5-9-2024

A NLP Approach to Automating the Generation of Surveys for Market Research

Anav Chug
Georgia Southern University

A NLP Approach to Automating the Generation of Surveys for Market Research

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in
Computer Science

By

Anav Chug

Under the mentorship of *Dr. Andrew Allen*

ABSTRACT

Market Research is vital but includes activities that are often laborious and time consuming. Survey questionnaires are one possible output of the process and market researchers spend a lot of time manually developing questions for focus groups. The proposed research aims to develop a software prototype that utilizes Natural Language Processing (NLP) to automate the process of generating survey questions for market research. The software uses a pre-trained Open AI language model to generate multiple choice survey questions based on a given product prompt, send it to a targeted email list, and also provides a real-time analysis of the responses stored in a SQL database. The idea is for market researchers to provide minimal information and have the system propose to them options for potential questions for the focus group. The project involves creating a web-based software prototype that features user-friendly interfaces for seamless interaction between companies and users. The expected outcome of the project is to provide a more efficient and effective method for companies to generate meaningful survey questions for market research.

Thesis Mentor: Dr. Andrew Allen

Honors Dean: Dr. Steven Engel

April 2024
Department of Computer Science
Honors College
Georgia Southern University

Acknowledgements

The completion of this thesis would not have been possible without the expertise and encouragement of my thesis mentor Dr. Andrew Allen. His support and thoughtful insights pushed me to think outside the box and write this thesis. He was very kind and patient throughout this project and I would always learn something new from him during every meeting I had with him. I also want to extend my gratitude to Dr. Ryan Florin for his feedback on my initial designs of the UI and database. He played a big role in helping me understand software design principles at a deeper level, both inside and outside of classwork and my ability to write better code.

In addition, my sincere gratitude goes to my sister Aakarsha Chug for her constant support and feedback during this project. I would also like to thank my family and friends for being a source of strength and motivation for me throughout my academic journey and helping me always.

Introduction

Natural language processing (NLP) is a subdomain of artificial intelligence (AI) that deals with the interaction of computers and humans using natural language. It requires the development of algorithms and models that allow machines to comprehend, translate, and generate human language. NLP is an interdisciplinary field that combines linguistics, computer science, and statistical mathematics in order to analyze and process large amounts of text-based information. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on [1]. These tools are used globally and have become increasingly sophisticated with advancements in machine learning and deep learning algorithms. Some popular examples of NLP tools include Google Translate, which can translate text, speech, images, and web pages in over 100 languages, Amazon Echo and Apple Siri, which use speech recognition to perform various tasks through voice commands. For sentiment analysis, IBM Watson Natural Language Understanding and Google Cloud Natural Language API can analyze text for sentiment, entities, and syntax, while for text summarization, TextRank can identify important sentences and summarize long articles into a few sentences. In recent years, the emergence of large language models has heralded a significant leap forward for the future of Natural Language Processing (NLP). These models have revolutionized the field, pushing the boundaries of what is possible and opening up new avenues for research and applications. One such model is the Generative Pre-Trained Transformer (GPT-3) which is a third-generation, autoregressive language model that uses deep learning to produce human-like text [6]. It is developed by OpenAI and is the third iteration in the GPT series, following GPT and GPT-2. GPT-3 is built on the Transformer architecture, a deep learning model architecture that has proven effective for various NLP tasks. It consists of a vast neural network with 175 billion parameters, making it one of the largest language models ever created. NLP tools have become essential for businesses, researchers, and individuals who deal with large volumes of text data and want to derive insights, make decisions, and communicate with others more efficiently. These tools are continually improving themselves as they receive more input data and hold enormous potential for revolutionizing how we interact with machines and each other. This paper explores the application of NLP in survey automation and analysis, demonstrated through a Flask web application utilizing the GPT-3 API. This application aims to facilitate companies in conducting market research and analyzing their products more efficiently. The benefits of using this application are twofold. Firstly, it enables companies to create survey questionnaires quickly, saving them valuable time. Instead of manually crafting each question, the application leverages the language model to automatically generate relevant and contextually appropriate survey questions. Secondly, the application empowers companies to analyze user responses collected through the generated surveys. By understanding the feedback, opinions, and demographics of their target audience, companies can gain valuable insights to improve their products. This feedback-driven approach helps them refine their offerings, enhance customer satisfaction, and effectively target their desired audience.

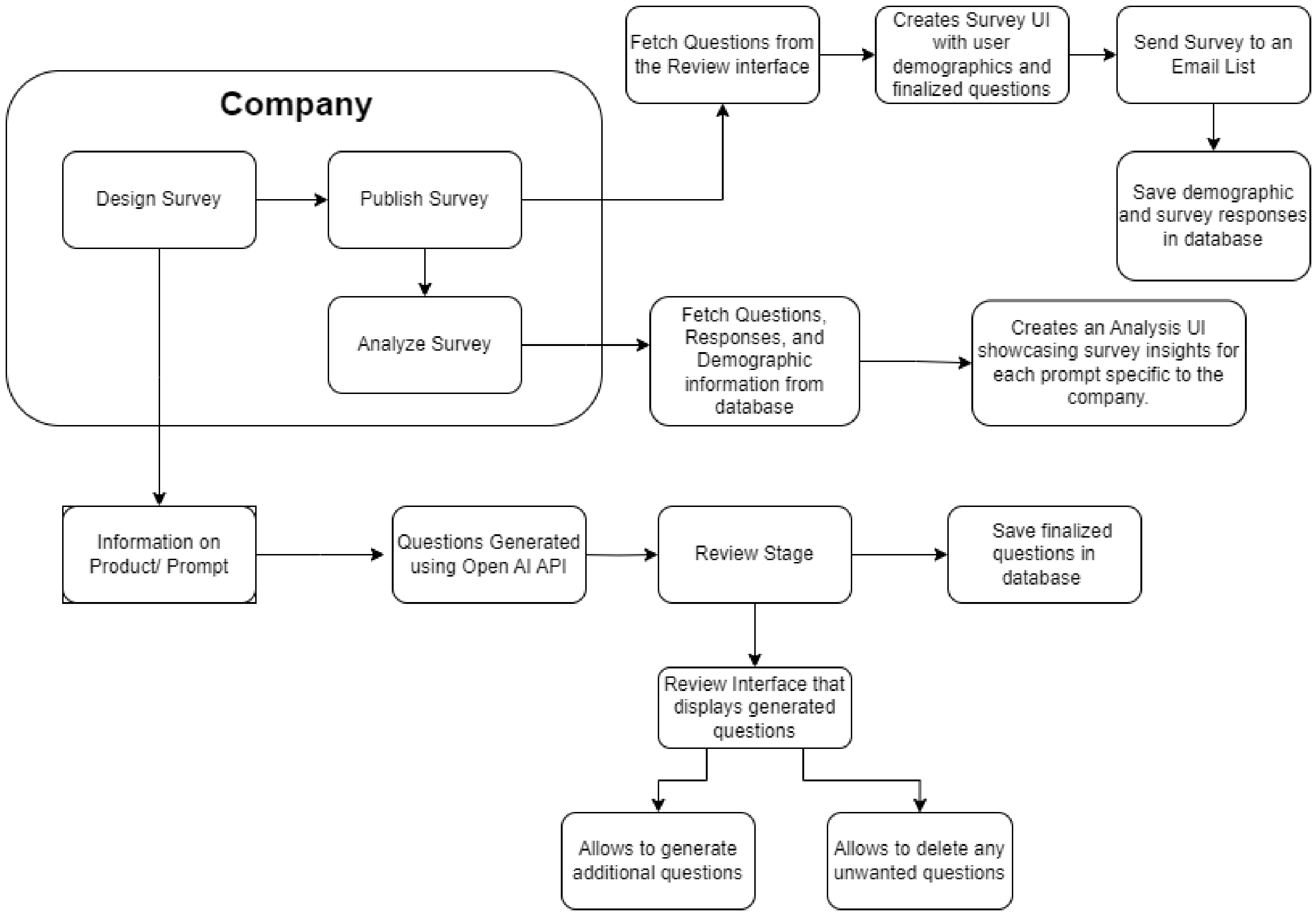
Background and Related Work

A lot of the research on question generation using NLP has applications in the education domain as e-learning in combination with artificial intelligence is a very acute topic today [2]. In recent years, NLP techniques have made significant progress in automatically generating questions from text prompts. These automatic question generation models have various applications, including education, chatbots, and information retrieval. These models use different approaches, such as rule-based, template-based, and machine learning-based methods, to generate grammatically correct and meaningful questions. Rule-based and template-based methods rely on predefined patterns and heuristics to generate questions, while machine learning-based methods use statistical models to learn from a large amount of data and generate questions that are more context-aware and relevant. In [3] a method is proposed to automate questionnaire surveys by leveraging collective messages from the Internet, enabling efficient and real-time data collection. However, this method still requires researchers to design the questionnaire manually which is something we aim to overcome in this research.

In [4], a framework is proposed for generating factual questions from unstructured text in English using NLP techniques, which combines traditional linguistic approaches with machine learning methods and includes a question evaluation module. In [5], an attention-based sequence learning model for automatic question generation was proposed, which outperforms the state-of-the-art rule-based system and produces more natural and challenging questions according to both automatic and human evaluations. The advantage of the data-driven approach used in these papers is that it can be customized and fine-tuned to work specifically with the type of text and questions one is interested in generating. However, one limitation of these models is that they have been trained on a relatively smaller dataset compared to large language models (LLMs) like GPT-3. As a result, they might not have the same level of contextual understanding and ability to generate relevant questions tailored to different contexts or domains. Additionally, the model's training data might not capture as many nuanced language patterns as GPT-3, potentially affecting the quality and diversity of the generated survey questions. GPT-3 supports zero-shot and few-shot learning, which means it can perform tasks with minimal or no specific training examples. In the context of survey automation, this allows us to generate survey questions without explicitly training the model on a large set of question-answer pairs. Instead, we can simply provide a prompt to GPT-3, and it can generate questions based on that input alone. GPT-3 has demonstrated the ability to generate creative and diverse outputs. It can provide a wider variety of question structures and phrasings, potentially leading to more engaging and interesting survey questions. This can help improve user engagement and response rates. Although there is a lack of research on using NLP for automating survey question generation in market research, this project aims to fill the gap and contribute to the development of the field by using NLP to generate survey questions based on the provided information, thus paving the way for more efficient and effective market research.

