API Design Basics: Story

Task Management

Purpose

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

Data Properties

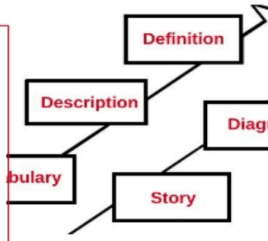
In this first pass at the application, we need to keep track of the following data properties:

- **id** : a globally unique value for each record [uuid]
- **title** : the text content of the record (the title of the task) [string]
- **description** : the description of the record (the task details) [text]
- **dueDate** : the date the record is due to be completed [date]
- **status** : Indicates the status of the record (active, completed) [enumerated string]
- **priority** : the priority of the task (a number between 1 and 5) [enumerated number]
- **assignedUser** : the user assigned to handle the task (a simple name string) [string]

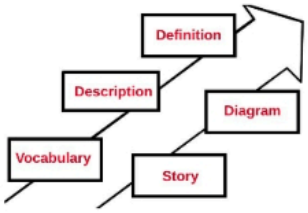
Resources

The following are resources, or states, of the API. Each state has one or more possible Actions.

- **Home** : The home or landing page of the API. Users typically start here.
  - Actions: **GetTaskList**, **GetFilteredTaskList**, **ShowHomePage**
- **TaskCollection** : The list of tasks in the system are displayed here.
  - Actions: **ShowHomePage**, **GetTaskList**, **GetFilteredTaskList**, **CreateNewTask**,**GetTaskItem**
- **TaskItem** : This resource shows a single task (selected from the TaskCollection)
  - Actions: **EditExistingTask**, **UpdateStatusOfTask**, **SetDueDateOfTask**, **AssignUserToTask**, **GetTaskList**, **GetFilteredTaskList**, **ShowHomePage**

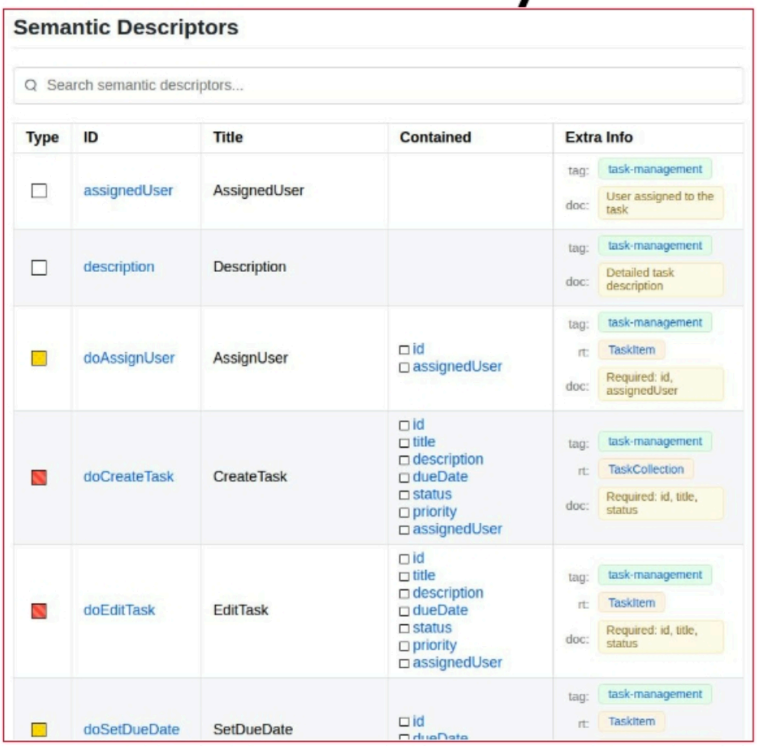


API Design Basics: Vocabulary



- All the "magic words" related to the domain
- Not just the properties:
  - `id`, `givenName`, etc.
- Also :
  - Actions (`create, approve, save`, etc.)
  - Enumerators (`status.active, status.inactive`)
  - Containers (`item.id, item.givenName`, etc.)
  - Resources/Topics (`home, collection, item`)

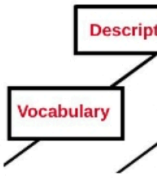
API Design Basics: Vocabulary



Semantic Descriptors

Search semantic descriptors...

