name?: string | null; barcode?: string | null; gtin?: string | null; upc?: string
+     | null; unit_price?: number | null; total_fulfillable_quantity?: number; total_onhand_quantity?: number; total_committed_quantity?: number; fulfillable_inventory_items?: Array<ShipbobProductsApiViewModelsPublicInventoryItemViewModel> | null;
+     fulfillable_quantity_by_fulfillment_center?: Array<ShipbobProductsApiViewModelsPublicFulfillmentCenterQuantityViewModel> | null; }",
+     "ShipbobProductsApiViewModelsPublicBundleRootInformationViewModel": "interface ShipbobProductsApiViewModelsPublicBundleRootInformationViewModel { id?: number; name?: string | null; }",
+     "ShipbobProductsApiViewModelsPublicChannelViewModel": "interface ShipbobProductsApiViewModelsPublicChannelViewModel { id?: number; name?: string | null; }",
+     "ShipbobProductsApiViewModelsPublicFulfillmentCenterQuantityViewModel": "interface ShipbobProductsApiViewModelsPublicFulfillmentCenterQuantityViewModel { id?: number; name?: string | null; fulfillable_quantity?: number; onhand_quantity?:
+     number; committed_quantity?: number; }",
+     "ShipbobProductsApiViewModelsPublicInventoryItemViewModel": "interface ShipbobProductsApiViewModelsPublicInventoryItemViewModel { id?: number; name?: string | null; quantity?: number; }"
+   }
+ },
```

## Example: All + Imports

[illegible]

## All the Code Generated

```
import MindbodyClient, { SaleApiGetSalesRequest } from '@pandium/mindbody-client';
import ActivecampaignClient, {
  DefaultApiCreateOrderRequest,
  CreateOrderRequest,
  CreateOrderRequestEcomOrder,
  CreateOrderRequestEcomOrderOrderProducts
} from '@pandium/activecampaign-client';
import { Config } from '../lib.js';
```

Imports



# Example: Mapper

```
1  ✓ const saleToOrder = (sale: any, connectionId: string, customerId: string): DefaultApiCreateOrderRequest | undefined => {
2    try {
3      const orderProducts: CreateOrderRequestEcomOrderOrderProducts[] = sale.Items?.map((item: any) => ({
4        name: item.Description || 'Unknown Product',
5        price: item.UnitPrice || 0,
6        quantity: item.Quantity || 1,
7        externalid: item.Id?.toString() || `item-${Date.now()}`,
8        description: item.Description || '',
9        sku: item.Id?.toString() || ''
10     })) || [];
11
12     const ecomOrder: CreateOrderRequestEcomOrder = {
13       externalid: sale.Id?.toString() || `sale-${Date.now()}`,
14       source: 'Mindbody',
15       email: sale.ClientEmail || '',
16       orderProducts: orderProducts,
17       totalPrice: sale.TotalAmount || 0,
18       taxAmount: sale.TaxAmount || 0,
19       discountAmount: 0, // Mindbody doesn't provide discount directly
20       currency: sale.PaymentMethod?.Currency || 'USD',
21       connectionid: parseInt(connectionId),
22       customerid: parseInt(customerId),
23       orderNumber: sale.Id?.toString() || '',
24       externalCreatedDate: sale.SaleDateTime || new Date().toISOString()
25     };
26
27     const orderRequest: CreateOrderRequest = {
28       ecomOrder: ecomOrder
29     };
30
31     return {
32       createOrderRequest: orderRequest
33     };
34   } catch (error) {
35     console.error(`There was an error in transforming Mindbody sale ${sale.Id || 'unknown'} into Activecampaign order payload: ${error}`);
36   }
37   return undefined;
38 }
```

# Example: The loop