Design and Methodology

This research paper presents a web application designed to facilitate the generation of questionnaires and analysis of survey data. The application aims to provide an intuitive and user-friendly interface for creating customized surveys, collecting responses, and generating valuable insights. The system is divided into several key components, starting with an interface that allows marketing research teams to create and manage surveys. The interface includes features for defining survey prompts and generating corresponding questions based on product information prompts. The application incorporates data storage and analysis capabilities, utilizing a well-defined database schema to store survey-related data, including company information, survey prompts, generated questions, user demographics, and survey analytics. The database design ensures efficient data retrieval and analysis by establishing appropriate relationships between tables

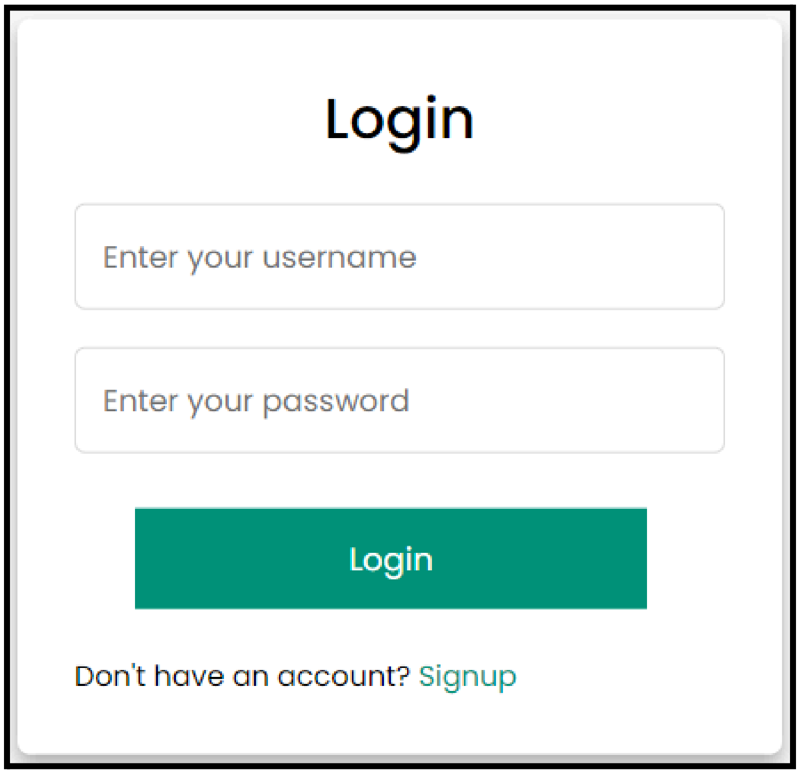

Company
Design Survey
Publish Survey
Analyze Survey
Fetch Questions from the Review interface
Creates Survey UI with user demographics and finalized questions
Send Survey to an Email List
Save demographic and survey responses in database
Fetch Questions, Responses, and Demographic information from database
Creates an Analysis UI showcasing survey insights for each prompt specific to the company.
Information on Product/ Prompt
Questions Generated using Open AI API
Review Stage
Save finalized questions in database
Review Interface that displays generated questions
Allows to generate additional questions
Allows to delete any unwanted questions

.Application Workflow

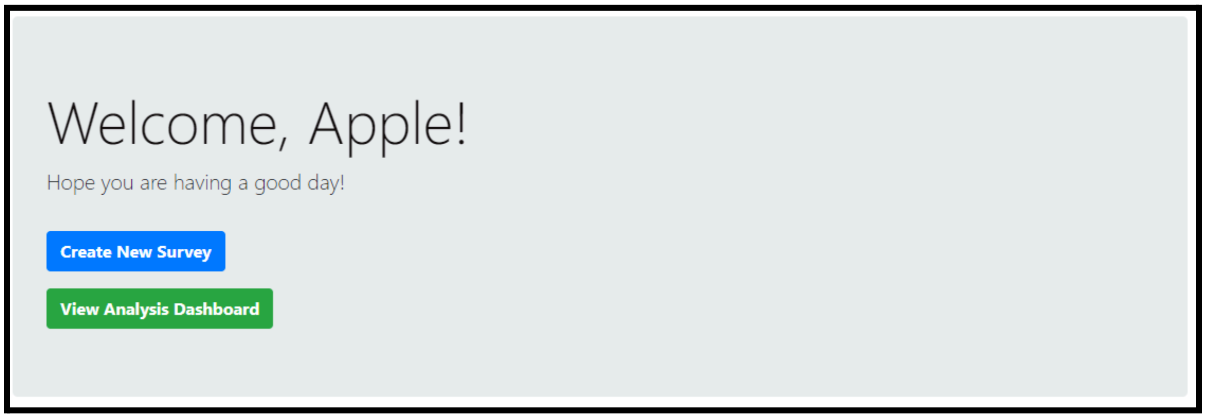
Front End Design

The front-end component of the application comprises six interfaces:

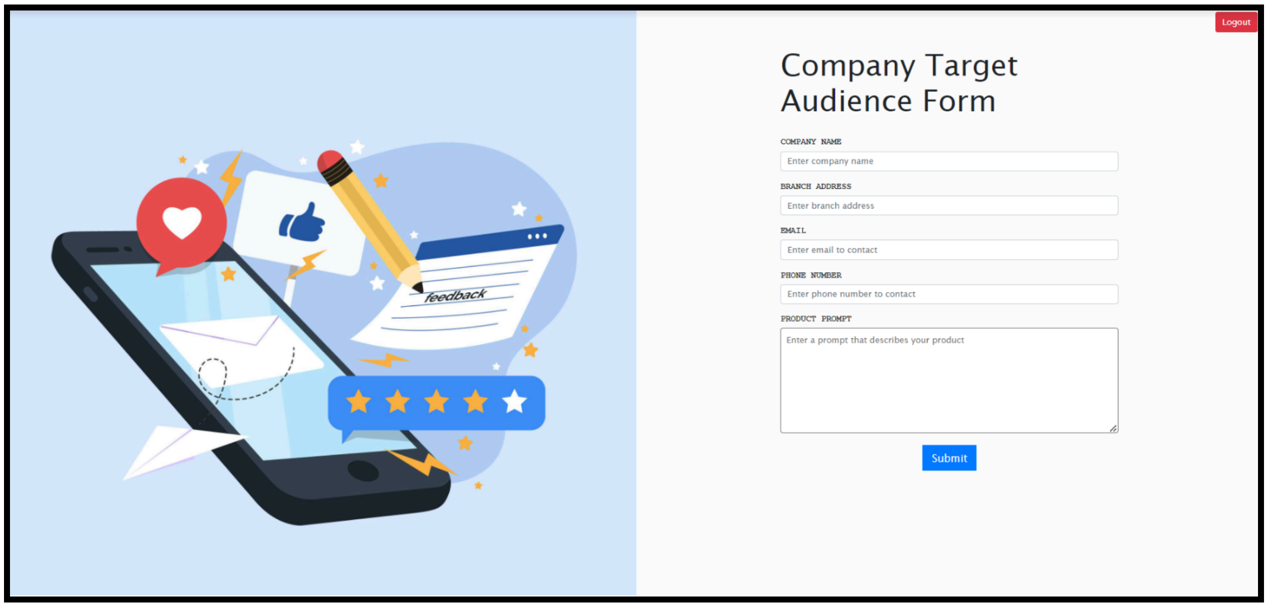
- **Login & Register Page-** Sign in to your account or create a new one to access the system.



- **Company Dashboard-** Create new surveys or view existing survey analysis



- **Company Information Page-** Input company information and provide a description of their product. Companies that have already entered their information once will only see the prompt text box.



