RV UNIVERSITY

Go, change the world

an initiative of RV EDUCATIONAL INSTITUTIONS

Interdisciplinary Minor Project | Semester: 6

SIREN – An application to tackle online predation

Names of the authors:

1. Surabhi M Singh (SOLAS)
2. Arushi Basal (SDI)
3. Niral Koul (SDI)
4. Meda Siva Vardhan (SOB)
5. Shriya Prithvi (SOLAS)

Associated Faculty Members:

1. Dr Manjul Krishna Gupta
2. Dr Veena A
3. Dr Parth Krishnatre

Associated students from SOCSE:

1. Karmishtha Patnaik
2. Renu Bojja
3. Uzair Sunkad

Date of submission: 22/05/2024

CERTIFICATE

This is to certify that this Minor Project Report titled as "Siren: An application to tackle online predation" submitted by Surabhi M Singh (1RVU21BLA026), Arushi Basal (1RVU21BDE013), Niral Koul (1RVU21BDE040), Meda Siva Vardhan (1RVU21BBA024) and Shriya Prithvi (1RVU21BLA021) to The School of Computer Science Engineering, RV University, Bengaluru, in partial fulfilment of the requirements for their Bachelor's degree in their respective majors, is a bonafide record of work carried out by his/her under my supervision. The content of this report, in full or in part have not submitted in any form to any other institute or university for the award of any degree or diploma.

SUPERVISOR NAME:

DATE: 22.05.2024

DECLARATION

I, "Surabhi M Singh/Arushi Basal/Meda Siva Vardhan/Niral Koul/Shriya Prithvi" a bonafide student of RV University, Bengaluru, hereby declare that, the Minor Project Report titled as "SIREN: An App to Tackle Online Predation" submitted in partial fulfilment of the requirements for the degree of Bachelor's in my respective Major, from my respective school in R.V. University. The information presented in this project is original work and does not form any part of the project undertaken previously.

DATE: 22.05.2024 **PLACE:** RV UNIVERSITY, BENGALURU-560059

NAME & SIGNATURE OF STUDENT:

ABSTRACT:

The "SIREN" app is a pioneering solution designed to safeguard teenagers in their online interactions, ensuring a secure and positive social media experience. The app employs advanced technologies such as machine learning and natural language processing to proactively identify potential predatory behavior. It analyses linguistic patterns, behavior shifts, sentiment, profile, and media content to detect threats. The real-time alert system driven by AI sentiment analysis notifies teenagers and involves designated guardians, enhancing the ability to address potential threats promptly.

Key objectives of the project include the early detection of predatory behavior, the establishment of a proactive alert system, and the empowerment of teenagers through educational resources on online safety, healthy relationships, and manipulation recognition. The app offers customizable alert thresholds and robust privacy protection, ensuring tailored safety settings for users. Continuous improvement through machine learning updates and user feedback mechanisms, as well as potential community support features, aims to enhance user experience and safety.

Key findings underscore the urgency of addressing online predation, with significant statistics revealing the widespread risks faced by teenagers. The project recognizes the complex digital landscape and aims to provide a crucial layer of defence, equipping teenagers with tools and knowledge to navigate the online world safely. The integration of educational content, customizable settings, and privacy measures ensures a comprehensive approach to online safety.

TABLE OF CONTENTS:

CHAPTER NO.		PARTICULARS	PAGE NO.
		ABSTRACT	i
1.		**INTRODUCTION** • Background information about the project • Objectives and goals of the project • Scope and limitations	
2.		**LITURATURE REVIEW**	
3.		**Architecture, Design, Methodology:** • Description of the software development methodology used (e.g., Agile, Waterfall) • Explanation of the tools, languages, and technologies used • Overview of the project's architecture and design	
4.		**Implementation:** • Detailed explanation of how the project was implemented • Code snippets, algorithms, and data structures used • Challenges faced during implementation and how they were overcome	
5.		**Test Results and Discussion:** • Presentation of the results achieved feature wise • Comparison of the results with the project objectives • Discussion of any unexpected findings or issues encountered	
6.		**Conclusion:** • Summary of the project and its outcomes • Achievements and contributions of the project • Suggestions for future work or improvements	
		REFERENCES ***APPENDICES***	

CHAPTER – 01

INTRODUCTION:

The “SIREN” (notification sound) app stands as a pioneering solution dedicated to safeguarding teenagers in their online interactions, fostering a secure & positive social media experience. By employing advanced technologies like machine learning & natural language processing, the app proactively identifies potential predatory behaviour through the analysis of linguistic patterns, behaviour shifts, sentiment analysis & profile & media content. The real-time alert system, driven by AI sentiment analysis, not only promptly notifies teenagers but also involves designated guardians in addressing potential threats. Crucially, SIREN goes beyond mere detection by empowering teenagers through an educational platform covering online safety, healthy relationships, & manipulation recognition. The customizable alert thresholds & robust privacy protection mechanisms ensure tailored safety settings, granting users control over their online environment. As the app continuously evolves through machine learning improvements, user feedback mechanisms, & potential community support features, SIREN seeks to create a digital space where teenagers can engage safely & confidently through social media, enhancing their overall well-being in the digital age.

The Relevance of the identified problem: Online Predators

Addressing online predation is of paramount importance as it poses a severe threat to the well-being of teenagers worldwide. The Indian Society of Ergonomics & Association (ISEA) defines online predators as internet users who exploit children & teens for sexual & violent purposes, encompassing activities such as child grooming, engaging in sexual acts, unwanted exposure of materials, online harassment, & threats. Disturbingly, a study by the National Crime Records Bureau (NCRB) in India uncovered 24,212 cases of crimes against children & teenagers in 2019, including instances of online predation, highlighting the urgent need for early detection & prevention. The urgency surrounding online predation is further emphasized by statistics from the Pew Research Centre, revealing that 95% of teenagers in the United States have access to smartphones, with 45% claiming to be online "almost constantly." This extensive online presence exposes teenagers to various risks, with 59% having experienced some form of cyberbullying, as reported by the Cyberbullying Research Centre. The National Centre for Missing & Exploited Children (NCMEC) reported a staggering 97% increase in online child exploitation reports in 2020, underscoring the pervasive nature of threats, including grooming, exploitation, & manipulation. The identified problem revolves around the heightened vulnerability of teenagers

to a spectrum of online threats & predatory behaviour. The internet, while integral to teenagers' lives for education, entertainment, socialization, & self-expression, exposes them to dangers that can have severe consequences on their physical, mental, & emotional well-being, as well as future prospects. The multifaceted risks include cyberbullying, online predation, exposure to inappropriate content, identity theft, & online scams. This heightened vulnerability is exacerbated by the evolving nature of digital communication & teenagers' often limited awareness of online risks. The project, spearheaded by an undergraduate tool developer, acknowledges the pressing need to address & mitigate these threats comprehensively. By combining advanced technologies like machine learning & artificial intelligence with educational resources, the project aims to empower teenagers & enhance their online safety. This interdisciplinary approach recognizes the complexity of the digital landscape, where fake profiles & deceptive practices further complicate the challenges faced by adolescents. In essence, the project seeks to provide a crucial layer of defence, proactively identifying potential predatory behaviour & equipping teenagers with the tools & knowledge to navigate the complexities of the digital world safely.

OBJECTIVES:

1. Early Detection of Predatory Behaviour:

• Goal: Develop a tool that can analyse user profiles & text interactions to identify early

signs of predatory behaviour.

• Functionality:

- ❖ Utilize machine learning models & natural language processing algorithms to analyse linguistic patterns, behaviour shifts, & context, enabling the system to identify potential threats.
- ❖ Leverage image recognition algorithms & AI content analysis to scrutinize profile pictures & posted content for inconsistencies or signs of stock photos, which are common in fake profiles. This functionality enables the system to identify anomalies or reused content associated with fake profiles.

2. Proactive Alert System:

• Goal: Establish a real-time alert system to notify teenagers and, when appropriate,

designated guardians about potential predatory behaviour.

• Functionality: Implement real-time analysis of user interactions, leveraging AI for sentiment analysis & machine learning to recognize patterns associated with predatory behaviour, triggering alerts when necessary.

3. User Empowerment Through Education:

• Goal: Empower teenagers with knowledge & resources to recognize & respond to potential threats independently.

• Functionality: Provide educational content, including articles, videos, & interactive modules, on topics such as online safety, healthy relationships, & recognizing manipulation.

4. Customizable Alert Thresholds:

• Goal: Allow users & guardians to customize alert thresholds & safety settings based on individual preferences.

• Functionality: Integrate a user-friendly interface that enables customization of safety features, providing flexibility to match the comfort levels of individual users & their guardians by implementing machine learning models that consider user feedback & preferences.

5. Privacy Protection:

• Goal: Ensure the privacy & confidentiality of user data & interactions within the platform.

• Functionality: Implement robust security measures, including encryption, to safeguard user information & foster a secure online environment.

6. Continuous Machine Learning Improvement: (Future)

• Goal: Regularly update & improve the ML models based on new data & insights.

• Functionality: Establish a system for regular updates, incorporating new data & insights from ongoing research in psychology & online behaviour.

7. User Feedback Mechanism: (Future)

• Goal: Gather feedback from users to improve the accuracy of alerts & the overall user experience.

• Functionality: Implement features that allow users to provide feedback on the effectiveness of alerts & report any false positives or negatives.

8. Community Support & Resources: (Optional)

• Goal: Foster a supportive community within the platform where users can share experiences, advice, & support.

• Functionality: Integrate community features, such as forums or discussion boards, & provide access to mental health resources & helplines.

SCOPE AND LIMITATIONS

Scope:

The SIREN project aims to significantly enhance the safety of teenagers in their online interactions through a multifaceted approach. The primary scope includes the detection of predatory behavior using advanced machine learning and natural language processing techniques to analyze text interactions and user profiles. The project also incorporates image recognition algorithms to scrutinize profile pictures and posted content for inconsistencies that may indicate fake profiles.

