UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Translator Limbaj Natural – HTML

Irini Karina
Pascan Daniela
Grupa 30239

Cuprins:

1. Cerințele Proiectului

Scopul acestui proiect este de a dezvolta un sistem software care să poată converti un format simplu de text natural într-un cod HTML corespunzător, respectând anumite reguli de formatare.

Obiectivele Proiectului:

- Dezvoltarea unui fișier Lex pentru analiza lexicală a intrării de text și generarea de token-uri corespunzătoare.
- Implementarea unui parser Yacc pentru analiza sintactică a textului și generarea unei structuri de date interne.
- Definirea unor reguli de formatare a textului, inclusiv recunoașterea titlurilor, listelor, butoanelor colorate și textului formatat (italic, îngroșat, italic-îngoșat).
- Transformarea structurii de date interne în cod HTML valid, conform regulilor de formatare specificate.

2. Complexitate

Complexitatea în cadrul unei aplicații software se referă la nivelul de dificultate sau de rafinare a algoritmilor și a structurilor de date utilizate pentru a rezolva anumite probleme sau pentru a implementa anumite funcționalități. Comparând aplicația de interpretare folosind Lex și Yacc cu alte aplicații similare de interpretare a limbajelor de programare, putem evidenția atât diferențele, cât și similaritățile în ceea ce privește complexitatea și abordarea lor.

Compararea Aplicației noastre cu Alte Aplicații Similare:

Pentru a oferi o comparație relevantă, am ales să analizăm proiectul nostru în raport cu un exemplu din laborator, care transformă codul scris în limbaj C în fișierul corespunzător cu rezultatul codului prin intermediul Lex și Yacc. Această comparație ne va ajuta să înțelegem mai bine complexitatea celor două aplicații și abordările utilizate.

Din punct de vedere al complexității, aplicația noastră este capabilă să parseze fișiere mult mai mari și mai complexe, având mai multe funcții și mai multe reguli de parsare. Acest lucru se datorează nevoii de a gestiona diverse tipuri de text, culori și stiluri, precum și structura detaliată a documentului. În comparație, exemplul din laborator se concentrează pe interpretarea și execuția unui subset de cod C, ceea ce necesită gestionarea expresiilor și a structurilor de control, dar într-un context mai limitat din punct de vedere al dimensiunii și complexității codului deoarece acesta funcționează doar pentru secvențe de cod simpliste ce folosesc If, While, Else și Print.

Ambele aplicații demonstrează puterea și flexibilitatea lex și yacc în construirea de interpreți și compilatoare, fiecare abordând complexitatea din perspective diferite, în funcție de scopurile și cerințele specifice ale proiectului.

Tipuri de reguli:

Vom explora diferite tipuri de reguli și funcțiile lor, explicând cum sunt folosite în contextul proiectului.

Lex - Definirea Tokenilor și Expresiilor Regulate

Regulile Lex definesc tokeni utilizând expresii regulate. Aceste tokeni sunt folosiți de parserul Yacc pentru a identifica structura textului de intrare.
Exemplu: regulile pot recunoaște titluri, linii goale, asteriscuri pentru formatare și text general.

Acțiunile sunt efectuate atunci când sunt potrivite modele. Aceste acțiuni implică de obicei stocarea informațiilor sau transformarea textului de intrare.
Acțiunile pot stoca caractere și menține lungimile liniilor într-un buffer pentru procesare ulterioară.

Yacc - Definirea Regulilor de Gramatică și Producții

Regulile Yacc definesc gramatica limbajului folosind producții care descriu cum pot fi combinate tokenii. Yacc gestionează structuri imbricate și recursivitate pentru a administra caracteristici complexe ale limbajului, cum ar fi listele imbricate sau textul formatat în cadrul altui text.

Acțiunile semantice sunt blocuri de cod C din cadrul regulilor Yacc care efectuează acțiuni precum setarea atributelor sau stocarea valorilor. Aceste acțiuni pot seta stilul textului pe baza numărului de asteriscuri sau altor semne de formatare.

Atât Lex, cât și Yacc includ mecanisme pentru detectarea și gestionarea erorilor, asigurând o parsare robustă și mesaje de eroare semnificative.
Aceste mecanisme ajută la identificarea și corectarea problemelor în procesul de analiză.

