The product is a web application that interacts with a fake store API to enable the end-user to find clothing products. The application is targeted towards cost-conscious shoppers who do not want to potentially be tempted by sales and advertisements to buy more than they intended to. What makes the product unique as opposed to other shopping interfaces is its minimalism since it will not include targeted advertisements or products that weren't directly searched for.

The application will be hosted on Github, and will use Github Actions for continuous integration. The code will be written in Python and it will use Flask as a web framework. The web application will be deployed using Heroku. The application also uses SQLAlchemy which is used as an object relational mapper for a SQLite3 database.

Unit & integration tests using mock objects, patching and test doubles

Production monitoring using ScoutAMP as a Heroku addon

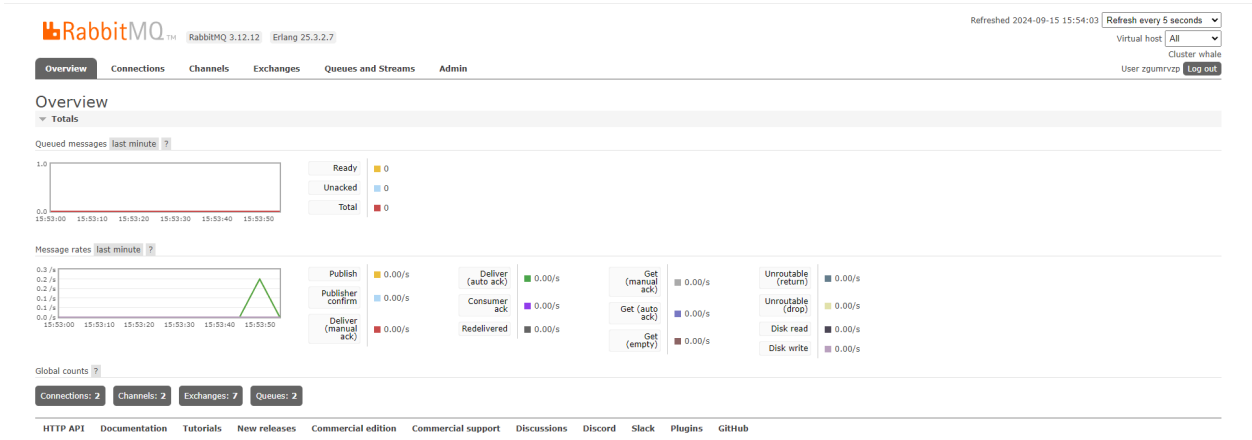
Event collaboration messaging using CloudAMQP as a Heroku addon

Heroku was chosen as a cloud platform over larger platforms such as Azure or AWS because it was easier and cheaper to host a small application on those with little setup required. If the application had to be scaled up then switching to a larger cloud platform would most likely be required.

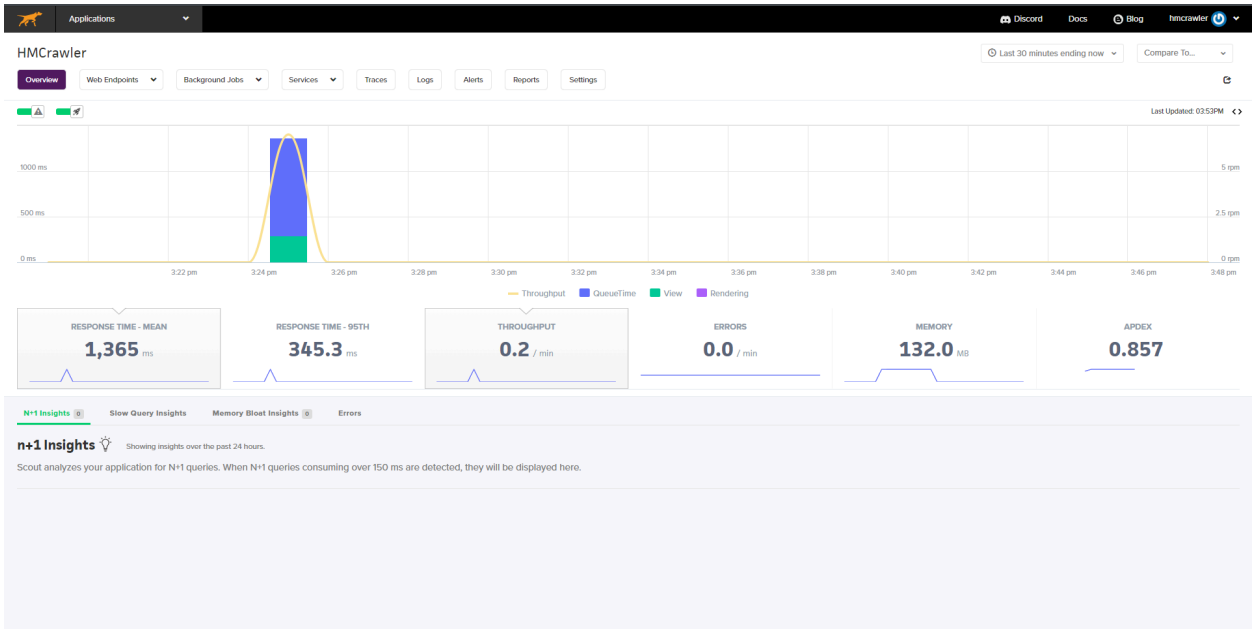
Github was chosen as a repository host because it has better integration with Heroku over something like Azure Devops and allowed access to Github Actions as an integration tool. Flask was chosen over something like React purely because I wanted to learn how to use Python better, in my opinion both web frameworks are good for creating web applications. SQLite was used because it is very lightweight and requires a lot less installation than other database options, it is also easier to port over to something like Github Actions or Heroku to be tested over something like MySQL. SQLAlchemy was used because it integrates well with Flask and saves time having to manually create SQL scripts, it also makes the code more readable since it wouldn't be filled with paragraphs of SQL.