A core feature of SIREN is its real-time alert system. This system utilizes AI-driven sentiment analysis to detect potential threats and provides immediate notifications to both teenagers and their designated guardians. This proactive alert mechanism ensures timely intervention, potentially mitigating the risk of harm.

In addition to detection and alerts, SIREN focuses on educational empowerment. The app offers a comprehensive educational platform with resources on online safety, healthy relationships, and manipulation recognition. This empowers teenagers with the knowledge and skills needed to navigate the digital world safely.

The app also provides customizable safety settings, allowing users to tailor alert thresholds and privacy settings according to their comfort levels. This ensures that the safety measures are adaptable to individual needs and preferences. Privacy protection is another critical aspect of SIREN, with robust security measures, including encryption, to safeguard user data.

SIREN is designed for continuous improvement, with regular updates to its machine learning models based on new data and user feedback. Future enhancements include offline mode, anonymized reporting, emergency response integration, access to mental health resources, and language translation support, further expanding the app's capabilities and reach.

Limitations:

Despite its comprehensive approach, the SIREN project faces several limitations. Technological constraints pose a significant challenge, as machine learning and natural language processing algorithms are not infallible. These technologies can produce false positives or negatives in threat detection, and while continuous improvement is planned, initial versions may lack the necessary sophistication for perfect accuracy.

User privacy and data security, although prioritized, are areas of potential vulnerability. Ensuring complete anonymity and security in reporting systems is complex, and the risk of data breaches or misuse cannot be entirely eliminated. Furthermore, the effectiveness of the educational component depends heavily on user engagement. Teenagers may not consistently utilize the resources or adhere to the guidelines provided by the app.

Customization and usability present additional challenges. The customizable alert thresholds and privacy settings require users to understand and correctly configure these options, which might be difficult for some. Balancing user control with effective protection is a delicate task that demands careful design considerations.

Cultural and linguistic barriers also impact the app's effectiveness. Language translation and cultural nuances can affect the accuracy and relevance of threat detection and educational content. Initially focusing on specific languages and regions may limit SIREN's global effectiveness.

The app’s dependency on technology is another limitation. Its functionality relies heavily on internet connectivity and access to smartphones, potentially excluding users with limited

technological access. The offline mode functionalities are inherently limited and cannot offer real-time threat detection.

Finally, addressing ethical and legal challenges is crucial. Monitoring and data collection raise ethical concerns, particularly across different legal jurisdictions. Collaborating with local authorities and emergency services can be difficult due to varying legal frameworks and response protocols.

CHAPTER – 02

LITERATURE REVIEW: Linguistic Patterns in Chats with Online Predators through the lens of Forensic Psychology

Introduction:

The proliferation of internet communication has provided online predators with unprecedented access to potential victims, primarily minors. Research into the linguistic patterns used by these predators has become critical in developing tools for early detection and prevention. This literature review synthesizes findings from various studies on the linguistic characteristics of predatory communication, highlighting both rule-based and machine learning approaches to identifying grooming behaviors.

Rule-Based Approaches to Identifying Predatory Communication:

One prominent study by McGhee et al. (2011) introduced a rule-based system called ChatCoder 2, designed to classify lines of chat logs into specific categories indicative of predatory behavior. ChatCoder 2 uses predefined linguistic rules to detect:

- Exchange of Personal Information (Code 200): Lines containing personal details, relationship information, or discussions about activities and hobbies.

- Grooming (Code 600): Lines with vulgar sexual language, discussions of sexual acts, or attempts to normalize sexual behavior.

- Approach (Code 900): Lines where the predator attempts to arrange meetings, obtain contact information, or isolate the victim from their support system.

The researchers compared ChatCoder 2 with machine learning algorithms like decision trees and instance-based learning. Although machine learning methods occasionally improved accuracy for individual transcripts, the rule-based system performed comparably well overall, achieving an average accuracy of 68.11%. The study concluded that manually developed rules were adequate for identifying key patterns of predatory communication【McGhee et al., 2011】.

Machine Learning and Linguistic Analysis

Black et al. (2015) conducted a linguistic analysis of grooming strategies used by online child sex offenders, examining the language of 44 convicted offenders. The study aimed to validate the five-stage model of online grooming proposed by O'Connell (2003). Key findings included:

- Risk Assessment: Predators often initiate conversations by assessing the risk of being caught, asking about the victim's location and parental supervision early on.

- Exclusivity and Sexual Content: Unlike offline grooming, sexual content is introduced relatively early in online interactions. Predators use language that builds exclusivity and trust, frequently interspersed with sexual language【Black et al., 2015】.

Stages of Grooming and Linguistic Features

Cano et al. (year) expanded on this by proposing a method to automatically identify grooming stages using various linguistic features. The stages include:

- Trust Development: Predators exchange personal information to build common ground with the victim.
- Grooming: Predators trigger the victim's sexual curiosity, using sexual terms to entrap the child.
- Approach: Predators seek to meet the victim physically, requesting details about schedules and locations.

The study utilized lexical, syntactical, sentiment, content, psycho-linguistic, and discourse patterns to classify these stages. The results indicated that combining these features improved precision in detecting grooming stages compared to using lexical features alone【Cano et al.】.

Function Words and Pronoun Usage

Research has also highlighted the importance of function words in analyzing predator language. Function words, such as articles, prepositions, and conjunctions, reflect psychological mechanisms that are difficult to control consciously. Two hypotheses explain the distinct pronoun usage by predators:

1. Psychological Distancing: Predators use fewer first-person pronouns to distance themselves from their actions.
2. Social Dominance: Predators use more second-person pronouns to establish dominance over their targets【Baryshevtsev】.

Heuristic and Learning-Based Detection

Gómez Hidalgo and Caurcel Díaz (2012) described a hybrid system combining a knowledge-based conversation filter with a learning-based detection sub-system. The conversation filter identifies suspicious behaviors using hand-coded patterns, while the learning-based system classifies speakers based on linguistic and chat-like features. Although the system showed promise, its performance on test data highlighted the need for improved pattern translation and inclusion of additional language-dependent techniques【Gómez Hidalgo & Caurcel Díaz, 2012】

Conclusion

The linguistic analysis of predatory communication has significantly advanced through both rule-based and machine learning approaches. Rule-based systems like ChatCoder 2 offer simplicity and reasonable accuracy, while machine learning methods provide nuanced insights into the linguistic features of grooming behaviors. Future research should continue refining these methods, focusing on the integration of complex linguistic features and improving the adaptability of detection systems across different languages and contexts.

CHAPTER – 03

ARCHITECTURE, DESIGN, METHODOLOGY:

Software Development Methodology:

The development of the SIREN app follows the Agile methodology, which emphasizes iterative progress, collaboration, and flexibility. Agile allows for continuous feedback and iterative improvements, ensuring the app evolves in response to user needs and technological advancements. This approach enables the development team to prioritize features, address issues promptly, and incorporate user feedback into subsequent iterations. Regular sprint cycles and review meetings ensure that the project remains on track and aligned with its goals, providing a dynamic framework for managing the complexities of developing a safety-focused application.

Tools, Languages, and Technologies:

The SIREN app leverages a range of modern tools and technologies to achieve its objectives. The frontend development is primarily carried out using Flutter, a versatile framework for building natively compiled applications from a single codebase. Dart programming language, the backbone of Flutter, is used to write the code, ensuring high performance and smooth user experiences. For integrated development, Visual Studio Code (VS Code) IDE is employed, providing a robust environment for coding, debugging, and testing. Machine learning capabilities are a critical component of SIREN, particularly for sentiment analysis and predatory behavior detection. TensorFlow Lite is used to deploy machine learning models on mobile devices, ensuring efficient performance and low latency. Specifically, Support Vector Machine (SVM) algorithms are utilized for sentiment analysis, implemented on Google Collab to harness its powerful computational resources and collaborative features.

Design aspects of the app are meticulously crafted using Figma, a powerful tool for interface design and prototyping. Figma enables the team to create intuitive and visually appealing designs, ensuring a user-friendly experience.

Additionally, technologies like ChatGPT, Gemini, and Copilot provide valuable suggestions and guidance throughout the development process, enhancing the team's productivity and problem-solving capabilities.

Project Architecture and Design:

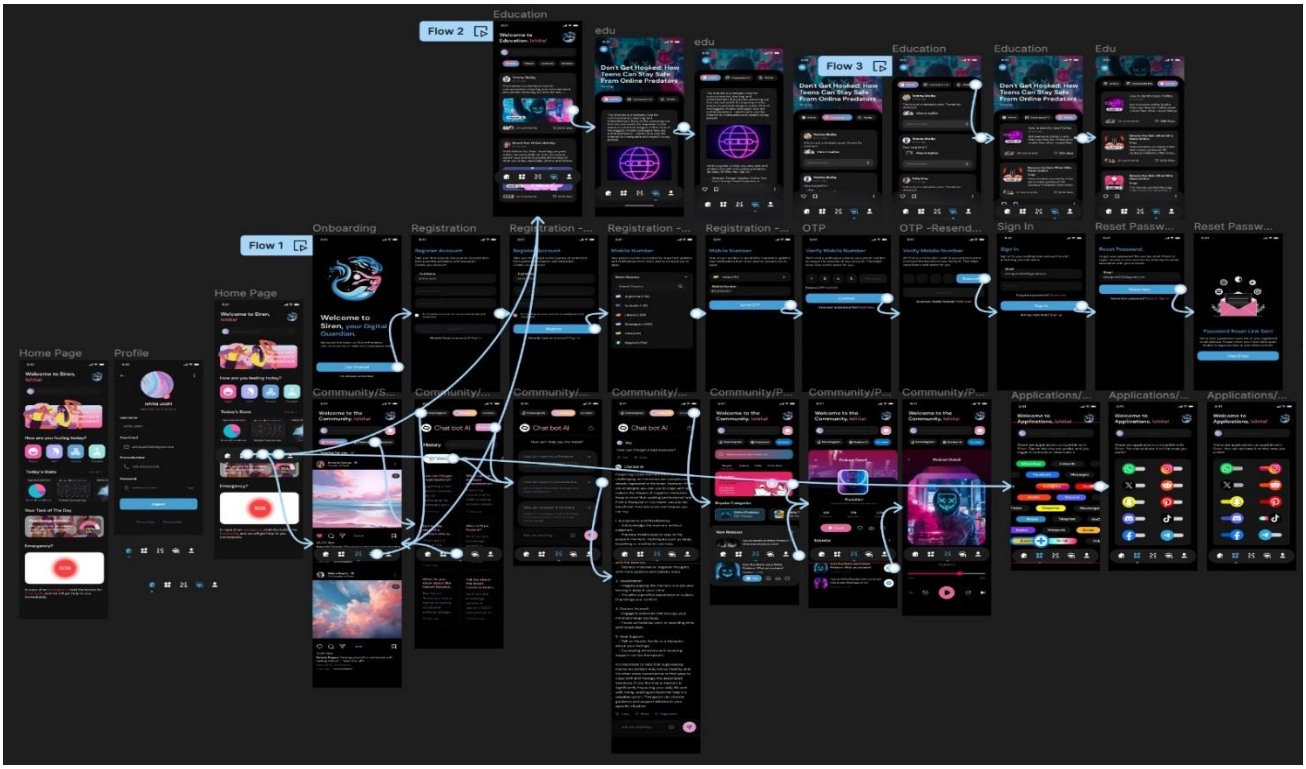
The architecture of the SIREN app is designed to be modular and scalable, ensuring it can adapt to future enhancements and evolving user needs. The frontend, developed in Flutter, interacts seamlessly with various APIs and dependencies, facilitating real-time data processing and interaction.

The backend architecture integrates TensorFlow Lite models for machine learning tasks, ensuring efficient on-device processing for real-time threat detection. The app employs a client-server model, where the client-side (user's device) handles user interactions, data collection, and initial processing, while the server-side manages more intensive computations, data storage, and continuous model training and updates.

Design: User-Friendly Interface for Siren

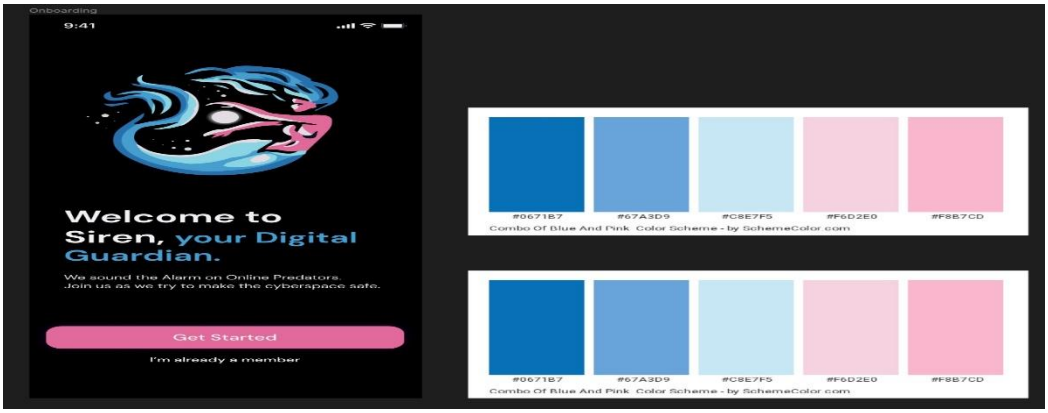
The design phase of Siren focused on crafting a user-friendly interface that prioritizes both aesthetics and functionality. We utilized Figma, a popular design software, throughout the process. Figma's robust features were instrumental in various stages, including:

Wireframing: We began by creating low-fidelity wireframes to establish the core structure and layout of the app. This stage focused on user flows and information hierarchy, ensuring a clear and intuitive navigation experience.



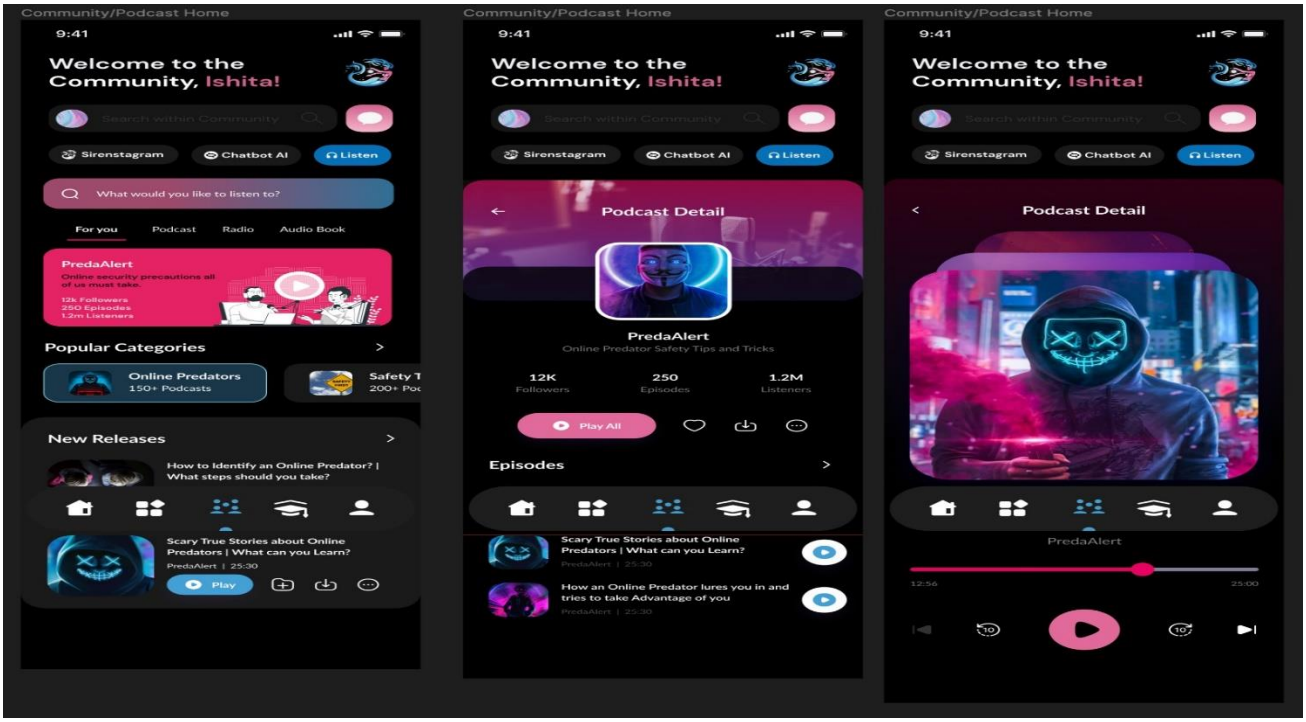
Rapid Prototyping: To test user interaction and refine the layout, we created interactive prototypes using Figma's prototyping tools. This allowed us to simulate the app's functionality and gather valuable feedback from potential users in an early stage.

Color Scheme Selection: A crucial aspect of the design was the selection of a color palette that reflected Siren's core message – safety and empowerment. We explored various color combinations, prioritizing a calming yet confident feel. The final color scheme strikes a balance between professionalism and approachability, aligning with the target audience of teenagers.



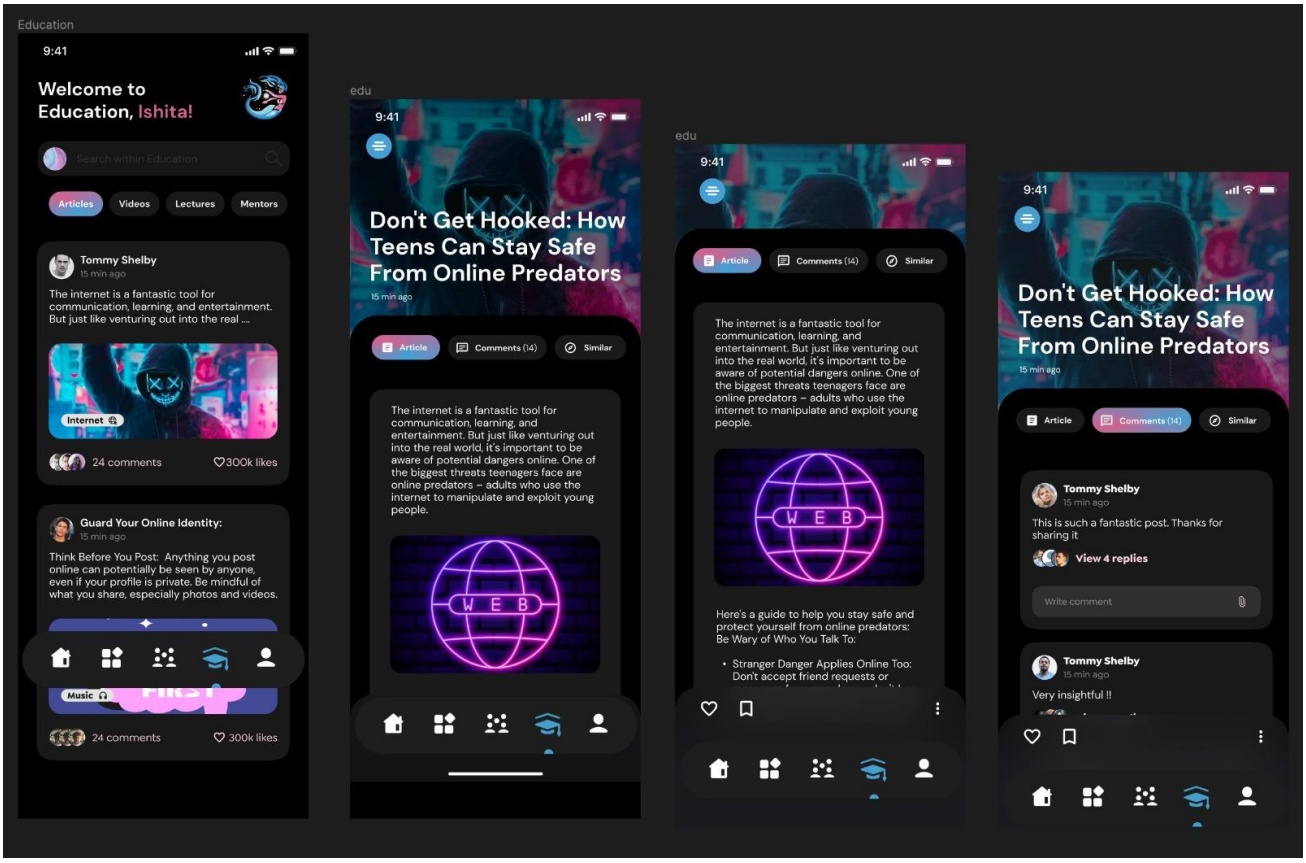
High-Fidelity Mock-ups: Following the initial design iterations and user feedback, we created high-fidelity mock-ups in Figma. These mock-ups showcased the final visual design,

incorporating detailed icons, typography, and imagery that embodies the "Siren" theme while maintaining simplicity and ease of use.