- Reguli pentru Text Normal (TEXT):

Aceste reguli sunt utilizate pentru a identifica și procesa secțiunile de text normal din document. Textul normal este convertit fără stiluri speciale sau formatare.
Yacc: element: TEXT
Lex: <INITIAL>{STRING}.

- Reguli pentru Titluri (BEGINTITLE, FINALIZETITLE):

Aceste reguli identifică începutul și sfârșitul unui titlu. Titlurile sunt convertite în etichete HTML <h1>, <h2>, etc., în funcție de nivelul titlului determinat de numărul de caractere #.
Yacc: title: BEGINTITLE TEXT FINALIZETITLE.

- Reguli pentru Liste (STARTLIST, ITEMLIST, FINALIZELIST):

Aceste reguli sunt utilizate pentru a defini structura listelor în document. Listele pot avea mai mulți itemi, fiecare fiind procesat pentru a permite formatarea corespunzătoare în HTML.
Yacc: list: STARTLIST list_texts next_lists.

- Reguli pentru Stiluri de Text (ASTERISK):

Aceste reguli permit aplicarea stilurilor de text, cum ar fi italic, bold și bold italic. Stilurile sunt indicate prin caractere specifice în text (* pentru italic, ** pentru bold).
Yacc: italic: ASTERISK TEXT ASTERISK {lines[$2].style = ITALIC_STYLE;}.

- Reguli pentru Gestionarea Listei (ITEMLIST, FINALIZELIST):

Aceste reguli determină tipul fiecărui item din listă (de exemplu, primul, nou, ultimul) și ajustează formatul HTML corespunzător.
Yacc: next_lists: ITEMLIST list_texts next_lists și next_lists: FINALIZELIST.

- Reguli pentru Butoane (BUTTON)

Aceste reguli identifică și procesează butoanele din document. Fiecare buton este convertit într-o etichetă HTML <button> cu proprietăți specifice, cum ar fi culoarea și textul butonului.

Yacc: button: BUTTON color=BUTTONCOLOR text=TEXT
Lex: BUTTON "BUTTON"

- Reguli pentru Text Bold (BOLD)

Aceste reguli identifică și procesează secțiunile de text bold. Textul bold este indicat prin caractere specifice (**).

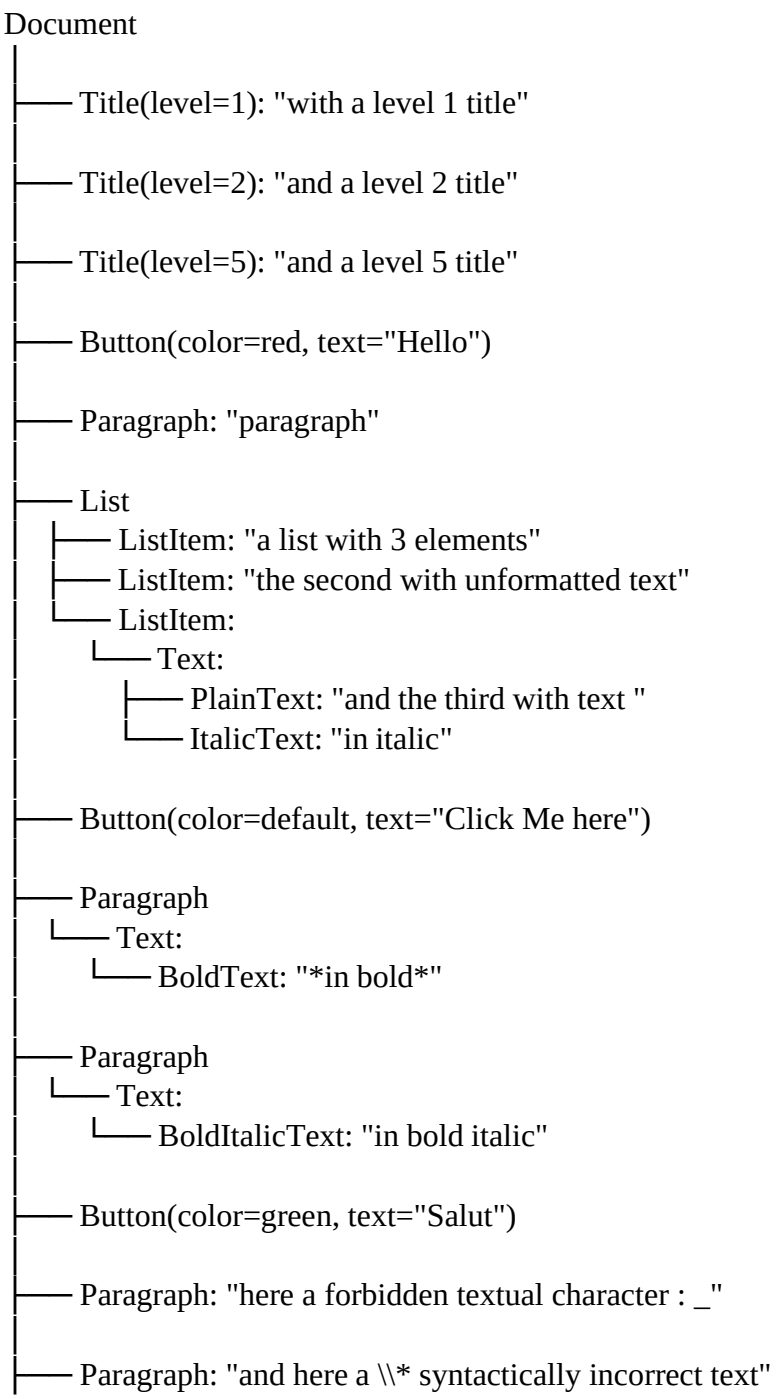
Yacc: bold: DOUBLEASTERISK TEXT DOUBLEASTERISK {lines[$2].style = BOLD_STYLE;}
Lex: DOUBLEASTERISK "**"