Scout AMP and CloudAMQP were chosen as monitoring and message queueing systems simply because they were already available as addons on Heroku and were free which would save time and money for me. I don't have much experience with monitoring and messaging systems so I have no way to compare the services outside of that.

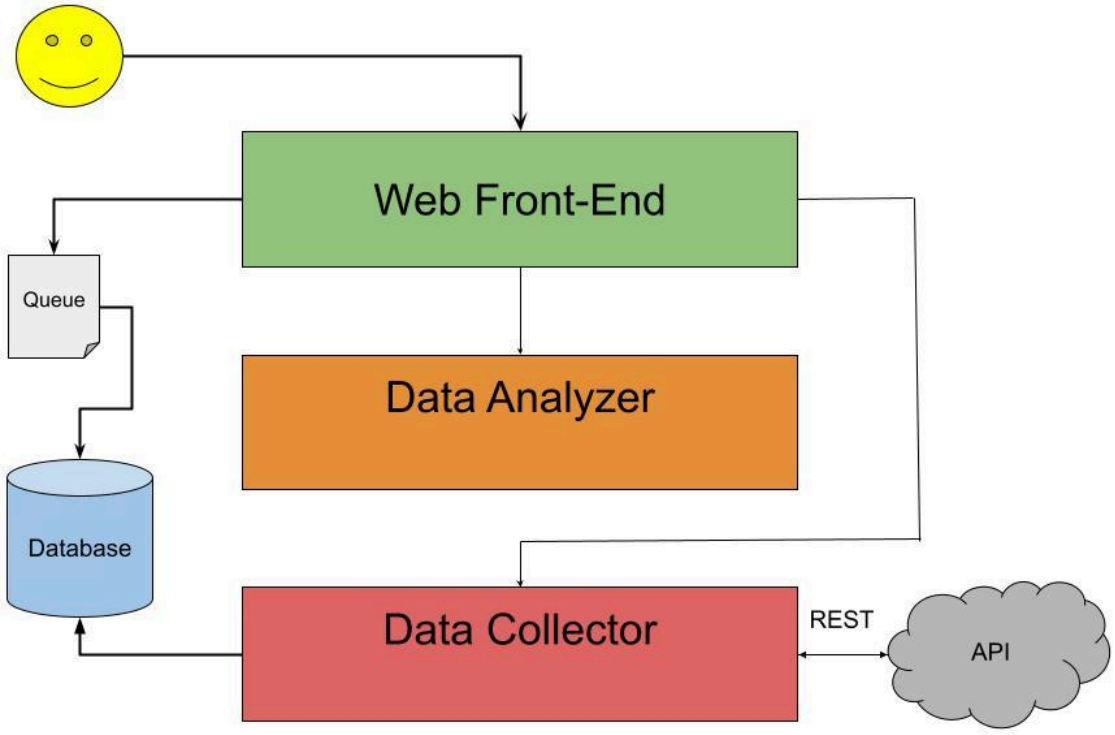
RabbitMQ provided through CloudAMQP:



Scout APM monitoring:



White Board architecture:



It can be inferred that the system requirements of the application would be the same as the system requirements for a browser such as Google Chrome, which are Windows 10 or later or Windows Server 2016 or later and an Intel Pentium 4 processor or later that's SSE3 capable.