**Part 3 - Detailed Design**

**Architecture-**

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

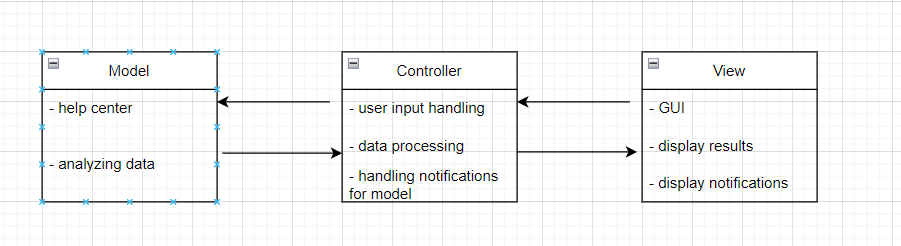
**MVC Architecture in our project:**

* **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** -

The reason we selected Python is that we need to make a statistics analysis. For that reason, we chose to work with Matplotlib, Pandas, and Numpy libraries for statistics and graph generators and SKlearn for the machine learning model.

For the view we choose to use in js in bootsrap for better style.

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.