XPath / CSS

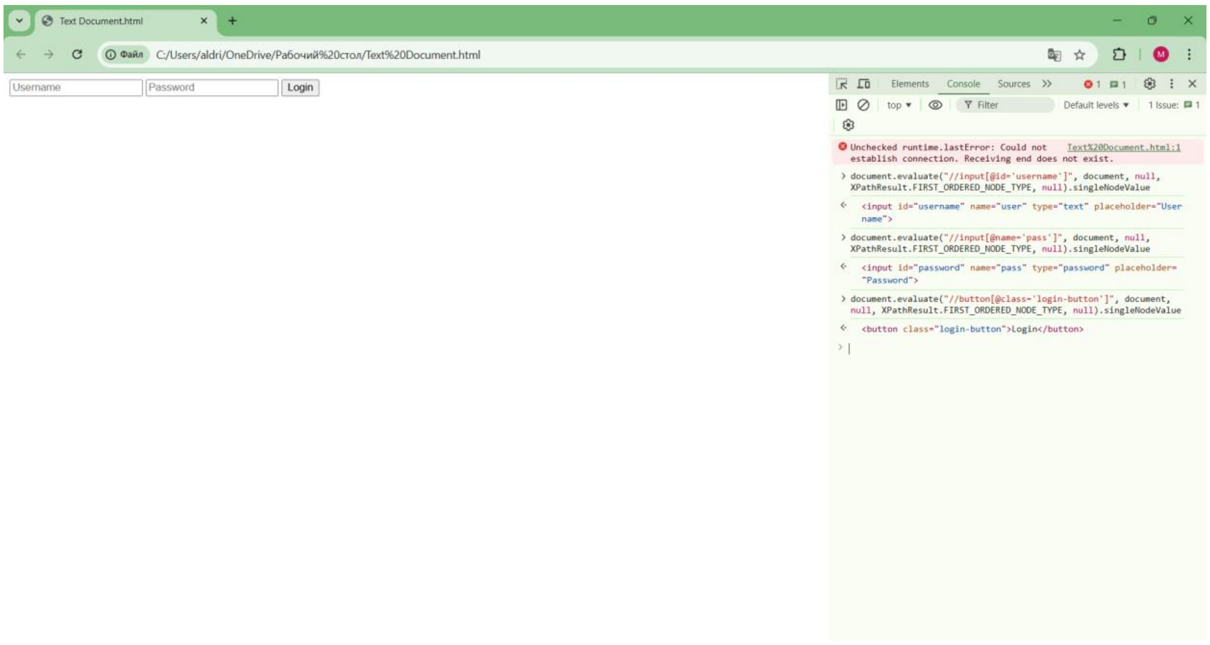
Task 1: Locating Elements by Basic Attributes:

XPath:

1. //input[@id='username']

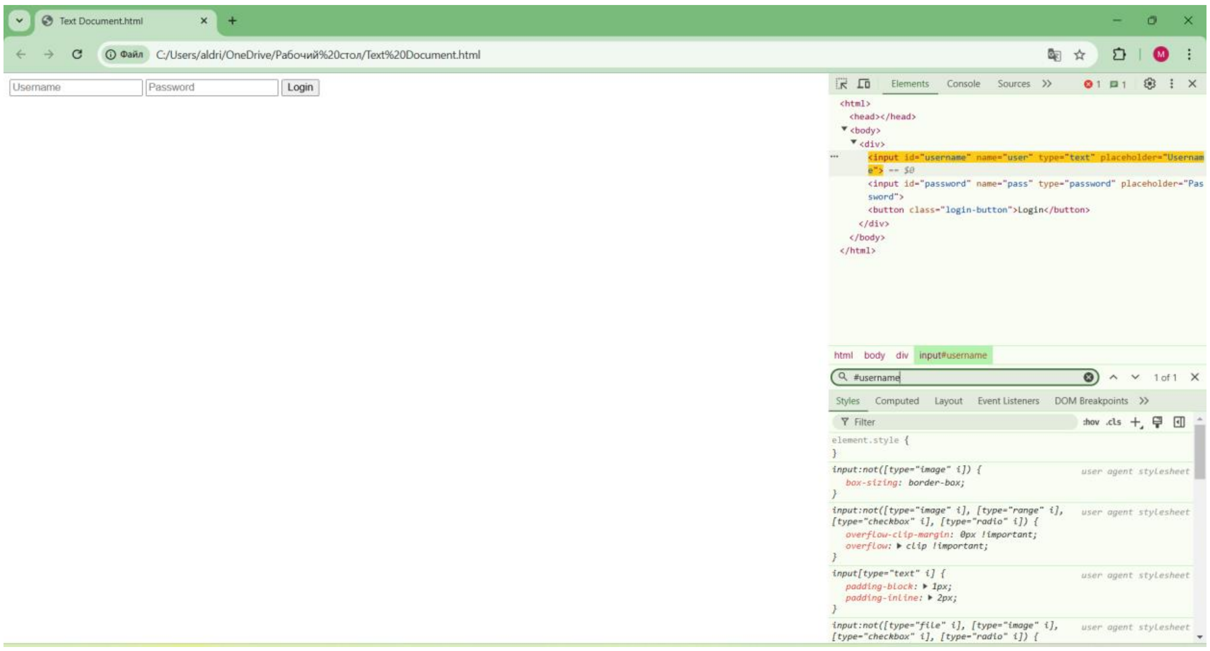
2. //input[@name='pass']

3. //button[@class='login-button']



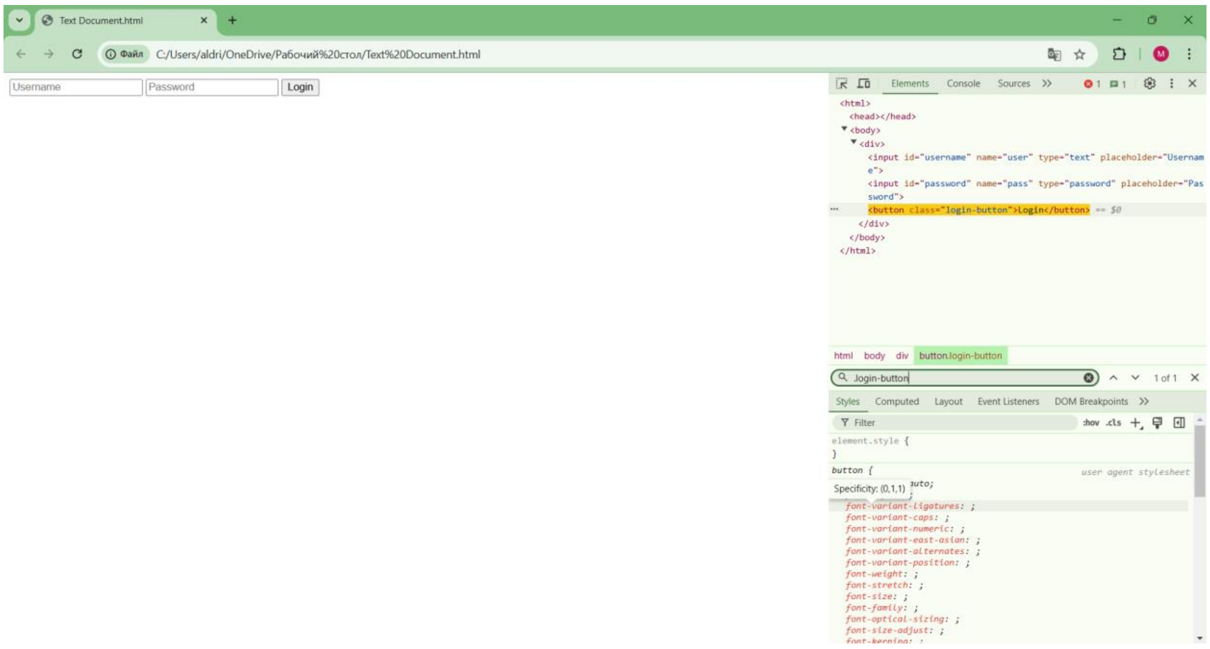
```
Unchecked runtime.lastError: Could not    Text%20Document.html:1
establish connection. Receiving end does not exist.
> document.evaluate("//input[@id='username']", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue
<·  <input id="username" name="user" type="text" placeholder="User
    name">
> document.evaluate("//input[@name='pass']", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue
<·  <input id="password" name="pass" type="password" placeholder=
    "Password">
> document.evaluate("//button[@class='login-button']", document,
  null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue
<·  <button class="login-button">Login</button>
> |
```

CSS Selectors: 1. #username 2. [name='pass'] 3.   .login-button


Text Document.html
C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html
Username
Password
Login
Elements
Console
Sources
html body div input#username
#username
1 of 1
Styles Computed Layout Event Listeners DOM Breakpoints

```
<html>
  <head></head>
  ▼<body>
    ▼<div>
        <input id="username" name="user" type="text" placeholder="Username"> == $0
        <input id="password" name="pass" type="password" placeholder="Password">
        <button class="login-button">Login</button>
      </div>
    </body>
</html>
```

```
element.style {
}
input:not([type="image" i]) {                                user agent stylesheet
  box-sizing: border-box;
}
input:not([type="image" i], [type="range" i],                user agent stylesheet
[type="checkbox" i], [type="radio" i]) {
  overflow-clip-margin: 0px !important;
  overflow: ▶ clip !important;
}
input[type="text" i] {                                       user agent stylesheet
  padding-block: ▶ 1px;
  padding-inline: ▶ 2px;
}
input:not([type="file" i], [type="image" i],                 user agent stylesheet
[type="checkbox" i], [type="radio" i]) {
```


Text Document.html
C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html
Username
Password
Login
Elements
Console
Sources
html body div button.login-button
.login-button
1 of 1
Styles Computed Layout Event Listeners DOM Breakpoints
Specificity: (0,1,1)

```
<html>
  <head></head>
  ▼<body>
    ▼<div>
        <input id="username" name="user" type="text" placeholder="Username">
        <input id="password" name="pass" type="password" placeholder="Password">
        <button class="login-button">Login</button> == $0
      </div>
    </body>
</html>
```

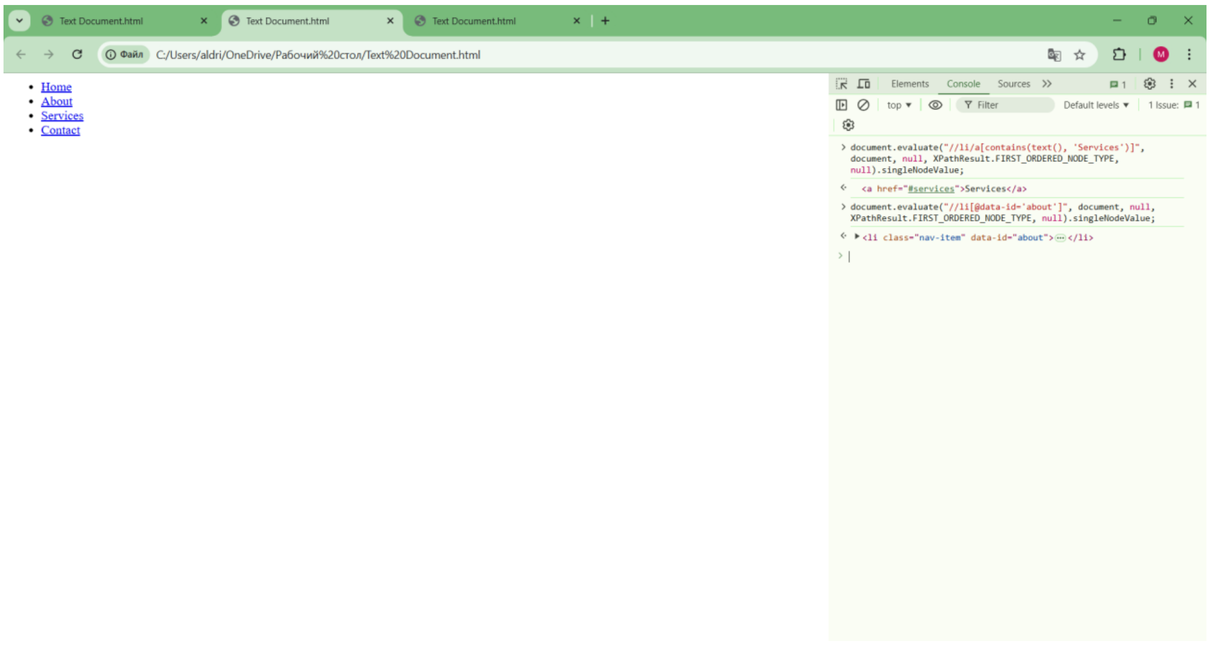
```
element.style {
}
button {                                                     user agent stylesheet
  auto;
  font-variant-ligatures: ;
  font-variant-caps: ;
  font-variant-numeric: ;
  font-variant-east-asian: ;
  font-variant-alternates: ;
  font-variant-position: ;
  font-weight: ;
  font-stretch: ;
  font-size: ;
  font-family: ;
  font-optical-sizing: ;
  font-size-adjust: ;
```