Company Target Audience Form

COMPANY NAME
Enter company name

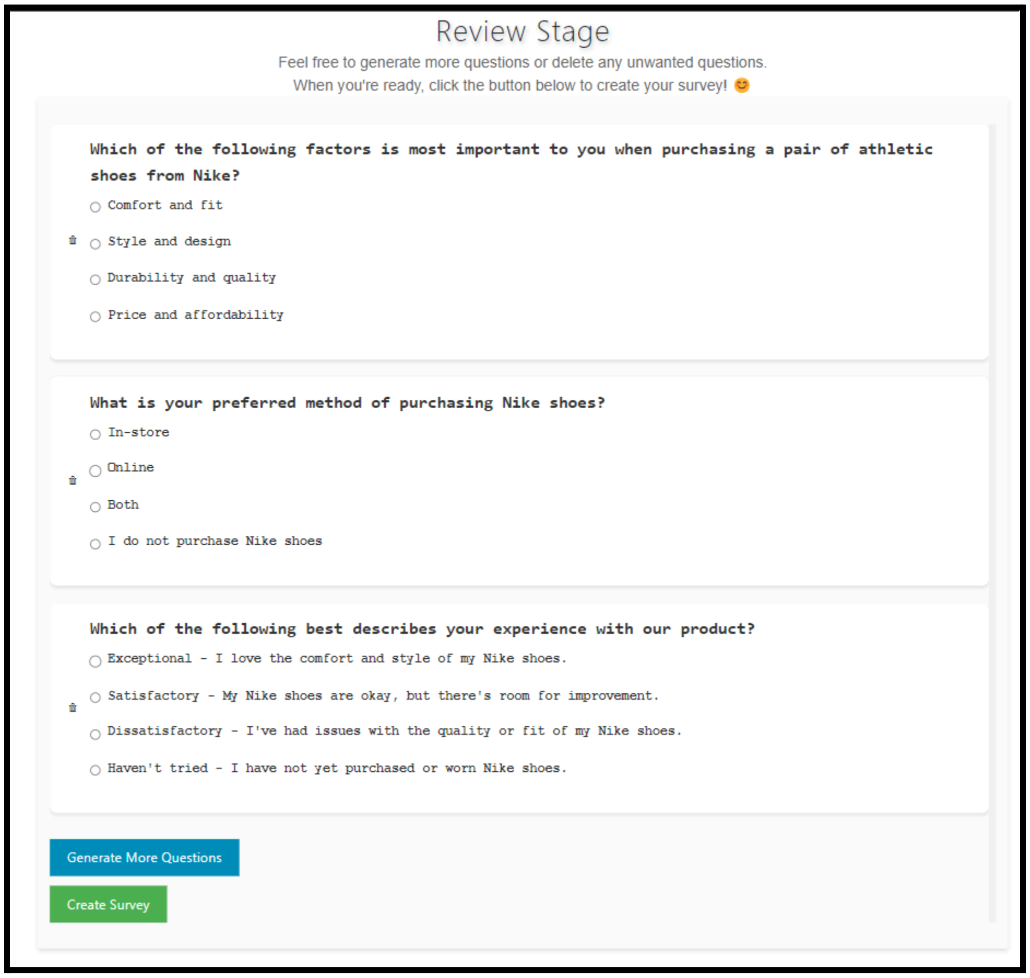
BRANCH ADDRESS
Enter branch address

EMAIL
Enter email to contact

PHONE NUMBER
Enter phone number to contact

PRODUCT PROMPT
Enter a prompt that describes your product

- **Review Page-** Add and delete questions to your survey



Review Stage

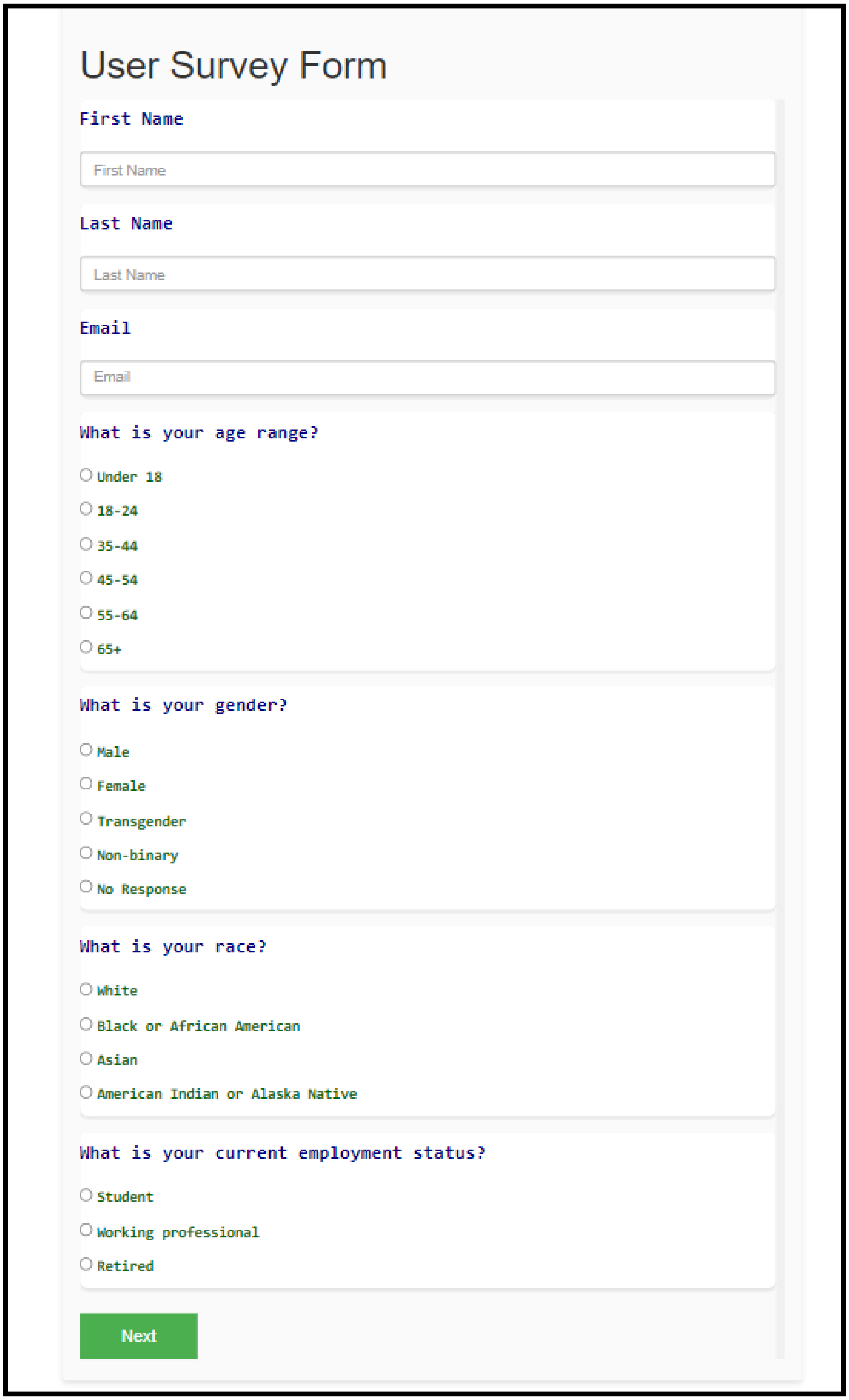
Feel free to generate more questions or delete any unwanted questions.
When you're ready, click the button below to create your survey! 😊

Which of the following factors is most important to you when purchasing a pair of athletic shoes from Nike?
○ Comfort and fit
○ Style and design
○ Durability and quality
○ Price and affordability

What is your preferred method of purchasing Nike shoes?
○ In-store
○ Online
○ Both
○ I do not purchase Nike shoes

Which of the following best describes your experience with our product?
○ Exceptional - I love the comfort and style of my Nike shoes.
○ Satisfactory - My Nike shoes are okay, but there's room for improvement.
○ Dissatisfactory - I've had issues with the quality or fit of my Nike shoes.
○ Haven't tried - I have not yet purchased or worn Nike shoes.

- **Survey Questionnaire Page-** View the finalized survey with demographic information fields



User Survey Form

First Name

First Name

Last Name

Last Name

Email

Email

What is your age range?

- ○ Under 18
- ○ 18-24
- ○ 35-44
- ○ 45-54
- ○ 55-64
- ○ 65+

What is your gender?

- ○ Male
- ○ Female
- ○ Transgender
- ○ Non-binary
- ○ No Response

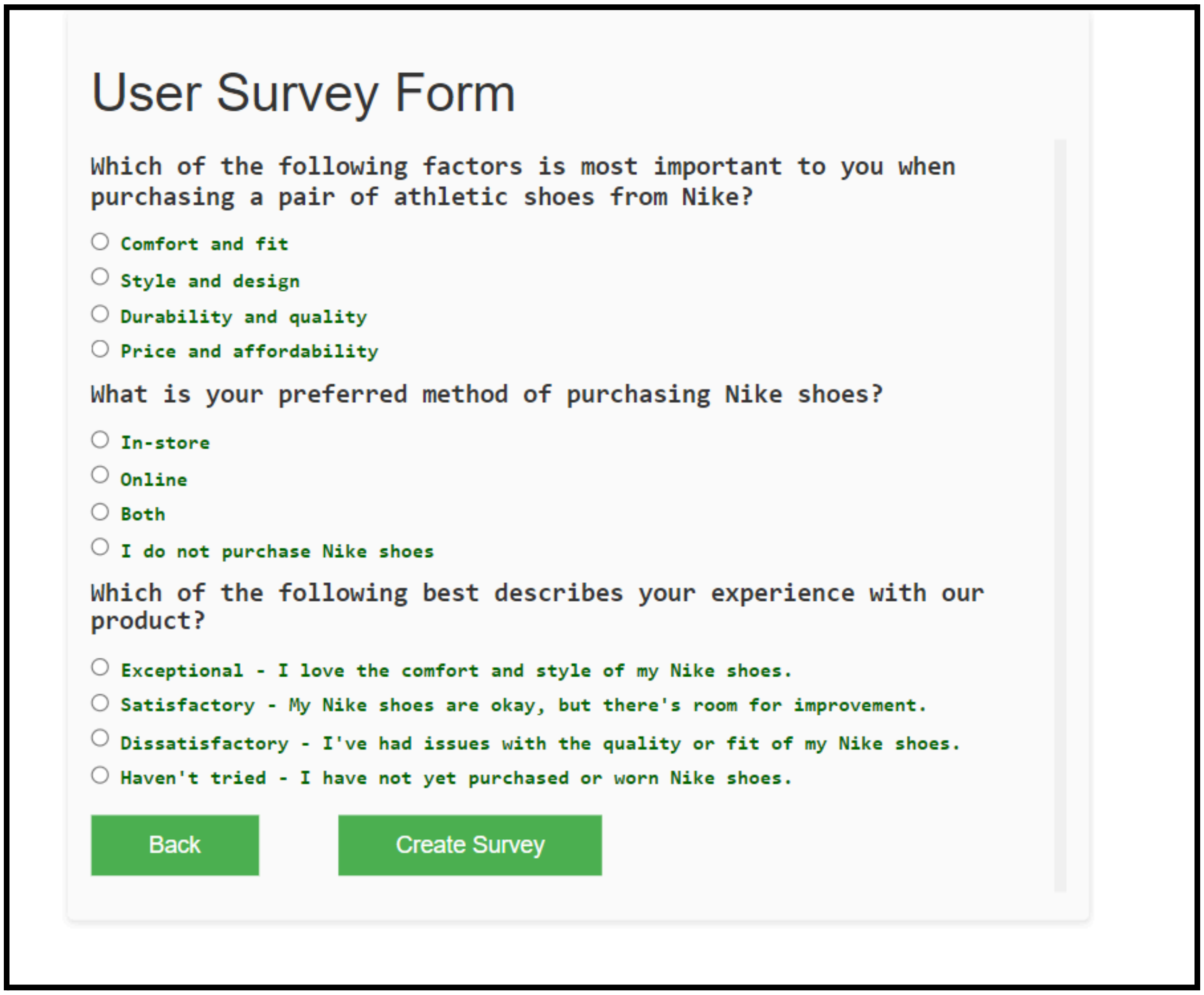
What is your race?

- ○ White
- ○ Black or African American
- ○ Asian
- ○ American Indian or Alaska Native

What is your current employment status?

