WebWatch

(Formerly Cron Jobs as a Service)

October 14th, 2024

Team name: Cass

Team members

- Alex Kekchidis
- Cameron Candau (Project Owner)
- Isaac To (Scrum Master)
- Jordan Nguy
- Leonardo Gallego
- Simon Zhao

WebWatch

The web is constantly changing, too much, too often.

We automate the checking so the client doesn’t have to.

Who benefits from our solution?

Anyone who **places worth on their attention and productivity**, but **doesn’t want to miss out** on key events will find value in our solution

Project Scope

- Provide a solution that **monitors a web page for changes**
  - Notifies the user (by **email**, Discord, Slack, or other integrations)
- Create a **web app** to improve ease of use
  - Removes the need for users to figure out installation, configuration, and hosting
- Allow users to **choose specific content** within a page to monitor, rather than the entire page

Sprint 1

User Stories

- As a user, I would like to receive an email notifying me that a webpage’s content has changed. (2)
- As a user, I would like a modular CLI service to interact with the backend. (2)
- As a user I would like to be able to scrape dynamic web-pages to get more accurate monitoring results with modern web pages. (3)
- I would like to receive notifications through multiple channels (email/discord/slack/etc) for detections. (2)

Spikes

- Research options for building the project’s core functionality:
  - Task scheduler
  - Web scraper
  - Web content storage
  - Diff functionality

Infrastructure tasks

- Create GitHub Organization
- Create a development container image to ensure consistent behavior throughout testing

Sprint 2

User Stories

- As a non-technical user, I would like a web interface to interact with the scheduler, so I don't have to install and run programs locally. (5)
- As a user, I would like to be able to log in with external authentication providers, like signing in with Google (OpenID/OAuth). (5)

Spikes

- Designing a user interface for the web app
- Learning React, TailwindCSS

Infrastructure Tasks

- Virtual Private Server for web app hosting
- Set up Figma for shared, real-time UI designing with team

Sprint 3

User Stories

- As a user, I would like to be able to see what changed on the webpage and selectively choose what parts are monitored, to reduce “noise” generated by other parts of a page changing which I’m not concerned with. (8)
- As a user, I would like to be able to track the prices on online marketplaces so I can buy things cheaply. (5)

Spikes

- Research how we can
    - selectively monitor elements user chooses with the web scraper
    - track price changes if a user chooses to

Infrastructure Tasks

- N/A; core functionality relies on changing our code, but no new infrastructure

Sprint 4

User Stories

- As a user, I would like to have the option to wire together my own custom action blocks in order to have the scheduler perform specific tasks I define. (21)
- As a user, I would like to have the ability to create and share templates of my custom action workflow. That way, other users can extend my workflow and further customize it. (3)

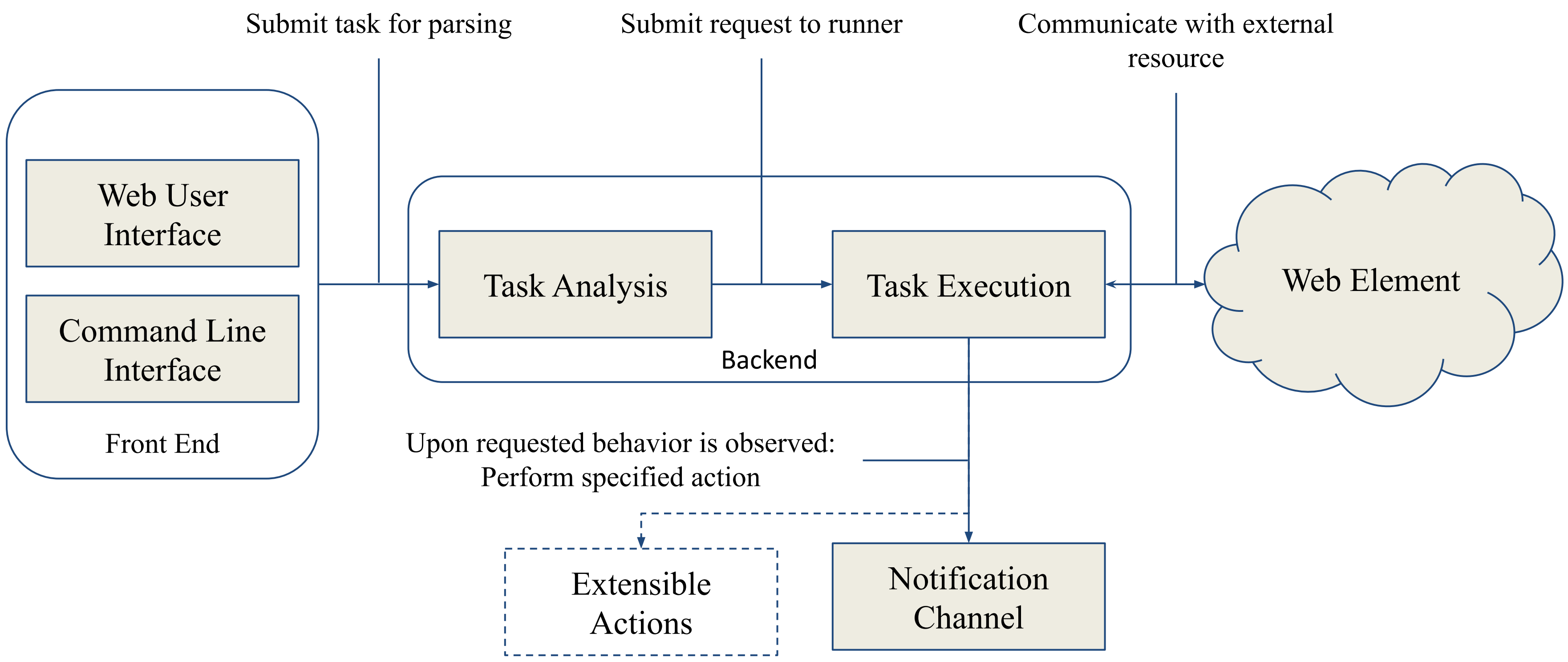
Spikes

- Plan backend and frontend additions that allow for creating custom workflows
- Build more integrations to be ran in any order

Infrastructure Tasks

- Rework code where necessary to improve modularity and allow for workflow customization

Architecture



Technologies

DevOps:

- Containerized Deployments
- Consistent Development Environments

Notification Avenues:

- SMTP - Email
- Webhooks - Service Integration

Frontend - React.JS

Backend - Python

- Selenium - Browser Automation
- Celery - Task Scheduling

Challenges/Risks

CAPTCHAS / Anti-Bot software

Many websites use CAPTCHAs and since the service is fully automated it is very **possible that behaviors performed can be flagged and prevented from accessing the requested data**

Do we **bypass this** (and how) or **respect the anti-bot sentiments**?

Challenges/Risks

Limiting malicious behaviors from client requests and **minimizing liability**

Imagine a scenario where the client instructs the service to request malicious code from a website or is used to perform a DDOS attack on a server

How do we **prevent** presenting opportunities like this?

Minimum Viable Product (MVP)

Automated web page monitoring and notifies the user through some notification channel