Task 2: Handling Dynamic Elements:

XPath:

1. //li[@data-id='about']

2. //li/a[contains(text(), 'Services')]

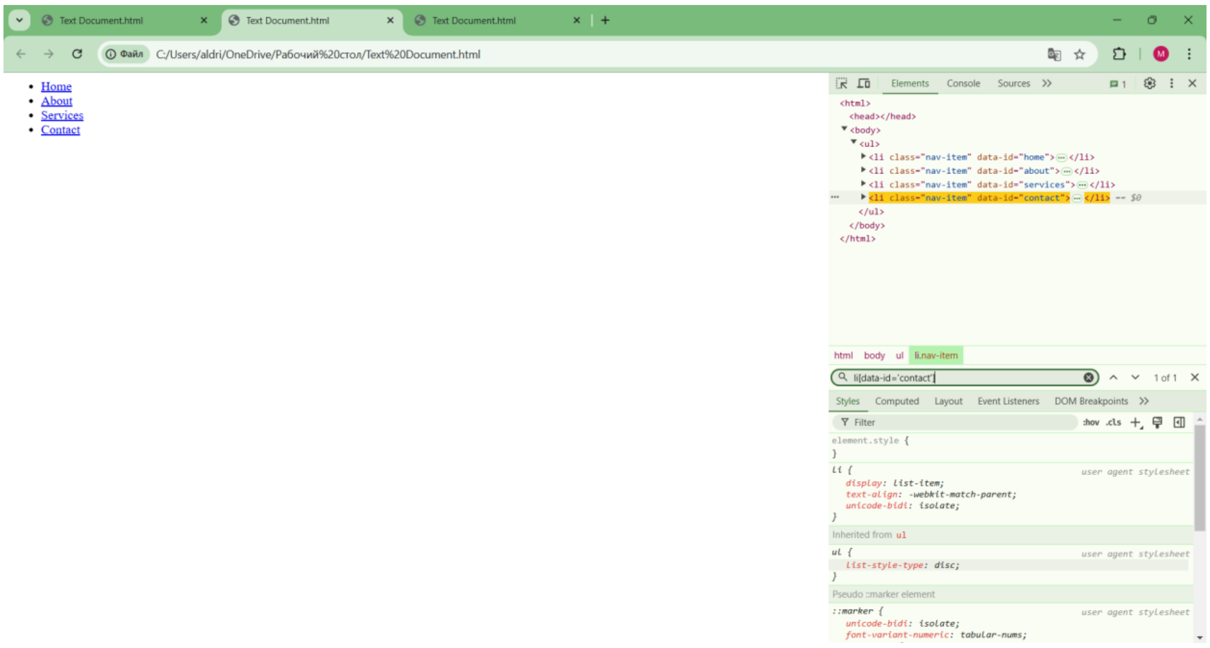


C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

- Home
- About
- Services
- Contact

```
> document.evaluate("//li/a[contains(text(), 'Services')]",
  document, null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <a href="#services">Services</a>
> document.evaluate("//li[@data-id='about']", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<· ▶<li class="nav-item" data-id="about">⋯</li>
> |
```

CSS Selectors: 1. li[data-id='contact'] 2. li a[href='#home']



C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

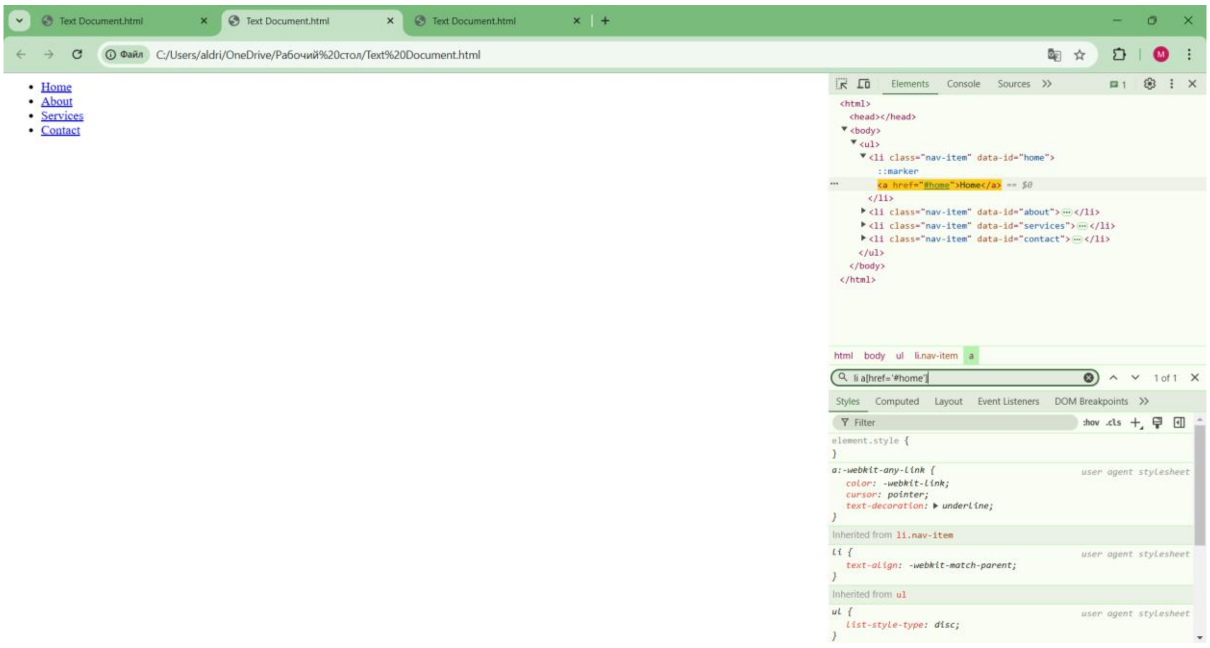
- Home
- About
- Services
- Contact

```
<html>
  <head></head>
 ▼<body>
   ▼<ul>
     ▶<li class="nav-item" data-id="home">⋯</li>
     ▶<li class="nav-item" data-id="about">⋯</li>
     ▶<li class="nav-item" data-id="services">⋯</li>
···  ▶<li class="nav-item" data-id="contact">⋯</li> == $0
    </ul>
  </body>
</html>
```

html body ul li.nav-item

li[data-id='contact'] 1 of 1

```
element.style {
}
li {                                    user agent stylesheet
   display: list-item;
   text-align: -webkit-match-parent;
   unicode-bidi: isolate;
}
Inherited from ul
ul {                                    user agent stylesheet
   list-style-type: disc;
}
Pseudo ::marker element
::marker {                              user agent stylesheet
   unicode-bidi: isolate;
   font-variant-numeric: tabular-nums;
```



```
<html>
  <head></head>
  <body>
    <ul>
      <li class="nav-item" data-id="home">
        ::marker
        <a href="#home">Home</a> == $0
      </li>
      <li class="nav-item" data-id="about">…</li>
      <li class="nav-item" data-id="services">…</li>
      <li class="nav-item" data-id="contact">…</li>
    </ul>
  </body>
</html>
```

```
li a[href='#home']    1 of 1
```

```
element.style {
}
a:-webkit-any-link {                          user agent stylesheet
  color: -webkit-link;
  cursor: pointer;
  text-decoration: ▶ underline;
}
Inherited from li.nav-item
li {                                          user agent stylesheet
  text-align: -webkit-match-parent;
}
Inherited from ul
ul {                                          user agent stylesheet
  list-style-type: disc;
}
```

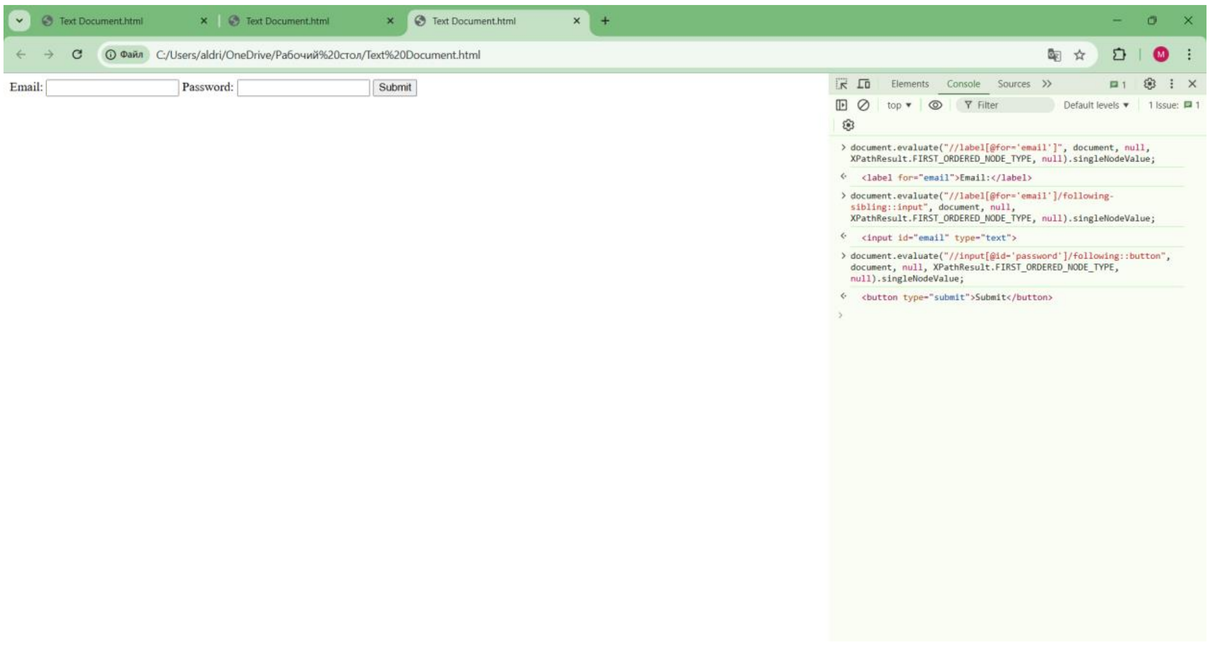
Task 3: Using XPath Axes to Navigate the DOM:

XPath:

1. //label[@for='email']