- ○ Student
- ○ Working professional
- ○ Retired

Next



User Survey Form

Which of the following factors is most important to you when purchasing a pair of athletic shoes from Nike?

- ○ Comfort and fit
- ○ Style and design
- ○ Durability and quality
- ○ Price and affordability

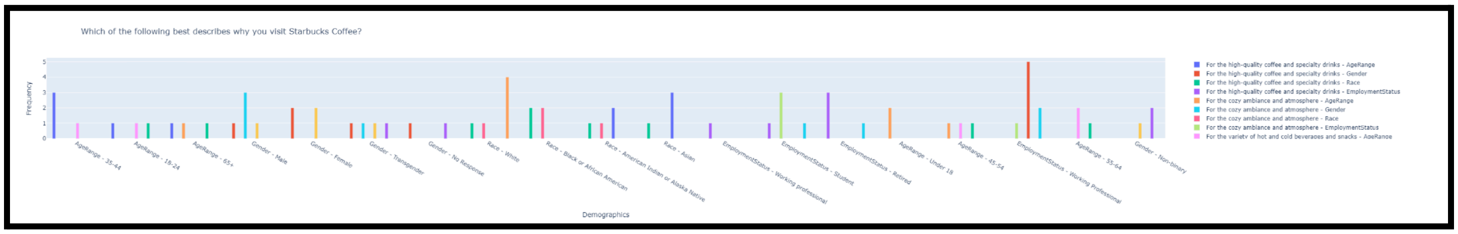
What is your preferred method of purchasing Nike shoes?

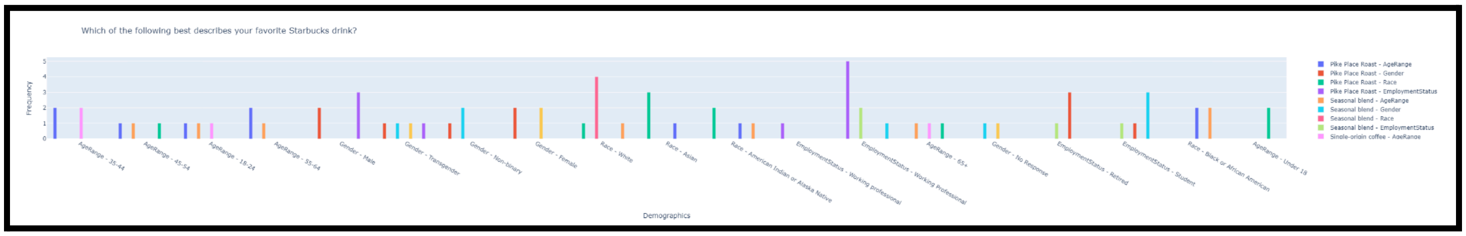
- ○ In-store
- ○ Online
- ○ Both
- ○ I do not purchase Nike shoes

Which of the following best describes your experience with our product?

- ○ Exceptional - I love the comfort and style of my Nike shoes.
- ○ Satisfactory - My Nike shoes are okay, but there's room for improvement.
- ○ Dissatisfactory - I've had issues with the quality or fit of my Nike shoes.
- ○ Haven't tried - I have not yet purchased or worn Nike shoes.

- **Analysis Dashboard-** View bar charts for response frequency and demographic analysis for each question





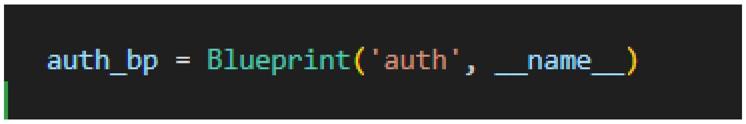
The web application is developed with a clear separation of concerns to ensure codebase manageability and maintainability. The user interfaces were built using HTML, Bootstrap CSS, jQuery, and JavaScript to handle the presentation logic, while Flask, a lightweight and flexible microframework, is used for server-side logic and data processing. This division of responsibilities facilitated a more structured and organized development process. Using Bootstrap, I was able to achieve a responsive and mobile-friendly design, ensuring optimal user experience across different devices and screen sizes. Additionally, Flask's seamless integration with these front-end technologies provided a versatile and efficient development environment. By harnessing Flask's powerful routing and templating capabilities, I was able to create dynamic and interactive web pages, offering a rich user experience while benefiting from the simplicity and adaptability of Flask. The successful integration of front-end technologies with Flask's back-end functionality resulted in a cohesive and effective web application, promising broader applicability in various projects.

Back End Design

The back-end component of the web application plays a crucial role in handling client requests, processing data, and interacting with the database. I have chosen to use Python as the primary programming language due to its versatility, extensive libraries, and robust frameworks available. In particular, I have used the Flask framework, a lightweight and flexible web framework, to build the back-end functionality of the application.

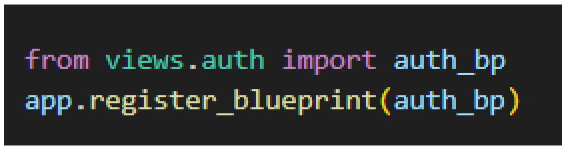
I. Blueprints

In the development of the Flask application, I have used the concept of "blueprints" to architecturally organize and enhance the scalability and maintainability of the web application. Blueprints enabled me to divide the application into smaller, self-contained components, each encapsulating a specific set of routes and views. By doing so, a more modular and structured approach was achieved to building the web application. Each blueprint represents a distinct functionality or feature of the application, such as authentication, data analysis, and company-related operations. Through blueprint registration, we have seamlessly integrated these components with the main Flask application, allowing for efficient and logical routing of incoming requests. Moreover, blueprints have significantly streamlined the development process, making it easier to manage and modify specific sections of the application without affecting others. With the Flask application acting as the central orchestrator, the use of blueprints has proven instrumental in creating a well-organized and maintainable web application architecture.



```
auth_bp = Blueprint('auth', __name__)
```

Creating Authentication Blueprint



```
from views.auth import auth_bp
app.register_blueprint(auth_bp)
```

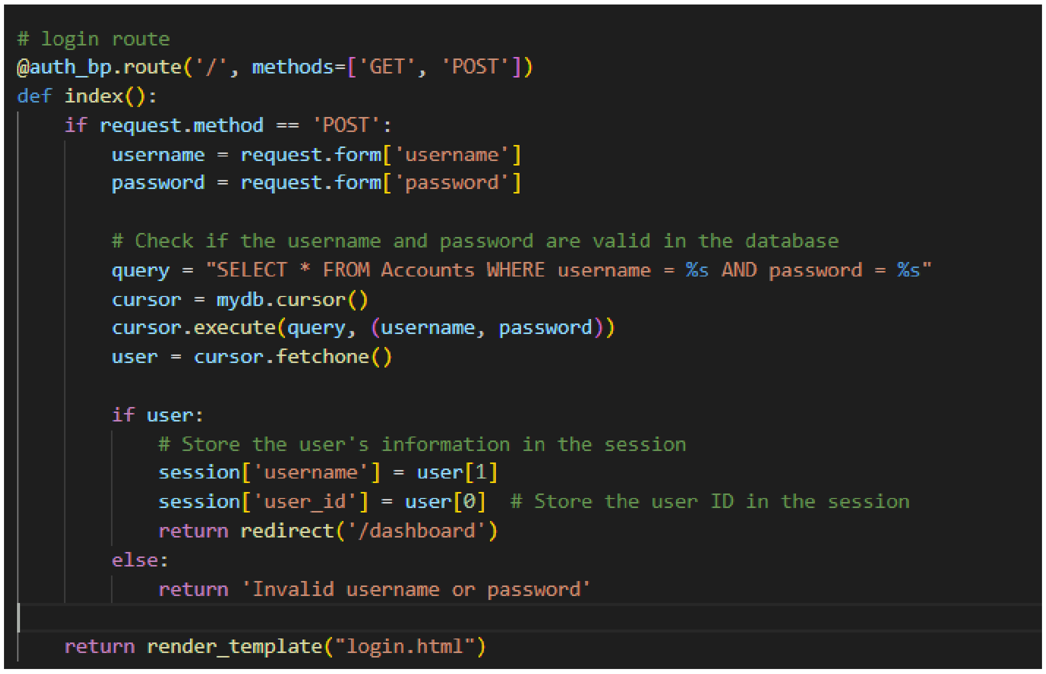
Importing and registering the Blueprint

II. Authentication and Authorization

Within the Flask web application, some basic authentication and authorization mechanisms have been implemented. The system uses the accounts table in the SQL to store user credentials and sessions for managing user states effectively. Since this application is a prototype, extensive focus was not given on data sanitization and this can be further improved in future versions of the application. The code is structured as a Flask Blueprint named auth_bp to manage related routes and logic. Here are some key functionalities:

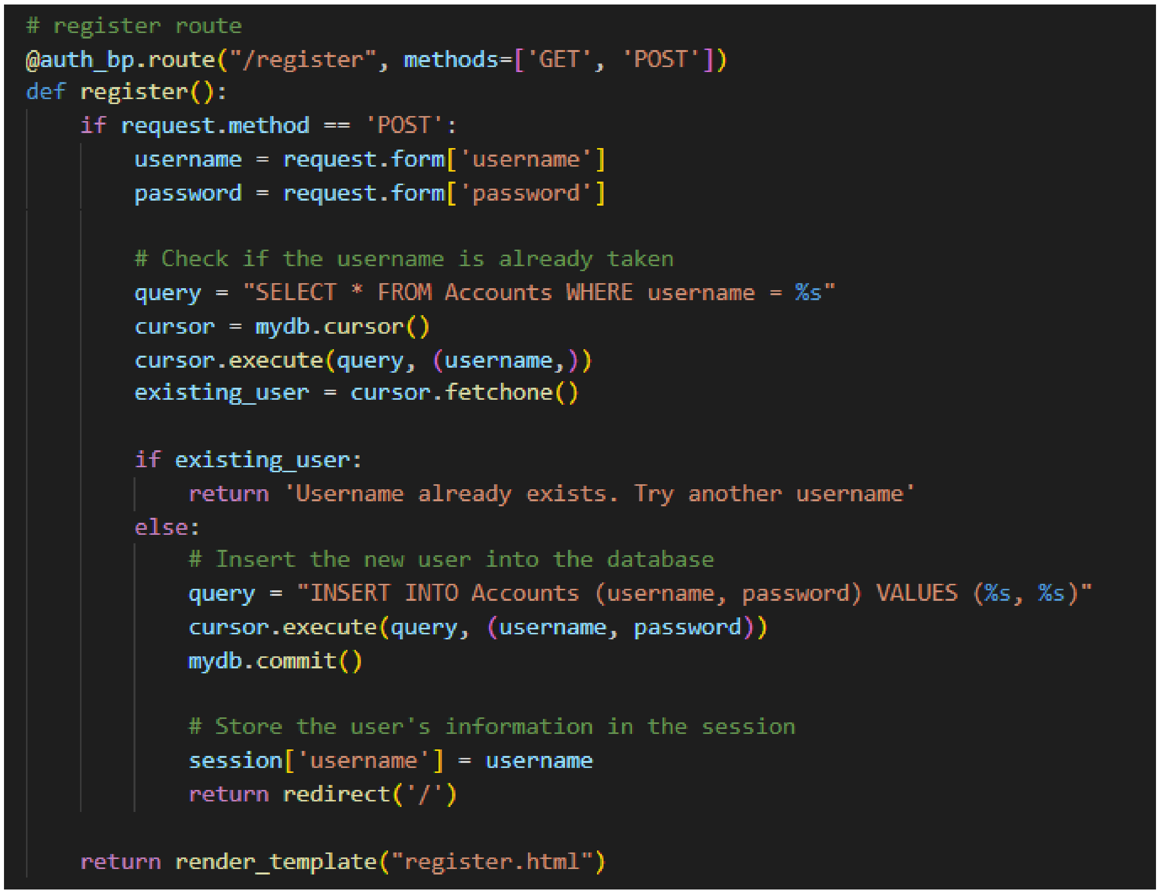
1. User Authentication

- **Login**
  - Captures username and password from an HTML form.
  - Validates credentials against data in the Accounts table within a connected database.
  - Upon successful authentication, stores user information in the user's session and redirects to the Company dashboard.
  - Returns an error message if authentication fails.



```python
# login route
@auth_bp.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Check if the username and password are valid in the database
        query = "SELECT * FROM Accounts WHERE username = %s AND password = %s"
        cursor = mydb.cursor()
        cursor.execute(query, (username, password))
        user = cursor.fetchone()

        if user:
            # Store the user's information in the session
            session['username'] = user[1]
            session['user_id'] = user[0]  # Store the user ID in the session
            return redirect('/dashboard')
        else:
            return 'Invalid username or password'

    return render_template("login.html")
```

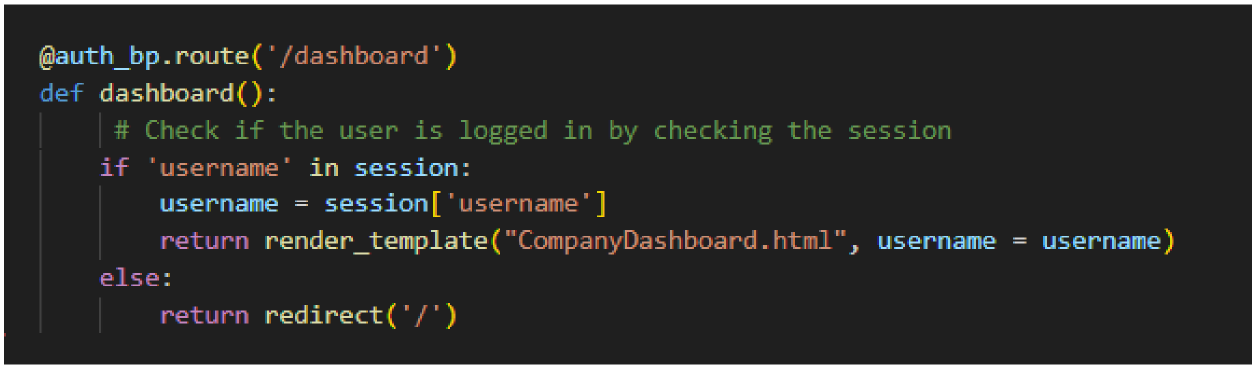
- **Registration**
  - Accepts new user registration data.
  - Checks for duplicate usernames and throws an error message if the system finds any duplicates
  - Inserts new user data into the Accounts table.
  - Stores user information in the session and redirects to the login page.



```python
# register route
@auth_bp.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Check if the username is already taken
        query = "SELECT * FROM Accounts WHERE username = %s"
        cursor = mydb.cursor()
        cursor.execute(query, (username,))
        existing_user = cursor.fetchone()

        if existing_user:
            return 'Username already exists. Try another username'
        else:
            # Insert the new user into the database
            query = "INSERT INTO Accounts (username, password) VALUES (%s, %s)"
            cursor.execute(query, (username, password))
            mydb.commit()

            # Store the user's information in the session
            session['username'] = username
            return redirect('/')

    return render_template("register.html")
```

2. **User Authorization**

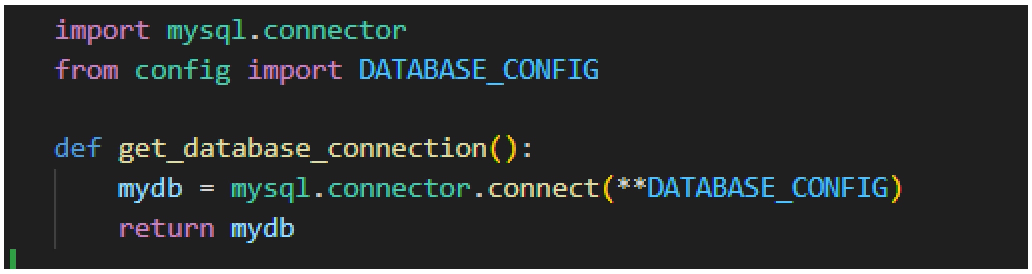
- Protects the dashboard route, allowing access only to authenticated users.
- Validates the presence of a username in the session before rendering the dashboard template.



```python
@auth_bp.route('/dashboard')
def dashboard():
    # Check if the user is logged in by checking the session
    if 'username' in session:
        username = session['username']
        return render_template("CompanyDashboard.html", username = username)
    else:
        return redirect('/')
```

3. **Database Interactions**

- The system establishes a database connection using a get_database_connection() function.



```python
import mysql.connector
from config import DATABASE_CONFIG

def get_database_connection():
    mydb = mysql.connector.connect(**DATABASE_CONFIG)
    return mydb
```

- Employs SQL queries for user authentication, registration, and data retrieval.
- Commits changes to the database upon user registration using the commit method

III. Session Management

Session management plays an important role in ensuring the secure and efficient handling of user interactions within web applications. For this application, Flask's session management capabilities are used to maintain user state across multiple HTTP requests seamlessly. Through the use of session variables, such as storing user information upon successful authentication and maintaining prompts and questions during the survey creation process, the application ensures a personalized and continuous user experience. Furthermore, session management facilitates secure data exchange between different routes within the application, ensuring that sensitive information, such as user credentials and survey responses, is transmitted securely without exposing it in the URL or request body. This robust session management mechanism enhances the overall usability and security of the web application, contributing to a smooth and reliable user experience.

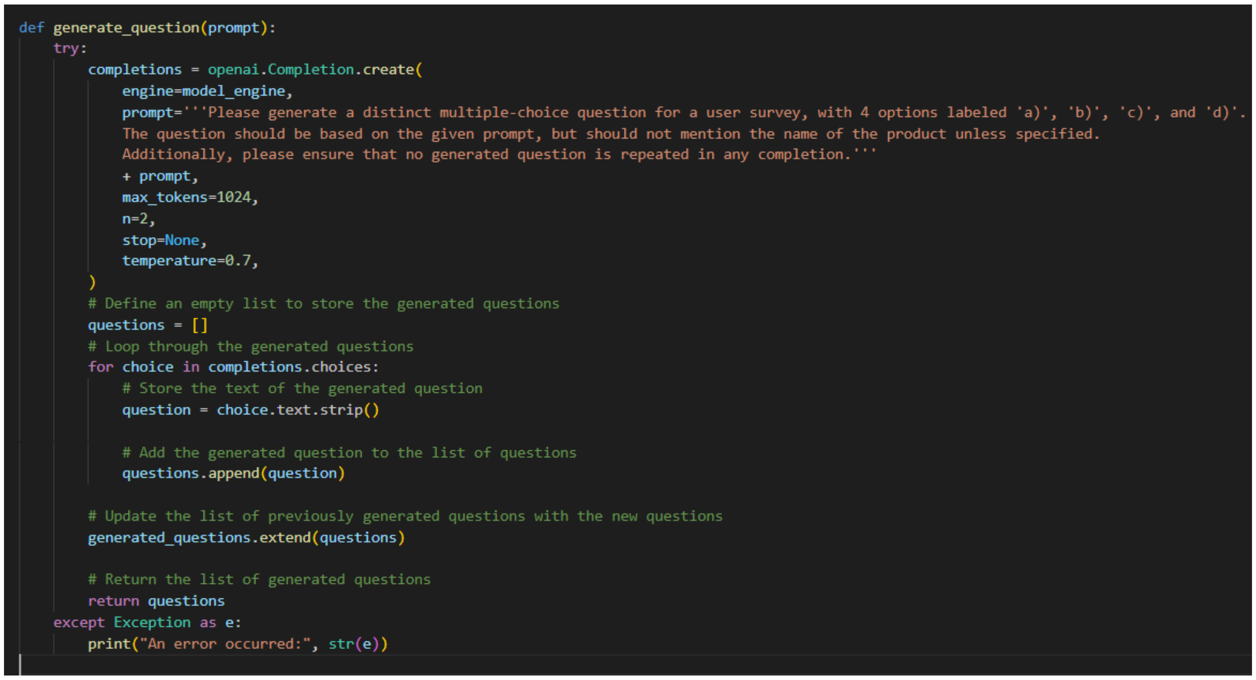
IV. Web Application Structure

The web application follows the Model-View-Controller (MVC) architectural pattern, which promotes a clear separation of concerns and improves code maintainability. The Flask framework inherently supports this pattern, making it easier to organize and manage our application's components.