- Reguli pentru Paragrafe (PARAGRAPH)

Aceste reguli identifică secțiunile de text care constituie paragrafe. Fiecare paragraf este convertit într-o etichetă HTML <p>.

Yacc: paragraph: PARAGRAPH TEXT
Lex: PARAGRAPH "PARAGRAPH"

3. Arbore de parsare



4. Rezolvare Ambiguitate

Ambiguitatea într-o gramatică apare atunci când un șir de intrare poate fi interpretat (parsat) în mai multe moduri diferite, ceea ce duce la mai multe arbori de parsare pentru același șir. Ambiguitatea poate fi problematică deoarece face ca comportamentul parserului să fie nedeterminist.

Ambiguitatea este gestionată în codul nostru prin utilizarea:

- **Stărilor Lexer**: Permite comutarea între diferite moduri de analiză lexicală, clarificând contextul.
- **Delimitatori Specifici**: Simboluri speciale și reguli clare pentru a diferenția între tipurile de text sau culorile butoanelor.
- **Ordinea Reguli în Yacc**: Ordinea în care sunt definite regulile în Yacc ajută la rezolvarea ambiguităților, garantând că regula corectă este aplicată în contextul potrivit.

Prin aceste tehnici, codul nostru reușește să evite ambiguitățile comune și să parseze corect inputul în format HTML.

5. Implementare

Proiectul este structurat in doua fisiere principale: lex & yacc.

```
// Structura urmatoare stocheaza informatii despre fiecare linie de text.
typedef struct structLine {
    int position;    // Pozitia: Indicele de inceput al liniei in bufferul original de text.
    int length;      // Lungimea: Numarul de caractere din linie.
    int textType;    // Tipul textului: Indica tipul de text (normal, titlu, item de lista, linie goala).
    int titleType;   // Tipul titlului: Specifica nivelul titlului (h1, h2, h3), utilizat doar daca linia este un titlu.
    int style;       // Stilul: Specifica stilul textului (italic, bold, bold italic), utilizat doar pentru text normal.
    int itemType;    // Tipul itemului: Specifica tipul in cadrul unei liste (primul item, item nou, ultimul item), utilizat daca linia este
    un item de lista.
    int buttonStyle; // Stilul: Specifica stilul butonului.
    char buttonColor[30];
} line;

// Definitii pentru diferite tipuri de text.
#define TEXT_NORMAL 0
#define TEXT_ITEM 1
#define TEXT_TITLE 2
#define EMPTY_LINE 3
#define TEXT_BUTTON 4

// Definitii pentru diferite tipuri de butoane.
#define BUTTON_STYLE_NORMAL 0
#define BUTTON_STYLE_PRIMARY 1
#define BUTTON_STYLE_SECONDARY 2

// Definitii pentru diferite tipuri de stiluri.
#define NONE_STYLE -1
#define ITALIC_STYLE 0
#define BOLD_STYLE 1
#define BOLD_ITALIC_STYLE 2
```