2. //label[@for='email']/following-sibling::input

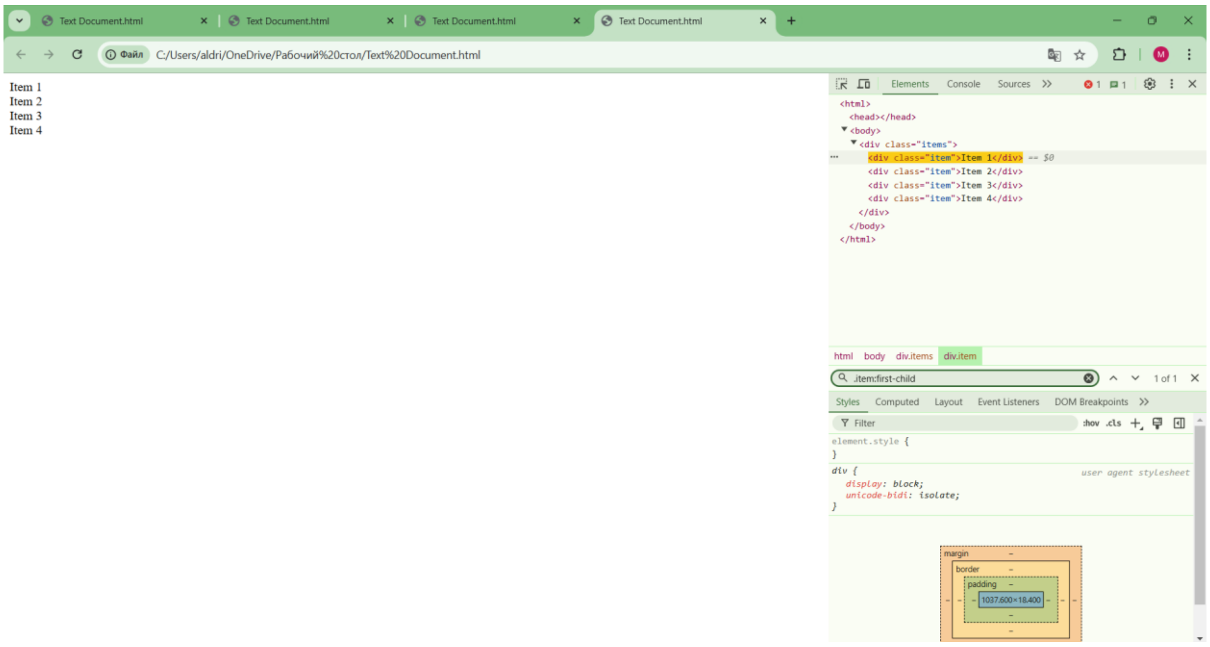
3. //input[@id='password']/following::button



```
> document.evaluate("//label[@for='email']", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <label for="email">Email:</label>
> document.evaluate("//label[@for='email']/following-
  sibling::input", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <input id="email" type="text">
> document.evaluate("//input[@id='password']/following::button",
  document, null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <button type="submit">Submit</button>
>
```

Task 4: Writing CSS Selectors with Pseudo-Classes:

CSS Selectors: 1. .item:first-child 2. .item:last-child 3. .item:nth-child(3)

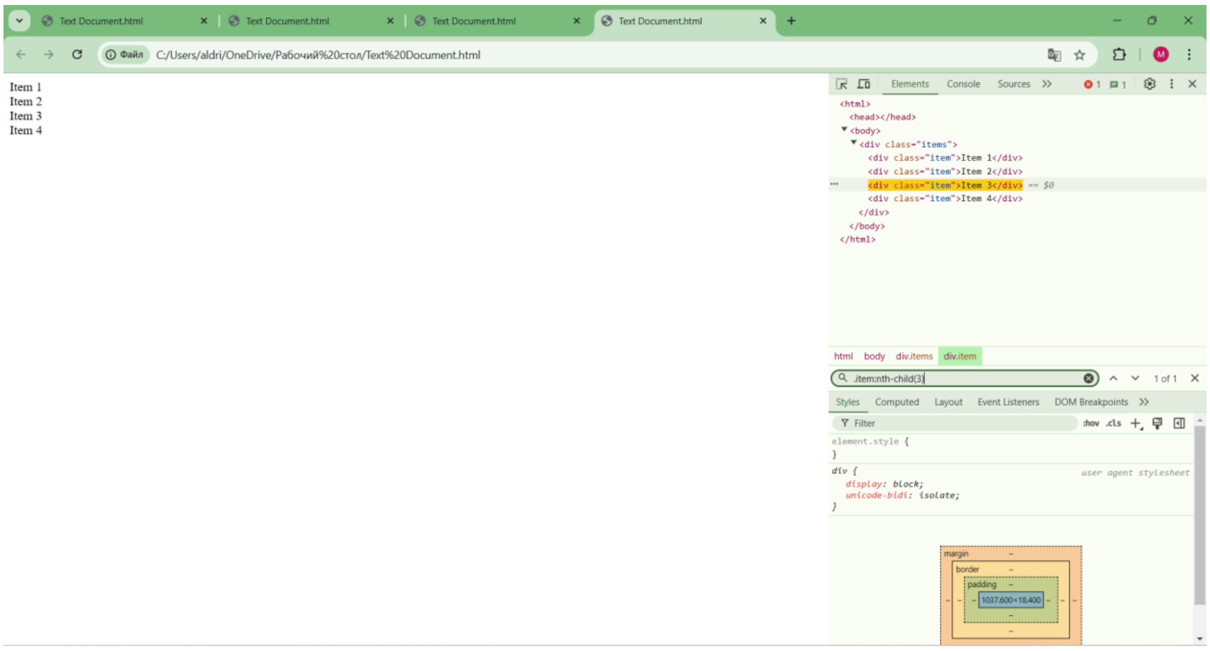


Item 1
Item 2
Item 3
Item 4

```
<html>
  <head></head>
  ▼<body>
    ▼<div class="items">
      <div class="item">Item 1</div> == $0
      <div class="item">Item 2</div>
      <div class="item">Item 3</div>
      <div class="item">Item 4</div>
    </div>
  </body>
</html>
```

.item:first-child — 1 of 1

```
element.style {
}
div {                                   user agent stylesheet
  display: block;
  unicode-bidi: isolate;
}
```



Item 1
Item 2
Item 3
Item 4

```
<html>
  <head></head>
  ▼<body>
    ▼<div class="items">
      <div class="item">Item 1</div>
      <div class="item">Item 2</div>
      <div class="item">Item 3</div> == $0
      <div class="item">Item 4</div>
    </div>
  </body>
</html>
```

.item:nth-child(3) — 1 of 1

```
element.style {
}
div {                                   user agent stylesheet
  display: block;
  unicode-bidi: isolate;
}
```

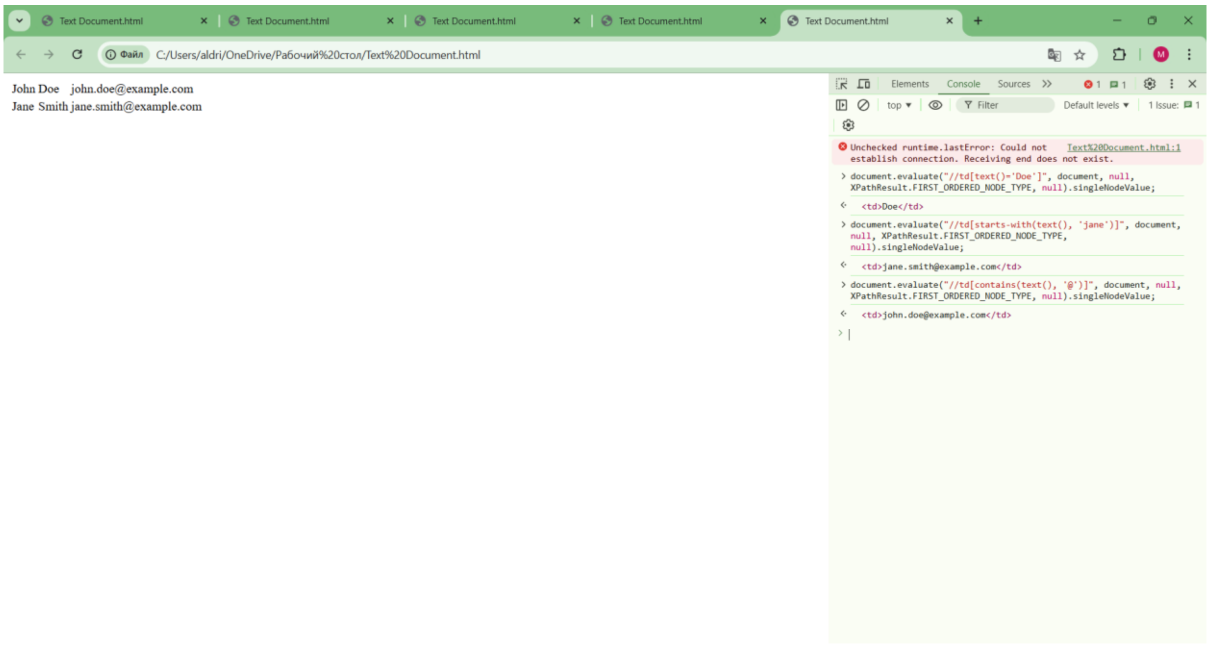
Task 5: Using XPath with Wildcards and Functions:

XPath:

1. //td[text()='Doe']

2. //td[starts-with(text(), 'jane')]

3. //td[contains(text(), '@')]



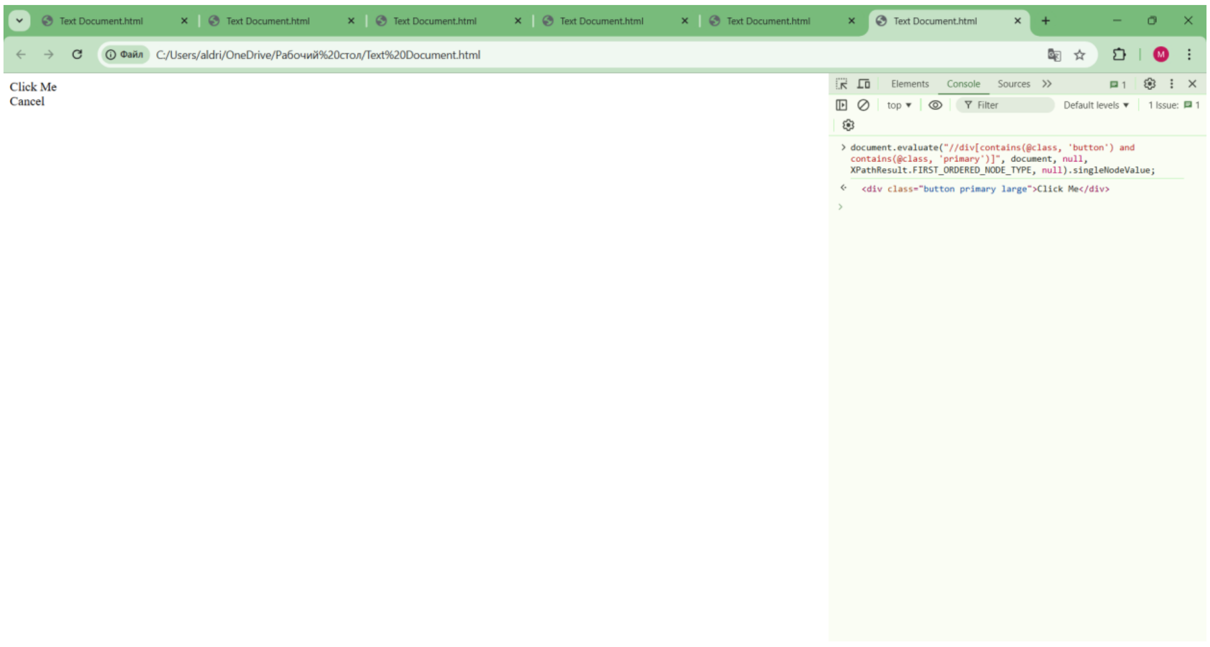
```
C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

John Doe    john.doe@example.com
Jane Smith jane.smith@example.com
```

```
Unchecked runtime.lastError: Could not    Text%20Document.html:1
establish connection. Receiving end does not exist.
> document.evaluate("//td[text()='Doe']", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <td>Doe</td>
> document.evaluate("//td[starts-with(text(), 'jane')]", document,
  null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <td>jane.smith@example.com</td>
> document.evaluate("//td[contains(text(), '@')]", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <td>john.doe@example.com</td>
> |
```