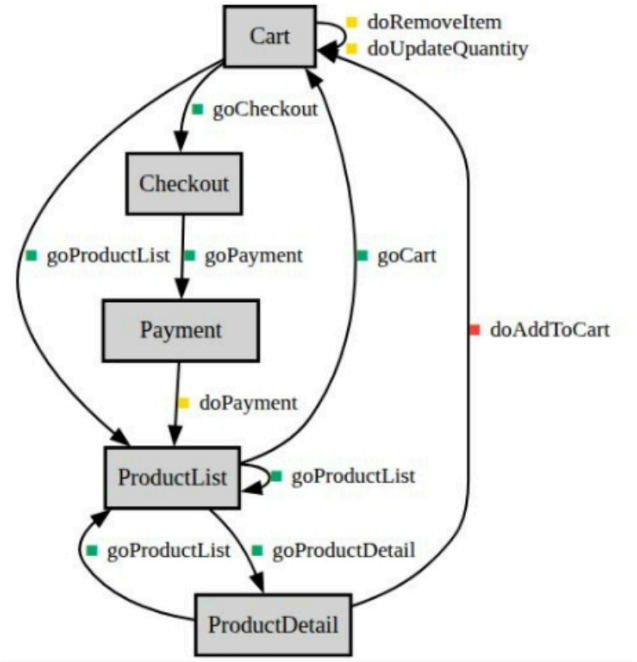
Type	ID	Title	Contained	Extra Info
☐	assignedUser	AssignedUser		tag: task-management doc: User assigned to the task
☐	description	Description		tag: task-management doc: Detailed task description
☐	doAssignUser	AssignUser	☐ id ☐ assignedUser	tag: task-management rt: TaskItem doc: Required: id, assignedUser
☐	doCreateTask	CreateTask	☐ id ☐ title ☐ description ☐ dueDate ☐ status ☐ priority ☐ assignedUser	tag: task-management rt: TaskCollection doc: Required: id, title, status
☐	doEditTask	EditTask	☐ id ☐ title ☐ description ☐ dueDate ☐ status ☐ priority ☐ assignedUser	tag: task-management rt: TaskItem doc: Required: id, title, status
☐	doSetDueDate	SetDueDate	☐ id	tag: task-management rt: TaskItem



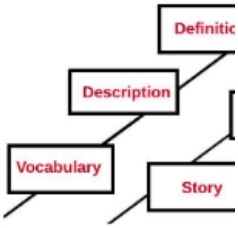
API Design Basics: Diagram

- Visual representation of the API
- Not ..
  - a state diagram
  - a flow diagram
- An *interaction* diagram

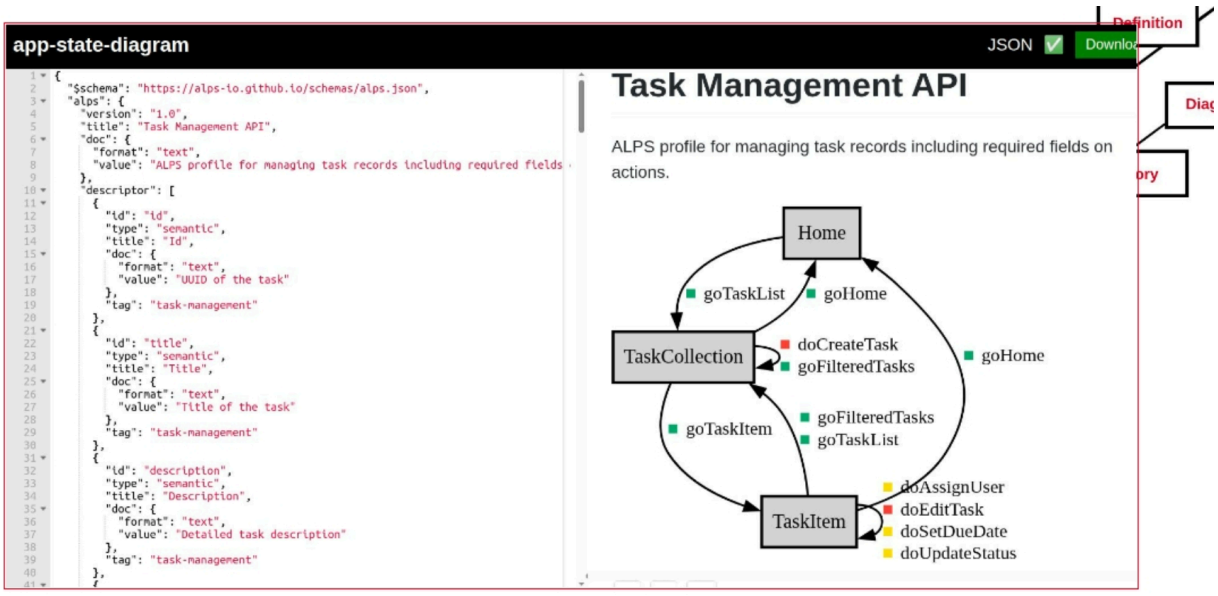
API Design Basics: Diagram



API Design Basics: Description



- Why "Descriptions"?
  - The complete design in one place, in one voice.
- Who uses descriptions?
  - Architects, Designers, Developers, Test, Security, etc.
- Time Traveling w/ descriptions
  - Options for the future, replays for the past
- Interaction-centric and Operation-centric



API Design Basics: Definition

- API definitions translate design into implementation
- API definitions are implementation-specific
  - OpenAPI (HTTP)
  - AnsycAPI (Event-Driven)
  - ProtoBuf (gRPC)
  - SDL (GraphQL)
- Usually API definitions power "generators"
  - Code
  - Diagrams
  - Documentation

AI Prompt Basics : Tips

- "What are my options?"
- "Do you have all you need?"
- "Ask me questions until you are confident you can perform the task."
- "What do I need to repeat this in a new conversation tomorrow?"

