Lab 4: Spec Driven Development

Overview

In this lab, you will learn how to use Kiro's **Spec-Driven Development** feature to build complex applications through a structured, methodical approach. Unlike Vibe Coding's conversational style, Spec-Driven Development follows a formal software development lifecycle with clear phases and documentation.

What is Spec-Driven Development? Specs (specifications) are structured artifacts that formalize the development process for complex features in your application. They provide a systematic approach to transform high-level ideas into detailed implementation plans with clear tracking and accountability.

With Kiro's Spec-Driven Development, you can:

- **Break down requirements** into user stories with acceptance criteria using EARS notation
- **Build comprehensive design docs** with sequence diagrams and architecture plans
- **Track implementation progress** across discrete, manageable tasks
- **Collaborate effectively** between product and engineering teams
- **Maintain documentation** throughout the development process

What You'll Build:

- **AI-powered chatbot service** integrated with your existing e-commerce website
- **Python-based backend** using Strands Agents SDK and Nova Pro LLM
- **Independent microservice** with API integration to the main backend
- **Intelligent product recommendations** based on inventory data
- **Interactive chat interface** with action capabilities (add to cart, view products)

What You'll Learn:

- How to structure complex development projects using specifications
- Understanding the three-phase workflow: Requirements → Design → Implementation
- Working with AI agent frameworks and LLM integration
- Building microservices that integrate with existing applications
- Testing and validating AI-powered features

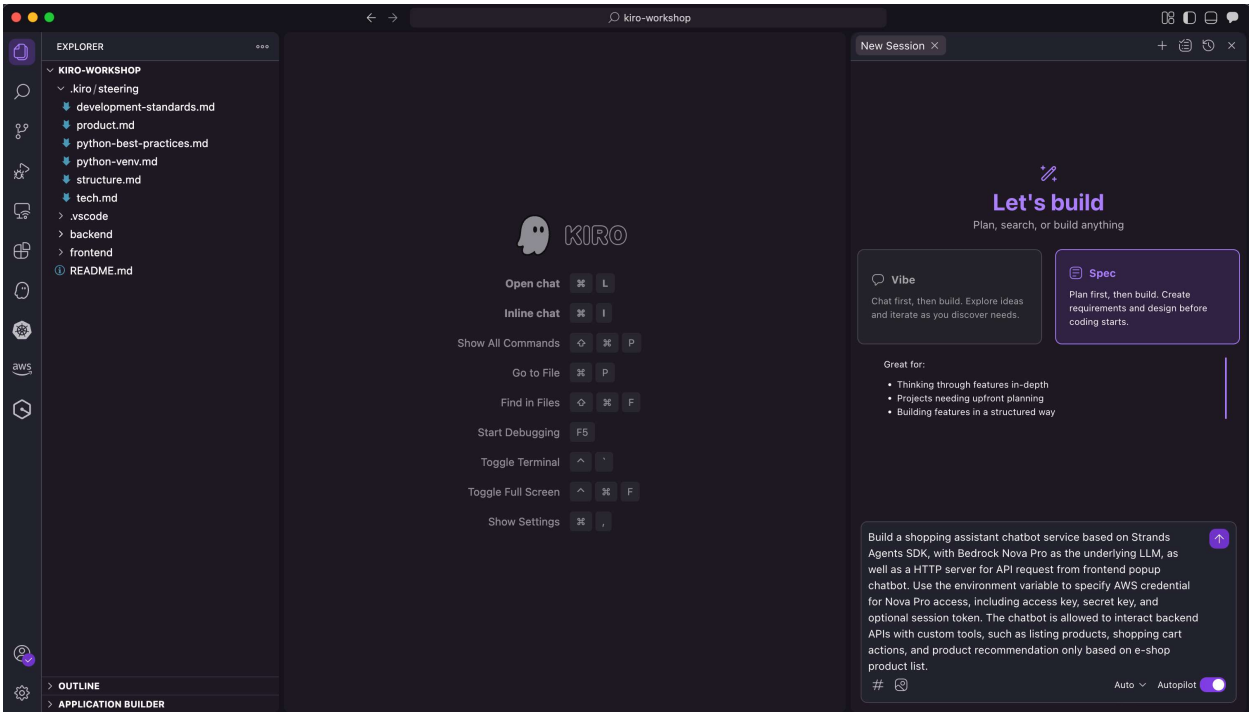
This lab demonstrates how Spec-Driven Development provides structure and accountability for complex feature development, ensuring thorough planning before implementation while maintaining clear progress tracking throughout the process.

