
ALPS + AI = API
Mike Amundsen
@mamund


Mike Amundsen
@mamund

Let's create
a functional API prototype
using only an API story
and
some GenAI prompts.

Overview

- Writing API Stories
- Building the ALPS Description
- Creating the NodeJS API

But First …


ALPS

Application-Level Profile Semantics (ALPS)

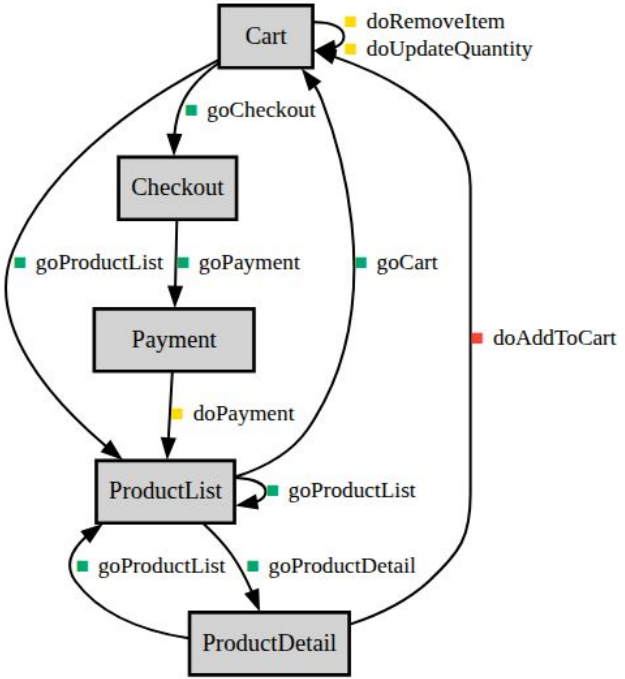
- Goal
  - A Format for Clarifying Application-Level Meaning and Structure of Interfaces
- History
  - First experiment was in at 2011 RESTFest
  - Based on XML Meta Data Profiles (XMDP) from 2003 by Tantek Çelik
- Tooling
  - ALPS online editor
  - ASD (app-state-diagram) CLI
  - LLMs Prompt Library
  - ALPS OpenAI Assistant

app-state-diagram aaa XML ✅ Download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
Welcome to Alps Editor! Let's make API design fun and effective.

Quick tips:
- Press Ctrl + Space to show snippets for auto-completion (suggested terms are from Schema.org)
- To start from scratch, delete all content and press Ctrl + Space, then select "Skeleton"
- Drag and drop an ALPS file (JSON, XML, or HTML) into the editor to open it
  (For HTML files, the ALPS profile contained within will be extracted)
- Hit Ctrl + S to download your work anytime

ALPS bridges vision and implementation, creating APIs that speak business and tech fluently.

Learn more about ALPS:
- app-state-diagram: https://www.app-state-diagram.com/
- Official ALPS website: http://alps.io/

Happy modeling! Remember, solid semantics supports the long-term evolution of your APIs. :)
-->
<alps
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="https://alps-io.github.io/schemas/alps.xsd">
    <title>ALPS Online Shopping</title>
    <doc>This is a sample ALPS profile demonstrating the semantic descriptors
        and operations for a basic e-commerce system. It includes product listing,
        shopping cart management, and checkout process, serving as an educational
        example for ALPS implementation in online shopping contexts.</doc>

    <!-- Ontology -->
    <descriptor id="id" def="https://schema.org/identifier" title="identifier"/>
    <descriptor id="name" def="https://schema.org/name" title="name"/>
    <descriptor id="description" def="https://schema.org/description" title="description"/>
    <descriptor id="price" def="https://schema.org/price" title="price"/>
    <descriptor id="quantity" def="https://schema.org/Quantity" title="quantity"/>
    <descriptor id="email" def="https://schema.org/email" title="email"/>
    <descriptor id="address" def="https://schema.org/address" title="address"/>

    <!-- Taxonomy -->
    <descriptor id="ProductList" def="https://schema.org/ItemList" title="Product List" tag="collecti
        <descriptor href="#id"/>
        <descriptor href="#name"/>
        <descriptor href="#description"/>
        <descriptor href="#goProductDetail"/>
        <descriptor href="#goCart"/>
        <descriptor href="#goProductList"/>
    </descriptor>

    <descriptor id="ProductDetail" def="https://schema.org/Product" title="Product Detail" tag="item'
        <descriptor href="#id"/>
        <descriptor href="#name"/>
        <descriptor href="#description"/>
        <descriptor href="#price"/>
        <descriptor href="#goProductList"/>
        <descriptor href="#doAddToCart"/>
    </descriptor>

    <descriptor id="Cart" def="https://schema.org/Cart" title="Shopping Cart" tag="collection">
        <descriptor href="#id"/>
        <descriptor href="#goProductList"/>
        <descriptor href="#goCheckout"/>
        <descriptor href="#doUpdateQuantity"/>
        <descriptor href="#doRemoveItem"/>
```

ALPS Online Shopping

This is a sample ALPS profile demonstrating the semantic descriptors and operations for a basic e-commerce system. It includes product listing, shopping cart management, and checkout process, serving as an educational example for ALPS implementation in online shopping contexts.



app-state-diagram aaa XML ✅ Download

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Welcome to Alps Editor! Let's make API design fun and effective.

Quick tips:
- Press Ctrl + Space to show snippets for auto-completion (suggested terms are from Schema.org)
- To start from scratch, delete all content and press Ctrl + Space, then select "Skeleton"
- Drag and drop an ALPS file (JSON, XML, or HTML) into the editor to open it
```

