**Part 3 - Detailed Design**

**Architecture-**

The architecture best suits our project is **MVC**.

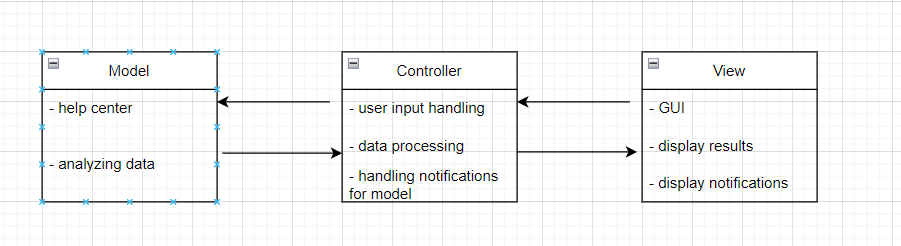
**MVC Architecture in our project:**

* **Model**:
  + Manages data, trained ML data, and other relevant information.
  + Handles data retrieval, storage, and processing.
* **View**:
  + GUI for upload data.
  + Notifications in real-time on survivor interactions.
  + Graphics for analyzing data.
* **Controller:**
  + Acts as a bridge between the Model and the View. Connect view to backend in real-time.
  + Handles user input, processes it, and updates the Model and View accordingly.

**Data Storage:**

* WhatsApp call transcripts of the aid center are stored in CSV.
* Trained ML models will be stored in files.
* Other relevant data is stored in separate files.