Let's Build Chatbot with Spec-Driven Development

In Kiro's chat panel, open a new session. Choose **Spec** and send the prompt below:

```
Build a shopping assistant chatbot service based on Strands Agents SDK, with Bedrock Nova Pro as t                nd
Use the environment variable to specify AWS credential for Nova Pro access, including access key, secret

Setup a HTTP server for API request from frontend popup chatbot component to the chatbot service.
The chatbot is allowed to interact backend APIs with custom tools, such as listing products, shopping ca
```



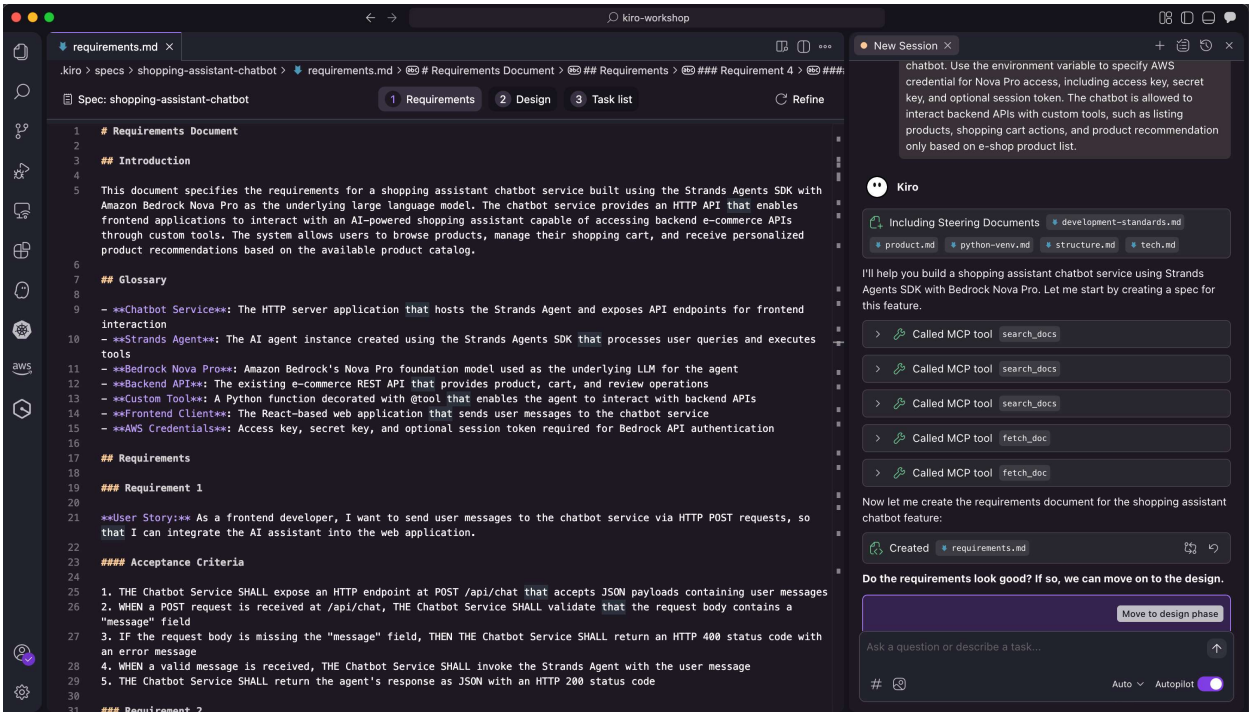
Kiro generates three key files that follow the three phase workflow through Requirements → Design → Implementation, to form the foundation of each specification:

1. `requirements.md` - Captures user stories and acceptance criteria in structured EARS notation
2. `design.md` - Documents technical architecture, sequence diagrams, and implementation considerations
3. `tasks.md` - Provides a detailed implementation plan with discrete, trackable tasks

1. Requirements

Simulate the SDLC Planning phase - The Development team collects requirements from several stakeholders such as customers, internal and external experts, and managers to create a software requirement specification document.