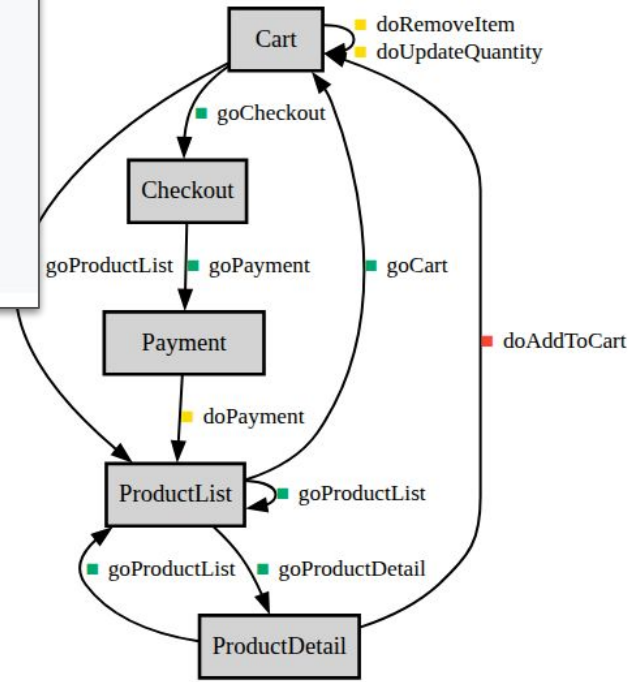
Command Line Options

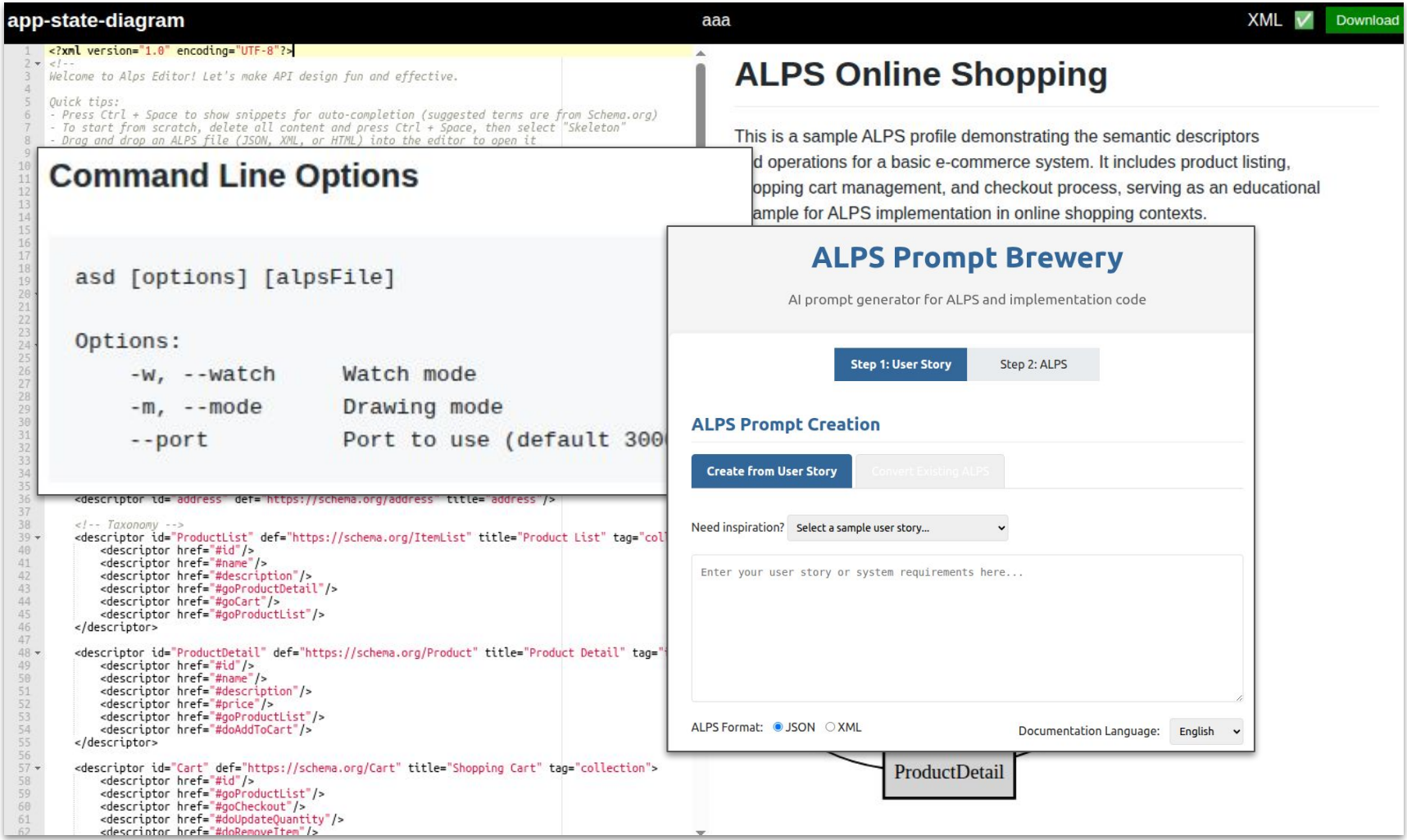
```
asd [options] [alpsFile]

Options:
     -w, --watch       Watch mode
     -m, --mode        Drawing mode
     --port            Port to use (default 3000)
```

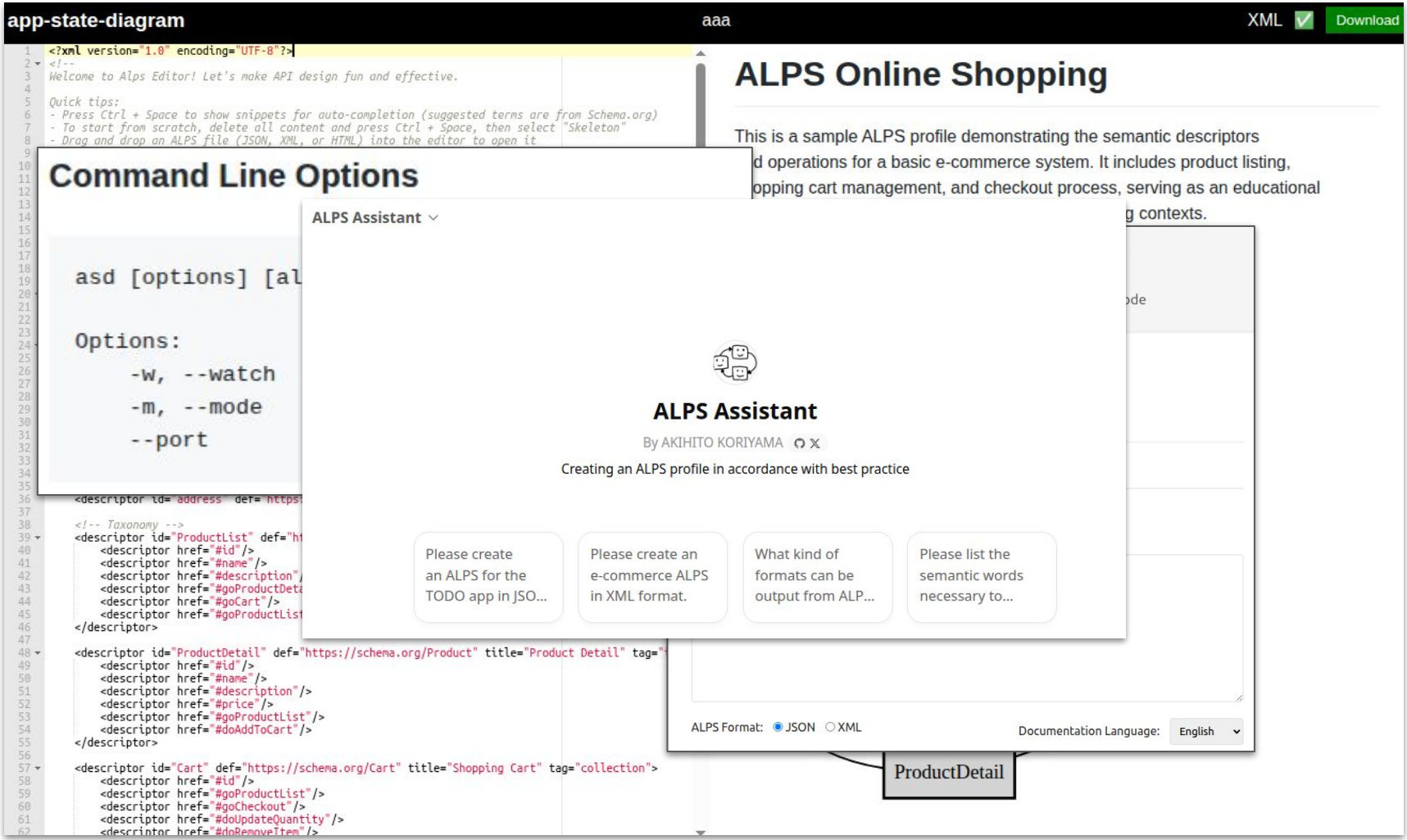
```
    <descriptor id="address" def="https://schema.org/address" title="address" />

    <!-- Taxonomy -->
    <descriptor id="ProductList" def="https://schema.org/ItemList" title="Product List" tag="collecti
        <descriptor href="#id" />
        <descriptor href="#name" />
        <descriptor href="#description" />
        <descriptor href="#goProductDetail" />
        <descriptor href="#goCart" />
        <descriptor href="#goProductList" />
    </descriptor>

    <descriptor id="ProductDetail" def="https://schema.org/Product" title="Product Detail" tag="item"
        <descriptor href="#id" />
        <descriptor href="#name" />
        <descriptor href="#description" />
        <descriptor href="#price" />
        <descriptor href="#goProductList" />
        <descriptor href="#doAddToCart" />
    </descriptor>

    <descriptor id="Cart" def="https://schema.org/Cart" title="Shopping Cart" tag="collection">
        <descriptor href="#id" />
        <descriptor href="#goProductList" />
        <descriptor href="#goCheckout" />
        <descriptor href="#doUpdateQuantity" />
        <descriptor href="#doRemoveItem" />
```

