
```

function varargout = CarParkGuiNew(varargin)
% CARPARKGUINEW MATLAB code for CarParkGuiNew.fig
%     CARPARKGUINEW, by itself, creates a new CARPARKGUINEW or raises
%     the existing
%     singleton*.
%
%     H = CARPARKGUINEW returns the handle to a new CARPARKGUINEW or
%     the handle to
%     the existing singleton*.
%
%     CARPARKGUINEW('CALLBACK',hObject,eventData,handles,...) calls
%     the local
%     function named CALLBACK in CARPARKGUINEW.M with the given input
%     arguments.
%
%     CARPARKGUINEW('Property','Value',...) creates a new
%     CARPARKGUINEW or raises the
%     existing singleton*. Starting from the left, property value
%     pairs are
%     applied to the GUI before CarParkGuiNew_OpeningFcn gets called.
%     An
%     unrecognized property name or invalid value makes property
%     application
%     stop. All inputs are passed to CarParkGuiNew_OpeningFcn via
%     varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
%     only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CarParkGuiNew

% Last Modified by GUIDE v2.5 04-Jan-2024 20:50:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @CarParkGuiNew_OpeningFcn, ...
                  'gui_OutputFcn',  @CarParkGuiNew_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});

```

```

end
% End initialization code - DO NOT EDIT

% --- Executes just before CarParkGuiNew is made visible.
function CarParkGuiNew_OpeningFcn(hObject, eventdata, handles,
    varargin)

% Choose default command line output for CarParkGuiNew
handles.output = hObject;
    % Load the image from our excel file
    img = imread('C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
folder\Images\carPark.jpg');

    % Make handles to put image on axes
    handles.axesImage = findobj('Tag', 'axesImage');

    % Display the image in the Axes
    imshow(img, 'Parent', handles.axesImage);

    % Update handles structure
    guidata(hObject, handles);

% UIWAIT makes CarParkGuiNew wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = CarParkGuiNew_OutputFcn(hObject, eventdata,
    handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes when entered data in editable cell(s) in uitable1.
function uitable1_CellEditCallback(hObject, eventdata, handles)
% hObject      handle to uitable1 (see GCBO)
% eventdata    structure with the following fields (see
    MATLAB.UI.CONTROL.TABLE)
% Indices: row and column indices of the cell(s) edited
% PreviousData: previous data for the cell(s) edited
% EditData: string(s) entered by the user
% NewData: EditData or its converted form set on the Data property.
    Empty if Data was not changed
% Error: error string when failed to convert EditData to appropriate
    value for Data
% handles      structure with handles and user data (see GUIDATA)

```

```

% --- Executes on button press in showbtn.
function showbtn_Callback(hObject, eventdata, handles)

% Specify the file path and name directly
fullFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
    folder\NewRecoeds.xlsx';

% Read data from the Excel file
try
    data = readtable(fullFilePath);
catch
    errordlg('Error reading the Excel file. Make sure the file is
        valid.','Error','modal');
    return;
end

% Display data in the uitable
set(handles.uitable1, 'Data', table2cell(data));

% --- Executes on button press in clearbtn.
function clearbtn_Callback(hObject, eventdata, handles)
% hObject    handle to clearbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Clear the content of the table
    set(handles.uitable1, 'Data', {});

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
%        as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%            called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% --- Executes on button press in entercarbtn is on gui show as Enter
Car.
function entercarbtn_Callback(hObject, eventdata, handles)
    % Get the entered number plate
    % edit1 is suppose to be car number plate editEdit
    numberPlate = get(handles.edit1, 'String');

    % Get the current table data uitable2 = Current Car park status
    currentData = get(handles.uitable2, 'Data');

    % Create a new entry
    newEntry = {numberPlate, datestr(datetime('now', 'Format', 'yyyy-MM-dd
        HH:mm:ss')), '', ''};

    % Display size information for debugging
    disp(['Size of currentData: ' num2str(size(currentData))]);
    disp(['Size of newEntry: ' num2str(size(newEntry))]);

    % Ensure the new entry has the same number of columns as the existing
    data
    if size(currentData, 2) ~= numel(newEntry)
        errordlg('Number of columns in the new entry does not match the
            existing data.', 'Error', 'modal');
        return;
    end

    % Concatenate new entry with existing data along the rows and columns
    updatedData = [newEntry; currentData];

    % Update the table in the GUI uitable2 = Current Car park status
    set(handles.uitable2, 'Data', updatedData);

    % Inform the user
    msgbox('Entry recorded successfully!', 'Success', 'modal');

% --- Executes on button press in exitcarbtn is on gui show as Exit
Car.
function exitcarbtn_Callback(hObject, eventdata, handles)
    % Get the entered number plate
    % edit2 is suppose to be car number plate editEdit
    numberPlate = get(handles.edit2, 'String');

    % Get the current table data uitable2 = Current Car park status
    currentData = get(handles.uitable2, 'Data');

    % Find the corresponding entry
    entryIndex = find(strcmp(currentData(:, 1), numberPlate), 1);

    if isempty(entryIndex)
        errordlg('No entry found for the given number
            plate.', 'Error', 'modal');
        return;
    end
end