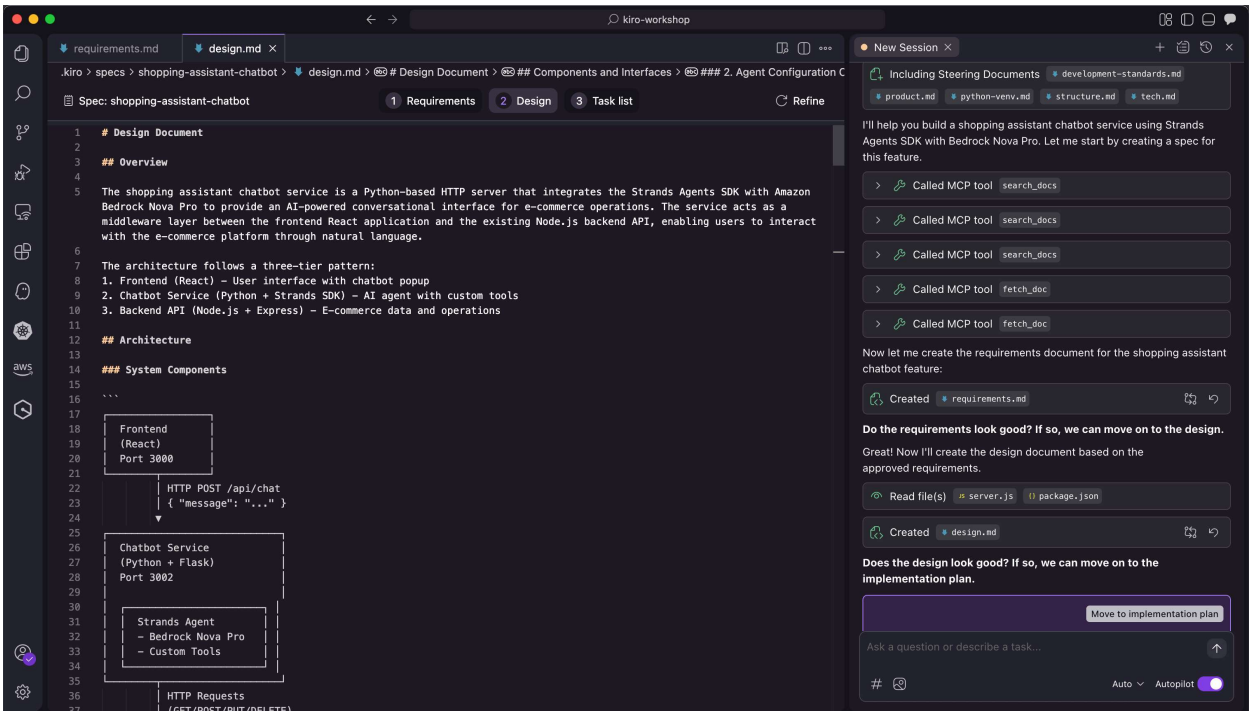
Review the requirements specification and modify if needed. Then click **Move to design phase**.



2. Design

Simulate the SDLC Design phase - Software Engineers analyze requirements and identify the best solutions to create the software. For example, they may consider integrating pre-existing modules, make technology choices, and identify development tools.

Review the design specification and modify if needed (eg. Nova Pro vs Nova Premier). Then click **Move to implementation plan**.