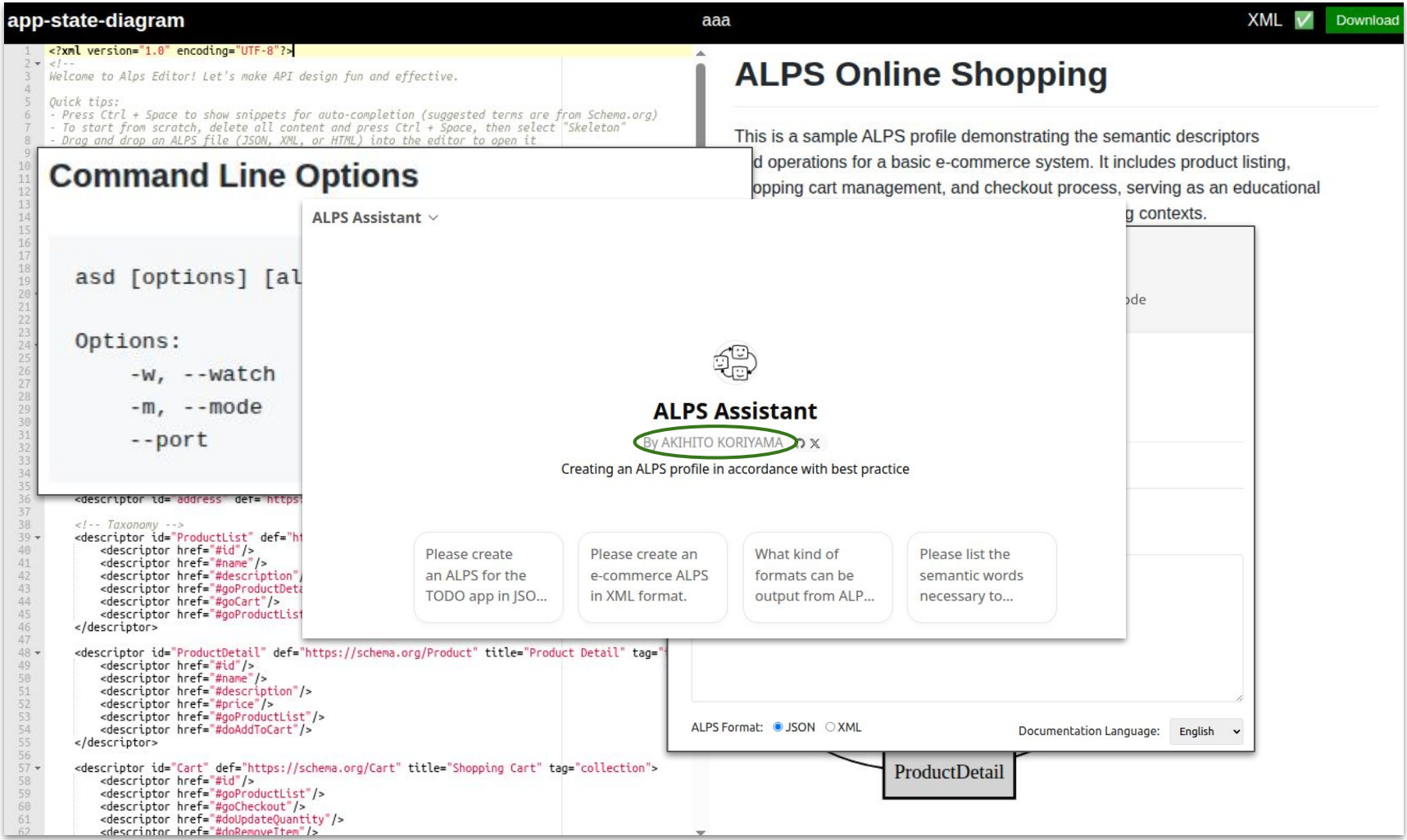
ALPS Online Shopping

This is a sample ALPS profile demonstrating the semantic descriptors
d operations for a basic e-commerce system. It includes product listing,
opping cart management, and checkout process, serving as an educational
ample for ALPS implementation in online shopping contexts.




app-state-diagram
aaa
XML
Download
<?xml version="1.0" encoding="UTF-8"?>
<!--
Welcome to Alps Editor! Let's make API design fun and effective.
Quick tips:
- Press Ctrl + Space to show snippets for auto-completion (suggested terms are from Schema.org)
- To start from scratch, delete all content and press Ctrl + Space, then select "Skeleton"
- Drag and drop an ALPS file (JSON, XML, or HTML) into the editor to open it
Command Line Options
asd [options] [alpsFile]
Options:
-w, --watch Watch mode
-m, --mode Drawing mode
--port Port to use (default 300
<descriptor id="address" def="https://schema.org/address" title="address" />
<!-- Taxonomy -->
<descriptor id="ProductList" def="https://schema.org/ItemList" title="Product List" tag="col
<descriptor href="#id" />
<descriptor href="#name" />
<descriptor href="#description" />
<descriptor href="#goProductDetail" />
<descriptor href="#goCart" />
<descriptor href="#goProductList" />
</descriptor>
<descriptor id="ProductDetail" def="https://schema.org/Product" title="Product Detail" tag="
<descriptor href="#id" />
<descriptor href="#name" />
<descriptor href="#description" />
<descriptor href="#price" />
<descriptor href="#goProductList" />
<descriptor href="#doAddToCart" />
</descriptor>
<descriptor id="Cart" def="https://schema.org/Cart" title="Shopping Cart" tag="collection">
<descriptor href="#id" />
<descriptor href="#goProductList" />
<descriptor href="#goCheckout" />
<descriptor href="#doUpdateQuantity" />
<descriptor href="#doRemoveItem" />
ALPS Online Shopping
This is a sample ALPS profile demonstrating the semantic descriptors
d operations for a basic e-commerce system. It includes product listing,
opping cart management, and checkout process, serving as an educational
ample for ALPS implementation in online shopping contexts.
ALPS Prompt Brewery
AI prompt generator for ALPS and implementation code
Step 1: User Story
Step 2: ALPS
ALPS Prompt Creation
Create from User Story
Convert Existing ALPS
Need inspiration?
Select a sample user story...
Enter your user story or system requirements here...
ALPS Format: JSON XML
Documentation Language:
English
ProductDetail