În Lex avem urmatoarele funcții principale:

```
void getTextBetweenFromAndTo(const char* text, const int from, const int to, char* out)
```

Copiază o sectiune de text dintr-un buffer text intr-un alt buffer out dintre 2 pozitii.

void endParagraphIfOpen()
Se ocupa de inchiderea unui paragraf HTML daca acesta este deschis.

void processText(line *currentLine*)
Se ocupa de procesarea si afisarea textului formatat intr-un paragraf HTML.

void processList(line *currentLine*)
Se ocupa de procesarea si afisarea textului formatat intr-o lista HTML.

void processTitle(line *currentLine*)
Se ocupa de procesarea si afisarea titlului formatat in HTML.

void processEmptyLine()
Se ocupa de procesarea si inchiderea unui paragraf HTML deschis in cazul aparitiei unei linii goale.

void insertText(*char** *text*)
Insereaza textul dat in bufferul de caractere si in structura de linii.

void addTitle(*char** *title*)
Insereaza textul dat ca titlu in bufferul de caractere si in structura de linii.

void addItemToList(*char** *item*)
Insereaza textul dat ca element de lista in bufferul de caractere si in structura de linii.

void addEmptyLine()
Insereaza o linie goala in structura de linii.

int getTitleLevel(*char** *title*)
Determina nivelul titlului dat pe baza numarului de caractere '#' din sirul de caractere.

*char** clean(*char** *text*)
Curata sirul de caractere de caracterele speciale "\\" urmate de "*" si returneaza un nou sir curatat.

*char** getColorHex(*char** *colorName*)
Converteste numele culorii in formatul hexazecimal.

void processButton(line *currentLine*)
Se ocupa de procesarea unui buton.

void addButton(*char** *button*)
Insereaza textul dat ca buton in bufferul de caractere si in structura de linii.

void addButtonWithColor(*char** *button*, *char** *colorHex*)
Insereaza textul dat ca buton in bufferul de caractere si in structura de linii, cu culoarea specificata.

Fișierul yacc:

```
button: BUTTON {
    lines[$1].textType = TEXT_BUTTON;
}

list: STARTLIST list_texts next_lists {
    if ($1 == lastIndexEndList)
        lines[$1].itemType = FIRST_AND_LAST_ITEM;
    else
        lines[$1].itemType = FIRST_ITEM;
};

next_lists: ITEMLIST list_texts next_lists {
    if ($1 == lastIndexEndList)
        lines[$1].itemType = NEW_AND_LAST_ITEM;
    else
        lines[$1].itemType = NEW_ITEM;
}
| FINALIZELIST {
    lines[$1].itemType = LAST_ITEM;
    lastIndexEndList = $1;
};

text_formatted: italic | bold | bold_italic {};

italic: ASTERISK TEXT ASTERISK {
    lines[$2].style = ITALIC_STYLE;
};
```

2. Funcționalitate

Mai departe vom vedea cum poate fi folosită această aplicație.
Am creat un fisier MakeFile de unde se vor rula automat comenzile.

src > M makefile

```
@:
    lex project.lex
    yacc project.yacc
    yacc -d project.yacc
    gcc -Wall -c lex.yy.c
    gcc -Wall y.tab.c lex.yy.o -lfl -o analyzer
clean:
    rm -f *.c *.h *.o analyzer
```

Pentru a rula acest fișier vom folosi comanda: make.
La final, pentru a avea access la rezultat vom rula comanda: ./analyzer < test.md > outputFile.html

3. Avantajele Gramaticii Alese

În procesul de definire a gramaticii pentru interpretarea limbajului de programare în HTML, alegerea unei gramatici bine gândite poate oferi numeroase avantaje. Acestea sunt evidente în comparație cu gramaticile folosite în alte proiecte similare. În special, gramatica utilizată în proiectul nostru aduce o serie de beneficii, cu accent pe flexibilitatea în compunerea simbolurilor descrise.

Gramatica aleasă permite o varietate de simboluri pentru marcarea textului, inclusiv titluri, elemente de listă și formatarea textului (italic, bold). Elementele gramaticii sunt bine structurate și documentate, permițând o dezvoltare eficientă și o gestionare ușoară a modificărilor. În alte proiecte, gramatica poate fi mai complexă sau mai puțin clar definită, ceea ce poate duce la dificultăți în înțelegerea și gestionarea acesteia.