Task 6: Handling Elements with Multiple Classes:

XPath: //div[contains(@class, 'button') and contains(@class, 'primary')]

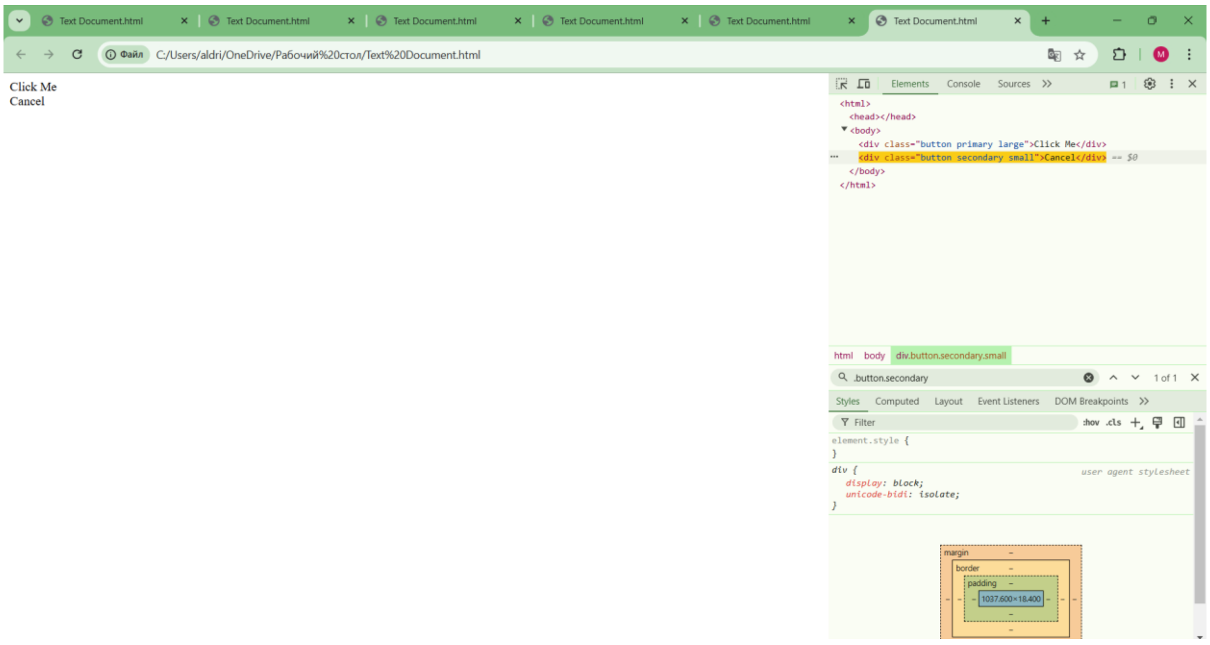


```
C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

Click Me
Cancel
```

```
> document.evaluate("//div[contains(@class, 'button') and
  contains(@class, 'primary')]", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <div class="button primary large">Click Me</div>
>
```

CSS Selectors: 1.   .button.secondary 2.   .primary.large


C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

Click Me
Cancel

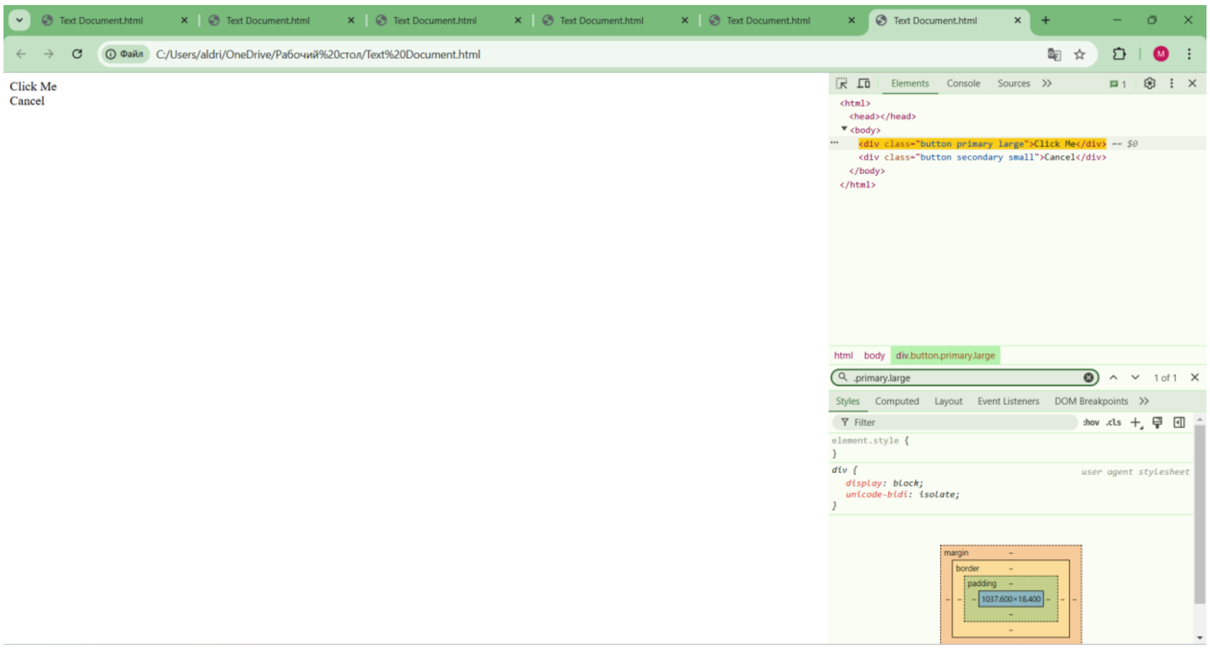
```
<html>
  <head></head>
  ▼<body>
    <div class="button primary large">Click Me</div>
    <div class="button secondary small">Cancel</div> == $0
  </body>
</html>
```

html body div.button.secondary.small

.button.secondary 1 of 1

```
element.style {
}
div {                                  user agent stylesheet
  display: block;
  unicode-bidi: isolate;
}
```

1037.600×18.400


C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

Click Me
Cancel

```
<html>
  <head></head>
  ▼<body>
    <div class="button primary large">Click Me</div> == $0
    <div class="button secondary small">Cancel</div>
  </body>
</html>
```

html body div.button.primary.large

.primary.large 1 of 1

```
element.style {
}
div {                                  user agent stylesheet
  display: block;
  unicode-bidi: isolate;
}
```

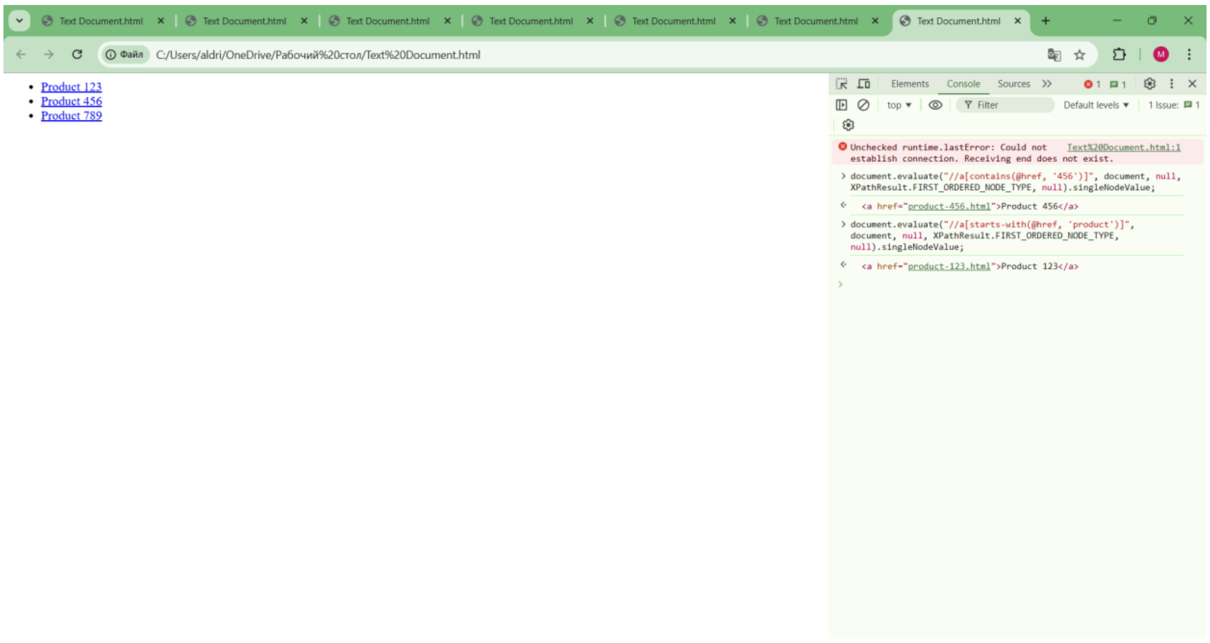
1037.600×18.400

Task 7: Locating Elements Using Contains and Starts-With:

XPath:

1. //a[contains(@href, '456')]

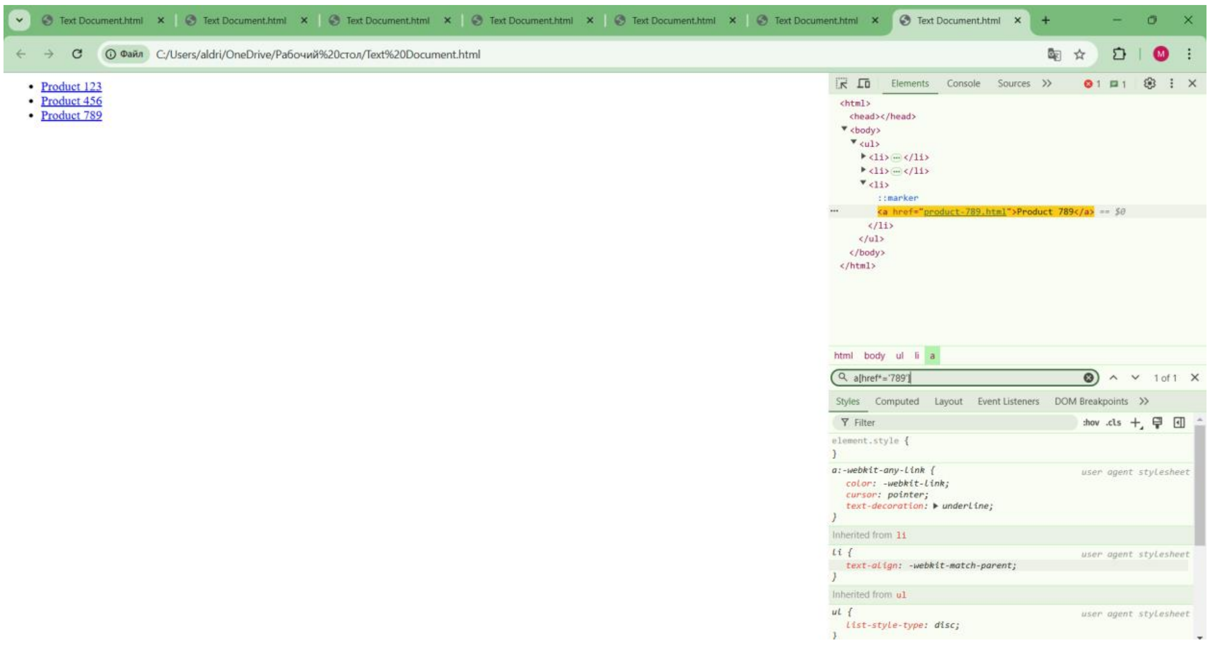
2. //a[starts-with(@href, 'product')]



- Product 123
- Product 456
- Product 789

```
Unchecked runtime.lastError: Could not     Text%20Document.html:1
establish connection. Receiving end does not exist.
> document.evaluate("//a[contains(@href, '456')]", document, null,
  XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;
<·  <a href="product-456.html">Product 456</a>
> document.evaluate("//a[starts-with(@href, 'product')]",
  document, null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <a href="product-123.html">Product 123</a>
```

