

---

# FORECASTING CONFERENCE REGISTRATIONS

GROUP PROJECT REPORT  
(CSC-40038)

---

Keele University, UK  
School of Computing and Mathematics



## Team Members

|                         |          |
|-------------------------|----------|
| Hashim Ibrahim Thanalil | 24008144 |
| Tomi Kolawole           | 23017287 |
| Benjamin Mensa-Bonsu    | 24007706 |
| Mohit Basit             | 24006984 |
| Muhammad Usama          | 23042908 |
| Syed Ahmad Ali Shah     | 23042163 |

# Contents

|   |    |
|---|----|
| 1. Introduction.....                                    | 3  |
| 2. Final Application .....                              | 4  |
| 3. User Documentation .....                             | 7  |
| 4. Technical Documentation .....                        | 10 |
| 5. Client Interactions and Requirements Gathering ..... | 12 |
| 6. Software Design and User Interactions.....           | 14 |
| 7. Software and Interface Implementation.....           | 15 |
| 8. Evaluation and Testing of Finished Product.....      | 25 |
| 9. Conclusion .....                                     | 26 |
| References.....   | 29 |

# 1. Introduction

## Purpose of the Project

Our main goal for this project is to create a forecasting model that can accurately predict the number of final registrations for a conference. This will be based on the patterns we observe during the registration period. This project aims to address the business problem of determining the necessity and timing of additional advertising to achieve the desired attendance levels for the conference. The model will enhance decision-making for marketing and promotional activities by offering a dependable forecast of final registrations at different stages during the registration period. This will help ensure that the desired attendance levels are met in an efficient and cost-effective manner.

## Scope

**1. Data Collection and Preparation:** Collecting historical registration data from similar conferences and getting it ready for analysis. Part of the process involves ensuring the data is clean and addressing any missing or inconsistent entries.

**2. Conducting Exploratory Data Analysis (EDA)** to gain insights into the trends and patterns observed in the registration data. Visualizing the data and identifying the key factors that impact registration numbers is an important aspect of this task.

**3. Model Development:** Creating a forecasting model utilizing a combination of statistical and machine learning techniques. The model will be trained using past data to forecast future registrations by analyzing the current registration patterns.

**4. Model Evaluation:** Assessing the effectiveness of the forecasting model through the use of suitable metrics and validation techniques. By following this process, the model is able to deliver precise and dependable predictions.

**5. Implementation:** Developing the final model into a practical format that meets the client's requirements, such as a user-friendly web application, a versatile Python script, or a convenient Excel spreadsheet.

**6. Documentation:** Offering thorough documentation that encompasses user guides, technical documentation, and information about the client interactions and requirements gathering process.

## Limitations and Extent of the Solution

- The forecasting model will rely on past registration data from conferences of a similar nature. The accuracy of the report relies on the quality and relevance of the historical data.
- The model assumes that the initial marketing efforts are uniform across all conferences, with the first email campaign sent out two days before the start of registrations.
- The target attendance levels are adaptable and not fixed. The model aims to provide a forecast based on current trends, without considering potential future changes in marketing strategies or external factors.

- The project will primarily concentrate on developing a model to forecast registrations at different stages of the registration period. However, it will not explore the optimization of marketing strategies or the prediction of the effects of specific marketing campaigns.

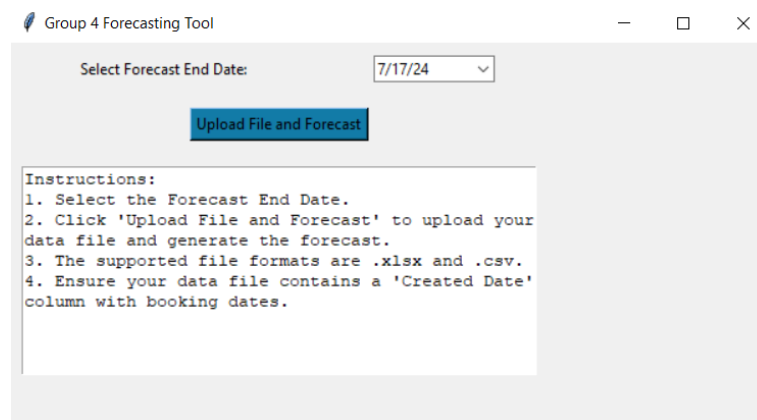
## 2. Final Application

The final application developed by the team is a forecasting tool designed to predict conference registration numbers based on historical data. This application utilizes statistical and machine learning methods to examine historical registration patterns and offer predictions for future registration numbers at different stages of the registration period

GitHub repo: <https://github.com/ahmad865/CSC40038-Group4-ForecastingApplication.git>

### Various Components of the Application

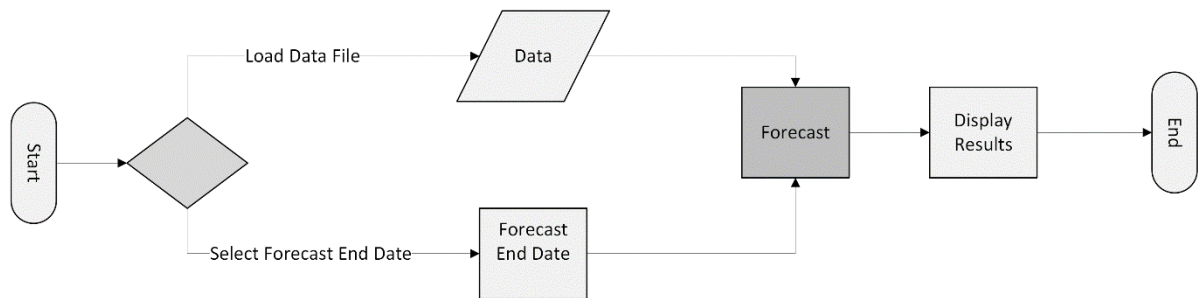
- **User Interface (UI) with Tkinter:** The Python Tkinter is used to create a graphical user interface (GUI) for the application. This includes:
  - A date picker for selecting the forecast end date.
  - A button to upload an Excel or CSV file.
  - A background image for aesthetic purposes.



*Figure 1: User Interface for forecasting*

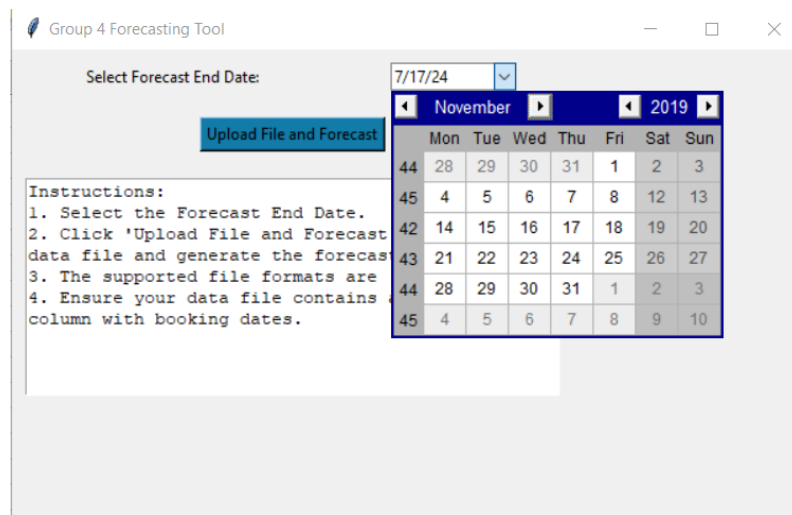
- The forecasting algorithm utilized is the ARIMA (AutoRegressive Integrated Moving Average) model from the statsmodels package. It predicts future booking values by analyzing previous data. This approach is appropriate for the task of predicting future values in a time series.
- The interactivity graph utilizes Plotly to build a dynamic graph that showcases the current and projected number of bookings on a web browser. This graph is interactive and provides the ability to zoom in and out, move across the graph, and view detailed information by hovering over data points.