Throughout the design process, we maintained a user-centric approach. Simplifying the user interface was a key priority, ensuring teens of all technical skill levels can navigate the app with ease. Additionally, we incorporated subtle visual cues that align with the "Siren" theme, creating a brand identity that resonates with the app's functionality. The final design delivers a visually pleasing and intuitive interface that empowers users to navigate online interactions safely and confidently.

Security measures are deeply embedded in the architecture, with encryption protocols ensuring the confidentiality and integrity of user data. The system also incorporates mechanisms for anonymized reporting and emergency response integration, enhancing user trust and safety.


Education
9:41
Welcome to Education, Ishita!
Search within Education
Articles
Videos
Lectures
Mentors
Tommy Shelby
15 min ago
The internet is a fantastic tool for communication, learning, and entertainment. But just like venturing out into the real ...
Internet
24 comments
300k likes
Guard Your Online Identity:
15 min ago
Think Before You Post: Anything you post online can potentially be seen by anyone, even if your profile is private. Be mindful of what you share, especially photos and videos.
Music
24 comments
300k likes
edu
9:41
Don't Get Hooked: How Teens Can Stay Safe From Online Predators
15 min ago
Article
Comments (14)
Similar
The internet is a fantastic tool for communication, learning, and entertainment. But just like venturing out into the real world, it's important to be aware of potential dangers online. One of the biggest threats teenagers face are online predators – adults who use the internet to manipulate and exploit young people.
WEB
edu
9:41
Article
Comments (14)
Similar
The internet is a fantastic tool for communication, learning, and entertainment. But just like venturing out into the real world, it's important to be aware of potential dangers online. One of the biggest threats teenagers face are online predators – adults who use the internet to manipulate and exploit young people.
WEB
Here's a guide to help you stay safe and protect yourself from online predators:
Be Wary of Who You Talk To:
• Stranger Danger Applies Online Too:
Don't accept friend requests or
9:41
Don't Get Hooked: How Teens Can Stay Safe From Online Predators
15 min ago
Article
Comments (14)
Similar
Tommy Shelby
15 min ago
This is such a fantastic post. Thanks for sharing it
View 4 replies
Write comment
Tommy Shelby
15 min ago
Very insightful !!

CHAPTER – 04

IMPLEMENTATION:

Detailed Explanation of How the Project Was Implemented

The implementation of the SIREN app involved several stages, each addressing different aspects of development. The project began with detailed planning and requirement analysis, followed by the iterative development of the app's core functionalities using the Agile methodology. This approach allowed for flexibility and continuous improvement based on user feedback and testing results.

Frontend Development:

The frontend of the SIREN app was developed using Flutter, a powerful framework that allows for the creation of natively compiled applications for mobile, web, and desktop from a single codebase. Dart programming language was used to write the application logic, ensuring smooth and efficient performance across different platforms. The development environment was set up in Visual Studio Code (VS Code), which provided a robust IDE for coding, debugging, and testing.

Backend Development and Machine Learning Integration:

For the backend, TensorFlow Lite was integrated to deploy machine learning models directly on mobile devices. This choice was driven by the need for real-time analysis and low latency in detecting predatory behavior and performing sentiment analysis. The Support Vector Machine (SVM) algorithm was employed for sentiment analysis tasks. The training and testing of the SVM models were conducted on Google Colab, leveraging its computational power and collaborative features.

Design and User Experience:

The user interface (UI) and user experience (UX) design were meticulously crafted using Figma. The design phase focused on creating an intuitive and visually appealing interface, ensuring that the educational content and customization options were easily accessible to users. Interactive prototypes were developed and tested to refine the user experience continuously.

Educational Content and Customization:

The app features a comprehensive educational platform, providing resources on online safety, healthy relationships, and manipulation recognition. This content was integrated into the app, allowing users to access valuable information and tools to protect themselves online. Customizable alert thresholds and privacy settings were implemented to give users control over their safety features, enhancing the app's adaptability to individual needs.

Challenges In Implementing the App:

Integration of Machine Learning Models:

One of the significant challenges was integrating TensorFlow Lite models into the mobile application. Ensuring that the models performed efficiently on mobile devices required extensive optimization. This was addressed by selecting lightweight models and conducting thorough testing to balance accuracy and performance.

Real-time Analysis and Alerts:

Implementing real-time analysis and alert systems posed a challenge due to the need for immediate processing and notification. This was overcome by optimizing the algorithms and leveraging the efficient processing capabilities of TensorFlow Lite. Additionally, careful design of the alert system ensured timely and accurate notifications without overwhelming the user.

Ensuring Data Privacy and Security:

Maintaining data privacy and security was a critical challenge, given the sensitive nature of the information handled by the app. Robust encryption methods and secure data storage practices were implemented to protect user data. Regular security audits and updates were conducted to identify and address potential vulnerabilities.

User Engagement and Education:

Encouraging user engagement with the educational content was challenging. To overcome this, the content was made interactive and easily accessible, with a focus on relevance and practical advice. Gamification elements and regular updates helped maintain user interest and engagement.

Code snippets, algorithms, and data structures used:

Data Parsing and Cleaning Process:

This code is designed to read and process a text file containing WhatsApp chat data, specifically to extract and structure the messages into a more analysable format. The script begins by reading the entire content of a specified file line-by-line into a list called `data`. It skips the first line, which typically contains metadata or an introductory message from WhatsApp. Each subsequent line is then parsed to extract the date, time, sender's name, and message content. These components are stripped of any extraneous characters and stored in a cleaned list of lists structure. This cleaned data is then converted into a pandas Data Frame with columns named 'Date', 'Time', 'Name', and 'Message'. Finally, the Data Frame is saved as a CSV file for further analysis or storage, ensuring the extracted data is formatted correctly and ready for use.

Data Filtering and Exploration Process:

This code snippet further explores and filters the previously parsed WhatsApp chat data. Initially, it checks the structure and basic information about the Data Frame df by displaying its shape (number of rows and columns), detailed info (including column data types and non-null counts), and the last few rows of the Data Frame. This helps in understanding the dataset's dimensions and content. Subsequently, a filter is applied to remove rows where the 'Message' column contains the placeholder '<Media omitted>', which typically indicates that media files were shared but are not included in the text data. The filtered Data Frame df1 contains only text messages, providing a cleaner dataset for further analysis.

Data Cleaning and Handling Missing Values:

In this part of the code, the focus is on cleaning the Data Frame df1 by handling missing values. The code replaces any empty string values in the Data Frame with NaN (Not a Number) using the replace method from pandas. This is important because empty strings can interfere with data analysis and it's more useful to explicitly mark these entries as missing values. After replacing empty strings with NaN, the code drops all rows that contain any NaN values using the dropna method with the parameter how='any'. This ensures that only complete cases (rows without any missing values) are retained in the Data Frame. This cleaned Data Frame is more reliable for subsequent analysis and operations.

```
import numpy as np
df1.replace('', np.nan, inplace=True)

# Drop rows where any column contains missing (NaN) values
df1 = df1.dropna(how='any')
```

Filtering Specific Messages:

In this part of the data cleaning process, additional filtering is performed to remove specific types of messages that are not useful for analysis. Specifically, the code filters out messages with the content 'Waiting for this message' and 'This message was deleted'. These messages typically indicate temporary issues with message delivery or that a user has deleted their message, and hence do not contribute meaningful content for analysis. By removing these entries, the dataset becomes more refined and focused on actual conversations.

```
df1 = df1[df1['Message'] != 'Waiting for this message']
df1
```

Removing Messages Consisting Only of Emojis:

In this part of the data cleaning process, the aim is to remove rows from the Data Frame where the 'Message' column consists entirely of emojis. Such messages, while part of normal chat behavior, often do not provide substantial textual content for analysis. The implementation involves defining a function that uses a regular expression to check if a message contains only emojis, and then applying this function to filter out these rows.

	Date	Time	Name	Message
3	17/08/22	15:56	+91 80505 49839	https://classroom.google.com/c/NTI2MzcwMjY1OTQ...
10	17/08/22	16:03	+91 80505 49839	https://classroom.google.com/c/NTI2MzcwMjY1OTQ...
16	17/08/22	16:08	+91 91102 50661	https://classroom.google.com/c/NTI2MzcwMjY1OTQ...
22	17/08/22	16:16	+91 80505 49839	https://classroom.google.com/c/NTI2MzcwMjY1OTQ...
28	17/08/22	17:51	Phani Sir	Boss, those who are doing structured innovatio...
...	...	...	...	...
14490	06/04/24	14:04	Mydhili Nair RVU	*IA-2*
14503	06/04/24	14:10	Mydhili Nair RVU	*Headsup*: The gaps in time when exams are not...
14508	06/04/24	14:11	Mydhili Nair RVU	Kindly ask your Profs for the portions for the...
14511	06/04/24	16:18	Karthi DMGT	Thanks for the schedule @919880092392 👍 👍
14518	06/04/24	17:27	Karthi DMGT	Sorry this wasn't meant for here.