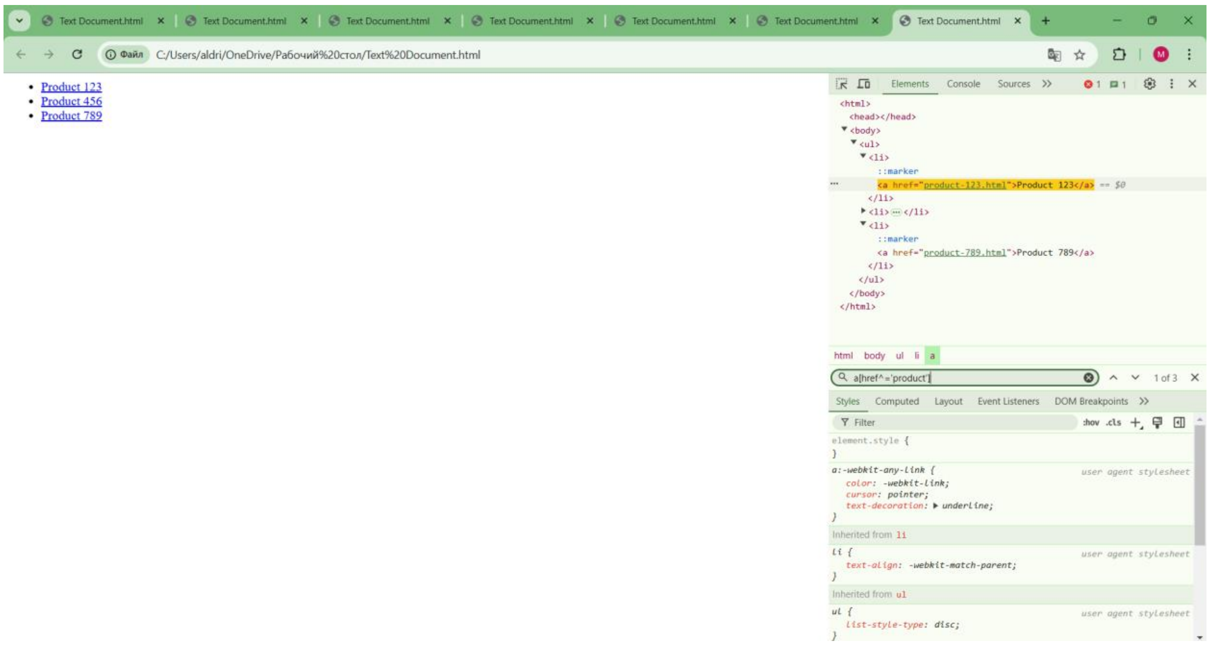
CSS Selectors: 1. a[href*='789'] 2. a[href^='product']



- Product 123
- Product 456
- Product 789

```
<html>
  <head></head>
 ▼<body>
   ▼<ul>
     ▶<li> … </li>
     ▶<li> … </li>
     ▼<li>
        ::marker
        <a href="product-789.html">Product 789</a> == $0
      </li>
    </ul>
  </body>
</html>
```

a[href*='789'] — 1 of 1



C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

- Product 123
- Product 456
- Product 789

```
<html>
  <head></head>
  ▼<body>
    ▼<ul>
      ▼<li>
          ::marker
          <a href="product-123.html">Product 123</a> == $0
        </li>
      ▶<li> ⋯ </li>
      ▼<li>
          ::marker
          <a href="product-789.html">Product 789</a>
        </li>
      </ul>
    </body>
</html>
```

a[href^='product'] — 1 of 3

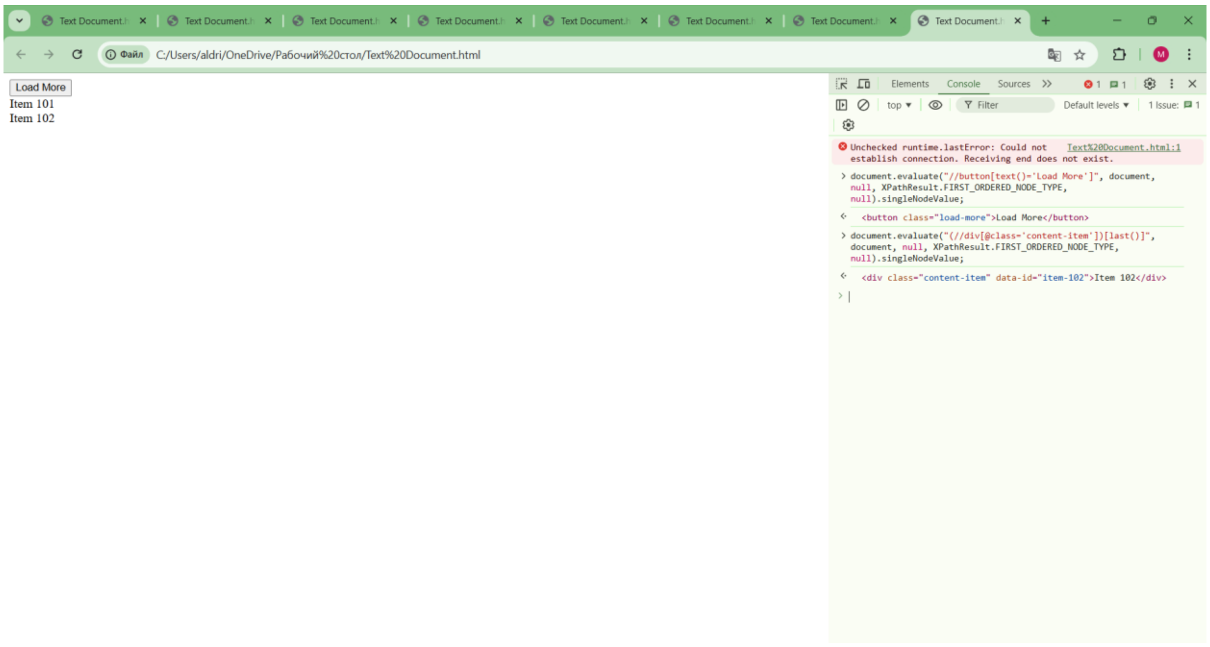
```
element.style {
}
a:-webkit-any-link {                          user agent stylesheet
    color: -webkit-link;
    cursor: pointer;
    text-decoration: ▶ underline;
}
Inherited from li
li {                                          user agent stylesheet
    text-align: -webkit-match-parent;
}
Inherited from ul
ul {                                          user agent stylesheet
    list-style-type: disc;
}
```

Task 8: Dynamic Element Handling in Automation Scripts:

XPath:

1. //button[text()='Load More']

2. (//div[@class='content-item'])[last()]

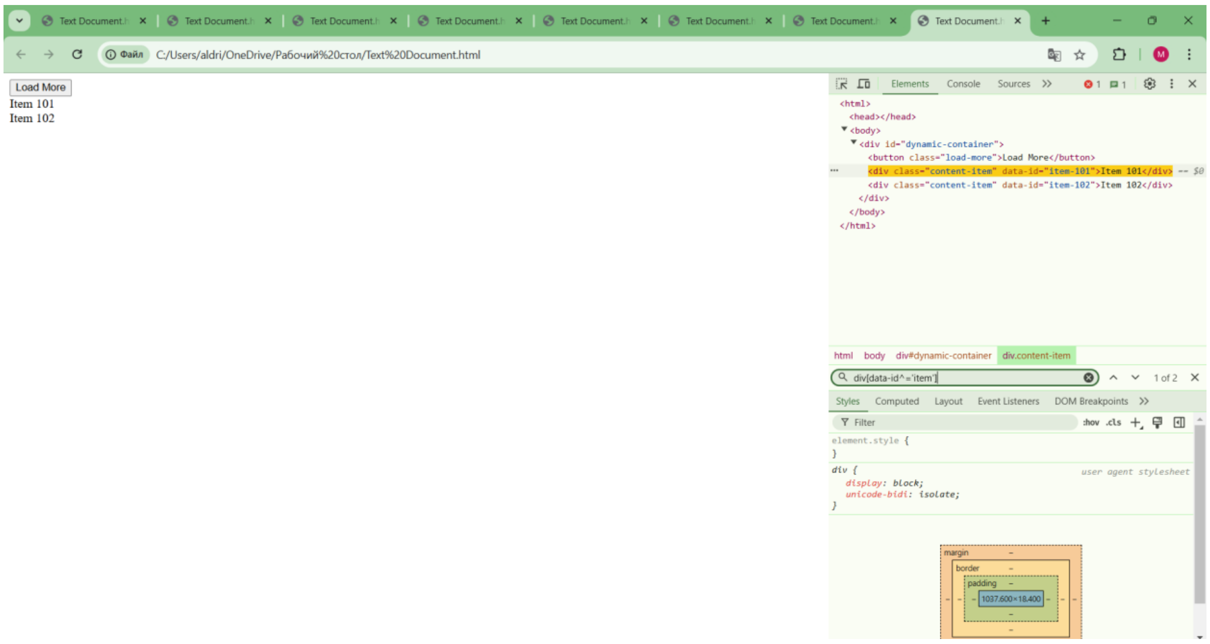


C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

Load More
Item 101
Item 102

```
Unchecked runtime.lastError: Could not        Text%20Document.html:1
establish connection. Receiving end does not exist.
> document.evaluate("//button[text()='Load More']", document,
  null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <button class="load-more">Load More</button>
> document.evaluate("(//div[@class='content-item'])[last()]",
  document, null, XPathResult.FIRST_ORDERED_NODE_TYPE,
  null).singleNodeValue;
<·  <div class="content-item" data-id="item-102">Item 102</div>
> |
```

CSS Selectors: div[data-id^='item']



C:/Users/aldri/OneDrive/Рабочий%20стол/Text%20Document.html

```
Load More
Item 101
Item 102
```

```
<html>
  <head></head>
  ▼<body>
    ▼<div id="dynamic-container">
      <button class="load-more">Load More</button>
      <div class="content-item" data-id="item-101">Item 101</div> == $0
      <div class="content-item" data-id="item-102">Item 102</div>
    </div>
  </body>
</html>
```

html body div#dynamic-container div.content-item

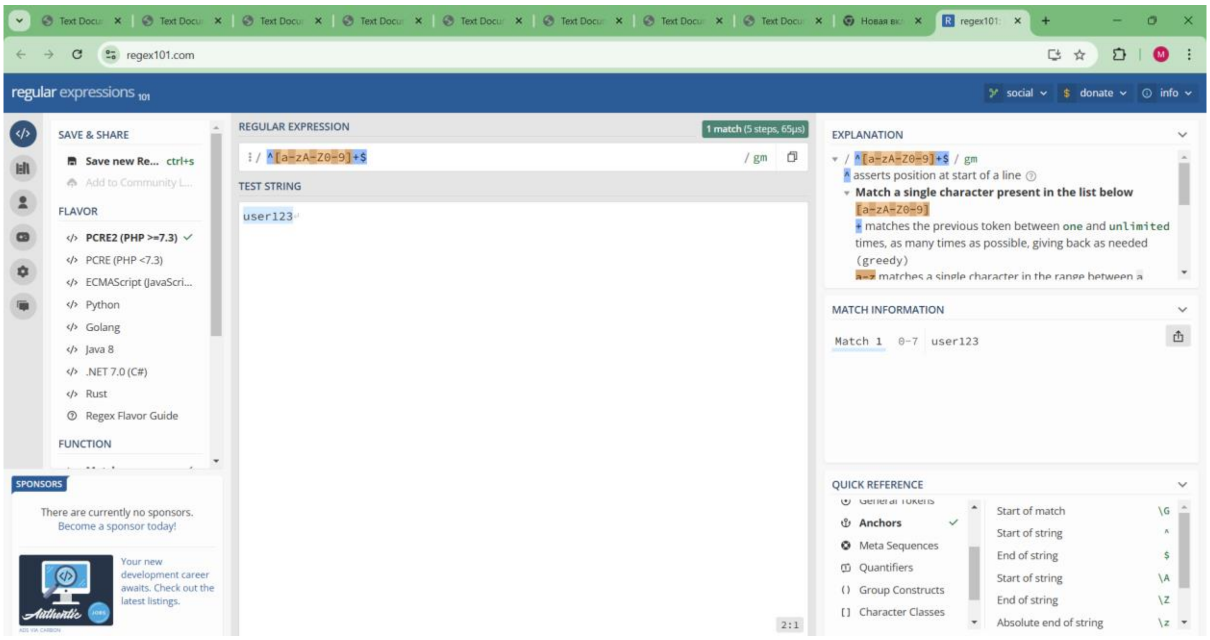
div[data-id^='item'] 1 of 2

```
element.style {
}
div {                                   user agent stylesheet
  display: block;
  unicode-bidi: isolate;
}
```

