1. INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

In an era marked by profound technological advancements and digital transformation, the realm of education stands as a beacon of innovation and adaptation. The advent of educational technology has revolutionized the way knowledge is imparted and acquired, transcending the traditional boundaries of the classroom. It is within this context that the project "Smarty Pants" emerges, an educational application meticulously crafted to cater to the intellectual and developmental needs of the youngest scholars - children aged 4 to 7.

Early childhood education is a critical juncture in a child's life, forming the foundation upon which future academic and personal success is built. Recognizing this, Smarty Pants is conceived as an educational tool that transcends conventional pedagogical methods, harnessing the power of technology to engage and inspire young learners. With a singular focus on providing a holistic learning experience, Smarty Pants seamlessly integrates interactive content and gamified elements, making learning an enjoyable and immersive endeavor

The project is structured around four distinct yet interlinked sections: "Learn," "Read," "Play," and "Profile." Each section serves a unique purpose in nurturing the intellectual, emotional, and social growth of children within the specified age range. In the "Learn" section, age-appropriate learning content is thoughtfully presented, spanning from the basics of the alphabet to more complex subjects such as animal classification and nursery rhymes. This content is enriched with multimedia resources, including text, audio, and visual aids. Furthermore, it introduces a novel dimension by incorporating speech recognition technology. Children have the opportunity to engage with the content by repeating it aloud, with their responses recorded and analyzed for accuracy. This innovative approach enables real-time feedback, distinguishing correct, incorrect, and missed words through a color-coded interface.

To instill a sense of achievement and progress, Smarty Pants employs a gamification strategy. It imposes a 75% accuracy threshold to unlock successive levels, nurturing a sense of accomplishment and motivating young learners to excel. This deliberate approach not only facilitates learning but also encourages perseverance and determination.

In the "Read" section, Smarty Pants offers a curated collection of moral stories presented through animated book features, fostering language proficiency and moral development in an entertaining format. Meanwhile, the "Play" section provides a variety of games, from puzzles to math challenges, engagingly blending learning and recreation.

The "Profile" section allows children to monitor their learning journey, track their performance, and showcase their achievements through high scores and graphical representations of progress.

This project report endeavors to comprehensively elucidate the features, design, and impact of Smarty Pants, elucidating the development process, technological intricacies, and the pedagogical principles underlying this innovative educational application. It is a testament to the commitment to revolutionizing early childhood education and empowering children with the tools they need to embark on a lifelong journey of learning and discovery.

1.2 MOTIVATION

The motivation behind the "Smarty Pants" project is deeply rooted in the recognition of the critical importance of early childhood education in a rapidly evolving digital age. Research has consistently shown that the formative years of a child's development lay the foundation for their lifelong learning journey. "Smarty Pants" was conceived with the belief that technology can be harnessed to not only engage but also empower young learners. The project seeks to address the inherent challenges in catering to the unique learning needs of children aged 4 to 7 by providing an interactive and educational platform that fosters language development, problem-solving skills, and a love for learning. It aims to bridge the gap between traditional and modern learning methods, providing a seamless fusion of education and entertainment, and making learning an enjoyable and enriching experience. The motivation for "Smarty Pants" is driven by the commitment to ensuring that young children have access to age-appropriate, interactive, and comprehensive learning tools that facilitate their cognitive and linguistic development and prepare them for a brighter and more successful educational future.

1.3 PROBLEM DEFINITION

The traditional educational resources available for young children often lack interactivity and fail to maintain their interest. As a result, children may struggle to stay engaged in the learning process, hindering their ability to grasp essential concepts effectively. There is a need for an educational platform that can effectively cater to the unique learning preferences of young kids, making the learning journey both effective and enjoyable. Early childhood education is a critical phase in a child's development and the need for effective and engaging learning tools for children. Traditional methods may not always capture their attention and enthusiasm for learning. The challenge is to create an educational platform that not only caters to their specific age group but also leverages technology to provide an interactive and captivating learning experience.

1.4 OBJECTIVE OF THE PROJECT

The primary objective of the "Smarty Pants" project is to deliver a comprehensive and age-appropriate educational platform for children aged 4 to 7, ensuring that children encounter materials that are aligned with their cognitive and developmental abilities. This encompassing initiative is aimed at engaging young learners interactively, utilizing multimedia content that integrates images, audio, and text to enhance language development and cognitive skills. A key focus is to facilitate language proficiency by incorporating speech recognition technology, enabling children to actively practice and refine their language skills while receiving constructive feedback. The project also seeks to implement a robust tracking and analysis system, recording and assessing a child's performance, thus delivering insights into their learning progress, including correct, incorrect, and missed words, along with an overall accuracy score. Beyond its educational focus, "Smarty Pants" also aims to strike a balance between entertainment and education, fostering curiosity, critical thinking, and problem-solving skills in young learners to make the learning process both enjoyable and engaging. Ultimately, this initiative strives to be a comprehensive educational resource that caters to various aspects of early childhood development, including language skills, cognitive abilities, and reading habits, all while prioritizing the privacy and safety of its young users by adhering to relevant regulations and guidelines governing data collection and storage in the context of children's educational applications.

1.5 LIMITATIONS OF THE PROJECT

Limitations of the project are:

- "Smarty Pants" is a digital application, which means it requires access to a compatible device (smartphone, tablet, or computer) and a stable internet connection. This may limit its accessibility for children who do not have access to these resources

- While the app offers age-specific content, it may not provide highly personalized learning experiences tailored to the individual needs and learning styles of each child.

- The effectiveness of speech recognition technology can be limited by factors like background noise and variations in pronunciation. It may not always accurately capture a child's speech, potentially leading to frustration.

- The app collects data on children's speech and usage patterns for analysis. Ensuring strict privacy controls and complying with data protection laws is essential to address potential privacy concerns.

- The app should not be relied upon as the sole educational resource for children. A well-rounded education also includes real-world experiences, social interaction, and hands-on learning.

- "Smarty Pants" may primarily cater to children who are proficient in the language in which the content is presented. It may not be as effective for children who speak languages other than the one supported by the app.

1.6 ORGANIZATION OF THE REPORT

This is to follow up the next chapters i.e., Chapter 2 contains the information about the system specifications. It clearly explains the libraries offered by the system. Software requirements and hardware requirements are also mentioned in the chapter. The next chapter i.e., Chapter 3 deals with the design and implementation of the project. It covers the technology that is used for the project i.e., Speech Recognition. It also contains the source code of the project and the output screenshots of the project. The last chapter i.e., Chapter 4 provides the concluding information of the project. The report ends with a list of references that have been used.

2. SYSTEM SPECIFICATIONS

2.SYSTEM SPECIFICATIONS

2.1 INTRODUCTION

The system specifications are a foundational aspect of "Smarty Pants" and serve as the blueprint for its development and functionality. These specifications outline the technical and operational requirements of the application, providing a clear roadmap for the project. They encompass a wide range of details, including hardware and software prerequisites, user interface design, data storage and management, security measures, and the overall architecture of the application. The system specifications not only define the parameters within which the application will operate but also align with the specific objectives of "Smarty Pants," ensuring that it effectively addresses the learning needs of children. By setting these specifications, we aim to create a robust and user-friendly educational platform that delivers a seamless and engaging experience while prioritizing privacy, safety, and the highest standards of technological advancement.

Nonfunctional Requirements

Nonfunctional requirements are the functions offered by the system. It includes time constraints on the development process and standards. The non-functional requirements are as follows:

- **Speed:** The system should process the given input into output within an appropriate time.
- **Ease of use:** The software should be user-friendly. Then the customers can use it easily, so it does not require much training time.
- **Reliability:** The rate of failure should be less then only the system is more reliable
- **Portability:** It should be easy to implement in any system.

Specific Requirements

The specific requirements are:

- **User Interfaces:** The external users are clients. All the clients can use this software for indexing and searching.
- **Hardware Interfaces:** The external hardware interface used for indexing and searching is the personal computers of the clients. The PCs may be laptops with wireless LANs as the internet connections provided will be wireless.
- **Software Interfaces:** The Operating System can be any version of Windows.

- **Performance Interfaces:** Design the system to handle increasing user loads as the platform gains popularity.

2.2 SOFTWARE SPECIFICATIONS

Software specification deals with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Software Requirements:

- Frontend   : HTML, CSS, JavaScript
- Backend   : Flask, Speech Recognition, Bokeh
- Database   : SQLite

2.3 HARDWARE SPECIFICATIONS

The hardware specifications of "Smarty Pants" are a crucial component of the application's design and functionality. In this section, we delve into the technical requirements that will enable the smooth operation of the platform on various devices. These specifications play a pivotal role in ensuring that "Smarty Pants" offers an engaging and accessible learning experience for children.

Hardware Requirements:

"Smarty Pants" is designed to be highly versatile and compatible with a wide range of devices. It will run seamlessly on Desktop / Laptop / Tablet / Mobile. This multi-platform approach ensures that children and their caregivers can access "Smarty Pants" conveniently on the devices they are most comfortable with, enhancing the accessibility and flexibility of the learning experience.

Audio Input and Output:

To enable the critical speech recognition functionality, "Smarty Pants" will require devices with audio input capabilities. This includes microphones that can capture the child's speech for pronunciation assessment. Additionally, the application will utilize audio output to provide real-time feedback on spoken language. This ensures that children can engage actively in spoken learning exercises, making the platform interactive and immersive.

Display and Graphics:

"Smarty Pants" places a strong emphasis on the visual and interactive elements of its content. Therefore, the application will be optimized for devices with varying screen sizes, resolutions, and graphics capabilities. This includes supporting high-resolution displays to present visually engaging content, interactive quizzes, and stories in a vibrant and child-friendly manner. The user interface will be designed to adapt seamlessly to the screen dimensions of different devices.

Internet Connectivity:

Internet connectivity is a fundamental requirement for accessing and utilizing "Smarty Pants." To ensure a smooth and uninterrupted learning experience, users will need a stable internet connection when accessing the application. This connectivity will be essential for content updates, data synchronization, and the real-time feedback provided during speech recognition exercises.

Operating System Compatibility:

"Smarty Pants" will be developed to run on a range of operating systems, including iOS, Android, and popular web browsers. The application will be optimized for compatibility with different versions of these operating systems to accommodate users with various device configurations. This adaptability is crucial to ensure that "Smarty Pants" can reach a broad user base, irrespective of their device's operating system.

3. LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 INTRODUCTION

The development of "Smarty Pants" as an educational application for children aged 4 to 7 has been guided by a thorough exploration of existing educational resources and platforms. This literature survey serves as a critical precursor to the project, offering a comprehensive review of four prominent platforms that have contributed to the understanding of effective strategies in early childhood education. By examining the strengths and limitations of each platform, we gain valuable insights that inform the development and enhancement of "Smarty Pants."