app-state-diagram
aaa
XML
Download
ALPS Online Shopping
This is a sample ALPS profile demonstrating the semantic descriptors
d operations for a basic e-commerce system. It includes product listing,
opping cart management, and checkout process, serving as an educational
g contexts.
Command Line Options
asd [options] [al
Options:
-w, --watch
-m, --mode
--port
ALPS Assistant
ALPS Assistant
By AKIHITO KORIYAMA
Creating an ALPS profile in accordance with best practice
Please create an ALPS for the TODO app in JSO...
Please create an e-commerce ALPS in XML format.
What kind of formats can be output from ALP...
Please list the semantic words necessary to...
ALPS Format: JSON XML
Documentation Language: English
ProductDetail


app-state-diagram
aaa
XML
Download
<?xml version="1.0" encoding="UTF-8"?>
<!--
Welcome to Alps Editor! Let's make API design fun and effective.
Quick tips:
- Press Ctrl + Space to show snippets for auto-completion (suggested terms are from Schema.org)
- To start from scratch, delete all content and press Ctrl + Space, then select "Skeleton"
- Drag and drop an ALPS file (JSON, XML, or HTML) into the editor to open it
Command Line Options
asd [options] [al
Options:
-w, --watch
-m, --mode
--port
ALPS Assistant
ALPS Assistant
By AKIHITO KORIYAMA
Creating an ALPS profile in accordance with best practice
Please create an ALPS for the TODO app in JSO...
Please create an e-commerce ALPS in XML format.
What kind of formats can be output from ALP...
Please list the semantic words necessary to...
ALPS Online Shopping
This is a sample ALPS profile demonstrating the semantic descriptors
d operations for a basic e-commerce system. It includes product listing,
opping cart management, and checkout process, serving as an educational
g contexts.
ALPS Format: JSON XML
Documentation Language: English
ProductDetail

Akihito Koriyama

Creator of BEAR.Sunday, a resource-oriented PHP framework. He explores API architecture and REST through the lens of semantic architecture and philosophical insight.

Akihito Koriyama

Creator of BEAR.Sunday, a resource-oriented PHP framework. He explores API architecture and REST through the lens of semantic architecture and philosophical insight.

"Traditionally, business-side requirements have been interpreted and formalized by API engineers. I'd be happy if this tool could help bridge that gap."

OK, we can move on now …

Writing API Stories

API Stories

- APIs start with a story
  - "We need…"
  - "Our customers requested…"
  - "I have an idea…"
- Stories are shared understanding
  - Our brains are wired for stories, not data
  - Stories are accessible
  - Stories are repeatable

Task Management

Purpose

We need to track 'Task' records in order to improve both timeliness and accuracy of customer follow-up activity.

Data

In this first pass at the application, we need to keep track of the following data properties:

- **id** : a globally unique value for each record
- **title** : the text content of the record
- **description** : the description of the record
- **dueDate** : the date the record is due to be completed
- **priority** : the priority of the task
- **assignedUser** : the user assigned to handle the task

Actions

This edition of the application needs to support the following operations:

create tasks, update status, and mark them complete.

- **List** : return a list of all active ToDo records in the system
- **Create** : add a new ToDo record to the system
- **UpdateStatus** : update the status of a single record
- **MarkComplete** : mark a single record as completed

Rules

None

Task Management - Distilled

As a project manager, I need a task tracking system.

Tasks have a title, description, due date, priority, and assigned user.

Users should be able to create tasks, update status, and mark them complete.

The system should display task lists filtered by status or assigned user.

Stories are shared understanding

Building the ALPS Description

Build the ALPS Description

- **Convert** the User Story into an LLM prompt
- Pass the prompt to an LLM to **generate** draft ALPS document
- Apply **review** tips to update the ALPS document
- Load ALPS document into an editor for final **validation**

ALPS Prompt Brewery

AI prompt generator for ALPS and implementation code

Step 1: User Story | Step 2: ALPS

ALPS Prompt Creation

Create from User Story | Convert Existing ALPS

Need inspiration? Task Management App

```
As a project manager, I need a task tracking system.
Tasks have a title, description, due date, priority, and assigned user.
Users should be able to create tasks, update status, and mark them complete.
The system should display task lists filtered by status or assigned user.
```

ALPS Format: (•) JSON ( ) XML

Documentation Language: English

Generate ALPS Prompt

Step 1: User Story | **Step 2: ALPS**

Convert ALPS to Implementation Format

Generated ALPS Prompt

Copy ALPS Prompt

```
# ALPS Profile Creation Prompt

Please create an ALPS profile based on the following requirements. This profile should represent a complete and consistent
application state design.

* Format: JSON
* Language: English
* Content:
```

💡 **Pro Tip:** After receiving your ALPS profile from the AI, consider asking: `"Please review this ALPS profile to verify that there are no isolated states (unreachable or exit-less states) and that all state transitions are properly connected. Also check if all semantic descriptors are consistently tagged and grouped."` Copy Tip

💡 **Next Step:** After confirming that ALPS is rendered correctly at https://editor.app-state-diagram.com/, Paste your ALPS profile into textarea and proceed directly to the format conversion step.

ALPS Assistant

By AKIHITO KORIYAMA

Creating an ALPS profile in accordance with best practice

Please create an ALPS for the TODO app in JSO...

Please create an e-commerce ALPS in XML format.

