NoCap: Fact Checking with AI Milestone 3

Anthony Ciero, Joshua Pechan, Varun Doddapaneni, Thomas Chamberlain
Faculty Advisor: Professor Silaghi

Milestone 3 Matrix

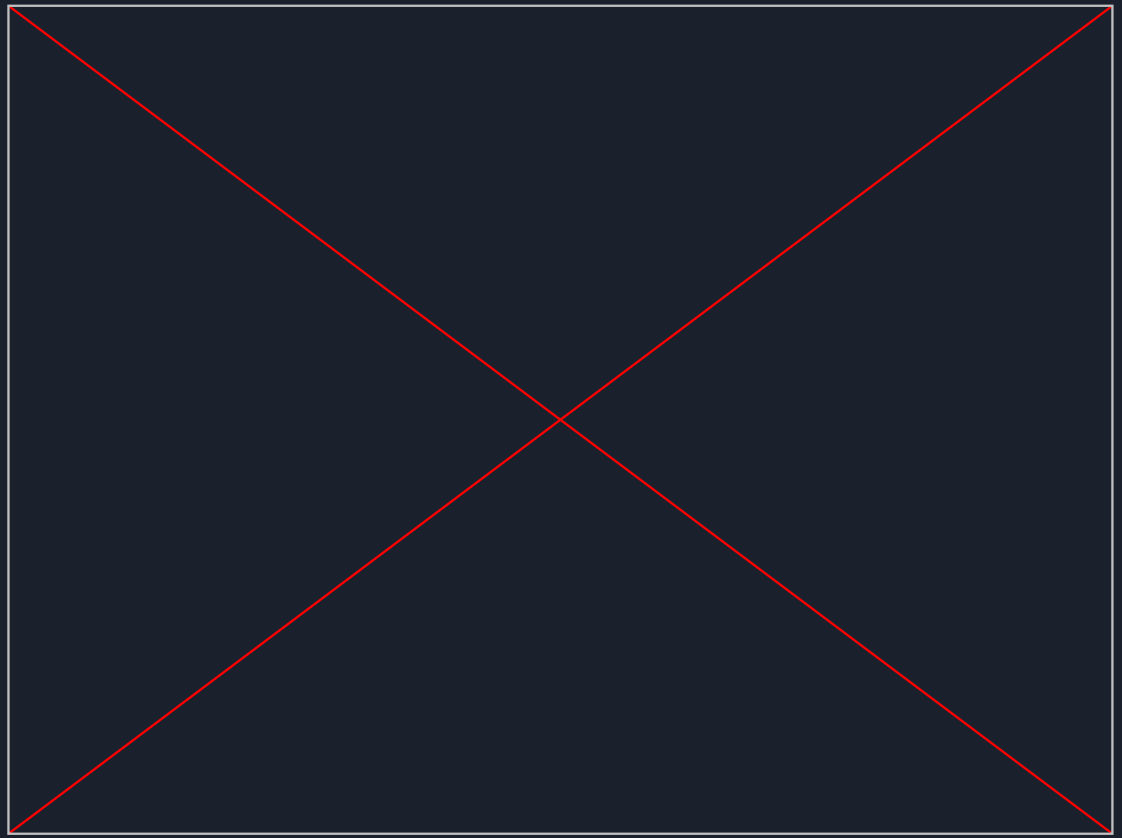
Task	Completion	Thomas	Anthony	Josh	Varun
1. Prompt engineering	75%	20%	0%	20%	60%
2. Get a basic score of an article	100%	0%	0%	0%	100%
3. Start process to break text down into tokens	100%	0%	0%	50%	50%
4. Develop the backend database	80%	0%	0%	100%	0%
5. Article Report/Publisher Cards	100%	0%	50%	50%	0%
6. Article meta data connection	50%	15%	15%	70%	0%

Prompt Engineering

- Consistent Grading - by telling the model how to judge a website step by step, we can get close to repeatable scores
- We can outline the criteria and define what “bias” actually means
- By structuring the prompt, we get back both a numerical score and its bias report
- Gives the model direction, without a understandable prompt, the AI model would just output whatever it feels like, our prompt forces it to look at: bias, tone, claims, and evidence.

```
system_instruction = (
    "You are a strict fact-checking assistant. Your job is to evaluate how factually "
    "accurate an article is. You should:\n"
    "1. Check for logical consistency.\n"
    "2. Note any claims that contradict widely established facts.\n"
    "3. Ignore writing tone and style. Only judge factual correctness.\n\n"
    "Return ONLY a JSON object in exactly this format:\n"
    "{\n"
    '  \"score\": <integer between 0 and 100>,\n'
    '  \"explanation\": \"<1-3 sentences explaining the score>\"\n'
    "}\n"
    "Do NOT include any additional text outside this JSON."
)

user_prompt = (
    "Evaluate the factual accuracy of the following article text:\n\n"
    f"{article_text}\n\n"
    "Reply ONLY with the required JSON."
)
```

