```
classdef FUNMAP_Output < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TabGroup                        matlab.ui.container.TabGroup
        WaveHeightMapTab                matlab.ui.container.Tab
        GeneralLayoutLabel_3            matlab.ui.control.Label
        Panel_19                        matlab.ui.container.Panel
        ButtonGroup_11                  matlab.ui.container.ButtonGroup
        PlotseparatelyButton_2          matlab.ui.control.RadioButton
        PlotalldatainonefigureButton    matlab.ui.control.RadioButton
        SettoDefaultButton_2            matlab.ui.control.Button
        CloseFiguresButton_4            matlab.ui.control.Button
        FigureSizeLabel_2               matlab.ui.control.Label
        ThicknessLabel_9                matlab.ui.control.Label
        Width_2                         matlab.ui.control.NumericEditField
        AutoSetCheckBox_2               matlab.ui.control.CheckBox
        ThicknessLabel_8                matlab.ui.control.Label
        Height_2                        matlab.ui.control.NumericEditField
        SouthEditField                  matlab.ui.control.NumericEditField
        SouthEditFieldLabel             matlab.ui.control.Label
        NorthEditField                  matlab.ui.control.NumericEditField
        NorthEditFieldLabel             matlab.ui.control.Label
        WestEditField                   matlab.ui.control.NumericEditField
        WestEditFieldLabel              matlab.ui.control.Label
        EastEditField                   matlab.ui.control.NumericEditField
        EastEditFieldLabel              matlab.ui.control.Label
        PlotindegreesCheckBox_2         matlab.ui.control.CheckBox
        BoundaryLimitsLabel             matlab.ui.control.Label
        GENERATEButton                  matlab.ui.control.Button
        SaveMapLabel                    matlab.ui.control.Label
        Panel_6                         matlab.ui.container.Panel
        mp4CheckBox                     matlab.ui.control.CheckBox
        FramerateEditField_2            matlab.ui.control.NumericEditField
        FramerateEditField_2Label       matlab.ui.control.Label
        OutputDirectoryEditField        matlab.ui.control.EditField
        OutputDirectoryLabel            matlab.ui.control.Label
        Button_2                        matlab.ui.control.Button
        tifCheckBox_2                   matlab.ui.control.CheckBox
        jpgCheckBox                     matlab.ui.control.CheckBox
        FileFormatLabel                 matlab.ui.control.Label
        pngCheckBox                     matlab.ui.control.CheckBox
        BasemapLabel                    matlab.ui.control.Label
        Panel_5                         matlab.ui.container.Panel
        ColorbarTextSize                matlab.ui.control.NumericEditField
        TextSizeLabel_2                 matlab.ui.control.Label
        CoastlinecolourDropDown_2       matlab.ui.control.DropDown
        FlipBasemapLabel_2              matlab.ui.control.Label
        VerticalCheckBox                matlab.ui.control.CheckBox
        HorizontalCheckBox              matlab.ui.control.CheckBox
        CoastlinecolourDropDownLabel_9  matlab.ui.control.Label
        mLabel_2                        matlab.ui.control.Label
        mLabel                          matlab.ui.control.Label
        toLabel_2                       matlab.ui.control.Label
        MinEditField_2                  matlab.ui.control.NumericEditField
        MaxEditField_2                  matlab.ui.control.NumericEditField
        DivisionLabel                   matlab.ui.control.Label
        ColorinterpolationEditField_2   matlab.ui.control.NumericEditField
        FlipCheckBox_2                  matlab.ui.control.CheckBox
        CoastlinecolourDropDown_7       matlab.ui.control.DropDown
        etaLabel                        matlab.ui.control.Label
        toLabel                         matlab.ui.control.Label
        CoastlinecolourDropDownLabel_10  matlab.ui.control.Label
        toEditField                     matlab.ui.control.NumericEditField
        MinEditField                    matlab.ui.control.NumericEditField
        hmaxLabel                       matlab.ui.control.Label
        CoastlinecolourDropDownLabel_8  matlab.ui.control.Label
        CoastlinecolourDropDown_8       matlab.ui.control.DropDown
        FlipCheckBox_5                  matlab.ui.control.CheckBox
        DivisionEditField               matlab.ui.control.NumericEditField
        DivisionEditFieldLabel          matlab.ui.control.Label
        OverlayFeaturesLabel            matlab.ui.control.Label
        Panel_7                         matlab.ui.container.Panel
        GaugesCheckBox                  matlab.ui.control.CheckBox
        ButtonGroup_17                  matlab.ui.container.ButtonGroup
        NoneButton                      matlab.ui.control.RadioButton
        ArrivalTimeButton               matlab.ui.control.RadioButton
        WaveHeightButton                matlab.ui.control.RadioButton
        BathymetryButton_2              matlab.ui.control.RadioButton
        TabGroup2                       matlab.ui.container.TabGroup
        BathymetryTab                   matlab.ui.container.Tab
        IntervalEditField_3             matlab.ui.control.NumericEditField
        LabelSpacingEditFieldLabel      matlab.ui.control.Label
        SpacingEditField_4Label_4       matlab.ui.control.Label
        TextLabelSize                   matlab.ui.control.NumericEditField
        AddLabelsCheckBox_2             matlab.ui.control.CheckBox
        ColorDropDown                   matlab.ui.control.DropDown
        ColorDropDownLabel              matlab.ui.control.Label
        StyleDropDown                   matlab.ui.control.DropDown
        StyleDropDownLabel              matlab.ui.control.Label
        LineLabel                       matlab.ui.control.Label
        WidthEditField                  matlab.ui.control.NumericEditField
        WidthEditFieldLabel             matlab.ui.control.Label
        LabelSpacingEditField           matlab.ui.control.NumericEditField
        LabelSpacingEditFieldLabel_2    matlab.ui.control.Label
        DepthRangeLabel                 matlab.ui.control.Label
        IntervalEditField_5             matlab.ui.control.NumericEditField
        IntervalEditField_5Label        matlab.ui.control.Label
        MinimumEditField                matlab.ui.control.NumericEditField
        MinimumEditFieldLabel           matlab.ui.control.Label
        MaximumEditField                matlab.ui.control.NumericEditField
        MaximumEditFieldLabel           matlab.ui.control.Label
        WaveHeightTab                   matlab.ui.container.Tab
        wh_interval                     matlab.ui.control.NumericEditField
        LabelSpacingEditField_2Label    matlab.ui.control.Label
        LabelSpacingEditField_2         matlab.ui.control.NumericEditField
        LabelSpacingEditField_2Label_2  matlab.ui.control.Label
        SpacingEditField_4Label_5       matlab.ui.control.Label
        TextLabelSize_2                 matlab.ui.control.NumericEditField
        AddLabelsCheckBox_3             matlab.ui.control.CheckBox
        StyleDropDown_2                 matlab.ui.control.DropDown
        StyleDropDown_2Label            matlab.ui.control.Label
        ColorDropDown_2                 matlab.ui.control.DropDown
        ColorDropDown_2Label            matlab.ui.control.Label
        ThicknessEditField 2            matlab.ui.control.NumericEditField
        ThicknessEditField_2Label       matlab.ui.control.Label
        LineLabel_2                     matlab.ui.control.Label
        ContourRangeLabel_2             matlab.ui.control.Label
        MaximumdepthEditFieldLabel_3    matlab.ui.control.Label
        MinimumdepthEditFieldLabel_2    matlab.ui.control.Label
        MaximumdepthEditFieldLabel_2    matlab.ui.control.Label
        LineintervalEditField_2         matlab.ui.control.NumericEditField
        MaximumdepthEditField_2         matlab.ui.control.NumericEditField
        MinimumdepthEditField_2         matlab.ui.control.NumericEditField
        ArrivalTimeTab                  matlab.ui.container.Tab
        LabelSpacingEditField_2Label_3  matlab.ui.control.Label
        LabelSpacingEditField_4         matlab.ui.control.NumericEditField
        StyleDropDown_4                 matlab.ui.control.DropDown
        StyleDropDown_4Label_2          matlab.ui.control.Label
        ColorDropDown_6                 matlab.ui.control.DropDown
        ColorDropDown_6Label            matlab.ui.control.Label
        ThicknessEditField_2Label_2     matlab.ui.control.Label
        ThicknessEditField_3            matlab.ui.control.NumericEditField
        LineLabel_5                     matlab.ui.control.Label
        AddLabelsCheckBox_5             matlab.ui.control.CheckBox
        SpacingEditField_4Label_6       matlab.ui.control.Label
        TextLabelSize_3                 matlab.ui.control.NumericEditField
        IntervalEditField_7             matlab.ui.control.NumericEditField
        LabelSpacingEditField_2Label_4  matlab.ui.control.Label
        MinimumdepthEditFieldLabel_5    matlab.ui.control.Label
        MinimumdepthEditField_3         matlab.ui.control.NumericEditField
        MaximumdepthEditFieldLabel_10   matlab.ui.control.Label
        LineintervalEditField_3         matlab.ui.control.NumericEditField
        MaximumdepthEditFieldLabel_9    matlab.ui.control.Label
        MaximumdepthEditField_3         matlab.ui.control.NumericEditField
        ContourRangeLabel               matlab.ui.control.Label
        FileEditField_3                 matlab.ui.control.EditField
        FileEditField_3Label            matlab.ui.control.Label
        TimeFileButton                  matlab.ui.control.Button
        GaugesTab                       matlab.ui.container.Tab
        GLspacing                       matlab.ui.control.NumericEditField
        GLspacingLabel                  matlab.ui.control.Label
        SizeEditField_3                 matlab.ui.control.NumericEditField
        SizeEditField_3Label            matlab.ui.control.Label
        CoastlinecolourDropDown_5       matlab.ui.control.DropDown
        CoastlinecolourDropDown_4       matlab.ui.control.DropDown
        AlignmentLabel                  matlab.ui.control.Label
        AddLabelsCheckBox_4             matlab.ui.control.CheckBox
        SizeEditField                   matlab.ui.control.NumericEditField
        SizeEditFieldLabel              matlab.ui.control.Label
        CoastlinecolourDropDown_3       matlab.ui.control.DropDown
        CoastlinecolourDropDownLabel_3  matlab.ui.control.Label
        ColorDropDown_3                 matlab.ui.control.DropDown
        ColorDropDown_3Label            matlab.ui.control.Label
        FileEditField                   matlab.ui.control.EditField
        GaugeFileLabel                  matlab.ui.control.Label
        Button_8                        matlab.ui.control.Button
        Tab                             matlab.ui.container.Tab
        InputDataLabel                  matlab.ui.control.Label
        Panel_3                         matlab.ui.container.Panel
        gridY                           matlab.ui.control.NumericEditField
        YEditField_2Label               matlab.ui.control.Label
        gridX                           matlab.ui.control.NumericEditField
        XEditField_2Label               matlab.ui.control.Label
        GridSizeLabel_2                 matlab.ui.control.Label
        StartLabel                      matlab.ui.control.Label
        StartTime                       matlab.ui.control.NumericEditField
        IntervalLabel                   matlab.ui.control.Label
        TotalSimuilationTimesecEditField_3  matlab.ui.control.NumericEditField
        TotalSimuilationTimesecEditFieldLabel_3  matlab.ui.control.Label
        BathymetryLabel                 matlab.ui.control.Label
        FilesLabel                      matlab.ui.control.Label
        DepthFileEditField              matlab.ui.control.EditField
        Button_15                       matlab.ui.control.Button
        LatitudeEditField               matlab.ui.control.NumericEditField
        LatitudeEditFieldLabel          matlab.ui.control.Label
        LongitudeEditField              matlab.ui.control.NumericEditField
        LongitudeEditFieldLabel         matlab.ui.control.Label
        SouthwestCornerLabel            matlab.ui.control.Label
        Button_14                       matlab.ui.control.Button
        FileTextArea                    matlab.ui.control.TextArea
        GaugeRecordsTab                 matlab.ui.container.Tab
        PlotStyleandLayoutLabel         matlab.ui.control.Label
        SavePlotLabel                   matlab.ui.control.Label
        Panel_13                        matlab.ui.container.Panel
        epsCheckBox                     matlab.ui.control.CheckBox
        txtCheckBox                     matlab.ui.control.CheckBox
        OutputDirectoryEditField_2      matlab.ui.control.EditField
        OutputDirectoryLabel_3          matlab.ui.control.Label
        Button_18                       matlab.ui.control.Button
        pdfCheckBox_2                   matlab.ui.control.CheckBox
        jpgCheckBox_2                   matlab.ui.control.CheckBox
        FileFormatLabel_2               matlab.ui.control.Label
        pngCheckBox_2                   matlab.ui.control.CheckBox
        InputDataLabel_3                matlab.ui.control.Label
        Panel_11                        matlab.ui.container.Panel
        XaxisUseDataDropDown            matlab.ui.control.DropDown
        YaxisUseDataDropDown            matlab.ui.control.DropDown
        YaxisSetcolumncountLabel        matlab.ui.control.Label
        Label_11                        matlab.ui.control.Label
        Label_10                        matlab.ui.control.Label
        Label_7                         matlab.ui.control.Label
        Label_6                         matlab.ui.control.Label
        Label_5                         matlab.ui.control.Label
        denomeratorX                    matlab.ui.control.NumericEditField
        numeratorX                      matlab.ui.control.NumericEditField
        CheckBox_4                      matlab.ui.control.CheckBox
        CheckBox_3                      matlab.ui.control.CheckBox
        YaxisEditFieldLabel             matlab.ui.control.Label
        Label_9                         matlab.ui.control.Label
        denomeratorY                    matlab.ui.control.NumericEditField
        numeratorY                      matlab.ui.control.NumericEditField
        ConvertLabel                    matlab.ui.control.Label
        XaxisdataLabel                  matlab.ui.control.Label
        SelectColumnLabel               matlab.ui.control.Label
        XaxisLabel                      matlab.ui.control.Label
        Button_16                       matlab.ui.control.Button
        FileTextArea_2                  matlab.ui.control.TextArea
        FilesLabel_2                    matlab.ui.control.Label
        PLOTButton                      matlab.ui.control.Button
        Panel 10                        matlab.ui.container.Panel
        MarkerDropDown                  matlab.ui.control.DropDown
        MarkerDropDownLabel             matlab.ui.control.Label
        ThicknessLabel_4                matlab.ui.control.Label
        LegendSize                      matlab.ui.control.NumericEditField
        LocationDropDown                matlab.ui.control.DropDown
        LocationDropDownLabel           matlab.ui.control.Label
        legendfirsttext                 matlab.ui.control.EditField
        legendfirsttextlabel            matlab.ui.control.Label
        LegendLabel                     matlab.ui.control.Label
        ThicknessLabel_12               matlab.ui.control.Label
        LineYThickness_2                matlab.ui.control.NumericEditField
        CloseFiguresButton_2            matlab.ui.control.Button
        AutoSetCheckBox                 matlab.ui.control.CheckBox
        ThicknessLabel_7                matlab.ui.control.Label
        Width                           matlab.ui.control.NumericEditField
        FigureSizeLabel                 matlab.ui.control.Label
        ThicknessLabel_6                matlab.ui.control.Label
        Height                          matlab.ui.control.NumericEditField
        FlipCheckBox_3                  matlab.ui.control.CheckBox
        GaugeLineColor_multiple         matlab.ui.control.DropDown
        GridLabelSize                   matlab.ui.control.NumericEditField
        ThicknessLabel_5                matlab.ui.control.Label
        ThicknessLabel_3                matlab.ui.control.Label
        LineYThickness                  matlab.ui.control.NumericEditField
        Lineaty0DropDown                matlab.ui.control.DropDown
        Lineaty0DropDownLabel           matlab.ui.control.Label
        GridStyle                       matlab.ui.control.DropDown
        StyleDropDown_4Label            matlab.ui.control.Label
        BackgroundGridLabel             matlab.ui.control.Label
        XAxisEditField                  matlab.ui.control.EditField
        YAxisLabelEditField_2Label      matlab.ui.control.Label
        XandYLimitsLabel_2              matlab.ui.control.Label
        YAxisEditField                  matlab.ui.control.EditField
        LabelEditFieldLabel             matlab.ui.control.Label
        XAutoSetLabel                   matlab.ui.control.Label
        YAutoSetLabel                   matlab.ui.control.Label
        CheckBox                        matlab.ui.control.CheckBox
        yMaxLimit                       matlab.ui.control.NumericEditField
        MinimumdepthEditFieldLabel_3    matlab.ui.control.Label
        yMinLimit                       matlab.ui.control.NumericEditField
        MaximumdepthEditFieldLabel_8    matlab.ui.control.Label
        yLimInterval                    matlab.ui.control.NumericEditField
        MaximumdepthEditFieldLabel_4    matlab.ui.control.Label
        CheckBox_2                      matlab.ui.control.CheckBox
        MaximumdepthEditFieldLabel_6    matlab.ui.control.Label
        xLimInterval                    matlab.ui.control.NumericEditField
        MaximumdepthEditFieldLabel_5    matlab.ui.control.Label
        xMaxLimit                       matlab.ui.control.NumericEditField
        MinimumdepthEditFieldLabel_4    matlab.ui.control.Label
        xMinLimit                       matlab.ui.control.NumericEditField
        TextArea2_4                     matlab.ui.control.TextArea
        AxesLabels                      matlab.ui.control.Label
        WidthLabel                      matlab.ui.control.Label
        DataLineWidth                   matlab.ui.control.NumericEditField
        ButtonGroup_10                  matlab.ui.container.ButtonGroup
        PlotseparatelyButton_4          matlab.ui.control.RadioButton
        PlotalldatainonegraphButton     matlab.ui.control.RadioButton
        GaugeLineStyle                  matlab.ui.control.DropDown
        LineStyleLabel                  matlab.ui.control.Label
        GaugeLineColor                  matlab.ui.control.DropDown
        ColorDropDownLabel_2            matlab.ui.control.Label
        PlotPropertiesLabel             matlab.ui.control.Label
        VelocityMapTab                  matlab.ui.container.Tab
        GeneralLayoutLabel_2            matlab.ui.control.Label
        Panel_18                        matlab.ui.container.Panel
        ButtonGroup_12                  matlab.ui.container.ButtonGroup
        PlotseparatelyButton_3          matlab.ui.control.RadioButton
        PlotalldatainonefigureButton_2  matlab.ui.control.RadioButton
        SettoDefaultButton              matlab.ui.control.Button
        CloseFiguresButton_3            matlab.ui.control.Button
        FigureSizeLabel_3               matlab.ui.control.Label
        ThicknessLabel_11               matlab.ui.control.Label
        Width_3                         matlab.ui.control.NumericEditField
        AutoSetCheckBox_3               matlab.ui.control.CheckBox
        ThicknessLabel_10               matlab.ui.control.Label
        Height_3                        matlab.ui.control.NumericEditField
        SouthEditField_2                matlab.ui.control.NumericEditField
        SouthEditField_2Label           matlab.ui.control.Label
        NorthEditField_2                matlab.ui.control.NumericEditField
        NorthEditField_2Label           matlab.ui.control.Label
        WestEditField_2                 matlab.ui.control.NumericEditField
        WestEditField_2Label            matlab.ui.control.Label
        EastEditField_2                 matlab.ui.control.NumericEditField
        EastEditField_2Label            matlab.ui.control.Label
        PlotindegreesCheckBox           matlab.ui.control.CheckBox
        BoundaryLimitsLabel_2           matlab.ui.control.Label
        BasemapandOverlaysLabel         matlab.ui.control.Label
        SaveMapLabel_2                  matlab.ui.control.Label
        Panel_17                        matlab.ui.container.Panel
        tifCheckBox                     matlab.ui.control.CheckBox
        txtCheckBox_2                   matlab.ui.control.CheckBox
        OutputDirectoryEditField_3      matlab.ui.control.EditField
        OutputDirectoryLabel_2          matlab.ui.control.Label
        mp4CheckBox_2                   matlab.ui.control.CheckBox
        FramerateEditField              matlab.ui.control.NumericEditField
        FramerateEditFieldLabel         matlab.ui.control.Label
        Button_26                       matlab.ui.control.Button
        jpgCheckBox_3                   matlab.ui.control.CheckBox
        FileFormatLabel_3               matlab.ui.control.Label
        pngCheckBox_3                   matlab.ui.control.CheckBox
        Panel_16                        matlab.ui.container.Panel
        TabGroup3                       matlab.ui.container.TabGroup
        BasemapTab                      matlab.ui.container.Tab
        LandColor                       matlab.ui.control.DropDown
        CoastlinecolourDropDownLabel_12  matlab.ui.control.Label
        FlipBasemapLabel                matlab.ui.control.Label
        VerticalCheckBox_2              matlab.ui.control.CheckBox
        HorizontalCheckBox_2            matlab.ui.control.CheckBox
        FileTextArea_5                  matlab.ui.control.TextArea
        ColorbarLabel                   matlab.ui.control.Label
        ColormapLabel                   matlab.ui.control.Label
        ThirdTabColorMapPanel           matlab.ui.container.Panel
        TextSizeLabel                   matlab.ui.control.Label
        LimitsLabel                     matlab.ui.control.Label
        MaxBarValue                     matlab.ui.control.NumericEditField
        mLabel 3                        matlab.ui.control.Label
        toLabel_3                       matlab.ui.control.Label
        MinBarValue                     matlab.ui.control.NumericEditField
        InterpolationDivisionEditField_2  matlab.ui.control.NumericEditField
        InterpolationDivisionEditFieldLabel_2  matlab.ui.control.Label
        FlipCheckBox_4                  matlab.ui.control.CheckBox
        BackgroundMapColorDropDown      matlab.ui.control.DropDown
        maplabelsize                    matlab.ui.control.NumericEditField
        ButtonGroup_16                  matlab.ui.container.ButtonGroup
        VorticityButton                 matlab.ui.control.RadioButton
        BathymetryButton                matlab.ui.control.RadioButton
        hmaxButton                      matlab.ui.control.RadioButton
        etaButton                       matlab.ui.control.RadioButton
        VelocityButton                  matlab.ui.control.RadioButton
        BathymetryContoursTab_2         matlab.ui.container.Tab
        IntervalEditField_8             matlab.ui.control.NumericEditField
        SpacingEditField                matlab.ui.control.NumericEditField
        SpacingEditField_4Label_2       matlab.ui.control.Label
        SpacingEditField_4Label_3       matlab.ui.control.Label
        LabelSizeCont                   matlab.ui.control.NumericEditField
        IntervalEditField_8Label        matlab.ui.control.Label
        WidthEditField_2                matlab.ui.control.NumericEditField
        WidthEditField_2Label           matlab.ui.control.Label
        StyleDropDown_3                 matlab.ui.control.DropDown
        StyleDropDown_3Label_2          matlab.ui.control.Label
        ColorDropDown_4                 matlab.ui.control.DropDown
        ColorDropDown_4Label            matlab.ui.control.Label
        AddLabelCheckBox_2              matlab.ui.control.CheckBox
        LineLabel_4                     matlab.ui.control.Label
        IntervalEditField_6             matlab.ui.control.NumericEditField
        IntervalEditField_6Label        matlab.ui.control.Label
        MaximumEditField_2              matlab.ui.control.NumericEditField
        MaximumEditField_2Label         matlab.ui.control.Label
        MinimumEditField_2              matlab.ui.control.NumericEditField
        MinimumEditField_2Label         matlab.ui.control.Label
        PlotBathymetryContoursCheckBox  matlab.ui.control.CheckBox
        RangeLabel                      matlab.ui.control.Label
        GaugesTab_2                     matlab.ui.container.Tab
        SizeEditField_2                 matlab.ui.control.NumericEditField
        SizeEditField_2Label            matlab.ui.control.Label
        CoastlinecolourDropDown_11      matlab.ui.control.DropDown
        CoastlinecolourDropDown_10      matlab.ui.control.DropDown
        AlignmentLabel_2                matlab.ui.control.Label
        GLspacingLabel_2                matlab.ui.control.Label
        GLspacing_2                     matlab.ui.control.NumericEditField
        FontSizeEditField_2             matlab.ui.control.NumericEditField