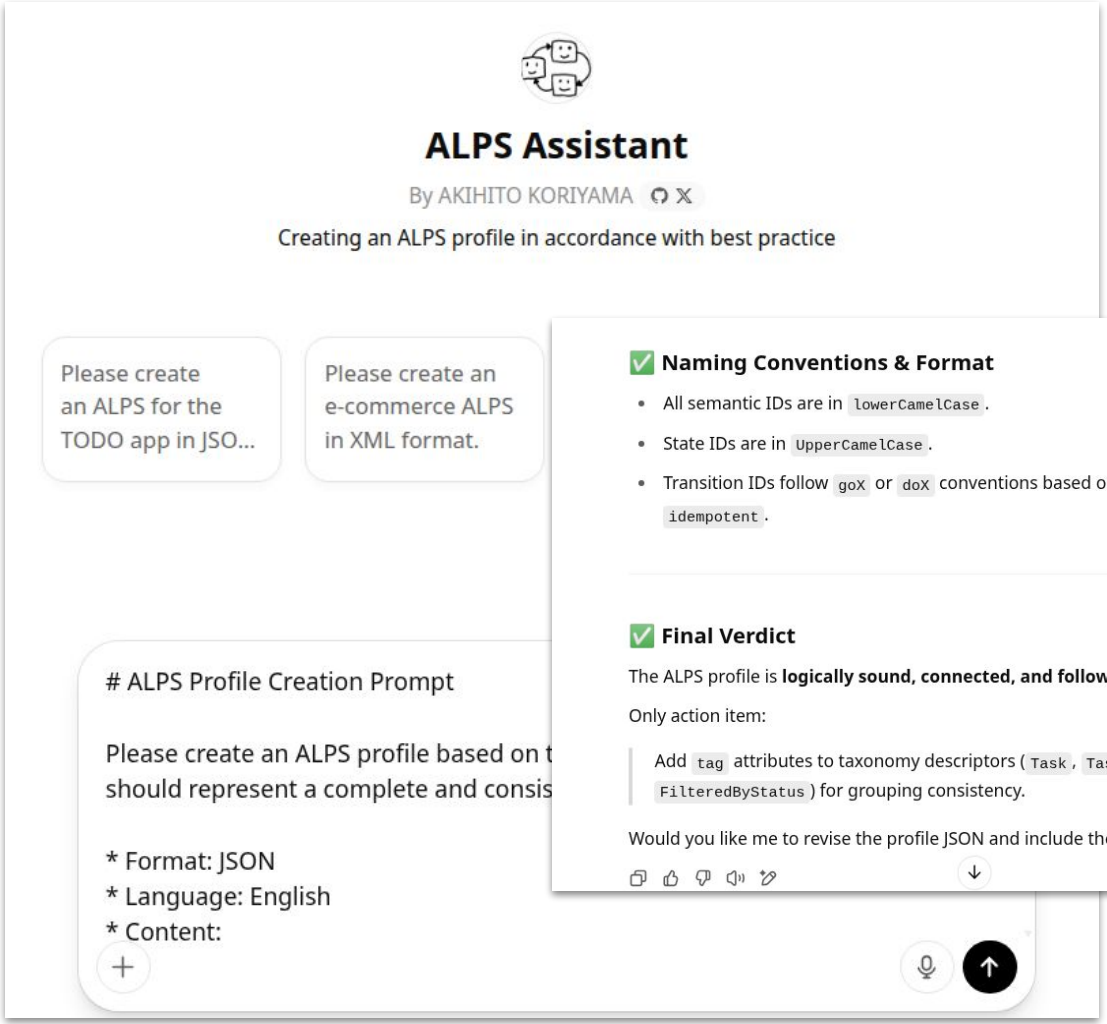
What kind of formats can be output from ALP...

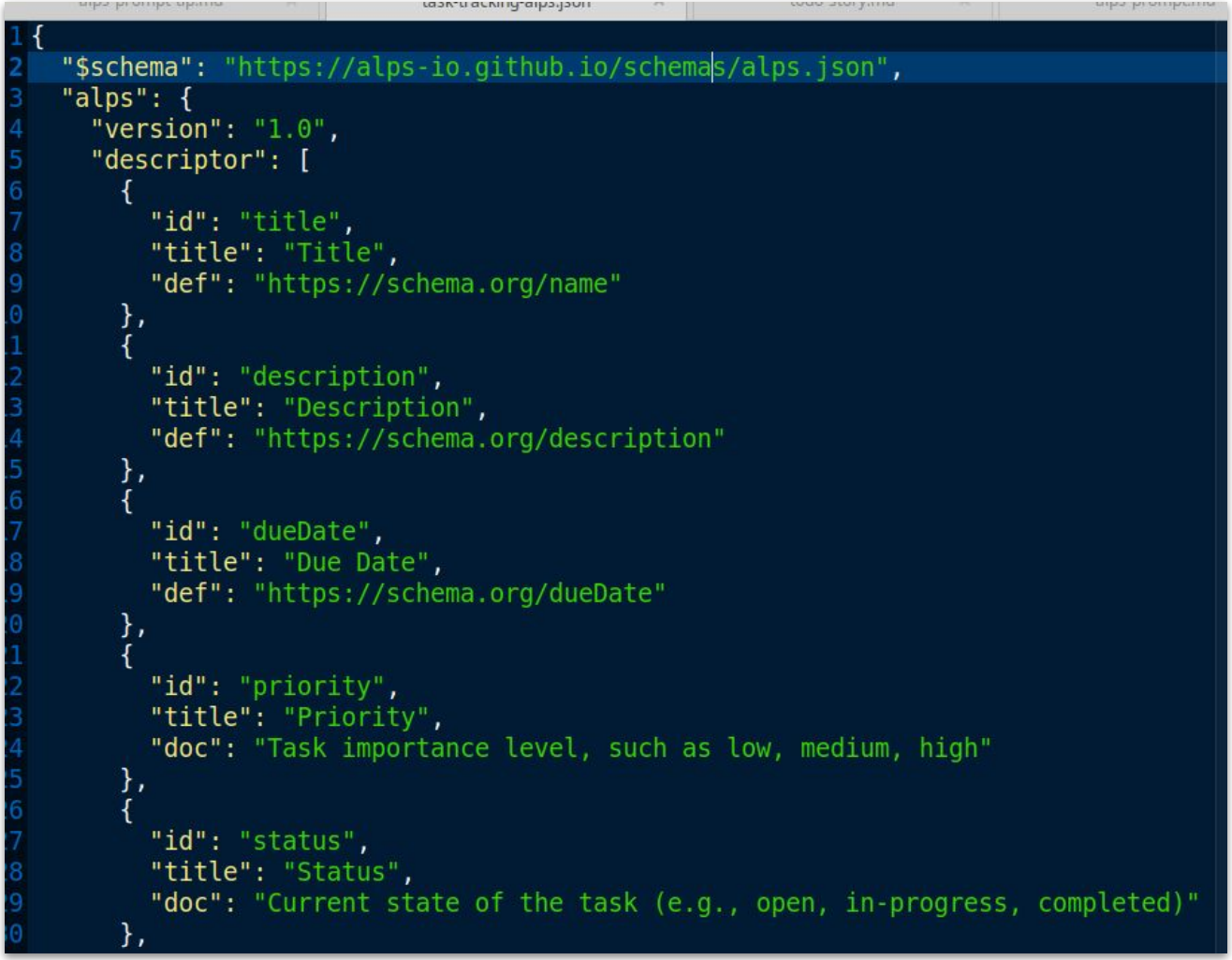
Please list the semantic words necessary to...

ALPS Profile Creation Prompt

Please create an ALPS profile based on the following requirements. This profile should represent a complete and consistent application state design.

* Format: JSON
* Language: English
* Content:


ALPS Assistant
By AKIHITO KORIYAMA
Creating an ALPS profile in accordance with best practice
Please create an ALPS for the TODO app in JSO...
Please create an e-commerce ALPS in XML format.
ALPS Profile Creation Prompt
Please create an ALPS profile based on t
should represent a complete and consis
* Format: JSON
* Language: English
* Content:
✅ Naming Conventions & Format
All semantic IDs are in lowerCamelCase.
State IDs are in UpperCamelCase.
Transition IDs follow goX or doX conventions based on safe, unsafe, or idempotent.
✅ Final Verdict
The ALPS profile is logically sound, connected, and follows best practices.
Only action item:
Add tag attributes to taxonomy descriptors (Task, TaskList, FilteredByUser, FilteredByStatus) for grouping consistency.
Would you like me to revise the profile JSON and include the missing tags?