3809 rows × 4 columns

Sentiment Analysis with Textblob:

In this part of the implementation, sentiment analysis is performed on the messages in the Data Frame using TextBlob. TextBlob is a Python library for processing textual data and provides simple APIs for diving into common natural language processing (NLP) tasks, such as part-of-

speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. The goal here is to categorize each message as having a positive, negative, or neutral sentiment.

```
from textblob import TextBlob

# Example DataFrame
# df = pd.DataFrame({'Message': ['This movie is great!', 'I don\'t like this product.', 'Amazing!', 'Terrible.']})

# Define a function to calculate sentiment polarity using TextBlob
def get_sentiment(text):
    blob = TextBlob(text)
    return 'positive' if blob.sentiment.polarity > 0 else 'negative' if blob.sentiment.polarity < 0 else 'neutral'

# Apply the function to the 'Message' column
df1['Sentiment'] = df1['Message'].apply(get_sentiment)
```

1. SVM:

Extracting Features and Labels:

The Message column is extracted and assigned to X, which will serve as the feature set containing the text data. The Sentiment column is extracted and assigned to y, which will serve as the target labels indicating the sentiment (positive, negative, or neutral) of each message.

Splitting Data into Training and Testing Sets:

To prepare the data for machine learning tasks, the dataset is divided into training and testing sets. This is accomplished using the train_test_split function from the sklearn.model_selection library. The training set, comprising 80% of the data, is used to train the machine learning model, while the remaining 20% forms the testing set, which is reserved for evaluating the model's performance on unseen data.

Converting Text Data into Numerical Features:

To enable machine learning algorithms to process text data, it must be converted into numerical features. This is achieved using the `TfidfVectorizer` from the `sklearn.feature_extraction.text` module. The `TfidfVectorizer` transforms the text data into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which reflect the importance of each word in a document relative to the entire dataset. The process begins by fitting the `TfidfVectorizer` to the training data (`X_train`), which involves learning the vocabulary and computing the IDF values. This results in the transformation of the training text data into a sparse matrix of TF-IDF features (`X_train_vectorized`). Subsequently, the same vectorizer is applied to the testing data (`X_test`)

to ensure consistency in feature representation, transforming it into `X_test_vectorized`. This step is crucial for preparing the text data for machine learning models, as it converts the unstructured text into structured numerical data that algorithms can effectively utilize for training and predictions.

```
# 3. Convert text data into numerical features
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

Defining and Training SVM Model:

In this step, a Support Vector Machine (SVM) model with a linear kernel is defined and trained using the transformed training data. The linear kernel signifies that the decision boundary between classes will be linear, suitable for scenarios where classes are separable by a straight line or hyperplane. The model is trained using the `fit` method, where it learns to identify the optimal decision boundary by maximizing the margin between classes while minimizing classification errors. By adjusting its parameters during training, such as regularization strength, the SVM aims to generalize well to unseen data, ensuring robust performance in sentiment classification tasks.

```
# 4. Define and train SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_vectorized, y_train)
```

```
SVC
SVC(kernel='linear')
```

Model Evaluation:

In this step, the trained SVM model is evaluated using the testing data. Predictions are generated for the sentiment labels of the testing set using the `predict` method of the SVM model. These predicted labels (`y_pred`) are compared with the actual labels (`y_test`) from the testing set. The `classification_report` function from the `sklearn.metrics` module is then used to generate a comprehensive report, providing key metrics such as precision, recall, F1-score, and support for each sentiment class. This report offers valuable insights into the model's performance, including its ability to correctly classify sentiments and its overall accuracy. By analyzing these metrics, the

effectiveness and reliability of the SVM model for sentiment classification tasks can be determined.

	precision	recall	f1-score	support
negative	0.00	0.00	0.00	4
neutral	0.67	1.00	0.80	10
positive	1.00	0.50	0.67	2
accuracy			0.69	16

The provided classification report outlines the performance metrics of the SVM model on the testing data. For the 'negative' sentiment class, the precision, recall, and F1-score are all reported as 0.00. This indicates that the model did not correctly identify any instances of the 'negative' class, leading to a lack of both precision and recall for this category. For the 'neutral' sentiment class, the model achieved a relatively high precision of 0.67, indicating that when it predicted a sentiment as 'neutral', it was correct 67% of the time. Additionally, the recall for 'neutral' sentiment is 1.00, meaning the model correctly identified all instances of 'neutral' sentiment. The F1-score, which balances precision and recall, is reported as 0.80 for 'neutral'. Regarding the 'positive' sentiment class, the precision is perfect at 1.00, indicating that all predictions for 'positive' sentiment were correct. However, the recall is 0.50, suggesting that the model missed some instances of 'positive' sentiment. The F1-score for 'positive' sentiment is 0.67. Overall, the accuracy of the model is reported as 0.69, indicating that it correctly classified sentiments for approximately 69% of the instances in the testing set.

Analysing Sentiment for Specific Name

This script performs sentiment analysis on messages associated with a specific name from a Data Frame. It first filters the messages for the given name and then calculates the percentage of positive, negative, and neutral sentiments among those messages. The sentiment analysis is conducted using TextBlob, where each message's polarity score determines its sentiment. The script then prints the sentiment analysis results for the specified name, indicating the percentage of positive, negative, and neutral sentiments. Additionally, it evaluates whether the sentiment indicates predatory behavior by comparing the negative sentiment percentage to a predefined threshold. If the negative sentiment percentage exceeds the threshold, the script identifies the name as exhibiting highly predatory behavior; otherwise, it concludes that the name does not exhibit highly predatory behavior. This approach provides insights into the sentiment

distribution and helps identify potential predatory behavior based on negative sentiment patterns in the messages associated with a particular name.

Sentiment Analysis for Victim1:

The sentiment analysis conducted for the name "Victim1" reveals that out of the analysed messages, approximately 27.78% are classified as positive, 11.11% as negative, and 61.11% as neutral. Despite the majority of messages being neutral, the negative sentiment percentage surpasses the predefined threshold, indicating a potential cause for concern. The calculated negative sentiment percentage suggests that "Victim1" exhibits highly predatory behavior, warranting further attention or investigation into the nature of the messages associated with this name.

```
Enter name: Victim1
Sentiment analysis for Victim1:
Positive: 27.78%
Negative: 11.11%
Neutral: 61.11%
Victim1 exhibits highly predatory behavior.
```

2. BERT:

Analysing Context for Specific Name:

This script conducts contextual analysis on messages associated with a specific name from a Data Frame. It first filters the messages for the given name and then tokenizes them using the BERT (Bidirectional Encoder Representations from Transformers) tokenizer. The pre-trained BERT model is then utilized to predict the sentiment of each message. The sentiment predictions are aggregated to calculate a predator score, where positive sentiment predictions contribute to the score indicating predatory behavior, while negative sentiment predictions contribute to the score indicating affectionate behavior. The script prints the contextual analysis results for the specified name, determining whether the individual exhibits predatory behavior, affectionate behavior, or no predatory behavior based on the calculated predator score. This approach provides insights into the context of the conversation associated with a particular name, helping identify potential predatory or affectionate behavior patterns.

```
Enter name: Victim1
Contextual analysis for Victim1:
Victim1 exhibits affectionate behavior.
```

The output indicates that the contextual analysis for the name "Victim1" suggests that the behavior exhibited is affectionate. This determination was made based on the sentiment predictions generated by the BERT model, where the aggregation of sentiment scores led to the conclusion of affectionate behavior. This analysis provides insights into the context of the conversation associated with the name "Victim1," aiding in the identification of behavioral patterns.

3. TEXTBLOB:

Analysing Context for Specific Name:

This script performs contextual analysis on messages associated with a specific name from a Data Frame. The `analyze_context` function iterates through each message, using TextBlob to tokenize and convert the text to lowercase for uniform processing. It then checks for specific phrases that may indicate potentially predatory behavior, such as "stop," "do not," or "leave me alone," incrementing the `predator_score` accordingly. Additionally, it evaluates phrases suggesting affectionate behavior, such as "love you" or "miss you," adjusting the `predator_score` based on the sender and receiver profiles provided. If the sender and receiver profiles match certain criteria, the `predator_score` is modified to reflect different levels of risk or appropriateness. The function `analyze_context_for_name` filters messages for the given name and calls `analyze_context` to compute the `predator_score`. Finally, it prints the contextual analysis results, identifying whether the individual exhibits potentially predatory behavior, affectionate behavior, or no predatory behavior based on the `predator_score`. This approach provides insights into the context of the conversation associated with a particular name and helps identify potential red flags or concerning patterns in communication.