Basic Article Score



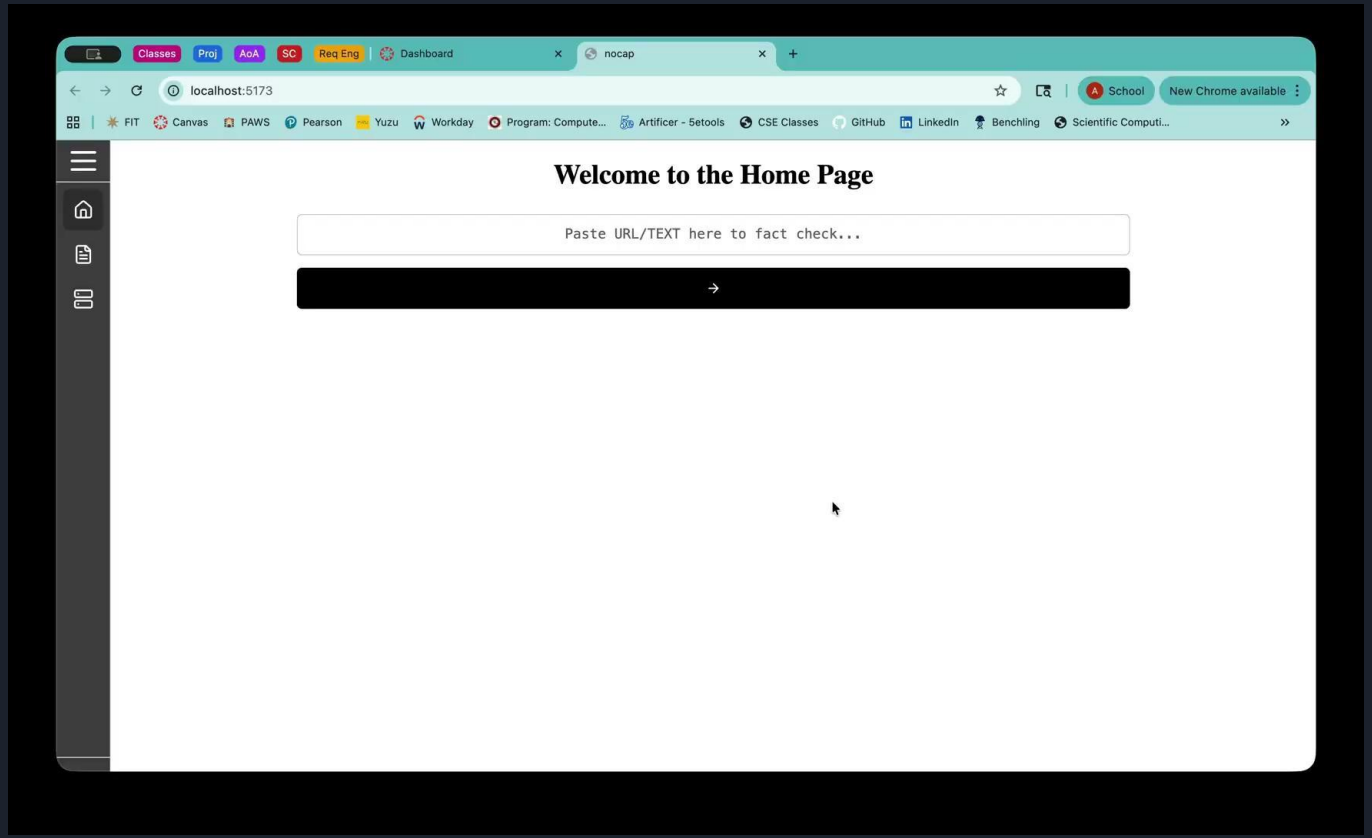
Break down text into tokens

- We can scrape the content of an article
- The text content is then broken down and fed to the model