4. Teste

Test 1:
Fișier de intrare:

```
1   ###  with a level 3 title
2
3   paragraph
4   * a list with 3 elements
5   * the second
6   * and the third
7   * and the forth
8
9   then a text *in bold*
10
11  button Hello
12
13  project LFT
14
15  button green Yay
```

Fișier de ieșire:

```
Input text :
 with a level 3 title paragraph a list with 3 elements the second and the third and the forth  then a text  in bold Hello project LFT Yay

<!DOCTYPE html>
<html>
<head>
    <title>Title html</title>
    <meta charset="utf-8"/>
</head>
<body>
<h3>with a level 3 title</h3>
<p>paragraph</p>
<ul>
  <li>a list with 3 elements</li>
  <li>the second</li>
  <li>and the third</li>
  <li>and the forth </li>
</ul>
<p>then a text <i>in bold</i></p>
<button>Hello</button>
<p>project LFT</p>
<button style="background-color: #00FF00;">Yay</button>
</body>
</html>
```

Pagină web:

with a level 3 title

paragraph

- a list with 3 elements
- the second
- and the third
- and the forth

then a text *in bold*

Hello

project LFT

Yay

Test 2:

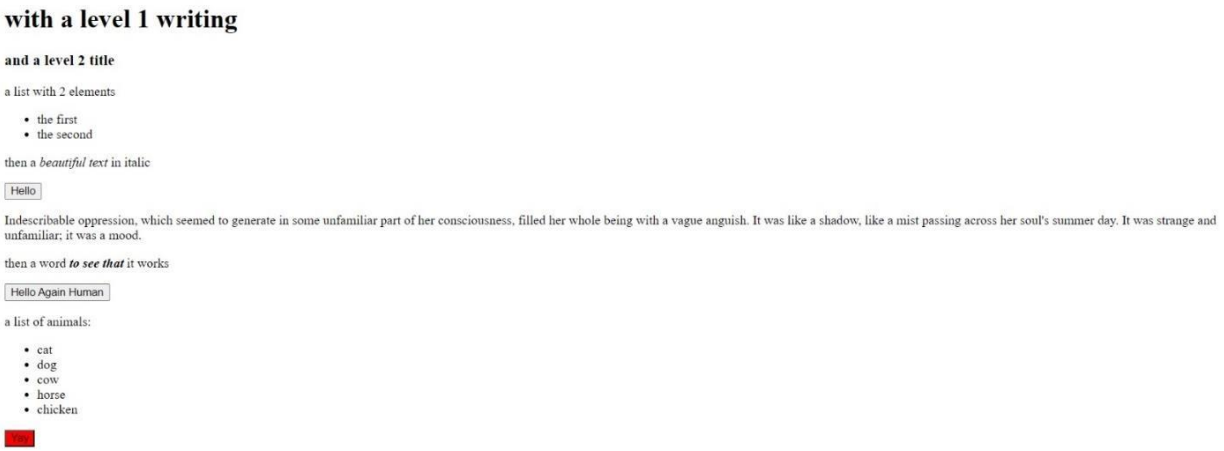
Fișier de intrare:

```
#  with a level 1 writing
### and a level 2 title

a list with 2 elements
* the first
* the second

then a beautiful text in italic

button Hello

Indescribable oppression, which seemed to generate in some unfamiliar part of her consciousness, filled her whole being with a vague anguish. It was like a shadow, like a mist passing across her soul's summer day. It was strange and unfamiliar; it was a mood.

then a word **to see that** it works

button Hello Again Human

a list of animals:
* cat
* dog
* cow
* horse
* chicken

button red Yay
```