```

```

% Update exit time
currentData{entryIndex, 3} = datestr(datetime('now', 'Format', 'yyyy-
MM-dd HH:mm:ss'));

% Calculate entry and exit times
entryTime = datetime(currentData{entryIndex, 2}, 'InputFormat', 'dd-
MMM-yyyy HH:mm:ss', 'Format', 'yyyy-MM-dd HH:mm:ss');
exitTime = datetime(currentData{entryIndex, 3}, 'InputFormat', 'dd-
MMM-yyyy HH:mm:ss', 'Format', 'yyyy-MM-dd HH:mm:ss');

% Calculate duration in hours
entryDurationHours = hours(exitTime - entryTime);

% Calculate charge amount with a cap of $20 per hour
chargeAmount = max(entryDurationHours * 20, 20);

% Update the charge amount in the table
currentData{entryIndex, 4} = chargeAmount;

% Update the table in the GUI uitable2 = Current Car parl status
set(handles.uitable2, 'Data', currentData);

% Inform the user
msgbox('Exit recorded successfully!', 'Success', 'modal');

function edit2_Callback(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%         str2double(get(hObject,'String')) returns contents of edit2
%         as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
%              called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes when entered data in editable cell(s) in uitable2.
function uitable2_CellEditCallback(hObject, eventdata, handles)
% hObject      handle to uitable2 (see GCBO)
% eventdata    structure with the following fields (see
%              MATLAB.UI.CONTROL.TABLE)

```

```

% Indices: row and column indices of the cell(s) edited
% PreviousData: previous data for the cell(s) edited
% EditData: string(s) entered by the user
% NewData: EditData or its converted form set on the Data property.
    Empty if Data was not changed
% Error: error string when failed to convert EditData to appropriate
    value for Data
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton10 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.uitable2, 'Data', {});

% --- Executes on button press in saveexbtn.
function saveexbtn_Callback(hObject, eventdata, handles)

% Get the current table data
    % Get the current table data
    currentData = get(handles.uitable2, 'Data');

    % Specify the Excel file path
    filePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
folder\NewRecoeds.xlsx';

    try
        % Check if the Excel file already exists
        if exist(filePath, 'file') == 2
            % Load existing data from Excel file
            existingData = readtable(filePath);

            % Convert 'ChargeAmount' to cell if it is a numeric array
in existing data
            if ~iscell(existingData.ChargeAmount)
                existingData.ChargeAmount =
num2cell(existingData.ChargeAmount);
            end

            % Convert cell array to table
            newData = cell2table(currentData, 'VariableNames',
{'CarPlate', 'EntryTime', 'ExitTime', 'ChargeAmount'});

            % Convert 'ChargeAmount' to cell if it is a numeric array
in new data
            if ~iscell(newData.ChargeAmount)
                newData.ChargeAmount = num2cell(newData.ChargeAmount);
            end

            % Append new data to existing data
            updatedData = [existingData; newData];