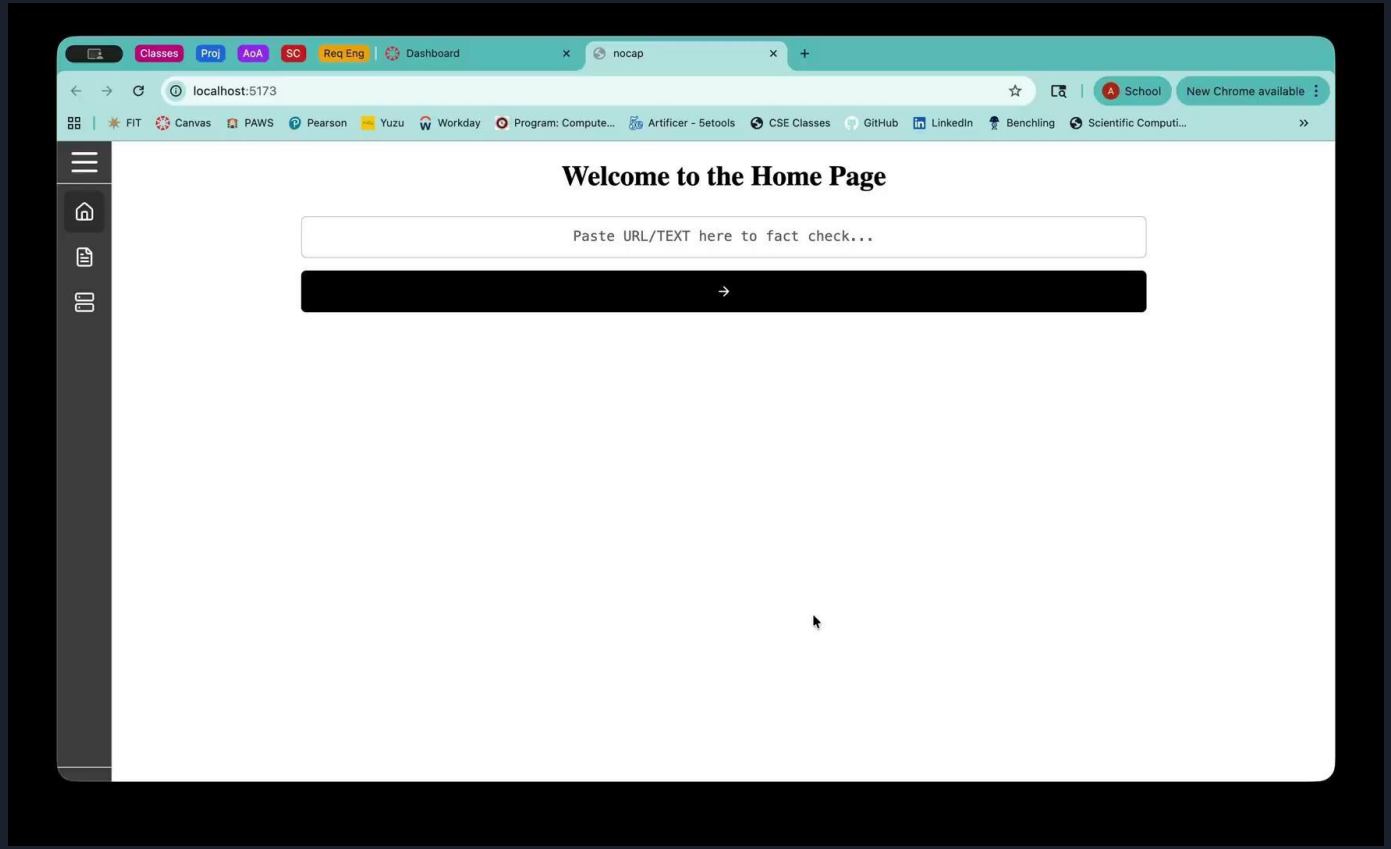
Backend Database

```
import { type ClientSchema, a, defineData } from '@aws-amplify/backend';

const schema = a.schema({
  Reports: a
    .model({
      articleTitle: a.string(),
      articleAuthor: a.string(),
      articlePublisher: a.string(),
      articleDate: a.datetime(),
      articleURL: a.url(),
      reportDate: a.datetime(),
      reportScore: a.integer(),
      reportSummaryReference: a.string(),
      reportBodyReference: a.string(),
    })
    .authorization((allow) => [allow.publicApiKey().to(["read", "create", "delete"])]) // temportarily like this
});

export type Schema = ClientSchema<typeof schema>;

export const data = defineData({
  schema,
  authorizationModes: {
    defaultAuthorizationMode: 'apiKey',
    apiKeyAuthorizationMode: {
      expiresInDays: 30
    }
  }
});
```

Article Report/Publisher Cards



Article Meta Data Connection



Milestone 4 Matrix

Task	Thomas	Anthony	Josh	Varun
1. Prompt Engineering	0%	0%	50%	50%
2. Improve model output	0%	0%	50%	50%
3. Show default home page cards	50%	50%	0%	0%
4. Article data connection to report page via input	30%	0%	0%	70%
5. Create logo and branding	50%	50%	0%	0%
6. Chrome extension	50%	50%	0%	0%