Authoring API Stories : Story Parts

- Purpose
- Data Properties
- Resources
- Actions
- Rules

Authoring API Stories : Story Parts

Actions

This edition of the application needs to support the following operations. Each action has zero or more input properties and always has one return value (to another state). Each action is also marked as either Safe (GET), Unsafe (POST), Idempotent (PUT/PATCH), or Delete (DELETE)

- **ShowHomePage** : Use this action to display the `home` resource
  - Inputs: None
  - Returns: **Home**
  - Type: Safe
- **GetTaskCollection** : Use this action to return a list of all Task records in the system
  - Inputs: None
  - Returns: **TaskCollection**
  - Type: Safe
- **GetTaskItem**: Use this action to get a single existing task record.
  - Inputs: id
  - Required: id
  - Returns: **TaskItem**
  - Type: Safe
- **CreateNewTask** : Use this action to add a new Task record to the system
  - Inputs: id, title, description, dueDate, status, priority, assignedUser

Rules

- When executing **CreateNewTask**, the client should supply a unique `id` value.

- Purpose
  - Sentence or two
- Data Properties
  - All data fields (`id, givenName`, etc,)
- Resources
  - All "resting states" (`home, list, item`)
- Actions
  - All actions (`save, share, filter`, etc.)
- Rules
  - Constraints (id must be unique, etc.)

https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/tasks-api-story.md
Task Management

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

Data Properties
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 [uuid]
* **title** : the text content of the record (the title of the task) [string]
* **description** : the description of the record (the task details) [text]
* **dueDate** : the date the record is due to be completed [date]
* **status** : Indicates the status of the record (active, completed) [enumerated string]
* **priority** : the priority of the task (a number between 1 and 5) [enumerated number]
* **assignedUser** : the user assigned to handle the task (a simple name string) [string]

Resources
The following are resources, or states, of the API. Each state has one or more possible Actions.

 * **Home** : The home or landing page of the API. Users typically start here.
   * Actions: **GetTaskList**, **GetFilteredTaskList**, **ShowHomePage**
 * **TaskCollection** : The list of tasks in the system are displayed here.
   * Actions: **ShowHomePage**, **GetTaskList**, **GetFilteredTaskList**,
CreateNewTask,**GetTaskItem**
 * **TaskItem** : This resource shows a single task (selected from the TaskCollection)
   * Actions: **EditExistingTask**, **UpdateStatusOfTask**, **SetDueDateOfTask**,
AssignUserToTask, **GetTaskList**, **GetFilteredTaskList**, **ShowHomePage**

Actions
This edition of the application needs to support the following operations. Each action has zero or more input properties and always has one return value (to another state). Each action is also marked as either Safe (GET), Unsafe (POST), Idempotent (PUT/PATCH), or Delete (DELETE)

 * **ShowHomePage** : Use this action to display the `home` resource
   * Inputs: None
   * Returns: **Home**
   * Type: Safe
 * **GetTaskCollection** : Use this action to return a list of all Task records in the system
   * Inputs: None
   * Returns: **TaskCollection**
   * Type: Safe
 * **GetTaskItem**: Use this action to get a single existing task record.
   * Inputs: id
   * Required: id
   * Returns: **TaskItem**
   * Type: Safe
 * **CreateNewTask** : Use this action to add a new Task record to the system
   * Inputs: id, title, description, dueDate, status, priority, assignedUser
   * Required: id, title, status
   * Returns: **TaskCollection**
   * Type: Unsafe
 * **EditExistingTask** : Use this action to modify an existing Task record to the system
   * Inputs: id, title, description, dueDate, status, priority, assignedUser
   * Required: id, title, status
   * Returns: **TaskItem**
   * Type: Unsafe
 * **UpdateStatusOfTask** : Use this action to update the `status` of a single record
   * Inputs: id, status
   * Required: id, status

  * Returns: **TaskItem**
  * Type: Idempotent
* **SetDueDateOfTask** : Use this action to set the `dueDate` of a single record
  * Inputs: id, dueDate
  * Required: id, dueDate
  * Returns: **TaskItem**
  * Type: Idempotent
* **AssignUserToTask** : Use this action to update the name of the `assignedUser` of a single record
  * Inputs: id, assignedUser
  * Required: id, assignedUser
  * Returns: **TaskItem**
  * type: Idempotent
* **GetFilteredTaskCollection** : Use this action to filter the list of tasks by `title`, `dueDate`, `status`, `priority` and/or `assignedUser`
  * Inputs: title, dueDate,status, priority
  * Returns: **TaskCollection**
  * Type: Safe

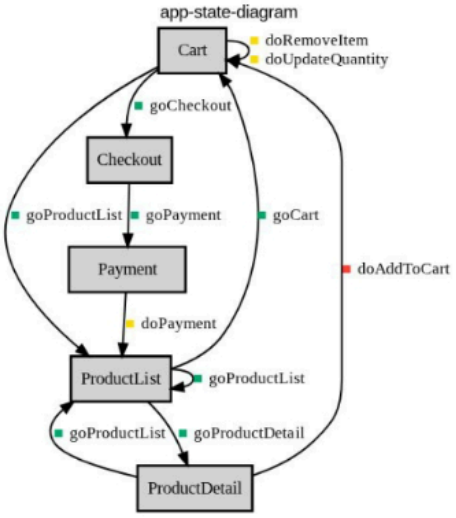
Rules
* When executing **CreateNewTask**, the client should supply a unique `id` value.