```
Enter name: Predator
Enter sender profile (e.g., teacher, boyfriend): teacher
Enter receiver profile (e.g., child): student
Contextual analysis for Predator:
No predatory behavior detected for Predator.
```

The output indicates that after conducting contextual analysis for the name "Predator," considering the sender profile as "teacher" and the receiver profile as "student," no predatory

behavior was detected. This determination was made based on the absence of phrases indicating potentially predatory behavior and the specific conditions set in the code to evaluate the sender and receiver profiles. Therefore, the analysis concludes that "Predator" does not exhibit predatory behavior in the provided context.

4. NEURAL NETWORKS:

Analysing Chat Data with Linguistic Features:

This script is designed to analyze chat data using linguistic features and train a Random Forest classifier for message classification. It begins by importing necessary libraries such as pandas for data manipulation, numpy for numerical operations, spaCy for natural language processing, and scikit-learn for machine learning tasks. The pre-trained spaCy model `en_core_web_sm` is loaded to facilitate text processing. After reading chat data from a CSV file and removing rows with missing values in the 'Message' column, the script defines a function `extract_features` to extract linguistic features from text messages using spaCy, specifically part-of-speech (POS) tags for each token. These features are then applied to each message in the Data Frame, and the extracted linguistic features are stored in a new column called 'Linguistic_Features'. Subsequently, the linguistic features are transformed into TF-IDF vectors using `TfidfVectorizer`, and the data is split into training and testing sets using `train_test_split`. A Random Forest classifier is then trained on the training data using `RandomForestClassifier`. Finally, the trained classifier is used to make predictions on the test data, and accuracy along with a classification report is printed to evaluate the model's performance. This comprehensive pipeline enables the analysis of chat data leveraging linguistic features and machine learning techniques for message classification.

Accuracy: The overall accuracy of the model is reported as 0.1875, indicating that the model correctly predicted 18.75% of the test samples.

Classification Report: This report provides precision, recall, and F1-score for each class label in the dataset. Here are some key points from the classification report:

Precision: Precision measures the proportion of correctly predicted instances among all instances predicted for a particular class. For example, for the class 'Please', the precision is 0.50, indicating that half of the instances predicted as 'Please' were correct.

Recall: Recall measures the proportion of correctly predicted instances of a class among all instances of that class in the dataset. For example, for the class 'Please', the recall is 1.00, indicating that all instances of 'Please' were correctly predicted.

F1-score: F1-score is the harmonic mean of precision and recall and provides a balance between precision and recall. It is particularly useful when classes are imbalanced.

CHAPTER – 05

TEST RESULTS AND DISCUSSION:

1. Profile Creation:

Test Case Description	Coding Status	App Status
User registration	Passed	Passed
User login/logout	Passed	Passed
Profile creation/editing	Passed	Passed

2. Analysing Predatory Behaviour:

Test Case Description	Coding Status	App Status
Analysis of linguistic patterns for predatory behavior	Passed	Working
Code using SVM algorithm, BERT, NLP for predatory behavior prediction	Passed	Working
Divide the predators into three stages: 1. friend/boyfriend 2. predator(risk) 3. sexual (high risk)	Passed	Working

3. Alert System Functionality:

Test Case Description	Coding Status	App Status
Real-time alert notifications	Passed	Working
Customizable alert thresholds	Working	Working

4. Educational Content Integration:

Test Case Description	Coding Status	App Status
Access to educational articles	Working	Passed
Viewing YouTube educational videos	Working	Passed
Integration with YouTube API	Working	Working
Integration with Google's API for articles	Working	Working

5. Data Security and Privacy:

Test Case Description	Coding Status	App Status
Encryption of user data	Working	Working
Backup and restoration of data	Working	Working
Privacy Protection	Working	Working

CHAPTER – 06

CONCLUSION:

The project is centered around the development of a multifaceted tool aimed at detecting and addressing predatory behavior in online interactions, with a particular focus on safeguarding teenagers. A primary goal of the project is the early identification of potential threats, achieved through the utilization of cutting-edge machine learning models and natural language processing algorithms. These tools enable the system to analyze linguistic patterns, detect behavior shifts, and contextual cues within text interactions. Additionally, the system employs image recognition algorithms to scrutinize profile pictures and content postings for anomalies or signs indicative of fake profiles, thereby enhancing its ability to identify potential threats at an early stage

Integral to the project is the establishment of a proactive alert system that operates in real-time. This system leverages artificial intelligence for sentiment analysis and employs machine learning to recognize patterns associated with predatory behavior. Through this mechanism, the system can promptly trigger alerts to notify both teenagers and designated guardians about potential threats as they arise, facilitating timely intervention and protection.

In addition to proactive threat detection, the project emphasizes empowering teenagers with knowledge and resources to recognize and respond to potential threats independently. Educational content, including articles, videos, and interactive modules, is provided on topics such as online safety, healthy relationships, and the identification of manipulation tactics. By equipping teenagers with this information, the project aims to foster informed decision-making and promote safer online interactions.

Furthermore, the project incorporates customizable alert thresholds and safety settings to cater to the individual preferences and comfort levels of users and guardians. Through a user-friendly interface, stakeholders can adjust these settings based on their specific needs, providing a personalized and adaptable safety framework. This customization ensures that the system remains responsive to evolving threats and user requirements, enhancing its effectiveness in safeguarding users from predatory behavior.

FINDINGS:

The outcomes of the project represent a significant advancement in the realm of online safety, particularly for teenagers who are vulnerable to predatory behavior in digital spaces. Through an extensive literature review, key insights were gleaned, leading to the identification of crucial keywords indicative of predatory behavior. Furthermore, the classification of predators into distinct stages—ranging from acquaintances to high-risk individuals—provided a comprehensive framework for understanding and addressing various levels of threat.

The development of multiple machine learning models, including SVM, BERT, TextBlob, and neural networks (specifically Random Forest), showcased the project's commitment to leveraging cutting-edge technology for real-time predator identification. By integrating these models with the identified keywords from the literature review, the system was able to effectively detect and classify predators into different risk categories, enabling prompt intervention and protection measures.

The completion of the Figma design for the app marked a significant milestone in the project, offering a tangible visualization of the platform's user interface and functionality. The interconnectedness of various app pages and the ability to run a demo app within Figma provided stakeholders with a clear understanding of the app's layout and user experience, fostering confidence in its usability and effectiveness.

Additionally, the basic backend implementation using Flutter laid the foundation for essential functionalities such as user profile creation. While still in its nascent stage, this backend development represents a crucial step towards the eventual deployment and usability of the app, underscoring the project's progress and potential for future expansion.

In conclusion, the project has made substantial strides towards its overarching goal of enhancing online safety for teenagers through early detection and proactive intervention against predatory behavior. By combining insights from literature reviews with advanced machine learning techniques and robust app development efforts, the project has positioned itself as a promising tool in the ongoing battle against online threats, empowering users with the knowledge and resources needed to navigate digital spaces safely and confidently.

FUTURE IMPROVEMENTS:

1. Adaptive Alert Thresholds: (Future)
    a. Goal: Develop adaptive alert thresholds that dynamically adjust based on user behavior and interaction patterns.
    b. Functionality: Implement machine learning algorithms that continuously analyze user interactions and adjust alert thresholds accordingly. This functionality ensures that alerts are tailored to each user's specific context and risk tolerance, optimizing the balance between sensitivity and specificity.
2. Multimodal Analysis: (Future)
    a. Goal: Expand the analysis capabilities to include multimodal data sources such as images, audio, and video.
    b. Functionality: Integrate image and audio recognition algorithms to analyze multimedia content for potential signs of predatory behavior. By considering multiple modalities, the system can enhance its detection capabilities and provide more comprehensive insights into user interactions.
3. Enhanced Privacy Controls: (Future)
    a. Goal: Provide users with granular control over their privacy settings and data sharing preferences.
    b. Functionality: Develop a user-friendly interface that allows users to customize privacy settings, including the level of data sharing with the platform and third-party services. Additionally, implement encryption techniques to further safeguard user data and ensure compliance with privacy regulations.
4. Continuous Machine Learning Improvement: (Future)
    a. Goal: Regularly update & improve the ML models based on new data & insights.
    b. Functionality: Establish a system for regular updates, incorporating new data & insights from ongoing research in psychology & online behavior.
5. User Feedback Mechanism: (Future)
    a. Goal: Gather feedback from users to improve the accuracy of alerts & the overall user experience.
    b. Functionality: Implement features that allow users to provide feedback on the effectiveness of alerts & report any false positives or negatives.

APPENDICES:

RAW DATA:

https://docs.google.com/spreadsheets/d/1v2Zr7kSRHb6ohr1Z2_R7LMJvTfME8iB_sU-2hxn5d-I/edit?usp=sharing

CODE SNIPPETS:

1. **Importing Data and Data Parsing and Cleaning Process:**

```
from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

# read file by lines

# file_path = "/content/drive/My Drive/RVU S4/CSectiontxt.txt"

file_path = "/content/drive/My Drive/RVU S4/RVU-BTech22 Students.txt"

# file_path = "/content/drive/My Drive/RVU S4/Staker.txt"

f = open(file_path, 'r')

data = f.readlines()

f.close()

# sanity stats

print('num lines: %s' %(len(data)))

# parse text and create list of lists structure

# remove first whatsapp info message

dataset = data[1:]

cleaned_data = []

for line in dataset:
```

```
        # grab the info and cut it out
        date = line.split(",")[0]
        line2 = line[len(date):]
        time = line2.split("-")[0][2:]
        line3 = line2[len(time):]
        name = line3.split(":")[0][4:]
        line4 = line3[len(name):]
        message = line4[6:-1] # strip newline charactor
        #print(date, time, name, message)
        cleaned_data.append([date, time, name, message])
# Create the DataFrame
df = pd.DataFrame(cleaned_data, columns = ['Date', 'Time', 'Name', 'Message'])
# check formatting
if 0:
        print(df.head())
        print(df.tail())
# Save it!
df.to_csv(r'/content/drive/My Drive/CMRVU.csv', index=False)
```

2. **Data Filtering and Exploration Process**

```
df.shape
df.info()
df.tail()
df1 = df[df['Message'] != '<Media omitted>']
df1
```

3. **Data Cleaning and Handling Missing Values**

```
import numpy as np

df1.replace('', np.nan, inplace=True)

# Drop rows where any column contains missing (NaN) values

df1 = df1.dropna(how='any')
```

4. **Removing Messages Consisting Only of Emojis**