        FontSizeEditField_2Label        matlab.ui.control.Label
        AddLabelCheckBox                matlab.ui.control.CheckBox
        PlotGaugesCheckBox              matlab.ui.control.CheckBox
        FileEditField_2                 matlab.ui.control.EditField
        GaugeFileLabel_2                matlab.ui.control.Label
        Button_27                       matlab.ui.control.Button
        ColorDropDown_7                 matlab.ui.control.DropDown
        ColorDropDown_7Label            matlab.ui.control.Label
        gaugemarkerVelocityTab          matlab.ui.control.DropDown
        CoastlinecolourDropDownLabel_19  matlab.ui.control.Label
        ArrowsLabel                     matlab.ui.control.Label
        InputDataLabel_4                matlab.ui.control.Label
        Panel 15                        matlab.ui.container.Panel
        Button_19                       matlab.ui.control.Button
        FileTextArea_4                  matlab.ui.control.TextArea
        gridX_2                         matlab.ui.control.NumericEditField
        XEditField_2Label_2             matlab.ui.control.Label
        TotalSimuilationTimesecEditField_4  matlab.ui.control.NumericEditField
        IntervalLabel_2                 matlab.ui.control.Label
        StartLabel_2                    matlab.ui.control.Label
        StartTime2                      matlab.ui.control.NumericEditField
        TotalSimuilationTimesecEditFieldLabel_4  matlab.ui.control.Label
        gridY_2                         matlab.ui.control.NumericEditField
        YEditField_2Label_2             matlab.ui.control.Label
        GridSizeLabel                   matlab.ui.control.Label
        LatitudeEditField_2             matlab.ui.control.NumericEditField
        LatitudeEditField_2Label        matlab.ui.control.Label
        LongitudeEditField_2            matlab.ui.control.NumericEditField
        LongitudeEditField_2Label       matlab.ui.control.Label
        SouthwestCornerLabel_2          matlab.ui.control.Label
        Button_22                       matlab.ui.control.Button
        BathymetryLabel_2               matlab.ui.control.Label
        DepthFileEditField_2            matlab.ui.control.EditField
        UvectorsLabel                   matlab.ui.control.Label
        FileTextArea_3                  matlab.ui.control.TextArea
        Panel_14                        matlab.ui.container.Panel
        ArrowSpacing                    matlab.ui.control.NumericEditField
        SpacingLabel                    matlab.ui.control.Label
        PlotvectorsCheckBox             matlab.ui.control.CheckBox
        ArrowHeadSize                   matlab.ui.control.NumericEditField
        ArrowSizeLabel                  matlab.ui.control.Label
        ColorLabel_2                    matlab.ui.control.Label
        ArrowThickness                  matlab.ui.control.NumericEditField
        ArrowThicknessLabel             matlab.ui.control.Label
        arrowscale                      matlab.ui.control.NumericEditField
        XEditField_2Label_3             matlab.ui.control.Label
        QuiverColorDropDown             matlab.ui.control.DropDown
        GENERATEButton_2                matlab.ui.control.Button
    end


    properties (Access = private)
        ArrivalTimeConverted
        ArrivalTimename
        bathyCol
        bathymetrycheck
        bathyRow
        colFirstFile
        ETAfilenames
        ETAfullfile
        FigureDirectoryVector
        FileINPUT
        FileINPUT1
        FileINPUT2
        FileINPUT3
        FileINPUT4
        FileNAME
        FileNAME1
        FileNAME2
        FileNAME3
        FileNAME4
        FileNAME5
        GAUGEFILE
        GAUGEFILE2
        GAUGEFILE3
        GaugeDirectorylist
        GaugeFname
        GaugeFname3
        GaugeNames
        HMAXfilenames
        HMAXfullfile
        INfile
        LegendGauge
        LegendLocation
        MainDirectoryVector
        STATION_FILE
        Ufiles
        Uvectorfullfile
        Vfile
        Vvectorfilenames
        Vvectorfullfile
        WORKFOLDER0
        WORKFOLDER
        WORKFOLDER2
        WORKFOLDER3
        WORKFOLDER4
        WORKFOLDER5
        XCOLGAUGE
        YCOLGAUGE
        bathymetrydata
        bathymetrydata2
        bathymetryinputdata
        bathymetryinputdata2
        bathymetryname
        bathymetryname2
        col1
        col1vel
        colorcombi
        fin
        inputtxt_directory0
        inputtxt_directory1
        LandAreaColor
        latGauge
        latGauge2
        longGauge
        longGauge2
        row1
        row1vel
        rowFirstFile
        vectorpath
        x
        xcoord
        y
        ycoord
    end


    % Callbacks that handle component events
    methods (Access = private)
```

```

   % Code that executes after component creation
   function startupFcn(app)
%Set the screenshot mode to 'manual' to disable automatic screenshots,
%to improve app efficiency
       currentAppModel.MetadataModel.ScreenshotMode = 'manual';

   end

   % Button pushed function: Button_14
   function Button_14Pushed(app, event)
      %Disable all the buttons
       app.CoastlinecolourDropDown_8.Enable = "off";
       app.FlipCheckBox_5.Enable = "off";
       app.toEditField.Enable = "off";
       app.MinEditField.Enable = "off";
       app.CoastlinecolourDropDown_7.Enable = "off";
       app.FlipCheckBox_2.Enable = "off";
       app.MaxEditField_2.Enable = "off";
       app.MinEditField_2.Enable = "off";
       app.DivisionEditField.Enable = "off";
       app.DivisionEditFieldLabel.Enable = "off";
       app.DivisionLabel.Enable = "off";
       app.ColorinterpolationEditField_2.Enable = "off";
       drawnow


       %Find files to import from a directory
       [files, path] = uigetfile('*.*', 'Select the files', 'MultiSelect', 'on');
       if isequal(files, 0) || isequal(path, 0)
           return;
       end

       cd(fullfile(path))

       if ischar(files) % Only 1 file is uploaded
           files = {files};
       end

       %Sort the files by snapshot timing, and display the list of filenames in the app's textbox
       app.FileNAME = sort(string(files));
       allfiles = fullfile(path, app.FileNAME);
       app.FileINPUT = allfiles;
       app.FileTextArea.Value = app.FileNAME; %
       app.FileTextArea.FontColor = 'k';


       %Set variables for tracking the file types
       containsHmax = false;
       containsEta = false;
       containsTime = false;

       %Verify that each file is readable
       for i = 1:length(allfiles)
           if exist(allfiles{i}, 'file') ~= 2
               errordlg(['Unable to find or open file: ' allfiles{i}],
'File Error', 'modal');
                    continue; % Skip to the next file
                end

                %Update T/F statement for file-type tracking
                str = app.FileNAME(i);
                if contains(str, "hmax")
                    containsHmax = true;
                end
                if contains(str, "eta")
                    containsEta = true;
                end
                if contains(str, "time")
                    containsTime = true;
                end
            end

            %Enable or disable colormap options based on file type
            if containsHmax || containsTime
                app.CoastlinecolourDropDown_8.Enable = "on";
                app.FlipCheckBox_5.Enable = "on";
                app.toEditField.Enable = "on";
                app.MinEditField.Enable = "on";
                app.ColorbarTextSize.Enable = "on";
                app.DivisionEditFieldLabel.Enable = "on";
                app.DivisionEditField.Enable = "on";
            end
            if containsEta
                app.CoastlinecolourDropDown_7.Enable = "on";
                app.FlipCheckBox_2.Enable = "on";
                app.MaxEditField_2.Enable = "on";
                app.MinEditField_2.Enable = "on";
                app.ColorinterpolationEditField_2.Enable = "on";
                app.DivisionLabel.Enable = "on";
                app.ColorbarTextSize.Enable = "on";
            else
                app.CoastlinecolourDropDown_8.Enable = "on";
                app.FlipCheckBox_5.Enable = "on";
                app.toEditField.Enable = "on";
                app.MinEditField.Enable = "on";
                app.ColorbarTextSize.Enable = "on";
                app.DivisionEditFieldLabel.Enable = "on";
                app.DivisionEditField.Enable = "on";

            end


            %Read the first file to extract its matrix size, which will be used to set up the map boundary limit
            try
                f = readmatrix(app.FileNAME(1));
                [row,col] = size(f);
                app.col1 = col-1;
                app.row1 = row-1;
            end
        end

        % Button pushed function: GENERATEButton
        function GENERATEButtonPushed(app, event)
%Please cite this paper when you use the applications:
%Felix, R., Tan, E. H. Z., Watanabe, M., Verolino, A., Puah, J. Y., & Switzer, A. D. (2025). Funwave-based open-access mapping applications (FUNMAP) applied to Tsunami modelling from the Manila Trench to Manila Bay, Philippines. Geoscience Letters. https://doi.org/10.1186/s40562-025-00422-5




% NANYANG TECHNOLOGICAL UNIVERSITY - NTUITIVE PTE LTD (NTUITIVE) Dual License Agreement
% Non-Commercial Use Only
% This NTUITIVE License Agreement, including all exhibits ("NTUITIVE-LA") is a legal agreement between you and NTUITIVE (or “we”) located at 71 Nanyang Drive, NTU Innovation Centre, #01-109, Singapore 637722, a wholly owned subsidiary of Nanyang Technological University (“NTU”) for the software or data identified above, which may include source code, and any associated materials, text or speech files, associated media and "online" or electronic documentation and any updates we provide in our discretion (together, the "Software").
%
% By installing, copying, or otherwise using this Software, found at https://github.com/raquelpfelix, you agree to be bound by the terms of this NTUITIVE-LA. If you do not agree, do not install copy or use the Software. The Software is protected by copyright and other intellectual property laws and is licensed, not sold.   If you wish to obtain a commercial royalty bearing license to this software please contact us at rf761@bath.ac.uk.
%
% SCOPE OF RIGHTS:
% You may use, copy, reproduce, and distribute this Software for any non-commercial purpose, subject to the restrictions in this NTUITIVE-LA. Some purposes which can be non-commercial are teaching, academic research, public demonstrations and personal experimentation. You may also distribute this Software with books or other teaching materials, or publish the Software on websites, that are intended to teach the use of the Software for academic or other non-commercial purposes.
% You may not use or distribute this Software or any derivative works in any form for commercial purposes. Examples of commercial purposes would be running business operations, licensing, leasing, or selling the Software, distributing the Software for use with commercial products, using the Software in the creation or use of commercial products or any other activity which purpose is to procure a commercial gain to you or others.
% If the Software includes source code or data, you may create derivative works of such portions of the Software and distribute the modified Software for non-commercial purposes, as provided herein.
% If you distribute the Software or any derivative works of the Software, you will distribute them under the same terms and conditions as in this license, and you will not grant other rights to the Software or derivative works that are different from those provided by this NTUITIVE-LA.
% If you have created derivative works of the Software, and distribute such derivative works, you will cause the modified files to carry prominent notices so that recipients know that they are not receiving the original Software. Such notices must state: (i) that you have changed the Software; and (ii) the date of any changes.
%
% You may not distribute this Software or any derivative works.
% In return, we simply require that you agree:
% 1.    That you will not remove any copyright or other notices from the Software.
% 2.    That if any of the Software is in binary format, you will not attempt to modify such portions of the Software, or to reverse engineer or decompile them, except and only to the extent authorized by applicable law.
% 3.    That NTUITIVE is granted back, without any restrictions or limitations, a non-exclusive, perpetual, irrevocable, royalty-free, assignable and sub-licensable license, to reproduce, publicly perform or display, install, use, modify, post, distribute, make and have made, sell and transfer your modifications to and/or derivative works of the Software source code or data, for any purpose.
% 4.    That any feedback about the Software provided by you to us is voluntarily given, and NTUITIVE shall be free to use the feedback as it sees fit without obligation or restriction of any kind, even if the feedback is designated by you as confidential.
% 5.    THAT THE SOFTWARE COMES "AS IS", WITH NO WARRANTIES. THIS MEANS NO EXPRESS, IMPLIED OR STATUTORY WARRANTY, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ANY WARRANTY AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE SOFTWARE OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT. THERE IS NO WARRANTY THAT THIS SOFTWARE WILL FULFILL ANY OF YOUR PARTICULAR PURPOSES OR NEEDS. ALSO, YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE SOFTWARE OR DERIVATIVE WORKS.
% 6.    THAT NEITHER NTUITIVE NOR NTU NOR ANY CONTRIBUTOR TO THE SOFTWARE WILL BE LIABLE FOR ANY DAMAGES RELATED TO THE SOFTWARE OR THIS NTUITIVE-LA, INCLUDING DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES, TO THE MAXIMUM EXTENT THE LAW PERMITS, NO MATTER WHAT LEGAL THEORY IT IS BASED ON. ALSO, YOU MUST PASS THIS LIMITATION OF LIABILITY ON WHENEVER YOU DISTRIBUTE THE SOFTWARE OR DERIVATIVE WORKS.
% 7.    That we have no duty of reasonable care or lack of negligence, and we are not obligated to (and will not) provide technical support for the Software.
% 8.    That if you breach this NTUITIVE-LA or if you sue anyone over patents that you think may apply to or read on the Software or anyone's use of the Software, this NTUITIVE-LA (and your license and rights obtained herein) terminate automatically.  Upon any such termination, you shall destroy all of your copies of the Software immediately.  Sections 3, 4, 5, 6, 7, 8, 11 and 12 of this NTUITIVE-LA shall survive any termination of this NTUITIVE-LA.
% 9.    That the patent rights, if any, granted to you in this NTUITIVE-LA only apply to the Software, not to any derivative works you make.
% 10.   That the Software may be subject to U.S. export jurisdiction at the time it is licensed to you, and it may be subject to additional export or import laws in other places.  You agree to comply with all such laws and regulations that may apply to the Software after delivery of the software to you.
% 11.   That all rights not expressly granted to you in this NTUITIVE-LA are reserved.
% 12.   That this NTUITIVE-LA shall be construed and controlled by the laws of the Republic of Singapore without regard to conflicts of law.  If any provision of this NTUITIVE-LA shall be deemed unenforceable or contrary to law, the rest of this NTUITIVE-LA shall remain in full effect and interpreted in an enforceable manner that most nearly captures the intent of the original language.
%
%
% Do you accept all of the terms of the preceding NTUITIVE-LA license agreement? If you accept the terms, click “I Agree,” then “Next.”  Otherwise click “Cancel.”
%
% Copyright (c) NTUITIVE. All rights reserved.



%A copy of FUNWAVE's BDS license agreement:
% !    Copyright (c) 2016, FUNWAVE Development Team
% !
% !    (See http://www.udel.edu/kirby/programs/funwave/funwave.html
% !     for Development Team membership)
% !
% !    All rights reserved.
% !
% !    FUNWAVE_TVD is free software: you can redistribute it and/or modify
% !    it under the terms of the Simplified BSD License as released by
% !    the Berkeley Software Distribution (BSD).
% !
% !    Redistribution and use in source and binary forms, with or without
% !    modification, are permitted provided that the following conditions are met:
% !
% !    1. Redistributions of source code must retain the above copyright notice, this
% !       list of conditions and the following disclaimer.
% !    2. Redistributions in binary form must reproduce the above copyright notice,
% !    this list of conditions and the following disclaimer in the documentation
% !    and/or other materials provided with the distribution.
% !
% !    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
% !    ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
% !    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
% !    DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
% !    ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
% !    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
% !    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
% !    ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
% !    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% !    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
% !
% !    The views and conclusions contained in the software and documentation are those
% !    of the authors and should not be interpreted as representing official policies,
% !    either expressed or implied, of the FreeBSD Project.



%This is the start of the app's code:
close all

            %-------------------------
            %------BASEMAP INPUT------
            %-------------------------
            %List of the  colormap items in the dropdown list
            keys = {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag',
    'gray', 'hot', 'hsv', 'jet', ...
                'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', 'blue', 'blue - green', ...
                'blue - purple', 'green - blue', 'greens', 'grays', 'oranges', 'orange - red', 'purple - blue', ...
                'purple - blue -green', 'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', ...
                'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', 'brown - teal', ...
                'pink - light green', 'purple - green', 'purple - orange', 'red - blue', 'red - gray', ...
                'red - yellow - blue', 'red - yellow - green', 'spectral', 'accent', 'dark 2', 'paired', ...
                'pastel 1', 'pastel 2', 'set 1', 'set 2', 'set 3', '--- MATLAB default ----', '---  CBREWER 2 ---', ...
                '< sequential >', '< divergent >', '< qualitative >'};

            %The equivalent colormap code
            keysColor = {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', ...
                'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter','Blues','BuGn','BuPu','GnBu','Greens', ...
                'Greys','Oranges','OrRd','PuBu','PuBuGn', ...
                'PuRd','Purples','RdPu','Reds','YlGn','YlGnBu','YlOrBr','YlOrRd','BrBG','PiYG','PRGn','PuOr','RdBu','RdGy', ...
                'RdYlBu','RdYlGn','Spectral','Accent', ...
                'Dark2','Paired','Pastel1','Pastel2','Set1','Set2','Set3','parula','parula','parula','parula','parula'};

            %The colorbrewer
            keysBrewer = {'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', ...
                'none', 'none', 'none', 'none', 'none', 'none', 'none', 'seq', 'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', 'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', 'div', 'div', 'div', ...
                'div', 'div', 'div', 'div', 'div', ...
                'div', 'qual', 'qual', 'qual', ...
                'qual', 'qual', 'qual', 'qual', 'qual', ...
                'none', 'none', 'none','none', 'none'};

            % Match dropdown list items to their corresponding colors and ColorBrewer
            colorMap = containers.Map(keys, keysColor);
            colorBrewer = containers.Map(keys, keysBrewer);

            %For eta colormap
            datacolorETA = colorMap(app.CoastlinecolourDropDown_7.Value);
            datacolorbrewerETA = colorBrewer(app.CoastlinecolourDropDown_7.Value);

            %For hmax colormap
            datacolor = colorMap(app.CoastlinecolourDropDown_8.Value);
            datacolorbrewer  = colorBrewer(app.CoastlinecolourDropDown_8.Value);
```

```


            %----------------------------
            %------LAND AREA COLOR------
            %----------------------------

            %List of commonly used colors and styles
            colorMap = containers.Map({'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'}, {'k', '[0.4 0.4 0.4]', '[0.3 0.3 0.3]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'}); %[0.7 0.7 0.7]
            lineStyleMap = containers.Map({'Solid', 'Dashed', 'Dotted', 'Dash-dotted'}, {'-', '--', ':', '-.'});

            %Landmass color
            app.LandAreaColor = colorMap(app.CoastlinecolourDropDown_2.Value);

            %------------------
            %------GAUGES------
            %------------------
            %Marker Style
            GaugeMarkerOptions = containers.Map({'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'}, ...
                {'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'});
            GaugeMarker = GaugeMarkerOptions(app.CoastlinecolourDropDown_3.Value);

            %Marker Color
            GaugeColor =  colorMap(app.ColorDropDown_3.Value);

            %Label - horizontal alignment
            horzLabelMap = containers.Map({'Centre', 'Right', 'Left'}, {'center', 'left', 'right'});
            HorzLabel = horzLabelMap(app.CoastlinecolourDropDown_4.Value);

            %Label - vertical alignment
            vertLabelMap = containers.Map({'Centre', 'Top', 'Bottom'}, {'middle', 'bottom', 'top'});
            VertLabel = vertLabelMap(app.CoastlinecolourDropDown_5.Value);


            %--------------------------------
            %------BATHYMETRY CONTOURS------
            %--------------------------------
            %Line Colours
            ContourColor = colorMap(app.ColorDropDown.Value);
            %Line style
            ContourLineStyle = lineStyleMap(app.StyleDropDown.Value);


            %---------------------------------
            %------WAVE HEIGHT CONTOURS------
            %---------------------------------
            %Line Colours
            WaveContourColor = colorMap(app.ColorDropDown_2.Value);
            %Line style
            WaveContourLineStyle = lineStyleMap(app.StyleDropDown_2.Value);


            %---------------------------------------
            %------FIRST WAVE ARRIVAL CONTOURS------
            %---------------------------------------
            %Default threshold: 0.001 m
            %Line Colours
            ArrivalTimeContourColor = colorMap(app.ColorDropDown_6.Value);
            %Line Style
            ArrivalTimeLineStyle = lineStyleMap(app.StyleDropDown_4.Value);

            %--------------------------------------
            %------SAVE MAP: Output Directory------
            %--------------------------------------
            %Check if the output directory has been manually set by the user
            if app.OutputDirectoryEditField.Value == string(app.WORKFOLDER0)
                app.inputtxt_directory0  = fullfile(app.WORKFOLDER);
                app.inputtxt_directory1 = fullfile(app.WORKFOLDER0);

            else % Set the default directory to the Desktop
                if ismac  %macOS
                    defaultDir = fullfile(getenv('HOME'), 'Desktop');
                elseif ispc  %Windows
                    defaultDir = fullfile(getenv('USERPROFILE'), 'Desktop');
                else %Others
                    defaultDir = pwd;
                end

                %Create 'OUTPUT_FILES' folder and 'Figures' subfolder
                Dir0 = fullfile(defaultDir, 'OUTPUT_FILES');
                Dir1 = fullfile(defaultDir, 'OUTPUT_FILES/Figures');
                if ~exist(Dir0, 'dir')
                    mkdir(Dir0);
                end
                if ~exist(Dir1, 'dir')
                    mkdir(Dir1);
                end

                app.inputtxt_directory0 = Dir0;
                app.inputtxt_directory1 = Dir1;

                %Display the path directory of the 'Figures' subfolder in the textbox
                app.OutputDirectoryEditField.Value = deblank(string(Dir1));
                app.OutputDirectoryEditField.FontColor = 'k';
            end



            %Create animation/ creating and opening the file
            if app.mp4CheckBox.Value
                vidObj = VideoWriter(fullfile(char(app.inputtxt_directory1), 'animation_waveheight.mp4'), 'MPEG-4');
                vidObj.FrameRate = app.FramerateEditField_2.Value;
                open(vidObj);
            end

```

```matlab
%-----------------------
%------LOG REPORT------
%-----------------------
%Create 'Log_Files' subfolder in 'OUTPUT_FILES'
LogFolder = fullfile(app.inputtxt_directory0, 'Log_Files');
if ~exist(LogFolder, 'dir')
    mkdir(LogFolder);
end
%Create the text file
filePath = fullfile(LogFolder, 'Log_Report_WaveHeightMap.txt');
fileId = fopen(filePath ,'w');

% Template for the default section of the log report (includes information from the 'Input Data' section)
headerTitle = '******************** LOG REPORT ******************** ';
timestamp = datestr(now, 'yyyy-mm-dd HH:MM:SS');
endHeaderLine = '****************************************************';
footerSeparator = '____________________________________________________';
logContent = [
    sprintf('%s\n', headerTitle), ...
    sprintf('%-30s%-s\n', 'Map Type:', 'Wave Height Map'), ...
    sprintf('%-30s%-s\n', 'Timestamp:', timestamp), ...
    sprintf('%s\n\n', endHeaderLine), ...
    sprintf('%-30s%-s\n', 'Southwest Corner:', sprintf('Long: %f Lat: %f', app.LongitudeEditField.Value, app.LatitudeEditField.Value)), ...
    sprintf('%-30s%-s\n', 'Grid Size:', sprintf('x: %d y: %d', app.gridX.Value, app.gridY.Value)), ...