**Graphic Description**



**data description:**

* **Calls 1.8.23 to 19.9.23 with classification list** - a CSV file that has the following fields:
  + Conversation id
  + brandId
  + startTimeL
  + startTime
  + startTimeUTC
  + startTimeLOCAL
  + startTimeDate
  + startTimeYear
  + startTimeMonth
  + startTimeMonthStr
  + startTimeDay
  + startTimeWeekday
  + startTimeWeekdayStr
  + startTimeTimestamp
  + startTimeHour
  + startTimeMinute
  + startTimeWeekSun
  + startTimeWeekMon
  + endTimeL
  + endTime
  + endTimeUTC
  + endTimeLOCAL
  + endTimeDate
  + endTimeYear
  + endTimeMonth
  + endTimeMonthStr
  + endTimeDay
  + endTimeWeekday
  + endTimeWeekdayStr
  + endTimeTimestamp
  + endTimeHour
  + endTimeMinute
  + endTimeWeekSun
  + endTimeWeekMon
  + Duration
  + summaryText
  + summaryTimeUTC
  + closeReason
  + closeReasonDescription
  + device source browser
  + operatingSystem
  + Status
  + firstConversation
  + Interactive
  + isPartial
  + latestAgentGroupId
  + latestQueueState
  + latestSkillId
  + userType
  + agentParticipantsCount
  + agentParticipantsTimeUTC
  + agentParticipantsTimeL
  + agentParticipantsDeleted
  + agentParticipantsUserType
  + agentParticipantsUserTypeName
  + consumerParticipantsCount
  + consumerParticipantsTimeUTC
  + consumerParticipantsTimeL
  + transfersCount
  + transfersTimeUTC
  + transfersTimeL
  + interactionsAgentCount
  + interactionsAgentTimeUTC
  + interactionsAgentTimeL
  + messageCount
  + messageTime
  + responseTime
  + responseTimeAssignment
  + responseCount
  + messageCountAgent
  + messageCountAgentHuman
  + messageCountAgentSystem
  + messageCountAgentBot
  + messageCountConsumer
  + messageTimeAgent
  + messageTimeAgentHuman
  + messageTimeAgentSystem
  + messageTimeAgentBot
  + messageTimeConsumer
  + averageMessageTime
  + averageMessageTimeAgent
  + averageMessageTimeAgentHuman
  + averageMessageTimeAgentSystem
  + averageMessageTimeAgentBot
  + averageMessageTimeConsumer
  + responseCountAgent
  + responseCountAgentHuman
  + responseCountAgentSystem
  + responseCountAgentBot
  + responseCountConsumer
  + responseTimeAgent
  + responseTimeAgentHuman
  + responseTimeAgentSystem
  + responseTimeAgentBot
  + responseTimeAssignmentAgent
  + responseTimeAssignmentAgentHuman
  + responseTimeAssignmentAgentSystem
  + responseTimeAssignmentAgentBot
  + responseTimeConsumer
  + responseTimeAssignmentConsumer
  + averageResponseTime
  + averageResponseTimeAssignment
  + averageResponseTimeAgent
  + averageResponseTimeAgentHuman
  + averageResponseTimeAgentSystem
  + averageResponseTimeAgentBot
  + averageResponseTimeAssignmentAgent
  + averageResponseTimeAssignmentAgentHuman
  + averageResponseTimeAssignmentAgentSystem
  + averageResponseTimeAssignmentAgentBot
  + averageResponseTimeConsumer
  + averageResponseTimeAssignmentConsumer
  + firstRespondent
  + firstResponseTimeAgentFromConsumer
  + firstResponseTimeAgentHumanFromConsumer
  + firstResponseTimeAgentSystemFromConsumer
  + firstResponseTimeAgentBotFromConsumer
  + firstResponseTimeConsumerFromAgent
  + firstResponseTimeAgentFromAssignment
  + firstResponseTimeAgentHumanFromAssignment
  + firstResponseTimeAgentSystemFromAssignment
  + firstResponseTimeAgentBotFromAssignment
  + firstResponseTimeAgentFromAssignmentCount
  + firstResponseTimeAgentHumanFromAssignmentCount
  + firstResponseTimeAgentSystemFromAssignmentCount
  + firstResponseTimeAgentBotFromAssignmentCount
  + averageFirstResponseTimeAgentFromAssignment
  + averageFirstResponseTimeAgentHumanFromAssignment
  + averageFirstResponseTimeAgentSystemFromAssignment
  + averageFirstResponseTimeAgentBotFromAssignment
  + firstResponseTimeAgentFromFirstAssignment
  + firstResponseTimeAgentHumanFromFirstAssignment
  + firstResponseTimeAgentSystemFromFirstAssignment
  + firstResponseTimeAgentBotFromFirstAssignment
  + firstAssignmentTimeAgentFromStart
  + firstAssignmentTimeAgentHumanFromStart
  + firstAssignmentTimeAgentSystemFromStart
  + firstAssignmentTimeAgentBotFromStart
  + words
  + wordsAgent
  + wordsAgentHuman
  + wordsAgentSystem
  + wordsAgentBot
  + wordsConsumer
  + questions
  + questionsAgent
  + questionsAgentHuman
  + questionsAgentSystem
  + questionsAgentBot
  + questionsConsumer
  + transcriptAll
  + transcriptAgent
  + transcriptConsumer
  + classification
  + classification 2 despair
  + loneliness
  + emotional overflow
  + self blame
  + anxiety distrust / confusion
  + new assault / new exposure
  + level of suicide/ level of risk
  + obligation to report occording law
  + support for support circuls
  + additional notes
* **Calls 1.8.23 to 19.9.23 without classification list** - a CSV file that has not classification of the calls.

Contains the same fields without classification fields.

**API specification:**

User interface:

* Upload CSV of Calls 1.8.23 to 19.9.23 with classification list.
* choose a parameter for the graph generator - for one parameter graphs.
* Show a graph with one parameter.
* Save graphs as jpeg.
* Prediction results of the ML model display.
* Connect to whatsapp web via system.