- **Models:** We utilize a MySQL database to store and manage the application's data. We have employed the mysql.connector library to establish a connection with the database. This library allows us to interact with the MySQL server, execute SQL queries, and retrieve or modify data.
- **Views:** The views in the application are implemented using Flask's routing mechanism. We define multiple routes that correspond to different functionalities, such as rendering HTML templates, handling form submissions, and redirecting users to appropriate pages. The system utilizes Flask's render_template function to dynamically generate HTML pages and present them to the user.
- **Controllers:** The controllers handle the business logic of our application. They are responsible for processing client requests, validating data, interacting with the database, and generating appropriate responses. The system has several controller functions that map to specific routes and perform the necessary operations.

V. Handling Client Requests and Data Processing

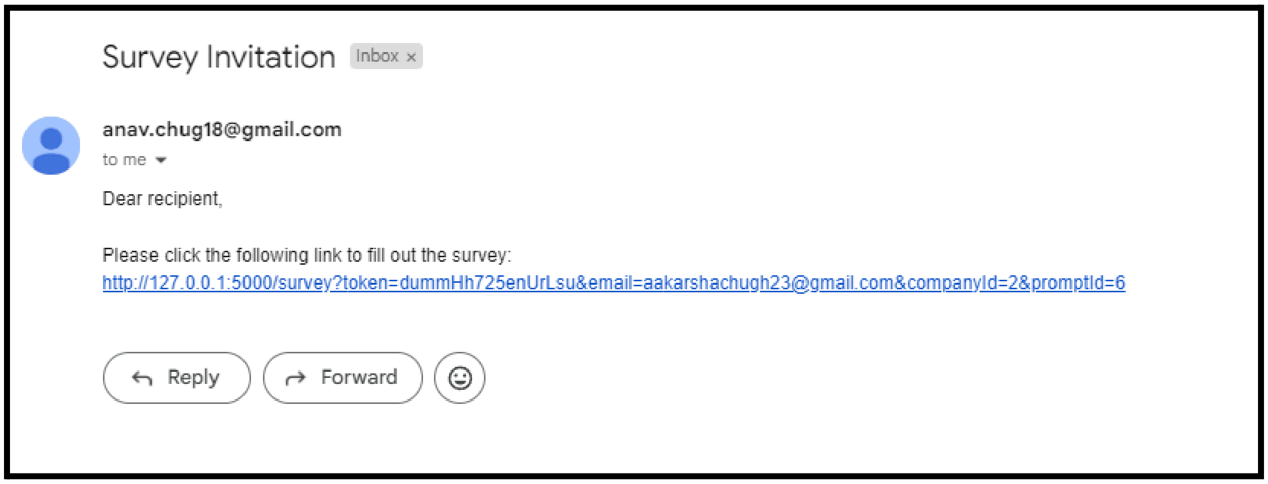
When a client sends a request to the web application, Flask's routing mechanism directs the request to the appropriate controller function based on the defined routes. For example, the "/" route corresponds to the index() function, which handles the company information form submission. Upon receiving the form data, the system extracts the relevant information and stores it in the MySQL database. The system then uses SQL queries executed through the mysql.connector library to insert the data into the corresponding tables. The system integrates OpenAI's gpt-3.5-turbo-instruct language model into the application to generate survey questions dynamically. GPT-3.5 Turbo models can understand and generate natural language or code and have been optimized for chat using the Chat Completions API but work well for non-chat tasks as well [7]. After storing the company's prompt in the database, the system uses the generate_question() function to generate distinct multiple-choice questions based on the prompt. The generated questions are then formatted and presented to the user for further processing.



```python
def generate_question(prompt):
    try:
        completions = openai.Completion.create(
            engine=model_engine,
            prompt='''Please generate a distinct multiple-choice question for a user survey, with 4 options labeled 'a)', 'b)', 'c)', and 'd)'.
            The question should be based on the given prompt, but should not mention the name of the product unless specified.
            Additionally, please ensure that no generated question is repeated in any completion.'''
            + prompt,
            max_tokens=1024,
            n=2,
            stop=None,
            temperature=0.7,
        )
        # Define an empty list to store the generated questions
        questions = []
        # Loop through the generated questions
        for choice in completions.choices:
            # Store the text of the generated question
            question = choice.text.strip()

            # Add the generated question to the list of questions
            questions.append(question)

        # Update the list of previously generated questions with the new questions
        generated_questions.extend(questions)

        # Return the list of generated questions
        return questions
    except Exception as e:
        print("An error occurred:", str(e))
```

VI. Sending Email Invitations

The system automates sending survey invitations via email after a company has finalized a survey. This application assumes we already have a list of targeted customers from the companies. In the system, we are using a hardcoded email list which would be used for all companies. This is just for demonstration purposes for this prototype and can be improved by allowing companies to provide an email list in the UI when they finalize a survey. The idea is to essentially improve the target audience for the companies that can allow them to do better market research.

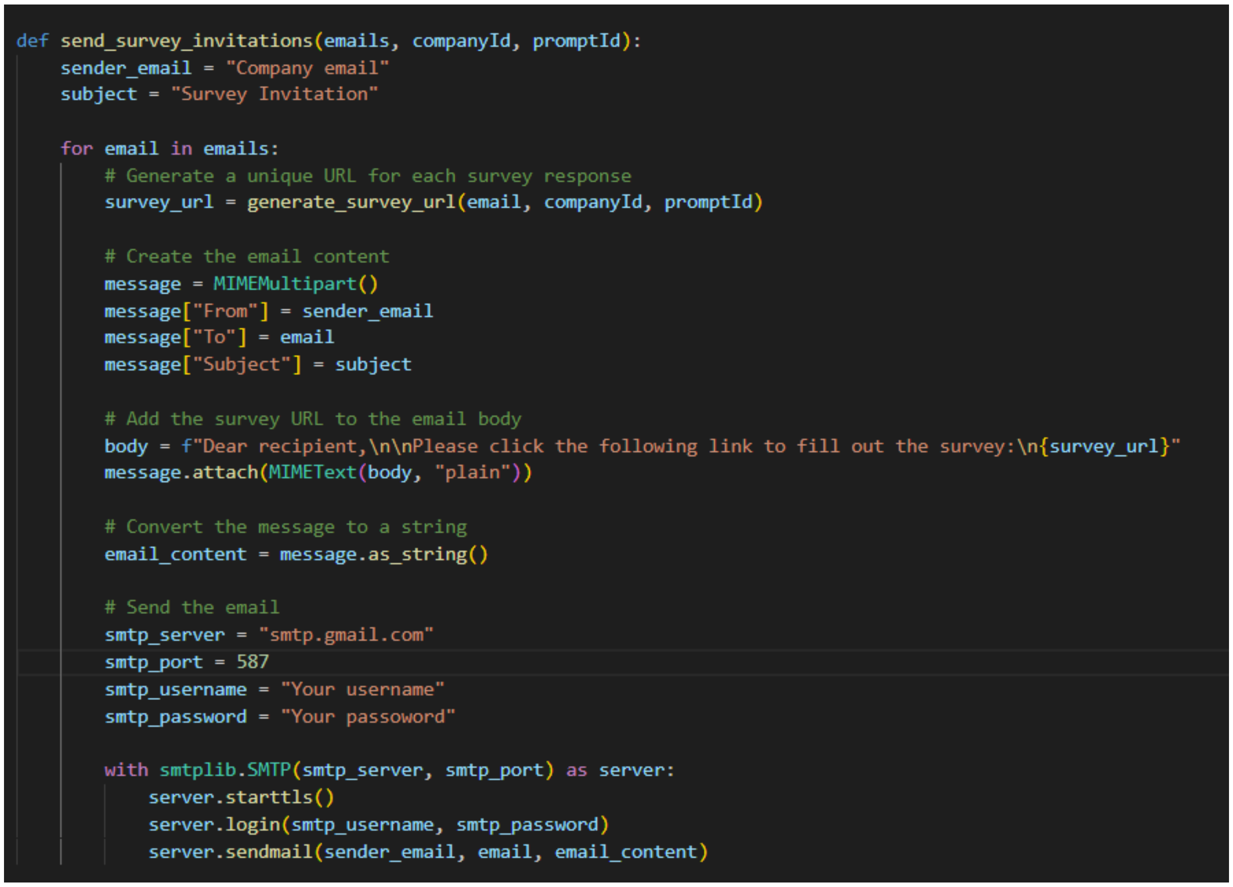

Survey Invitation

anav.chug18@gmail.com
to me

Dear recipient,

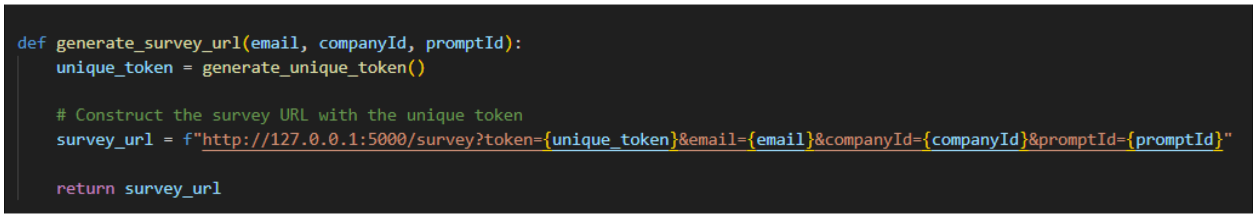
Please click the following link to fill out the survey:
http://127.0.0.1:5000/survey?token=dummHh725enUrLsu&email=aakarshachugh23@gmail.com&companyId=2&promptId=6