## General Flow of the Application



### I. Initialization:

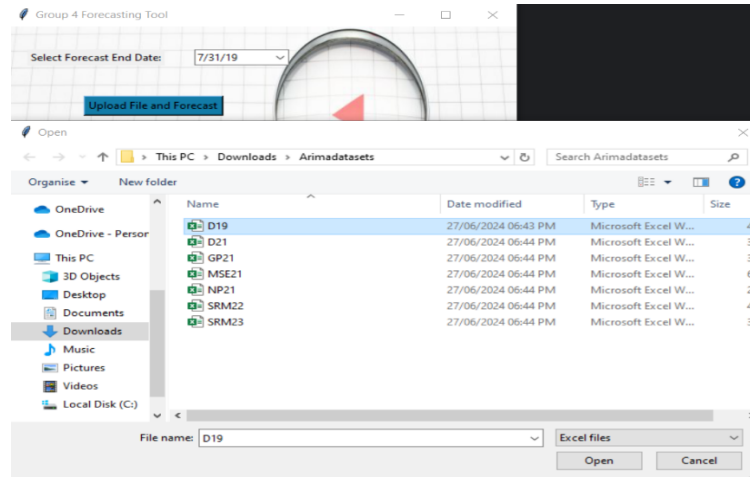
The user initiates the application, which presents a Tkinter interface featuring a date picker and an upload button. The user chooses the desired date for the forecast to conclude.



*Figure 2: User Selects specific Forecast End Date*

### II. File Upload:

The user clicks the "Upload File and Forecast" button to upload an Excel or CSV file. The file dialog opens for the user to select a file from their own computer. When the user uploads a file, the `upload\_file` function is called and the selected file is read into a pandas DataFrame.



*Figure 3: User uploads a file from their computer for the forecast.*

### III. Data Processing:

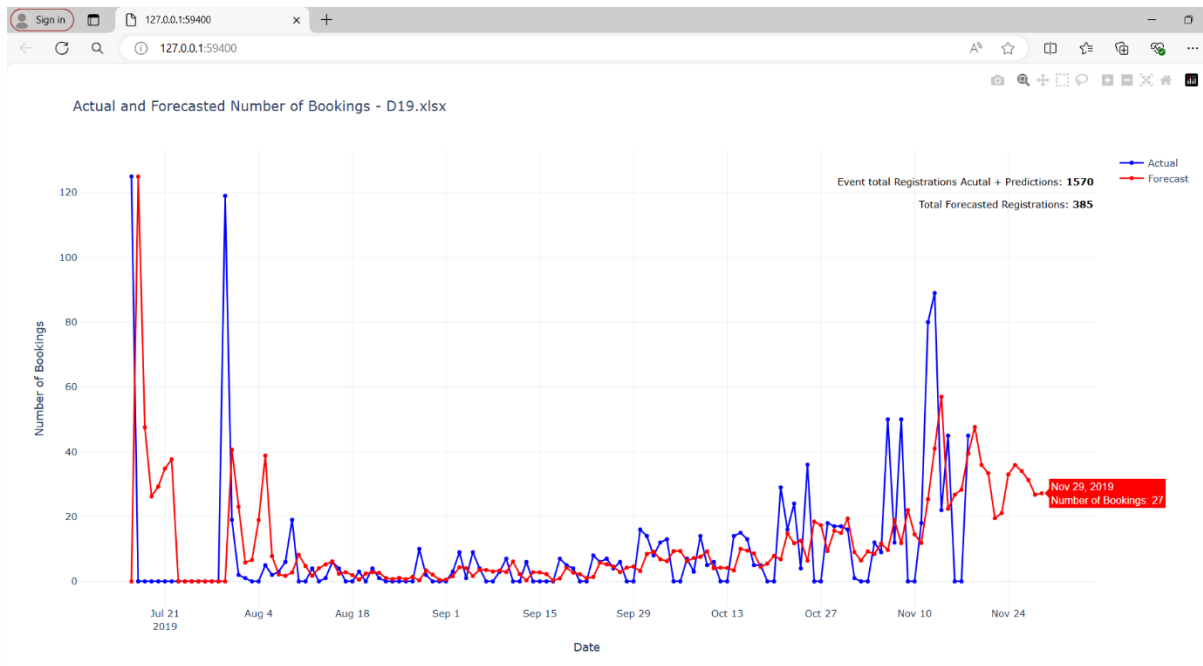
Dates are parsed, and the data is sorted by the 'Created Date'. A time series is created for the count of 'BookingReference' per day. The time series is resampled to a daily frequency. The `load\_data\_and\_forecast` function processes the data, fits an ARIMA model, generates forecasts, and creates a plot.

### IV. Forecasting:

The ARIMA model is fitted to the time series data. Forecasting is done from the earliest date in the data to the user-selected forecast end date. Negative forecast values are replaced with zero. ARIMA models can sometimes produce negative forecasts, especially when dealing with low counts or volatile data. This is a limitation of the model, not necessarily a reflection of reality. Nonetheless, by setting negative values to zero, the forecast becomes more interpretable and usable for business planning. Business stakeholders may consider zero bookings as a more realistic minimum than negative ones.

### V. Visualization:

A Plotly figure is created to plot the actual and forecasted data. The figure is displayed in the browser, showing the title with the uploaded file name. The user can hover over forecast lines to see the forecasted bookings at a specific date.



*Figure 4: Interactive Plot line to see number of bookings at a specific date.*

## VI. Application Termination:

The Tkinter window closes automatically after displaying the plot, preventing multiple forecasts in one session.

## 3. User Documentation

The Group 4 Forecasting Tool is an intuitive application created to assist event organizers in anticipating the number of registrations for upcoming conferences. By utilizing past registration data, this tool employs statistical techniques to predict future registration patterns. This enables users to make well-informed choices regarding their marketing strategies and guarantee that their desired attendance goals are achieved. The application boasts a user-friendly graphical user interface (GUI) that streamlines the tasks of data loading, analysis, and visualization.

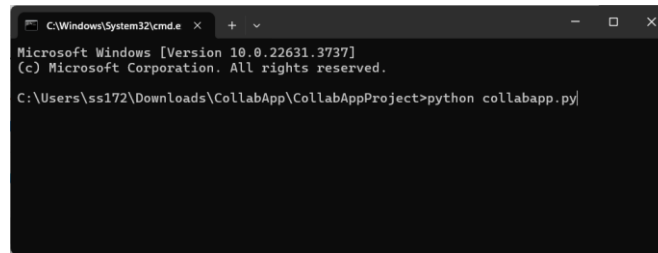
### User Guide

#### Step-by-Step Instructions:

##### Launch the Application:

- Open a terminal or command prompt.
- Navigate to the directory containing the collabapp.py file.

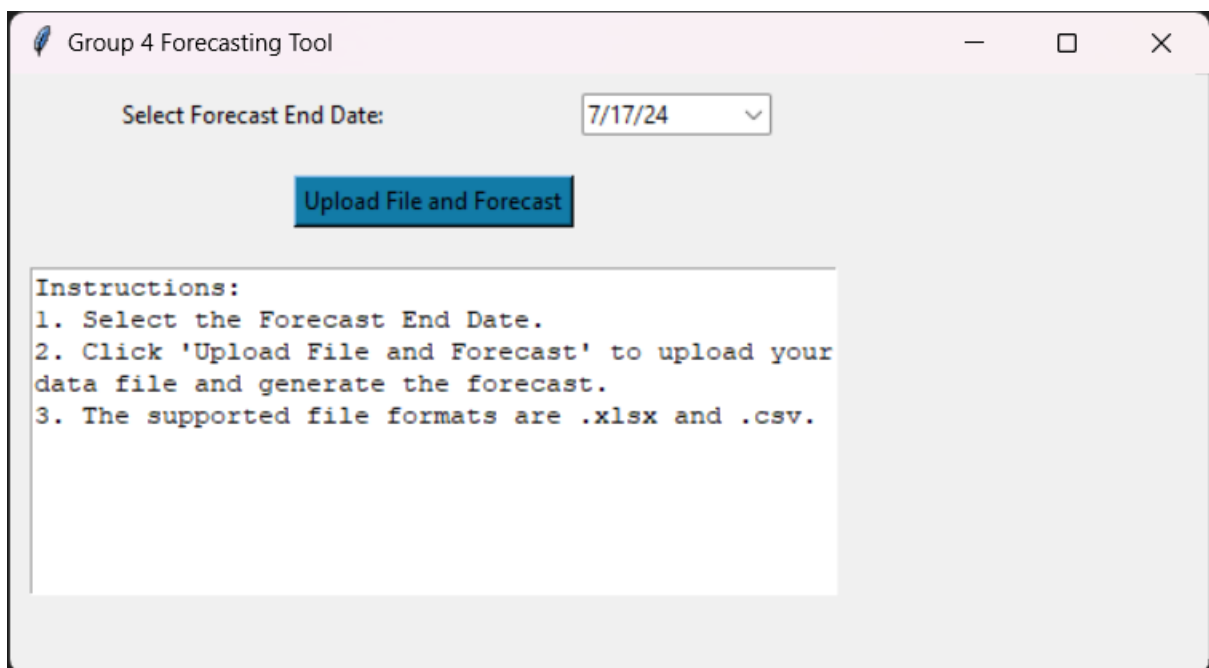
- Run the following command to start the application:  
`python collabapp.py`



