



# PartSelect E-commerce Chat Agent

A Specialized Assistant for  
Refrigerator and Dishwasher Parts

Eugenia Bornacini

# Project Overview

## Project Goal:

- Develop a specialized chat agent for PartSelect e-commerce website

## Focus Areas:

- Refrigerator and Dishwasher parts

## Key Functionalities:

- Provide accurate product information
- Assist with customer transactions

## User Experience:

- Intuitive chat interface in Chrome's side panel
- Context-aware responses for natural conversation flow

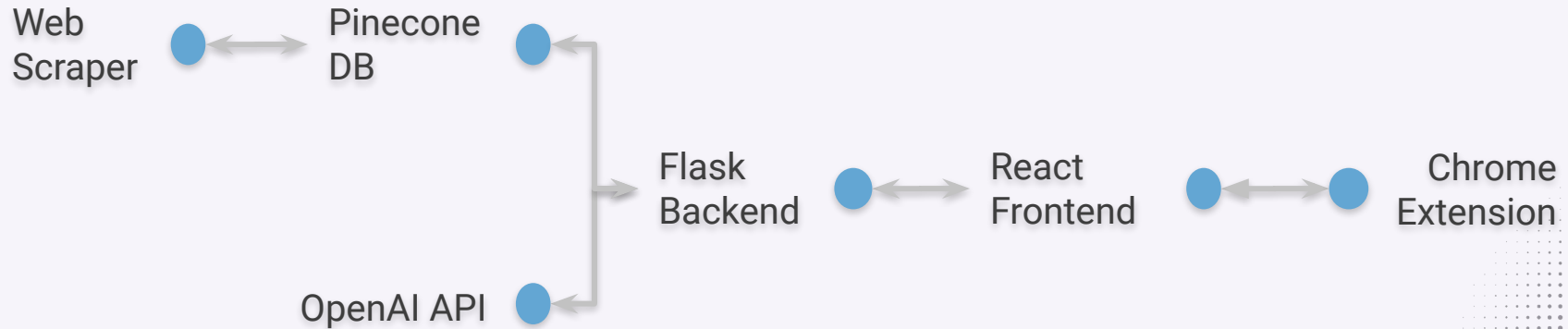
## Technical Highlights:

- Chrome extension frontend for seamless user interaction
- Flask backend with OpenAI integration for natural language processing
- Pinecone vector database for efficient product data storage and retrieval
- Web scraping for up-to-date product information

## Extensibility:

- Modular architecture allowing easy expansion to other product categories
- Scalable database solution for growing product catalog