**interface design:**

Notifications

APP

DATA-

CSV file

output - graphs + prediction results

ML algorithms

statistics algorithms

user

**Programming Languages and Tools:**

1. **Python:**
   * Description: Python was chosen as the primary programming language for its versatility, ease of use, and extensive libraries for data analysis and machine learning.
   * Purpose: Python will be used for data preprocessing, statistical analysis, machine learning model development, and integration with various libraries and frameworks.
   * **Libraries:**
     + Matplotlib: For generating statistical graphs and visualizations.
     + Pandas: For data manipulation, analysis, and handling structured data.
     + NumPy: For numerical computing, array operations, and mathematical functions.
     + Scikit-learn (SKlearn): For implementing machine learning algorithms and model training.
2. **PostgreSQL:**
   * Description: PostgreSQL is chosen as the relational database management system (RDBMS) for its robustness, scalability, and support for complex queries and transactions.
   * Purpose: PostgreSQL will be used for storing and managing structured data related to the project, such as user information, call transcripts, and analysis results.
3. **Flask:**
   * Description: Flask is a lightweight web framework for Python that provides tools and utilities for building web applications.
   * Purpose: Flask will be used as the server-side framework for handling HTTP requests, routing, and serving web pages and APIs.
4. **JavaScript and Bootstrap:**
   * Description: JavaScript and Bootstrap are chosen for enhancing the user interface and styling of the web application.
   * Purpose: JavaScript will be used for client-side interactions and dynamic content, while Bootstrap will provide pre-designed CSS styles and components for a visually appealing interface.
5. **NLTK (Natural Language Toolkit)**:
   * **Description**: NLTK is a comprehensive library for natural language processing (NLP) tasks, providing tools and resources for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, and sentiment analysis.
   * **Purpose**: NLTK can be used for analyzing and processing text data, including WhatsApp call transcripts, for tasks such as sentiment analysis, keyword extraction, and topic modeling.
6. **Requests**:
   * **Description**: Requests is a popular library for making HTTP requests in Python, providing a simpler and more user-friendly interface compared to the built-in urllib module.
   * **Purpose**: Requests can be used for interacting with external APIs, such as WhatsApp web integration, to send and receive data over the internet.
7. **Beautiful Soup**:
   * **Description**: Beautiful Soup is a library for parsing HTML and XML documents, providing tools for navigating the parse tree and extracting data from web pages.
   * **Purpose**: Beautiful Soup can be used for web scraping tasks, such as extracting text data or call transcripts from web pages, to supplement the analysis of WhatsApp call transcripts.
8. **SQLAlchemy**:
   * **Description**: SQLAlchemy is an ORM (Object-Relational Mapping) library for Python, providing a high-level interface for interacting with relational databases.
   * **Purpose**: SQLAlchemy can be used for database interactions in conjunction with PostgreSQL, simplifying database operations such as querying, inserting, updating, and deleting data.
9. **Flask-SQLAlchemy**:
   * **Description**: Flask-SQLAlchemy is an extension for Flask that integrates SQLAlchemy with Flask applications, providing additional features and utilities for database management.
   * **Purpose**: Flask-SQLAlchemy streamlines the integration of SQLAlchemy with Flask-based web applications, making it easier to define database models, manage database connections, and execute database queries.
10. **Flask-WTF**:
    * **Description**: Flask-WTF is an extension for Flask that integrates WTForms with Flask applications, providing tools for creating and validating web forms.
    * **Purpose**: Flask-WTF simplifies the process of handling form data in Flask-based web applications, including user input for login forms, data upload forms, and configuration settings.

Algorithm description

Machine Learning Algorithms

1. **Naive Bayes Classifier:**

* Description: Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence given the class. It's simple, efficient, and often used as a baseline model for text classification tasks.
* Complexity: Naive Bayes has low computational complexity and training time. It scales well with large datasets and high-dimensional feature spaces. However, it may suffer from the "zero-frequency problem" when encountering unseen feature combinations.