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

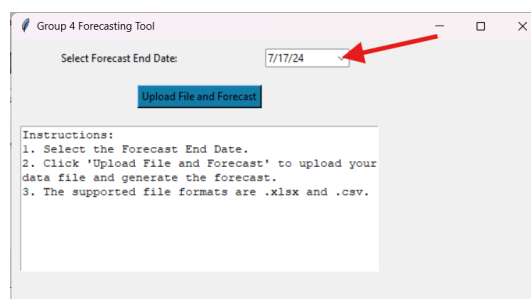
C:\Users\ss172\Downloads\CollabApp\CollabAppProject>python collabapp.py
```

- The main window of the Group 4 Forecasting Tool will appear.



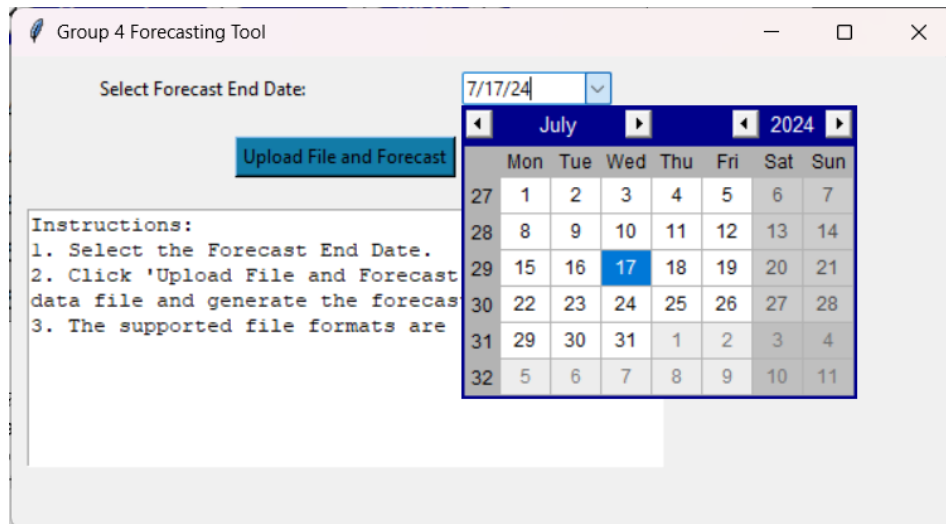
### Select the Forecast End Date:

- Find the label that says "Select the forecast end date" in the main window.



- Click on the date entry widget next to the label.
- A calendar will appear. Choose the date up to which you want to forecast the registrations.

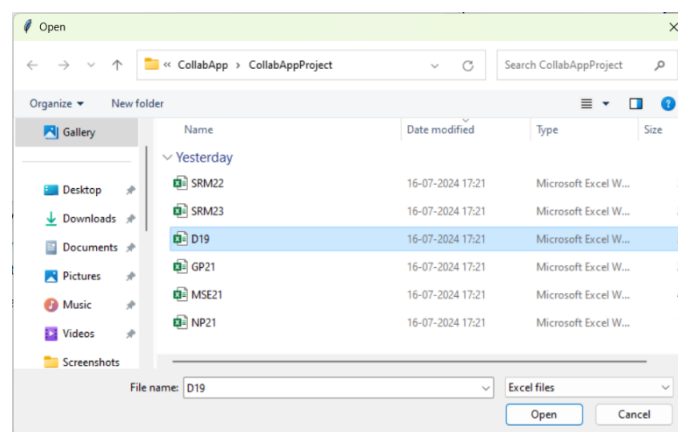




- The selected date will be displayed in the widget.

### Select the Registration Data File:

- Click on `Upload File and forecaste` Button
- In the main window, locate the label that says "Select the registration data file".
- Click the "Browse" button next to it.
- A file dialog will open. Navigate to and select the CSV file containing your historical registration data.



- The selected file's path will be displayed in the application.

### Perform the Forecast:

- Click the "Forecast" button located at the bottom of the main window.
- The application will process the data, fit the ARIMA model, and generate a forecast for the selected period.
- A Plotly interactive graph will be displayed, showing both the historical registration data and the forecasted values.

### **Interpreting the Results:**

- The graph will have two lines: one representing the actual registration data and the other showing the forecasted registrations.
  - Hover over the graph to see detailed data points and trends.
  - Use the zoom and pan features of Plotly to explore the graph in more detail.
- 2. Saving the Results (Optional):**
    - If you wish to save the forecast results or the graph, use the built-in options in the Plotly interface to download the plot as an image or export the data.
  - 3. Close the Application:**
    - Once you are done with your analysis, you can close the main window of the application.

### **Note:**

- Ensure that your CSV file contains a 'Created Date' column with dates in a recognizable format (e.g., DD/MM/YYYY) for the application to process it correctly.
- The forecast end date should be within a reasonable range to ensure the accuracy of the predictions.

## **4. Technical Documentation**

The Group 4 Forecasting Tool is designed to predict conference registration numbers using historical data. The application is built with Python and incorporates various libraries for data manipulation, time series forecasting, and graphical user interface (GUI) development. The architecture of the application is modular, separating the concerns of data handling, forecasting, visualization, and user interaction.

### **Architecture and Components**

- 1. User Interface (UI) Layer:**
  - Developed using Tkinter for creating the main application window and handling user inputs.
  - Includes elements such as buttons, labels, and date entry widgets.
- 2. Data Processing Layer:**
  - Utilizes Pandas for reading and cleaning the registration data.
  - Converts the data into a time series format suitable for forecasting.
- 3. Forecasting Layer:**
  - Implements the ARIMA model from the Statsmodels library to perform time series forecasting.
  - Handles the generation of forecasted values based on user-defined parameters.
- 4. Visualization Layer:**
  - Uses Plotly to create interactive graphs that display both historical data and forecasted trends.
  - Provides an intuitive interface for users to explore the results.

## Diagrams

### 1. Application Flowchart:

```
A[Start Application] --> B[Load Data File]
B --> C[Process Data]
C --> D[Select Forecast End Date]
D --> E[Perform Forecast]
E --> F[Display Results]
F --> G[End]
```

## Functionality

### 1. MainWindow Class:

- **Description:** Manages the GUI and user interactions.
- **Key Methods:**
  - `file_dialog()`: Opens a file dialog to select the registration data file.
  - `run_forecast()`: Initiates the data loading, processing, and forecasting workflow.

### 2. DataProcessor Class:

- **Description:** Handles the loading and processing of registration data.
- **Key Methods:**
  - `load_data(file)`: Reads the data file into a Pandas DataFrame.
  - `clean_data(data)`: Cleans and prepares the data for analysis.
  - `convert_to_timeseries(data)`: Converts the cleaned data into a time series format.

### 3. Forecaster Class:

- **Description:** Performs time series forecasting using the ARIMA model.
- **Key Methods:**
  - `fit_model(timeseries)`: Fits an ARIMA model to the time series data.
  - `forecast(model, steps)`: Generates forecasted values based on the fitted model.

### 4. Visualizer Class:

- **Description:** Creates interactive visualizations of the data and forecast results.
- **Key Methods:**
  - `plot_data(timeseries, forecast)`: Plots the historical data and forecasted values using Plotly.

## Setup Instructions

### Development Environment Setup:

### 1. Install Python:

- Download and install Python from [python.org](https://python.org).

### 2. Install Required Libraries:

- Open a terminal or command prompt.
- Use pip to install the necessary libraries:

```
pip install pandas plotly statsmodels tkinter tkcalendar openpyxl
```

## Running the Application:

### 1. Save the Script:

- Save the `collabapp.py` script to a directory on your computer.

### 2. Navigate to the Directory:

- Open a terminal or command prompt and navigate to the directory containing `collabapp.py`.

### 3. Execute the Script:

- Run the script by executing:

```
python collabapp.py
```

## 5. Client Interactions and Requirements Gathering

### Client Meetings

#### Meeting on June 13, 2024:

- **Purpose:** Initial discussion with Gerard, the client, to understand the project requirements and objectives.
- **Key Points:**
  - Gerard's company develops software for events, such as conferences and exhibitions.
  - The primary goal is to reduce uncertainty regarding conference attendance, helping organizers decide on necessary arrangements (e.g., venue size, facilities, and catering).
  - Discussion on predicting the number of attendees based on early registrations and determining the optimal time for additional advertising.
  - The dataset includes registration details from six conferences, requiring cleaning and analysis.
  - Importance of real data, with a focus on cleaning and selecting useful data.
  - Gerard expressed the need for practical application rather than purely academic insights.
  - Plan for data processing, model creation, and the development of a user-friendly interface for predictions.

#### Meeting on June 20, 2024:

- **Purpose:** Follow-up discussion to clarify specific requirements and refine the project plan.
- **Key Points:**
  - Gerard emphasized focusing on registrations rather than actual attendance.
  - Requirement for a prediction model that can provide end-of-period registration forecasts at various points.
  - Importance of removing irrelevant columns (e.g., attended, cancelled) and preprocessing the data consistently.
  - Exploration of advertising trends and their impact on registration rates.
  - Plan to train the model on a portion of the data and test it at different stages to ensure accuracy.
  - Distribution of tasks among team members to preprocess data, explore advertising trends, and evaluate initial data.

## Techniques Used

### Interviews:

- **Objective:** Direct discussions with the client to gather detailed requirements and understand the project context.
- **Method:** Face-to-face and virtual meetings with Gerard to discuss goals, expectations, and specific needs.

### Document Analysis:

- **Objective:** Review existing documentation and data provided by the client to understand the current state and historical trends.
- **Method:** Analyzing the datasets and previous conference data to identify patterns and potential challenges.

## Findings

### Results from Requirements Gathering:

- **Focus on Registrations:** The client is primarily interested in predicting registrations rather than actual attendance. This shifted the focus of the data analysis and model development.
- **Data Cleaning Needs:** The provided datasets contain irrelevant columns and inconsistent entries, necessitating thorough data cleaning.
- **Model Requirements:** The prediction model needs to provide forecasts at various stages (e.g., after 25%, 50%, and 75% of the registration period).
- **User Interface:** The final product should be user-friendly, allowing the client to input data and generate forecasts easily without requiring technical expertise.

### Application Design Informed by Findings:

- **Data Processing:** Comprehensive data cleaning procedures were implemented to ensure the dataset is suitable for analysis and modeling.

- **Forecasting Model:** The ARIMA model was selected for its robustness in time series forecasting, and parameters were fine-tuned based on historical data.
- **User Interface Design:** An intuitive GUI was developed using Tkinter, enabling easy data input, selection of forecast end dates, and viewing of results.
- **Visualization:** Interactive graphs using Plotly were incorporated to provide clear visual representations of registration trends and forecasts.
- **Functionality:** Features such as file selection, date entry, and forecast generation were prioritized to meet the client's need for simplicity and efficiency.

## 6. Software Design and User Interactions

### Design Methods

#### Wireframes and Prototypes:

- **Wireframes:** Initial wireframes were created to outline the layout of the application's graphical user interface (GUI). These wireframes served as a visual guide for positioning elements such as buttons, labels, and date entry widgets.
- **Prototypes:** Low-fidelity prototypes were developed using Tkinter to simulate the user interactions and ensure that the flow of actions within the application was intuitive. These prototypes were iteratively refined based on feedback from potential users and stakeholders.
- **User-Centered Design (UCD):** The application was designed with a user-centered approach, focusing on ease of use and accessibility by minimizing the inputs required from the user and providing clear, interactive output.
- **Modular Design:** Functions are used to encapsulate different functionalities such as file upload, data processing, forecasting, and visualization, making the code easier to manage and extend.

### User Interface Design

#### Process and Considerations:

1. **Simplicity and Clarity:**
  - **Layout:** The layout of the application was kept simple and uncluttered. Key functions such as file selection and forecast execution are prominently placed and easily accessible.
  - **Labels and Instructions:** Clear labels and instructions guide the user through each step of the process, from selecting a data file to generating and viewing the forecast.
2. **Interactive Elements:**
  - **Buttons:** Actionable buttons like "Browse" and "Forecast" are used to trigger key functions. These buttons are sized appropriately and placed to ensure they are easily clickable.
  - **Date Picker:** A date entry widget is provided for selecting the forecast end date. This allows users to input dates conveniently and reduces the likelihood of input errors.
3. **Feedback Mechanisms:**

- **Loading Indicators:** Visual cues, such as progress indicators, inform users that the application is processing data and generating forecasts. This helps in managing user expectations during longer processing times.
  - **Error Messages:** Clear and concise error messages are displayed to guide users in case of invalid inputs or other issues.
4. **Visualization:**
- **Interactive Graphs:** The use of Plotly for visualization ensures that the graphs are interactive, allowing users to zoom, pan, and hover over data points for more detailed information.
  - **Color Coding:** Different colors are used to distinguish between actual registration data and forecasted values, enhancing the readability of the graphs.

## Justification

### User Needs and Requirements:

1. **Ease of Use:**
  - The primary users of the application are event organizers who may not have a technical background. Therefore, the application is designed to be user-friendly and intuitive, requiring minimal technical knowledge to operate.
2. **Functionality:**
  - The key functionalities, such as data loading, processing, and forecasting, are streamlined to ensure users can perform tasks efficiently without navigating through complex menus or settings.
3. **Accessibility:**
  - The use of Tkinter for the GUI ensures that the application is accessible on various operating systems where Python can be installed, making it versatile and easy to deploy.
4. **Feedback and Interaction:**
  - Providing immediate visual feedback through interactive graphs and progress indicators ensures that users are informed about the status of their tasks, which enhances the overall user experience.
5. **Accuracy and Reliability:**
  - The application uses established statistical methods (ARIMA) for forecasting, ensuring that the predictions are based on robust mathematical models. This reliability is crucial for users making business decisions based on the forecasts.

## 7. Software and Interface Implementation

### Implementation Process

#### Development Phases:

1. **Requirement Analysis:**
  - The first step involved understanding the client's requirements and the business problem. This was achieved through initial meetings and discussions, which clarified the objectives of forecasting conference registrations.

## 2. Design:

- Based on the requirements, wireframes and low-fidelity prototypes were created. These designs were iteratively refined based on user feedback to ensure that the application was intuitive and user-friendly.

## 3. Data Preparation:

- The historical registration data was cleaned and preprocessed to be used in the forecasting model. This involved handling missing values, converting dates, and sorting the data.

## 4. Model Development:

# Prophet Approach

Prophet is a forecasting tool developed by META(Facebook). It is designed for handling and predicting time series data. It has many useful features like recognizing seasonal data, holidays and trends in a time series dataset. But since in our dataset there is no seasonality and holiday effects, we Prophet to find trends and recognize patterns in our data and make predictions for future dates according to those patterns.

The trend feature of it captures, increases and decreases in data over time which is suited for our type of data. Another benefit of using prophet is that it tries to cover for the missing values in the dataset by looking at the previous trends, which is very important for our application, because our data had some missing values between the registrations.

## Data Pre-Processing

The very first step in our application development was to prepare the data, so it can be used for training the Prophet model. We applied various steps to process the data which are as follows,

### *Cleaning/Removing Inconsistencies*

A few events data had some huge gaps and inconsistencies between the data, mainly due to Covid 19 pandemic, which was causing inaccurate model training and predictions, so we removed the dates during Covid lockdowns all together. This made the data consistent and uniform across all events and made it easier for models to recognize similar patterns/trends in the data. The data for event NP21 was very inconsistent and spanned over more than one year with very few registrations,

### *Combining Events data/ Creating a Time series*

We realized that instead of training a different Prophet model for each event data, it would be better to combine all events data in a single data frame. We combined all events CSVs files into a single Data frame making a time series of events from 2019 to 2023. Data with similar dates was aggregated in the time series. These new time series was named historical data. The code for this process is given below.