# Architecture Overview



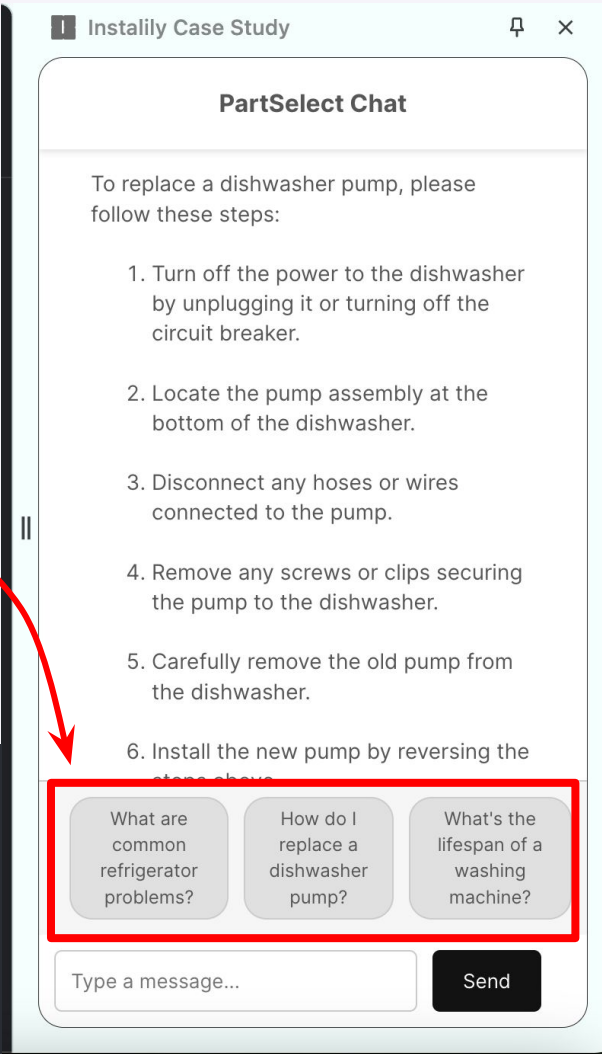
# Frontend Architecture

Built upon provided React frontend as a starting point

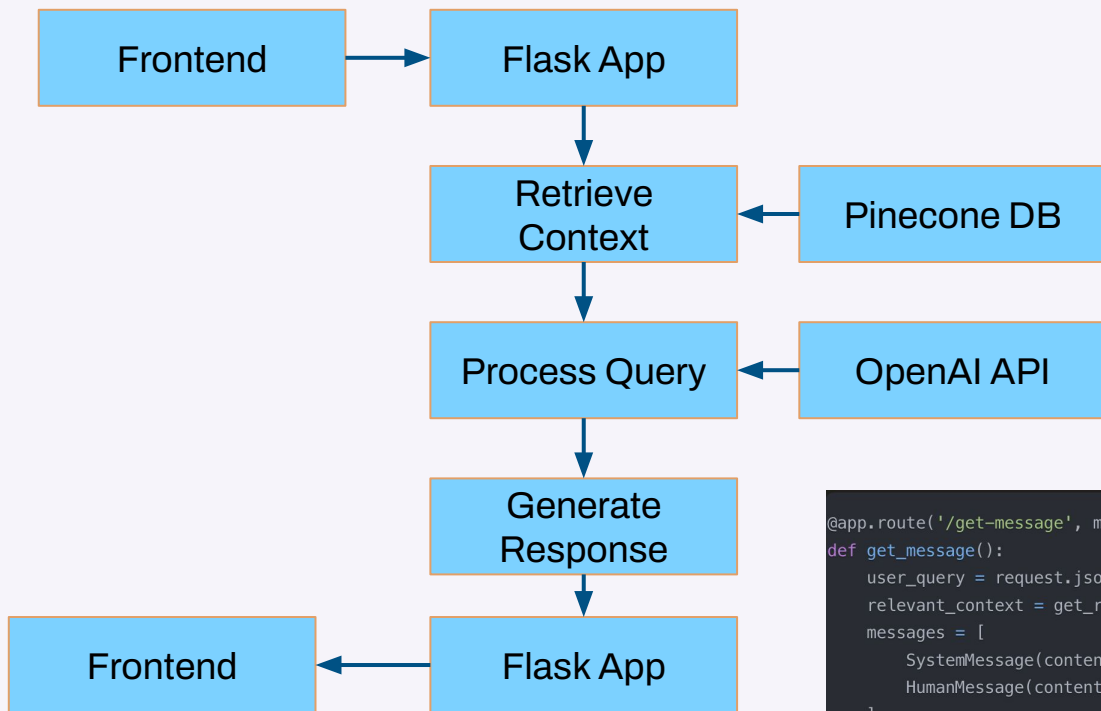
Key Changes:

- Changed name to reflect PartSelect focus
- Added suggested questions for user engagement
- Backend Integration:
  - api.js facilitates simple connection to Flask app
  - Implements error handling for robust user experience

```
export const getAIDMessage = async (userInput) => {
  try {
    const response = await fetch('/get-message', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ query: userInput }),
    });
    if (!response.ok) throw new Error('Network response was not ok');
    return await response.json();
  } catch (error) {
    console.error('Error:', error);
    return { role: 'assistant', content: 'Sorry, an error occurred.' };
  }
};
```



# Backend Architecture



```
@app.route('/get-message', methods=['POST'])
def get_message():
    user_query = request.json.get('query', 'No query provided')
    relevant_context = get_relevant_context(user_query)
    messages = [
        SystemMessage(content="You are an expert sales representative..."),
        HumanMessage(content=f"Context: {relevant_context}\n\nUser query: {user_query}")
    ]
    response = llm(messages)
    return jsonify({"role": "assistant", "content": response.content})
```

# Data Management

## Pinecone Vector Database:

- Scalable
- Low latency
- Easy API Integration

## Web Scraping:

- Scope: Focused on refrigerator and dishwasher parts
- Configurable scraping depth (default: 0)
- Efficient

## Scraping Considerations:

- Respects website's robots.txt and rate limits
- Extracts structured data for consistent storage
- Manual periodic updates; room for expansion on update frequency