    sprintf('%-30s%-s\n', 'Simulation Time Start:', sprintf('%d sec', app.StartTime.Value)), ...
    sprintf('%-30s%-s\n', 'Simulation Time Interval:', sprintf('%d sec', app.TotalSimuilationTimesecEditField_3.Value)), ...
    sprintf('%s\n\n', footerSeparator)
    ];
fprintf(fileId, '%s', logContent);


%------LOG REPORT--------------------
%------Input Data: Import Files------
%------------------------------------
%Check the type of output file loaded in the app
patterns = {'hmax_', 'eta_','time_'};
matches1 = false(size(string(app.FileNAME)));
nonMatchingFiles = ~contains(string(app.FileNAME), patterns);
for i = 1:numel(patterns)
    pattern = patterns{i};
    matches1 = matches1 | contains(string(app.FileNAME), pattern);
end

if any(nonMatchingFiles)
    if contains(app.FileTextArea.Value, "eta_xxxx")
        fprintf(fileId, '%-30s%-s\n\n', 'IMPORT FILE/S:','Empty. No basemap is plotted'); %Add an error note in the log report
    else
        % if (app.FileTextArea.Value ~= "0") % If non- eta/hmax file is loaded
```

```matlab
                fprintf(fileId, '%-30s%-s\n', 'IMPORT FILE/S:', 'Use filenames with ''eta_'', ''hmax_'' initials');
                fprintf(fileId, '%-30s%-s\n', ' ' , 'Incorrect files are loaded:');
                fprintf(fileId, '%-30s%-s\n\n', ' ', join(app.FileNAME(nonMatchingFiles), ', '));% List all filenames with errors in the log report

            end
        end

        %------LOG REPORT------------------------
        %------Input Data: Southwest Corner------
        %----------------------------------------
        %Check if the values have not changed and add an error note to the log report
        if app.LongitudeEditField.Value == 0 && app.LatitudeEditField.Value == 0
            fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','Both latitude and longitude values are zeroes');
        elseif app.LongitudeEditField.Value == 0 && app.LatitudeEditField.Value ~= 0
            fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','A default zero value is used for longitude');
        elseif app.LongitudeEditField.Value ~= 0 && app.LatitudeEditField.Value == 0
            fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','A default zero value is used for latitude');
        end


        %------LOG REPORT-----------------
        %------Input Data: Grid Size------
        %--------------------------------
        %Check if the values have not changed and add an error note to the log report
        if app.gridX.Value == 0 && app.gridY.Value == 0
            fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','Both x and y values are zeroes');
        elseif app.gridX.Value == 0 && app.gridY.Value ~= 0
            fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','X value is zero');
        elseif app.gridX.Value ~= 0 && app.gridY.Value == 0
            fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','Y value is zero');
        end

        %------LOG REPORT-----------------------
        %------Input Data: Load Bathymetry------
        %---------------------------------------
        %Check if the matrix size of the loaded bathymetry file matches the imported eta/hmax file
        if matches(app.FileTextArea.Value,string(app.FileNAME))  % Check if the imported eta/hmax files match the names in the textbox
            if matches(app.DepthFileEditField.Value,string(app.bathymetryname)) %Check if the imported bathymetry matches the name in the textbox
                if app.rowFirstFile ~= app.bathyRow
                    fprintf(fileId, '%-30s%-s\n', 'MATRIX ROWS:','Rows in eta/hmax/time/ file and bathymetry do not match');
                end
                if app.colFirstFile ~= app.bathyCol
                    fprintf(fileId, '%-30s%-s\n', 'MATRIX COLUMNS:','Columns in eta/hmax/time file and bathymetry do not match');
                end
            end
        end
        if matches(app.DepthFileEditField.Value,"0") || ~matches(app.DepthFileEditField.Value,string(app.bathymetryname)) % Add a log note that no bathymetry is loaded, so the features to edit land area and bathymetry contours are unavailable.
            fprintf(fileId, '%-30s%-s\n\n', 'BATHYMETRY:', 'Empty. Land color and bathymetry contours are not applicable.');
            app.DepthFileEditField.FontColor = 'w';
            app.DepthFileEditField.BackgroundColor = 'r';
            app.DepthFileEditField.Value = 'NO FILE';
        end


        %------LOG REPORT-----------------
        %------Basemap: Wave heights------
        %--------------------------------
        if all(contains(string(app.FileNAME),  patterns)) %If all files are in eta_ /hmax_ format
            %Verify that the minimum, interval, and maximum input values are correct; update values if conditions are not met
            %hmax option
            if app.MinEditField.Value > app.toEditField.Value
                    temp = app.MinEditField.Value;
                    app.MinEditField.Value = app.toEditField.Value;
                    app.toEditField.Value = temp;
                    drawnow
                fprintf(fileId, '%-30s%-s\n', 'BASEMAP HMAX COLORBAR:','Max value must exceed min value');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Min value adjusted to load the map');
            end
            %eta option
            if app.MinEditField_2.Value > app.MaxEditField_2.Value
                temp = app.MinEditField_2.Value;
                app.MinEditField_2.Value = app.MaxEditField_2.Value;
                app.MaxEditField_2.Value = temp;
                drawnow
                fprintf(fileId, '%-30s%-s\n', 'BASEMAP ETA COLORBAR:','Max value must exceed min value');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Min value adjusted to load the map');
            end
        end

        %------LOG REPORT-----------------------------------
        %------Overlay Features: Bathymetry contours------
        %-------------------------------------------------
        if matches(app.DepthFileEditField.Value,string(app.bathymetryname))
            if app.MaximumEditField.Value < app.MinimumEditField.Value % Update the max value if it is less than the user-defined minimum value.
                app.MaximumEditField.Value = 10000;
```

```
                    fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOURS:','Max value must exceed min value');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Max value set to 10,000');
                end
                if app.IntervalEditField_5.Value > (app.MaximumEditField.Value -  app.MinimumEditField.Value) %Update the contour interval value when the condition is not met
                    app.IntervalEditField_5.Value = (app.MaximumEditField.Value -  app.MinimumEditField.Value)/2;
                    fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOUR INTERVAL:','Value must be less than the max-min difference');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
                end

                if app.AddLabelsCheckBox_2.Value %Add labels
                    %Check if the label interval is divisible by the value provided in the 'Range' section
                    if mod(app.IntervalEditField_3.Value, app.IntervalEditField_5.Value) ~= 0
                        app.IntervalEditField_3.Value = app.IntervalEditField_5.Value;
                        fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOUR LABEL:','Value must be divisible by the Interval value in the Range section');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                    end

                    %Check if the label's interval value is less than the maximum value in 'Contour Range'
                    if app.IntervalEditField_3.Value > app.MaximumEditField.Value
                        app.IntervalEditField_3.Value = app.IntervalEditField_5.Value;
                        fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOUR LABEL:','Value must be less than the maximum value of the contour line assigned in the Range section');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                    end
                end
            end

            if app.BathymetryButton_2.Value && (strcmp(app.DepthFileEditField.Value, "0") || strcmp(app.DepthFileEditField.Value, "00"))
                %No bathymetry file found, show "NO FILE" warning in the bathymetry input window
                app.DepthFileEditField.Value = "NO FILE";
                app.DepthFileEditField.FontColor = 'w';
                app.DepthFileEditField.BackgroundColor = 'r';
            end

            %------LOG REPORT---------------------------------
            %------Overlay Features: Wave height contours------
            %------------------------------------------------
            if app.MaximumdepthEditField_2.Value < app.MinimumdepthEditField_2.Value
                app.MaximumdepthEditField_2.Value = 1;
                app.MinimumdepthEditField_2.Value = 0;
                fprintf(fileId, '%-30s%-s\n', 'WAVE HEIGHT CONTOURS:','Max value must exceed min value');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Max and min values reset to default [0,1]');
            end

            if app.LineintervalEditField_2.Value  >  app.MaximumdepthEditField_2.Value - app.MinimumdepthEditField_2.Value
                app.LineintervalEditField_2.Value = (app.MaximumdepthEditField_2.Value - app.MinimumdepthEditField_2.Value)/2;
                fprintf(fileId, '%-30s%-s\n', 'WAVE HEIGHT CONTOUR INTERVAL:','Value must be less than the max-min difference');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
            end

            if app.AddLabelsCheckBox_3.Value %Add labels
                if mod(app.wh_interval.Value, app.LineintervalEditField_2.Value) ~= 0
                    app.wh_interval.Value = app.LineintervalEditField_2.Value;
                    fprintf(fileId, '%-30s%-s\n', 'WAVE HEIGHT CONTOUR LABEL:','Value must be divisible by the Interval value in the Range section');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                end

                %Check if the label's interval value is less than the maximum value in 'Contour Range'
                if app.wh_interval.Value > app.MaximumdepthEditField_2.Value
                    app.wh_interval.Value = app.LineintervalEditField_2.Value;
                    fprintf(fileId, '%-30s%-s\n', 'WAVE HEIGHT CONTOUR LABEL:','Value must be less than the maximum value of the contour line assigned in the Range section');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                end
            end

            %------LOG REPORT--------------------------
            %------Overlay Features: Arrival Time------
            %-----------------------------------------
            if app.ArrivalTimeButton.Value
                if ~matches(app.FileEditField_3.Value, string(app.ArrivalTimename)) ||  matches(app.FileEditField_3.Value,"0")%Check if there is an uploaded file
                    fprintf(fileId, '%-30s%-s\n\n', 'ARRIVAL TIME:', 'Empty. No Arrival Time Contour plotted');
                    app.FileEditField_3.FontColor = 'w';
                    app.FileEditField_3.BackgroundColor = 'r';
                    app.FileEditField_3.Value = 'NO FILE';
                end
                if app.MinimumdepthEditField_3.Value > app.MaximumdepthEditField_3.Value
                    % Swap the values
                    tempArrivalTime = app.MinimumdepthEditField_3.Value;
                    app.MinimumdepthEditField_3.Value = app.
```

```
MaximumdepthEditField_3.Value;
                    app.MaximumdepthEditField_3.Value = tempArrivalTime;
                    fprintf(fileId, '%-30s%-s\n', 'ARRIVAL TIME:', 'Max value must exceed min value');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
                end
                if app.LineintervalEditField_3.Value > app.MaximumdepthEditField_3.Value - app.MinimumdepthEditField_3.Value
                    app.LineintervalEditField_3.Value  = (app.MaximumdepthEditField_3.Value - app.MinimumdepthEditField_3.Value)/2;
                    fprintf(fileId, '%-30s%-s\n', 'ARRIVAL TIME INTERVAL:','Value must be less than the max-min difference');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
                end

                if app.AddLabelsCheckBox_5.Value %Add Labels
                    if mod(app.IntervalEditField_7.Value, app.LineintervalEditField_3.Value) ~= 0
                        app.IntervalEditField_7.Value = app.LineintervalEditField_3.Value;
                        fprintf(fileId, '%-30s%-s\n', 'ARRIVAL TIME LABEL:','Value must be divisible by the Interval value in the Range section');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                    end

                    %Check if the label's interval value is less than the maximum value in 'Contour Range'
                    if app.IntervalEditField_7.Value > app.MaximumdepthEditField_3.Value
                        app.IntervalEditField_7.Value = app.LineintervalEditField_3.Value;
                        fprintf(fileId, '%-30s%-s\n', 'ARRIVAL TIME LABEL:','Value must be less than the maximum value of the contour line assigned in the Range section');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                    end
                end
            end


            %------LOG REPORT--------------------
            %------Overlay Features: Gauges------
            %------------------------------------
            if app.GaugesCheckBox.Value && ~matches(app.FileEditField.Value , string(app.GAUGEFILE))
                fprintf(fileId, '%-30s%-s\n\n', 'GAUGES:', 'Empty. No virtual gauges plotted');
                app.FileEditField.Value = "NO FILE";
                app.FileEditField.FontColor = 'w';
                app.FileEditField.BackgroundColor = 'r';
            end

            %------LOG REPORT---------------------------
            %------General Layout: Boundary Limits------
            %--------------------------------------------
            if app.WestEditField.Value > app.EastEditField.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:','East boundary must be greater than West boundary');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
            elseif app.EastEditField.Value == app.WestEditField.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:', 'East and West values are the same');
                fprintf(fileId, '%-30s%-s\n\n', 'Resolved:', 'The default values are used');
            end

            if app.SouthEditField.Value > app.NorthEditField.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:','North boundary must be greater than South boundary');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
            elseif  app.NorthEditField.Value == app.SouthEditField.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:', 'North and South values are the same');
                fprintf(fileId, '%-30s%-s\n\n', 'Resolved:', 'The default values are used');
            end

            %--------------------------
            %------SAVE MAP: .tif-------
            %--------------------------
            if app.tifCheckBox_2.Value && (app.LatitudeEditField.Value > 90 || app.LatitudeEditField.Value < -90)
                fprintf(fileId, '%-30s%-s\n\n', 'BOUNDARY LIMIT:','The Latitude values should be within -90 to 90.');
            end

            fclose(fileId); %Close the log report file

            %---------------------------------
            %------PRE-SET MAP OPTIONS-------
            %---------------------------------
            clear depth f1
            FileLength = length(app.FileINPUT); %Read all the imported eta/hmax files

            %------Plot all data in one figure------
            if app.PlotalldatainonefigureButton.Value
                %Create the figure
                figure(1);

                %Formatting the subplots
                if FileLength == 1
                    plotCol = 1;
                    plotRow = 1;
                elseif FileLength > 1 && FileLength <= 10
                    plotCol = 2;
                    plotRow = ceil(FileLength/2);
                elseif FileLength > 10
                    plotCol = 3;
                    plotRow = ceil(FileLength/3);
                end

            end

            %-----Bathymetry data------
            % bathymetryFileExists = exist(app.bathymetryinputdata, 'file') == 2;
            % if bathymetryFileExists
                depth = app.bathymetrydata;

                % Flip the data horizontally
                if app.HorizontalCheckBox.Value
                    depth = fliplr(depth);
                end

                % Flip the data vertically
                if app.VerticalCheckBox.Value
                    depth = flipud(depth);
                end
            % end


            %------Basemap Colormap-------
            colorbarHandles = [];
            if strcmp(datacolorbrewerETA,'none') %Using default matlab colormap options
                datacolor2ETA = evalin('base',datacolorETA);
            else %Using cbrewer2
                datacolor2ETA = cbrewer2(datacolorbrewerETA,datacolorETA,app.ColorinterpolationEditField_2.Value,'pchip');
            end

            if strcmp(datacolorbrewer,'none')  %Using default matlab colormap options
                datacolor2 =  evalin('base',datacolor);
            else %Using cbrewer2
                datacolor2 = cbrewer2(datacolorbrewer,datacolor,app.DivisionEditField.Value,'pchip');
            end

            colormapMinHMAX = app.MinEditField.Value;
            colormapMaxHMAX = app.toEditField.Value;
            colorbartitleHMAX = 'Maximum \eta (m)';

            colormapMinETA = app.MinEditField_2.Value;
            colormapMaxETA = app.MaxEditField_2.Value;
            colorbartitleETA = '\eta (m)';

            colorbarFontSize = app.ColorbarTextSize.Value;
            flipCheckBoxETA = app.FlipCheckBox_2.Value;
            flipCheckBoxHMAX = app.FlipCheckBox_5.Value;

            %------Bathymetry Contours------
            if app.BathymetryButton_2.Value && matches(app.DepthFileEditField.Value,string(app.bathymetryname))
                kernel = ones(6) / 36;  %Make the contour smoother
                wavebathymetry=  filter2(kernel, depth);
                contourLevels = app.MinimumEditField.Value:app.
```

```
IntervalEditField_5.Value:app.MaximumEditField.Value;
                manualLabels = app.MinimumEditField.Value:app. ↙
IntervalEditField_3.Value:app.MaximumEditField.Value;
            end


            isBathymetryChecked = app.BathymetryButton_2.Value;
            isMatchingBathymetry = matches(app.DepthFileEditField.Value, ↙
string(app.bathymetryname));
            isAddingLabels = app.AddLabelsCheckBox_2.Value;
            fontSize = app.TextLabelSize.Value;
            labelSpacing = app.LabelSpacingEditField.Value;
            lineWidth = app.WidthEditField.Value;
            contourCol = ContourColor; % Capture contour color if it's same↙
for all
            contourStyle = ContourLineStyle; % Capture contour style if it's↙
same for all


            %------Wave Height Contours------
            if app.WaveHeightButton.Value
                wavecontour = app.fin;
                if exist('app.bathymetrydata', 'var')
                    wavecontour(depth<=0) = NaN; %Remove values on land
                end
                contourLevelsWH = app.MinimumdepthEditField_2.Value:app. ↙
LineintervalEditField_2.Value:app.MaximumdepthEditField_2.Value;
                manualLabels2 = app.MinimumdepthEditField_2.Value:app. ↙
wh_interval.Value:app.MaximumdepthEditField_2.Value;
            end

            waveContourColor = WaveContourColor;
            waveContourStyle = WaveContourLineStyle;
            waveLineWidth = app.ThicknessEditField_2.Value;
            fontSize2 = app.TextLabelSize_2.Value;
            labelSpacing2 = app.LabelSpacingEditField_2.Value;

            %------Arrival Time Contours------
            if app.ArrivalTimeButton.Value && matches(app.FileEditField_3. ↙
Value, string(app.ArrivalTimename))
                atc = app.ArrivalTimeConverted;

                %Flip the data vertically
                if app.VerticalCheckBox.Value
                    atc = flipud(app.ArrivalTimeConverted);
                end

                %Flip the data horizontally
                if app.HorizontalCheckBox.Value
                    atc = fliplr(app.ArrivalTimeConverted);
                end

                contourLevelsATC = app.MinimumdepthEditField_3.Value:app. ↙
LineintervalEditField_3.Value:app.MaximumdepthEditField_3.Value;
                manualLabels3 = app.MinimumdepthEditField_3.Value:app. ↙
IntervalEditField_7.Value:app.MaximumdepthEditField_3.Value;
            end

            atcContourColor = ArrivalTimeContourColor;
            atcContourStyle = ArrivalTimeLineStyle;
            atcLineWidth = app.ThicknessEditField_3.Value;
            fontSize3 = app.TextLabelSize_3.Value;
            labelSpacing3 = app.LabelSpacingEditField_4.Value;

            %------Gauges------
            markerSize = app.SizeEditField.Value;
            labelFontSize = app.SizeEditField_3.Value;

            %------Boundary limit-------
            eastValue = app.EastEditField.Value;
            westValue = app.WestEditField.Value;

            if eastValue < westValue
                % Swap West and East input values if the West value is greater than the East
                xMin_WH = eastValue;
                xMax_WH = westValue;
                app.EastEditField.Value = xMax_WH;
                app.WestEditField.Value = xMin_WH;
            elseif eastValue == westValue
                % Set the limits to the default values
                app.EastEditField.Value = max(app.x);
                app.WestEditField.Value = min(app.x);
                xMin_WH = app.WestEditField.Value;
                xMax_WH = app.EastEditField.Value;
            else
                xMin_WH = westValue;
                xMax_WH = eastValue;
            end

            northValue = app.NorthEditField.Value;
            southValue = app.SouthEditField.Value;

            if northValue < southValue
                % Swap North and South input values if the South value is greater than the North
                yMin_WH = northValue;
                yMax_WH = southValue;
                app.NorthEditField.Value = yMax_WH;
                app.SouthEditField.Value = yMin_WH;
            elseif northValue == southValue
                % Set the limits to the default values
                app.NorthEditField.Value = max(app.y);
                app.SouthEditField.Value = min(app.y);
            else
                yMin_WH = southValue;
                yMax_WH = northValue;
            end

            %------Map Title-------
            totalSimulationTime = app.TotalSimuilationTimesecEditField_3.Value;
            startTime = app.StartTime.Value;
            colorbarTextSize = app.ColorbarTextSize.Value + 1;
            pats = digitsPattern;

```

```


%-------------------------------------------
%------Start of the loop for plotting-------
%-------------------------------------------
for i = 1:FileLength
    if all(contains(string(app.FileNAME), patterns)) % Check if each file is either an eta or hmax file
        f1 = readmatrix(app.FileINPUT(i));

        % Flip the data horizontally
        if app.HorizontalCheckBox.Value
            f1 = fliplr(f1);
        end

        % Flip the data vertically
        if app.VerticalCheckBox.Value
            f1 = flipud(f1);
        end

        if matches(app.DepthFileEditField.Value,string(app.bathymetryname))
            %If the bathymetry file is loaded, delete negative land values
            f1(depth <= 0) = NaN;
        end

        % if matches(app.DepthFileEditField.Value, string(app.bathymetryname))
        %     % Ensure depth is defined and non-empty
        %     if ~isempty(depth)
        %         % Preallocate f1 to match the size of depth
        %         f1 = NaN(size(depth));  % Initialize with NaNs
        %         % Optionally assign valid values where depth > 0
        %         % f1(depth > 0) = someComputation(depth(depth > 0));
        %     % else
        %     %     warning('Depth data is empty. Skipping bathymetry filtering.');
        %     end
        % end
        app.fin = f1;

        %----------------------------------------
        %------PLOT ALL DATA IN ONE FIGURE------
        %----------------------------------------
        if app.PlotalldatainonefigureButton.Value
            gca(i)= subplot(plotRow,plotCol,i);
            %---------------------------
            %------PLOT SEPARATELY------
            %---------------------------
        elseif app.PlotseparatelyButton_2.Value
            figure();
            gca(i) = subplot(1,1,1);
        end

        %----------------------
        %------MAIN PLOT------
```

```
                    %---------------------

                    f = pcolor(app.x,  app.y, (app.fin)); %(X coordinates, X coordinates, eta/hmax values)
                    shading interp;
                    set(f, 'EdgeColor', 'none', 'FaceColor', 'interp');
                    set(gca,'Color', app.LandAreaColor) %Set the background color for the land
                    hold on


                    %------------------------------
                    %------BASEMAP: Colormap-------
                    %------------------------------
                    cb = colorbar;
                    colorbarHandles = [colorbarHandles, cb];

                    % Set the properties once
                    cb.Label.FontSize = colorbarFontSize; % Colorbar label
                    cb.FontSize = colorbarFontSize;       % Size of the tick number% Size of the tick number

                    if contains(string(app.FileNAME(i)), "eta") % For ETA/sea surface displacement at time t
                        % Flip the colorbar if needed
                        if flipCheckBoxETA
                            colormap(gca(i), flipud(datacolor2ETA));
                        else
                            colormap(gca(i), datacolor2ETA);
                        end
                        caxis(gca(i), [colormapMinETA, colormapMaxETA]); % Colorbar min and max values
                        cb.Label.String = colortitleETA;
                    else % For hmax/maximum wave height
                        % Flip the colorbar if needed
                        if flipCheckBoxHMAX
                            colormap(gca(i), flipud(datacolor2));
                        else
                            colormap(gca(i), datacolor2);
                        end
                        caxis(gca(i), [colormapMinHMAX, colormapMaxHMAX]); % Colorbar min and max values
                        cb.Label.String = colorbartitleHMAX;
                    end

                    %--------------------------------------------------
                    %------OVERLAY FEATURES: Bathymetry Contours-------
                    %--------------------------------------------------
                    if isBathymetryChecked && isMatchingBathymetry
                        %                          hold on
                        % Plot the contours
                        [C, h] = contour(app.x, app.y, wavebathymetry, contourLevels, 'EdgeColor', contourCol, 'LineStyle', contourStyle, 'LineWidth', lineWidth);
                        % Add contour labels if needed
                        if isAddingLabels
                            clabel(C, h, manualLabels, 'FontSize', fontSize, 'Color', 'k');
```

```matlab
                    set(findobj(h, '-property', 'LabelSpacing'), 'LabelSpacing', labelSpacing); % Adjust the label spacing
                end
            end

            %----------------------------------------------------
            %------OVERLAY FEATURES: Wave Height Contours-------
            %----------------------------------------------------
            if app.WaveHeightButton.Value
                % Plot the contours
                [D, g] = contour(app.x, app.y, wavecontour, contourLevelsWH, 'EdgeColor', waveContourColor, 'LineStyle', waveContourStyle, 'LineWidth', waveLineWidth);

                % Add contour labels
                if app.AddLabelsCheckBox_3.Value
                    clabel(D, g, manualLabels2, 'FontSize', fontSize2, 'Color', 'k');
                    set(findobj(g, '-property', 'LabelSpacing'), 'LabelSpacing', labelSpacing2); % Adjust the label spacing
                end
            end

            %-----------------------------------------------------
            %------OVERLAY FEATURES: Arrival Time Contours-------
            %-----------------------------------------------------
            if app.ArrivalTimeButton.Value && matches(app.FileEditField_3.Value, string(app.ArrivalTimename))
                % Plot the contours
                [E, ggg] = contour(app.x, app.y, atc, contourLevelsATC, 'EdgeColor', atcContourColor, 'LineStyle', atcContourStyle, 'LineWidth', atcLineWidth);

                if app.AddLabelsCheckBox_5.Value
                    clabel(E, ggg, manualLabels3, 'FontSize', fontSize3, 'Color', 'k');
                    set(findobj(ggg, '-property', 'LabelSpacing'), 'LabelSpacing', labelSpacing3); % Adjust the label spacing
                end
            end

            %-------------------------------------
            %------OVERLAY FEATURES: Gauges-------
            %-------------------------------------
            if app.GaugesCheckBox.Value && matches(app.FileEditField.Value, string(app.GAUGEFILE))
                % Plot the points
                gaugePoint = plot(app.longGauge, app.latGauge, GaugeMarker);
                set(gaugePoint, 'MarkerSize', markerSize, 'MarkerFaceColor', GaugeColor, 'MarkerEdgeColor', GaugeColor); % Size and color of the points

                % Add labels
                if app.AddLabelsCheckBox_4.Value
                    horzLabel = HorzLabel; % Ensure HorzLabel is precomputed
                    vertLabel = VertLabel; % Ensure VertLabel is
```

```
precomputed
                            gSpacing = app.GLspacing.Value;

                            for k = 1:length(app.latGauge)
                                gtxt = text(app.longGauge(k) + gSpacing, app.latGauge(k), num2str(k), 'FontSize', labelFontSize, 'HorizontalAlignment', horzLabel, 'VerticalAlignment', vertLabel);
                                set(gtxt, 'Clipping', 'on');
                            end
                        end
                    end
                    hold on

                    %--------------------------------------------
                    %------GENERAL LAYOUT: Boundary limit-------
                    %--------------------------------------------
                    xlim([xMin_WH,xMax_WH])
                    ylim([yMin_WH,yMax_WH])
                    hold on

                        %----------------------------------
                        %------GENERAL LAYOUT: Ticks-------
                        %----------------------------------
                        %Reduce the number of ticks to 2 on the X axis and 2 on the Y axis for simplicity
                        % Get the current ticks from the first plot
                        if i == 1
                            xAllTicks = xticks(gca);
                            yAllTicks = yticks(gca);

                            % Ensure the ticks are numeric arrays
                            if iscell(xAllTicks)
                                xAllTicks = cellfun(@str2double, xAllTicks, 'UniformOutput', true);
                            end
                            if iscell(yAllTicks)
                                yAllTicks = cellfun(@str2double, yAllTicks, 'UniformOutput', true);
                            end

                            % Remove any NaN values that might result from str2double conversion
                            xAllTicks = xAllTicks(~isnan(xAllTicks));
                            yAllTicks = yAllTicks(~isnan(yAllTicks));

                            % Select only the 2nd and second-to-last ticks
                            % For X axis
                            if length(xAllTicks) >= 4  % Check first if there are enough ticks to select from
                                xticksSelected = sort([xAllTicks(2), xAllTicks(end-1)]);
                            elseif length(xAllTicks) >= 2
                                xticksSelected = sort([xAllTicks(1), xAllTicks(end)]); % Use the first and last if less than 4
                            else
                                xticksSelected = xAllTicks; % Use whatever ticks are available
                            end

                            % For Y axis
                            if length(yAllTicks) >= 4
                                yticksSelected = sort([yAllTicks(2), yAllTicks(end-1)]);
                            elseif length(yAllTicks) >= 2
                                yticksSelected = sort([yAllTicks(1), yAllTicks(end)]); % Use the first and last if less than 4
                            else
                                yticksSelected = yAllTicks; % Use whatever ticks are available
                            end

                        end

                        % Update the number of ticks on the axes
                        xticks(gca, xticksSelected);
                        yticks(gca, yticksSelected);

                        % Determine the maximum decimal places for x and y ticks
                        xmaxDecimals = max(cellfun(@(x) length(regexp(x, '(?<=\.)\d+', 'match', 'once')), ...