```
# previous events CSV file names
historical_files = ['D19.csv', 'D21.csv', 'GP21.csv', 'NP21.csv', 'MSE21.csv',
'SRM22.csv', 'SRM23.csv']

# Loading and combining historical data
historical_data = pd.concat([pd.read_csv(file) for file in historical_files])
```



```
historical_data['Created Date'] = pd.to_datetime(historical_data['Created Date'], format='%d/%m/%Y')
historical_aggregated = historical_data.groupby('Created Date').size().reset_index(name='registrations')
```

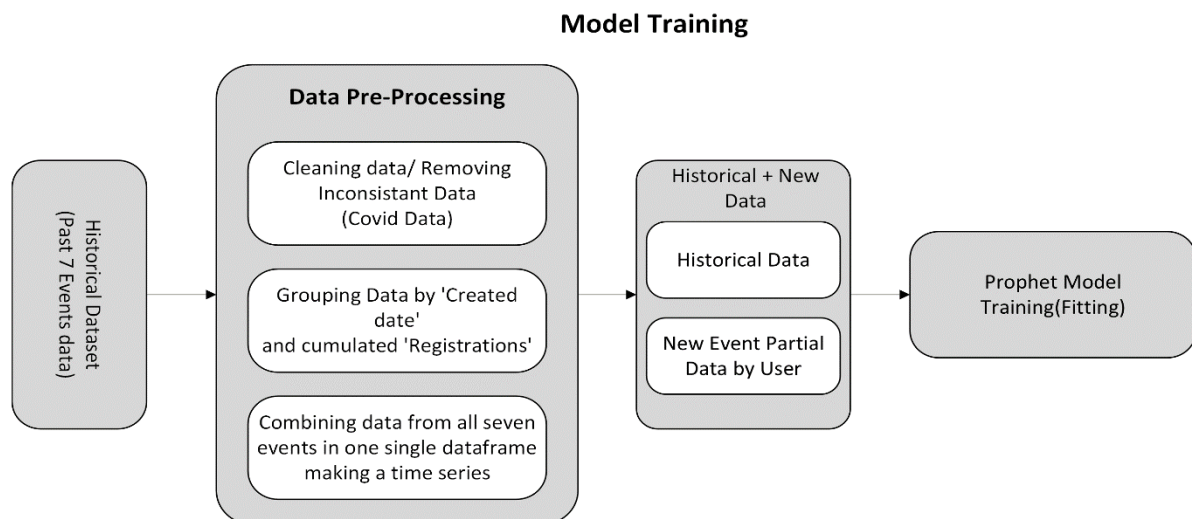
### *Combining Historical Data and New Event Partial Data*

In our prediction model, the user gives partial data for new events as a CSV file. We combined this new partial data at the end of historical data, processed in the previous step. Thus, making a time series from earliest date in historical data to latest date in the partial data. This time the series was then fitted in Prophet model for training. Below is code for this process. Created a new column in data frame as cumulative count of 'created date' column, which is in fact number of registrations. The new column was named 'registrations'.

```
# Loading and processing partial data for the new event
partial_data = pd.read_csv('TestMSE21.csv')
partial_data['Created Date'] = pd.to_datetime(partial_data['Created Date'],
format='%d/%m/%Y')
partial_aggregated = partial_data.groupby('Created Date').size().reset_index(name='registrations')

# Combining historical and partial data
combined_data = pd.concat([historical_aggregated, partial_aggregated])
```

The figure below depicts the complete data pre-processing process to make a single time series and train the prophet model.



### Model Training and Forecasting

The steps after data pre-pressing were to use the data to train a Prophet model using the combined time series data and forecasting the registrations for future dates. This process was done as follows,

## *Creating and Fitting Prophet Model*

The Prophet libraries have many ways to create and train prediction models, it depends whether the time series data has some seasonality to it, or it has holidays effects and trends. It also has various adjusts for changes point scales and seasonality scales, although our data doesn't have any seasonal component to it, the event does take place in different months and years with considerable gaps, which by Prophet model can be seen as seasonality. SO, we tried try differnet models with and without seasonality to get the best possible results. Below code is for a model without seasonality.

```
df = combined_data.rename(columns={'Created Date': 'ds', 'y': 'y'})

# Train the model
model = Prophet()
model.fit(df)
```

Now these is another model with some seasonality and change point scale.

```
# Train the model with tuned parameters
model = Prophet(daily_seasonality=True, weekly_seasonality=False,
yearly_seasonality=False,
                changepoint_prior_scale=0.05, seasonality_prior_scale=10)
model.fit(df)
```

We tried different Prophet models to get the best results for our application.

## *Generating Prediction Dates*

After training the next step was to determine the dates for which we want the registrations number predictions. The Prophet does this in two ways, one can predict for a specific number of days in future (from last date in time series) and other is predict between two specific dates (start date and end date). We tried doing this in both ways but the most optimal for our program was to predict for number of days into future. We calculated this number by looking at latest date in new event partial data (which is also last date in combined time series) and new event start date (which is given by user). This gave use number of days until the event start, so we could predict registrations until the event starts

Below is the code for this step,

```
# Get new event date from the user.
event_start_date_str = input("Enter the new event start date (dd/mm/yyyy): ")
event_start_date = datetime.strptime(event_start_date_str, '%d/%m/%Y')

# Calculate the number of days to predict
last_partial_date = partial_aggregated['Created Date'].max()
days_to_event_start = (event_start_date - last_partial_date).days
```

## *Forecasting Future Registrations*

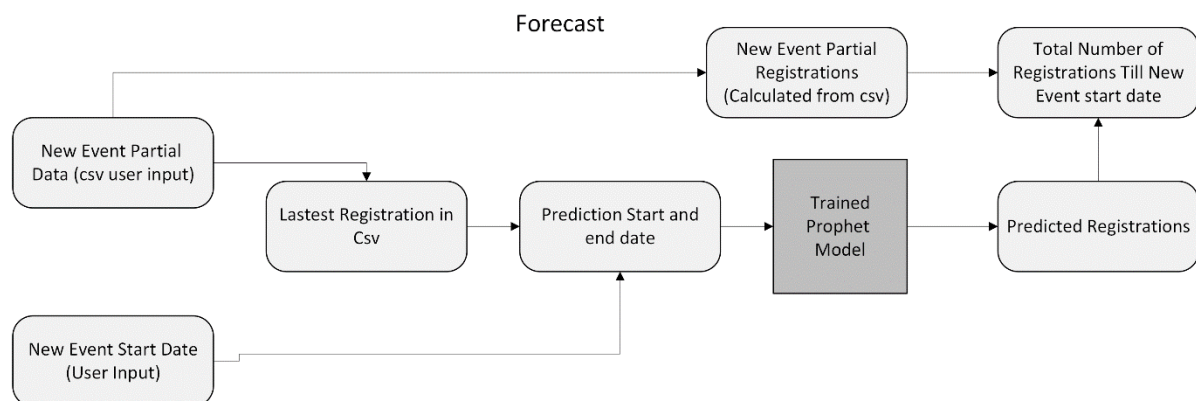
Now that we had a trained Prophet model and number of days to predict for, the final step was to predict for those days in future. By default, the Prophet model shows decrease in trends or days with no registration activity as negative values to we added some code to ignore negative values making them zero. The code is as follows,

```
# Making future dataframe
future_dates = model.make_future_dataframe(periods=days_to_event_start)