```

```

        % Write the updated table to the Excel file
        writetable(updatedData, filePath);
    else
        % If the file doesn't exist, create a new one
        tableData = cell2table(currentData, 'VariableNames',
{'CarPlate', 'EntryTime', 'ExitTime', 'ChargeAmount'});
        writetable(tableData, filePath);
    end

    msgbox('Data saved to Excel file
successfully!', 'Success', 'modal');
    catch exception
        % Display the specific error message in the command window
        disp(['Error saving data to Excel file: ' exception.message]);

        % Display a more informative error message to the user
        errordlg(['Error saving data to Excel file: '
exception.message], 'Error', 'modal');
    end

% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes on button press in Showavg.
function Showavg_Callback(hObject, eventdata, handles)

% File path to the Excel file
excelFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
folder\NewRecoeds.xlsx';

% Read data from Excel file
carParkTable = readtable(excelFilePath);

% Convert entryTime and exitTime to datetime
entryTime =
datetime(carParkTable.EntryTime, 'ConvertFrom', 'excel');
exitTime =
datetime(carParkTable.ExitTime, 'ConvertFrom', 'excel');

% Calculate parked duration in hours for each car
parkedDuration = hours(exitTime - entryTime); %#ok<NASGU>

% Calculate daily earnings
dailyEarnings = accumarray(day(entryTime),
carParkTable.ChargeAmount);

% Calculate total number of cars parked per day
totalCarsPerDay = accumarray(day(entryTime), 1);

```

```

% Calculate average earnings per car per day
avgEarningsPerCarPerDay = dailyEarnings ./ totalCarsPerDay;

% Find days with recorded data
validDays = find(~isnan(avgEarningsPerCarPerDay));

% Create a cell array to store the data for the uitable
uitableData = cell(numel(validDays), 2);

% Populate the uitableData with date and average earnings
for i = 1:numel(validDays)
    dayIndex = validDays(i);

    % Get the date corresponding to the dayIndex
    currentDate = entryTime(day(entryTime) == dayIndex);

    % Convert the date to a cell array of strings
    formattedDate = cellstr(datestr(currentDate, 'dddd, mmmm dd,
YYYY'));

    uitableData{i, 1} = formattedDate{1}; % Extract the first element
    uitableData{i, 2} = avgEarningsPerCarPerDay(dayIndex);
    uitableData{i, 3} = dailyEarnings(dayIndex);
end
% Display the uitable with date and average earnings
set(handles.uitable3, 'Data', uitableData);

% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton14 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.uitable3, 'Data', {});

% --- Executes on button press in plotavg mean plot total button on
gui
function plotavg_Callback(hObject, eventdata, handles)

% File path to the Excel file
excelFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
folder\NewRecoeds.xlsx';

% Read data from Excel file
carParkTable = readtable(excelFilePath);

% Convert entryTime and exitTime to datetime
entryTime = datetime(carParkTable.EntryTime, 'ConvertFrom', 'excel');
exitTime = datetime(carParkTable.ExitTime, 'ConvertFrom', 'excel');

% Calculate parked duration in hours for each car
parkedDuration = hours(exitTime - entryTime);

% Calculate daily earnings

```

```

dailyEarnings = accumarray(day(entryTime), carParkTable.ChargeAmount);

% Calculate total number of cars parked per day
totalCarsPerDay = accumarray(day(entryTime), 1);

% Calculate average earnings per car per day
avgEarningsPerCarPerDay = dailyEarnings ./ totalCarsPerDay;

% Find days with recorded data
validDays = find(~isnan(avgEarningsPerCarPerDay));

% Plot the average earnings over days using a bar plot
figure;
bar(round(validDays),
    round(avgEarningsPerCarPerDay(validDays)), 'g', 'LineWidth', 1);

xlabel('Day');
ylabel('Average Earnings per Car');
title('Average Earnings per Car over Days (Bar Plot)');
grid on;
legend('Average Earnings');

% --- Executes on button press in showsum.
function showsum_Callback(hObject, eventdata, handles)
% hObject      handle to showsum (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% File path to the Excel file

%--- Executes on button press in plottotal.
function plottotal_Callback(hObject, eventdata, handles)

excelFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
    folder\NewRecoeds.xlsx';

% Read data from Excel file
carParkTable = readtable(excelFilePath);

% Convert entryTime and exitTime to datetime
entryTime = datetime(carParkTable.EntryTime, 'ConvertFrom', 'excel');
exitTime = datetime(carParkTable.ExitTime, 'ConvertFrom', 'excel');

% Calculate parked duration in hours for each car
parkedDuration = hours(exitTime - entryTime);

% Calculate daily earnings
dailyEarnings = accumarray(day(entryTime), carParkTable.ChargeAmount);

% Find days with recorded data
validDays = find(~isnan(dailyEarnings));

% Create mesh grid for X and Y values
[X, Y] = meshgrid(1:31, validDays);