Safe: idempotent

Generating ALPS Design Docs

Application-Level Profile Semantics

- ALPS is a format that expresses application-level meaning and structure
- Focus on Actions and Resources
- Supporting RESTful Design

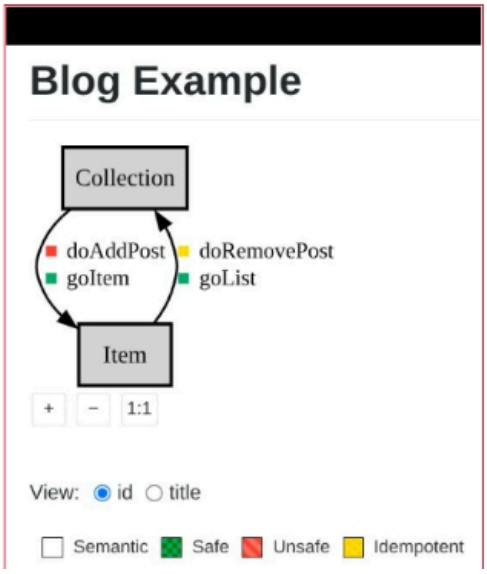


lides

Generating ALPS Design Docs

ALPS Basics

- Establish Meaning
- Identify Resources
- Describe Actions



- **Establish Meaning**
- Identify Resources
- Describe Actions

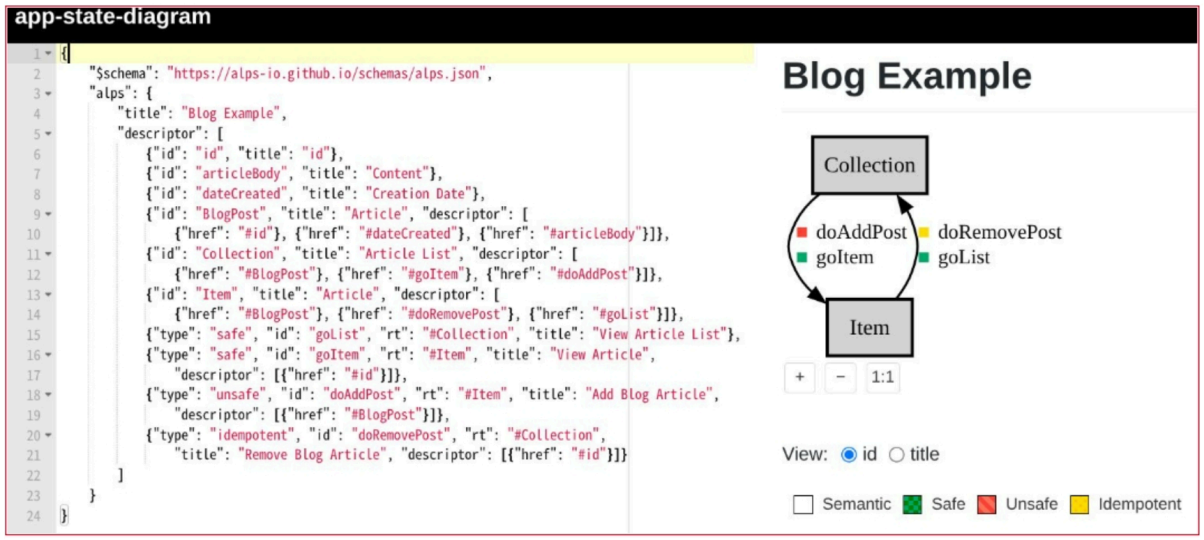
```
{
    "$schema": "https://alps-io.github.io/schemas/alps.json",
    "alps": {
        "title": "Blog Example",
        "descriptor": [
            {"id": "id", "title": "id"},
            {"id": "articleBody", "title": "Content"},
            {"id": "dateCreated", "title": "Creation Date"},
            {"id": "BlogPost", "title": "Article", "descriptor": [
                {"href": "#id"}, {"href": "#dateCreated"}, {"href": "#articleBody"}]},
```

- Establish Meaning
- **Identify Resources**
- Describe Actions

```
{"id": "Collection", "title": "Article List", "descriptor": [
    {"href": "#BlogPost"}, {"href": "#goItem"}, {"href": "#doAddPost"}]},
{"id": "Item", "title": "Article", "descriptor": [
    {"href": "#BlogPost"}, {"href": "#doRemovePost"}, {"href": "#goList"}]},
```