# Forecasting
forecast = model.predict(future_dates)

# Ignoring negative values in forecast
forecast['yhat'] = forecast['yhat'].apply(lambda x: max(0, x))
```

The figure below shows the complete architecture of forecasting data using combined time series data and event start date given by the user.



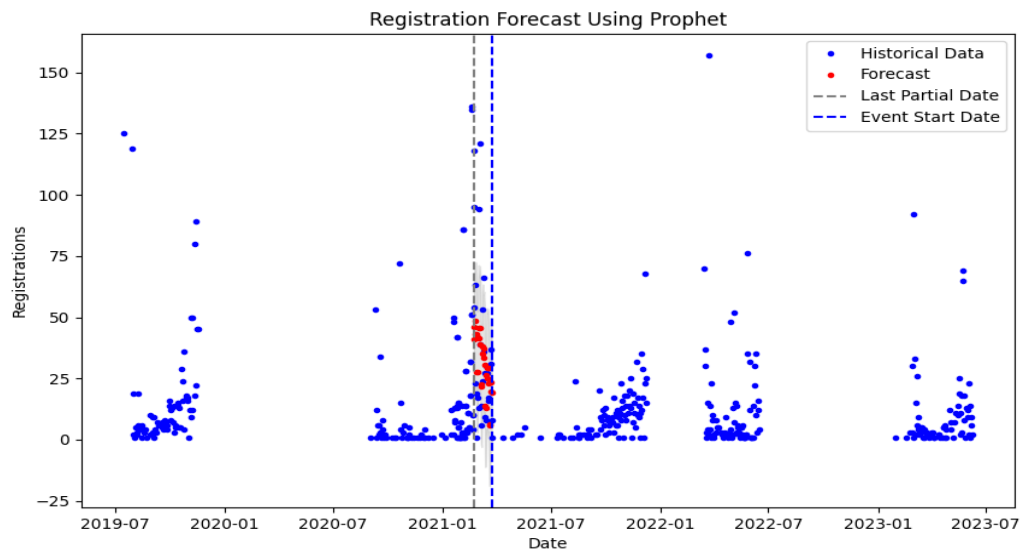
## Prophet Testing

**Preparing Test Data** The event dataset we were working with wasn't very large, so we did not have enough data to train model and then test it on unseen data but, we tried to hide some data from an event csv by removing some end dates from it. In this way we made an event csv file with partial registration, and by getting event start date for that event from user, we tried completing the partial data by predicting the registrations for future dates. We also tried predicting for unseen values, number of days in future (from the end of our time series into future dates).

## Results

We perform two types of tests on the trained Prophet model. First, we tested the model with partial data from a csv it was trained. For known data the mode predict was accurate in range 10% from original data value. The model is accurate for the data or timeline, it was trained

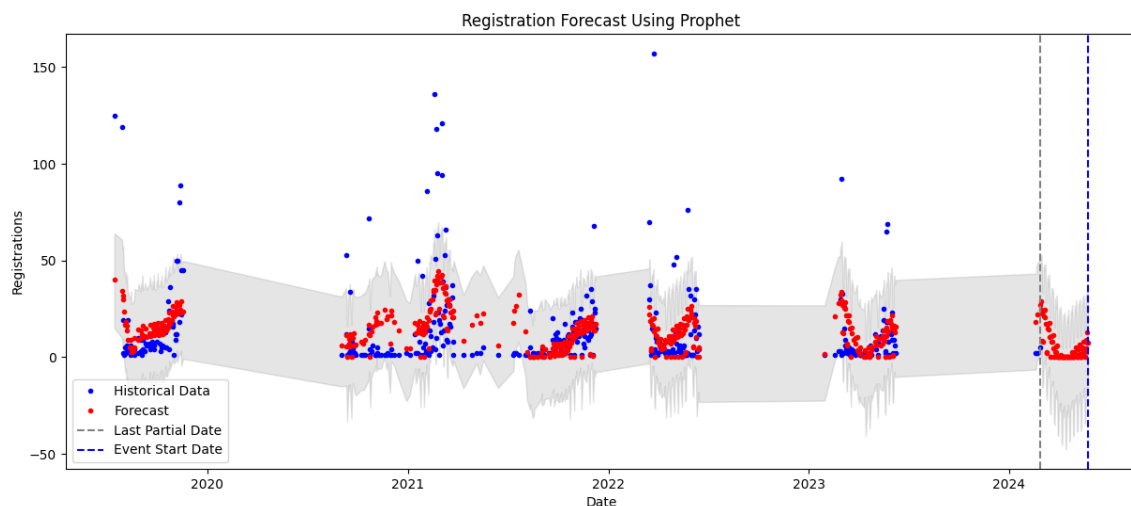
on. Below is a plot for testing partial known data.



And here is the figure with predicted number of registrations.

|  | ds         | yhat      | yhat_lower | yhat_upper |
|--|------------|-----------|------------|------------|
| 1  | 2021-02-23 | 45.971870 | 21.016597  | 71.542061  |
| 2  | 2021-02-24 | 43.225826 | 18.200597  | 67.698032  |
| 3  | 2021-02-25 | 48.589821 | 23.058064  | 74.663323  |
| 4  | 2021-02-26 | 43.248479 | 19.258636  | 67.808443  |
| 5  | 2021-02-27 | 27.631654 | 2.120499   | 52.314569  |
| 6  | 2021-02-28 | 27.724922 | 2.796203   | 52.520567  |
| 7  | 2021-03-01 | 41.924374 | 18.284868  | 67.792104  |
| 8  | 2021-03-02 | 45.505297 | 19.222609  | 69.438422  |
| 9  | 2021-03-03 | 41.448665 | 17.185183  | 65.846169  |
| 10   | 2021-03-04 | 45.579282 | 20.817280  | 70.380936  |
| 11   | 2021-03-05 | 39.099459 | 14.218057  | 65.038472  |
| 12   | 2021-03-06 | 22.453756 | -2.633898  | 46.983852  |
| 13   | 2021-03-07 | 21.639240 | -4.362515  | 47.539365  |
| 14   | 2021-03-08 | 35.060129 | 9.280254   | 61.573462  |
| 15   | 2021-03-09 | 37.996428 | 11.593808  | 61.067756  |
| 16   | 2021-03-10 | 33.430461 | 9.317651   | 58.788308  |
| 17   | 2021-03-11 | 37.185160 | 12.705282  | 60.668542  |
| 18   | 2021-03-12 | 30.457956 | 4.263061   | 55.469883  |
| 19   | 2021-03-13 | 13.685828 | -11.219919 | 37.445492  |
| 20   | 2021-03-14 | 12.855915 | -11.692991 | 36.823252  |
| 21   | 2021-03-15 | 26.360583 | 2.911268   | 51.130893  |
| 22   | 2021-03-16 | 29.466508 | 2.181612   | 53.874507  |
| 23   | 2021-03-17 | 25.141686 | 1.338248   | 49.912034  |
| 24   | 2021-03-18 | 29.194208 | 4.334718   | 54.022491  |
| 25   | 2021-03-19 | 22.806628 | -3.052693  | 47.504406  |
| 26   | 2021-03-20 | 6.401466  | -19.816284 | 30.034922  |
| 27   | 2021-03-21 | 5.952250  | -18.616255 | 30.049893  |
| 28   | 2021-03-22 | 19.838952 | -6.658389  | 44.176409  |
| 29   | 2021-03-23 | 23.317396 | -3.845891  | 48.589921  |
| 30   | 2021-03-24 | 19.346522 | -3.445431  | 45.697847  |
| Total predicted registrations from 2021-02-22 00:00:00 to 2021-03-24 00:00:00: 902.5407225417977 |            |           |            |            |
| PS C:\Users\syedk\Desktop\collab>  |            |           |            |            |

The second test was done on mockup data of a future event, we created a csv file for a mockup future event with couple of registrations already in it and tried to predict for registration couple of months into future. The predicted result was within 20% of registration number noticed during that time in previous years (events). This suggests our model recognizes patterns in our training dataset. And tries to predict registrations for the future based on those trends. The figure below shows predictions results for a supposed future event from 27-02-2024 to 24-05-2024.



Also, the numbers output for same event,

```

      ds      yhat  yhat_lower  yhat_upper
410 2024-02-28  24.672097   -1.152284   49.556667
411 2024-02-29  28.941200    3.655701   52.854261
412 2024-03-01  22.507806   -0.559798   46.113852
413 2024-03-02   8.422734  -16.271922   32.020856
414 2024-03-03   7.996209  -17.251709   32.020828
..      ...      ...      ...
492 2024-05-20   4.618697  -19.521682   29.119813
493 2024-05-21   9.028227  -15.319140   32.963781
494 2024-05-22   7.423152  -15.648691   32.231149
495 2024-05-23  12.638800  -11.681732   38.076487
496 2024-05-24   7.278652  -15.430073   31.802763