RegEx

Task 1: Basic RegEx Pattern Matching (Required):

1. /^[a-zA-Z0-9]+$/gm



regex101.com

REGULAR EXPRESSION 1 match (5 steps, 65μs)

```
/ ^[a-zA-Z0-9]+$ / gm
```

TEST STRING

```
user123
```

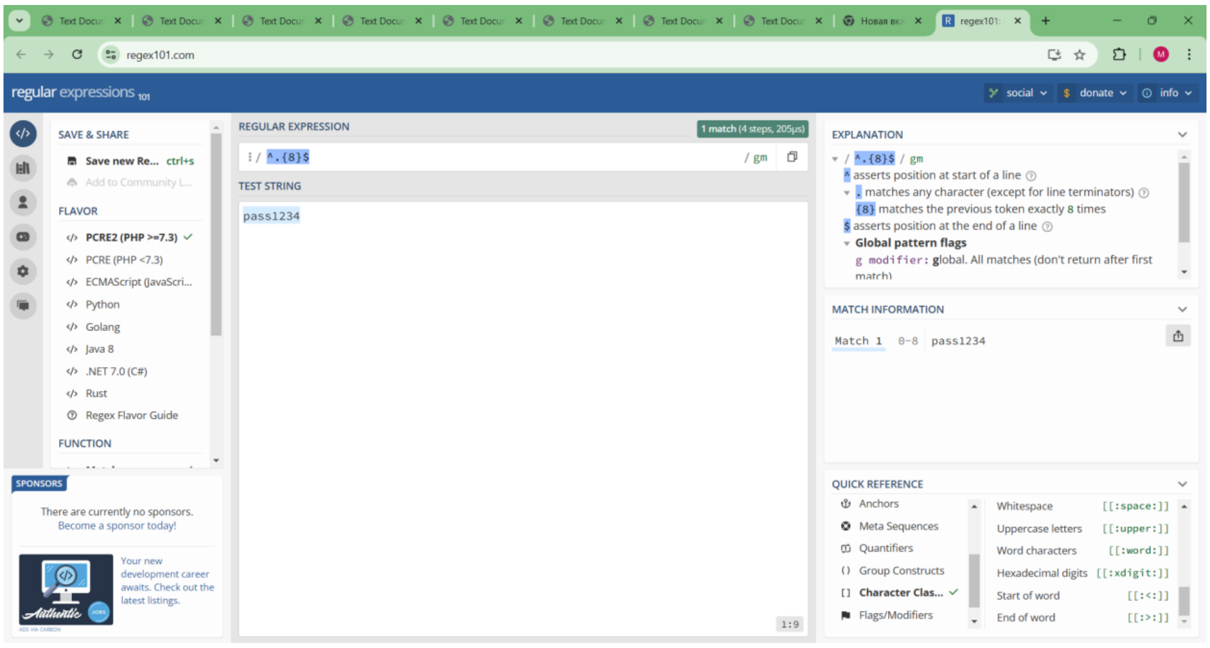
EXPLANATION

/ ^[a-zA-Z0-9]+$ / gm
^ asserts position at start of a line
Match a single character present in the list below
[a-zA-Z0-9]
+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
a-z matches a single character in the range between a

MATCH INFORMATION

```
Match 1   0-7   user123
```

2. /^.{8}$/gm



REGULAR EXPRESSION 1 match (4 steps, 205μs)

```
/ ^.{8}$ / gm
```

TEST STRING

```
pass1234
```

EXPLANATION

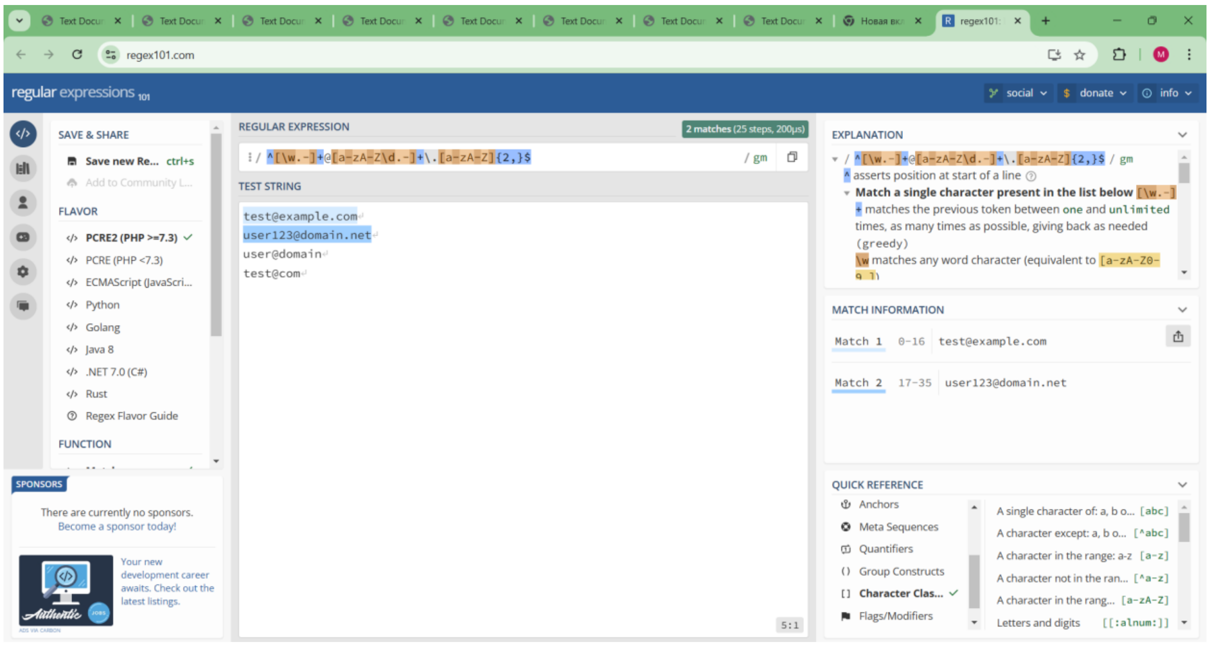
/ ^.{8}$ / gm
^ asserts position at start of a line
. matches any character (except for line terminators)
{8} matches the previous token exactly 8 times
$ asserts position at the end of a line
Global pattern flags
g modifier: global. All matches (don't return after first match)

MATCH INFORMATION

```
Match 1   0-8   pass1234
```

Task 2: Matching Email Addresses (Required):

/^[\w.-]+@[a-zA-Z\d.-]+\.[a-zA-Z]{2,}$/gm



REGULAR EXPRESSION 2 matches (25 steps, 200μs)

```
/ ^[\w.-]+@[a-zA-Z\d.-]+\.[a-zA-Z]{2,}$ / gm
```

TEST STRING

```
test@example.com
user123@domain.net
user@domain
test@com
```

EXPLANATION

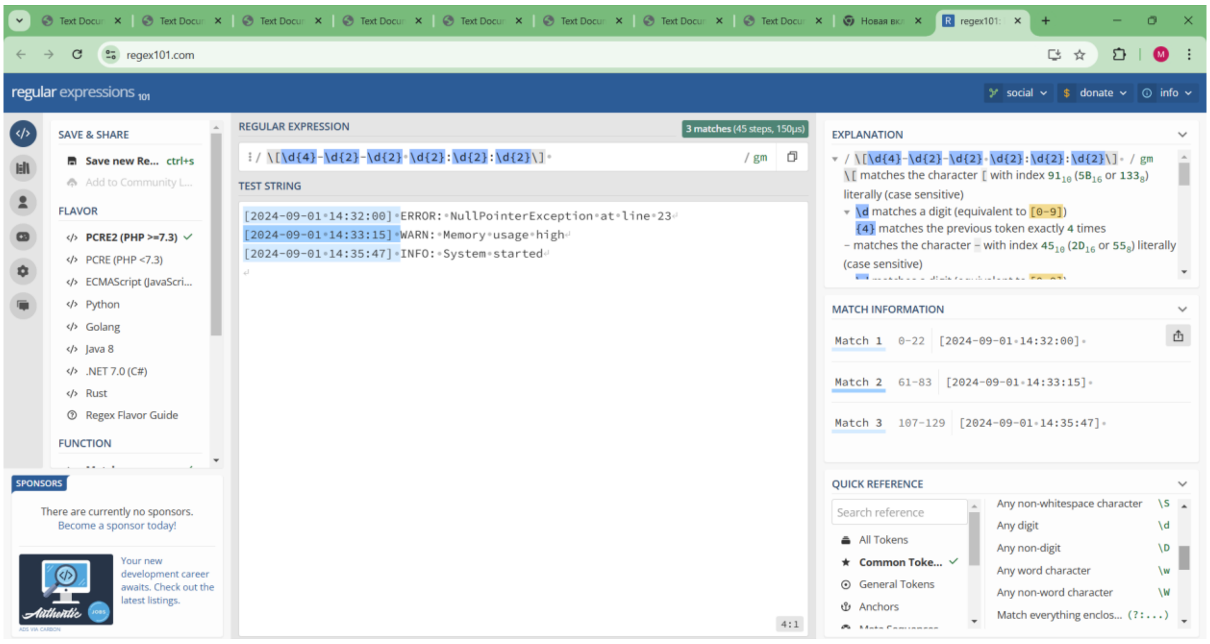
/ ^[\w.-]+@[a-zA-Z\d.-]+\.[a-zA-Z]{2,}$ / gm
^ asserts position at start of a line
Match a single character present in the list below [\w.-]
+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
\w matches any word character (equivalent to [a-zA-Z0-9_])

MATCH INFORMATION

```
Match 1   0-16    test@example.com
Match 2   17-35   user123@domain.net
```

Task 3: Extracting Data from Log Files (Required):

/\[\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\] /gm


REGULAR EXPRESSION — 3 matches (45 steps, 150μs)

```
/ \[\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\] / gm
```

TEST STRING

```
[2024-09-01 14:32:00] ERROR: NullPointerException at line 23
[2024-09-01 14:33:15] WARN: Memory usage high
[2024-09-01 14:35:47] INFO: System started
```

