

Project Milestone 4

Revised List of Features

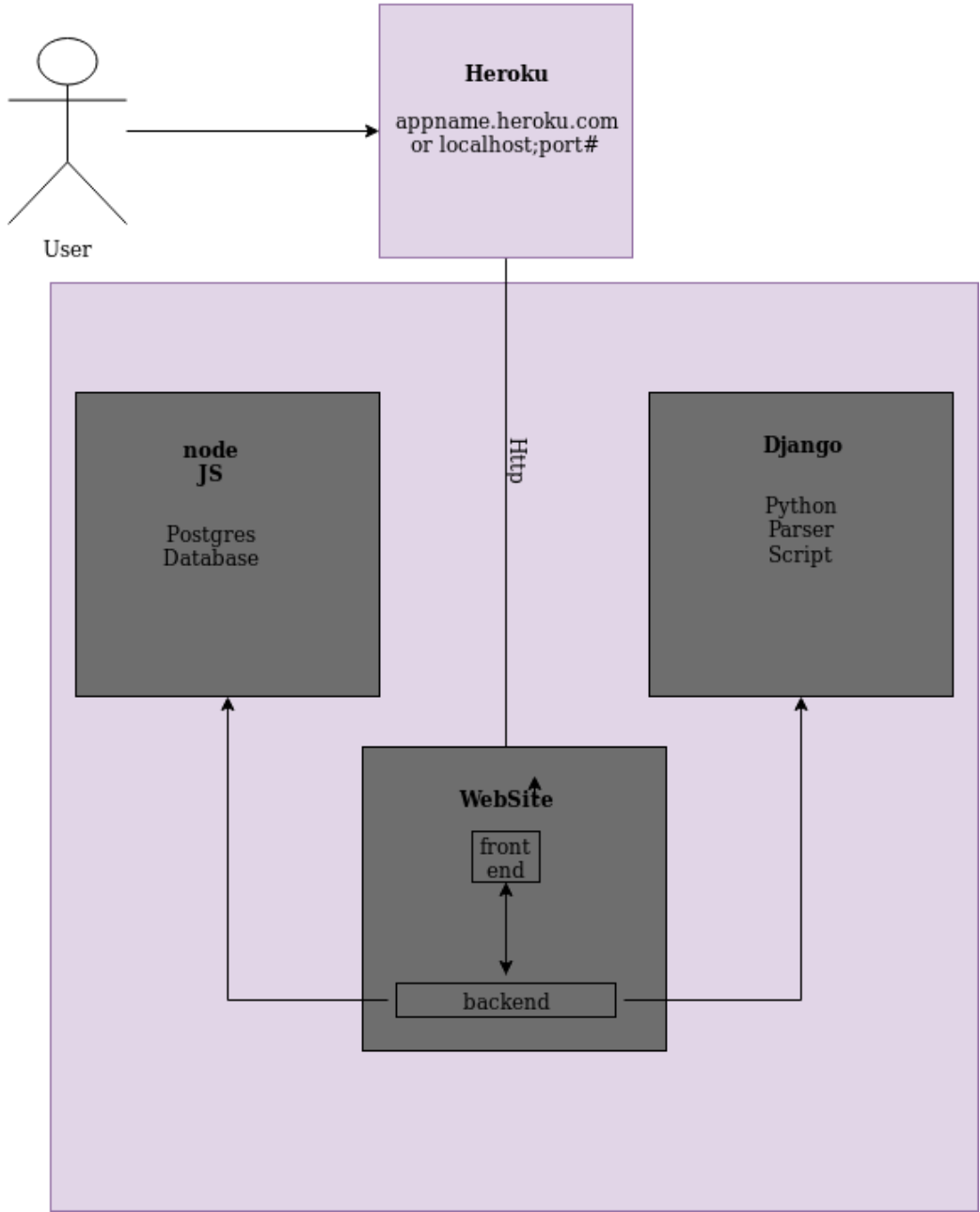
- **Ability to submit a webpage and analyze GUI Breakdown, Metadata** *(Priority 1, must get this first)*
 - Done through Python script *(Priority 1, but almost done!)*
 - Must figure out how to call the script on web and pass data to javascript
 - Displayed using JS, HTML *(Priority 1, needs JS + more CSS work)*
- **Login + SQL database that stores queries** *(Priority 2, can complete this iteratively, but still required)*
 - Storing User Data in a SQL server *(Priority 2, can complete this iteratively, but still required)*
- **Ability to retrieve information about your previous queries** *(done using the previous feature)*
- **Ability to parse multiple requests** *(can compare URL's on the "results" page) - (Priority 1/2 - is important to have, but something we should implement *after* we get it working + displaying with just one. We need to consider this as we continue to iterate on our design and make it dynamic, both from a front-end and back-end standpoint)*
- **Export Data in one (or more) formats** *(Priority 2, same as before, but should be worked on iteratively once the other most important things are done)*
- **(Stretch goal) Share Data in one (or more) formats** *(Priority 4, not necessary, but a stretch goal if we have time. Might be tricky linking this to twitter/emails and stuff)*

Architecture Diagram

<https://drive.google.com/file/d/1PI3xpbJXVJhBWJDuJxNWu-aod23Zgfge/view?usp=sharing>

We will be hosting the website using Heroku. On the database will be postgres on a nodeJS server and the python parser will be hosted on the same nodeJS Server with Django. Heroku will also hold the frontend/backend web page design and infrastructure.