The platforms under scrutiny encompass a diverse range of online resources, including FunBrain, Starfall, PBS Kids, and Duolingo Kids, each with a unique approach to engaging and educating young learners. Through this literature survey, we aim to identify both exemplary practices that can be adopted and areas for improvement that can be addressed in the design and implementation of "Smarty Pants." By critically evaluating these platforms, we lay the foundation for a project that not only offers an engaging and interactive learning experience but also overcomes potential challenges and shortcomings faced by existing educational resources.

This literature survey delves into the adaptability, content offerings, interactive features, and overall educational approaches of these platforms. It also highlights considerations such as ad distractions, geographical restrictions, and the need for additional content. The findings from this survey serve as a vital reference point for the development of "Smarty Pants," enabling us to create an educational tool that not only harnesses the strengths of existing resources but also addresses their limitations to provide a well-rounded and effective learning experience for young children.

3.2 EXISTING SYSTEM

The literature survey conducted as part of the research for "Smarty Pants" involved a comprehensive exploration of various existing educational platforms designed for children. These platforms offer a wide array of educational resources, interactive content, and engaging learning experiences. Each platform brings its unique approach to the world of digital education for young learners.

FunBrain, established in 1997, stands as a longstanding provider of free educational games. Its versatile content spans subjects like math, grammar, science, spelling, and history, making it a valuable resource for children. FunBrain's adaptability to different grade levels and user-friendly navigation has contributed to its popularity. Moreover, the platform presents engaging children's stories that stimulate creativity and imagination.

Starfall Education Foundation focuses on early childhood education and places a significant emphasis on positive reinforcement to foster children's confidence, motivation, and overall success. It offers a diverse range of features, from kindergarten math to engaging rhymes, making it a comprehensive educational resource for young children.

PBS KIDS is committed to creating a positive impact on children's lives by providing curriculum-based entertainment. The platform's 360-degree approach integrates various media and technology to nurture knowledge, critical thinking, and imagination. It encourages children to interact as respectful citizens in a diverse society, involving parents, teachers, caregivers, and communities as learning partners

Duolingo Kids offers a unique approach by targeting language learning for children aged 6 to 12. Its interactive and engaging platform allows children to embark on the journey of learning a new language within a secure and enjoyable environment. The platform empowers parents to oversee their child's language learning progress and establish specific learning objectives.

These platforms collectively represent a diverse landscape of digital educational resources, each with its unique strengths and approaches to engaging and educating young learners. The insights gathered from this literature survey will inform the development of "Smarty Pants," ensuring it stands as a valuable addition to the educational technology landscape for children aged 4 to 7.

3.3 DISADVANTAGES OF EXISTING SYSTEM

The identified disadvantages across these platforms include the presence of distracting advertisements that can divert young users' attention from the educational content. Some sections of these platforms may require a paid membership for access, potentially limiting the availability of certain learning materials. Additionally, there may be occasional performance issues that affect the user experience. Video content accessibility may be restricted exclusively to a specific audience, potentially limiting access for international users. Furthermore, the absence of a feature for assessing

and refining pronunciation could enhance the language learning experience. These disadvantages serve as valuable points of consideration for the development of more effective and user-friendly educational platforms for young learners.

3.4 PROPOSED SYSTEM

Fig 2.4.1: Existing system and proposed system

In response to the identified limitations of existing educational platforms for children, the proposed system for "Smarty Pants" aims to bridge these gaps and provide an improved learning experience. Recognizing the disadvantages such as distracting advertisements, access limitations, and performance issues found in some existing platforms, "Smarty Pants" will be thoughtfully designed to ensure a distraction-free environment and equitable access for all users. The system will prioritize a seamless and engaging user experience, mitigating occasional performance issues. Building upon the insights gathered, "Smarty Pants" will address international access to educational content, striving to make it available to a broader audience. Moreover, the system will introduce features for assessing and refining pronunciation, thus enhancing language acquisition. By taking into account these disadvantages and leveraging them as areas for improvement, "Smarty Pants" will emerge as a user-centric and effective educational platform, committed to providing an optimal and enriching learning environment for children aged 4 to 7.

4. DESIGN AND IMPLEMENTATION

4.DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

HTML

Webpages are written in HTML - a simple scripting language. HTML is short for Hyper Text Markup Language.

- **Hypertext** is simply a piece of text that works as a link
- **Markup Language** is a way of writing layout information within documents.

An HTML document is a plain text file that contains text and nothing else. When a browser opens an HTML file, the browser will look for HTML codes in the text and use them to change the layout, insert images, or create links to other pages. Since HTML documents are just text files, they can be written in even the simplest text editor

CSS

CSS is an abbreviation for Cascading Style Sheets. CSS works with HTML and other Markup Languages to control the way the content is presented. Cascading Style Sheets is a means to separate the appearance of a webpage from the content of a webpage. CSS is a recommendation of the World Wide Web Consortium.

The W3C is a consortium of web stakeholders: universities, companies such as Microsoft, Netscape, and Macromedia, and experts in many web-related fields. The presentation is specified by styles, which are presented in a style sheet.

If you are familiar with word processing programs like Microsoft Word, you have probably played around at least a little bit with styles. For example, when you want to make the headline text of your document big and bold, the hard way to do it would be to select the text, select a font face and weight, and select the color.

The easier way to do it is to create a "rule", or style, for all the headlines in your document. Then all you must do is to make one rule and keep on applying that to all your headers. CSS in its most basic form works exactly like this. Instead of using tags repeatedly to control little sections of

your page, you can establish some rules to apply globally, to a single page or all the pages on your site. CSS is a great time saver.

JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node Js, Apache CouchDB, and Adobe Acrobat. JavaScript is a protype based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (functional programming) styles

It is a dynamic computer programming language. It is lightweight and mostly used as a part of web pages, whose implementations allow client-side scripts to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML
- Open and cross-platform.

The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other

databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

Flask

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management, etc.

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

WSGI

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug

It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinja2

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have a built-in abstraction layer for database handling, nor does it have formed validation support. Instead, Flask supports the extensions to add such functionality to the application. Some of the popular Flask extensions are discussed later in the tutorial.

Speech Recognition

Speech recognition in Python is the process of converting spoken language into text or commands that a computer can understand and process. It has a wide range of applications, from voice-controlled virtual assistants and transcription services to speech-to-text conversion for accessibility and more. In this detailed overview, we'll explore how to perform speech recognition in Python, including the key attributes and libraries commonly used for this task.

Key Attributes of Speech Recognition are

- The primary goal of speech recognition systems is to accurately transcribe spoken words into text. Achieving high accuracy is crucial, especially in applications like transcription services, where even small errors can be problematic.
- Many applications require real-time speech recognition, such as voice assistants or automated phone systems. Achieving low latency in processing spoken language is essential in such cases.
- Speech recognition systems should ideally support multiple languages and accents. Robust language support makes these systems more versatile and accessible.
- Dealing with background noise is a common challenge in real-world scenarios. A good speech recognition system should be able to filter out noise and focus on the primary spoken content.
- Some applications require the system to recognize the speech of different speakers. Speaker-independent models should be capable of handling various voice characteristics and styles.

Speech Recognition Libraries in Python:

Python offers several libraries and APIs for implementing speech recognition. The choice of the library depends on your specific project requirements and whether you need to work with pre-trained models or create custom ones. Here are some popular options:

- SpeechRecognition is a versatile Python library that provides an easy-to-use interface for working with various speech recognition engines, including Google Web Speech API, Sphinx, and more. It's a great choice for simple speech-to-text applications.
- CMU Sphinx (PocketSphinx) is a set of speech recognition systems developed at Carnegie Mellon University. PocketSphinx, a lightweight version of Sphinx, is often used for embedded and mobile applications.

- Google Cloud Speech-to-Text API offers a cloud-based API that provides high-quality automatic speech recognition. It supports multiple languages and can handle both real-time and batch processing.
- Mozilla DeepSpeech is an open-source automatic speech recognition engine by Mozilla. It's based on deep learning and has achieved competitive results in various benchmarks.

Basic Speech Recognition Process:

To perform speech recognition in Python, you typically follow these steps:

- You start with an audio source, which can be a live microphone feed or a recorded audio file. Many libraries offer tools for capturing audio input.
- You may need to preprocess the audio data to remove noise, normalize volume, or convert it into the required format (e.g., WAV).
- You pass the preprocessed audio data to a speech recognition engine or API, which transcribes the speech into text. The choice of engine/API depends on your specific requirements.
- After obtaining the recognized text, you can perform post-processing, such as text cleaning, spell checking, or custom transformations.
- Finally, you can use the transcribed text for various applications, such as text search, voice commands, or transcriptions.

Bokeh

Bokeh is a powerful and versatile Python library for creating interactive data visualizations. It is particularly well-suited for generating web-based visualizations with a focus on interactivity. Bokeh's capabilities make it a popular choice for data scientists, analysts, and developers who need to present their data engagingly and dynamically.

One of Bokeh's standout features is its ability to produce interactive plots and dashboards that can be viewed directly in web browsers. This interactivity allows users to explore data, zoom in on specific details, pan through large datasets, and even interact with widgets for filtering and updating visualizations in real time.

Bokeh offers a high-level and concise API for creating a wide range of visualization types, including bar charts, line plots, scatter plots, heatmaps, and more. It's also possible to combine different types of plots into complex layouts and applications, making it a valuable tool for creating comprehensive data presentations.

Bokeh provides flexibility and customization options, allowing users to control the appearance of visual elements, such as colors, shapes, and labels. This level of control is essential for tailoring visualizations to specific project requirements and ensuring the visualizations are visually appealing and informative.

Furthermore, Bokeh supports a wide variety of output formats, including standalone HTML documents, server applications, and interactive dashboards that can be easily shared with others. It seamlessly integrates with popular Python libraries such as Pandas and NumPy, enabling data manipulation and analysis before creating visualizations.

Bokeh's architecture supports both Python and JavaScript, which facilitates the creation of highly interactive and responsive web-based applications. Users can harness the power of Bokeh by defining interactions and actions through Python callbacks, making it relatively easy to add dynamic behavior to visualizations.

In summary, Bokeh is a valuable asset in the Python data visualization ecosystem, offering a range of features for generating interactive and web-friendly plots and dashboards. Its ease of use, flexibility, and support for interactivity make it an excellent choice for anyone seeking to communicate complex data effectively through visually engaging web-based applications. Whether you're a data scientist, developer, or analyst, Bokeh is a versatile tool to have in your toolkit for crafting compelling data visualizations.