- Establish Meaning
- Identify Resources
- **Describe Actions**

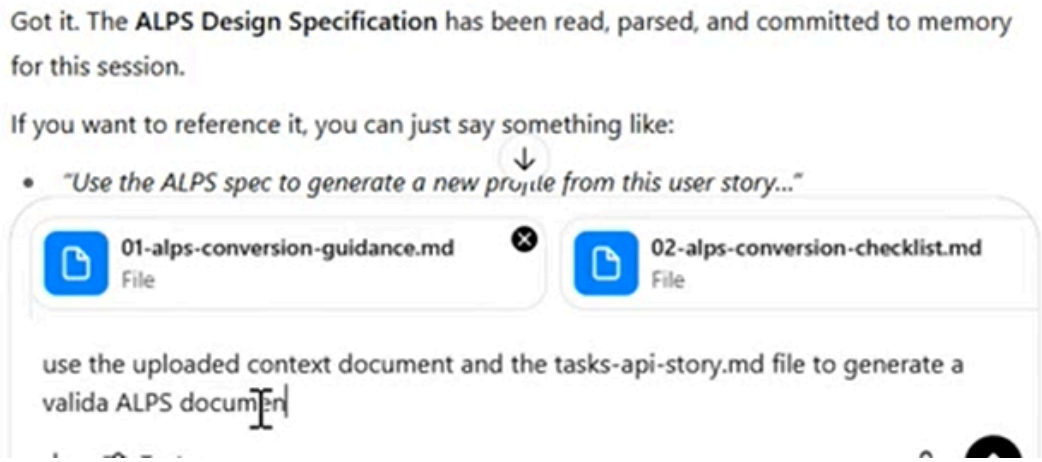

```
{"type": "safe", "id": "goList", "rt": "#Collection", "title": "View Article List"},
{"type": "safe", "id": "goItem", "rt": "#Item", "title": "View Article",
    "descriptor": [{"href": "#id"}]},
{"type": "unsafe", "id": "doAddPost", "rt": "#Item", "title": "Add Blog Article",
    "descriptor": [{"href": "#BlogPost"}]},
{"type": "idempotent", "id": "doRemovePost", "rt": "#Collection",
    "title": "Remove Blog Article", "descriptor": [{"href": "#id"}]}
```



To chatgpt:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/background/00-alps-design-spec.md

Upload
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-s

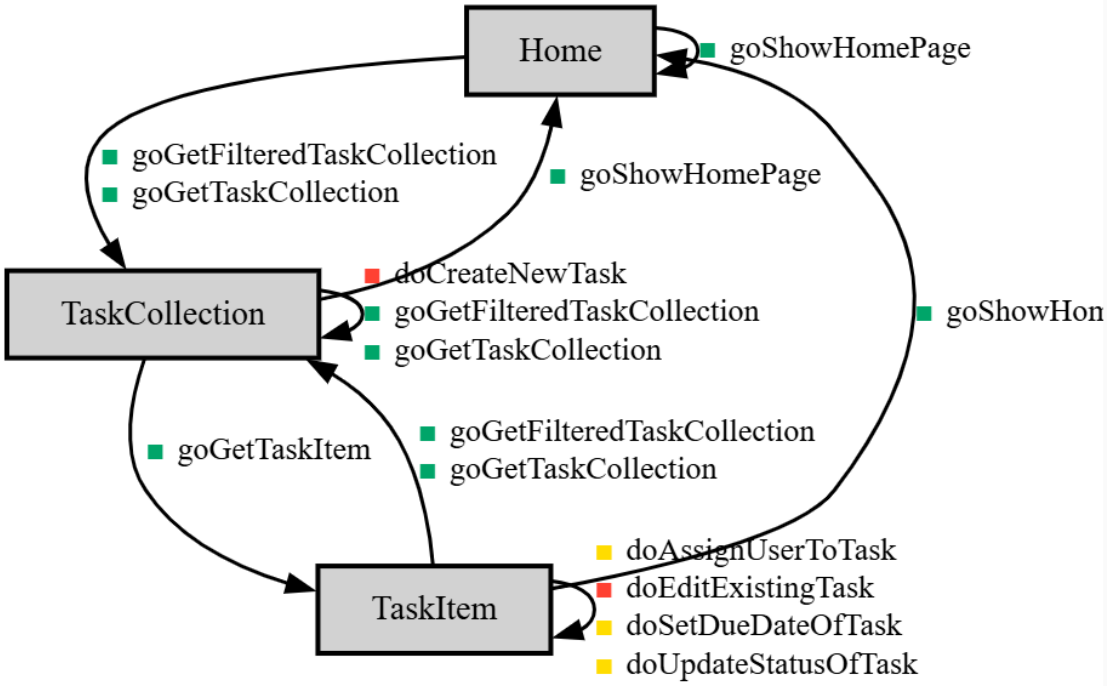
tories-to-alps/01-alps-conversion-guidance.md,
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/02-alps-conversion-checklist.md,
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/03-alps-prompt-template.md,
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/05-alps-verification-checklist.md and
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/tasks-api-story.md to Chatgpt



Save the json file generated.
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/01-api-stories-to-alps/tasks-alps-profile.json

Go to editor for alps and paste the json
https://editor.app-state-diagram.com/?i=1

ALPS


Home
goShowHomePage
goGetFilteredTaskCollection
goGetTaskCollection
goShowHomePage
TaskCollection
doCreateNewTask
goGetFilteredTaskCollection
goGetTaskCollection
goShowHon
goGetTaskItem
goGetFilteredTaskCollection
goGetTaskCollection
doAssignUserToTask
doEditExistingTask
doSetDueDateOfTask
doUpdateStatusOfTask
TaskItem

Generating API Documentation

What are Docs?