```
import pandas as pd

import re

# Example DataFrame

# df = pd.DataFrame({'Message': ['Hello 😊', 'How are you? 😳', '😊', '👍', '😂😂', '']})

# Define a function to check if the message consists only of emojis

def contains_only_emojis(text):

  emoji_pattern = re.compile("["

                u"\U0001F600-\U0001F64F"  # emoticons

                u"\U0001F300-\U0001F5FF"  # symbols & pictographs

                u"\U0001F680-\U0001F6FF"  # transport & map symbols

                u"\U0001F1E0-\U0001F1FF"  # flags (iOS)

                u"\U00002500-\U00002BEF"  # chinese char

                u"\U00002702-\U000027B0"

                u"\U00002702-\U000027B0"

                u"\U000024C2-\U0001F251"

                u"\U0001f926-\U0001f937"

                u"\U00010000-\U0010ffff"

                u"\u2640-\u2642"
```

```
                u"\u2600-\u2B55"
                u"\u200d"
                u"\u23cf"
                u"\u23e9"
                u"\u231a"
                u"\ufe0f"  # dingbats
                u"\u3030"
                "]+", flags=re.UNICODE)
    return bool(emoji_pattern.fullmatch(text))
# Remove rows where 'Message' column contains only emojis
df1 = df1[~df1['Message'].apply(contains_only_emojis)]
# Now df_filtered contains rows where 'Message' column does not contain only emojis
df1
```

5. **Sentiment Analysis with Textblob**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from textblob import TextBlob
# Example DataFrame
# df = pd.DataFrame({'Message': ['This movie is great!', 'I don\'t like this product.',
'Amazing!', 'Terrible.']})
# Define a function to calculate sentiment polarity using TextBlob
```

```
def get_sentiment(text):
    blob = TextBlob(text)
    return 'positive' if blob.sentiment.polarity > 0 else 'negative' if blob.sentiment.polarity < 0 else 'neutral'
# Apply the function to the 'Message' column
df1['Sentiment'] = df1['Message'].apply(get_sentiment)
# Now df contains a new 'Sentiment' column with sentiment labels (positive, negative, neutral)
```

6. **SVM:**

- **Selecting The Features:**

```
X = df1['Message']
y = df1['Sentiment']
```

- **Split data into training and testing sets**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **Convert text data into numerical features**

```
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

- **Define and train SVM model**

```
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_vectorized, y_train)
```

- **Evaluate model**

```
y_pred = svm_model.predict(X_test_vectorized)
print(classification_report(y_test, y_pred))
```

- **Analysing Sentiment for Specific Name**

```
import pandas as pd
from textblob import TextBlob
def analyze_sentiment(messages):
    positive_count = 0
    negative_count = 0
    neutral_count = 0
    total_count = len(messages)
    for message in messages:
        blob = TextBlob(message)
        polarity = blob.sentiment.polarity
        if polarity > 0:
            positive_count += 1
        elif polarity < 0:
            negative_count += 1
        else:
            neutral_count += 1
    positive_percentage = (positive_count / total_count) * 100
    negative_percentage = (negative_count / total_count) * 100
    neutral_percentage = (neutral_count / total_count) * 100
    return positive_percentage, negative_percentage, neutral_percentage, total_count
def analyze_sentiment_for_name(name):
    # Filter messages for the given name
    filtered_messages = df[df['Name'] == name]['Message']
    # Analyze sentiment for the filtered messages
```

```
    positive_percentage, negative_percentage, neutral_percentage, total_count =
analyze_sentiment(filtered_messages)
    print(f"Sentiment analysis for {name}:")
    print(f"Positive: {positive_percentage:.2f}%")
    print(f"Negative: {negative_percentage:.2f}%")
    print(f"Neutral: {neutral_percentage:.2f}%")
    # Determine if the sentiment indicates predatory behavior
    if total_count == 0:
        print("No messages found for analysis.")
    else:
        predator_threshold = 30  # Adjust threshold as needed
        predator_score = (negative_percentage / (total_count + 1)) * 100  # Adding 1 to avoid
division by zero
        if predator_score > predator_threshold:
            print(f"{name} exhibits highly predatory behavior.")
        else:
            print(f"{name} does not exhibit highly predatory behavior.")
# Example usage:
name = input("Enter name: ")
analyze_sentiment_for_name(name)
```

7. **BERT:**

```
import pandas as pd
from transformers import BertTokenizer, BertForSequenceClassification
import torch
# Load pre-trained BERT model and tokenizer
```

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
model.eval()
def analyze_context(messages):
    predator_score = 0
    # Convert messages to tokens
    tokens = tokenizer(messages, padding=True, truncation=True, return_tensors='pt',
max_length=512, return_attention_mask=True)
    # Predict sentiment using BERT model
    with torch.no_grad():
        outputs = model(tokens)
    # Calculate predator score based on sentiment predictions
    predictions = torch.argmax(outputs.logits, dim=1)
    for prediction in predictions:
        if prediction == 1:  # Label 1 indicates negative sentiment
            predator_score += 1
        elif prediction == 0:  # Label 0 indicates positive sentiment
            predator_score -= 1
    return predator_score
def analyze_context_for_name(name):
    # Filter messages for the given name
    filtered_messages = df[df['Name'] == name]['Message'].tolist()
    # Analyze the context of the conversation for the filtered messages
    predator_score = analyze_context(filtered_messages)
```

```
    print(f"Contextual analysis for {name}:")
    if predator_score > 0:
        print(f"{name} exhibits predatory behavior.")
    elif predator_score < 0:
        print(f"{name} exhibits affectionate behavior.")
    else:
        print(f"No predatory behavior detected for {name}.")
# Example usage:
name = input("Enter name: ")
analyze_context_for_name(name)
```

8. **TEXTBLOB:**

```
import nltk
nltk.download('punkt')
import pandas as pd
from textblob import TextBlob
def analyze_context(messages, sender_profile, receiver_profile):
    predator_score = 0
    for message in messages:
        blob = TextBlob(message.lower())
        print ("hello")
        # Check for specific phrases indicating potentially predatory behavior
        if "stop" in blob.words or "do not" in blob.words or "leave me alone" in blob.words:
            predator_score += 1
        elif "love you" in blob.words or "miss you" in blob.words:
```

```
# Adjust the threshold based on sender and receiver profiles
if sender_profile == "teacher" and receiver_profile == "child":
    predator_score += 2  # Highly predatory behavior
elif sender_profile == "doctor" and receiver_profile == "patient":
    predator_score += 1  # Inappropriate behavior
elif sender_profile == "mentor" and receiver_profile == "mentee":
    predator_score += 1  # Abuse of power
elif sender_profile == "stranger" and receiver_profile == "minor":
    predator_score += 2  # Stranger danger
elif sender_profile == "online predator" and receiver_profile == "minor":
    predator_score += 3  # High risk of predatory behavior
elif sender_profile == "ex-partner" and receiver_profile == "victim":
    predator_score += 2  # Stalking or harassment
elif sender_profile == "employer" and receiver_profile == "employee":
    predator_score += 1  # Workplace harassment
elif sender_profile == "caretaker" and receiver_profile == "dependent":
    predator_score += 1  # Abuse of trust
# Additional criteria for friendly interactions between best friends
elif "best friend" in blob.words or "bff" in blob.words or "bestie" in blob.words:
    predator_score -= 1  # Reduce predator score for best friend references
    print ("hellobf")
    print(predator_score)
elif "fun" in blob.words or "joking" in blob.words or "laugh" in blob.words:
    print ("hellojk")
```

```
            predator_score -= 1  # Reduce predator score for references to fun or joking
    return predator_score
def analyze_context_for_name(name, df, sender_profile, receiver_profile):
    filtered_messages = df[df['Name'] == name]['Message']
    predator_score = analyze_context(filtered_messages, sender_profile, receiver_profile)
    print(f"Contextual analysis for {name}:")
    if predator_score > 0:
        print(f"{name} exhibits potentially predatory behavior.")
    elif predator_score < 0:
        print(f"{name} exhibits affectionate behavior.")
    else:
        print(f"No predatory behavior detected for {name}.")

# Example usage:
name = input("Enter name: ")
sender_profile = input("Enter sender profile (e.g., teacher, boyfriend): ")
receiver_profile = input("Enter receiver profile (e.g., child): ")
analyze_context_for_name(name, df, sender_profile, receiver_profile)
```

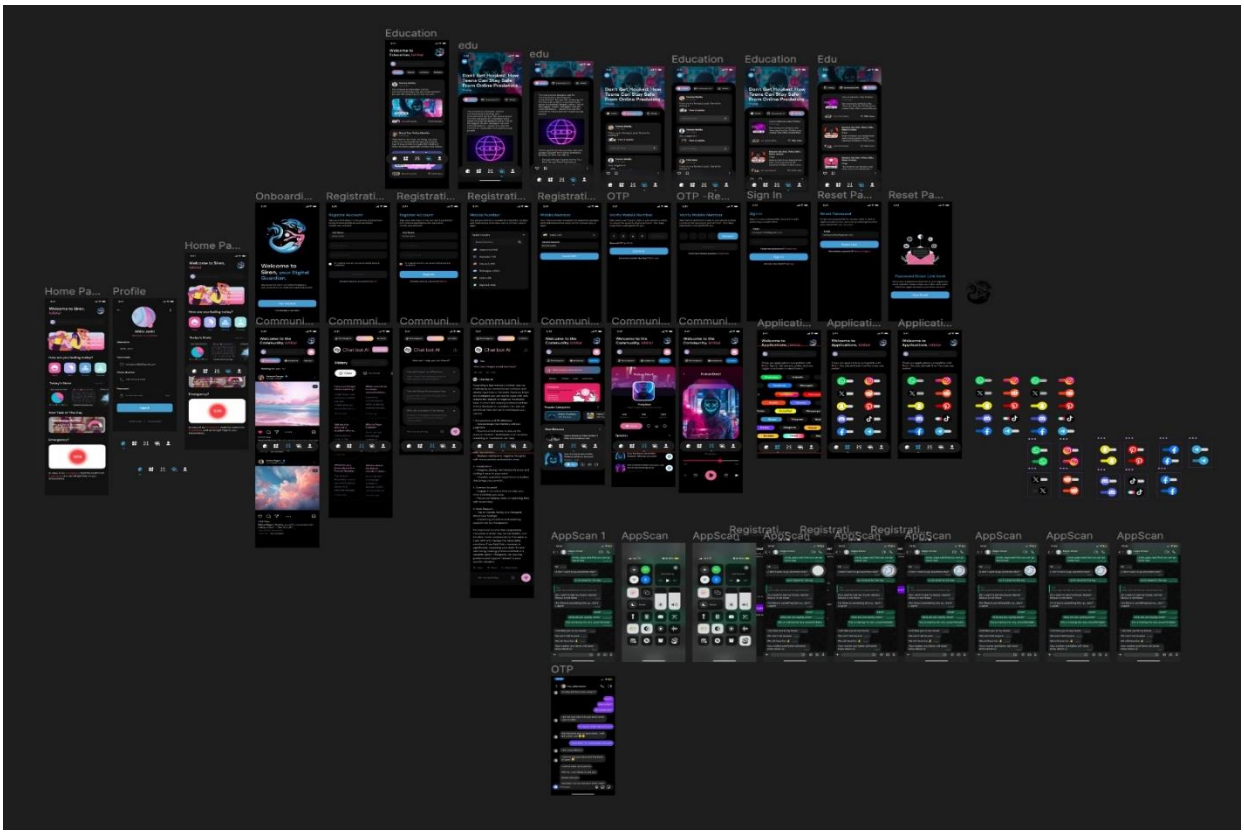
9. NEURAL NETWORKS:

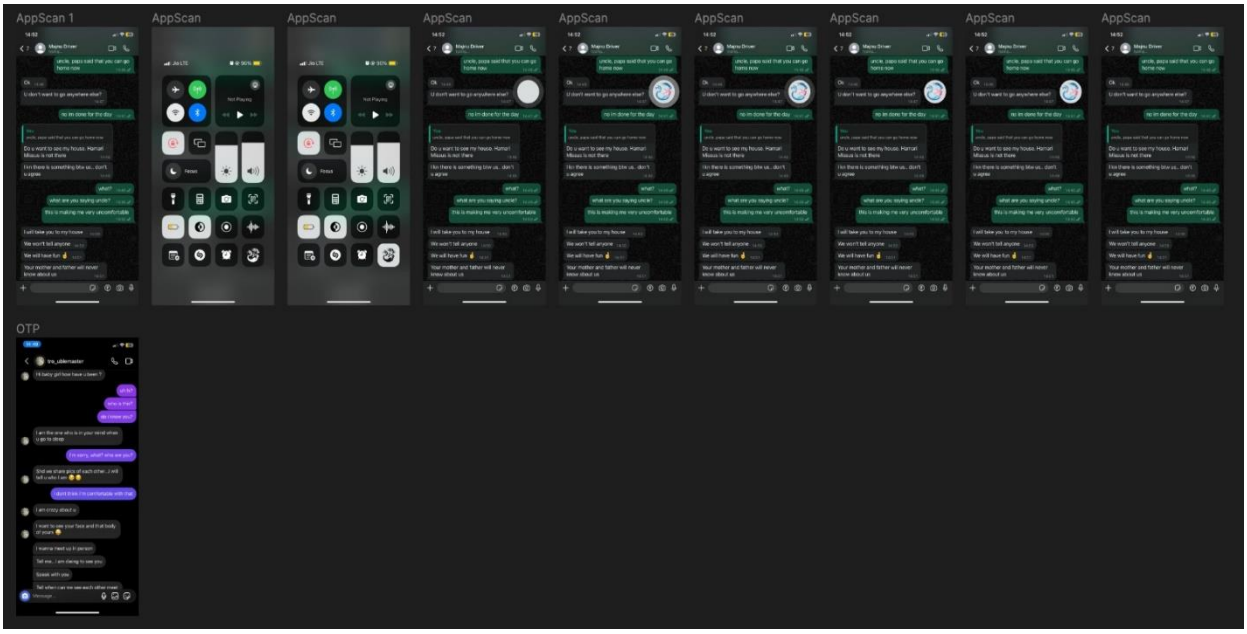
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load pre-trained spaCy model
```

```
nlp = spacy.load("en_core_web_sm")

# Read the CSV file containing chat data

# df = pd.read_csv("/content/drive/My Drive/CMStaker.csv")

# Remove rows with missing values in the 'Message' column

df1.dropna(subset=['Message'], inplace=True)

# Function to extract linguistic features from messages

def extract_features(text):

  doc = nlp(str(text))  # Convert to string to handle possible NaN values

  # Example: Extracting part-of-speech tags

  pos_tags = [token.pos_ for token in doc]

  # You can extract other linguistic features here

  return pos_tags

# Extract linguistic features from messages

df1['Linguistic_Features'] = df1['Message'].apply(extract_features)

# Convert linguistic features into TF-IDF vectors

tfidf_vectorizer = TfidfVectorizer(analyzer=lambda x: x)

tfidf_matrix = tfidf_vectorizer.fit_transform(df1['Linguistic_Features'])

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix, df1['Message'], test_size=0.2,
random_state=42)

# Train a Random Forest classifier

rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

rf_classifier.fit(X_train, y_train)

# Predictions
```

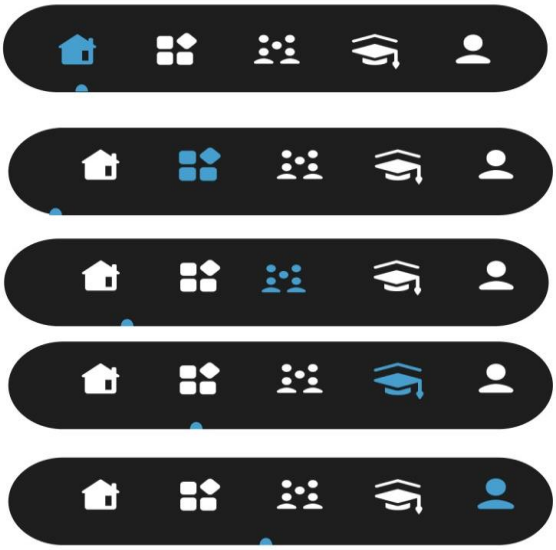
```
y_pred = rf_classifier.predict(X_test)

# Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Classification Report:")

print(classification_report(y_test, y_pred))
```

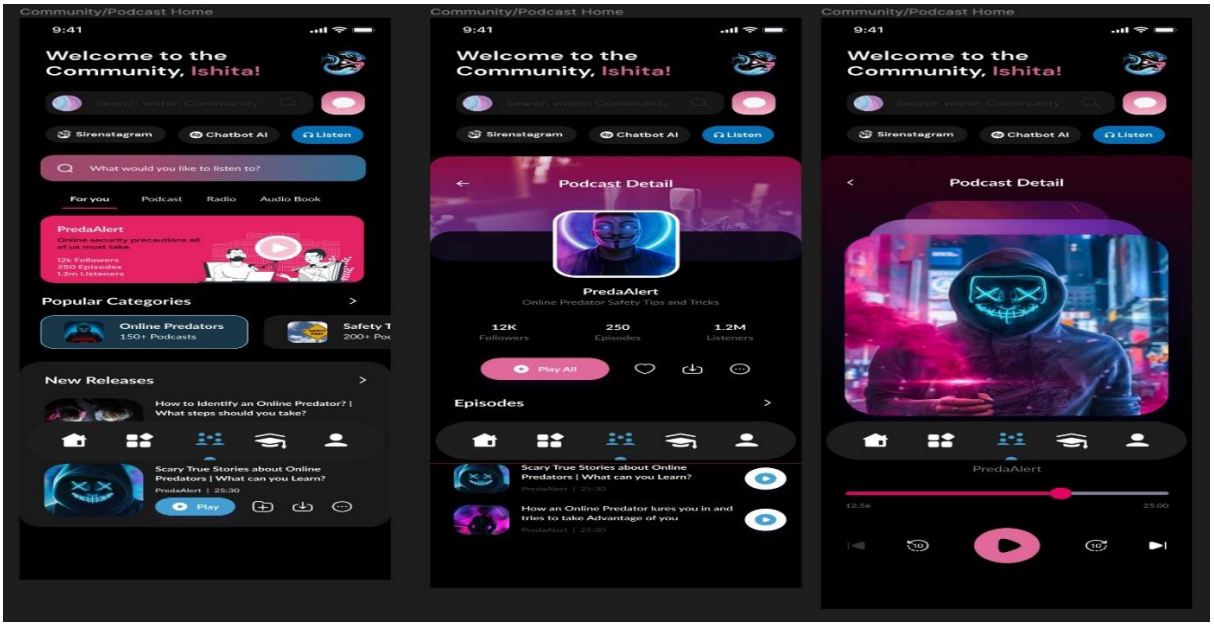
DESIGN:




AppScan 1
AppScan
AppScan
AppScan
AppScan
AppScan
AppScan
AppScan
AppScan
OTP


Home Page
9:41
Welcome to Siren, Ishita!
Search within Siren
You are safe, protected & empowered.
How are you feeling today?
Happy
Calm
Relaxed
Focused
Today's Stats
See all >
App Used with Siren
Predator Count per App
Safety St
Meet and discuss like minded individuals who have shared experiences.
75% done
Emergency?
SOS
In case of an emergency, hold the button for 5 seconds, and we will get help to you immediately.




Community/Podcast Home
9:41
Welcome to the Community, Ishita!
Search within Community
Sirenstagram
Chatbot AI
Listen
What would you like to listen to?
For you
Podcast
Radio
Audio Book
PredaAlert
Online security precautions all of us must take.
12k Followers
250 Episodes
1.2m Listeners
Popular Categories
Online Predators
150+ Podcasts
New Releases
How to Identify an Online Predator? | What steps should you take?
Scary True Stories about Online Predators | What can you Learn?
PredaAlert | 25:30
Play
Podcast Detail
PredaAlert
Online Predator Safety Tips and Tricks
12K
Followers
250
Episodes
1.2M
Listeners
Play All
Episodes
How an Online Predator lures you in and tries to take Advantage of you
12:56
25:00