```

```

% Create a waterfall plot
figure;
waterfall(X, Y, repmat(dailyEarnings(validDays), 1, 31)');
xlabel('Day');
ylabel('Recorded Days');
zlabel('Total Earnings per Car');
title('Total Earnings per Car over Days (Waterfall Plot)');
legend('Total Earnings');

% --- Executes on button press in subplotbtn mean plot3 min and max
      button on gui
function subplotbtn_Callback(hObject, eventdata, handles)
% File path to the Excel file
    excelFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
    folder\NewRecoeds.xlsx';

    % Read data from Excel file
    carParkTable = readtable(excelFilePath);

    % Convert entryTime to datetime
    entryTime =
datetime(carParkTable.EntryTime, 'ConvertFrom', 'excel');

    % Calculate daily earnings
    dailyEarnings = accumarray(day(entryTime),
carParkTable.ChargeAmount, [], @(x) {x});

    % Find days with recorded data
    validDays = find(~cellfun('isempty', dailyEarnings));

    % Initialize arrays to store min and max earnings for each day
    minEarnings = zeros(length(validDays), 1);
    maxEarnings = zeros(length(validDays), 1);

    % Calculate min and max earnings for each day
    for i = 1:length(validDays)
        dayIndex = validDays(i);
        earningsForDay = dailyEarnings{dayIndex};
        minEarnings(i) = min(earningsForDay);
        maxEarnings(i) = max(earningsForDay);
    end

    % Create separate subplots for min and max earnings
    figure;

    % Plot for Min Earnings
    subplot(2, 1, 1);
    bar(validDays, minEarnings, 'b', 'LineWidth', 2);
    xlabel('Day');
    ylabel('Min Earnings per Car');
    title('Min Earnings per Car over Days');
    grid on;

```

```

    % Plot for Max Earnings
    subplot(2, 1, 2);
    bar(validDays, maxEarnings, 'r', 'LineWidth', 2);
    xlabel('Day');
    ylabel('Max Earnings per Car');
    title('Max Earnings per Car over Days');
    grid on;

% --- Executes on button press in showrevenuebtn mean show total
revenue button in gui
function showrevenuebtn_Callback(hObject, eventdata, handles)

% Specify the Excel file path
excelFilePath = 'C:\Users\ruchi\OneDrive\Desktop\MatlabProject\New
    folder\NewRecoeds.xlsx';

% Read charge amounts from Excel file
chargeAmounts = xlsread(excelFilePath);

% Assuming charge amounts are in dollars per hour, and you have the
time intervals
timeIntervals = 1:length(chargeAmounts);

% Use trapz to approximate the total revenue
totalRevenue = trapz(timeIntervals, chargeAmounts);

% Display the result in the label
set(handles.intlbl, 'String', sprintf('Total Revenue: $%.2f',
    totalRevenue));

%--- Executes on button press in AnualRevenue.
function RequiredAnualRevenue_Callback(hObject, eventdata, handles)

% Call the function from the other file and get the result string
resultString = calculateAnnualRevenue();

    disp(resultString);
% Display the result in a msgbox
msgbox(resultString, 'Revenue and Profit Information', 'modal');

% --- Executes on button press in areabtn.
function areabtn_Callback(hObject, eventdata, handles)
    % Call the CarParkAndCARSpaceArea function and get the result
string
    resultString = CarParkAndCARSpaceArea();

    % Use strvcats to concatenate the result string
    combinedString = strvcats(resultString);

    % Display the result string in a message box
    msgbox(combinedString, 'Area Calculation', 'modal');

```

CarParkGuiNew

Car Park Main Dashboard

Car Enter :

Car Number Plate

Enter Car

Car Exit :

Car Number Plate

Exit Car

Current Car Park status

	Car Plate	Entry Date and Time	Exit Date and Time	Charge Amount
1				
2				
3				
4				

Save to excel

Clear

Show Average and Total per day

	Date	Average	Total
1			
2			
3			

Show Status

Clear

Plot Average

Plot Total

SubPlot Max and Min Per Day

Show Total Revenue

Investors Question

Required Annual Revenue

CarPark and CarSpace Area

Check Car Park Status From File

	Car Plate	Entry Date and Time	Exit Date and Time	Charge Amount
1				
2				
3				
4				

Show Status

Clear

Published with MATLAB® R2016a