EXPLANATION

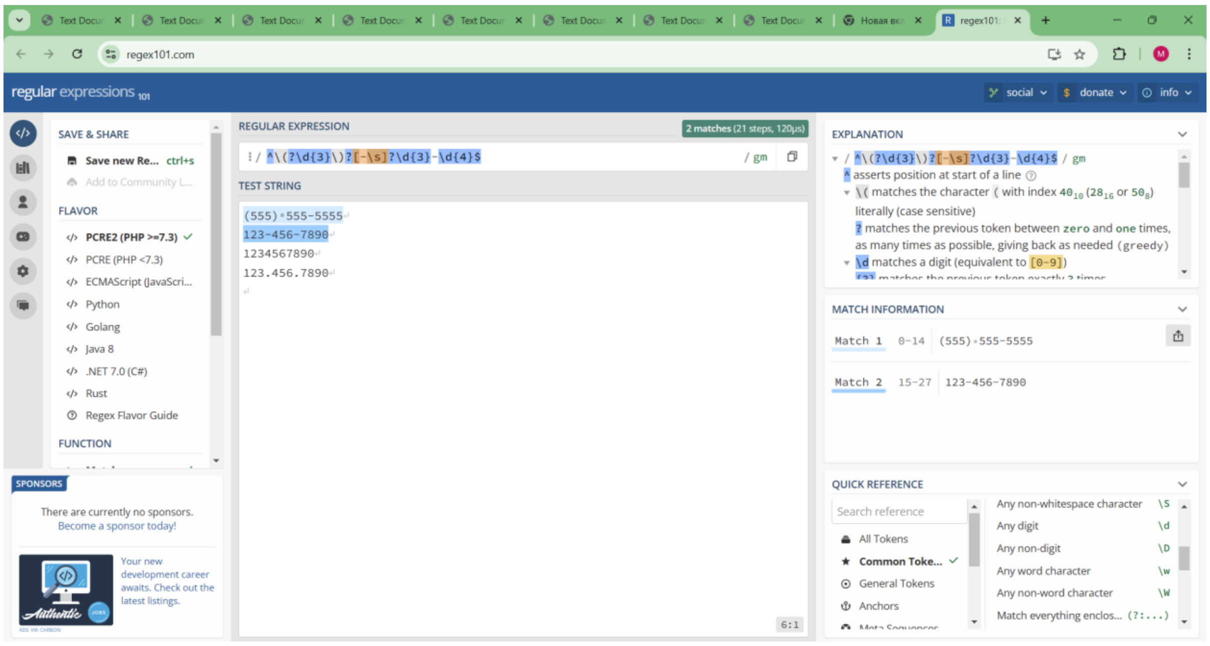
/ \[\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\] / gm
\[matches the character [with index 91_{10} ($5B_{16}$ or 133_8) literally (case sensitive)
\d matches a digit (equivalent to [0-9])
{4} matches the previous token exactly 4 times
- matches the character - with index 45_{10} ($2D_{16}$ or 55_8) literally (case sensitive)

MATCH INFORMATION

```
Match 1   0-22     [2024-09-01 14:32:00]
Match 2   61-83    [2024-09-01 14:33:15]
Match 3   107-129  [2024-09-01 14:35:47]
```

Task 4: Validating Phone Numbers (Required):

/^\(?\d{3}\)?[-\s]?\d{3}-\d{4}$/gm


REGULAR EXPRESSION — 2 matches (21 steps, 120μs)

```
/ ^\(?\d{3}\)?[-\s]?\d{3}-\d{4}$ / gm
```

TEST STRING

```
(555) 555-5555
123-456-7890
1234567890
123.456.7890
```

EXPLANATION

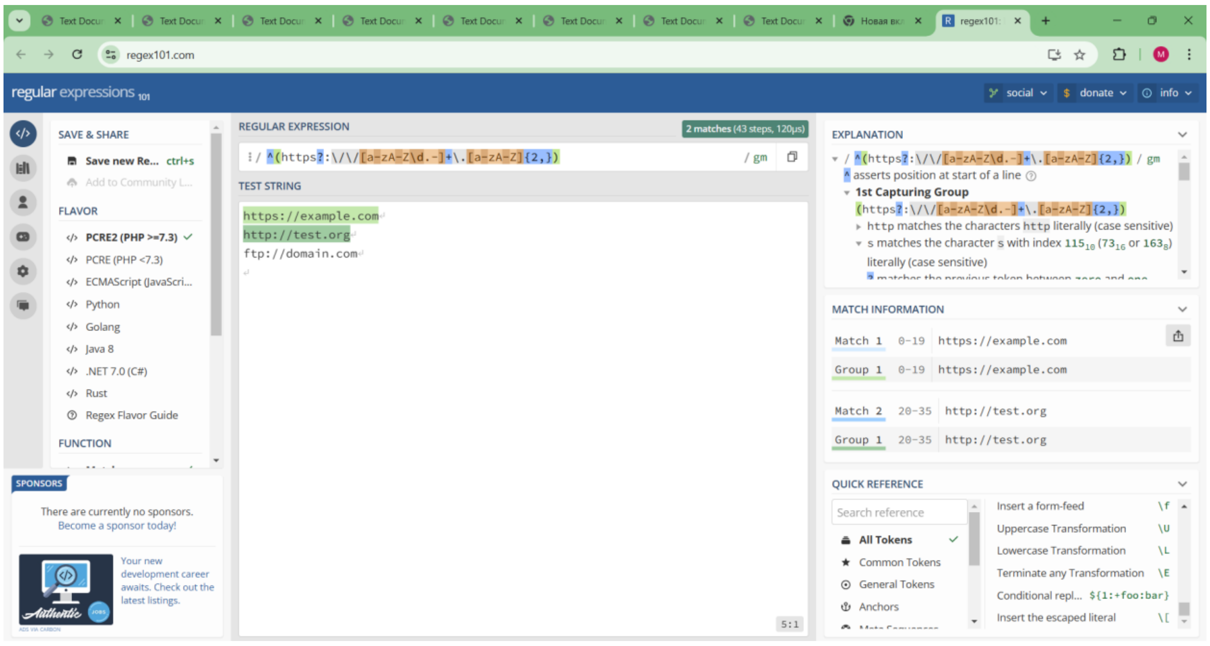
/ ^\(?\d{3}\)?[-\s]?\d{3}-\d{4}$ / gm
^ asserts position at start of a line
\(matches the character (with index 40_{10} (28_{16} or 50_8) literally (case sensitive)
? matches the previous token between zero and one times, as many times as possible, giving back as needed (greedy)
\d matches a digit (equivalent to [0-9])

MATCH INFORMATION

```
Match 1   0-14    (555) 555-5555
Match 2   15-27   123-456-7890
```

Task 5: Parsing URLs (Required):

/^(https?:\/\/[a-zA-Z\d.-]+\.[a-zA-Z]{2,})/gm


REGULAR EXPRESSION — 2 matches (43 steps, 120µs)

```
/ ^(https?:\/\/[a-zA-Z\d.-]+\.[a-zA-Z]{2,}) / gm
```

TEST STRING

```
https://example.com
http://test.org
ftp://domain.com
```

FLAVOR: PCRE2 (PHP >=7.3) ✓

EXPLANATION

```
/ ^(https?:\/\/[a-zA-Z\d.-]+\.[a-zA-Z]{2,}) / gm
^ asserts position at start of a line
1st Capturing Group
(https?:\/\/[a-zA-Z\d.-]+\.[a-zA-Z]{2,})
http matches the characters http literally (case sensitive)
s matches the character s with index 115₁₀ (73₁₆ or 163₈) literally (case sensitive)
```

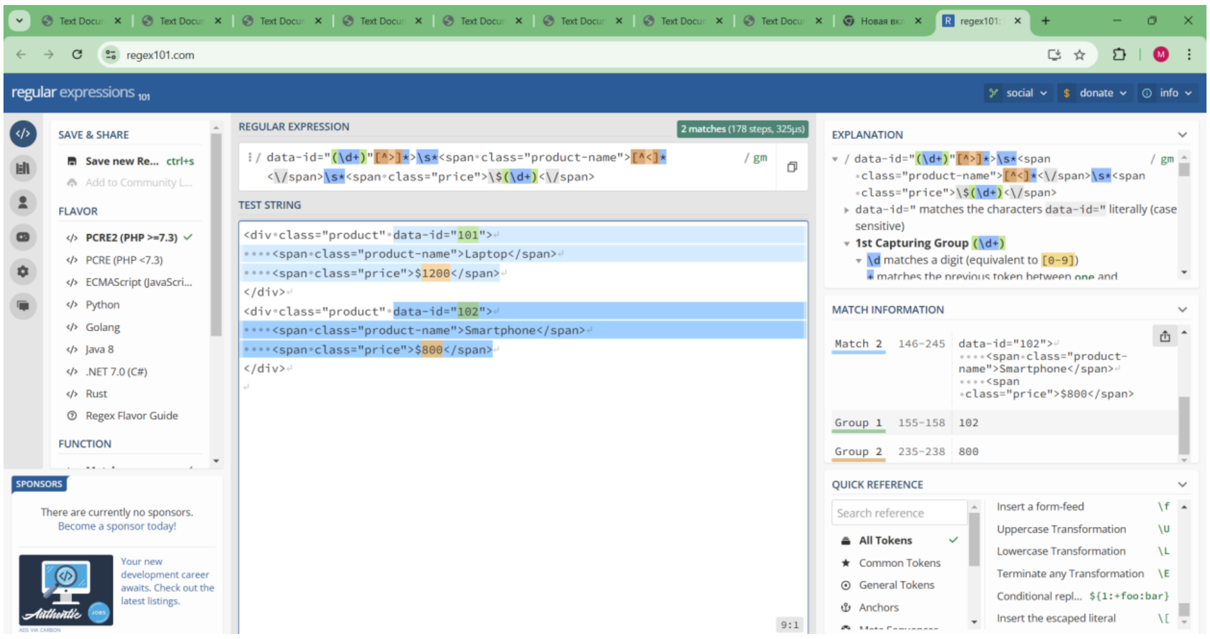
MATCH INFORMATION

```
Match 1   0-19    https://example.com
Group 1   0-19    https://example.com

Match 2   20-35   http://test.org
Group 1   20-35   http://test.org
```

Task 6: Extracting Dynamic Data from HTML Content (Required):

/data-id="(\d+)"[^>]*>\s*<span class="product-name">[^<]*<\/span>\s*<span class="price">\$(\d+)<\/span>/gm


REGULAR EXPRESSION — 2 matches (178 steps, 325µs)

```
/ data-id="(\d+)"[^>]*>\s*<span class="product-name">[^<]*
<\/span>\s*<span class="price">\$(\d+)<\/span> / gm
```

TEST STRING

```
<div class="product" data-id="101">
    <span class="product-name">Laptop</span>
    <span class="price">$1200</span>
</div>
<div class="product" data-id="102">
    <span class="product-name">Smartphone</span>
    <span class="price">$800</span>
</div>
```

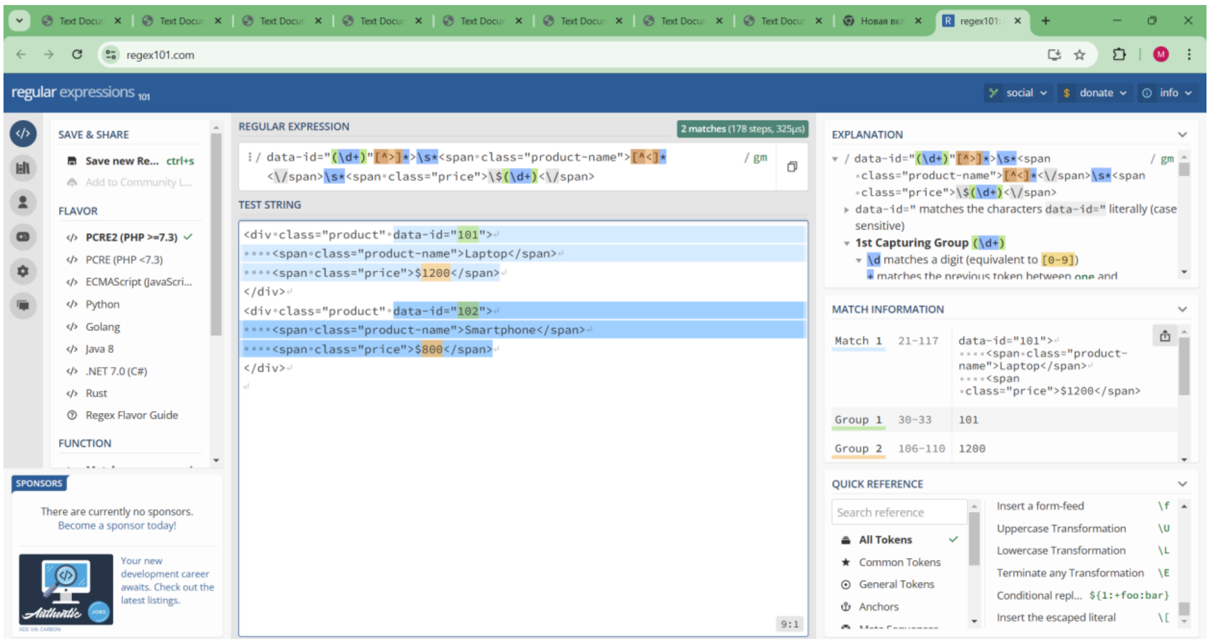
FLAVOR: PCRE2 (PHP >=7.3) ✓

EXPLANATION

```
/ data-id="(\d+)"[^>]*>\s*<span class="product-name">[^<]*<\/span>\s*<span class="price">\$(\d+)<\/span> / gm
data-id=" matches the characters data-id=" literally (case sensitive)
1st Capturing Group (\d+)
\d matches a digit (equivalent to [0-9])
```