[87 rows x 4 columns]
Total predicted registrations from 2024-02-27 00:00:00 to 2024-05-24 00:00:00: 426.3362984547655
PS C:\Users\syedk\Desktop\Collab>

```

## Prophet Model Conclusion

Prophet is a powerful tool for predictions, but it needs training data to be consistent overtime. And to get the most output from it, our data should have other components like seasonality, holidays, promotion campaigns etc. Our prophet models did learn trends patterns from training data and its prediction for data like its training data were almost accurate by margin of 20% to 25%. It predicted registration patterns for months or weeks same as the dataset it was trained on, it was probably looking at seasonality in our data too. Our dataset had some large gaps in it with no data for those dates, there were some months in our time series where no event took place and data for those months was missing in time series. SO, the prediction for those months

in a future event might not be accurate. This issue can be solved by getting more training data, which includes events all year.

## ARIMA Approach:

### Data Exploration and Preprocessing

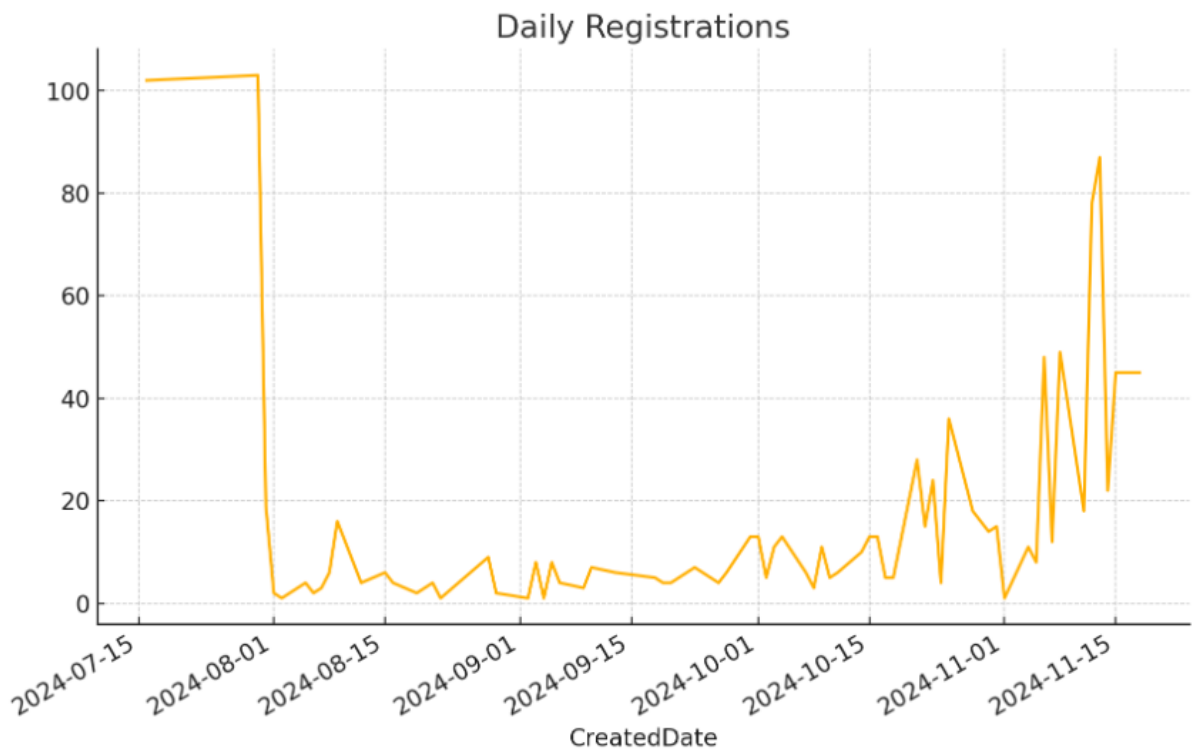
- **Load the Data:** Begin by loading the registration data from the `.csv` files.

### Data Processing

#### Steps Performed in Excel

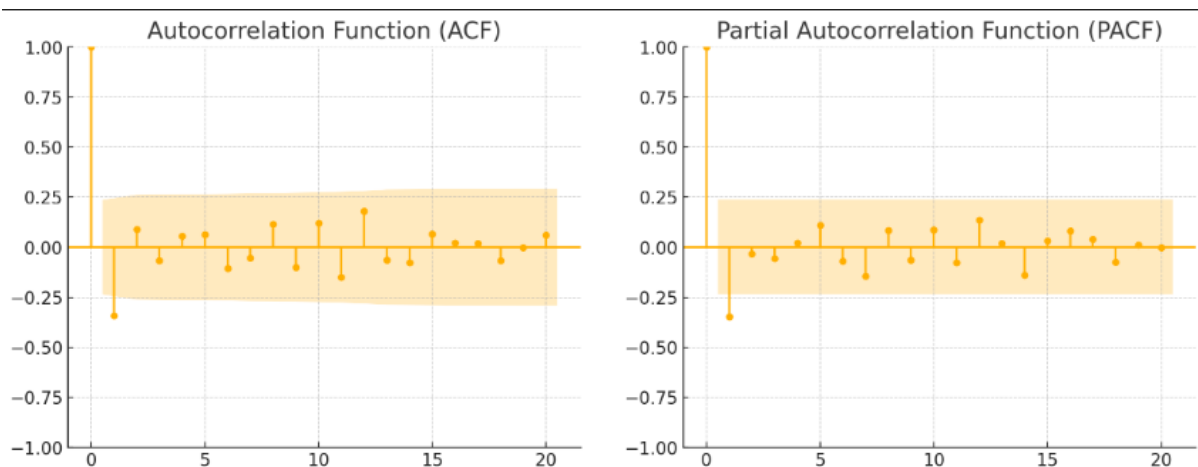
1. **Removed Row 1:**
    - **Action:** Deleted the first row of each CSV file.
    - **Reason:** The first row contained the file name instead of headers, which created errors for the Python file.
  2. **Kept Specific Headers:**
    - **Action:** Retained only the following headers:
      - `BookingReference`
      - `Created Date`
      - `Reference`
      - `Attendee Status`
    - **Reason:** These columns were essential for the analysis, and other columns (e.g., `Attended`) were not relevant since the number of people attended was not necessary for the analysis.
  3. **Converted 'Created Date' to Datetime Format:**
    - **Action:** Changed the format of the `Created Date` column to a datetime format.
    - **Reason:** Ensured that the dates are in a consistent format, which is necessary for sorting and analyzing date-related data.
  4. **Sorted Data by 'Created Date':**
    - **Action:** Sorted the entire dataset based on the `Created Date` column.
    - **Reason:** Organizing the records chronologically helps in understanding the timeline of events and makes the data easier to analyze. The first date is used as the starter date for our time series analysis using ARIMA.
  5. **Standardized 'Attendee Status':**
    - **Action:** Cleaned up the `Attendee Status` column by removing any leading or trailing spaces and converting all entries to a uniform format. Some entries had variations such as `Bookers not attended` instead of `Attended` or `Cancelled`, which were the other two options.
    - **Reason:** Standardization ensures consistency in the data, making it easier to filter and analyze attendee statuses.
- **Data Visualization:** Plot the data to identify patterns, trends, and seasonality.
  - **Filter Data:** Focus on registrations where `AttendeeStatus` is 'Attending'.
  - **Convert Dates:** Convert `CreatedDate` from string to a datetime format for time series analysis.

- **Aggregate Data:** Count the number of registrations per day.
- **Time Series Analysis:** Conduct exploratory analysis on this aggregated data.



## Model Development

- **Stationarity Check:** Use statistical tests (like ADF test) to check if the data is stationary. Differencing might be required for non-stationary data.
- **Determine ARIMA Parameters:** Use ACF and PACF plots and heuristic methods to estimate the parameters ( $p$ ,  $d$ ,  $q$ ).
- **Model Fitting:** Fit the ARIMA model to the historical data.



The ACF and PACF plots provide insights into selecting the parameters for the ARIMA model:

- **ACF Plot:** The autocorrelation function tails off gradually, suggesting that a Moving Average (MA) component might be necessary.
- **PACF Plot:** There is a sharp cut-off after the first lag, indicating that an Autoregressive (AR) component of order 1 may be suitable.

### Suggested ARIMA Parameters:

- **p (AR term):** The PACF plot suggests a value of 5.
- **d (Differencing order):** We've established 0 differencing was sufficient to make the series stationary.
- **q (MA term):** The ACF plot suggests a value of 1 or 2 could be appropriate, but starting with 1 is a conservative approach.

The ARIMA(5,0,1) model has been fitted to the data. Here's a summary of the model's performance:

- **Parameters:** The coefficients for the AR and MA components are not statistically significant ( $p > 0.05$ ), indicating they may not be contributing meaningfully to the model.
- **AIC (Akaike Information Criterion):** 599.461. This value helps us compare different models, with lower values indicating a better fit.
- **Ljung-Box Test:** The ppp-value is high, suggesting there is no significant autocorrelation in the residuals.
- **Jarque-Bera Test:** Indicates that the residuals are not normally distributed due to a very low ppp-value, which may suggest the presence of outliers or the need for a different model configuration.
- **Skewness and Kurtosis:** The residuals' skewness and kurtosis values suggest that they are not normally distributed, which can impact confidence in the forecast.

### Model Evaluation

- **Model Diagnostics:** Check residuals to ensure there are no patterns (should be noise-like).
- **Validation:** If possible, split the data into training and testing sets to evaluate the model's forecasting accuracy.

### Forecasting

- **Future Predictions:** Use the model to forecast the next six months' registrations.
- **Error Metrics:** Calculate error metrics like MAE, RMSE, etc., to quantify forecast accuracy.

#### 5. Interface Development:

- Tkinter was used to develop the graphical user interface (GUI). The interface was designed to be simple and intuitive, allowing users to upload data files, select forecast dates, and view results easily.

#### 6. Integration:

- The data processing, forecasting, and visualization components were integrated with the GUI. This ensured a seamless user experience from data input to forecast output.



## Challenges and Limitations

- **Manual Parameter Selection:** The process of identifying optimal parameters for the ARIMA model (p, d, q) was conducted manually due to challenges in automating this process. Automating parameter selection remains a key area for future development to improve model efficiency and accuracy.
- **Stationarity Checks:** Our attempts to automate checks for stationarity within the data were unsuccessful, which is crucial for ensuring the appropriateness of ARIMA modeling. Future efforts will focus on implementing automated stationarity testing to streamline the data preparation phase.
- **Handling Sudden Peak Values and Seasonality:** The model currently struggles with sudden peaks in registration data and capturing seasonality effects accurately. These issues have led to challenges in model reliability under varying data conditions. Addressing these limitations requires more sophisticated data processing techniques and possibly integrating additional analytical models that can better manage these dynamics.

## 8. Evaluation and Testing of Finished Product

### Testing Preparation

- **Test Data Preparation:** To simulate real-world scenarios, we created a test dataset by modifying an existing event CSV file to have partial registration data, specifically by removing some end dates. This approach allowed us to assess the model's capability to forecast future registrations by predicting the missing end dates based on partial data.