                            cellstr(num2str(xticksSelected'))));
                        ymaxDecimals = max(cellfun(@(y) length(regexp(y, '(?<=\.)\d+', 'match', 'once')), ...
                            cellstr(num2str(yticksSelected'))));

                        % Convert tick labels to their equivalent in degrees
                        if app.PlotindegreesCheckBox_2.Value
                            % Convert selected ticks to formatted strings and set them for degrees
                            xticklabels(gca, arrayfun(@(x) sprintf('%.1f°', mod(x + 360, 360)), xticksSelected, 'UniformOutput', false));
                            yticklabels(gca, arrayfun(@(y) sprintf('%.1f°', y), yticksSelected, 'UniformOutput', false));
                        else
                            % Correct the format string and apply calculated maximum decimals
                            xticklabels(gca, arrayfun(@(x) sprintf(['%.' num2str(xmaxDecimals) 'f'], x), xticksSelected, 'UniformOutput', false));
                            yticklabels(gca, arrayfun(@(y) sprintf(['%.' num2str(ymaxDecimals) 'f'], y), yticksSelected, 'UniformOutput', false));
                        end

                        % Set the y-axis tick angle
                        ytickangle(gca, -270);

                        %Fontsize of the tick labels
                        set(gca,'Layer', 'top','LineWidth',1,'FontSize',app.ColorbarTextSize.Value-1)

                        %Adjust tick lengths based on the number of subplots if 'Plot all data in one figure' is selected
                        if FileLength <= 2 && app.PlotalldatainonefigureButton.Value
                            set(gca,'TickLength',[0.015 0.015])
                        else
```

```
    set(gca,'TickLength',[0.01 0.01])
end

grid off
box on
daspect([1 1 1]) %check if it distorts the countries
hold off

%-------------------------------------
%------GENERAL LAYOUT: Map Title------
%-------------------------------------
str = string(app.FileNAME(i));
dbl1 = extract(str, pats); % Extract numbers from the eta/hmax filename to calculate the time frame

if startTime == 0 && str2double(dbl1) == 0 %Set the time to zero
    timeT = startTime + (str2double(dbl1)-1);
else
    originalTimeT = totalSimulationTime * (str2double(dbl1)-1); % Convert the filenumber to time
    timeT = startTime + originalTimeT; % Adjust the start of the time count
end

% Time conversion
if timeT < 60
    timeStr = sprintf("%d sec", timeT);
elseif timeT < 3600
    minutes = round(timeT / 60);
    if minutes == 1
        timeStr = "1 min";
    else
        timeStr = sprintf("%d mins", minutes);
    end
else
    raw_hours = timeT/3600;
    hours = fix(timeT/3600); %Extract the whole number%Extract the whole number
    hours_decimal = raw_hours - floor(raw_hours); % Extract the decimal places
    minutes = round(hours_decimal *  60);

    if hours > 0
        if minutes > 0
            timeStr = sprintf("%d hr, %d mins", hours, minutes);
        else
            timeStr = sprintf("%d hr", hours);
        end
    end
end

title(timeStr, 'FontSize', colorbarTextSize); % Add the title to the top of the map; Adjust the font size of the title

%----------------------------------------
%------GENERAL LAYOUT: Figure Size-------
```

```
                        %----------------------------------------
                        if ~app.AutoSetCheckBox_2.Value
                            %Manually set the figure size
                            figureHandle = ancestor(f, 'figure'); %
                            set(figureHandle, 'Units', 'Inches', 'Position',↙
[0, 0, app.Width_2.Value, app.Height_2.Value], 'PaperUnits', 'Inches', 'PaperSize',↙
[app.Width_2.Value, app.Height_2.Value]);
                            set(gcf, 'Units', 'Inches', 'Position', [0, 0,↙
app.Width_2.Value, app.Height_2.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.↙
Width_2.Value, app.Height_2.Value]);
                        end


                        %--------------------------------------
                        %------SAVE MAP: Plot Separately-------
                        %--------------------------------------
                        if app.PlotseparatelyButton_2.Value
                            %Save the maps with filenames matching the loaded↙
file
                            OF  = fullfile(app.inputtxt_directory1,app.↙
FileNAME(i));

                            %Options for saving format
                            %PNG
                            if app.pngCheckBox.Value
                                outputfile = OF + ".png";
                                exportgraphics(gcf,outputfile,'Resolution',↙
300)

                            end

                            %JPG
                            if app.jpgCheckBox.Value
                                outputfile = OF + ".jpg";
                                exportgraphics(gcf,outputfile,'Resolution',↙
300)

                            end

                            %TIF
                            if app.tifCheckBox_2.Value
                                XXX = app.x;
                                YYY = app.y;

                                west_limit = app.WestEditField.Value;
                                east_limit = app.EastEditField.Value;
                                north_limit = app.NorthEditField.Value;
                                south_limit = app.SouthEditField.Value;

                                %Make the longitude values within -180 to 180↙
range
                                if west_limit ~= 180
                                    west_limit = mod(west_limit + 180, 360) -↙
180;
                                end

                                if east_limit ~= 180
                                    east_limit = mod(east_limit + 180, 360) -↙
180;
                                end
```

```

% Convert x coordinates to correct degree range
XXX(XXX > 180) = XXX(XXX > 180) - 360;

% Create meshgrid
[X, Y] = meshgrid(XXX, YYY);
Z = f1;
%Check if the latitude is within the Geographic limits
if app.LatitudeEditField.Value > 90 || app.LatitudeEditField.Value < -90
    app.LatitudeEditField.Value = 0;
else
    %Clip the matrix based on boundary limits
    if west_limit >= 0 && east_limit >=0  || west_limit < 0 && east_limit < 0 % if east and west values are either both (+) or both (-) values
        x_indices_positive =  XXX >= west_limit & XXX <= east_limit;
        y_indices = YYY >= south_limit & YYY <= north_limit;
        Z_positive = Z(y_indices, x_indices_positive);
        X_positive = X(y_indices, x_indices_positive);
        Y = Y(y_indices, x_indices_positive); % Keep the same for both file
        if ~isempty(Z_positive)
            % Define spatial referencing information
            R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
            R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
            R_positive.RowsStartFrom = 'west';

            % Save as geotiff
            filename_positive = OF + ".tif";
            geotiffwrite(filename_positive, Z_positive, R_positive);
        end

    elseif west_limit >= 0 && east_limit < 0 %if longitude values are a combination of (+) west limit and (-) east limit
        x_indices_positive = XXX >= 0 & XXX >= west_limit;
        x_indices_negative = XXX < 0 & XXX <= east_limit;
        y_indices = YYY >= south_limit & YYY <= north_limit;
        Z_positive = Z(y_indices, x_indices_positive);
        Z_negative = Z(y_indices, x_indices_negative);
        X_positive = X(y_indices, x_indices_positive);
```

```
            X_negative = X(y_indices, x_indices_negative);
            Y = Y(y_indices, x_indices_positive); % Keep the same for both file

            if ~isempty(Z_positive)
                % Define spatial referencing information
                R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
                R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                R_positive.RowsStartFrom = 'west';

                % Save as geotiff
                filename_positive = OF + "_1.tif";
                geotiffwrite(filename_positive, Z_positive, R_positive);
            end

            if ~isempty(Z_negative)
                % Define spatial referencing information
                R_negative = georasterref('RasterSize', size(Z_negative), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_negative(:)), max(X_negative(:))]);
                R_negative.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                R_negative.RowsStartFrom = 'west';

                % Save as geotiff
                filename_negative = OF + "_2.tif";
                geotiffwrite(filename_negative, Z_negative, R_negative)
            end

        elseif west_limit < 0 && east_limit >= 0 %if longitude values are a combination of (-) west limit and (+) east limit
            x_indices_positive = XXX >= 0 & XXX <= east_limit;
            x_indices_negative = XXX < 0 & XXX >= west_limit;
            y_indices = YYY >= south_limit & YYY <= north_limit;
            Z_positive = Z(y_indices, x_indices_positive);
            Z_negative = Z(y_indices, x_indices_negative);
            X_positive = X(y_indices, x_indices_positive);
            X_negative = X(y_indices, x_indices_negative);
            Y = Y(y_indices, x_indices_positive); % Keep the same for both file

            if ~isempty(Z_positive)
                % Define spatial referencing
```

```
information
                                        R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
                                        R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                                        R_positive.RowsStartFrom = 'west';

                                        % Save as geotiff
                                        filename_positive = OF + "_1.tif";
                                        geotiffwrite(filename_positive, Z_positive, R_positive);
                                    end
                                    if ~isempty(Z_negative)
                                        % Define spatial referencing information
                                        R_negative = georasterref('RasterSize', size(Z_negative), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_negative(:)), max(X_negative(:))]);
                                        R_negative.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                                        R_negative.RowsStartFrom = 'west';

                                        % Save as geotiff
                                        filename_negative = OF + "_2.tif";
                                        geotiffwrite(filename_negative, Z_negative, R_negative);
                                    end
                                end
                            end
                        end

                        %Create animation / closing the file
                        if app.mp4CheckBox.Value
                            set(gcf,'Renderer','zbuffer');
                            f = getframe(gcf);
                            writeVideo(vidObj,f);
                        end
                    end

            end

        end

        %Close the animation
        if app.mp4CheckBox.Value
            close(vidObj);
        end

        %---------------------------------------------------
        %------SAVE MAP: Plot all data in one figure-------
        %---------------------------------------------------
        if app.PlotalldatainonefigureButton.Value
            %Set up the filename
            if FileLength == 1 %Use raw hmax/eta filename
                OF = fullfile(app.inputtxt_directory1,string(app.FileNAME));
            else %Use a generic name
```

```
                    OF = fullfile(app.inputtxt_directory1, "WaveHeight_Output");
                end

                %Save the figure
                if app.pngCheckBox.Value
                    outputfile = OF + ".png";
                    exportgraphics(gcf,outputfile,'Resolution',300)
                end

                if app.jpgCheckBox.Value
                    outputfile = OF + ".jpg";
                    exportgraphics(gcf,outputfile,'Resolution',300)
                end
            end



        end

        % Button pushed function: Button_15
        function Button_15Pushed(app, event)


            %Load the bathymetry file and display its filename in the textbox
            [bathymetryfile,path2] = uigetfile('*', 'Select the files', 'MultiSelect', 'on');
            if ~(ismcc || isdeployed)
                addpath(genpath(fullfile(string(path2))));
            end
            app.DepthFileEditField.Value = string(bathymetryfile);
            app.bathymetryinputdata = string(path2) + string(bathymetryfile);
            app.bathymetryname = string(bathymetryfile);
            app.DepthFileEditField.BackgroundColor = 'w';
            app.DepthFileEditField.FontColor =  'k';

            %Extract information from the bathymetry
            if app.DepthFileEditField.Value == string(bathymetryfile)
                if ~strcmp(app.DepthFileEditField.Value, "0") || ~strcmp(app.DepthFileEditField.Value, "00")
                    % If the file is .tif, read the matrix, extract the geographic values and grid sizes, and display them in the app textboxes
                    if contains(app.bathymetryinputdata,'.tif')
                        [A1,R2] = readgeoraster(app.bathymetryinputdata);
                        app.bathymetrydata = flipud(A1);

                        %Extract lat and long of the southwest corner
                        app.LongitudeEditField.Value = min(R2.LongitudeLimits);
                        app.LatitudeEditField.Value = min(R2.LatitudeLimits);
                        app.LongitudeEditField.FontColor = 'k';
                        app.LatitudeEditField.FontColor = 'k';

                        %Extract grid size
                        app.gridX.Value = R2.CellExtentInLatitude;
                        app.gridY.Value = R2.CellExtentInLongitude;
                        app.gridX.FontColor = 'k';
                        app.gridY.FontColor = 'k';
                        drawnow

                        %Update the X-axis limits in the 'General Layout' section
                        if isprop(app, 'col1')
                            app.x = app.LongitudeEditField.Value + [0:app.col1] * app.gridX.Value;
                            app.EastEditField.Value = max(app.x);
                            app.WestEditField.Value = min(app.x);
                            app.EastEditField.FontColor = 'k';
                            app.WestEditField.FontColor = 'k';
                        end

                        %Update the Y-axis limits in the 'General Layout' section
                        if isprop(app, 'row1')
                            app.y = app.LatitudeEditField.Value + [0:app.row1] * app.gridY.Value;
                            app.NorthEditField.Value = max(app.y);
                            app.SouthEditField.Value = min(app.y);
                            app.NorthEditField.FontColor = 'k';
                            app.SouthEditField.FontColor = 'k';
                        end
                        drawnow

                    else   %Only read the matrix for use as a basemap and for plotting bathymetry contours
                        app.bathymetrydata = readmatrix(app.bathymetryinputdata);
                        app.MaximumEditField.Value = max(max(readmatrix(app.bathymetryinputdata)));
                    end
                end
                %Enable the 'Land Color' dropdown list in 'Basemap' section
                app.CoastlinecolourDropDown_2.Enable = "on";
            end

            if isprop(app,'bathymetrydata')
                [app.bathyRow,app.bathyCol] = size(app.bathymetrydata);
            end

        end

        % Callback function
        function ImportinfoButtonPushed(app, event)

        end

        % Value changed function: FileTextArea
        function FileTextAreaValueChanged(app, event)
            if contains(app.FileTextArea.Value, "hmax") || contains(app.FileTextArea.Value, "eta")
                app.WaveHeightContoursButton.Enable = "on";
            end

        end

        % Callback function
        function ButtonGroup_2SelectionChanged(app, event)

        end

        % Callback function
        function MinimumEditFieldValueChanged(app, event)
            app.MinimumEditField.FontColor = 'k';

        end

        % Callback function
        function MaximumEditFieldValueChanged(app, event)
            app.MaximumEditField.FontColor = 'k';
        end

        % Callback function
        function IntervalEditField_5ValueChanged(app, event)
            app.IntervalEditField_5.FontColor = 'k';
            app.IntervalEditField_3.Value = app.IntervalEditField_5.Value; % Copy the values to the contour label interval textbox
            drawnow

        end

        % Value changed function: CoastlinecolourDropDown_8
        function CoastlinecolourDropDown_6ValueChanged(app, event)
            value = app.CoastlinecolourDropDown_8.Value;
            %List of values based on the cbrewer2 colormap
            validValues = {'blue', 'blue - green', 'blue - purple', 'green - blue', ...
                'greens', 'grays', 'oranges', 'orange - red', ...
                'purple - blue', 'purple - blue - green', 'purple - red', ...
                'purples', 'red - purple', 'reds', 'yellow - green', ...
                'yellow - green - blue', 'yellow - orange - brown', ...
                'yellow - orange - red', 'brown - teal', 'pink - light green', ...
                'purple - green', 'purple - orange', 'red - blue', ...
                'red - gray', 'red - yellow - blue', 'red - yellow - green', ...
                'spectral', 'accent', 'dark 2', 'paired', 'pastel 1', ...
                'pastel 2', 'set 1', 'set 2', 'set 3'};

            % If the selected value in the dropdown list is part of the cbrewer2 colormap, enable the color interpolation textbox
            if ismember(value, validValues)
                app.DivisionEditField.Enable = "on";
                app.DivisionEditFieldLabel.Enable = "on";
            else
                app.DivisionEditField.Enable= "off";
                app.DivisionEditFieldLabel.Enable = "off";
            end

        end

        % Callback function
        function Button_8Pushed(app, event)
            try
            [filename3,path3] = uigetfile('*.txt;*.shp');
            fullname = fullfile(path3,filename3);

            [~, name, ext] = fileparts(filename3);
            if ~(ismcc || isdeployed)
                addpath(genpath(string(path3)));
            end

            app.GaugeFname = name;
            app.GAUGEFILE = filename3;
            app.FileEditField.Value = filename3;
            app.FileEditField.FontColor = 'k';
            app.FileEditField.BackgroundColor = 'w';


            if strcmp(ext, '.shp')
                S = shaperead(fullname);
                latitudes= [S.Y]';
                longitudes= [S.X]';
                list = [round(latitudes, 4), round(longitudes, 4)];
                list = unique(list, 'rows');
                app.latGauge = list(:, 1);
                app.longGauge =  list(:, 2);
            elseif strcmp(ext, '.txt')
                fileID = fopen(fullname, 'r');
                dataArray = textscan(fileID, '%f %f', 'Delimiter', 'whitespace'); %Read text file and ensure that it is tab delimited
                fclose(fileID);
                app.STATION_FILE = string(filename3);
                app.latGauge  = dataArray{1};
                app.longGauge = dataArray{2};
            end
            end



        end

        % Value changed function: CoastlinecolourDropDown_2
        function CoastlinecolourDropDown_2ValueChanged(app, event)


        end

        % Callback function
        function ButtonGroup_6SelectionChanged(app, event)

        end

        % Button pushed function: Button_2
        function Button_2Pushed(app, event)
            % Set up the directory path for saving the files
            workingfolder = uigetdir;
            Dir1 = fullfile(string(workingfolder), 'OUTPUT_FILES');
            app.WORKFOLDER = Dir1;

            %Create the 'OUTPUT_FILES/Figures' folder in the working directory
            FigureFolder = fullfile(Dir1, 'Figures');
            app.WORKFOLDER0 = FigureFolder;
            app.OutputDirectoryEditField.Value = fullfile(deblank(FigureFolder));
            app.OutputDirectoryEditField.FontColor = 'k';

            %Create the folders
            if ~exist(Dir1, 'dir')
                mkdir(Dir1);
            end
            if ~exist(FigureFolder, 'dir')
                mkdir(FigureFolder);
            end
        end

        % Callback function
        function VirtualGaugesButtonValueChanged(app, event)
            %Display the gauges tab
            app.TabGroup2.SelectedTab = app.GaugesTab;


        end

        % Callback function
        function BathymetryContoursButtonValueChanged(app, event)

        end

        % Callback function
        function WaveHeightContoursButtonValueChanged(app, event)


        end

        % Value changed function: DepthFileEditField
        function DepthFileEditFieldValueChanged(app, event)
            value = app.DepthFileEditField.Value;
            if value == app.bathymetryname
                app.BathymetryContoursButton.Enable = "on";
            end
        end

        % Callback function
        function LineintervalEditField_2ValueChanged(app, event)
            app.wh_interval.Value = app.LineintervalEditField_2.Value;
            drawnow
            app.LineintervalEditField_2.FontColor = 'k';
        end

        % Callback function
        function MaximumdepthEditField_2ValueChanged(app, event)


            app.MaximumdepthEditField_2.FontColor = 'k';
        end

        % Button pushed function: PLOTButton
        function PLOTButtonPushed(app, event)
            close all
            %------------------------
            %------Marker Style------
            %-------------------------
            GaugeMarkerOptions = containers.Map({'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'}, ...