4.2 UML Diagrams

Flow Diagram[i]

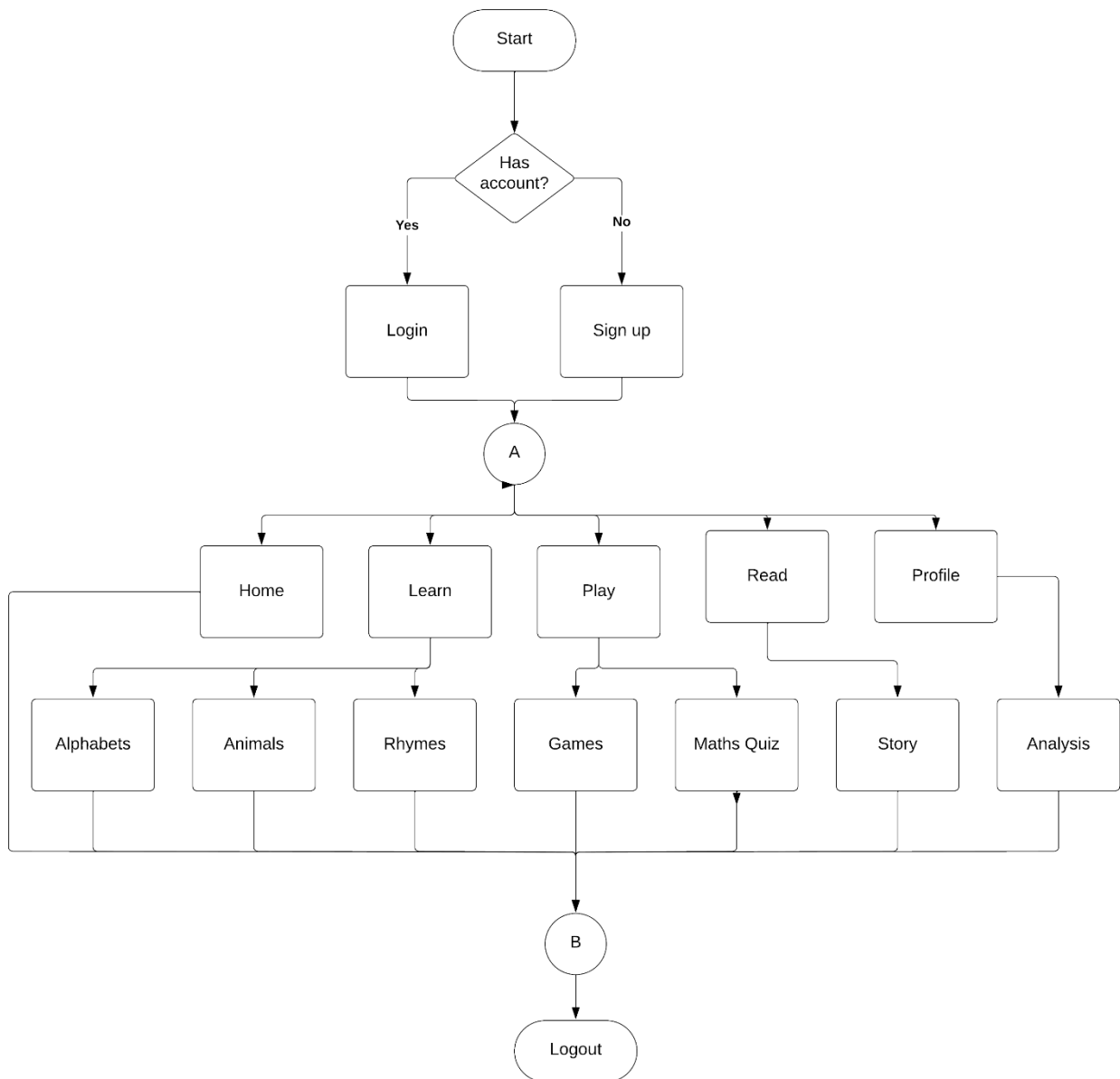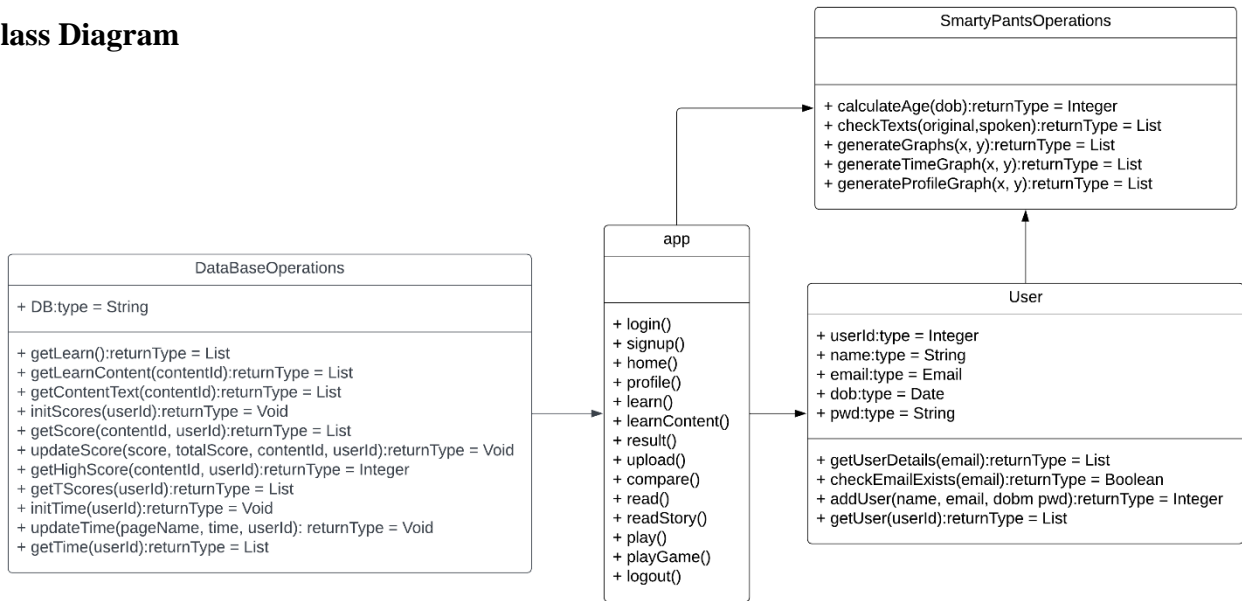Fig 4.2.1: Flow Diagram