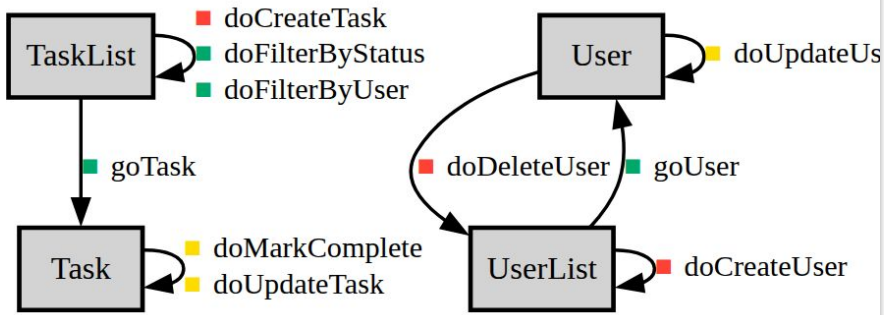

{
"$schema": "https://alps-io.github.io/schemas/alps.json",
"alps": {
"version": "1.0",
"descriptor": [
{
"id": "title",
"title": "Title",
"def": "https://schema.org/name"
},
{
"id": "description",
"title": "Description",
"def": "https://schema.org/description"
},
{
"id": "dueDate",
"title": "Due Date",
"def": "https://schema.org/dueDate"
},
{
"id": "priority",
"title": "Priority",
"doc": "Task importance level, such as low, medium, high"
},
{
"id": "status",
"title": "Status",
"doc": "Current state of the task (e.g., open, in-progress, completed)"
},

app-state-diagram

JSON ✅ Download


```
{
  "$schema": "https://alps-io.github.io/schemas/alps.json",
  "alps": {
    "version": "1.0",
    "descriptor": [
      {
        "id": "title",
        "title": "Title",
        "def": "https://schema.org/name"
      },
      {
        "id": "description",
        "title": "Description",
        "def": "https://schema.org/description"
      },
      {
        "id": "dueDate",
        "title": "Due Date",
        "def": "https://schema.org/dueDate"
      },
      {
        "id": "priority",
        "title": "Priority",
        "doc": "Task importance level, such as low, medium, high"
      },
      {
        "id": "status",
        "title": "Status",
        "doc": "Current state of the task (e.g., open, in-progress, completed)"
      },
      {
        "id": "assignedUser",
        "title": "Assigned User",
        "doc": "User to whom the task is assigned"
      },
      {
        "id": "userName",
        "title": "User Name",
        "def": "https://schema.org/name"
```


ALPS



\+ − 1:1

View: ◉ id ○ title

Tags: ☐ task-tracking ☐ user-management

☐ Semantic ▩ Safe ▩ Unsafe ▩ Idempotent

app-state-diagram

JSON ✅ Download

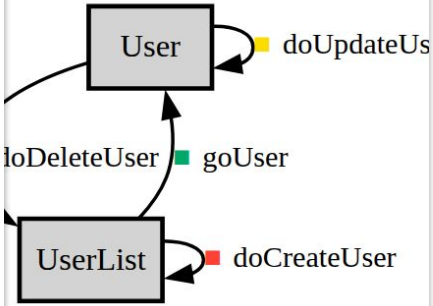
```
{
  "$schema": "https://alps-io.github.io/schemas/alps.json",
  "alps": {
    "version": "1.0",
    "descriptor": [
      {
        "id": "title",
        "title": "Title",
        "def": "https://schema.org
      },
      {
        "id": "description",
        "title": "Description",
        "def": "https://schema.org
      },
      {
        "id": "dueDate",
        "title": "Due Date",
        "def": "https://schema.org
      },
      {
        "id": "priority",
        "title": "Priority",
        "doc": "Task importance le
      },
      {
        "id": "status",
        "title": "Status",
        "doc": "Current state of t
      },
      {
        "id": "assignedUser",
        "title": "Assigned User",
        "doc": "User to whom the t
      },
      {
        "id": "userName",
        "title": "User Name",
        "def": "https://schema.org
```

ALPS

Semantic Descriptors

Search semantic descriptors...

Type	ID	Title	Contained	Extra Info
☐	assignedUser	Assigned User		doc: User to whom the task is assigned
☐	description	Description		def: schema.org/description
◪	doCreateTask	Create Task		tag: task-tracking rt: TaskList
◪	doCreateUser	Create User		tag: user-management rt: UserList
◪	doDeleteUser	Delete User		tag: user-management rt: UserList
▩	doFilterByStatus	Filter Tasks by Status		tag: task-tracking rt: TaskList
▩	doFilterByUser	Filter Tasks by Assigned User		tag: task-tracking rt: TaskList
■	doMarkComplete	Mark Task as Complete		tag: task-tracking rt: Task
■	doUpdateTask	Update Task		tag: task-tracking rt: Task
■	doUpdateUser	Update User Profile		tag: user-management