- Design Document
  - Intentions, scope, capabilities
- Implementation Plan
  - Inputs, outputs, states, protocols
- Operational Guide
  - Deployment, dependencies, monitoring, etc

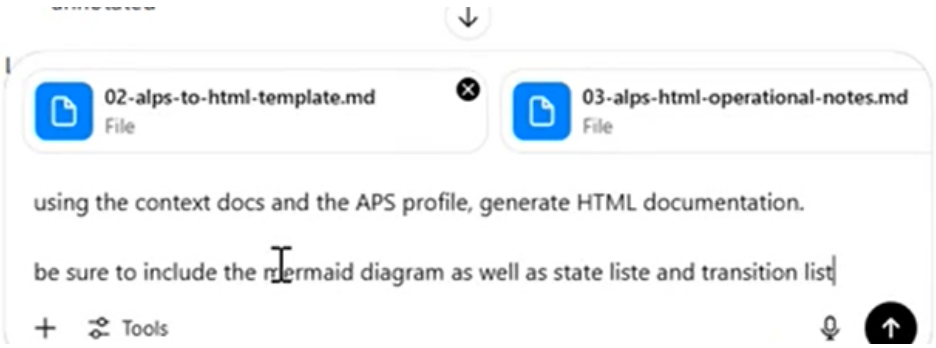
Slides

Generating API Documentation

Basic elements of API documentation

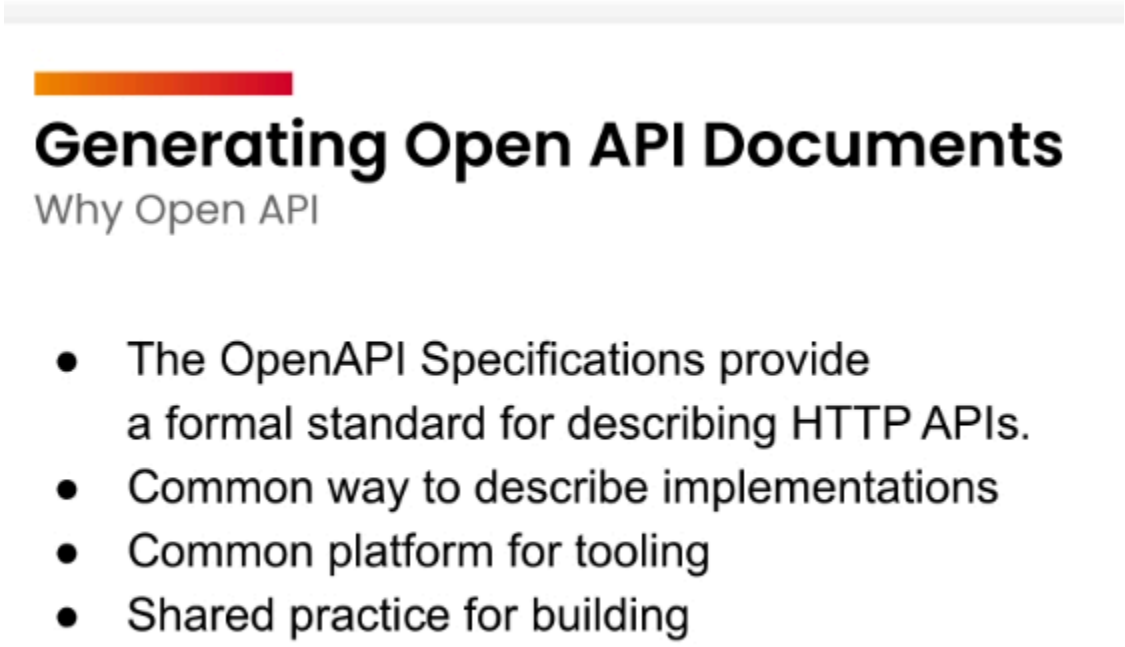
- Purpose
  - Short statement
- Data Properties
  - Name, type, description
- Resource States
  - Name, contents, links
- Action Transitions
  - Inputs, protocol details, outputs (states)
- Metadata
  - Errors, limits, security, etc.

Upload:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/02-alps-to-html/01-alps-to-html-step-by-step.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/02-alps-to-html/02-alps-to-html-template.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/02-alps-to-html/03-alps-html-operational-notes.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/background/tasks-alps-profile.json



o/p:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/02-alps-to-html/tasks-alps-doc.html

Generating OpenAPI docs:



Generating Open API Documents

Open API Basics

- **Basic Info**
- Component Section
- Paths Section

```yaml
openapi: 3.0.0
info:
  version: '1.0'
  title: Person API
  description: An example API for the "Designing APIs with OpenAPI" workshop
servers:
  # Added by API Auto Mocking Plugin
  - description: SwaggerHub API Auto Mocking
    url: https://virtserver.swaggerhub.com/amundsen/Person-API/1.0

paths: {}
components: {}
```