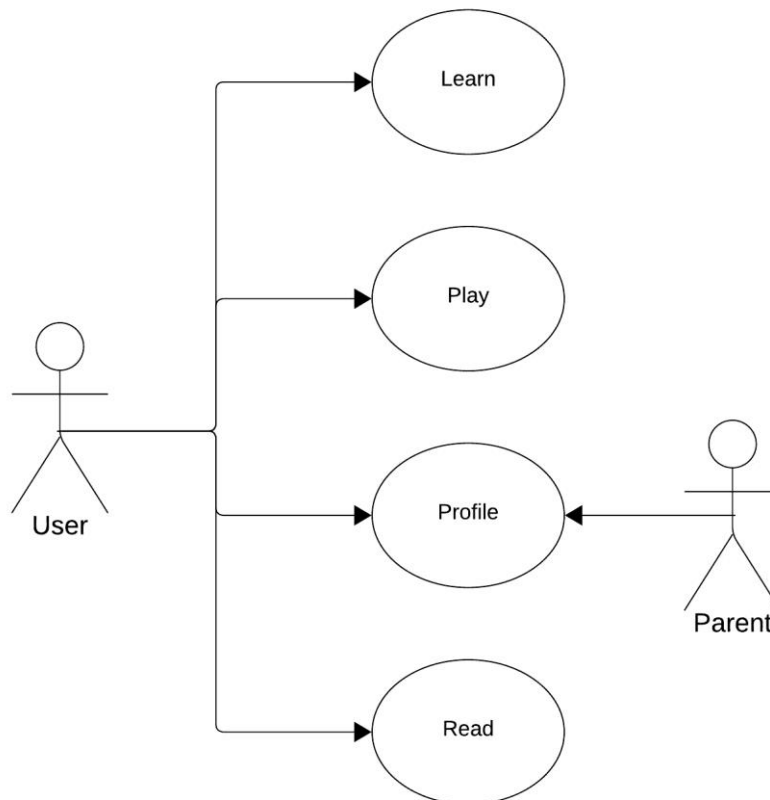## Class Diagram

Fig 4.2.2: Class Diagram

Use Case Diagram
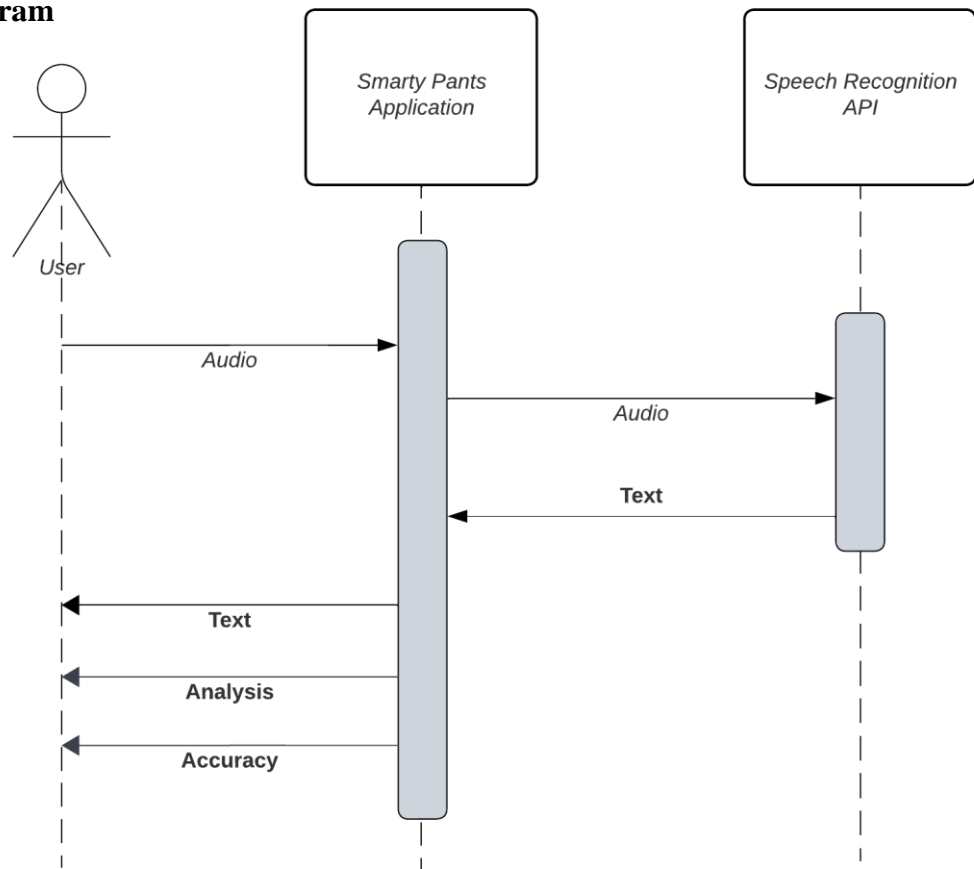

Fig 4.2.3: Use Case Diagram

Sequence Diagram

Fig 4.2.4: Sequence diagram

Component Diagram
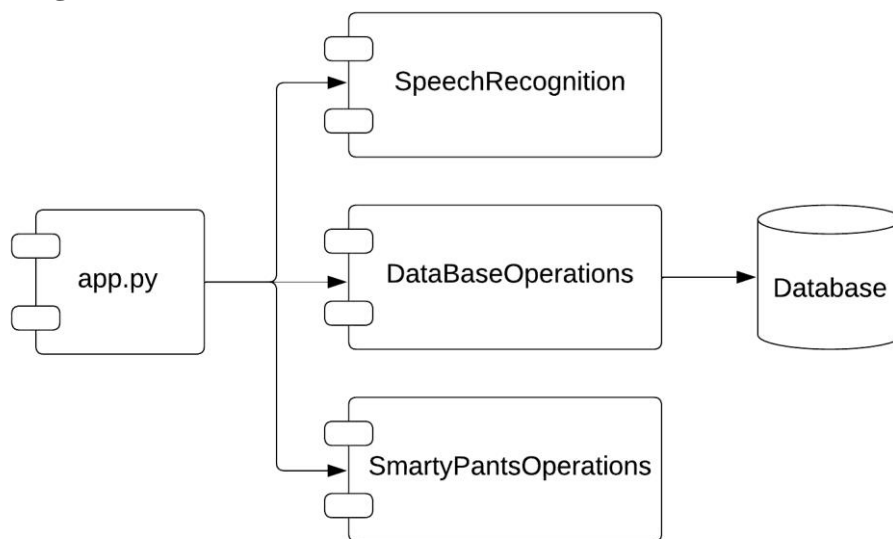

Fig 4.2.5: Component diagram

4.3 Source Code

app.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
from flask_wtf import CSRFProtect
from DataBase import DataBaseOperations as DB
from SmartyPantsOperations import Operations as OP
from flask_login import LoginManager, UserMixin, login_required, login_user, logout_user,
current_user
from datetime import timedelta, datetime
import speech_recognition as sr
import json
import pytz


app = Flask(__name__)

app.config.update(
    DEBUG = True,
    SECRET_KEY = 'SmartyPants_P_N_M#801')
csrf = CSRFProtect(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"
login_manager.login_message = u"Login to access this page."


class User(UserMixin):
    def __init__(self, userId):
        self.userId = userId
    def get_id(self):
        return self.userId
  def is_active(self):
        return True
```

```python
@app.route('/', methods=['post', 'get'])
@app.route('/login', methods=['post', 'get'])
def login():
    if current_user.is_active:
        return redirect(url_for('home'))
    if request.method == 'POST':
        email = request.form['email']
        pwd = request.form['pwd']
        exists = DB.checkEmailExists(email)
        if not exist:
            flash("No " + email + " Email exists!")
            return render_template("login.html")
        else:
            record = DB.getUserDetails(email)
            if pwd != record[2]:
                flash("Passwword does not match!")
                return render_template("login.html")
            else:
                user = User(record[0])
                login_user(user)
                session['user'] = True
                session['age'] = OP.calculateAge(record[1])
                return redirect(url_for("home"))
    return render_template("login.html")


@app.route('/signup', methods=['post', 'get'])
def signup():
    if current_user.is_active:
        return redirect('/home')
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
```

```python
        dob = request.form['dob']
        pwd = request.form['pwd']
        confPwd = request.form['confPwd']
        if pwd != confPwd:
            flash("Password does not match!")
            return render_template("login.html")
        else:
            ID = DB.addUser(name, email, dob, pwd)
            DB.initScores(ID)
            DB.initTime(ID)
            user = User(ID)
            session['user'] = True
            session['age'] = OP.calculateAge(dob)
            login_user(user)
        return redirect("/home")
    return render_template("login.html")


@app.route('/home')
@login_required
def home():
    return render_template("home.html")


@app.route('/profile')
@login_required
def profile():
    record = DB.getUser(current_user.userId)
    time = DB.getTime(current_user.userId)
    mins = []
    for sec in time:
        mins.append(round(sec/ 60, 2))
    script1, div1 = OP.generateTimeGraph(mins)
    scores = DB.getTScores(current_user.userId)
```

```
    modifiedScores = [0 if x == -1 else x for x in scores]
    script2, div2 = OP.generateProfileGraph(scores = modifiedScores)
    return render_template("profile.html", record = record, script1 = script1, div1 = div1, script2 =
script2, div2 = div2)


@app.route('/learn')
@login_required
def learn():
    records = DB.getLearn()
    scores = DB.getTScores(current_user.userId)
    age = session.get('age')
    return render_template("learn.html", records = records, scores = scores, age = age)


@app.route("/learn/<string:contentName>/<int:contentPos>", methods=['POST', 'GET'])
@login_required
def learnContent(contentName, contentPos):
    if contentPos == -1:
        return redirect(url_for('learn'))
    contentId = request.args.get('id')
    records = DB.getLearnContent(contentId)
    if 0 <= contentPos < len(records):
        lastContent = True if((contentPos + 1) == len(records)) else False
        record = records[contentPos]
        return render_template("contentLearn.html", record=record, contentPos=contentPos,
contentId = contentId, contentName = contentName,
                    lastContent = lastContent)
    elif contentPos == len(records):
        return redirect(url_for("results", contentName = contentName, contentId = contentId))
    else:
        return redirect(url_for('learn'))


@app.route("/results/<string:contentName>")
```

```python
@login_required
def results(contentName):
    contentId = request.args.get('contentId')
    scores = session.get(contentId)
    if scores == None:
        return redirect(url_for('learn'))
    texts = DB.getContentText(contentId)
    x, y  = [], []
    for text in texts:
        x.append(text[1])
    for learnId in texts:
        if str(learnId[0] - 1) not in scores.keys():
            y.append(0)
        else:
            y.append(scores[str(learnId[0] - 1)])
    totalScore = sum(y)
    avgAcc = round(totalScore / len(x), 2)
    script, div = OP.generateGraph(x, y)
    highScore = DB.getHighScore(contentId, current_user.userId)
    return render_template("results.html", script=script, div=div, totalScore = round(totalScore, 2),
highScore = highScore,
                  contentName = contentName, contentId = contentId, scores = scores, texts = texts,
avgAcc = avgAcc)


@app.route('/upload', methods=['POST'])
def upload():
    try:
        if 'audio' not in request.files:
            return jsonify({'error': 'No audio file provided'}), 400
        audio_file = request.files['audio']
        recognizer = sr.Recognizer()
        recognizer.adjust_for_ambient_noise()
```

```
      audio_data = recognizer.record(audio_file)

      text = recognizer.recognize_google(audio_data)

      return jsonify({'text': text})

   except Exception as e:

      return jsonify({'error': str(e)}), 500


@app.route('/compare', methods=['POST'])

def compare():

   try:

      speech_text = request.form.get('text')

      original_text = request.form.get('og')

      contentId = request.form.get('contentId')

      contentPos = request.form.get('contentPos')

      result, accuracy = compare_text(speech_text, original_text)

      score = json.loads(DB.getScore(contentId, current_user.userId))

      totalScore = 0

      overallScore = 0

      if contentId not in session or session[contentId] is None:

         session[contentId] = dict()

      if score == -1:

         score = dict()

      score[contentPos] = max(score.get(contentPos, 0), accuracy)

      session[contentId][contentPos] = max(session[contentId].get(contentPos, 0), accuracy)


      for individualScore in session[contentId]:

         totalScore += session[contentId][individualScore]

      for individualScore in score:

         overallScore += score[individualScore]

      maxScore = round(overallScore, 2)

      DB.updateScore(json.dumps(score), maxScore, contentId, current_user.userId)

      return jsonify({'result': result, 'accuracy': accuracy, 'score': round(totalScore, 2)})

   except Exception as e:
```

```
      app.logger.error(f'An error occurred: {str(e)}')
      return jsonify({'error': 'Internal Server Error'}), 500


@app.route('/play')
@login_required
def play():
   return render_template('play.html')


@app.route('/play/<string:game>')
@login_required
def playGame(game):
   return render_template('game/' + game + '.html')


@app.route('/read')
@login_required
def read():
   return render_template('read.html')


@app.route('/read/<string:story>')
@login_required
def readStory(story):
   return render_template('story/' + story + '.html')


@app.route('/logout')
def logout():
   logout_user()
   session.clear()
   return redirect(url_for('login'))


@login_manager.user_loader
def load_user(userId):
   return User(userId)
```

```python
@app.before_request
def before_request():
    session.permanent = True
    session.modified = True
    app.permanent_session_lifetime = timedelta(minutes=30)


@app.after_request
def after_request(response):
    utc = pytz.timezone('UTC')
    startTime = session.pop('startTime', None)
    pageName = request.path
    if startTime is not None:
        user_id = session.get('user')
        if user_id is not None:
            endTime = datetime.now(utc)
            if 'learn' in pageName:
                DB.updateTime('learn', (endTime - startTime).total_seconds(), current_user.userId)
            elif 'play' in pageName:
                DB.updateTime('play', (endTime - startTime).total_seconds(), current_user.userId)
            elif 'read' in pageName:
                DB.updateTime('read', (endTime - startTime).total_seconds(), current_user.userId)
        session['startTime'] = datetime.now(utc)
    else:
        session['startTime'] = datetime.now(utc)
    return response


if __name__ == '__main__':
    app.run(debug=True)
```

Database.py

```python
import sqlite3
class DataBaseOperations:
    _DB = "SmartyPants.db"
    def getUserDetails(email):
        con = sqlite3.connect(DataBaseOperations._DB)
        c = con.cursor()
        try:
            sql_select_query = "SELECT ID, DOB, PASSWORD FROM Users WHERE EMAIL = ?;"
            c.execute(sql_select_query, (email,))
            record = c.fetchall()
        finally:
            con.commit()
            c.close()
            con.close()
        return record[0]


    def checkEmailExists(email):
        con = sqlite3.connect(DataBaseOperations._DB)
        c = con.cursor()
        try:
            sql_select_query = "SELECT * FROM Users WHERE EMAIL = ?;"
            c.execute(sql_select_query, (email,))
            record = c.fetchall()
        finally:
            con.commit()
            c.close()
            con.close()
        if len(record) == 0:
            return False
        else:
            return True
```