                {'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'});
            GaugeMarker2 = GaugeMarkerOptions(app.MarkerDropDown.Value);


            %-----------------------------
            %------Line Style at y=0------
            %-----------------------------
            YLineStyleOptions = containers.Map({'None','Solid', 'Dashed', 'Dotted', 'Dash-dotted'}, {'off','-', '--', ':', '-.'});
            YLineStyle = YLineStyleOptions(app.Lineaty0DropDown.Value);

            %-----------------------
            %------Line Style------
            %-----------------------
            GLineStyleOptions = containers.Map({'Solid', 'Dashed', 'Dotted', 'Dash-dotted'}, {'-', '--', ':', '-.'});
            GLineStyle = GLineStyleOptions(app.GaugeLineStyle.Value);

            %---------------------------------------------
            %------Line Color for one data per plot------
            %---------------------------------------------
            colors = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            colorCodes = {'k', '[0.4 0.4 0.4]', '[0.7 0.7 0.7]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'};
            index = find(strcmp(colors, app.GaugeLineColor.Value));
            GLineColor0 = colorCodes{index};

            %-------------------------------------------------------
            %------Line Color for multiple data in one graph------
            %--------------------------------------------------------
            LineColorOptions = containers.Map({'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', ...
                'gray', 'hot', 'hsv', 'jet', 'parula', 'pink', 'prism', 'spring', 'summer', ...
                'turbo', 'winter', 'lines'}, ...
                {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', ...
                'hsv', 'jet', 'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', 'lines'});

            GLineColor = LineColorOptions(app.GaugeLineColor_multiple.Value);

            %---------------------------
            %------Legend Location------
            %---------------------------
            LegendLocationOptions = containers.Map({'North', 'South', 'East', 'West', 'Northeast', ...
                'Northwest', 'Southeast', 'Southwest', 'Northoutside', 'Southoutside', 'Eastoutside', ...
                'Westoutside', 'Northeastoutside', 'Northwestoutside', 'Southeastoutside', 'Southwestoutside', ...
                'Best', 'Bestoutside', 'None'}, ...
                {'north', 'south', 'east', 'west', 'northeast', 'northwest', 'southeast', 'southwest', ...
                'northoutside', 'southoutside', 'eastoutside', 'westoutside', 'northeastoutside', ...
                'northwestoutside', 'southeastoutside', 'southwestoutside', 'best', 'bestoutside', 'none'});
            legloc = LegendLocationOptions(app.LocationDropDown.Value);
            app.LegendLocation = legloc;

            %--------------------------------------
            %------SAVE MAP: Output Directory------
            %--------------------------------------
            %Check if the output directory has been manually set by the user
            if  app.OutputDirectoryEditField_2.Value == string(app.WORKFOLDER5)
                inputtxt_directory3 =  fullfile(app.WORKFOLDER2);
                Dir1 = fullfile(app.WORKFOLDER5);
                app.OutputDirectoryEditField_2.FontColor = 'k';
                if ~exist(inputtxt_directory3, 'dir')
                    mkdir(inputtxt_directory3);
                end
                if ~exist(Dir1, 'dir')
                    mkdir(Dir1);
                end
            else
                % Set the default directory to the Desktop
                if ismac
                    %macOS
                    defaultDir = fullfile(getenv('HOME'), 'Desktop');
                elseif ispc
                    %Windows
                    defaultDir = fullfile(getenv('USERPROFILE'), 'Desktop');
                else
                    %Others
                    defaultDir = pwd;
                end

                %Create 'OUTPUT_FILES' folder and 'Figures' subfolder
                inputtxt_directory3 = fullfile(defaultDir, 'OUTPUT_FILES');
%                 Dir1 = fullfile(defaultDir, 'OUTPUT_FILES/Figures');
                Dir1 = fullfile(inputtxt_directory3,'Figures');
                if ~exist(inputtxt_directory3, 'dir')
                    mkdir(inputtxt_directory3);
                end
                if ~exist(Dir1, 'dir')
                    mkdir(Dir1);
                end

                %Display the path directory of the 'Figures' subfolder in the textbox
                app.OutputDirectoryEditField_2.Value = deblank(string(Dir1));
                app.OutputDirectoryEditField_2.FontColor = 'k';
            end


            %----------------------
            %------LOG REPORT------
            %----------------------
            %Create 'Log_Files' subfolder in 'OUTPUT_FILES'
            LogFolder = fullfile(inputtxt_directory3, 'Log_Files');
            if ~exist(LogFolder, 'dir')
                mkdir(LogFolder);
            end

            %Create the text file
            fileId = fopen(fullfile(LogFolder, 'Log_Report_GaugeRecords.txt'),'w'); %if same file exists, overwrite the file
            fileList = app.FileTextArea_2.Value;

            %Template for the default section of the log report (includes information from the 'Input Data' section)
            fprintf(fileId, '%s\n','******************** LOG REPORT ******************** ');
            fprintf(fileId, '%-30s%-s\n', 'Type:', 'Gauge Records');
            timestamp = datestr(now, 'yyyy-mm-dd HH:MM:SS'); % Get the current timestamp
            fprintf(fileId, '%-30s%-s\n','Timestamp:', timestamp); %Add timestamp in the header
            fprintf(fileId, '%s\n','****************************************************');
            fprintf(fileId,'%-s\n\n','');
            fprintf(fileId,'%-30s%-s\n','X axis:',app.XAxisEditField.Value);
            fprintf(fileId,'%-30s%-s\n','Y axis:',app.YAxisEditField.Value);


            %------LOG REPORT--------------------
            %------Input Data: Import Files------
            %------------------------------------
            %If no file is uploaded
            if any(app.FileTextArea_2.Value == "0" | app.FileTextArea_2.Value == "00") || isempty(fileList)
                fprintf(fileId, '%-30s%-s\n', 'IMPORT FILE/S:', 'Empty. No data plotted');
            end

            %Check if the name of each uploaded file starts with 'sta_'
            for i = 1:numel(fileList)
                if any(contains(fileList, 'sta_xxxx')) || ~any(contains(fileList(i), 'sta_'))
                    fprintf(fileId, '%-30s%-s\n', 'IMPORT FILE/S:', 'Invalid filename format. Use filenames with "sta_"');
                    fprintf(fileId, '%-30s%-s\n', '', string(fileList(i)));
                end
            end

            %List all the files loaded
            fprintf(fileId, '%-30s%-s\n', 'Loaded Files:', [num2str(length(app.GaugeDirectorylist)),' files']);
            fprintf(fileId, '%s\n\n','____________________________________________________');

            %------LOG REPORT---------------------------
            %------General Layout: Boundary Limits------
            %--------------------------------------------
            %Check the X limits and interval of the plot if 'Auto Set' is not
```

```
selected for the limits
            if ~app.CheckBox_2.Value
                if app.xMinLimit.Value > app.xMaxLimit.Value
                    fprintf(fileId, '%-30s%-s\n', 'X Limit:','Max value must exceed min value');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Max and min values are swapped');
                end
                if app.xLimInterval.Value > app.xMaxLimit.Value - app.xMinLimit.Value
                    fprintf(fileId, '%-30s%-s\n', 'X Limit Interval:','Value must be less than the max-min difference');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
                end
            end

            %Check the Y limits and interval of the plot if 'Auto Set' is not selected for the limits
            if ~app.CheckBox.Value
                if app.yMinLimit.Value > app.yMaxLimit.Value
                    fprintf(fileId, '%-30s%-s\n', 'Y Limit:','Max value must exceed min value');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Max and min values are swapped');
                end
                if  app.yLimInterval.Value > app.yMinLimit.Value - app.yMaxLimit.Value
                    fprintf(fileId, '%-30s%-s\n', 'Y Limit Interval:','Value must be less than the max-min difference');
                    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
                end
            end

            %% PROCESS THE INPUT FILES AND VALUES, THEN GENERATE THE FIGURES
            %---------------------------------
            %------READ THE INPUT FILES------
            %---------------------------------
            clear xData yData yData2
            %Check if any file is loaded
            if app.FileTextArea_2.Value ~=  string(app.GaugeNames)
                app.FileTextArea_2.Value = "NO FILE";
            else
                %Read the data
                FileLength = length(app.GaugeDirectorylist);
                plotHandles = [];
                legendEntries = {};

                for i = 1:FileLength
                    g = readmatrix(app.GaugeDirectorylist(i)); %Read all matrices

                    %Calculate the magnitude of the Z-component of velocity using the U and V velocity values in columns 3 and 4.
                    Zvelocity = sqrt(g(:,3).^2 + g(:,4).^2);
                    %Include the Z velocity as the fifth column
                    g = [g, Zvelocity];
```

```
    %Extracting the station numbers from the filename
    GN = erase(app.GaugeNames(i),"sta_");  %remove 'sta_' from the name
    GN2 = strip(GN,"left",'0'); %remove the trailing zeroes on the left side

    if contains(GN2,'_')
        GaugeNumber{i} = regexprep(GN2,'_',' '); %Replace "_" to space
    else
        GaugeNumber{i} = GN2;
    end

    %Choose the column from which to extract data points for the X plot values
    app.XCOLGAUGE = str2double(app.XaxisUseDataDropDown.Value);
    app.YCOLGAUGE = str2double(app.YaxisUseDataDropDown.Value);

    %Convert data loaded for X axis
    if app.CheckBox_3.Value
        g(:,app.XCOLGAUGE) =  g(:,app.XCOLGAUGE) * (app.numeratorX.Value/app.denomeratorX.Value);
    end

   %If '3,4,5' or '5' option is selected, calculate Z velocity vector
    if strcmp(app.YaxisUseDataDropDown.Value, '3,4,5') || strcmp(app.YaxisUseDataDropDown.Value, '5')
    %Calculate the magnitude of the Z-component of velocity using the U and V velocity values in columns 3 and 4.
    Zvelocity = sqrt(g(:,3).^2 + g(:,4).^2);
    %Include the Z velocity as the fifth column
    g = [g, Zvelocity];
    end

    %Convert data loaded for Y axis
    if app.CheckBox_4.Value %Convert data loaded for Y axis
        if ~strcmp(app.YaxisUseDataDropDown.Value, '3,4,5') %If only one column of data will be used
            g(:,app.YCOLGAUGE) =  g(:,app.YCOLGAUGE) * (app.numeratorY.Value / app.denomeratorY.Value);
        else
            %Convert the values
            g(:, 3) =  g(:, 3) * (app.numeratorY.Value / app.denomeratorY.Value);
            g(:, 4) =  g(:, 4) * (app.numeratorY.Value / app.denomeratorY.Value);
            g(:, 5) =  g(:, 5) * (app.numeratorY.Value / app.denomeratorY.Value);
        end
    end

    %Extract the required Y data based on dropdown selection
    if ~strcmp(app.YaxisUseDataDropDown.Value, '3,4,5')
        yData = g(:,str2double(app.YaxisUseDataDropDown.Value));
    else
        yData = g(:, 3);
        y2Data = g(:, 4);
        y3Data = g(:, 5);
    end

    %Sort data list to avoid connecting lines between the first and last points
    xData = double(g(:, app.XCOLGAUGE));
    [xDataArranged, sortIndex] = sort(xData);
    yDataArranged= yData(sortIndex);
    if strcmp(app.YaxisUseDataDropDown.Value, '3,4,5') %If columns 3, 4, 5 are to be plotted together
        y2DataArranged= y2Data(sortIndex);
        y3DataArranged= y3Data(sortIndex);
    end

    %---------------------
    %------MAIN PLOT------
    %---------------------
    %Select the line color of the plots
    if app.PlotalldatainonegraphButton.Value
        a = str2func(GLineColor);
        rndm = a(FileLength); %divide the colormap based on the number of uploaded dataset

        if app.FlipCheckBox_3.Value % Flip colors
            app.colorcombi = rndm(FileLength-i+1,:);
        else
            app.colorcombi = rndm(i,:);
        end
    elseif app.PlotseparatelyButton_4.Value
        app.colorcombi = GLineColor0; %For single line plot
    end

    %--------------------------------------
    %------PLOT ALL DATA IN ONE GRAPH------
    %--------------------------------------
        if app.PlotalldatainonegraphButton.Value %Plot all sta_ data into one figure
            figure(1);
            set(gcf, 'Color', 'w');
            hold on;

            if strcmp(app.YaxisUseDataDropDown.Value, '3,4,5')
                %Main plotting loop
                %U vector
                ph1 = plot(xDataArranged, yDataArranged, 'LineWidth', app.DataLineWidth.Value,'LineStyle', '--', 'Marker', GaugeMarker2, ...
                    'MarkerEdgeColor', app.colorcombi, 'MarkerFaceColor', app.colorcombi, 'Color', [0.596, 0.404, 0.553]);
                hold on
                %V vector
                ph2 = plot(xDataArranged, y2DataArranged,
'LineWidth', app.DataLineWidth.Value, 'LineStyle','--','Marker',GaugeMarker2,'MarkerEdgeColor',app.colorcombi,'MarkerFaceColor',app.colorcombi,'Color', [0.4, 0.596, 0.447]);
                hold on
                %Z vector
                ph3 = plot(xDataArranged, y3DataArranged, 'LineWidth', app.DataLineWidth.Value, 'LineStyle', GLineStyle,'Marker',GaugeMarker2,'MarkerEdgeColor',app.colorcombi,'MarkerFaceColor',app.colorcombi,'Color','k');
                hold off


                plotHandles(end+1) = ph1;
                plotHandles(end+1) = ph2;
                plotHandles(end+1) = ph3;

                legendEntries{end+1} = sprintf('%s %s U vector', string(app.legendfirsttext.Value), string(GaugeNumber(min(i, length(GaugeNumber)))));
                legendEntries{end+1} = sprintf('%s %s V vector', string(app.legendfirsttext.Value), string(GaugeNumber(min(i, length(GaugeNumber)))));
                legendEntries{end+1} = sprintf('%s %s Z vector', string(app.legendfirsttext.Value), string(GaugeNumber(min(i, length(GaugeNumber)))));

            else %Plot data from only one column
                ph1 = plot(xDataArranged, yDataArranged, 'LineWidth', app.DataLineWidth.Value,'LineStyle', GLineStyle, 'Marker',GaugeMarker2, ...
                    'MarkerEdgeColor', app.colorcombi, 'MarkerFaceColor', app.colorcombi, 'Color', app.colorcombi);
                plotHandles(end+1) = ph1;   % Store first plot handle
                legendEntries{end+1} = sprintf('%s %s', string(app.legendfirsttext.Value), string(GaugeNumber(min(i, length(GaugeNumber)))));
            end

            % Update legend outside the loop with all entries
            legend(plotHandles, legendEntries, 'Location', legloc, 'FontSize', app.LegendSize.Value);
            legend box off;
            hold off;

            %-----------------------------------------
            %------GENERAL LAYOUT: Figure Size-------
            %-----------------------------------------
            if ~app.AutoSetCheckBox.Value
                %Manually set the figure size
                figureHandle = ancestor(ph1, 'figure');
                set(figureHandle, 'Units', 'Inches', 'Position', [0, 0, app.Width.Value, app.Height.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.Width.Value, app.Height.Value]);
                set(gcf, 'Units', 'Inches', 'Position', [0, 0, app.Width.Value, app.Height.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.Width.Value, app.Height.Value]);
```

```
    end

end

%----------------------------
%------PLOT SEPARATELY------
%----------------------------
if app.PlotseparatelyButton_4.Value %Plot each sta_ data in separate figures

    % Open a new figure for each file
    figure(i);
    clf %clean figure so legend won't repeat
    set(gcf, 'Color', 'w'); % Set figure background to white

    % Reset legend values for each figure
    plotHandles = [];
    legendEntries = {};

    hold on
    if strcmp(app.YaxisUseDataDropDown.Value, '3,4,5')
        %Main plotting loop
        %U vector
        ph1 = plot(xDataArranged, yDataArranged, 'LineWidth', app.DataLineWidth.Value,'LineStyle', '--', 'Marker', GaugeMarker2, ...
            'MarkerEdgeColor', app.colorcombi, 'MarkerFaceColor', app.colorcombi, 'Color', [0.596, 0.404, 0.553]);
        hold on
        %V vector
        ph2 = plot(xDataArranged, y2DataArranged, 'LineWidth', app.DataLineWidth.Value, 'LineStyle', '--','Marker',GaugeMarker2,'MarkerEdgeColor',app.colorcombi,'MarkerFaceColor',app.colorcombi,'Color',[0.4, 0.596, 0.447]);
        disp(ph2);
        hold on
        %Z vector
        ph3 = plot(xDataArranged, y3DataArranged, 'LineWidth', app.DataLineWidth.Value, 'LineStyle', GLineStyle,'Marker',GaugeMarker2,'MarkerEdgeColor',app.colorcombi,'MarkerFaceColor',app.colorcombi,'Color','k');
        hold off

        plotHandles(end+1) = ph1;
        plotHandles(end+1) = ph2;
        plotHandles(end+1) = ph3;

        %Add each legend entry
        legendEntries{end+1} = 'U vector';
        legendEntries{end+1} = 'V vector';
        legendEntries{end+1} = 'Z vector';

        legend(plotHandles, legendEntries, 'Location', legloc, 'FontSize', app.LegendSize.Value);
        legend box off;

    else %Plot data from only one column
```

```
                                ph1 = plot(xDataArranged, yDataArranged, 'LineWidth', app.DataLineWidth.Value,'LineStyle', GLineStyle, 'Marker', GaugeMarker2, ...