Slides

Generating Open API Documents

Open API Basics

- Basic Info
- **Component Section**
- Paths Section

```yaml
parameters:
  nameParam:
    name: name
    in: query
    descriptio
    required:
    schema:
      type: st
      example:
```

```yaml
PersonItem:
  type: object
  properties:
    id:
      type: string
      description:
      example: "q1
    name:
      type: string
      description:
      example: "Morton Muffley"
    email:
      type: string
      description: "Default email for
      example: "morton.muffley@example
PersonCollection:
  type: array
  items:
    $ref: '#/components/schemas/PersonItem'
```

```yaml
responses:
  GenericError:
    description: "
    content:
      application/
        schema:
          $ref: "#
```

```yaml
requestBodies:
  AddPerson:
    description: "body for a new Person record"
    required: true
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/PersonItem"
      application/x-www-urlencoded:
        schema:
          $ref: "#/components/schemas/PersonItem"
```

Generating Open API Documents

Open API Basics

- Basic Info
- Component Section
- **Paths Section**

```yaml
### work in progress record ###
/wip/{identifier}:
  get:
    operationId: wipItem
    summary: Use this to return a single onboarding record
    tags:
    - onboarding

    parameters:
      - $ref: "#/components/parameters/identifier"
      - $ref: "#/components/parameters/eTag"

    responses:
      '200':
        $ref: "#/components/responses/reply"

      default:
        $ref: "#/components/responses/error"
```

Upload:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/03-alps-to-open-api/01-alps-to-openapi-guidance-updated.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/03-alps-to-open-api/02-alps-to-openapi-checklist-updated.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/03-alps-to-open-api/99-alps-to-openapi-system-prompt.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/03-alps-to-open-api/03-alps-to-openapi-metadata.yaml
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/background/tasks-alps-profile.json

Use these and generate the openapi document

o/p:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/03-alps-to-open-api/tasks-openapi.yaml

https://app.swaggerhub.com/payal2/home
: create new -> Import and document API

Generating Prototype code:

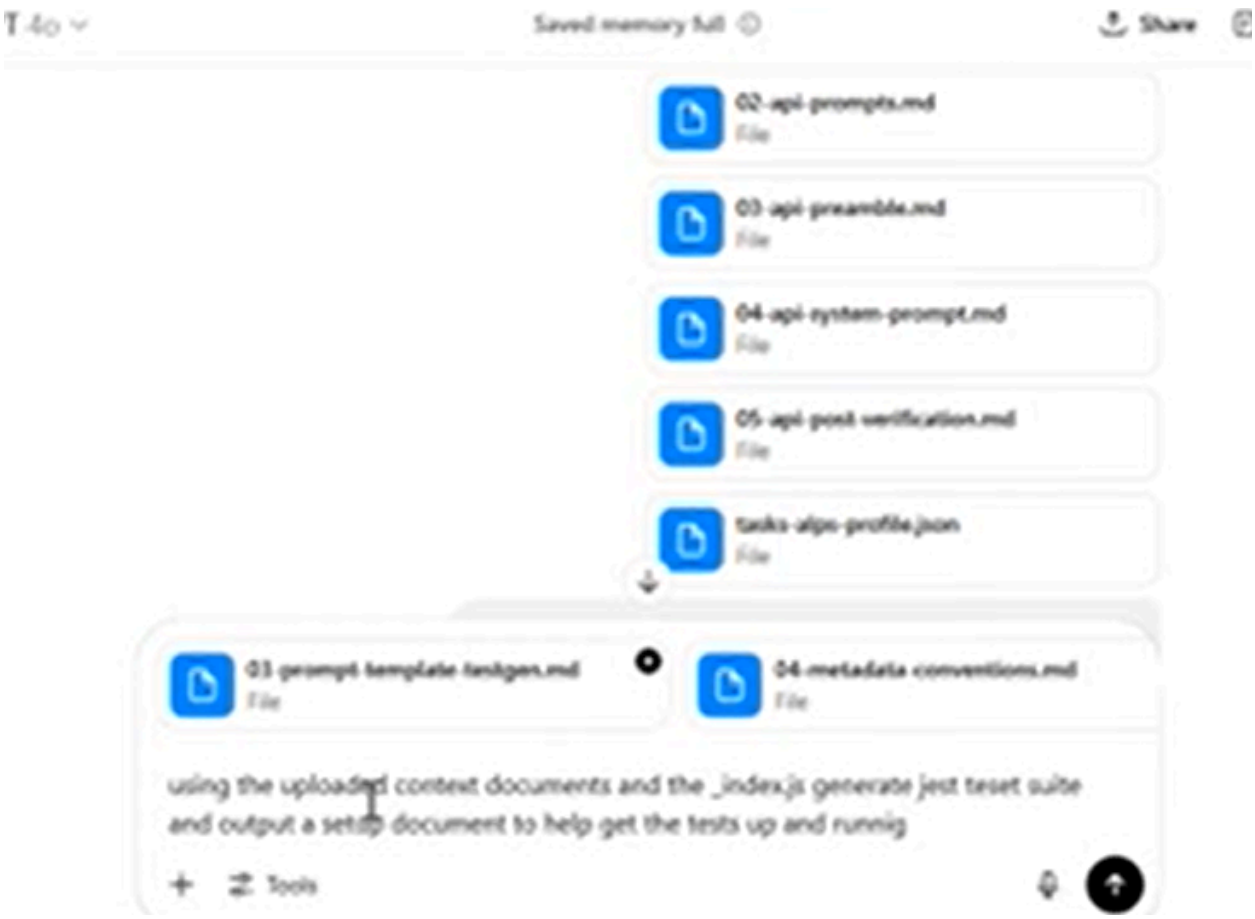
Upload
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/01-api-template-notes-v2.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/02-api-checklist-v2.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/02-api-prompts.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/03-api-preamble.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/04-api-system-prompt.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/05-api-post-verification.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/background/tasks-alps-profile.json