## Trade-offs and Optimizations:

- Limited initial dataset for quicker setup and testing
- Prioritized response quality over quantity
- Potential for depth increase to expand knowledge base

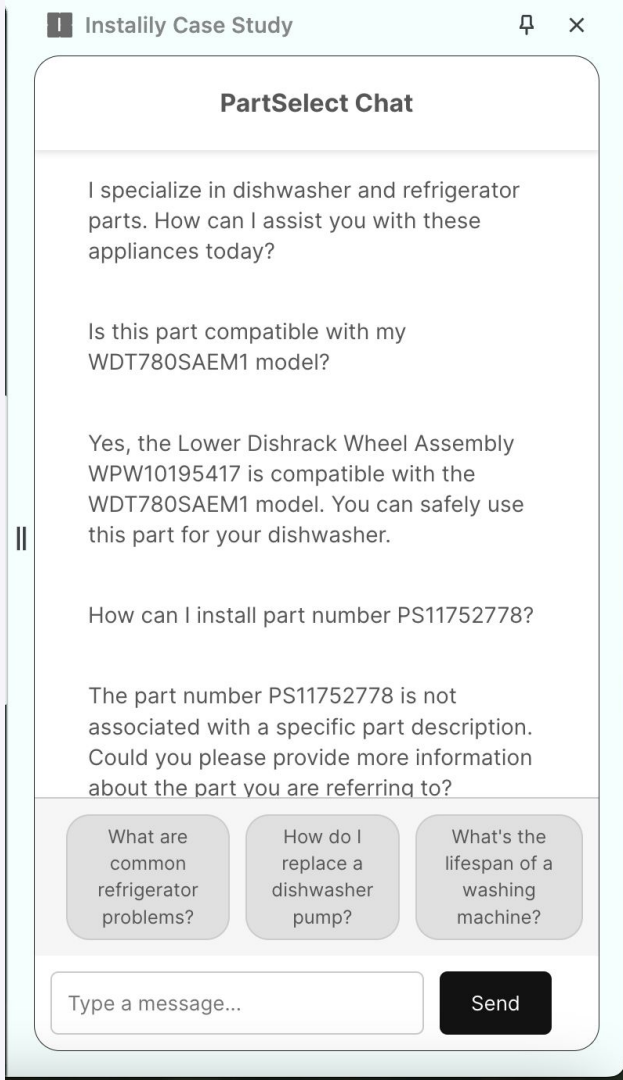
## Data Flow:

- Enables semantic search and relevant context retrieval



# RAG Pipeline

- OpenAI Integration
- Langchain Framework
  - Future maintainability
- Context Retrieval with Pinecone
  - Semantic search
- Prompt Engineering
- Conversation Memory



# Extensibility and Scalability

## Modular Architecture

- Separation of frontend, backend, and data storage
- Potential to extend to other product categories beyond refrigerators and dishwashers

## Scalable Database Solution

- Pinecone vector database can efficiently handle millions of product entries
- Real-time updates allow for dynamic product information management

## Frontend Adaptability

- Chrome extension can be adapted for other browsers
- Potential for developing standalone web or mobile applications

## Flexible AI Integration

- Langchain framework allows easy switching between AI models; adaptable to future advancements in natural language processing

## Expandable Web Scraping

- Scraping depth can be increased to cover more products
- Framework in place to add new product categories

## Backend Scalability

- Flask application can be scaled horizontally
- Potential for containerization (e.g., Docker) for easier deployment and scaling
- Can implement load balancing for handling increased traffic



# Future Expansion

## Routing Prompt

- Implement an initial classification layer to categorize user queries

## Automatic Scraping

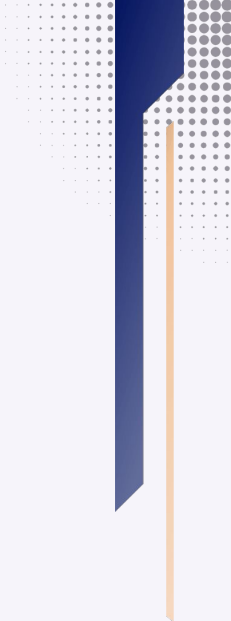

- Develop a scheduled, automated web scraping system

## +Product Categories

- Extend beyond refrigerators and dishwashers to cover all PartSelect offerings

## Prompt Refinements

- Continuously improve and specialize prompt engineering



# Q&A