MATCH INFORMATION

```
Match 2   146-245   data-id="102">
                        <span class="product-name">Smartphone</span>
                        <span class="price">$800</span>
Group 1   155-158   102
Group 2   235-238   800
```


REGULAR EXPRESSION

2 matches (178 steps, 325µs)

```
/ data-id="(\d+)"[^>]*>\s*<span class="product-name">[^<]*<\/span>\s*<span class="price">\$(\d+)<\/span> / gm
```

TEST STRING

```
<div class="product" data-id="101">
    <span class="product-name">Laptop</span>
    <span class="price">$1200</span>
</div>
<div class="product" data-id="102">
    <span class="product-name">Smartphone</span>
    <span class="price">$800</span>
</div>
```

EXPLANATION

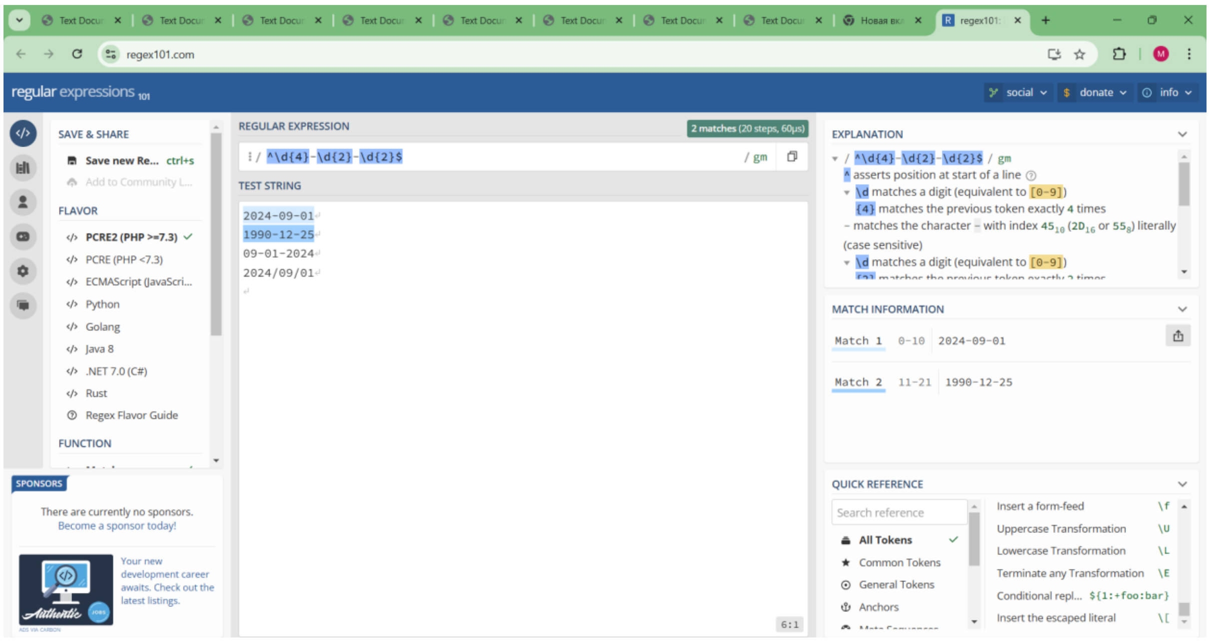
```
/ data-id="(\d+)"[^>]*>\s*<span class="product-name">[^<]*<\/span>\s*<span class="price">\$(\d+)<\/span> / gm
data-id=" matches the characters data-id=" literally (case sensitive)
1st Capturing Group (\d+)
\d matches a digit (equivalent to [0-9])
+ matches the previous token between one and
```

MATCH INFORMATION

```
Match 1   21-117    data-id="101">
                        <span class="product-name">Laptop</span>
                        <span class="price">$1200</span>
Group 1   30-33     101
Group 2   106-110   1200
```

Task 7: Validating Dates (Required):

/^\d{4}-\d{2}-\d{2}$/gm


REGULAR EXPRESSION

2 matches (20 steps, 60µs)

```
/ ^\d{4}-\d{2}-\d{2}$ / gm
```

TEST STRING

```
2024-09-01
1990-12-25
09-01-2024
2024/09/01
```

EXPLANATION

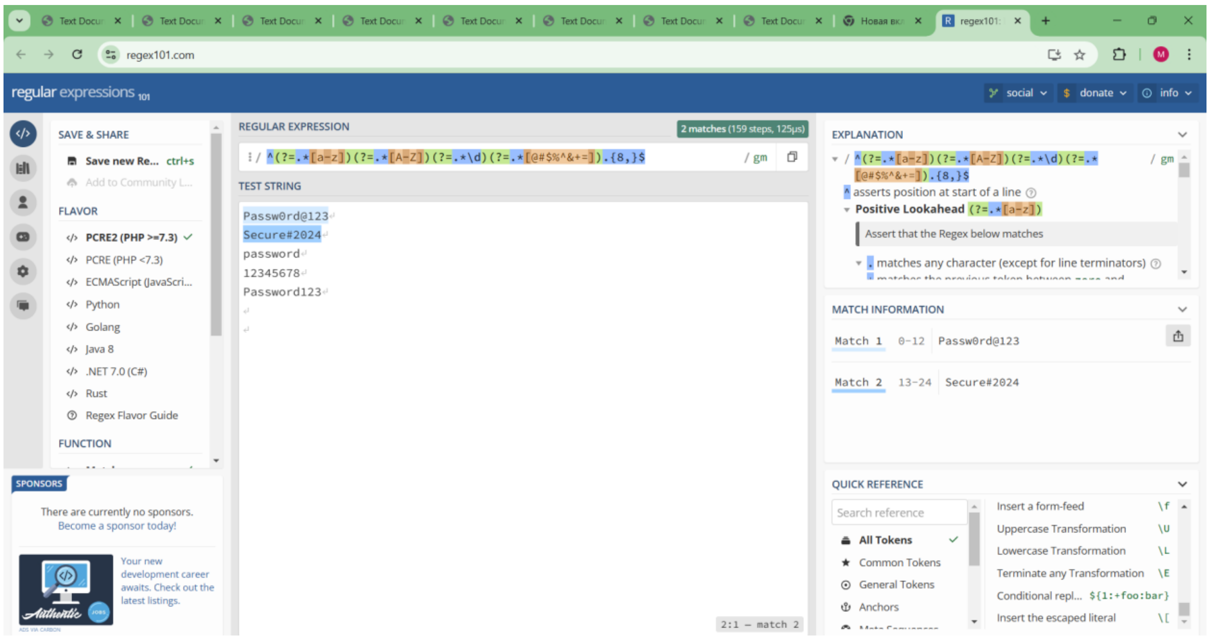
```
/ ^\d{4}-\d{2}-\d{2}$ / gm
^ asserts position at start of a line
\d matches a digit (equivalent to [0-9])
{4} matches the previous token exactly 4 times
- matches the character - with index 45₁₀ (2D₁₆ or 55₈) literally (case sensitive)
\d matches a digit (equivalent to [0-9])
```

MATCH INFORMATION

```
Match 1   0-10    2024-09-01
Match 2   11-21   1990-12-25
```

Task 8: Validating Complex Passwords (Optional):

/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@#$%^&+=]).{8,}$/gm



REGULAR EXPRESSION — 2 matches (159 steps, 125µs)

```
/ ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@#$%^&+=]).{8,}$ / gm
```

TEST STRING

```
Passw0rd@123
Secure#2024
password
12345678
Password123
```

EXPLANATION

/ ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@#$%^&+=]).{8,}$ / gm

^ asserts position at start of a line

Positive Lookahead (?=.*[a-z])

Assert that the Regex below matches

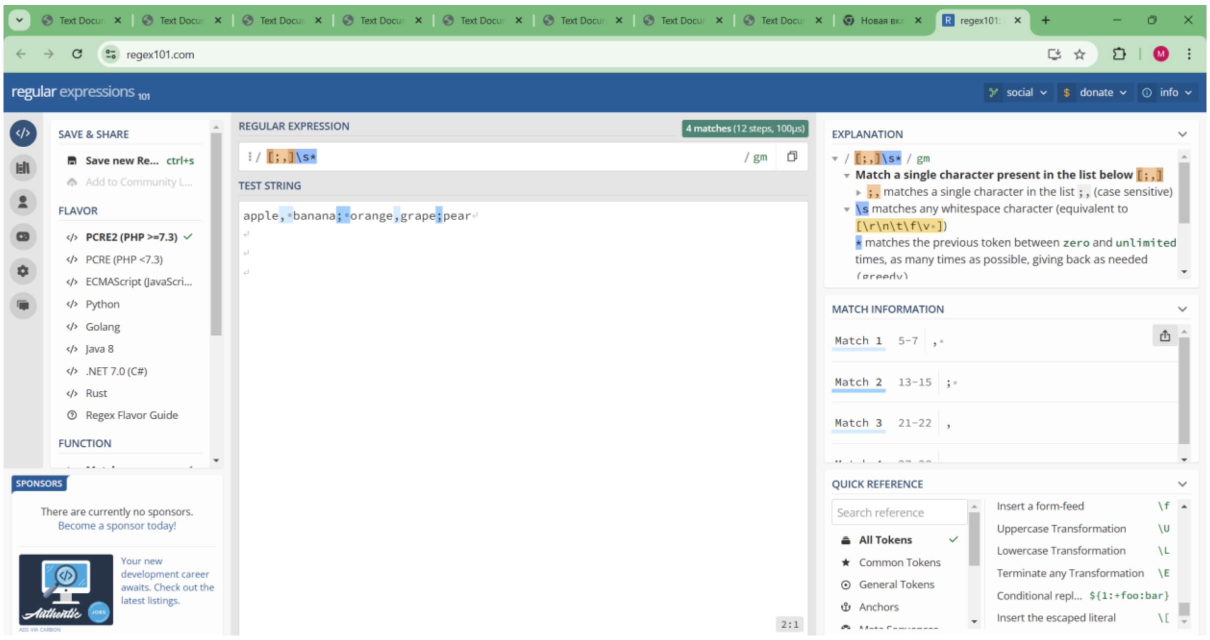
. matches any character (except for line terminators)

MATCH INFORMATION

```
Match 1   0-12    Passw0rd@123
Match 2   13-24   Secure#2024
```

Task 9: Splitting a String with RegEx (Optional):

/[;,]\s*/gm



REGULAR EXPRESSION — 4 matches (12 steps, 100µs)

```
/ [;,]\s* / gm
```

TEST STRING

```
apple, banana; orange,grape;pear
```

EXPLANATION

/ [;,]\s* / gm

Match a single character present in the list below [;,]

;, matches a single character in the list ;, (case sensitive)

\s matches any whitespace character (equivalent to [\r\n\t\f\v])

* matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)

MATCH INFORMATION

```
Match 1   5-7     , 
Match 2   13-15   ; 
Match 3   21-22   ,
```