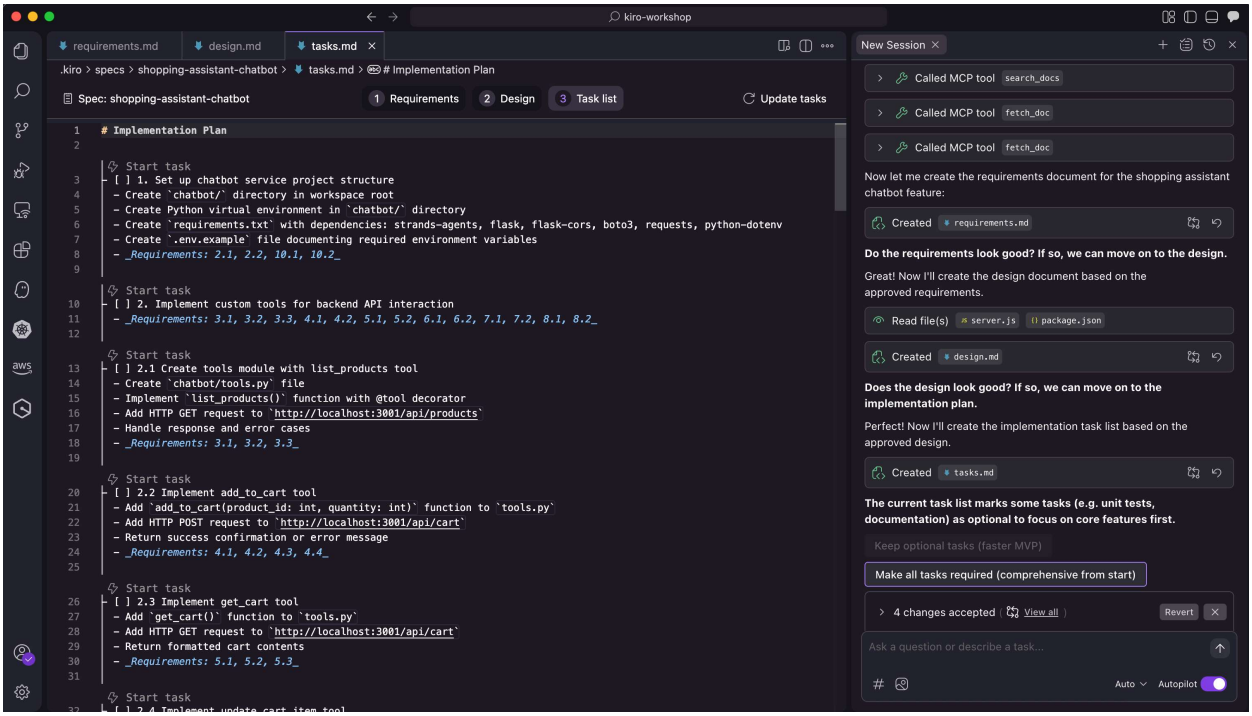


3. Implementation

Simulate the SDLC Implement phase - In the implementation phase, the development team codes the product. They analyze the requirements to identify smaller coding tasks they can do daily to achieve the final result.

Review the design specification and modify if needed. When task specification is generated, some tasks are marked as optional (e.g. unit tests, documentation) to focus on core features first. You can either **keep optional tasks** or **make all tasks required**.

In this workshop, choose **Keep optional tasks (faster MVP)** for efficiency.



Task Execution

In `task.md`, click ⚡ **Start task** button above each tasks one by one to execute the task, making necessary code changes and running commands. Or you can execute all the tasks in your `tasks.md` file by send the following prompt to Kiro:

```
Execute all tasks in the spec
```

> *Note: we do not recommend to execute all tasks at once as we recommend a task-wise execution to get better results.*

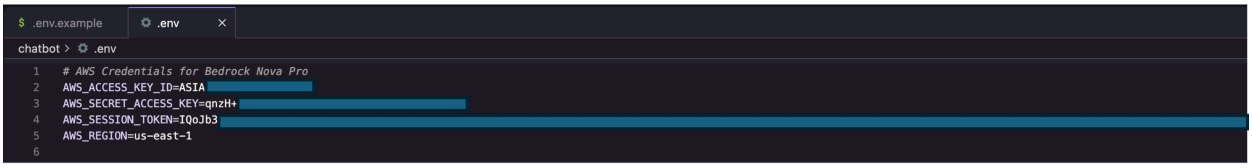
It takes around 20-30 minutes to complete all tasks. Once the Task Run are done, update AWS Credentials in **.env** file under chatbot service folder. If **.env** file doesn't exist, copy the **.env.example** file and rename to **.env** file for `python-dotenv` library to manage environment variable.