- **Forecasting for Unseen Values:** We attempted to predict registration numbers for future dates extending beyond the end of our existing time series to evaluate the model's performance in completely unseen scenarios.

### Results

- **Prediction Accuracy:** The predictions on the event CSV with partial data were compared against the actual data in the complete event CSV. The results were within a 25% error margin of the actual numbers, indicating a significant potential for model improvement.
- **Future Improvements:** Enhancing prediction accuracy could be achieved by collecting and analyzing a larger dataset from similar events, which would provide more comprehensive insights and refine the forecasting model.

## 9. Conclusion

### Summary

The Group 4 Forecasting Tool project aimed to develop an application that predicts conference registration numbers based on historical data. The project successfully met its objectives by creating a user-friendly application that utilizes statistical and machine learning techniques to provide accurate forecasts. The key outcomes of the project include:

1. **Development of a Functional Application:** The application allows users to load historical registration data, process it, and generate forecasts for future registrations. The user interface is intuitive and accessible, ensuring that users with minimal technical expertise can operate the tool efficiently.
2. **Accurate Forecasting Model:** By implementing the ARIMA model, the application provides reliable predictions of future registration trends. The model was fine-tuned and validated to ensure accuracy.
3. **Interactive Visualization:** The use of Plotly for visualizing data and forecasts enhances the user experience by offering interactive and detailed graphical representations of registration trends.
4. **Robust and Portable Solution:** The application is designed to be cross-platform compatible and incorporates robust error handling and security measures, ensuring reliability and user data privacy.

### Key Learnings:

- **Importance of User-Centered Design:** Focusing on the end-user experience is crucial for developing applications that are not only functional but also user-friendly.
- **Data Quality Management:** Handling and cleaning data effectively is essential for accurate forecasting and overall application performance.
- **Iterative Development:** Regular testing and user feedback are vital in refining the application and ensuring it meets the intended requirements.
- **Integration of Components:** Seamless integration of data processing, forecasting, and visualization components is key to creating a cohesive and efficient application.

Integrating the previously mentioned challenges into the "Future Work" section of your report can provide a clear roadmap for addressing these issues. Here's how you could present them professionally:

### Future Work

As we continue to refine and enhance the forecasting application, we will focus on overcoming the following challenges, which have been identified during the evaluation phase:

1. **Automation of Model Parameter Selection:**
  - **Goal:** Develop an automated system for selecting the optimal ARIMA model parameters ( $p$ ,  $d$ ,  $q$ ). This automation will reduce manual intervention,

streamline the modeling process, and potentially increase the accuracy and reliability of the forecasts.

2. **Enhanced Stationarity Checks:**

- **Goal:** Implement automated checks to rigorously assess the stationarity of the time series data. This enhancement is crucial for ensuring the appropriateness of the ARIMA modeling approach and for reducing the likelihood of model mis-specification.

3. **Handling Sudden Peak Values and Seasonality:**

- **Goal:** Improve the model's ability to accurately forecast in the presence of sudden peaks and seasonal variations. This will involve integrating more sophisticated algorithms or hybrid models that can better account for these complexities in the data.

In addition to addressing these specific challenges, our ongoing development efforts will also include:

4. **Advanced Forecasting Models:**

- Explore advanced machine learning models like LSTM networks for better handling of complex time series data.
- Develop hybrid models that combine various forecasting techniques to improve both accuracy and robustness.

5. **Enhanced User Interface and Usability:**

- Implement dynamic data loading capabilities to connect with live data sources for real-time forecasting.
- Provide customization options for visualization and model settings to cater to user-specific needs.

6. **Scalability and Performance Optimization:**

- Enhance the application's ability to process large datasets efficiently, possibly through parallel processing techniques.
- Develop a cloud-based deployment option to facilitate access and scalability.

7. **Expansion of Features:**

- Introduce a notification system to alert users of significant trends or anomalies.
- Add functionality for generating detailed reports that provide actionable insights from forecasts and historical data analysis.

# Team Contributions

## **Syed Ahmad Ali Shah**

- Explored the development of a Prediction System using the Prophet Forecasting library by Meta.
- Created a Model and conducted testing.
- Completed the section on Prophet Development and Testing Process in the report.

## **Benjamin Mensa-Bonsu**

- Created the UI and Interactivity graph for the ARIMA-based forecast model.
- Produced a detailed analysis of the user interface and interactivity graph of the ARIMA forecast model.
- Helped gather information from team members to create the final report.

## **Tomi Kolawole**

- Created the ARIMA model.
- Completed the section on ARIMA in the report.
- Helped gather information from team members to create the final report.
- Actively engaged in all group activities.

## **Hashim Ibrahim Thanalil**

- Played a role in improving the ARIMA model.
- Worked together on the creation of the GUI.
- Provided support in the compilation of the final report.
- Prepared demo materials for the presentation.

## **Muhammad Usama**

- Helped with the development and improvement of the ARIMA model.
- Played a role in front-end development.
- Successfully implemented the feedback system.

## **Mohit Bisht**

- Played a key role in the development and improvement of the ARIMA model.
- Played a role in back-end development.
- Implemented the feedback system as requested.

## References

1. Hyndman, R. J. and Athanasopoulos, G. (2021) 'Forecasting: principles and practice'. [Online] Available at: <https://otexts.com/fpp3/arma.html> [Accessed 17 Jul. 2024].
2. Plotly (2024) 'Plotly Python Graphing Library'. [Online] Available at: <https://plotly.com/python/> [Accessed 17 Jul. 2024].
3. Pandas (2024) 'pandas: powerful Python data analysis toolkit'. [Online] Available at: <https://pandas.pydata.org/pandas-docs/stable/> [Accessed 17 Jul. 2024].
4. TkDocs (2024) 'TkDocs - Tk Tutorial'. [Online] Available at: <https://tkdocs.com/tutorial/> [Accessed 17 Jul. 2024].
5. Seabold, S. and Perktold, J. (2010) 'statsmodels: Econometric and statistical modeling with python'. [Online] Available at: <https://www.statsmodels.org/stable/index.html> [Accessed 17 Jul. 2024].
6. Bazarevsky, A. (2024) 'tkcalendar: A Calendar and Date Entry Widget for Tkinter'. [Online] Available at: <https://github.com/j4321/tkcalendar> [Accessed 17 Jul. 2024].
7. Python Software Foundation (2024) 'Python Documentation'. [Online] Available at: <https://docs.python.org/3/> [Accessed 17 Jul. 2024].