The system uses Python libraries for email creation (email.mime.multipart) , sending emails (SMTP), and secure random number generation (secrets). The send_survey_invitations function iterates through recipient email and for each recipient, it generates a unique token using the generate_survey_url function. This ensures that we have a unique URL for each survey response. This function constructs a URL specific to the recipient's survey response by combining a base URL with a unique token, recipient email, company ID, and prompt ID. The send_survey_invitations function then builds the email content with the survey URL and sends the email using an SMTP server.



```
def send_survey_invitations(emails, companyId, promptId):
    sender_email = "Company email"
    subject = "Survey Invitation"

    for email in emails:
        # Generate a unique URL for each survey response
        survey_url = generate_survey_url(email, companyId, promptId)

        # Create the email content
        message = MIMEMultipart()
        message["From"] = sender_email
        message["To"] = email
        message["Subject"] = subject

        # Add the survey URL to the email body
        body = f"Dear recipient,\n\nPlease click the following link to fill out the survey:\n{survey_url}"
        message.attach(MIMEText(body, "plain"))

        # Convert the message to a string
        email_content = message.as_string()

        # Send the email
        smtp_server = "smtp.gmail.com"
        smtp_port = 587
        smtp_username = "Your username"
        smtp_password = "Your passoword"

        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(smtp_username, smtp_password)
            server.sendmail(sender_email, email, email_content)
```

Function to send survey invitations



```
def generate_survey_url(email, companyId, promptId):
    unique_token = generate_unique_token()

    # Construct the survey URL with the unique token
    survey_url = f"http://127.0.0.1:5000/survey?token={unique_token}&email={email}&companyId={companyId}&promptId={promptId}"

    return survey_url
```

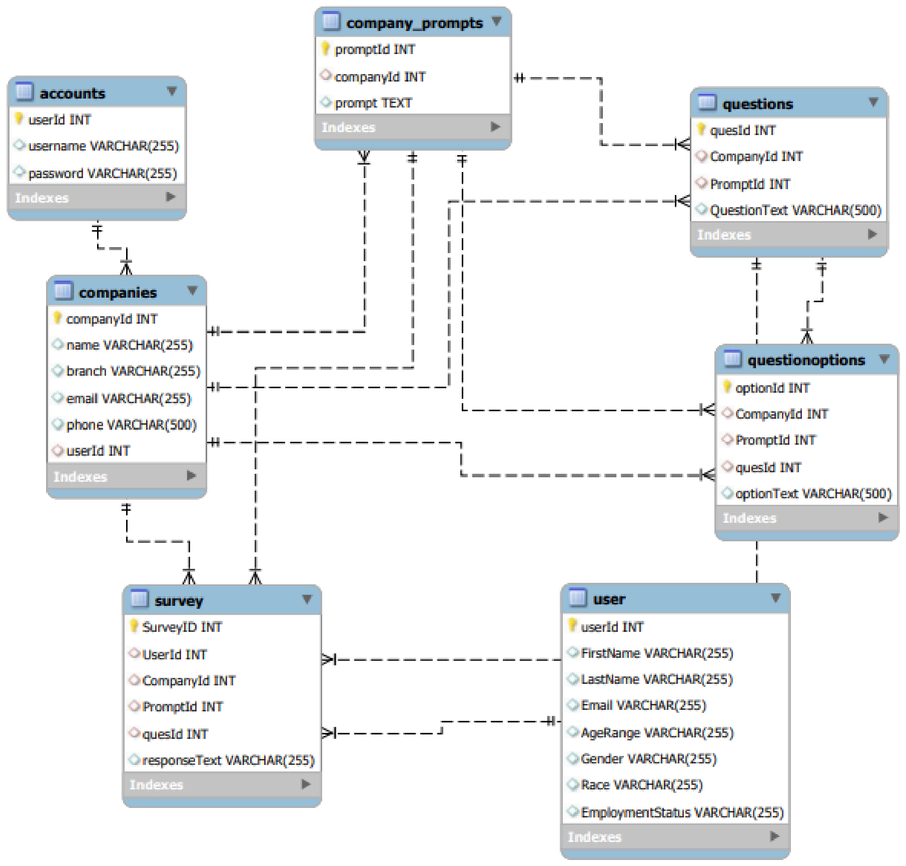
Function to generate unique survey urls

Database Design

The application utilizes the MySQL database management system. MS SQL Server was used to write queries and perform database related operations. The database design allows for the collection, storage, and analysis of survey data from multiple companies and users. By establishing relationships between the various entities, it enables meaningful insights into the survey responses provided by users affiliated with different companies. The design offers a flexible and scalable structure to accommodate future enhancements or modifications to meet evolving research requirements.

It relies on the following tables within the database:

- **accounts**- This table stores information about company usernames and passwords
- **company**: This table stores information about companies including name, branch, email, and phone number. Each company has a unique companyId.
- **user**: This table stores user details like first name, last name, email and demographic information like age range, gender, race, and employment status. Each user has a unique userId.
- **company_prompts**: This table stores prompts associated with each company. A company can have multiple prompts. Each prompt has a unique promptId and uses the companyId as a foreign key.
- **questions**: This table stores the questions for the surveys that a company creates for a given prompt. Each question has a unique questionId and uses companyId and promptId as foreign keys.
- **questionoptions**- This table stores the multiple choice options of a question that belongs to a given survey prompt. Each option has a unique optionId and uses companyId, promptId, and quesId as foreign keys.
- **survey**: This table stores survey responses from users for each survey prompt of a given company. Each survey has a unique surveyId and uses companyId, promptId, quesId, and userId as foreign keys.



EER Diagram

Database Interaction with Backend

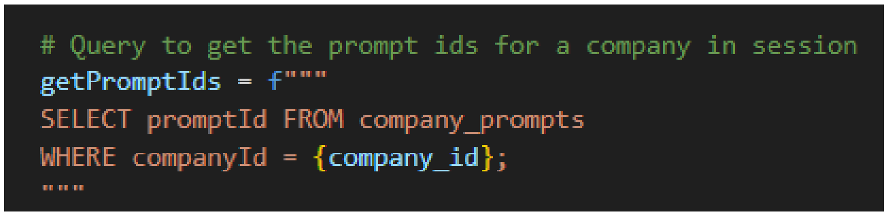
The application interacts with the MySQL database using the mysql.connector library. We establish a connection to the database by providing the necessary credentials, including the host, username, password, and database name. Once connected, we utilize SQL queries to insert, retrieve, or modify data. To ensure data integrity and prevent SQL injection attacks, I employ parameterized queries and sanitize user input before executing any SQL statements. This practice mitigates security risks and enhances the overall robustness of our application. This design approach provides a scalable and maintainable foundation for our web application's back-end functionality.

Survey Analysis

The application also includes a powerful demographic analysis feature that provides valuable insights to companies about their survey responses. This feature utilizes the open-source Python graphing library, Plotly, to create interactive and visually appealing bar charts.

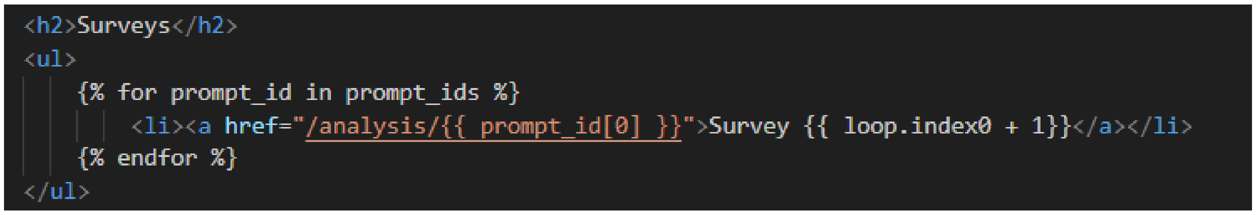
Companies have access to a comprehensive analysis dashboard enabling them to delve into demographic insights for each of their surveys. Each survey prompt is encapsulated within its own analysis page, ensuring clarity and ease of navigation for users. The dashboard focuses on four pivotal demographic features: Age Range, Gender, Race, and Employment Status, providing a better understanding of survey responses. The primary objective of this analysis is to enable companies to gain profound insights into their audience demographics, thereby enhancing their market research capabilities. By identifying and targeting previously untapped user segments, companies can strategically refine their marketing strategies, thereby increasing product popularity and sales potential. This demographic breakdown helps identify any variations in opinions and preferences among different segments of the audience.

To implement this dashboard, the first step was to separate the dashboard into two routes- /analysis and /analysis/<int:prompt_id>. The /analysis route simply displays links to the surveys created by a company. This is done by executing a simple SQL query to get all the promptIds for the company in session.



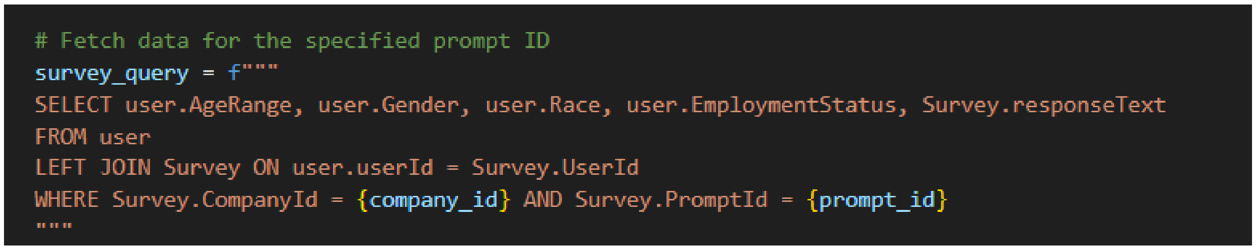
```
# Query to get the prompt ids for a company in session
getPromptIds = f"""
SELECT promptId FROM company_prompts
WHERE companyId = {company_id};
"""
```

We then loop through these promptIds on the front end to generate links for those prompts.