> *If AWS hosted event, go to **Workshop Studio Event Homepage** and obtain the AWS Credential in **Get AWS CLI credentials**. This will open a popup with your temporary AWS credentials that you can copy the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN` with value to `.env` file. Keep `AWS_REGION` or `AWS_DEFAULT_REGION` to **us-east-1**. ⚠ The temporary AWS credentials expire after 1 hour. You will need to obtain again and update the value in `.env` after expiry.*

For example:

```
AWS_ACCESS_KEY_ID=xxxxxxx
AWS_SECRET_ACCESS_KEY=yyyyyyy
```

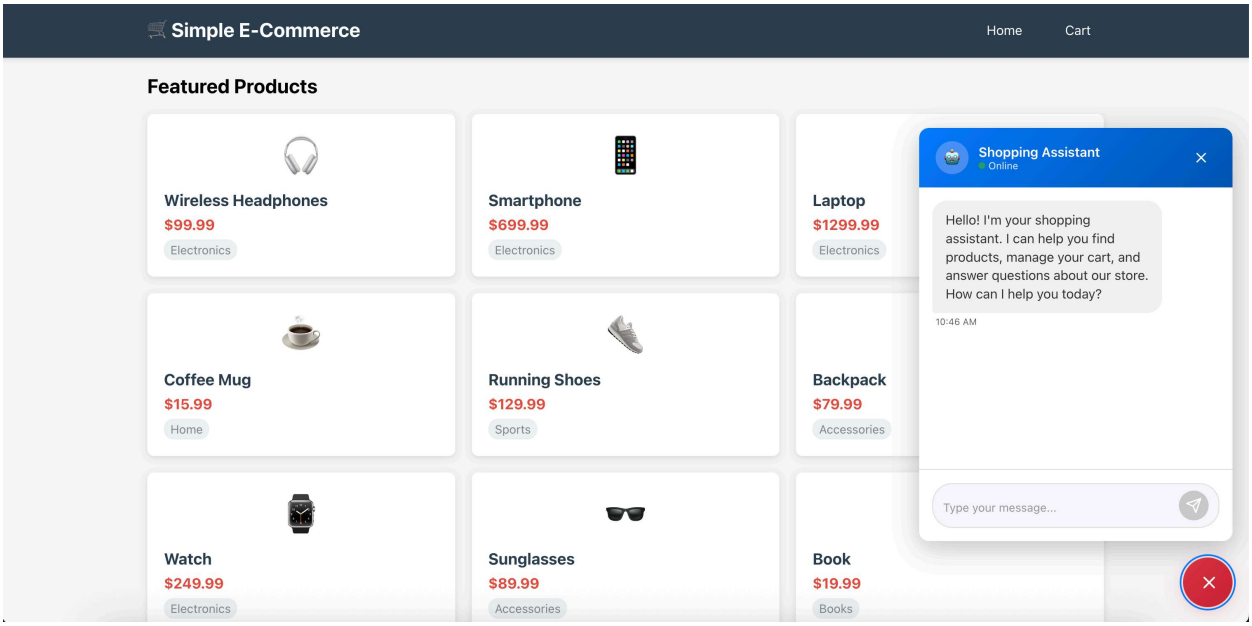
```
AWS_SESSION_TOKEN=zzzzzzzzzzzzzzzzzzzzz
AWS_REGION=us-east-1
```



Send the prompt below to start the E-commerce website with AI Agent Chatbot:

```
Start the chatbot service, backend and frontend if not running.
```