                                    'MarkerEdgeColor', app.colorcombi, 'MarkerFaceColor', app.colorcombi, 'Color', app.colorcombi);
                            end

                            titleText = sprintf('%s %s', string(app.legendfirsttext.Value), string(GaugeNumber(min(i, length(GaugeNumber)))));
                            title(titleText);

                            %Figure Size
                            if ~app.AutoSetCheckBox.Value
                                figureHandle = ancestor(ph1, 'figure'); % Get the figure handle containing the axes
                                set(figureHandle, 'Units', 'Inches', 'Position', [0, 0, app.Width.Value, app.Height.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.Width.Value, app.Height.Value]);
                                set(gcf, 'Units', 'Inches', 'Position', [0, 0, app.Width.Value, app.Height.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.Width.Value, app.Height.Value]);
                            end
                        end


                        %-------------------------------------------
                        %------GENERAL LAYOUT: Boundary Limit-------
                        %-------------------------------------------
                        %Set X axis limit
                        if app.CheckBox_2.Value
                            xlim("auto")
                            xticks('auto')
                        else %Using the input values
                            %Swap the minimum and maximum input values of X-axis limits when needed
                            if app.xMinLimit.Value > app.xMaxLimit.Value
                                temp3 = app.xMinLimit.Value;
                                app.xMinLimit.Value = app.xMaxLimit.Value;
                                app.xMaxLimit.Value = temp3;
                            end

                            %Check the interval value if it is within the limits
                            if app.xLimInterval.Value > (app.xMaxLimit.Value-app.xMinLimit.Value)
                                app.xLimInterval.Value = (app.xMaxLimit.Value-app.xMinLimit.Value)/2;
                            end

                            xlim([app.xMinLimit.Value app.xMaxLimit.Value])
                            xticksLOC = app.xMinLimit.Value:app.xLimInterval.Value:app.xMaxLimit.Value;
                            xticks(xticksLOC)
                            tickXname = string(xticksLOC);
                            xticklabels(tickXname)
                        end

```

```
                %Set Y axis limit
                if app.CheckBox.Value
                    ylim("auto")
                    yticks('auto')
                else %Using the input values
                    % Swap the minimum and maximum input values of Y-axis limits when needed
                    if app.yMinLimit.Value > app.yMaxLimit.Value
                        temp4 = app.yMinLimit.Value;
                        app.yMinLimit.Value = app.yMaxLimit.Value;
                        app.yMaxLimit.Value = temp4;
                    end

                    %Check the interval value if it is within the limits
                    if app.yLimInterval.Value > (app.yMaxLimit.Value-app.yMinLimit.Value) %Check the interval value if it is within the limits
                        app.yLimInterval.Value = (app.yMaxLimit.Value-app.yMinLimit.Value)/2;
                    end

                    ylim([app.yMinLimit.Value app.yMaxLimit.Value]);
                    yticksLOC = [app.yMinLimit.Value:app.yLimInterval.Value:app.yMaxLimit.Value];
                    yticks(yticksLOC)
                    tickYname = string(yticksLOC);
                    yticklabels(tickYname)
                    ytickformat('%.1f')
                end


            %---------------------------------------------
            %------GENERAL LAYOUT: Background Grid-------
            %---------------------------------------------
            ax = gca;
            ax.FontSize = app.GridLabelSize.Value;
            ax.LineWidth = app.LineYThickness_2.Value;
            if strcmp(app.GridStyle.Value, 'none')
                grid(ax, 'off');
            elseif strcmp(app.GridStyle.Value, 'X axis only')
                ax.XGrid = 'on';
                ax.YGrid = 'off';
                ax.XMinorGrid = 'off';
                ax.YMinorGrid = 'off';
            elseif strcmp(app.GridStyle.Value, 'Y axis only')
                ax.XGrid = 'off';
                ax.YGrid = 'on';
                ax.XMinorGrid = 'off';
                ax.YMinorGrid = 'off';
            elseif strcmp(app.GridStyle.Value, 'Both - major lines')
                grid(ax, 'on');
                ax.XMinorGrid = 'off';
                ax.YMinorGrid = 'off';
            elseif strcmp(app.GridStyle.Value, 'Both - with minor lines')
                grid(ax, 'on');
```

```
    ax.XMinorGrid = 'on';
    ax.YMinorGrid = 'on';
end

%Grid line at y = 0 for all data in one graph
if ~strcmp(app.Lineaty0DropDown.Value,"None")
    yhorzline = yline(0,YLineStyle,'LineWidth',app.LineYThickness.Value);
    yhorzline.Annotation.LegendInformation.IconDisplayStyle = 'off';
end


%-----------------------------------
%------GENERAL LAYOUT: Ticks-------
%-----------------------------------
%Labels of Axes and Ticks
xlabel(app.XAxisEditField.Value,'FontSize',app.GridLabelSize.Value);
ylabel(app.YAxisEditField.Value,'FontSize',app.GridLabelSize.Value);


%---------------------------------------
%------SAVE MAP: Plot Separately-------
%---------------------------------------
if app.PlotseparatelyButton_4.Value
    %Save the maps with filenames matching the loaded file
    PlotsFolder = fullfile(Dir1);
    OF = fullfile(PlotsFolder, app.GaugeNames(i));

    %Options for saving format
    %PNG
    if app.pngCheckBox_2.Value
        outputfile = OF + ".png";
        exportgraphics(ax,outputfile,'Resolution',300)
    end

    %JPG
    if app.jpgCheckBox_2.Value
        outputfile = OF + ".jpg";
        exportgraphics(ax,outputfile,'Resolution',300)
    end

    %PDF
    if app.pdfCheckBox_2.Value
        outputfile = OF + ".pdf";
        exportgraphics(ax,outputfile,'ContentType','vector','Resolution',300)
    end

    %EPS
    if app.epsCheckBox.Value %save as .eps
        outputfile = OF + ".eps";
        saveas(ax,outputfile,'epsc')
    end

```

```
                            %TXT for saving the raw 'sta_' files
                            if app.txtCheckBox.Value
                                outputfile = OF + ".txt";
                                writematrix(g, outputfile, 'Delimiter', '\t');
                                dlmwrite(outputfile,g,'delimiter', '\t');
                            end
                        end
                end

                %Add legend in the plot
                if app.PlotseparatelyButton_4.Value
                    if strcmp(app.YaxisUseDataDropDown.Value, '3,4,5')
                        legend(plotHandles, legendEntries, 'Location', legloc, 'FontSize', app.LegendSize.Value);
                        legend box off;
                    end
                end

                %------------------------------------------------
                %------SAVE MAP: Plot all data in one graph-------
                %------------------------------------------------
                if app.PlotalldatainonegraphButton.Value
                    if length(app.GaugeNames) == 1
                        OF = fullfile(Dir1, app.GaugeNames);
                    else
                        OF = fullfile(Dir1, 'GaugeRecord');
                    end

                    if app.jpgCheckBox_2.Value
                        outputfile = OF + ".jpg";
                        exportgraphics(gcf,outputfile,'Resolution',300)
                    end

                    if app.pngCheckBox_2.Value
                        outputfile = OF + ".png";
                        exportgraphics(gcf,outputfile,'Resolution',300)
                    end
                    if app.pdfCheckBox_2.Value %save as .pdf
                        outputfile = OF + ".pdf";
                        exportgraphics(ax, outputfile,'ContentType','vector','Resolution',300)
                    end
                    if app.epsCheckBox.Value %save as .eps
                        outputfile = OF + ".eps";
                        saveas(gcf,outputfile,'epsc')
                        pause(2)
                        clf;
                    end
                    if app.txtCheckBox.Value %save as .txt
                        outputfile = OF + ".txt";
                        writematrix(g,outputfile);
                        dlmwrite(outputfile,g,'delimiter', '\t');
                    end
                end
            end
        end

        % Callback function
        function BathymetryCheckBoxValueChanged(app, event)

        end

        % Callback function
        function WaveHeightCheckBoxValueChanged(app, event)


        end

        % Value changed function: GaugesCheckBox
        function GaugesCheckBoxValueChanged(app, event)
            if app.GaugesCheckBox.Value
                %Display and enable the parameter options for the gauges tab
                app.TabGroup2.SelectedTab = app.GaugesTab;
                for i = 1:length(app.GaugesTab.Children)
                        app.GaugesTab.Children(i).Enable = 'on';
                end
                    %If 'Add Labels' is checked, enable the items below it
                    components = {app.GLspacing, app.GLspacingLabel, app. ↙
SizeEditField_3, app.SizeEditField_3Label, app.CoastlinecolourDropDown_5, app. ↙
CoastlinecolourDropDown_4, app.AlignmentLabel};

                       for i = 1:length(components)
                            if  app.AddLabelsCheckBox_4.Value
                                components{i}.Enable = 'on';
                            else
                                components{i}.Enable = 'off';
                            end
                       end
            else
                for i = 1:length(app.GaugesTab.Children)
                    if isprop(app.GaugesTab.Children(i), 'Enable')
                        app.GaugesTab.Children(i).Enable = 'off';
                    end
                end
            end
        end

        % Button pushed function: Button_16
        function Button_16Pushed(app, event)
             %Find files to import from a directory
            [files, path] = uigetfile('*.*','Select the files','MultiSelect',↙
'on');
            allfiles = string(path) + string(files);
            if ~(ismcc || isdeployed)
                addpath(genpath(fullfile(string(path))));
            end

            cd(fullfile(path))

            %Sort the files by snapshot timing, and display the list of↙
filenames in the app's textbox
            app.GaugeDirectorylist = sort(allfiles);
            app.GaugeNames = sort(string(files));
            app.FileTextArea_2.Value = string(app.GaugeNames);
            app.FileTextArea_2.FontColor = 'k';
        end

        % Value changed function: FileTextArea_2
        function FileTextArea_2ValueChanged(app, event)


        end

        % Value changed function: CheckBox
        function CheckBoxValueChanged(app, event)
            if app.CheckBox.Value
                app.yMaxLimit.Enable = "off";
                app.yMinLimit.Enable = "off";
                app.yLimInterval.Enable = "off";
            else
                app.yMaxLimit.Enable = "on";
                app.yMinLimit.Enable = "on";
                app.yLimInterval.Enable = "on";
            end
        end

        % Value changed function: CheckBox_2
        function CheckBox_2ValueChanged(app, event)
            if app.CheckBox_2.Value
                app.xMaxLimit.Enable = "off";
                app.xMinLimit.Enable = "off";
                app.xLimInterval.Enable = "off";
            else
                app.xMaxLimit.Enable = "on";
                app.xMinLimit.Enable = "on";
                app.xLimInterval.Enable = "on";
            end
        end

        % Value changed function: yLimInterval
        function yLimIntervalValueChanged(app, event)
            app.yLimInterval.FontColor = 'k';


        end

        % Value changed function: xLimInterval
        function xLimIntervalValueChanged(app, event)


        end

        % Button pushed function: Button_18
        function Button_18Pushed(app, event)
            % Set up the directory path for saving the files
            workingfolder = uigetdir;
            Dir1 = fullfile(string(workingfolder), 'OUTPUT_FILES');
            app.WORKFOLDER2 = Dir1;

            %Create the 'Figures' folder in the 'OUTPUT_FILES' working directory
            FigureFolder = fullfile(Dir1, 'Figures');
            app.WORKFOLDER5 = FigureFolder;
            app.OutputDirectoryEditField_2.Value = FigureFolder;
            app.OutputDirectoryEditField_2.FontColor = 'k';

            %Create the folders
            if ~exist(Dir1, 'dir')
                mkdir(Dir1);
            end
            if ~exist(FigureFolder, 'dir')
                mkdir(FigureFolder);
            end

        end

        % Value changed function: yMinLimit
        function yMinLimitValueChanged(app, event)
            app.yMinLimit.FontColor = 'k';
        end

        % Value changed function: yMaxLimit
        function yMaxLimitValueChanged(app, event)
            app.yMaxLimit.FontColor = 'k';
        end

        % Value changed function: xMinLimit
        function xMinLimitValueChanged(app, event)
            app.xMinLimit.FontColor = 'k';


        end

        % Value changed function: xMaxLimit
        function xMaxLimitValueChanged(app, event)

        end

        % Value changed function: GridStyle
        function GridStyleValueChanged(app, event)
            value = app.GridStyle.Value;
            if strcmp(value,"None")
                app.LineYThickness_2.Enable = "off";
                app.LineYThickness_2.Editable = "off";
            else
                app.LineYThickness_2.Enable = "on";
                app.LineYThickness_2.Editable = "on";
            end

        end

        % Drop down opening function: GridStyle
        function GridStyleDropDownOpening(app, event)

        end

        % Value changed function: Lineaty0DropDown
        function Lineaty0DropDownValueChanged(app, event)
            value = app.Lineaty0DropDown.Value;
            if strcmp(value,"None")
                app.LineYThickness.Enable = "off";
            else
```

```
                app.LineYThickness.Enable = "on";
            end
        end

        % Value changed function: LocationDropDown
        function LocationDropDownValueChanged(app, event)
            value = app.LocationDropDown.Value;
            if strcmp(value,"None")
                app.LegendSize.Enable = "off";
            else
                app.LegendSize.Enable = "on";
            end

        end

        % Callback function
        function XCOLValueChanged(app, event)


        end

        % Callback function
        function YCOLValueChanged(app, event)

        end

        % Value changed function: DataLineWidth
        function DataLineWidthValueChanged(app, event)
            app.DataLineWidth.FontColor = 'k';

        end

        % Value changed function: XAxisEditField
        function XAxisEditFieldValueChanged(app, event)
            app.XAxisEditField.FontColor = 'k';

        end

        % Value changed function: YAxisEditField
        function YAxisEditFieldValueChanged(app, event)
            app.YAxisEditField.FontColor = 'k';

        end

        % Value changed function: GridLabelSize
        function GridLabelSizeValueChanged(app, event)
            app.GridLabelSize.FontColor = 'k';

        end

        % Value changed function: LineYThickness
        function LineYThicknessValueChanged(app, event)
            app.LineYThickness.FontColor = 'k';

        end

        % Callback function
        function GridThicknessValueChanged(app, event)
```

```
            app.GridThickness.FontColor = 'k';

        end

        % Value changed function: LegendSize
        function LegendSizeValueChanged(app, event)
            app.LegendSize.FontColor = 'k';

        end

        % Value changed function: NorthEditField
        function NorthEditFieldValueChanged(app, event)
            app.NorthEditField.FontColor = 'k';


        end

        % Value changed function: SouthEditField
        function SouthEditFieldValueChanged(app, event)
            app.SouthEditField.FontColor = 'k';

        end

        % Value changed function: EastEditField
        function EastEditFieldValueChanged(app, event)
            app.EastEditField.FontColor = 'k';

        end

        % Value changed function: WestEditField
        function WestEditFieldValueChanged(app, event)
            app.WestEditField.FontColor = 'k';

        end

        % Value changed function: toEditField
        function toEditFieldValueChanged(app, event)
            app.toEditField.FontColor = 'k';

        end

        % Value changed function: MinEditField
        function MinEditFieldValueChanged(app, event)
            app.MinEditField.FontColor = 'k';

        end

        % Value changed function: MaxEditField_2
        function MaxEditField_2ValueChanged(app, event)
            app.MaxEditField_2.FontColor = 'k';

        end

        % Value changed function: MinEditField_2
        function MinEditField_2ValueChanged(app, event)
            app.MinEditField_2.FontColor = 'k';

        end

        % Callback function
        function LabelSpacingEditFieldValueChanged(app, event)
            app.LabelSpacingEditField.FontColor = 'k';

        end

        % Callback function
        function IntervalEditField_3ValueChanged(app, event)
            app.IntervalEditField_3.FontColor = 'k';

        end

        % Callback function
        function WidthEditFieldValueChanged(app, event)
            app.WidthEditField.FontColor = 'k';

        end

        % Value changed function: ColorinterpolationEditField_2
        function ColorinterpolationEditField_2ValueChanged(app, event)
            app.ColorinterpolationEditField_2.FontColor = 'k';

        end

        % Value changed function: DivisionEditField
        function DivisionEditFieldValueChanged(app, event)
            app.DivisionEditField.FontColor = 'k';

        end

        % Callback function
        function MinimumdepthEditField_2ValueChanged(app, event)
            app.MinimumdepthEditField_2.FontColor = 'k';

        end

        % Callback function
        function ThicknessEditField_2ValueChanged(app, event)
            app.ThicknessEditField_2.FontColor = 'k';

        end

        % Callback function
        function wh_intervalValueChanged(app, event)
            app.wh_interval.FontColor = 'k';
        end

        % Callback function
        function LabelSpacingEditField_2ValueChanged(app, event)
            app.LabelSpacingEditField_2.FontColor = 'k';

        end

        % Callback function
        function SizeEditFieldValueChanged(app, event)
            app.SizeEditField.FontColor = 'k';

        end

        % Callback function
        function SizeEditField_3ValueChanged(app, event)
            app.SizeEditField_3.FontColor = 'k';

        end

        % Value changed function: CheckBox_3
        function CheckBox_3ValueChanged(app, event)
            %If checked, activate the textboxes for X-axis data in the 'Convert Values' section of the 'Input Data'
            if app.CheckBox_3.Value
                app.denomeratorX.Enable = "on";
                app.numeratorX.Enable = "on";
            else
                app.denomeratorX.Enable = "off";
                app.numeratorX.Enable = "off";
            end
        end

        % Value changed function: CheckBox_4
        function CheckBox_4ValueChanged(app, event)
            %If checked, activate the textboxes for Y-axis data in the 'Convert Values' section of the 'Input Data'
            if app.CheckBox_4.Value
                app.denomeratorY.Enable = "on";
                app.numeratorY.Enable = "on";
            else
                app.denomeratorY.Enable = "off";
                app.numeratorY.Enable = "off";
            end
        end

        % Callback function
        function XaxisEditFieldValueChanging(app, event)

            if event.Value
                app.egconvertcolumn1datafromsecondstominutesLabel.Visible = "off";
            end
        end

        % Callback function
        function YaxisEditFieldValueChanging(app, event)
            if event.Value
                app.egconvertcolumn2datafrommeterstocentimetersLabel.Visible = "off";
            end
        end

        % Value changed function: denomeratorY
        function denomeratorYValueChanged(app, event)

        end

        % Selection changed function: ButtonGroup_10
        function ButtonGroup_10SelectionChanged(app, event)
```

```
            %Activate/deactivate the dropdown list for col  if str2double(app.YaxisUseDataDropDown.Value) == 1or line depending on the number of files uploaded and the selected plotting style.
            value = str2double(app.YaxisUseDataDropDown.Value);
            isSingleStation = length(app.GaugeDirectorylist) == 1;
            isMultipleStations = length(app.GaugeDirectorylist) > 1;
            plotAllData = app.PlotalldatainonegraphButton.Value;
            plotSeparately = app.PlotseparatelyButton_4.Value;

            if ~ismember(value, [3, 4, 5]) % Will only plot data from one column
                if isSingleStation
                    setGaugeLineColorVisibility(true, false, false);
                elseif isMultipleStations
                    if plotAllData
                        setGaugeLineColorVisibility(false, true, true);
                    elseif plotSeparately
                        setGaugeLineColorVisibility(true, false, false, false);
                    end
                end
            else % Use columns 3, 4, 5 to plot in Y-axis
                if isSingleStation
                    setGaugeLineColorVisibility(true, false, false);
                elseif isMultipleStations
                    if plotAllData
                        setGaugeLineColorVisibility(false, true, true);
                    elseif plotSeparately
                        setGaugeLineColorVisibility(true, false, false, false);
                    end
                end
            end

            function setGaugeLineColorVisibility(lineColorVisible, lineColorMultipleVisible, flipCheckBoxVisible, flipCheckBoxEnable)
                if nargin < 4
                    flipCheckBoxEnable = true; % Default to true if not specified
                end
                app.GaugeLineColor.Visible = lineColorVisible;
                app.GaugeLineColor_multiple.Visible = lineColorMultipleVisible;
                app.FlipCheckBox_3.Visible = flipCheckBoxVisible;
                app.FlipCheckBox_3.Enable = flipCheckBoxEnable;
            end
%             if ~ismember(str2double(app.YaxisUseDataDropDown.Value), [3, 4, 5])%Will only plot data from one column
%                 if length(app.GaugeDirectorylist) == 1 %one station, use dropdown for selecting one color
%                     app.GaugeLineColor.Visible = "on";
%                     app.GaugeLineColor_multiple.Visible = "off";
%                     app.FlipCheckBox_3.Visible = "off";
%                 elseif length(app.GaugeDirectorylist) > 1  && app.PlotalldatainonegraphButton.Value %multiple stations, use dropdown for selecting a set of color
%                     app.GaugeLineColor.Visible = "off";
%                     app.GaugeLineColor_multiple.Visible = "on";
%                     app.FlipCheckBox_3.Visible = "on";
%                 elseif length(app.GaugeDirectorylist) > 1  && app.PlotseparatelyButton_4.Value
%                     app.FlipCheckBox_3.Enable = "off";
%                     app.GaugeLineColor.Visible = "on";
%                     app.GaugeLineColor_multiple.Visible = "off";
%                 end
%             else %Use columns 3,4,5 to plot in Y-axis
%                 if length(app.GaugeDirectorylist) == 1 %one station, use dropdown for selecting one color
%                     app.GaugeLineColor.Visible = "on";
%                     app.GaugeLineColor_multiple.Visible = "off";
%                     app.FlipCheckBox_3.Visible = "off";
%                 elseif length(app.GaugeDirectorylist) > 1  && app.PlotalldatainonegraphButton.Value %multiple stations, use dropdown for selecting a set of color
%                     app.GaugeLineColor.Visible = "off";
%                     app.GaugeLineColor_multiple.Visible = "on";
%                     app.FlipCheckBox_3.Visible = "on";
%                 elseif length(app.GaugeDirectorylist) > 1  && app.PlotseparatelyButton_4.Value
%                     app.FlipCheckBox_3.Enable = "off";
%                     app.GaugeLineColor.Visible = "on";
%                     app.GaugeLineColor_multiple.Visible = "off";
%                 end
%             end

            % Activate/deactivate the options for the plot legend and the option to save the sta_ file as a .txt file.
            if app.PlotseparatelyButton_4.Value && length(app.GaugeDirectorylist) == 1
                app.legendfirsttext.Enable = "off";
                app.LegendSize.Enable = "off";
                app.LocationDropDown.Enable = "off";
                app.txtCheckBox.Enable = "on";
            elseif app.PlotseparatelyButton_4.Value && length(app.GaugeDirectorylist) > 1
                app.legendfirsttext.Enable = "on";
                app.LegendSize.Enable = "on";
                app.LocationDropDown.Enable = "on";
                app.txtCheckBox.Enable = "on";
            elseif app.PlotalldatainonegraphButton.Value
                app.legendfirsttext.Enable = "on";
                app.LegendSize.Enable = "on";
                app.LocationDropDown.Enable = "on";
                app.txtCheckBox.Enable = "off";
            end

        end

        % Value changed function: GaugeLineColor_multiple
        function GaugeLineColor_multipleValueChanged(app, event)

        end

        % Value changed function: AutoSetCheckBox
        function AutoSetCheckBoxValueChanged(app, event)
            if app.AutoSetCheckBox.Value
```

```
                app.Width.Enable = "off";
                app.Height.Enable = "off";
            else
                app.Width.Enable = "on";
                app.Height.Enable = "on";
            end
        end

        % Callback function
        function CloseFiguresButtonPushed(app, event)
            close all %figures
        end

        % Value changed function: mp4CheckBox
        function mp4CheckBoxValueChanged(app, event)
            if app.mp4CheckBox.Value
                app.FramerateEditField_2.Visible = "on";
                app.FramerateEditField_2Label.Visible="on";
            else
                app.FramerateEditField_2.Visible = "off";
                app.FramerateEditField_2Label.Visible="off";
            end
        end

        % Callback function
        function CloseFiguresButton_2Pushed(app, event)
            close all
        end

        % Selection changed function: ButtonGroup_11
        function ButtonGroup_11SelectionChanged(app, event)
            % Enable the tif and animation options in the 'Save Map' section when 'Plot all data in one figure' is selected
            if app.PlotseparatelyButton_2.Value
                app.mp4CheckBox.Enable = "on";
                app.tifCheckBox_2.Enable = "on";
                app.FramerateEditField_2.Enable = "on";
                app.FramerateEditField_2Label.Enable = "on";
            elseif app.PlotalldatainonefigureButton.Value
                app.mp4CheckBox.Enable = "off";
                app.tifCheckBox_2.Enable = "off";
                app.FramerateEditField_2.Enable = "off";
                app.FramerateEditField_2Label.Enable = "off";
            end
        end

        % Value changed function: AutoSetCheckBox_2
        function AutoSetCheckBox_2ValueChanged(app, event)
            %Disable the Width and Height textboxes
            if app.AutoSetCheckBox_2.Value
                app.Width_2.Enable = "off";
                app.Height_2.Enable = "off";
            else
                app.Width_2.Enable = "on";
                app.Height_2.Enable = "on";
            end

        end

        % Button pushed function: CloseFiguresButton_4
        function CloseFiguresButton_4Pushed2(app, event)
            close all
        end

        % Button pushed function: Button_19
        function Button_19Pushed(app, event)

            % Find U-vector files to import from a directory
            [files, path] = uigetfile('*.*', 'Select the files', 'MultiSelect', 'on');

            % Convert path and files to strings once
            pathStr = string(path);
            filesStr = string(files);

            if exist(pathStr, 'dir')
                cd(fullfile(pathStr));

                % Collect all the files
                allfiles = fullfile(pathStr + filesStr); % fullfile is used to handle folder names with spaces
                if ~(ismcc || isdeployed)
                    addpath(genpath(fullfile(pathStr)));
                end

                % Sort the files by snapshot timing, and display the list of filenames in the app's textbox
                sortedFiles = sort(filesStr);
                app.Uvectorfullfile = sort(allfiles);
                app.vectorpath = pathStr;
                app.Ufiles = sortedFiles;
                app.FileNAME5 = sortedFiles;
                app.FileTextArea_3.Value = sortedFiles;
                app.FileTextArea_3.FontColor = 'k';
                app.FileTextArea_3.BackgroundColor = 'w';
            end

            app.MinBarValue.Enable = "on";
            app.MaxBarValue.Enable = "on";

            ArrowSection = app.Panel_14.Children; % Get all the items in the Arrow section
            if contains(app.FileNAME5, "umax") % if the file is for plotting the maximum velocity, do not find the corresponding V-vector file
                % Disable the Arrow Section
                set(ArrowSection, 'Enable', 'off');
                app.PlotvectorsCheckBox.Value = false; % Uncheck the box

                % Change default value of the colormap and the bar limits
                app.BackgroundMapColorDropDown.Value = 'blue - purple';
                app.MaxBarValue.Value = 6;
                app.InterpolationDivisionEditField_2.Value = 4;
                app.InterpolationDivisionEditField_2.Enable = "on";
                app.InterpolationDivisionEditFieldLabel_2.Enable = "on";
                app.BackgroundMapColorDropDown.Enable = "on";
                app.FlipCheckBox_4.Enable = "on";
            else
                % Enable the Arrow Section
                set(ArrowSection, 'Enable', 'on');

                % Find the equivalent V Vector files, display the list of filenames in the app's textbox
                vfilenames = regexprep(sortedFiles, "u", "v", 'once'); % replaces only the first letter "u" in each of the filenames
                app.Vvectorfullfile = fullfile(pathStr + vfilenames);
                app.Vvectorfilenames = vfilenames;
                app.FileTextArea_4.Value = vfilenames;
                app.FileTextArea_4.FontColor = 'k';
                app.FileTextArea_4.BackgroundColor = 'w';
                app.BackgroundMapColorDropDown.Enable = "on";
                app.InterpolationDivisionEditField_2.Enable = "on";
                app.InterpolationDivisionEditFieldLabel_2.Enable = "on";
            end

            % Find corresponding 'eta' files when 'eta' is selected as the basemap option
            ETAfiles = regexprep(sortedFiles, "u", "eta", 'once'); % change 'u' to 'eta' in the filename
            app.ETAfullfile = fullfile(pathStr + ETAfiles);
            app.ETAfilenames = ETAfiles;

            % Find corresponding 'hmax' files when 'hmax' is selected as the basemap option
            HMAXfiles = regexprep(sortedFiles, "u", "hmax", 'once'); % change 'u' to 'hmax' in the filename
            app.HMAXfullfile = fullfile(pathStr + HMAXfiles);
            app.HMAXfilenames = HMAXfiles;

            % Read the first file to extract its matrix size, which will be used to set up the map boundary limit
            if exist(app.Uvectorfullfile{1}, 'file') == 2
                % Read the matrix from the file
                f = readmatrix(app.Uvectorfullfile{1});
                % Setup of the coordinates
                [row, col] = size(f);
                app.col1vel = col - 1;
                app.row1vel = row - 1;
            end

        end

        % Callback function
        function Button_20Pushed(app, event)

        end

        % Button pushed function: GENERATEButton_2
        function GENERATEButton_2Pushed(app, event)
            %Defining Parameters
            close all
            clear depthVM Uvector Vvector VMdata

            %------------------------
            %------ARROW COLOR------
            %------------------------
            quivercolorOptions = containers.Map({'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'}, {'k', '[0.4 0.4 0.4]', '[0.7 0.7 0.7]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'});
            quivercolor = quivercolorOptions(app.QuiverColorDropDown.Value);


            %----------------------------
            %-------LAND AREA COLOR------
            %----------------------------
            LandAreaColorOptions = containers.Map({'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'}, {'k', '[0.4 0.4 0.4]', '[0.7 0.7 0.7]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'});
            LandAreaColor2 = LandAreaColorOptions(app.LandColor.Value);


            %-----------------------------
            %-------BASEMAP COLORMAP------
            %-----------------------------
            %List of the  colormap items in the dropdown list
            keys = {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', ...
                'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', 'blue', 'blue - green', ...
                'blue - purple', 'green - blue', 'greens', 'gray', 'oranges', 'orange - red', 'purple - blue', ...
                'purple - blue - green', 'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', ...
                'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', 'brown - teal', ...
                'pink - light green', 'purple - green', 'purple - orange', 'red - blue', 'red - gray', ...
                'red - yellow - blue', 'red - yellow - green', 'spectral', 'accent', 'dark 2', 'paired', ...
                'pastel 1', 'pastel 2', 'set 1', 'set 2', 'set 3', '--- MATLAB default ----', '---  CBREWER 2 ---', ...
                '< sequential >', '< divergent >', '< qualitative >'};

            %The equivalent colormap code
            keysColor = {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', ...
                'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter','Blues','BuGn','BuPu','GnBu','Greens', ...
                'Greys','Oranges','OrRd','PuBu','PuBuGn', ...
'PuRd','Purples','RdPu','Reds','YlGn','YlGnBu','YlOrBr','YlOrRd','BrBG','PiYG','PRGn','PuOr','RdBu','RdGy', ...
                'RdYlBu','RdYlGn','Spectral','Accent', ...
'Dark2','Paired','Pastel1','Pastel2','Set1','Set2','Set3','parula','parula','parula','parula','parula'};

            %The colorbrewer
            keysBrewer = {'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', 'none', ...
                'none', 'none', 'none', 'none', 'none', 'none', 'none', 'seq',
'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', 'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', ...
                'seq', 'seq', 'seq', 'seq', 'seq', 'div', 'div', 'div', ...
                'div', 'div', 'div', 'div', 'div', ...
                'div', 'qual', 'qual', 'qual', ...
                'qual', 'qual', 'qual', 'qual', 'qual', ...
                'none', 'none', 'none','none', 'none'};

            % Match dropdown list items to their corresponding colors and ColorBrewer
            colorMap = containers.Map(keys, keysColor);
            colorBrewer = containers.Map(keys, keysBrewer);

            %Set up the colormap
            datacolor2  = colorMap(app.BackgroundMapColorDropDown.Value);
            datacolorbrewer2 = colorBrewer(app.BackgroundMapColorDropDown.Value);


            %-------------------------------
            %------BATHYMETRY CONTOURS------
            %-------------------------------
            %Line Colours
            ContourColor2Options = containers.Map({'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'}, {'k', '[0.4 0.4 0.4]', '[0.7 0.7 0.7]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'});
            ContourColor2 = ContourColor2Options(app.ColorDropDown_4.Value);

            %Line style
            ContourLineStyle2Options = containers.Map({'Solid', 'Dashed', 'Dotted', 'Dash-dotted'}, {'-', '--', ':', '-.'});
            ContourLineStyle2 = ContourLineStyle2Options(app.StyleDropDown_3.Value);


            %-------------------
            %------GAUGES------
            %-------------------
            %Marker Style
            GaugeMarkerOptions = containers.Map({'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'}, ...
                {'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'});
            GaugeMarker2 = GaugeMarkerOptions(app.gaugemarkerVelocityTab.Value);

            %Marker Color
            GaugeColor2Options = containers.Map({'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'}, {'k', '[0.4 0.4 0.4]', '[0.7 0.7 0.7]', '[0.8 0.8 0.8]', 'r', 'g', 'b', 'y', 'c', 'm', 'w'});
            GaugeColor2 = GaugeColor2Options(app.ColorDropDown_7.Value);

            %Label - horizontal alignment
            HorzLabel2Options = containers.Map({'Centre', 'Right', 'Left'},
{'center', 'left', 'right'});
            HorzLabel2 = HorzLabel2Options(app.CoastlinecolourDropDown_10.Value);

            %Label - vertical alignment
            VertLabel2Options = containers.Map({'Centre', 'Top', 'Bottom'},{'middle', 'bottom', 'top'});
            VertLabel2 = VertLabel2Options(app.CoastlinecolourDropDown_11.Value);


            %--------------------------------------
            %------SAVE MAP: Output Directory------
            %--------------------------------------
            %Check if the output directory has been manually set by the user
            if  app.OutputDirectoryEditField_3.Value == string(app.WORKFOLDER4)
                app.MainDirectoryVector= fullfile(app.WORKFOLDER3);
                app.FigureDirectoryVector = fullfile(app.WORKFOLDER4);
            else % Set the default directory to the Desktop
                if ismac %macOS
                    defaultDir = fullfile(getenv('HOME'), 'Desktop');
                elseif ispc %Windows
                    defaultDir = fullfile(getenv('USERPROFILE'), 'Desktop');
                else %Others
                    defaultDir = pwd;
                end

                %Create 'OUTPUT_FILES' folder and 'Figures' subfolder
                Dir0 = fullfile(defaultDir, 'OUTPUT_FILES');
                Dir1 = fullfile(defaultDir, 'OUTPUT_FILES/Figures');
                app.MainDirectoryVector= Dir0;
                app.FigureDirectoryVector = Dir1;
                if ~exist(Dir0, 'dir')
                    mkdir(Dir0);
                end
                if ~exist(Dir1, 'dir')
                    mkdir(Dir1);
                end

                %Display the path directory of the 'Figures' subfolder in the textbox
                app.OutputDirectoryEditField_3.Value = deblank(string(Dir1));
                app.OutputDirectoryEditField_3.FontColor = 'k';
            end



            %Create animation/ creating and opening the file
            if app.mp4CheckBox_2.Value
                vidObj = VideoWriter(fullfile(char(app.FigureDirectoryVector),'animation_vector.mp4'), 'MPEG-4');
                vidObj.FrameRate = app.FramerateEditField_2.Value;
                open(vidObj);
            end

            %-----------------------
            %------LOG REPORT------
            %-----------------------
            %Create 'Log_Files' subfolder in 'OUTPUT_FILES'
            LogFolder = fullfile(app.MainDirectoryVector, 'Log_Files');
            if ~exist(LogFolder, 'dir')
                mkdir(LogFolder);
            end
            %Create the text file
            filePath = fullfile(LogFolder, 'Log_Report_VelocityMap.txt');
            fileId = fopen(filePath ,'w');

            % Template for the default section of the log report (includes information from the 'Input Data' section)
            headerTitle = '******************** LOG REPORT ******************** ';
            timestamp = datestr(now, 'yyyy-mm-dd HH:MM:SS');
            endHeaderLine = '****************************************************';
            footerSeparator = '____________________________________________________';
            logContent = [
                sprintf('%s\n', headerTitle), ...
                sprintf('%-30s%-s\n', 'Map Type:', 'Velocity Map'), ...
                sprintf('%-30s%-s\n', 'Timestamp:', timestamp), ...
                sprintf('%s\n\n', endHeaderLine), ...
                sprintf('%-30s%-s\n', 'Southwest Corner:', sprintf('Long: %f Lat: %f', app.LongitudeEditField_2.Value, app.LatitudeEditField_2.Value)), ...
                sprintf('%-30s%-s\n', 'Grid Size:', sprintf('x: %d y: %d', app.gridX_2.Value, app.gridY_2.Value)), ...
                sprintf('%-30s%-s\n', 'Simulation Time Start:', sprintf('%d sec', app.StartTime2.Value)), ...
                sprintf('%-30s%-s\n', 'Simulation Time Interval:', sprintf('%d sec', app.TotalSimuilationTimesecEditField_4.Value)), ...
                sprintf('%s\n\n', footerSeparator)
                ];
            fprintf(fileId, '%s', logContent);


            %------LOG REPORT--------------------
            %------Input Data: Import Files------
            %-----------------------------------
            %Check loaded files in the U vector input form
            patterns = {'u_', 'umax_', 'umean_'};  %Check each loaded files if they contain "u"
            matches1 = false(size(app.FileNAME5));
            nonMatchingFiles = ~contains(string(app.FileNAME5), patterns);
            for i = 1:numel(patterns)
                pattern = patterns{i};
                matches1 = matches1 | contains(string(app.FileNAME5), pattern);
            end

            if any(nonMatchingFiles)
                fprintf(fileId, '%-30s%-s\n', 'U VECTOR:', 'Use filenames with ''u_'', ''umax_'', or ''umean_'' initials');
                fprintf(fileId, '%-30s%-s\n', ' ' , 'Incorrect files are loaded:');
                fprintf(fileId, '%-30s%-s\n', ' ', join(app.FileNAME5(nonMatchingFiles) , ', ')); % List all files that do not start with 'u' in the
```

```
filename
                fprintf(fileId, '%-30s%-s\n\n', ' ', 'No vectors are plotted');
            elseif all(contains(string(app.FileNAME5),  patterns)) %If all files are in U vector format
                %Check if V vector files exists in the same directory as the U vectors
                missingFiles = {};
                for i = 1:length(app.Vvectorfullfile) % Check each filename in the list
                    filePath = fullfile(app.Vvectorfullfile{i});
                    if exist(filePath, 'file') ~= 2
                        missingFiles = [missingFiles, app.Vvectorfilenames{i}]; %Make a list all corresponding V vector files that are not found
                    end
                end
                if ~isempty(missingFiles)
                    fprintf(fileId,'%-30s%-s\n', 'V VECTOR :','Missing Files. Ensure they are in the same folder as the U vector files:');
                    fprintf(fileId, '%-30s%-s\n\n', ' ', strjoin(missingFiles, ', '));
                end
            end

                %------LOG REPORT------------------------
                %------Input Data: Southwest Corner------
                %----------------------------------------
                %Check if the values have not changed and add an error note to the log report
                if app.LongitudeEditField_2.Value == 0 && app.LatitudeEditField_2.Value == 0
                    fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','Both latitude and longitude values are zeroes');
                elseif app.LongitudeEditField_2.Value == 0 && app.LatitudeEditField_2.Value ~= 0
                    fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','Longitude value is zero');
                elseif app.LatitudeEditField_2.Value ~= 0 && app.LatitudeEditField_2.Value == 0
                    fprintf(fileId, '%-30s%-s\n\n', 'SOUTHWEST CORNER:','Latitude value is zero');
                end

                %------LOG REPORT-----------------
                %------Input Data: Grid Size------
                %---------------------------------
                %Check if the values have not changed and add an error note to the log report
                if app.gridX_2.Value == 0 && app.gridY_2.Value == 0
                    fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','Both x and y values are zeroes');
                elseif app.gridX.Value == 0 && app.gridY.Value ~= 0
                    fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','X value is zero');
                elseif app.gridX.Value ~= 0 && app.gridY.Value == 0
                    fprintf(fileId, '%-30s%-s\n\n', 'GRID SIZE:','Y value is zero');
                end
```

```matlab

%------LOG REPORT-----------------------
%------Input Data: Load Bathymetry------
%---------------------------------------
%Check if the matrix size of the loaded bathymetry file matches the imported eta/hmax file
if matches(app.FileTextArea_3.Value,string(app.FileNAME5))  %Check if the imported V files match the names in the textbox
    if matches(app.DepthFileEditField_2.Value,string(app.bathymetryname2)) %Check if the imported bathymetry matches the name in the textbox
        [rowbath,colbath] = size(app.bathymetryname2);
        if   app.row1vel ~= rowbath
            fprintf(fileId, '%-30s%-s\n\n', 'MATRIX ROWS:','Rows in U-vector file and bathymetry do not match');
        end
        if app.col1vel ~= colbath
            fprintf(fileId, '%-30s%-s\n\n', 'MATRIX COLUMNS:','Columns in U-vector file and bathymetry do not match');
        end
    end
end

if app.DepthFileEditField_2.Value == "0" || ~matches(app.DepthFileEditField_2.Value,string(app.bathymetryname2))  % Add a log note that no bathymetry is loaded, so the features to edit land area and bathymetry contours are unavailable.
    fprintf(fileId, '%-30s%-s\n\n', 'BATHYMETRY:', 'Empty. Land color and bathymetry contours are not applicable.');
    app.DepthFileEditField_2.Value = "NO FILE";
    app.DepthFileEditField_2.FontColor = 'w';
    app.DepthFileEditField_2.BackgroundColor = 'r';
end

%Check if the bathymetry is has the correct file format
if app.bathymetrycheck == 1
    fprintf(fileId, '%-30s%-s\n\n', 'BATHYMETRY:', 'File must be .txt, .tif or mask_');
end

%------LOG REPORT-------------------------
%------Basemap and Overlays: Basemap------
%-----------------------------------------
if app.MinBarValue.Value > app.MaxBarValue.Value
    fprintf(fileId, '%-30s%-s\n', 'BASEMAP COLORBAR:','Max value must exceed min value');
    fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Max and min values are swapped');
end

%------LOG REPORT-------------------------------------
%------Basemap and Overlays: Bathymetry Contours------
%-----------------------------------------------------
if app.PlotBathymetryContoursCheckBox.Value
    if matches(app.DepthFileEditField_2.Value ,string(app.bathymetryname2))
        % Check if the minimum and maximum values are in the correct order
        if app.MinimumEditField_2.Value > app.
```

```
MaximumEditField_2.Value
                        temp = app.MaximumEditField_2.Value;
                        app.MaximumEditField_2.Value = app.MinimumEditField_2.Value;
                        app.MinimumEditField_2.Value = temp;
                        fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOURS:','Max value must exceed min value');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
                    end

                    %Check the interval value is within the limit
                    maxMinDifference = app.MaximumEditField_2.Value - app.MinimumEditField_2.Value;
                    if app.IntervalEditField_6.Value > maxMinDifference
                        app.IntervalEditField_6.Value = maxMinDifference / 2;
                        fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOUR INTERVAL:','Input must be less than max-min difference');
                        fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to half the max-min difference');
                    end

                    if app.AddLabelCheckBox_2.Value %Add labels
                        %Check if the label interval is divisible by the value provided in the 'Range' section
                        if mod(app.IntervalEditField_8.Value, app.IntervalEditField_6.Value) ~= 0
                            app.IntervalEditField_8.Value = app.IntervalEditField_6.Value;
                            fprintf(fileId, '%-30s%-s\n', 'BATHYMETRY CONTOUR LABEL:','Value must be divisible by the Interval value in the Range section');
                            fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Value adjusted to be the same as the Interval Value in the Range section');
                        end
                    end
                end
            end

            %------LOG REPORT------------------------
            %------Basemap and Overlays: Gauges------
            %----------------------------------------
            if app.PlotGaugesCheckBox.Value && ~matches(app.FileEditField_2.Value ,string(app.GAUGEFILE3))
                fprintf(fileId, '%-30s%-s\n\n', 'GAUGES:', 'Empty. No virtual gauges plotted');
                app.FileEditField_2.Value = "NO FILE";
                app.FileEditField_2.FontColor = 'w';
                app.FileEditField_2.BackgroundColor = 'r';
            end

            %------LOG REPORT---------------------------
            %------General Layout: Boundary Limits------
            %-------------------------------------------
            if app.WestEditField_2.Value > app.EastEditField_2.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:','East boundary must be greater than West boundary');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
            elseif app.EastEditField_2.Value == app.WestEditField_2.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:', 'East and West values are the same');
                fprintf(fileId, '%-30s%-s\n\n', 'Resolved:', 'The default values are used');
            end

            if app.SouthEditField_2.Value > app.NorthEditField_2.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:','North boundary must be greater than South boundary');
                fprintf(fileId, '%-30s%-s\n\n', ' ','Resolved: Values are swapped');
            elseif  app.NorthEditField_2.Value == app.SouthEditField_2.Value
                fprintf(fileId, '%-30s%-s\n', 'BOUNDARY LIMIT:', 'North and South values are the same');
                fprintf(fileId, '%-30s%-s\n\n', 'Resolved:', 'The default values are used');
            end

            fclose(fileId); %Close the log report file

            %% PROCESS THE INPUT FILES AND VALUES, THEN GENERATE THE FIGURES
            %--------------------------------
            %------READ THE INPUT FILES------
            %--------------------------------
            % Set up the bathymetry data
            bathymetryFileExists = exist(app.bathymetryinputdata2, 'file') == 2;
            if bathymetryFileExists
                depthVM = app.bathymetrydata2;
                if app.HorizontalCheckBox_2.Value
                    depthVM = fliplr(depthVM);
                end
                if app.VerticalCheckBox_2.Value
                    depthVM = flipud(depthVM);
                end
            end

            FileLength1 = length(app.Uvectorfullfile);
            patterns = {'u_', 'umax_', 'umean_'};

            %---------------------------------------
            %------PLOT ALL DATA IN ONE FIGURE------
            %---------------------------------------
            if app.PlotalldatainonefigureButton_2.Value

                %Create the figure
                f = figure(1);
                f.Visible = 'off';

                %Formatting the subplots
                if FileLength1 == 1
                    plotCol = 1;
                    plotRow = 1;
                elseif FileLength1 > 1 && FileLength1 <= 10
                    plotCol = 2;
                    plotRow = ceil(FileLength1/2);
                elseif FileLength1 > 10
                    plotCol = 3;
                    plotRow = ceil(FileLength1/3);
                end

                %----------------------------
                %------PLOT SEPARATELY------
                %----------------------------
            elseif app.PlotseparatelyButton_3.Value
                f = figure();
                f.Visible = 'off';
            end

            %---------------------------------
            %------PRE-SET MAP OPTIONS-------
            %---------------------------------
            %Allocate memory for the vectors
            Uvector = NaN(app.row1vel, app.col1vel);
            Vvector = NaN(app.row1vel, app.col1vel);
            colorbarHandles = [];

            %------Basemap Colormap-------
            if strcmp(datacolorbrewer2,'none') %Using default matlab colormap options
                datacolor3 = evalin('base',datacolor2);
            else %Using cbrewer2
                datacolor3 = cbrewer2(datacolorbrewer2,datacolor2,app.InterpolationDivisionEditField_2.Value,'pchip');
            end

            %------Basemap Colorbar-------
            colormapMin = app.MinBarValue.Value;
            colormapMax = app.MaxBarValue.Value;
            %Options for label
            if app.BathymetryButton.Value
                colorbartitle = 'Bathymetry(m)';
            elseif app.etaButton.Value
                colorbartitle = '\eta (m)';
            elseif app.hmaxButton.Value
                colorbartitle = 'Maximum \eta (m)';
            elseif app.VelocityButton.Value
                if contains(string(app.Ufiles(i)),'umax')
                    colorbartitle = 'Maximum Velocity (m/s)';
                else
                    colorbartitle = 'Velocity (m/s)';
                end
            elseif app.VorticityButton.Value
                colorbartitle = 'Vorticity (s^{-1})';
            end

            %Check if the minimum and maximum values are in the correct order; if not, swap them
            if app.BathymetryButton.Value
                if -app.MaxBarValue.Value > -app.MinBarValue.Value %Min and Max values of the colorbar limit are swapped
```

```
                        %Swap the values
                        temp = app.MaxBarValue.Value;
                        app.MaxBarValue.Value = app.MinBarValue.Value;
                        app.MinBarValue.Value = temp;
                    end
                else
                    if app.MaxBarValue.Value < app.MinBarValue.Value %Min and Max values of the colorbar limit are swapped
                        %Swap the values
                        temp = app.MaxBarValue.Value;
                        app.MaxBarValue.Value = app.MinBarValue.Value;
                        app.MinBarValue.Value = temp;
                    end
                end

                %------Boundary limit-------
                eastValue = app.EastEditField_2.Value;
                westValue = app.WestEditField_2.Value;

                if eastValue < westValue
                    % Swap West and East input values if the West value is greater than the East
                    xMin_VM = eastValue;
                    xMax_VM = westValue;
                    app.EastEditField_2.Value = xMax_VM;
                    app.WestEditField_2.Value = xMin_VM;
                elseif eastValue == westValue
                    % Set the limits to the default values
                    app.EastEditField_2.Value = max(app.x);
                    app.WestEditField_2.Value = min(app.x);
                    xMin_VM = app.WestEditField_2.Value;
                    xMax_VM = app.EastEditField_2.Value;
                else
                    xMin_VM = westValue;
                    xMax_VM = eastValue;
                end

                northValue = app.NorthEditField_2.Value;
                southValue = app.SouthEditField_2.Value;

                if northValue < southValue
                    % Swap North and South input values if the South value is greater than the North
                    yMin_VM = northValue;
                    yMax_VM = southValue;
                    app.NorthEditField_2.Value = yMax_VM;
                    app.SouthEditField_2.Value = yMin_VM;
                elseif northValue == southValue
                    % Set the limits to the default values
                    app.NorthEditField_2.Value = max(app.y);
                    app.SouthEditField_2.Value = min(app.y);
                else
                    yMin_VM = southValue;
                    yMax_VM = northValue;
                end

            %------Map Title-------
            totalSimulationTime = app.TotalSimuilationTimesecEditField_4.