2. **Maximum Entropy Classifier (MaxEnt):**

* Description: Maximum Entropy (MaxEnt) is a probabilistic model that assigns probabilities to possible outcomes based on the principle of maximum entropy. It's a versatile classifier capable of handling complex feature representations and nonlinear decision boundaries.
* Complexity: MaxEnt classifiers have moderate computational complexity during training, particularly when using large feature sets or complex feature representations. The optimization process involves iterative algorithms such as gradient descent or L-BFGS, which may require significant computational resources.

3. **Decision Trees:**

* Description: Decision Trees are a non-parametric supervised learning method used for classification and regression tasks. They recursively partition the feature space based on feature values and make decisions by traversing the tree from the root to the leaf nodes.
* Complexity: Decision Trees have low computational complexity during training but may suffer from overfitting, especially with deep trees and noisy data. The time complexity for building decision trees is generally *O*(*n*⋅*m*log*m*), where *n* is the number of samples and *m* is the number of features.

4. **Support Vector Machines (SVM):**

* Description: Support Vector Machines (SVM) are powerful supervised learning algorithms used for classification and regression tasks. They find the optimal hyperplane that separates data points into different classes while maximizing the margin between classes.
* Complexity: SVMs have moderate to high computational complexity during training, particularly with large datasets or high-dimensional feature spaces. The time complexity is typically *O*(*n*2⋅*m*) *O*(*n*3⋅*m*), where *n* is the number of samples and *m* is the number of features. SVMs also require careful selection of hyperparameters and tuning.

5. **Conditional Random Fields (CRF):**

* Description: Conditional Random Fields (CRF) are probabilistic graphical models used for sequence labeling tasks such as named entity recognition and part-of-speech tagging. They model the conditional probability of labels given input features and capture dependencies between adjacent labels.
* Complexity: CRFs have moderate computational complexity during training, which involves estimating parameters based on labeled sequence data. The time complexity depends on the size of the training dataset and the complexity of the feature representations. Inference with CRFs can be efficient during prediction.

Use case diagram –

A diagram of a diagram

Description automatically generated

**Use Case 1: User Login**

**Flow:**

1. User Access: User navigates to the web interface of the system.
2. Login Page: User is presented with a login page prompting for username and password.
3. Input Credentials: User enters their username and password into the respective fields.
4. Authentication: The system verifies the entered credentials against the stored user database.
5. Authorization: If the credentials are valid, the user is granted access to the system; otherwise, an error message is displayed indicating invalid credentials.
6. Session Management: Upon successful authentication, the system creates a session for the user, allowing them to access restricted features and functionalities.
7. Navigation: User is redirected to the main dashboard or landing page of the system, where they can proceed to perform various actions and tasks.

**Use Case 2: Analysing WhatsApp Call Transcripts**

**Flow:**

1. User Interaction: User navigates to the web interface and clicks on the "Upload WhatsApp Call Transcripts" button.
2. File Upload: User upload WhatsApp call transcript files from their local system through the web interface.
3. Processing: The system receives the uploaded files and performs preprocessing tasks such as text normalization, tokenization, and stopwords removal using NLTK.
4. Analysis: The system analyses the processed text data using NLTK for tasks such as sentiment analysis, keyword extraction, or topic modelling.
5. Visualization: The analysis results are visualized using graphs, charts, or other graphical representations, which are displayed on the web interface for the user to interpret.

**Use Case 3: Alerting high risk WhatsApp Calls**

**Flow:**

1. User Interaction: User connects their WhatsApp account to the system by clicking on the "Connect WhatsApp Web" button.
2. Authorization: The system prompts the user to authorize access to their WhatsApp account by scanning a QR code using their mobile device.
3. Monitoring: The system continuously monitors incoming WhatsApp calls and their transcripts using WhatsApp web integration.
4. Analysis: When a call is detected, the system analyses the call transcript using NLTK to assess its content for potential danger or risk.
5. Alert Generation: If the analysis identifies a dangerous call based on predefined criteria, the system generates an alert message.
6. Alert Delivery: The alert message is sent to the user the web interface, notifying them of the potentially harmful call.