```python
def addUser(name, email, dob, pwd):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        x = (name, email, dob, pwd)
        insert = """INSERT INTO Users (NAME, EMAIL, DOB, PASSWORD) VALUES
(?,?,?,?);"""
        c.execute(insert, x)
        ID = c.lastrowid
    finally:
        con.commit()
        c.close()
        con.close()
    return ID


def getUser(userId):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        sql_select_query = "SELECT NAME, EMAIL, DOB FROM Users where ID = ?;"
        c.execute(sql_select_query, (userId,))
        record = c.fetchall()
    finally:
        con.commit()
        c.close()
        con.close()
    return record[0]


def getLearn():
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
```

```
    try:
        sql_select_query = "SELECT * FROM Learn;"
        c.execute(sql_select_query)
        record = c.fetchall()
    finally:
        con.commit()
        c.close()
        con.close()
    return record


def getLearnContent(contentId):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        sql_select_query = "SELECT * FROM Content" + contentId + ";"
        c.execute(sql_select_query)
        record = c.fetchall()
    finally:
        con.commit()
        c.close()
        con.close()
    return record


def getContentText(contentId):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        sql_select_query = "SELECT contentId, text FROM Content" + contentId + ";"
        c.execute(sql_select_query)
        record = c.fetchall()
    finally:
        con.commit()
```

```
        c.close()

        con.close()

     return record


  def initScores(userId):

     con = sqlite3.connect(DataBaseOperations._DB)

     c = con.cursor()

     try:

        x = (userId,)

        insert = """INSERT INTO Score (userId) VALUES (?);"""

        c.execute(insert, x)

     finally:

        con.commit()

        c.close()

        con.close()


  def getScore(contentId, userId):

     con = sqlite3.connect(DataBaseOperations._DB)

     c = con.cursor()

     try:

        sql_select_query = "SELECT content" + str(contentId) + "Score FROM Score WHERE
userId = ?;"

        c.execute(sql_select_query, (userId,))

        record = c.fetchall()

     finally:

        con.commit()

        c.close()

        con.close()

     return record[0][0]


  def updateScore(score, totalScore, contentId, userId):

     con = sqlite3.connect(DataBaseOperations._DB)
```

```python
        c = con.cursor()
        try:
            x = (score, totalScore, userId)
            update = "UPDATE Score SET content" + contentId + "Score = ?, TContent" + contentId
+"Score = ? WHERE userId = ?;"
            c.execute(update, x)
        finally:
            con.commit()
            c.close()
            con.close()


    def getHighScore(contentId, userId):
        con = sqlite3.connect(DataBaseOperations._DB)
        c = con.cursor()
        try:
            sql_select_query = "SELECT TContent" + str(contentId) + "Score FROM Score WHERE
userId = ?;"
            c.execute(sql_select_query, (userId,))
            record = c.fetchall()
        finally:
            con.commit()
            c.close()
            con.close()
        return record[0][0]


    def getTScores(userId):
        con = sqlite3.connect(DataBaseOperations._DB)
        c = con.cursor()
        try:
            sql_select_query = """SELECT TContent1Score, TContent2Score, TContent3Score,
TContent4Score, TContent5Score,
            TContent6Score, TContent7Score, TContent8Score, TContent9Score, TContent10Score,
```

```
TContent11Score,
        TContent12Score, TContent13Score, TContent14Score, TContent15Score FROM Score
WHERE userId = ?;"""
        c.execute(sql_select_query, (userId,))
        record = c.fetchall()
    finally:
        con.commit()
        c.close()
        con.close()
    return record[0]


  def initTime(userId):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        x = (userId,)
        insert = """INSERT INTO Time (userId) VALUES (?);"""
        c.execute(insert, x)
        ID = c.lastrowid
    finally:
        con.commit()
        c.close()
        con.close()
    return ID


  def updateTime(pageName, time, userId):
    con = sqlite3.connect(DataBaseOperations._DB)
    c = con.cursor()
    try:
        sql_select_query = "SELECT " + pageName + " FROM Time WHERE userId = ?;"
        c.execute(sql_select_query, (userId,))
        record = c.fetchall()
```

```
        prevTime = record[0][0]

        x = (prevTime + time, userId)

        update = "UPDATE Time SET " + pageName + " = ? WHERE userId = ?;"

        c.execute(update, x)

    finally:

        con.commit()

        c.close()

        con.close()


    def getTime(userId):

        con = sqlite3.connect(DataBaseOperations._DB)

        c = con.cursor()

        try:

            sql_select_query = "SELECT learn, play, read FROM Time WHERE userId = ?;"

            c.execute(sql_select_query, (userId,))

            record = c.fetchall()

        finally:

            con.commit()

            c.close()

            con.close()

        return record[0]
```

SmartyPantsOperations.py

```
from datetime import date, datetime

from bokeh.embed import components

from bokeh.models import TabPanel, Tabs, Span, HoverTool, ColumnDataSource, Range1d,

PanTool, SaveTool, BoxZoomTool, WheelZoomTool, ResetTool

from bokeh.plotting import figure

import pandas as pd

from bokeh.palettes import Light, Category10

from math import pi

from bokeh.transform import cumsum
```

```python
from bokeh.layouts import gridplot


class Operations:
    def calculateAge(dob):
        x = datetime.strptime(dob, '%Y-%m-%d')
        today = date.today()
        one_or_zero = ((today.month, today.day) < (x.month, x.day))
        year_difference = today.year - x.year
        y = 1 if one_or_zero else 0
        curr_age = year_difference - y
        return curr_age


    def compare_text(original_text, given_text):
        original_words = original_text.split()
        given_words = given_text.split()
        word_status_list = []
        given_word_count = {}
        for word in given_words:
            given_word_count[word.lower()] = given_word_count.get(word.lower(), 0) + 1
        for original_word in original_words:
            word_dict = {}
            lowercase_original_word = original_word.lower()
            if lowercase_original_word in given_word_count and
        given_word_count[lowercase_original_word] > 0:
                word_dict["word"] = original_word
                word_dict["status"] = "correct"
                given_word_count[lowercase_original_word] -= 1
            else:
                word_dict["word"] = original_word
                word_dict["status"] = "wrong"
            word_status_list.append(word_dict)
        for word, count in given_word_count.items():
```

```python
        for _ in range(count):
            word_dict = {"word": word, "status": "missing"}
            word_status_list.append(word_dict)
    correct_count = sum(1 for word_dict in word_status_list if word_dict["status"] == "correct")
    missing_count = sum(1 for word_dict in word_status_list if word_dict["status"] ==
  "missing")
    wrong_count = sum(1 for word_dict in word_status_list if word_dict["status"] == "wrong")
    total_words = correct_count + missing_count + wrong_count
    accuracy = round((correct_count / total_words) * 100, 2)
    return word_status_list, accuracy


  def generateGraph(x, y):
    source = ColumnDataSource(data=dict(x=x, y=y))
    hover = HoverTool(tooltips=[('Score', "@y")]
    upper = Span(location=75, dimension='width', line_color='green', line_width=2,
line_dash='dashed', name="Pass", hover_line_cap="round")
    bar_plot = figure(title="Bar Plot", x_range=x, y_range=Range1d(0,101), width=800,
height=500, toolbar_location="below", tools = "pan,wheel_zoom,box_zoom,reset")
    bar_plot.vbar(x='x', top='y', width=0.5, source=source, line_color="navy",
fill_color="orange", fill_alpha=0.5)
    bar_plot.add_layout(upper)
    bar_plot.xaxis.axis_label = "Content"
    bar_plot.xaxis.axis_line_width = 3
    bar_plot.yaxis.axis_label = "Score"
    bar_plot.yaxis.major_label_text_color = "orange"
    bar_plot.toolbar.logo = None
    bar_plot.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
    circle_plot = figure(title="Circle Plot", x_range=x, y_range=Range1d(0,101), width=800,
height=500, toolbar_location="below",
                tools="pan,wheel_zoom,box_zoom,reset")
    circle_plot.circle(x, y, size=25, color="firebrick", alpha=0.5)
```

```
circle_plot.add_layout(upper)
circle_plot.add_tools(hover)
circle_plot.xaxis.axis_label = "Content"
circle_plot.xaxis.axis_line_width = 3
circle_plot.yaxis.axis_label = "Score"
circle_plot.yaxis.major_label_text_color = "orange"
circle_plot.toolbar.logo = None
circle_plot.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
tab1 = TabPanel(child=bar_plot, title="Bar Graph Analysis")
tab2 = TabPanel(child=circle_plot, title="Dot Graph Analysis")
tabs = Tabs(tabs=[tab1, tab2])
script, div = components(tabs)
return script, div


def generateTimeGraph(time):
    data = {"Activity": ["Learning", "Playing", "Reading"], "Time": time}
    data = pd.DataFrame(data)
    data['angle'] = data['Time'] / data['Time'].sum() * 2 * pi
    data['colors'] = Light[len(data)]
    hover = HoverTool(tooltips=[("Activity", "@Activity"),('Time spent', "@Time mins")])
    p = figure(title="Time Spent on Activities", height=400, x_range=(-0.5, 0.75), width=500)
    p.toolbar.logo = None
    p.xaxis.major_label_text_font_size = '0pt'
    p.xaxis.axis_label = None
    p.xgrid.grid_line_color = None
    p.xaxis.axis_line_width = 0
    p.xaxis.major_tick_line_color = None
    p.xaxis.minor_tick_line_color = None
    p.yaxis.major_label_text_font_size = '0pt'
    p.yaxis.axis_label = None
    p.ygrid.grid_line_color = None
```

```
    p.yaxis.axis_line_width = 0

    p.yaxis.major_tick_line_color = None

    p.yaxis.minor_tick_line_color = None

    p.wedge(x=0, y=1, radius=0.4,

    start_angle=cumsum('angle', include_zero=True), end_angle=cumsum('angle'),

    line_color="white",  legend_field='Activity', source=data, fill_color='colors')

    p.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover, ResetTool()]

    hover = HoverTool(tooltips=[("Activity", "@x"),('Time spent', "@top mins")])

    barGraph = figure(x_range=data['Activity'], height=400, width=500)

    barGraph.vbar(x=data['Activity'], top=data['Time'], width=0.5, color=Light[3])

    barGraph.toolbar.logo = None

    barGraph.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,

ResetTool()]

    barGraph.xaxis.axis_label = "Activity"

    barGraph.xaxis.axis_line_width = 3

    barGraph.yaxis.axis_label = "Time"

    barGraph.yaxis.major_label_text_color = "orange"

    grid = gridplot([[p, barGraph]])

    return components(grid)


  def generateProfileGraph(scores):

    x1 = ["Alphabets-1", "Alphabets-2", "Alphabets-3", "Alphabets-4"]

    y1 = scores[0:4]


    x2 = ["Herbivores Animals", "Carnivores Animals", "Omnivores Animals"]

    y2 = scores[4:7]


    x3 = ["Fishes", "Birds", "Prehistoric Animals"]

    y3 = scores[7:10]


    x4 = ["Rhyme-1", "Rhyme-2", "Rhyme-3", "Rhyme-4", "Rhyme-5"]

    y4 = scores[10:15]
```

```
color = Category10[10]
hover = HoverTool(tooltips=[("Category", "@x"), ('Score', "@top")])


plot1 = figure(x_range=x1, title="Alphabets", height=400, width=500)
plot1.vbar(x=x1, top=y1, width=0.5, color=color[0])
plot1.toolbar.logo = None
plot1.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
plot1.xaxis.axis_label = "Content"
plot1.xaxis.axis_line_width = 3
plot1.yaxis.axis_label = "Score"
plot1.yaxis.major_label_text_color = "orange"


plot2 = figure(x_range=x2, title="Animals-1", height=400, width=500)
plot2.vbar(x=x2, top=y2, width=0.5, color=color[1])
plot2.toolbar.logo = None
plot2.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
plot2.xaxis.axis_label = "Content"
plot2.xaxis.axis_line_width = 3
plot2.yaxis.axis_label = "Score"
plot2.yaxis.major_label_text_color = "orange"


plot3 = figure(x_range=x3, title="Animals-2", height=400, width=500)
plot3.vbar(x=x3, top=y3, width=0.5, color=color[2])
plot3.toolbar.logo = None
plot3.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
plot3.xaxis.axis_label = "Content"
plot3.xaxis.axis_line_width = 3
plot3.yaxis.axis_label = "Score"
```