The E-commerce website should be opened via a web browser automatically. If not, open any web browser, browse `http://localhost:3000` or other port for React Frontend. You will access the E-commerce website similar to this:

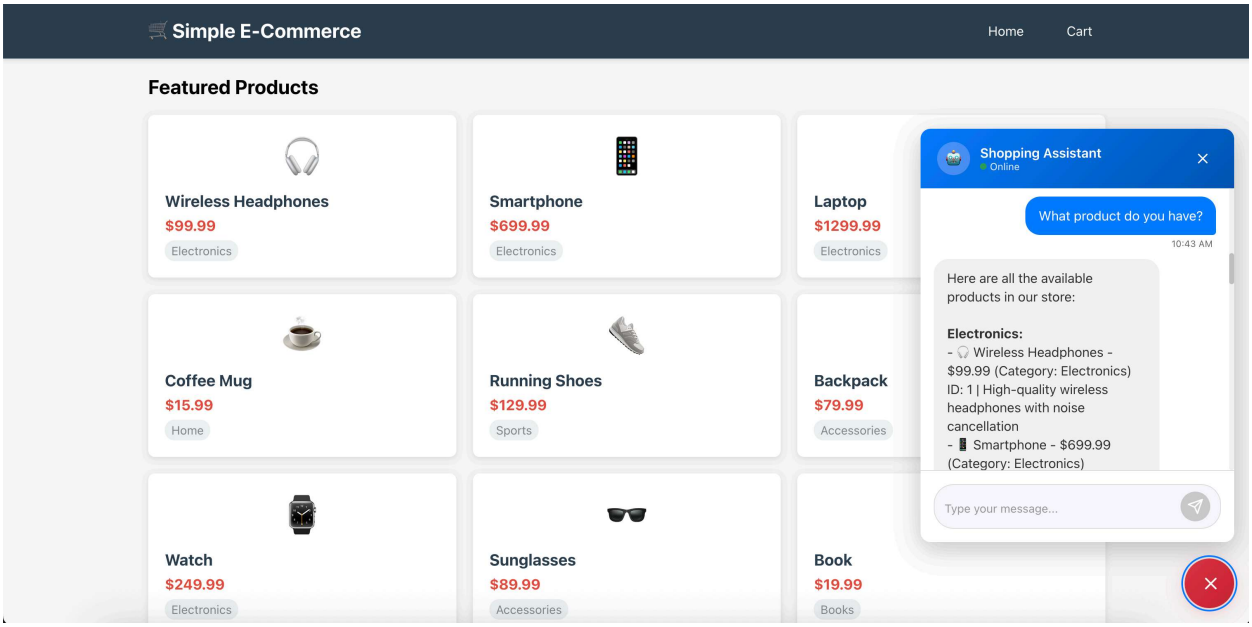


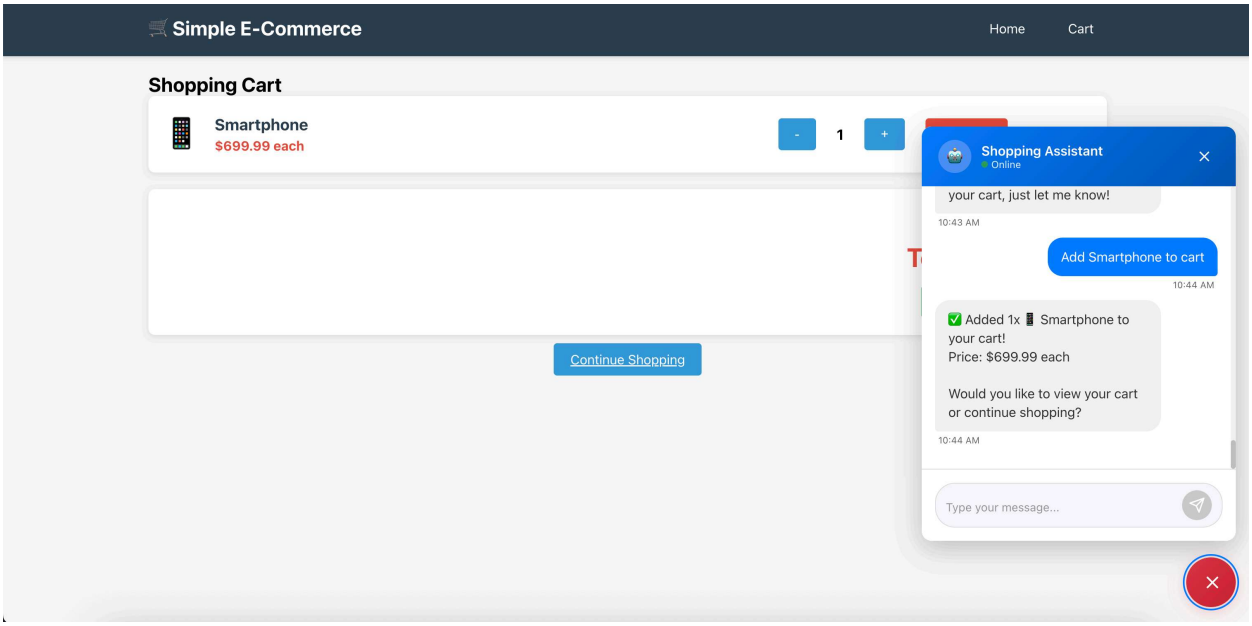
Testing the Chatbot

Once your e-commerce website is running, test the chatbot with the following questions:

1. `Hello`
2. `What products do you have?`
3. `Add <product_name> to cart`
4. `Show me the cart`
5. `Clear my cart`
6. `What do you suggest to buy for hiking?`
7. `Add all suggested items to cart`

If not work, describe the problem as prompt and send to Kiro for debug. Here are some common issues for reference.





Troubleshooting Common Issues

Chatbot icon not appear

If chatbot icon doesn't appear in frontend, send this prompt to Kiro:

```
Chatbot icon doesn't appear in frontend.
```

No response / dummy response in Chatbot

If you encounter problems with the functionality, for example no response from the chatbot, send this prompt to Kiro:

```
Chatbot has no response or provide dummy response after I ask question in the frontend.
```

Wrong action from Chatbot

If you encounter problems with chatbot action,for example replying wrong product list or no action to shopping cart, send this prompt to Kiro:

```
Chatbot list the products out of the inventory in this e-commerce website.
```

```
Chatbot doesn't add the product to shopping cart from my request.
```

No update in Shopping Cart page

If you encounter problems in frontend rendering, for example no re-rendering in shopping cart page after chatbot add a product to cart (try to refresh browser to see if rendering problem), send this prompt to Kiro:

```
Shopping cart page doesn't update automatically after chatbot helps to add a product in cart.
```

Application Won't Start

If the application fails to start, try:

```
The application is not starting properly.
```

Vibe vs Spec Tips

When to Vibe

1. **Interactive Q&A Format**: Vibe sessions are optimized for back-and-forth conversations about code, allowing you to ask questions and get immediate responses.
2. **Quick Assistance**: They're ideal for getting quick answers to coding questions, explanations of code behavior, or understanding concepts without going through a formal specification process.
3. **Contextual Understanding**: Like other Kiro sessions, Vibe sessions leverage context providers to understand your codebase, but with a focus on explanation rather than extensive code generation.
4. **Flexible Approach**: Vibe sessions offer a more fluid, less structured approach compared to Spec sessions, making them suitable for exploratory coding and learning.

When to Spec

1. **Complex Development Tasks**: Use Spec sessions for building complex features, entire applications, or significant refactoring that requires careful planning and execution.
2. **Structured Approach**: When you need a methodical, step-by-step approach to development with clear documentation of requirements and implementation details.
3. **Team Collaboration**: For projects where multiple team members need to understand the implementation plan and track progress against specifications.
4. **Documentation Needs**: When you want to generate detailed documentation alongside your code implementation for future reference or knowledge sharing.

Previous Next