**Architecture
Diagram**
Project Milestone
4-200-2

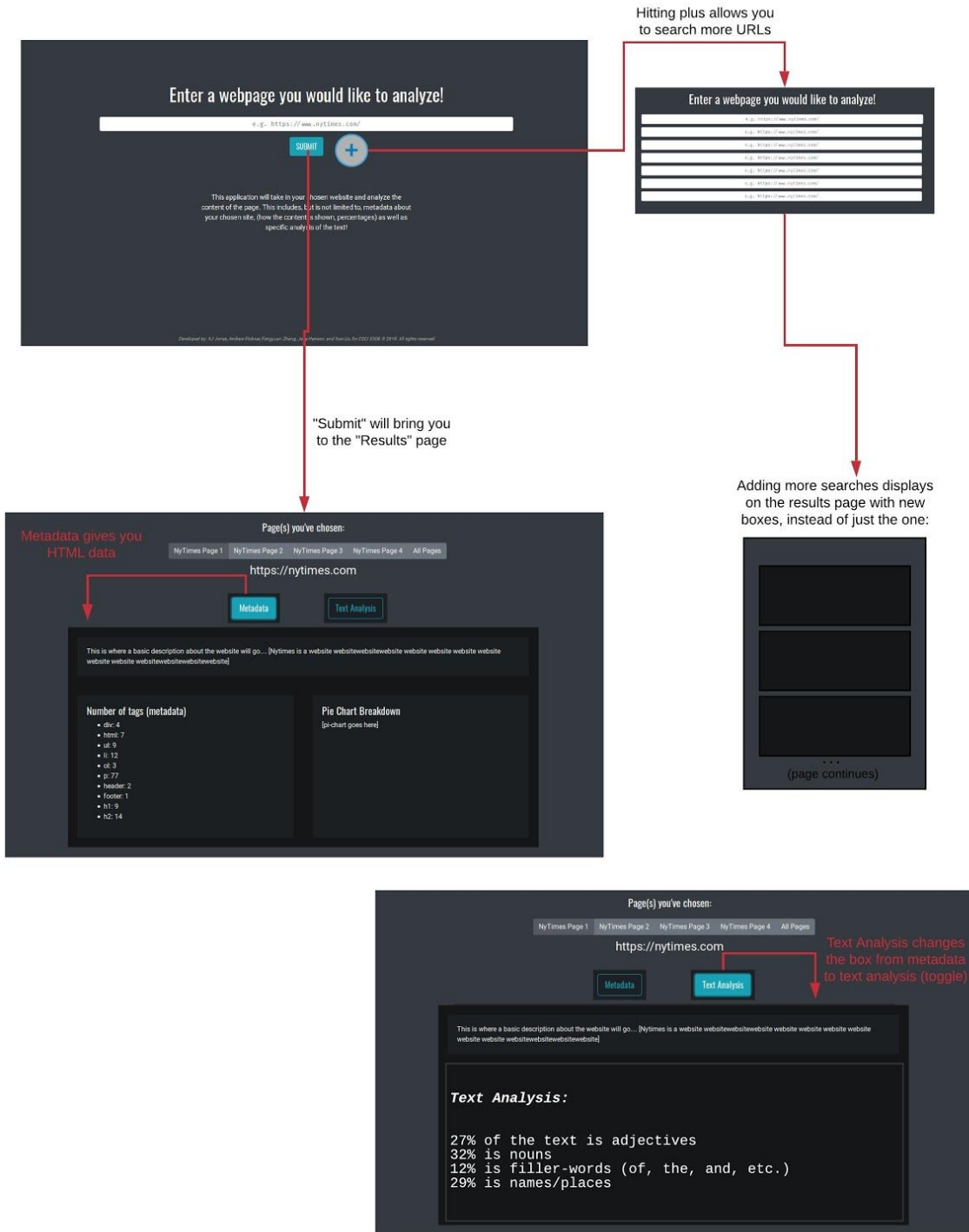


Front End design

<https://www.lucidchart.com/invitations/accept/769d9d4a-955e-47eb-8b88-bb99f1d7dbc3>

Front-End Design

Project Milestone 4 - 200-2



Web Service Design

Our app uses several smaller API's to pass the data around and create the useful displays and analysis of the webpage.

Requests: “allows you to send HTTP/1.1 requests extremely easily. There’s no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to [urllib3](#). (From [Requests](#) docs.)”

This library allows me to send a request via HTTP to go out and gather the page data from a given URL. If successful, the `get()` method returns the response from the request, if not successful we can raise an exception. The data this API receives is the URL the user wants to analyze. Then, this API outputs the actual html page.

Beautiful Soup: “is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. (From [Beautiful Soup](#) docs.)”

This library allows quick manipulation of the DOM so I can obtain data about the tags and about the actual text rather quickly. The data I pass to this API is the html page that I’ve requested using the API above. This API outputs the breakdown of html tags on the page, and the actual text of the page.

NLTK: “is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. (From [NLTK](#) docs.)”

I pass this API the raw text from the page, and allow the power of NLTK to tokenize and process this text. With this text processed and analyzed, I can output this data from this API. This API handles all text analysis, and that’s it pretty much it.

Database design

This deliverable provides a summary design of your application’s database.

A small example to show the relation between database and Login page:

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
#User's ID	Username	password	info: register Date	Occupation	purpose	feedback_1	feedback_2	
1	Mike	*****	10/31/19	student	for fun	4	Add more functions for sharing	
2	ezLife	*****	11/1/19	Data scientists	data research	5	cool	
3	GG	*****	11/5/19	Instructor	HW	/	No comment	
4	qwerty	*****	11/2/19	student	IDK	3	easily use but a little simple	

Entities(records)

Key Field

Attributes(fields)

Welcome to the Login Page

LOGIN

Not registered? [Create an account](#)