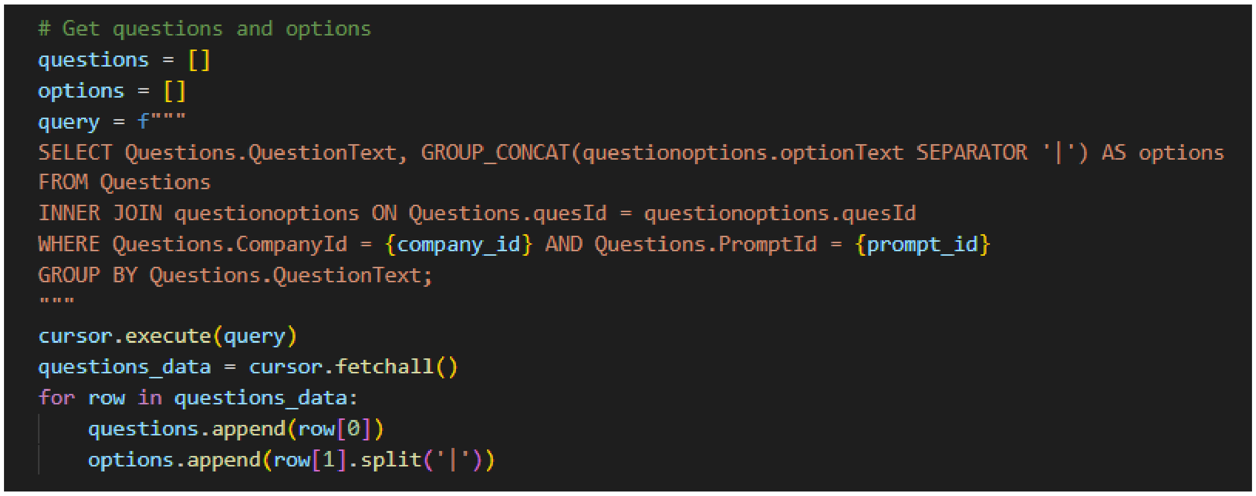
```
<h2>Surveys</h2>
<ul>
    {% for prompt_id in prompt_ids %}
        <li><a href="/analysis/{{ prompt_id[0] }}">Survey {{ loop.index0 + 1}}</a></li>
    {% endfor %}
</ul>
```

To get the user responses, we execute another SQL query and join the user and survey tables on the userId by performing a LEFT JOIN. We are selecting the demographic information and user responses for the company in session and the prompt link that is clicked on.



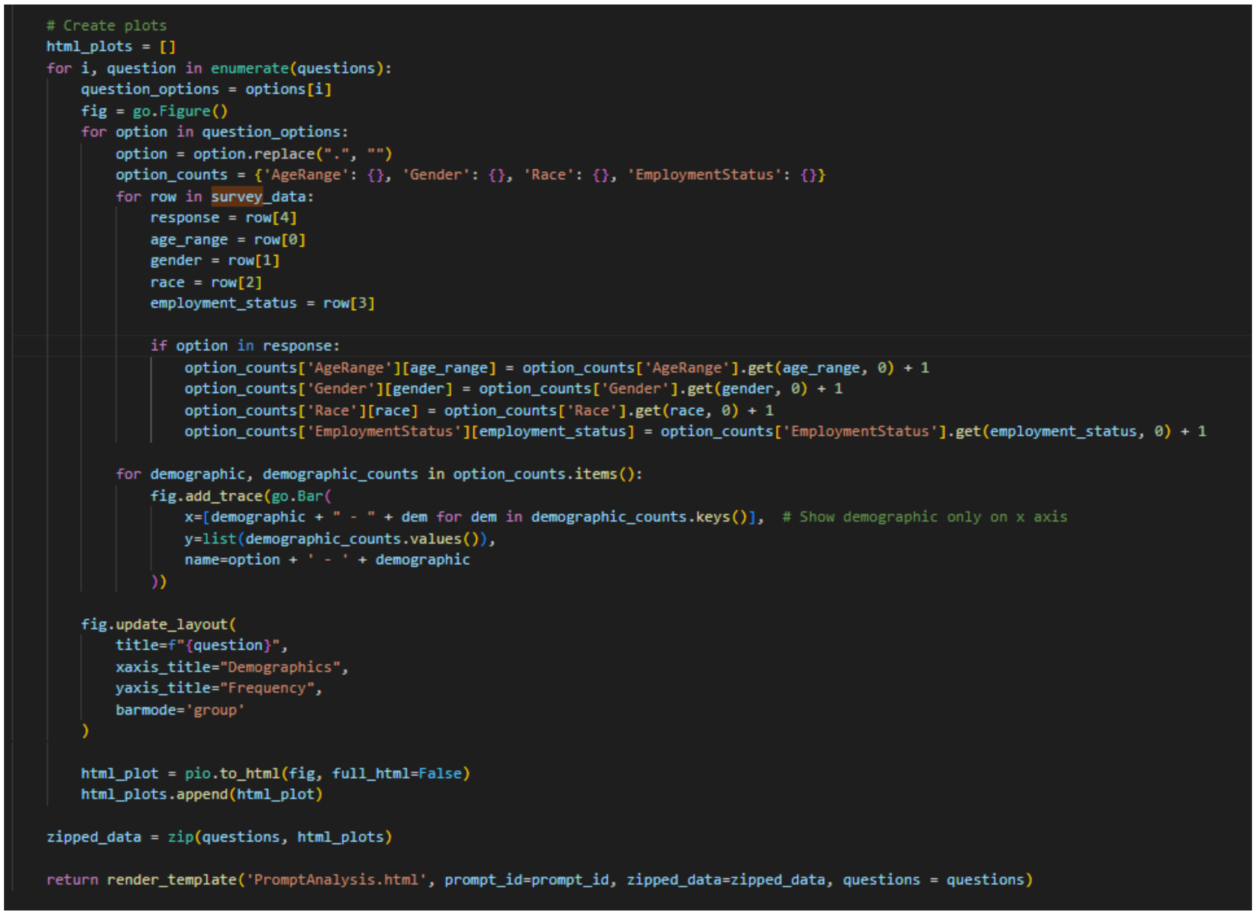
```
# Fetch data for the specified prompt ID
survey_query = f"""
SELECT user.AgeRange, user.Gender, user.Race, user.EmploymentStatus, Survey.responseText
FROM user
LEFT JOIN Survey ON user.userId = Survey.UserId
WHERE Survey.CompanyId = {company_id} AND Survey.PromptId = {prompt_id}
"""
```

The question and options data in the Analysis dashboard is obtained by joining the questions and questionoptions tables on the quesId by performing an INNER JOIN. This data is simply appended in two different python lists.



```
# Get questions and options
questions = []
options = []
query = f"""
SELECT Questions.QuestionText, GROUP_CONCAT(questionoptions.optionText SEPARATOR '|') AS options
FROM Questions
INNER JOIN questionoptions ON Questions.quesId = questionoptions.quesId
WHERE Questions.CompanyId = {company_id} AND Questions.PromptId = {prompt_id}
GROUP BY Questions.QuestionText;
"""
cursor.execute(query)
questions_data = cursor.fetchall()
for row in questions_data:
    questions.append(row[0])
    options.append(row[1].split('|'))
```

The system then iterates through each question retrieved from the database and for each option within a question, it calculates counts for different demographic groups (age range, gender, race, and employment status) based on the survey responses. The Plotly graph objects (go) used in the code are instrumental in creating interactive and visually appealing visualizations for data analysis. With Plotly, the code constructs bar plots for each option within a question, showcasing the frequency of responses across different demographic groups. Finally, it converts each plot to HTML format which is then rendered in an HTML template. Plotly's flexibility and rich features enable the creation of customizable plots with ease, empowering users to explore and interpret the data effectively within the context of demographic characteristics.



```python
# Create plots
html_plots = []
for i, question in enumerate(questions):
    question_options = options[i]
    fig = go.Figure()
    for option in question_options:
        option = option.replace(".", "")
        option_counts = {'AgeRange': {}, 'Gender': {}, 'Race': {}, 'EmploymentStatus': {}}
        for row in survey_data:
            response = row[4]
            age_range = row[0]
            gender = row[1]
            race = row[2]
            employment_status = row[3]

            if option in response:
                option_counts['AgeRange'][age_range] = option_counts['AgeRange'].get(age_range, 0) + 1
                option_counts['Gender'][gender] = option_counts['Gender'].get(gender, 0) + 1
                option_counts['Race'][race] = option_counts['Race'].get(race, 0) + 1
                option_counts['EmploymentStatus'][employment_status] = option_counts['EmploymentStatus'].get(employment_status, 0) + 1

        for demographic, demographic_counts in option_counts.items():
            fig.add_trace(go.Bar(
                x=[demographic + " - " + dem for dem in demographic_counts.keys()],  # Show demographic only on x axis
                y=list(demographic_counts.values()),
                name=option + ' - ' + demographic
            ))

    fig.update_layout(
        title=f"{question}",
        xaxis_title="Demographics",
        yaxis_title="Frequency",
        barmode='group'
    )

    html_plot = pio.to_html(fig, full_html=False)
    html_plots.append(html_plot)

zipped_data = zip(questions, html_plots)

return render_template('PromptAnalysis.html', prompt_id=prompt_id, zipped_data=zipped_data, questions = questions)
```

Conclusion

This paper proposed a web-based application and the results of this project showcase a more efficient and effective method for companies to generate survey questionnaires for market research and also get a real-time demographic analysis. By identifying and targeting previously untapped user segments, companies can strategically refine their marketing strategies, thereby increasing product popularity and sales potential. The future iterations of this project can be further improved by adding more features to the UI, increasing the interactions with the API by trying different types of prompts, and also generating questions other than just multiple choice questions. Incorporating NLP models into applications like this for market research will allow companies to reduce manual work of creating surveys and allow researchers to find relevant focus groups to target their product at a much faster pace.

References

[1] Chowdhary, KR1442, and K. R. Chowdhary. "Natural language processing." *Fundamentals of artificial intelligence* (2020): 603-649.

[2] Colchester, K., Hagras, H., Alghazzawi, D. and Aldabbagh, G. 2017. A survey of artificial intelligence techniques employed for adaptive educational systems within E-learning platforms. Journal of Artificial Intelligence and Soft Computing Research, 7(1): 47–64.

[3] Hou, Jiang-Liang, and Yuh-Zong Chu. "Automatic questionnaire survey by using the collective message over the Internet." *Advanced Engineering Informatics* 29.4 (2015): 813-829.

[4] Blšták, Miroslav, and Viera Rozinajová. "Automatic question generation based on sentence structure analysis using machine learning approach." *Natural Language Engineering* 28.4 (2022): 487-517.

[5] Du, Xinya, Junru Shao, and Claire Cardie. "Learning to ask: Neural question generation for reading comprehension." *arXiv preprint arXiv:1705.00106* (2017).

[6] Floridi, Luciano, and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences." *Minds and Machines* 30 (2020): 681-694.

[7] “Models.” *OpenAI API*, https://platform.openai.com/docs/models/gpt-3-5-turbo