nanagement

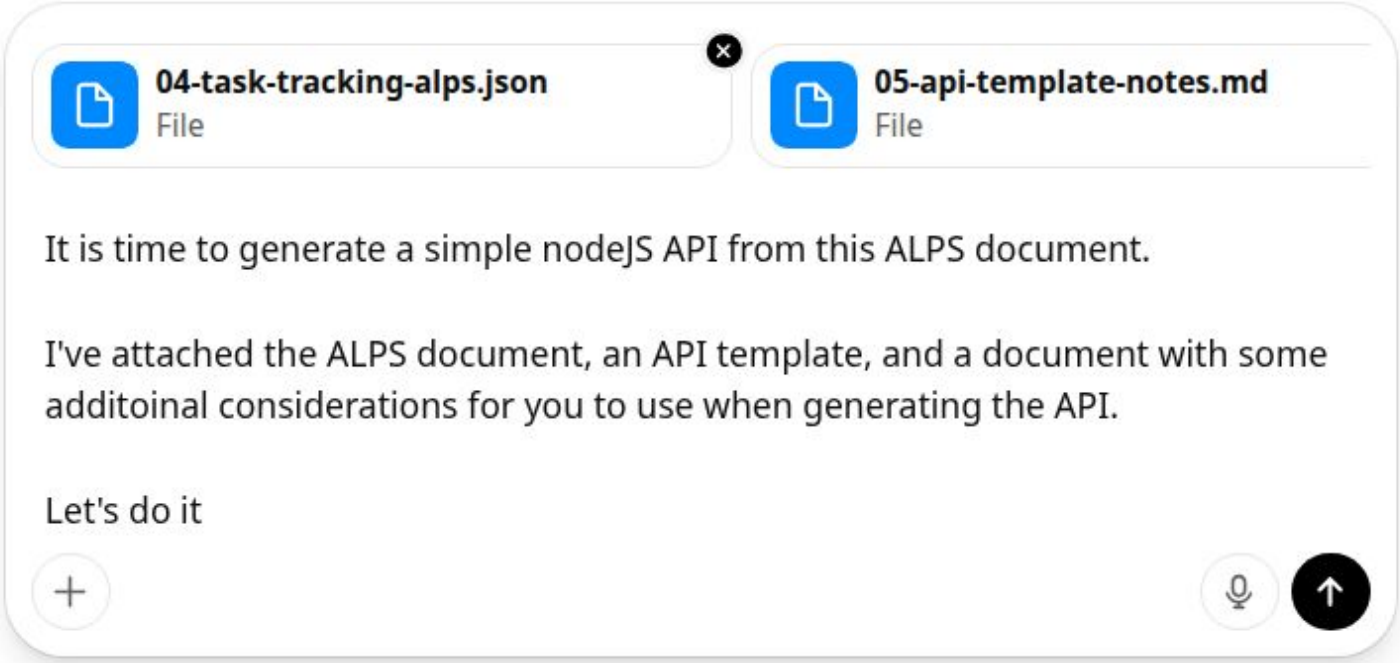
Idempotent

*Describe the **design**, not the Implementation*

Creating the NodeJS API

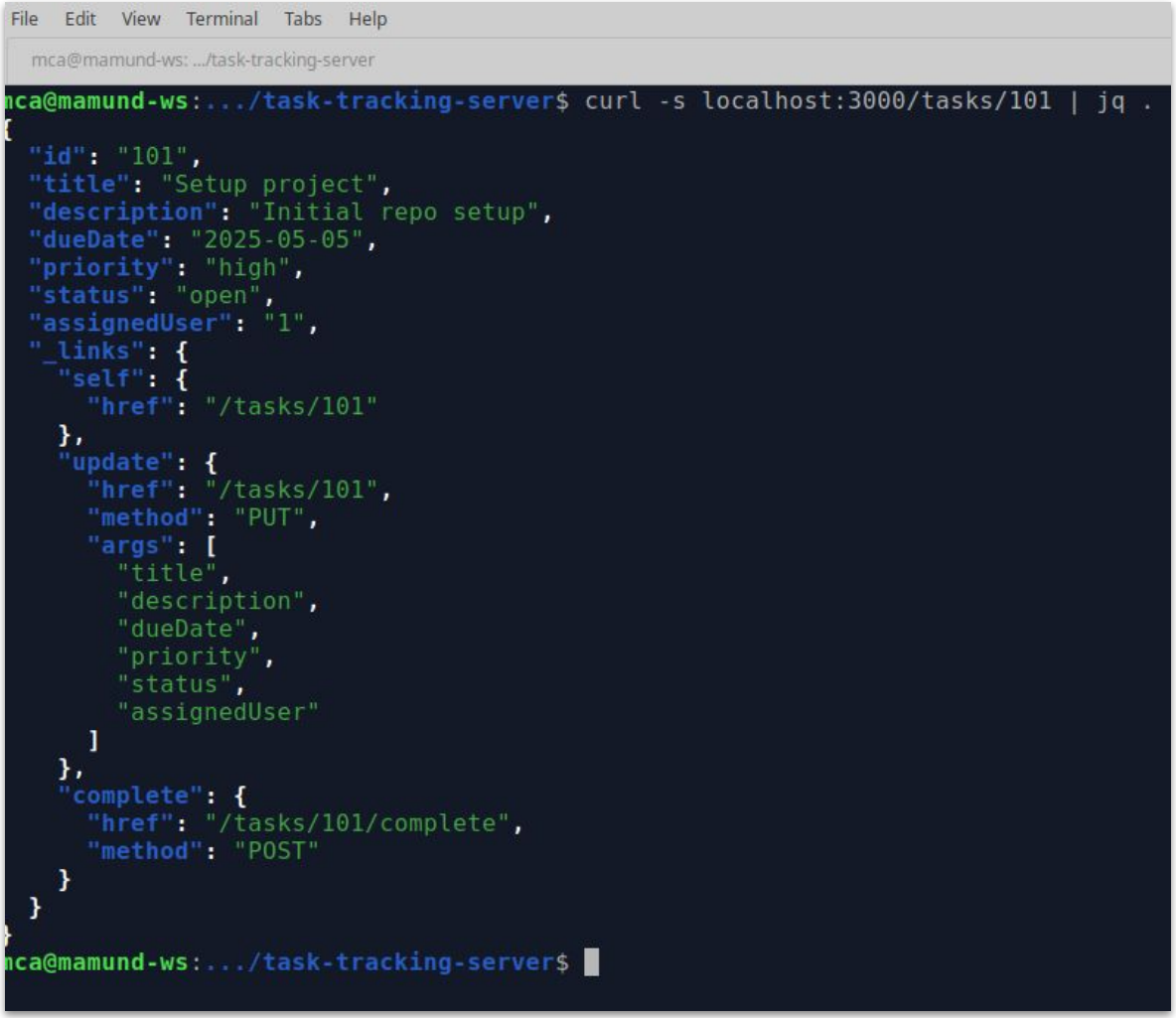
Creating the NodeJS API

- Use the API design description as a reference
  - Your ALPS is your map
- Focus on a working proof of concept (PoC)
  - Don't over-work the PoC code
- Share / Test the working PoC
  - PoC's are for feedback, not production


04-task-tracking-alps.json
File
05-api-template-notes.md
File
It is time to generate a simple nodeJS API from this ALPS document.
I've attached the ALPS document, an API template, and a document with some additoinal considerations for you to use when generating the API.
Let's do it

Task Api Nodejs

```
// app.js - Node.js API generated from ALPS profile
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;

app.use(bodyParser.json());

let users = [
  { id: 1, userName: "Alice", userEmail: "alice@example.com" },
  { id: 2, userName: "Bob", userEmail: "bob@example.com" }
];

let tasks = [
  { id: 1, title: "Write report", description: "Weekly report", dueDate: "2025-05-
  { id: 2, title: "Fix bug", description: "UI bug fix", dueDate: "2025-05-13", pri
];

// Root
app.get('/', (req, res) => {
  res.json({
    _links: {
      users: { href: "/users" },
      tasks: { href: "/tasks" }
    }
  });
});

// Users
app.get('/users', (req, res) => {
  res.json(users.map(u => ({ ...u, _links: userLinks(u.id) })));
});

app.get('/users/:id', (req, res) => {
  const user = users.find(u => u.id == req.params.id);
  if (user) res.json({    user,  links: userLinks(user.id) });
```


File Edit View Terminal Tabs Help
mca@mamund-ws: .../task-tracking-server
mca@mamund-ws:.../task-tracking-server$ curl -s localhost:3000/tasks/101 | jq .
{
"id": "101",
"title": "Setup project",
"description": "Initial repo setup",
"dueDate": "2025-05-05",
"priority": "high",
"status": "open",
"assignedUser": "1",
"_links": {
"self": {
"href": "/tasks/101"
},
"update": {
"href": "/tasks/101",
"method": "PUT",
"args": [
"title",
"description",
"dueDate",
"priority",
"status",
"assignedUser"
]
},
"complete": {
"href": "/tasks/101/complete",
"method": "POST"
}
}
}
mca@mamund-ws:.../task-tracking-server$

Working code and rough consensus

And so …

Summary

- Writing API Stories
  - Stories are shared understanding

Summary

- Writing API Stories
  - Stories are shared understanding
- Building the ALPS Design Document
  - Describe the design, not the implementation

Summary

- Writing API Stories
  - Stories are shared understanding
- Building the ALPS Design Document
  - Describe the design, not the implementation
- Creating the NodeJS API
  - Working code and rough consensus

Task Management

Purpose

We need to track 'Task' records in order to improve both timeliness and accuracy of customer follow-up activity.