```
48
49 export const syncMindbodySalestoActivecampaignOrders = async(
50   mindbody: MindbodyClient,
51   activecampaign: ActivecampaignClient,
52   config: Config
53 ) => {
54   const connectionId = config.activecampaign_connection_id;
55   const customerId = config.activecampaign_customer_id;
56
57   if(!connectionId) {
58     console.error('Cannot sync Mindbody sales to Activecampaign orders: missing activecampaign_connection_id config');
59     console.error('----- STOPPING SYNC OF MINDBODY SALES -----');
60     return;
61   }
62
63   if(!customerId) {
64     console.error('Cannot sync Mindbody sales to Activecampaign orders: missing activecampaign_customer_id config');
65     console.error('----- STOPPING SYNC OF MINDBODY SALES -----');
66     return;
67   }
68
69   console.error(
70     '----- STARTING TO SYNC MINDBODY SALES TO ACTIVECAMPAIGN ORDERS -----'
71   );
72
73   const request: SaleApiGetSalesRequest = {};
74   const salesGenerator = await mindbody.getSales(request);
75
76   for await (const sale of salesGenerator) {
77     const saleData = sale as any; // Type assertion to handle the 'never' type
78     const orderPayload = saleToOrder(saleData, connectionId, customerId);
79     if (!orderPayload) continue;
80
81     try {
82       await activecampaign.createOrder(orderPayload);
83       console.error('Successfully created an Activecampaign order for Mindbody sale ${saleData.Id || 'unknown'}');
84     } catch (error) {
85       console.error('There was an error in creating Activecampaign order based on Mindbody sale ${saleData.Id || 'unknown'}: ${error}');
86     }
87   }
88
89   console.error(
90     '----- COMPLETED SYNCING MINDBODY SALES TO ACTIVECAMPAIGN ORDERS -----'
91   );
92   };
```

## Example: Result

**99.999% deterministic code**

User given (including API clients) ~200k of code to start integration

With 96 lines trully generated by LLM. Rest is all normal Code Gen

# Key Lessons Learned

# Key Lessons Learned

## Technical Lessons

- Scope Limitation: AI excels at narrow, well-defined tasks
- Hybrid Approach: Combine AI with deterministic systems
- Validation is Critical: AI output must be thoroughly tested
- Failure Recovery: Build systems that handle AI mistakes gracefully

# Key Lessons Learned

## Business Lessons

- Solution-First Thinking: Find problems that fit AI capabilities
- Avoid "AI for AI's Sake": Value must be demonstrable
- Developer Experience: AI should enhance, not replace, developer control
- Iterative Development: Start small, prove value, scale up

# The Future of AI in Integration Development

# The Future of AI in Integration Development

## What's Next for Pandium

### Short-term (next 3 months):

- Multiple Objects to Single Objects Flows
- Event Based
- CLI Access

### Medium-term (3+ months):

- Forking and Build (i.e. I have a Xero Integration... now make a QBO one.)
- AI Assisted API Client generation
- Automated performance tuning



# The Future of AI in Integration Development

## Industry Predictions

### What will happen:

- AI will augment, not replace, developers
- Hybrid human-AI workflows will become standard
- Domain-specific AI tools will outperform general-purpose ones (just like real life)

### What won't happen:

- Complete automation of complex integrations
- AI replacing the need for technical expertise
- One-size-fits-all AI solutions

# Closing: The Real AI Revolution

# Closing: The Real AI Revolution

## The Bottom Line

### **AI's true value in integration development:**

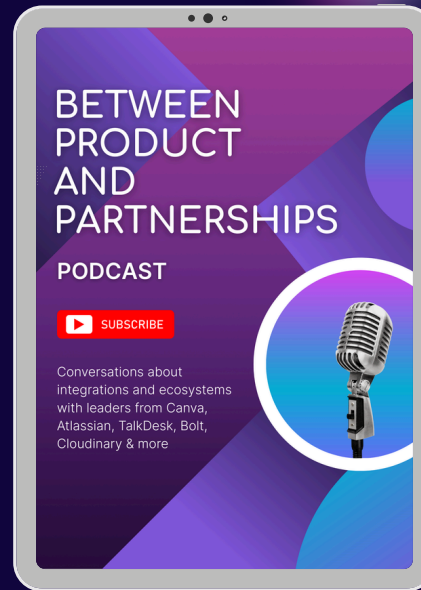
- Not: Building entire systems from scratch
- But: Accelerating the tedious, repetitive work
- Result: Developers focus on business logic

# Questions?



**Shon Urbas**

[Linkedin.com/in/shon-urbas/](https://www.linkedin.com/in/shon-urbas/)



**Subscribe to the  
podcast to hear  
conversations about  
integrations and  
ecosystems**

**Thank You! Any Questions?**