```
plot3.yaxis.major_label_text_color = "orange"


plot4 = figure(x_range=x4, title="Rhymes", height=400, width=500)
plot4.vbar(x=x4, top=y4, width=0.5, color=color[3])
plot4.toolbar.logo = None
plot4.tools = [SaveTool(), PanTool(), BoxZoomTool(), WheelZoomTool(), hover,
ResetTool()]
plot4.xaxis.axis_label = "Content"
plot4.xaxis.axis_line_width = 3
plot1.yaxis.axis_label = "Score"
plot4.yaxis.major_label_text_color = "orange"


grid = gridplot([[plot1, plot2], [plot3, plot4]])


script, div = components(grid)
return script, div
```

Templates > index.html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>{% block title %} {% endblock %}</title>
<link rel="stylesheet" href="{{url_for('static', filename='stylesheets/Main.css')}}">
<link rel="stylesheet" href="{{url_for('static', filename='stylesheets/nav.css')}}">
<link rel="stylesheet" href="{{url_for('static', filename='stylesheets/footer.css')}}">
<link rel="shortcut icon" type="image/ico" href="{{url_for('static', filename =
'images/favicon.ico') }}">


<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
</head>
<body>
        {% if current_user.is_authenticated %}
        {% block particles %} {% endblock %}
        <header class="navheader">
                    <div class="wrapper">
                        <div class="logo">
                                <a href="/"><img src="{{url_for('static', filename =
'images/logo.png') }}" alt="Joe Care"></a>
                        </div>
                        <div class="navbar">
                                <div class="close-nav"><button>×</button></div>
                                <nav>
                                    <ul>
                                            <li><a href="/">Home</a></li>
                                            <li><a href="/learn">Learn</a></li>
                                            <li><a href="/read">Read</a></li>
                                            <li><a href="/play">Play</a></li>
                                            <li><a href="/profile">Profile</a></li>
                                            <li><a href="/logout">Logout</a></li>


                                    </ul>
                                </nav>
                        </div>
                        <div class="menu-bar">
                                <button><i></i></button>
                        </div>
                    </div>
            </header>
        <script>
                    const theBody = document.querySelector('body');
                    const openNav = document.querySelector('.menu-bar button');
```

```
                    const closeNav = document.querySelector('.close-nav button');

                    const Navbar = document.querySelector('.navbar');

                    function showHide(){

                            Navbar.classList.toggle('show');

                    }


                    openNav.onclick = showHide;

                    closeNav.onclick = showHide;

                </script>

        {% endif %}

        {% block content %} {% endblock %}


        <!-- Footer -->

        <footer class="footer-distributed">


 <div class="footer-left">


   <h3>Smarty Pants<span><img style="height: 7.5vh;" src="{{url_for('static',

filename='images/logo.png')}}" alt=""></span></h3>


   <p class="footer-links">
     <a href="/home" class="link-1">Home</a>

     <a href="/learn">Learn</a>

     <a href="/play">Play</a>

     <a href="/read">Read</a>

     <a href="/profile">Profile</a>

     <a href="/logout">Logout</a>

   </p>

 </div>

 <div class="footer-center">

  <div>

        <p>Project By</p><i></i>
```

```
    <p><span></span>J Prem (209X1A0593) <br>P Sai Nivas (209X1A05A9) <br>M Sai Mithil
(209X1A05A3)</p>
  </div>
        <hr>
  <div>
        <p>Guided By</p><i></i>
    <p>Dr. N. Kasiviswanath<br>Head of the Department<br>Department of Computer Science
&amp; Engineering <br>G. Pulla Reddy Engineering College</p>
  </div>
 </div>
 <div class="footer-right">
   <p style="color: white; text-align: justify;" class="footer-company-about">
    <span>About the application</span>
    Smarty Pants is an innovative educational platform designed to inspire and empower young
learners aged 4 to 7. Our mission is to make learning an exciting adventure, nurturing a lifelong
love for knowledge.
    </p></div>
<hr>
</footer>
</body>
</html>
```

Templates > learnContent.html

```
<!doctype html>
{% extends 'index.html' %}
{% block title %} Learn - {{contentName}} {% endblock %}

{% block content %}
        <meta name="csrf-token" content="{{ csrf_token() }}">
        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```
<link rel="stylesheet" href="{{url_for('static', filename='stylesheets/learnContent.css')}}">
<script src="{{url_for('static', filename='js/audio.js')}}"></script>


<div class="total-container">
    <h1>{{contentName}}</h1><br>
    <div class="row">
    <div class="upper-left">
        <img src="{{url_for('static', filename='content/' + contentId|string + '/image/' +
record[1])}}" alt="IMAGE" width="600" height="300"><br>
    </div>
    <div class="upper-right">
        <div style="display: flex; font-size: 20px;"><p><strong>Say:</strong></p><p
style="padding-left: 10px">{{record[3]}}</p></div>
                <audio id="audio" class="player">
                    <source src="{{url_for('static', filename='content/' + contentId|string
+ '/audio/' + record[2])}}" type="audio/mpeg">
                </audio>
        <!-- <p>Click the button below and speak into your microphone:</p> -->
        <button id="recordButton" class="button buttonPlay" style="background-color:
#14A44D; color: #FBFBFB; width: 49%"><span><i class="fa fa-microphone"></i>Start
Recording</span></button>
        <button id="stopButton" class="button buttonStop" style="background-color:
#DC4C64; color: #FBFBFB; width: 48.9%;"><span>Stop Recording <i class="fa fa-microphone-
slash"></i></span></button>
    </div>
    </div>
    <div class="row">
    <div class="bottom-left">
    <canvas id="visualization" width="600" height="200"></canvas>
    </div>
    <div class="bottom-right"><div id="bottom-right" style="display: none;">
        <table style="margin-left: auto; margin-right: auto; font-size: 1rem; background-
```

color: #E8E8E8; border-radius: 15px; padding: 25px">

                                        <tr>

                                              <td style="text-align: left;color: #183e97; width: 25%">You Said</td>

                <td style="text-align: center; padding: 0px 25px;">:</td>

                <td style="text-align: left;"></td>

            </tr><tr>

                <td style="text-align: left;color: #183e97; width: 25%">Analysis</td>

                <td style="text-align: center; padding: 0px 25px;">:</td>

                <td style="text-align: left;"></td>

            </tr>

            <tr>

                <td style="text-align: left; color:#183e97; width: 25%">Accuracy</td>

                <td style="text-align: center; padding: 0px 25px;">:</td>

                <td style="text-align: left;">%</td>

            </tr><tr>

                <td style="text-align: left; color:#183e97; width: 25%">Total Score</td>

                <td style="text-align: center; padding: 0px 25px;">:</td>

                <td style="text-align: left;"></td>

            </tr>

        </table>

    </div>

    </div>

    </div>

    <div style="float: left;">

    {% if contentPos != 0%}

    <button class="buttonHover2" style="background-color: #ffffc1;">Previous</button>