Fișier de ieșire:

```
Input text :
 with a level 1 writing and a level 2 title a list with 2 elements the first the second then a  beautiful text  in italic Hello
 Indescribable oppression, which seemed to generate in some unfamiliar part of her consciousness, filled her whole being with a vague
 anguish. It was like a shadow, like a mist passing across her soul's summer day. It was strange and unfamiliar; it was a mood. then a word
 to see that  it works Hello Again Human a list of animals: cat dog cow horse chicken Yay

<!DOCTYPE html>
<html>
<head>
    <title>Title html</title>
    <meta charset="utf-8"/>
</head>
<body>
<h1>with a level 1 writing</h1>
<h3>and a level 2 title</h3>
<p>a list with 2 elements</p>
<ul>
  <li>the first</li>
  <li>the second</li>
</ul>
<p>then a <i>beautiful text</i> in italic</p>
<button>Hello</button>
<p>Indescribable oppression, which seemed to generate in some unfamiliar part of her consciousness, filled her whole being with a vague
anguish. It was like a shadow, like a mist passing across her soul's summer day. It was strange and unfamiliar; it was a mood.</p>
<p>then a word <i><strong>to see that</strong></i> it works</p>
<button>Hello Again Human</button>
<p>a list of animals:</p>
<ul>
  <li>cat</li>
  <li>dog</li>
  <li>cow</li>
  <li>horse</li>
  <li>chicken</li>
</ul>
<button style="background-color: #FF0000;">Yay</button>
</body>
```

Pagină web:



with a level 1 writing

and a level 2 title

a list with 2 elements

- the first
- the second

then a *beautiful text* in italic

Hello

Indescribable oppression, which seemed to generate in some unfamiliar part of her consciousness, filled her whole being with a vague anguish. It was like a shadow, like a mist passing across her soul's summer day. It was strange and unfamiliar; it was a mood.

then a word ***to see that*** it works

Hello Again Human

a list of animals:

- cat
- dog
- cow
- horse
- chicken

Yes

Test 3:

Fișier de intrare:

```
sample test text

#  with a level 1 title
## and a level 2 title
##### and a level 5 title
button red Hello
paragraph
* a list with 3 elements
* the second with unformatted text
* and the third with text *in italic*

button Click Me

then a text **in bold**

and another ***in bold italic***

button green Salut

here a forbidden textual character : _
and here a \* syntactically incorrect text
```

Fișierul de ieșire:

```
Input text :
 sample test text with a level 1 title and a level 2 title and a level 5 title Hello paragraph a list with 3 elements the second with
 unformatted text and the third with text  in italic Click Me then a text  in bold and another  in bold italic Salut here a forbidden
 textual character :  and here a * syntactically incorrect text

<!DOCTYPE html>
<html>
<head>
    <title>Title html</title>
    <meta charset="utf-8"/>
</head>
<body>
<p>sample test text</p>
<h1>with a level 1 title</h1>
<h2>and a level 2 title</h2>
<h5>and a level 5 title</h5>
<button style="background-color: #FF0000;">Hello</button>
<p>paragraph</p>
<ul>
  <li>a list with 3 elements</li>
  <li>the second with unformatted text</li>
  <li>and the third with text <i>in italic</i></li>
</ul>
<button>Click Me</button>
<p>then a text <strong>in bold</strong></p>
<p>and another <i><strong>in bold italic</strong></i></p>
<button style="background-color: #00FF00;">Salut</button>
<p>here a forbidden textual character : and here a * syntactically incorrect text </p>
</body>
</html>
```

Pagină web:

sample test text

with a level 1 title

and a level 2 title

and a level 5 title

Hello

paragraph

- a list with 3 elements
- the second with unformatted text
- and the third with text *in italic*

Click Me

then a text **in bold**

and another ***in bold italic***

Salut

here a forbidden textual character : and here a * syntactically incorrect text

5. Concluzii și Îmbunătățiri

Proiectul nostru de interpretare și formatare a textului în documente HTML reprezintă o soluție eficientă și versatilă pentru transformarea textului simplu în conținut web interactiv și structurat. Acesta combină puterea analizei lexicale și sintactice cu flexibilitatea gramaticii pentru a oferi utilizatorilor posibilitatea de a crea documente HTML complexe și estetice.

Prin intermediul instrumentelor Lex și Yacc, am reușit să dezvoltăm un parser robust și fiabil, capabil să gestioneze o varietate de construcții sintactice și să producă ieșiri HTML corespunzătoare.

Imbunătățiri ce pot fi aduse:

- Adăugarea de funcționalități suplimentare, cum ar fi suportul pentru alte formate de export în afară de HTML.
- Implementarea unor opțiuni avansate de formatare și stilizare sau adăugarea unor elemente de HTML mai complexe.
- Dezvoltarea unei interfețe utilizator mai prietenoase și interactive pentru a facilita utilizatorilor navigarea și utilizarea funcționalităților proiectului.

10. Bibliografie

http://lab.antlr.org/
https://stackoverflow.com/
https://www.gnu.org/software/bison/
https://moodle.cs.utcluj.ro/
https://github.com/