o/p:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/2025-06-15-tasks/index.js

Generating Interface Tests:

- Confirm the URL exists and responds as expected
- Validate the methods, input parameters
- Inspect the response bodies

Upload:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/02-index-modification-rules.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/03-prompt-template-testgen.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/04-metadata-conventions.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/05-test-verification-checklist.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/06-test-style-guide.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/07-test-context.json
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/04-alps-to-nodejs/task-management/_index.js



O/p:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/05-nodejs-to-test/api.test.js

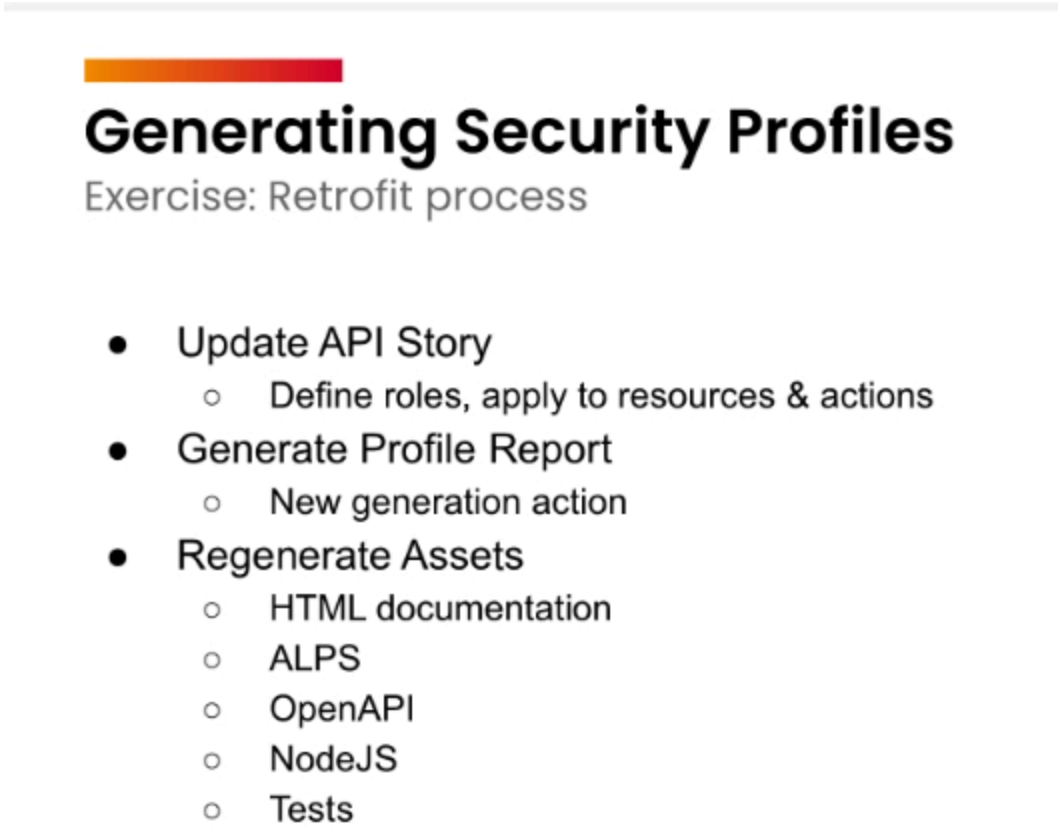
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/tree/main/04-alps-to-nodejs/task-management: Run tests

Generating security Profiles:

Generating Security Profiles

RBAC to the rescue

- Role-Based Access Control
  - RBAC
- Define Roles
  - `anon, user, admin`, etc
- Assign users to one or more roles
  - `user:mike, roles:user, admin`
- Secure API resources and actions
  - `/tasks/, /tasks/:id, addTask, updateStatus, etc.`



Updated story with security profile:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/output/tasks-api-story-security.md

Upload:
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/00-rbac-system-prompt.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/01-rbac-style-guide.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/02-rbac-generation-checklist.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/03-rbac-verification-checklist.md
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/output/tasks-api-story-security.md
Using context and updated story, generate the security profile.

O/p:

https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/rbac-map.json
https://github.com/LearningNewTechnology/2025-05-ai-apis-design-with-alps/blob/main/06-rbac-security-profile/output/tasks-api-security-profile.html