    {% endif %}

```
            </div>
        {% if lastContent == True %}
                <div style="float: right;">
                <a href="/learn/{{contentName}}/{{contentPos + 1}}?id={{contentId}}"><button
class="buttonHover1" style="background-color: #c1fdc1;"><span>End Learning
</span></button></a>
                </div>
        {% else %}
                <div style="float: right;">
                <a href="/learn/{{contentName}}/{{contentPos + 1}}?id={{contentId}}"><button
class="buttonHover1" style="background-color: #c1fdc1;"><span>Next </span></button></a>
                </div>
        {% endif %}


</div>
<br>
<script>
const recordButton = document.getElementById('recordButton');
const stopButton = document.getElementById('stopButton');
const output = document.getElementById('output');
var og = "{{record[3]}}";
var contentId = "{{contentId}}";
var contentPos = "{{contentPos}}";
var contentName = "{{contentName}}";
let isRecording = false;
let recognition = null;
let animationId = null;
const csrfToken = document.querySelector('meta[name="csrf-token"]').content;


function getRainbowColor(index, totalBars) {
  const hue = (360 / totalBars) * index;
  return `hsl(${hue}, 100%, 50%)`;
```

```
}
recordButton.addEventListener('click', () => {
        if (!isRecording) {
                recognition = new webkitSpeechRecognition() || new SpeechRecognition();
                recognition.continuous = true;

                recognition.onstart = () => {
                        recordButton.textContent = 'Recording...';
                        isRecording = true;
                };
                recognition.onend = () => {
                        recordButton.textContent = 'Start Recording';
                        isRecording = false;
                };
                recognition.onresult = (event) => {
                        const result = event.results[event.results.length - 1];
                        const text = result[0].transcript;
                        output.textContent = text;
                        // Send the text to the server for comparison
                        console.log(og);
                        $.ajax({
                                type: 'POST',
                                url: '/compare',
                                data: { text: text, og: og, contentId : contentId, contentPos :
contentPos, contentName: contentName},
                                headers: {
        'X-CSRFToken': csrfToken, // Include the CSRF token in the headers
},
success: function (response) {
        if (response.hasOwnProperty('result') && response.hasOwnProperty('accuracy')) {
                const resultArray = response.result;
        const accuracy = response.accuracy;
```

```
const totalScore = response.score;

document.getElementById('accuracy').textContent = accuracy;

document.getElementById('totalScore').textContent = totalScore;

console.log(response.score);

var resultText = '';

resultArray.forEach(function (wordStatus) {

        var word = wordStatus.word;

        var status = wordStatus.status;

        var cssClass = '';

        if (status === 'correct') {

                cssClass = 'correct';

        } else if (status === 'wrong') {

                cssClass = 'wrong';

        } else if (status === 'missing') {

                cssClass = 'missing';

        }

        resultText += '<span class="' + cssClass + '">' + word + '</span> ';

        document.getElementById('bottom-right').style.display = 'block';

});

document.getElementById('resultText').innerHTML = resultText;

                                }

                        },

                        error: function (xhr, status, error) {

                                console.error('Error:', error);

                        }

                });

        };

        recognition.start();

        const visualizationCanvas = document.getElementById('visualization');

        const visualizationContext = visualizationCanvas.getContext('2d');

        const audioContext = new (window.AudioContext ||
window.webkitAudioContext)();
```

```
const analyser = audioContext.createAnalyser();
analyser.fftSize = 8192;
const bufferLength = analyser.frequencyBinCount;
const dataArray = new Uint8Array(bufferLength);
navigator.mediaDevices
    .getUserMedia({ audio: true })
    .then((stream) => {
        const audioSource =
audioContext.createMediaStreamSource(stream);
        audioSource.connect(analyser);
        function drawVisualization() {
            analyser.getByteFrequencyData(dataArray);

            visualizationContext.clearRect(
                0, 0,
                visualizationCanvas.width,
                visualizationCanvas.height
            );

            visualizationContext.fillStyle = 'rgb(0, 0, 0)';
            const barWidth = (visualizationCanvas.width / bufferLength)
* 2.5;

            let x = 0;
            for (let i = 0; i < bufferLength; i++) {
                const barHeight = dataArray[i];
                const barColor = getRainbowColor(i,
dataArray.length);
                visualizationContext.fillStyle = barColor;
                    visualizationContext.fillRect(
                        /*x*/ i * barWidth
                        visualizationCanvas.height - barHeight,
                        barWidth,
```
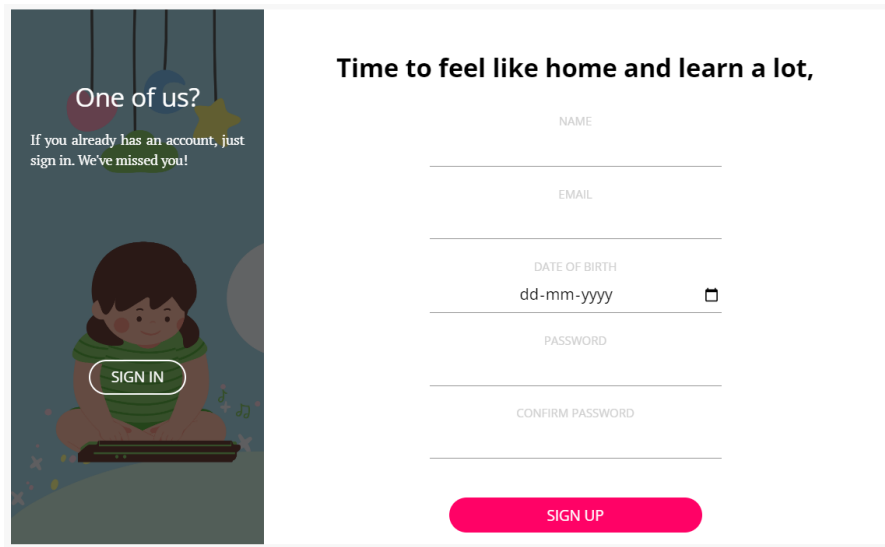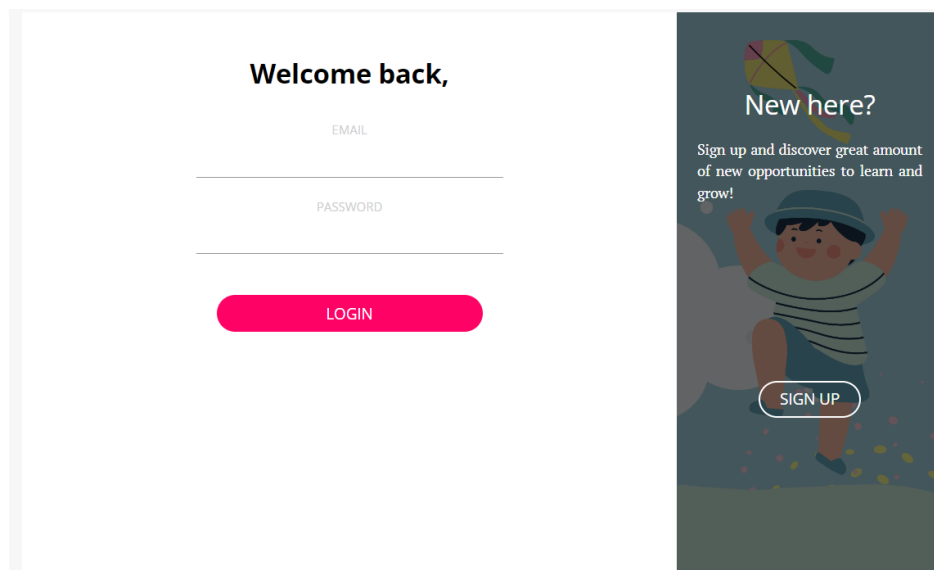
```
                                        barHeight
                                    );
                                    x += barWidth + 1;
                                }
                                if (isRecording) {
                                    animationId =
requestAnimationFrame(drawVisualization);
                                }
                            }
                            drawVisualization();
                    })
                    .catch((error) => {
                            console.error('Error accessing microphone:', error);
                    });
        } else {
                recognition.stop();
                cancelAnimationFrame(animationId);
                recordButton.textContent = 'Start Recording';
                output.textContent = '';
                isRecording = false;
        }
});
stopButton.addEventListener('click', () => {
        // Stop both recognition and visualization
        recognition.stop();
        cancelAnimationFrame(animationId);
        recordButton.textContent = 'Start Recording';
        output.textContent = '';
        isRecording = false;
});
</script>
{% endblock %}
```

4.4 OUTPUT SCREENS

Fig 4.4.1: Sign-Up page of the website

Fig 4.4.2: Login page of the website

Fig 4.4.3: Home Page of the website
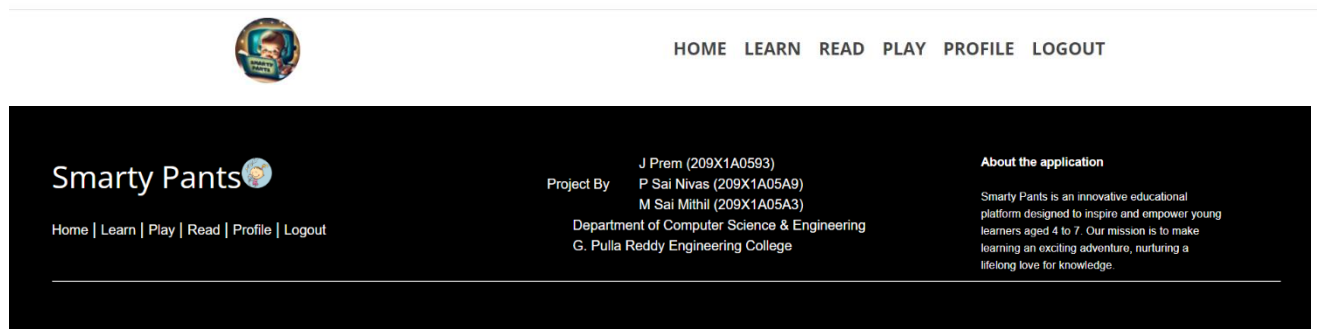
Fig 4.4.4: Navigation bar and footer of the website
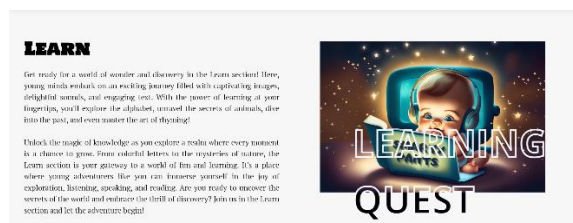
Fig 4.4.5: Home page – Learn Section
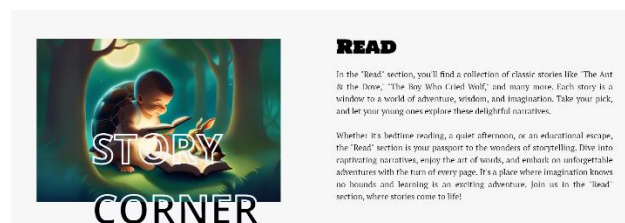
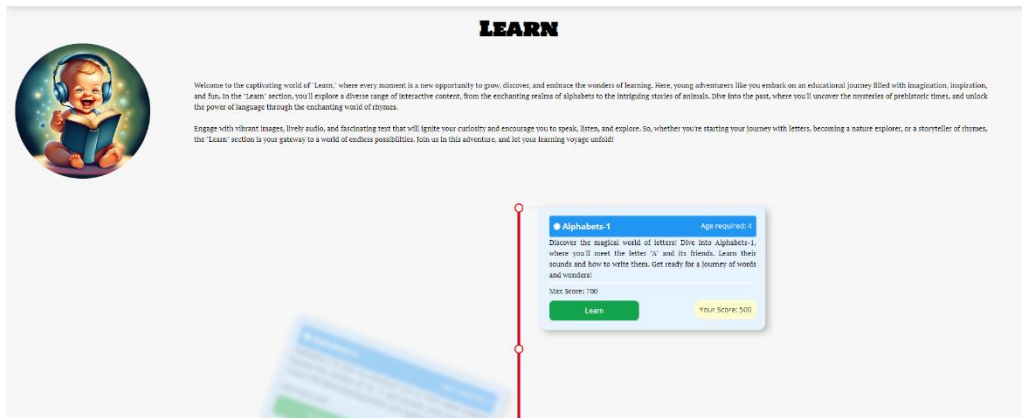Fig 4.4.6: Home page – Read Section