Data

In this first pass at the application, we need to keep track of the following data properties:

- **id** : a globally unique value for each record
- **title** : the text content of the record
- **description** : the description of the record
- **dueDate** : the date the record is due to be completed
- **priority** : the priority of the task
- **assignedUser** : the user assigned to handle the task

Actions

This edition of the application needs to support the following operations:

create tasks, update status, and mark them complete.

- **List** : return a list of all active ToDo records in the system
- **Create** : add a new ToDo record to the system
- **UpdateStatus** : update the status of a single record
- **MarkComplete** : mark a single record as completed

Rules

None

Task Management

Purpose

We need to track 'Task' records in order to improve both timeliness and accuracy of customer follow-up activity.

Data

In this first pass at the application, we need to keep track of the following data properties:

- **id** : a globally unique value for each record
- **title** : the text content of the record
- **description** : the description of the record
- **dueDate** : the date the record is due to be completed
- **priority** : the priority of the task
- **assignedUser** : the user assigned to handle the task

Actions

This edition of the application needs to support the following operations:

create tasks, update status, and mark them complete.

- **List** : return a list of all active ToDo records in the system
- **Create** : add a new ToDo record to the system
- **UpdateStatus** : update the status of a single record
- **MarkComplete** : mark a single record as completed

Rules

None

Task Api Nodejs

```
// app.js - Node.js API generated from ALPS profile
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;

app.use(bodyParser.json());

let users = [
  { id: 1, userName: "Alice", userEmail: "alice@example.com" },
  { id: 2, userName: "Bob", userEmail: "bob@example.com" }
];

let tasks = [
  { id: 1, title: "Write report", description: "Weekly report", dueDate: "2025-05-
  { id: 2, title: "Fix bug", description: "UI bug fix", dueDate: "2025-05-13", pri
];

// Root
app.get('/', (req, res) => {
  res.json({
    _links: {
      users: { href: "/users" },
      tasks: { href: "/tasks" }
    }
  });
});

// Users
app.get('/users', (req, res) => {
  res.json(users.map(u => ({ ...u, _links: userLinks(u.id) })));
});

app.get('/users/:id', (req, res) => {
  const user = users.find(u => u.id == req.params.id);
  if (user) res.json({    user,   links: userLinks(user.id) });
```

Finally …

Andrej Karpathy
@karpathy

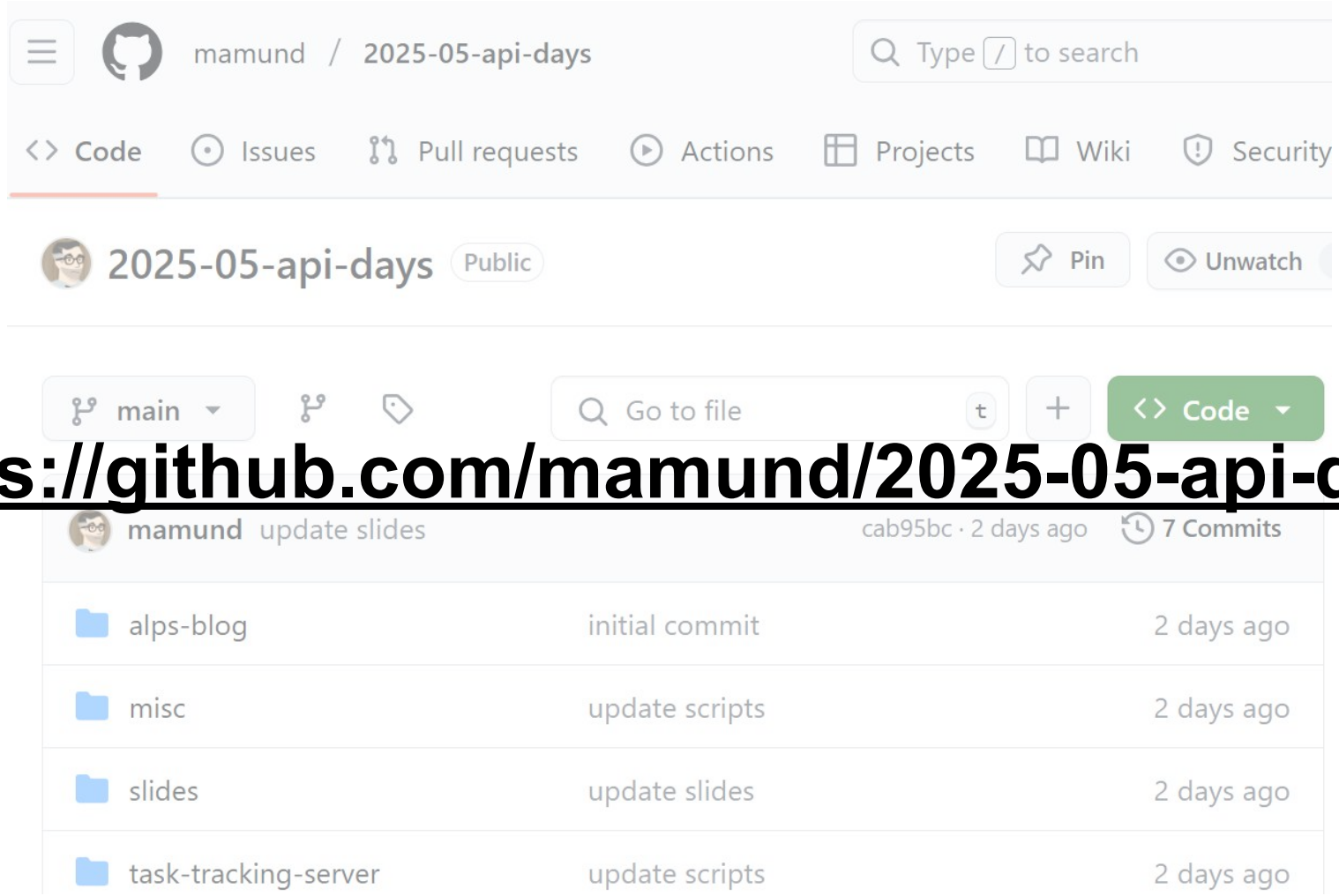
There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

6:17 PM · Feb 2, 2025 · **4.9M** Views

1.3K 5K 29K 15K

Andrej Karpathy

"Ultimately, vibe coding full web apps today is kind of messy and not a good idea for anything of actual importance."
— Andrej Karpathy


mamund / 2025-05-api-days
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
2025-05-api-days
Public
Pin
Unwatch
main
Go to file
t
Code
https://github.com/mamund/2025-05-api-days
mamund update slides
cab95bc · 2 days ago
7 Commits
alps-blog
initial commit
2 days ago
misc
update scripts
2 days ago
slides
update slides
2 days ago
task-tracking-server
update scripts
2 days ago

ALPS + AI = API

https://github.com/mamund/2025-05-api-days

Mike Amundsen
@mamund

ALPS + AI = API

https://github.com/mamund/2025-05-api-days

Mike Amundsen
@mamund