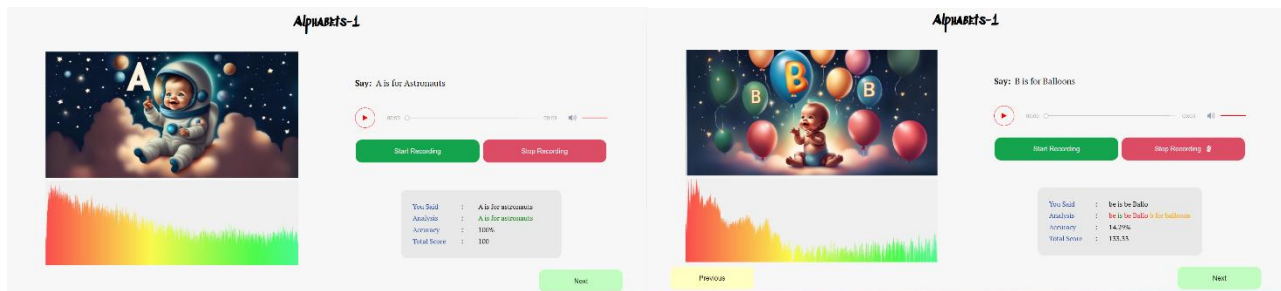Fig 4.4.7: Learn Page of the website

Fig 4.4.8: Learn page(a)   Fig 4.4.9: Learn page(b)
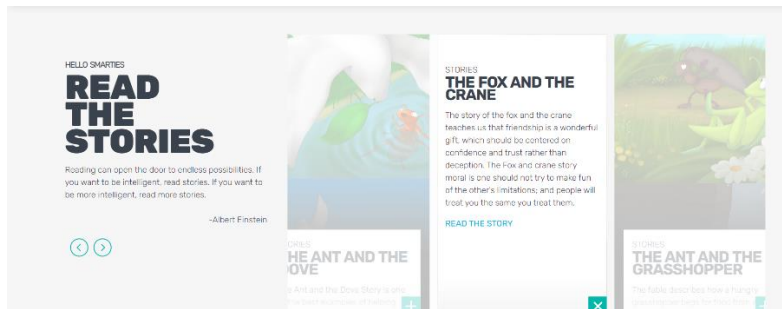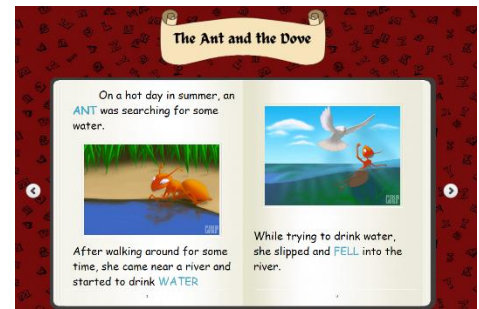

Fig 4.4.10: Results page
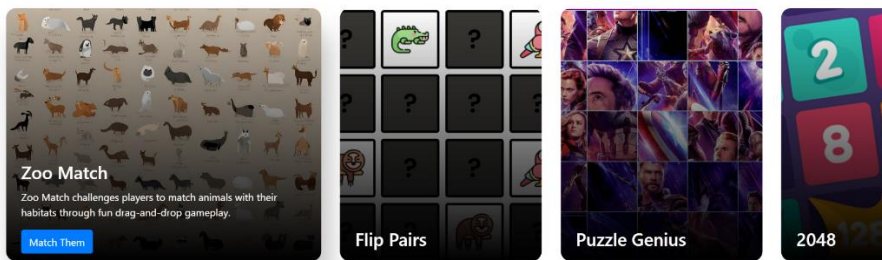
Fig 4.4.11: Read page

Fig 4.4.12: Story page

Fig 4.4.13: Play page – Education Games

Fig 4.4.14: Game page
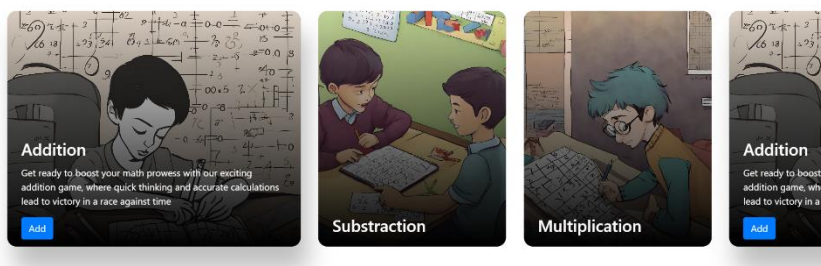
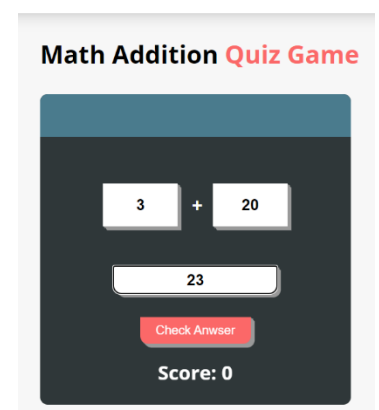Fig 4.4.15: Play page – Maths Games
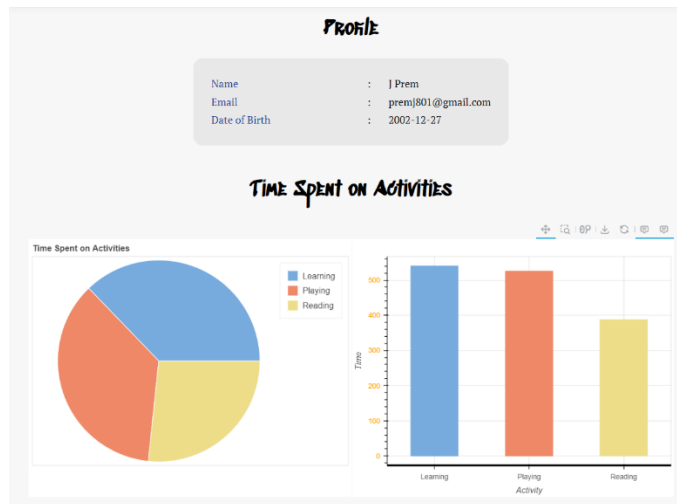

Fig 4.4.16: Addition page

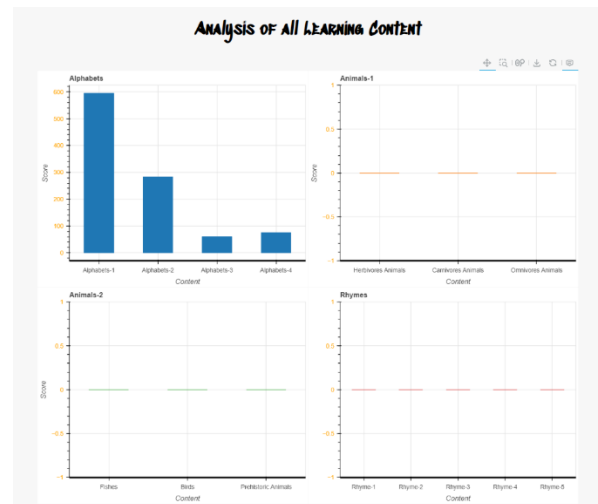Fig 4.4.17: Profile page(a)   Fig 4.4.18: Profile page(b)

4.5 TESTING AND VALIDATION

Effective testing and validation processes are pivotal to ensuring that "Smarty Pants" delivers a seamless and high-quality learning experience to its young users. The platform undergoes rigorous testing at various stages of development to identify and address any issues, guaranteeing its reliability and educational efficacy.

Levels of testing

Testing can be done at different levels of SDLC. They are

User Testing

The heart of testing for "Smarty Pants" lies in user testing, where children within the target age group (4 to 7 years old) actively engage with the application. This phase involves gathering real-world feedback by observing how young learners interact with the platform. It allows us to assess the user interface's intuitiveness, content engagement, and effectiveness of the speech recognition system. User testing provides insights into whether the learning content aligns with educational standards and is age-appropriate, ensuring that "Smarty Pants" delivers a valuable and engaging educational experience.

Functional Testing

Functional testing evaluates each feature and functionality of the application to ensure it operates as intended. This includes rigorous testing of speech recognition, progress tracking, level unlocking, and content presentation. Any issues or discrepancies are identified and addressed to guarantee the accuracy and effectiveness of the platform.

Performance Testing

Performance testing assesses the application's speed, responsiveness, and its ability to handle user loads. It ensures that "Smarty Pants" delivers a smooth and uninterrupted learning experience, regardless of the number of concurrent users. This type of testing guarantees that young learners can engage with the platform without any performance-related interruptions.

Security and Privacy Testing

Data privacy and security are paramount in a platform designed for children. Security and privacy testing is conducted to identify vulnerabilities and ensure that user data is safeguarded according to the highest standards. This includes extensive testing of user authentication and data encryption to protect children's personal information.

Cross-Platform Compatibility

Given that "Smarty Pants" is designed for iOS, Android, and web browsers, cross-platform compatibility testing is crucial. It ensures that the application operates consistently and optimally across a variety of devices and operating systems, providing a uniform learning experience for all users.

Validation and testing are ongoing processes in the development of "Smarty Pants," ensuring that the platform evolves to meet the needs of its young users effectively. By incorporating real-world user feedback, addressing functional and performance issues, and prioritizing security and data privacy, "Smarty Pants" aims to deliver a top-tier learning experience that enhances language skills and promotes educational excellence for children.

5. CONCLUSION

5. CONCLUSION

In the journey to create "Smarty Pants," a dynamic and engaging educational platform for children aged 4 to 7, we have successfully combined cutting-edge technology with a child-centric approach to foster language development, critical thinking, and problem-solving skills. The application's integration of speech recognition technology, in-depth content analysis, and interactive learning elements has the potential to transform the way young learners engage with educational content. Through extensive testing, rigorous validation, and responsive design, "Smarty Pants" has been tailored to meet the specific needs of its users. It strives to not only provide an effective and safe learning environment but also to be a source of inspiration and curiosity for young minds. By addressing the limitations of existing educational applications, "Smarty Pants" aims to set new standards in child-friendly, educational technology, making learning a joyous and enriching experience. This project underscores our commitment to advancing education in a digital age while ensuring that the next generation is equipped with the tools and knowledge to thrive in the future.

Future Enhancements

SMARTY PANTS can further be improved in many ways that benefit the children and some of the features are as follows:

- Expand the application's language options to include a wider array of languages, making it accessible to children from diverse linguistic backgrounds.
- Create a dedicated portal for parents and teachers to monitor a child's progress, providing valuable insights into their learning journey.
- Enhance the data analysis and reporting features to provide more in-depth insights into a child's learning patterns, strengths, and areas for improvement.
- Enable collaborative learning by allowing children to engage with their peers on educational projects and challenges, promoting social interaction and cooperative learning.
- Implement features for children with special needs, such as audio descriptions, larger text, and additional visual aids, to make the platform inclusive.
- Incorporate adaptive learning algorithms that tailor content to individual learning styles and paces, ensuring personalized and efficient learning experiences.
- Collaborate with educational institutions, content creators, and experts to continually enhance the quality and diversity of educational materials.

REFERENCES

1. https://flask.palletsprojects.com/en/3.0.x/
2. https://pypi.org/project/Flask-WTF/
3. https://pypi.org/project/Flask-Login/
4. https://pypi.org/project/SpeechRecognition/
5. https://docs.python.org/3/library/datetime.html
6. https://www.sqlite.org/index.html
7. https://sqlitebrowser.org/
8. https://bokeh.org/
9. https://www.geeksforgeeks.org/
10. https://codepen.io/
11. https://pbskids.org/
12. https://www.starfall.com/h/
13. https://www.funbrain.com/
14. https://englishtest.duolingo.com/

Textbooks

1. Flask Web Development 2nd Edition by Miguel Grinberg