```

```
Value;
            startTime = app.StartTime2.Value;
            colorbarTextSize = app.maplabelsize.Value + 1;
            pat = digitsPattern;


                %-------------------------------------------
                %------Start of the loop for plotting-------
                %-------------------------------------------
                for i = 1:FileLength1
                    if all(contains(string(app.FileTextArea_3.Value), patterns)) % Check if each file is a U vector
                        % Load the U vector files
                        if exist(app.Uvectorfullfile{i}, 'file') == 2
                            Uvector = readmatrix(app.Uvectorfullfile{i});

%                             %Remove values on land (negative values)
%                             if bathymetryFileExists
%                                 Uvector(depthVM <= 0) = NaN; % Set land values to NaN or another appropriate value
%                             end
                        end


                        % Load the V vector files
                        if ~contains(char(app.Ufiles{i}), 'umax') && exist(app.Vvectorfullfile{i}, 'file') == 2
                            Vvector = readmatrix(app.Vvectorfullfile{i});

                            %------Flip the Matrix-------
%                             if app.HorizontalCheckBox_2.Value
%                                 % Flip horizontally
%                                 Vvector = fliplr(Vvector);
%                             end
%                             if  app.VerticalCheckBox_2.Value
%                                 Vvector = flipud(Vvector);
%                             end

%                             % Remove values on land if the bathymetry file exists
%                             if bathymetryFileExists
%                                 Vvector(depthVM <= 0) = NaN; % Set land values to NaN or another appropriate value
%                             end

                        end

                        %----------------------------------------
                        %------PLOT ALL DATA IN ONE FIGURE------
                        %----------------------------------------
                        if app.PlotalldatainonefigureButton_2.Value
                            % Create a subplot in the specified position
                            gca(i) = subplot(plotRow, plotCol, i);
                            f.Visible = 'on';

                            %---------------------------
                            %------PLOT SEPARATELY------
                            %---------------------------
                        elseif app.PlotseparatelyButton_3.Value
                            f = figure();
                            f.Visible = 'off';
                            gca(i) = subplot(1,1,1);
                        end

                        %-----------------------------------------
                        %------Basemap and Overlays: Basemap------
                        %-----------------------------------------
                        %Check which option is selected for the basemap
                        clear VMdata
                        %Option 1: W Velocity
                        if app.VelocityButton.Value
                            if ~contains(char(app.Ufiles(i)),'umax') && exist(app.Vvectorfullfile(i),'file')
                                VMdata = sqrt(Uvector.^2 + Vvector.^2); %Calculate the Z magnitude based on the magnitudes of the U and V vectors
                            elseif contains(char(app.Ufiles(i)),'umax')
                                VMdata = Uvector;
                            end

                            %Option 2: eta / sea surface displacement at time t
                        elseif  app.etaButton.Value
                            VMdata = readmatrix(app.ETAfullfile(i));

                            %Option 3: hmax / maximum wave height
                        elseif app.hmaxButton.Value
                            if exist(app.HMAXfullfile(i),"file")
                                VMdata = readmatrix(app.HMAXfullfile(i));
                            end

                            %Option 4: Bathymetry
                        elseif app.BathymetryButton.Value
                            if matches(app.DepthFileEditField_2.Value,string(app.bathymetryname2))
                                VMdata = -depthVM;
                            end

                            %Option 5: Vorticity
                        elseif app.VorticityButton.Value
                            [VMdata, ~]=curl(app.xcoord,app.ycoord,Uvector, Vvector); %from simple_cases/rip_2d
                        end


                        %------Flip the Matrix-------
                        % Flip horizontally
                        if app.HorizontalCheckBox_2.Value && ~app.BathymetryButton.Value
                                VMdata = fliplr(VMdata);
                        end

                        % Flip vertically
                        if app.VerticalCheckBox_2.Value && ~app.BathymetryButton.Value
                            VMdata = flipud(VMdata);
                        end
```

```matlab

                        %Remove land values from the selected basemap
                        try
                            if isprop(app, 'bathymetryinputdata2')
                                VMdata(depthVM<=0) = NaN;
                            end
                        end

                        %----------------------
                        %------MAIN PLOT------
                        %----------------------
                        p = pcolor(app.xcoord,app.ycoord,VMdata);
                        shading interp
                        set(p, 'EdgeColor', 'none','FaceColor', 'interp');
                        set(gca,'Color', LandAreaColor2) %Set the background color for the land, %Adjust tick size label
                        hold on


                        %---------------------------------------------------
                        %------Basemap and Overlays: Basemap Colormap-------
                        %---------------------------------------------------
                        %Flip the colorbar
                        if app.FlipCheckBox_4.Value
                            colormap(gca(i), flipud(datacolor3));
                        else
                            colormap(gca(i), datacolor3);
                        end

                        %---------------------------------------------------
                        %------Basemap and Overlays: Basemap Colorbar-------
                        %---------------------------------------------------
                        hold on
                        %Check if the minimum and maximum values are in the correct order; if not, swap them
                        if app.BathymetryButton.Value
                            caxis(gca(i), [-colormapMax,-colormapMin]); % Colorbar min and max values
                        else
                            caxis(gca(i), [colormapMin,colormapMax]); % Colorbar min and max values
                        end

                        cb = colorbar;
                        colorbarHandles = [colorbarHandles, cb];

                        if cb.FontSize ~= colorbarTextSize
                            cb.FontSize = colorbarTextSize; % size of the tick numbers
                        end
                        cb.Label.String = colorbartitle; % Set the colorbar title

                        %---------------------------------------------------
                        %------Basemap and Overlays: Bathymetry Contours----
                        %---------------------------------------------------
                        if app.PlotBathymetryContoursCheckBox.Value && matches(app.DepthFileEditField_2.Value,string(app.bathymetryname2))
```

```
    hold on
    %Plot the contours
    depth0 = depthVM;
    kernel = ones(6) / 36; % Use averaging kernel to↙
make the contour smoother
    depthfiltered = filter2(kernel, depth0);
    [C1,h1] = contour(app.xcoord,app.ycoord, ↙
depthfiltered,app.MinimumEditField_2.Value:app.IntervalEditField_6.Value:app. ↙
MaximumEditField_2.Value,'EdgeColor',ContourColor2,'LineStyle',↙
ContourLineStyle2,'LineWidth',app.WidthEditField_2.Value);
    %Add contour labels
    if app.AddLabelCheckBox_2.Value
        hold on
        manualLabels3 = app.MinimumEditField_2.Value:↙
app.IntervalEditField_6.Value:app.MaximumEditField_2.Value;
        clabel(C1, h1, manualLabels3, 'FontSize', app.↙
LabelSizeCont.Value, 'Color', 'k');
        set(handle(h1), 'LabelSpacing', app.↙
SpacingEditField.Value);% Adjust the label spacing
    end
end


%---------------------------------------
%------Basemap and Overlays: Gauges----
%---------------------------------------
hold on
if app.PlotGaugesCheckBox.Value && matches(app.↙
FileEditField_2.Value,string(app.GAUGEFILE3))
    %Plot the points
    gaugepoint = plot(app.longGauge2,app.latGauge2, ↙
GaugeMarker2);
    set(gaugepoint, 'MarkerSize', app.SizeEditField_2.↙
Value,'MarkerFaceColor', GaugeColor2,'MarkerEdgeColor', GaugeColor2); %Size and↙
color of the points
    %Add labels
    if app.AddLabelCheckBox.Value
        hold on
        for k = 1:length(app.latGauge2)
            gtxt = text(app.longGauge2(k)+ app.↙
GLspacing_2.Value,app.latGauge2(k,1),num2str(k),'FontSize',app.FontSizeEditField_2.↙
Value, 'HorizontalAlignment',HorzLabel2,...
                'VerticalAlignment',VertLabel2); %↙
GLspacing_2.Value is added to longitude so there's space between the point and the↙
label
            set(gtxt, 'Clipping', 'on');
        end
    end
end

%---------------------------------
%------Arrows: Plot Vector-------
%---------------------------------
if app.PlotvectorsCheckBox.Value
    if any(~contains(app.FileNAME5, "umax") | app.↙
VorticityButton.Value == false)
        %------Flip the Matrix-------
        % Flip horizontally
        if app.HorizontalCheckBox_2.Value
            UvectorArrow =  fliplr(Uvector);
            VvectorArrow =  fliplr(Vvector);
        else
            UvectorArrow =  Uvector;
            VvectorArrow =  Vvector;
        end

        % Flip vertically
        if app.VerticalCheckBox_2.Value
            UvectorArrow =  flipud(Uvector);
            VvectorArrow =  flipud(Vvector);
        else
            UvectorArrow =  Uvector;
            VvectorArrow =  Vvector;
        end

        q1 = quiver(gca(i),app.xcoord(1:app.ArrowSpacing.Value:end,1:app.ArrowSpacing.Value:end),app.ycoord(1:app.ArrowSpacing.Value:end,1:app.ArrowSpacing.Value:end),UvectorArrow(1:app.ArrowSpacing.Value:end,1:app.ArrowSpacing.Value:end),VvectorArrow(1:app.ArrowSpacing.Value:end,1:app.ArrowSpacing.Value:end),app.arrowscale.Value);
        q1.Color = quivercolor;
        q1.LineWidth = app.ArrowThickness.Value;
        q1.MaxHeadSize = app.ArrowHeadSize.Value;
        axis equal  %use equal unit lengths
    end
end

%------------------------------------------
%------GENERAL LAYOUT: Boundary limit-------
%------------------------------------------
xlim([xMin_VM,xMax_VM])
ylim([yMin_VM,yMax_VM])
hold on
f.Visible = 'on';

%-----------------------------------
%------GENERAL LAYOUT: Ticks-------
%-----------------------------------
%Reduce the number of ticks to 2 on the X axis and 2 on the Y axis for simplicity
% Get the current ticks from the first plot
if i == 1
    xAllTicks = xticks(gca);
    yAllTicks = yticks(gca);

    % Ensure the ticks are numeric arrays
    if iscell(xAllTicks)
        xAllTicks = cellfun(@str2double, xAllTicks, 'UniformOutput', true);
    end
    if iscell(yAllTicks)
        yAllTicks = cellfun(@str2double, yAllTicks, 'UniformOutput', true);
    end

    % Remove any NaN values that might result from
```

```
str2double conversion
                            xAllTicks = xAllTicks(~isnan(xAllTicks));
                            yAllTicks = yAllTicks(~isnan(yAllTicks));

                            % Select only the 2nd and second-to-last ticks
                            % For X axis
                            if length(xAllTicks) >= 4  % Check first if there are enough ticks to select from
                                xticksSelected = sort([xAllTicks(2), xAllTicks(end-1)]);
                            elseif length(xAllTicks) >= 2
                                xticksSelected = sort([xAllTicks(1), xAllTicks(end)]); % Use the first and last if less than 4
                            else
                                xticksSelected = xAllTicks; % Use whatever ticks are available
                            end

                            % For Y axis
                            if length(yAllTicks) >= 4
                                yticksSelected = sort([yAllTicks(2), yAllTicks(end-1)]);
                            elseif length(yAllTicks) >= 2
                                yticksSelected = sort([yAllTicks(1), yAllTicks(end)]); % Use the first and last if less than 4
                            else
                                yticksSelected = yAllTicks; % Use whatever ticks are available
                            end

                        end

                        % Update the number of ticks on the axes
                        xticks(gca, xticksSelected);
                        yticks(gca, yticksSelected);

                        % Determine the maximum decimal places for x and y ticks
                        xmaxDecimals = max(cellfun(@(x) length(regexp(x, '(?<=\.)\d+', 'match', 'once')), ...
                            cellstr(num2str(xticksSelected'))));
                        ymaxDecimals = max(cellfun(@(y) length(regexp(y, '(?<=\.)\d+', 'match', 'once')), ...
                            cellstr(num2str(yticksSelected'))));

                        % Convert tick labels to their equivalent in degrees
                        if app.PlotindegreesCheckBox.Value
                            % Convert selected ticks to formatted strings and set them for degrees
                            xticklabels(gca, arrayfun(@(x) sprintf('%.1f°', mod(x + 360, 360)), xticksSelected, 'UniformOutput', false));
                            yticklabels(gca, arrayfun(@(y) sprintf('%.1f°', y), yticksSelected, 'UniformOutput', false));
                        else
                            % Correct the format string and apply calculated maximum decimals
                            xticklabels(gca, arrayfun(@(x) sprintf(['%.' num2str(xmaxDecimals) 'f'], x), xticksSelected, 'UniformOutput', false));
                            yticklabels(gca, arrayfun(@(y) sprintf(['%.' num2str(ymaxDecimals) 'f'], y), yticksSelected, 'UniformOutput', false));
                        end

                        % Set the y-axis tick angle
                        ytickangle(gca, -270);

                        %Fontsize of the tick labels
                        set(gca,'Layer', 'top','LineWidth',1,'FontSize',app.ColorbarTextSize.Value-1)

                        %Adjust tick lengths based on the number of subplots if 'Plot all data in one figure' is selected
                        if FileLength1 <= 2 && app.PlotalldatainonefigureButton_2.Value
                            set(gca,'TickLength',[0.015 0.015])
                        else
                            set(gca,'TickLength',[0.01 0.01])
                        end

                        grid off
                        box on
                        daspect([1 1 1])

                        %--------------------------------------
                        %------GENERAL LAYOUT: Map Title------
                        %--------------------------------------
                        str = string(app.FileNAME5(i));

                        dbl1 = extract(str,pat); % Extract numbers from the filename to calculate the time frame

                        if startTime == 0 && str2double(dbl1) == 0 %Set the time to zero
                              timeT = startTime + (str2double(dbl1)-1);
                        else
                            OriginaltimeT2 = totalSimulationTime * (str2double(dbl1)-1); % Convert the filenumber to time
                            timeT = startTime + OriginaltimeT2; % Adjust the start of the time count
                        end


                        % Time conversion
                        if timeT < 60
                            timeStr = sprintf("%d sec", timeT);
                        elseif timeT < 3600
                            minutes = round(timeT / 60);
                            if minutes == 1
                                timeStr = "1 min";
                            else
                                timeStr = sprintf("%d mins", minutes);
                            end
                        else
                            raw_hours = timeT/3600;
                            hours = fix(timeT/3600); %Extract the whole number%Extract the whole number
                            hours decimal = raw hours - floor(raw hours); %
```

```
Extract the decimal places
                            minutes = round(hours_decimal *  60);

                            if hours > 0
                                if minutes > 0
                                    timeStr = sprintf("%d hr, %d mins", hours,↙
minutes);
                                else
                                    timeStr = sprintf("%d hr", hours);
                                end
                            end
                        end

                        title(timeStr, 'FontSize',colorbarTextSize); %Add the↙
title to the top of the map; Adjust the font size of the title
                        hold on


                        %-----------------------------------------
                        %------GENERAL LAYOUT: Figure Size-------
                        %-----------------------------------------
                        %Figure Size
                        if ~app.AutoSetCheckBox_3.Value
                            figureHandle = ancestor(p, 'figure'); % Get the↙
figure handle containing the axes
                            set(figureHandle, 'Units', 'Inches', 'Position',↙
[0, 0, app.Width_3.Value, app.Height_3.Value], 'PaperUnits', 'Inches', 'PaperSize',↙
[app.Width_3.Value, app.Height_3.Value]);
%                             set(gcf, 'Units', 'Inches', 'Position', [0, 0,↙
app.Width_3.Value, app.Height_3.Value], 'PaperUnits', 'Inches', 'PaperSize', [app.↙
Width_3.Value, app.Height_3.Value]);
                        end

                        %Create animation / closing the file
                        if app.mp4CheckBox_2.Value
                            set(gcf,'Renderer','zbuffer');
                            f = getframe(gcf);
                            writeVideo(vidObj,f);
                        end

                        %---------------------------------------
                        %------SAVE MAP: Plot Separately-------
                        %---------------------------------------
                        if app.PlotseparatelyButton_3.Value
                            %Save the maps with filenames matching the loaded↙
file
                            OF  = fullfile(app.FigureDirectoryVector,app.↙
Ufiles(i));


                            % Set the figure size for exporting
                            if ~app.AutoSetCheckBox_3.Value
                                set(figureHandle, 'Units', 'Inches',↙
'Position', [0, 0, app.Width_3.Value, app.Height_3.Value], ...
                                    'PaperUnits', 'Inches', 'PaperSize', [app.↙
Width_3.Value, app.Height_3.Value]);
                                set(gcf, 'Units', 'Inches', 'Position', [0, 0,↙
app.Width_3.Value, app.Height_3.Value], ...
```

```
                                    'PaperUnits', 'Inches', 'PaperSize', [app.Width_3.Value, app.Height_3.Value]);
                            end

                            %Options for saving format
                            %PNG
                            if app.pngCheckBox_3.Value
                                outputfile = OF + ".png";
                                exportgraphics(gcf,outputfile,'Resolution',300)

                            end

                            %JPG
                            if app.jpgCheckBox_3.Value
                                outputfile = OF + ".jpg";
                                exportgraphics(gcf,outputfile,'Resolution',300)

                            end

                            %PDF
%                             if app.pdfCheckBox_3.Value
%                                 outputfile = OF + ".pdf";
%                                 exportgraphics(gcf, outputfile,'ContentType','vector','Resolution',300)
%                             end

                            %TXT - to save the raw vector files
                            if app.txtCheckBox_2.Value
                                outputfile1 = OF + "_U.txt";
                                outputfile2 = OF + "_V.txt";
                                writematrix(Uvector,outputfile1);
                                dlmwrite(outputfile1,Uvector,'delimiter','\t');

                                writematrix(Vvector,outputfile2);
                                dlmwrite(outputfile2,Vvector,'delimiter','\t');
                            end

                            %TIF
                            if app.tifCheckBox.Value
                                XXX = app.xcoord;
                                YYY = app.ycoord;

                                west_limit = app.WestEditField_2.Value;
                                east_limit = app.EastEditField_2.Value;
                                north_limit = app.NorthEditField_2.Value;
                                south_limit = app.SouthEditField_2.Value;

                                %Make the longitude values within -180 to 180 range
                                if west_limit ~= 180
                                    west_limit = mod(west_limit + 180, 360) - 180;
                                end

                                if east_limit ~= 180
                                    east_limit = mod(east_limit + 180, 360) - 180;
```

```
                        end

                        % Convert x coordinates to correct degree range
                        XXX(XXX > 180) = XXX(XXX > 180) - 360;

                        % Create meshgrid
                        [X, Y] = meshgrid(XXX, YYY);
                        Z = VMdata;

                        %Check if the latitude is within the Geographic limits
                        if app.LatitudeEditField_2.Value > 90 || app.LatitudeEditField_2.Value < -90
                            app.LatitudeEditField_2.Value = 0;
                        else
                            %Clip the matrix based on boundary limits
                            if west_limit >= 0 && east_limit >=0  || west_limit < 0 && east_limit < 0 % if east and west values are either both (+) or both (-) values
                                x_indices_positive =  XXX >= west_limit & XXX <= east_limit;
                                y_indices = YYY >= south_limit & YYY <= north_limit;
                                Z_positive = Z(y_indices, x_indices_positive);
                                X_positive = X(y_indices, x_indices_positive);
                                Y = Y(y_indices, x_indices_positive); % Keep the same for both file
                                if ~isempty(Z_positive)
                                    % Define spatial referencing information
                                    R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
                                    R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                                    R_positive.RowsStartFrom = 'west';

                                    % Save as geotiff
                                    filename_positive = OF + ".tif";
                                    geotiffwrite(filename_positive, Z_positive, R_positive);
                                end

                            elseif west_limit >= 0 && east_limit < 0 %if longitude values are a combination of (+) west limit and (-) east limit
                                x_indices_positive = XXX >= 0 & XXX >= west_limit;
                                x_indices_negative = XXX < 0 & XXX <= east_limit;
                                y_indices = YYY >= south_limit & YYY <= north_limit;
                                Z_positive = Z(y_indices, x_indices_positive);
                                Z_negative = Z(y_indices, x_indices_negative);
```

```
            X_positive = X(y_indices, x_indices_positive);
            X_negative = X(y_indices, x_indices_negative);
            Y = Y(y_indices, x_indices_positive); % Keep the same for both file

            if ~isempty(Z_positive)
                % Define spatial referencing information
                R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
                R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                R_positive.RowsStartFrom = 'west';

                % Save as geotiff
                filename_positive = OF + "_1.tif";
                geotiffwrite(filename_positive, Z_positive, R_positive);
            end

            if ~isempty(Z_negative)
                % Define spatial referencing information
                R_negative = georasterref('RasterSize', size(Z_negative), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_negative(:)), max(X_negative(:))]);
                R_negative.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                R_negative.RowsStartFrom = 'west';

                % Save as geotiff
                filename_negative = OF + "_2.tif";
                geotiffwrite(filename_negative, Z_negative, R_negative)
            end

        elseif west_limit < 0 && east_limit >= 0 %if longitude values are a combination of (-) west limit and (+) east limit
            x_indices_positive = XXX >= 0 & XXX <= east_limit;
            x_indices_negative = XXX < 0 & XXX >= west_limit;
            y_indices = YYY >= south_limit & YYY <= north_limit;
            Z_positive = Z(y_indices, x_indices_positive);
            Z_negative = Z(y_indices, x_indices_negative);
            X_positive = X(y_indices, x_indices_positive);
            X_negative = X(y_indices, x_indices_negative);
            Y = Y(y_indices, x_indices_positive); % Keep the same for both file

```

```matlab
                                        if ~isempty(Z_positive)
                                            % Define spatial referencing information
                                            R_positive = georasterref('RasterSize', size(Z_positive), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_positive(:)), max(X_positive(:))]);
                                            R_positive.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                                            R_positive.RowsStartFrom =  'west';

                                            % Save as geotiff
                                            filename_positive = OF + "_1.tif";
                                            geotiffwrite(filename_positive, Z_positive, R_positive);
                                        end
                                        if ~isempty(Z_negative)
                                            % Define spatial referencing information
                                            R_negative = georasterref('RasterSize', size(Z_negative), 'LatitudeLimits', [min(Y(:)), max(Y(:))], 'LongitudeLimits', [min(X_negative(:)), max(X_negative(:))]);
                                            R_negative.ColumnsStartFrom = 'south'; % Set the column orientation to start from the north
                                            R_negative.RowsStartFrom =  'west';

                                            % Save as geotiff
                                            filename_negative = OF + "_2.tif";
                                            geotiffwrite(filename_negative, Z_negative, R_negative);
                                        end
                                    end
                                end
                            end
                        end
                    end
                end

                %Change colorbar size
                        if cb.Label.FontSize ~= colorbarTextSize
                            cb.Label.FontSize =colorbarTextSize;
                        end

                %--------------------------------------------------
                %------SAVE MAP: Plot all data in one figure-------
                %--------------------------------------------------
                if app.PlotalldatainonefigureButton_2.Value

                    % Set the figure size for exporting
                    if ~app.AutoSetCheckBox_3.Value
                        set(gcf, 'Units', 'Inches', 'Position', [0, 0, app.Width_3.Value, app.Height_3.Value], ...
                            'PaperUnits', 'Inches', 'PaperSize', [app.Width_3.Value, app.Height_3.Value]);
                    end

                    if length(app.Ufiles) == 1
                        OF  = fullfile(app.FigureDirectoryVector,app.Ufiles);
                    else
```

```
                        OF  = fullfile(app.FigureDirectoryVector, "Vector_Output");
                    end

                    %PNG
                    if app.pngCheckBox_3.Value
                        outputfile = OF + ".png";
                        exportgraphics(gcf,outputfile,'Resolution',300)
                   end

                    %JPG
                    if app.jpgCheckBox_3.Value
                        outputfile = OF + ".jpg";
                        exportgraphics(gcf,outputfile,'Resolution',300)
                    end

                    %PDF
%                     if app.pdfCheckBox_3.Value
%                         outputfile = OF + ".pdf";
%                         exportgraphics(gcf, outputfile,'ContentType','vector','Resolution',300)
%                     end
                end

        end

        % Value changed function: QuiverColorDropDown
        function QuiverColorDropDownValueChanged(app, event)

        end

        % Button pushed function: CloseFiguresButton_3
        function CloseFiguresButton_3Pushed(app, event)
            close all
        end

        % Button pushed function: Button_22
        function Button_22Pushed(app, event)
            clear  app.bathymetryinputdata2 app.bathymetrydata2

            %Load the bathymetry file and display its filename in the textbox
            [bathymetryfile,path2] = uigetfile('*'); %*.txt;*.out;*.tif'
            if ~(ismcc || isdeployed)
                addpath(genpath(fullfile(string(path2))));
            end
            app.DepthFileEditField_2.Value = string(bathymetryfile);
            app.bathymetryinputdata2 = fullfile(string(path2), string(bathymetryfile));
            app.bathymetryname2 = string(bathymetryfile);
            app.DepthFileEditField_2.BackgroundColor = 'w';
            app.DepthFileEditField_2.FontColor =  'k';


            if app.DepthFileEditField_2.Value == string(bathymetryfile)
                if ~strcmp(app.DepthFileEditField_2.Value, "0") && ~strcmp(app.DepthFileEditField_2.Value, "00")
                    try
                        if contains(app.bathymetryinputdata2,'.tif')
```

```
                                [A1,R2] = readgeoraster(app.bathymetryinputdata2);
                                app.bathymetrydata2 =  flipud(A1);

                                %Extract lat and long of the southwest corner
                                app.LongitudeEditField_2.Value = min(R2.LongitudeLimits);
                                app.LatitudeEditField_2.Value = min(R2.LatitudeLimits);
                                app.LongitudeEditField_2.FontColor = 'k';
                                app.LatitudeEditField_2.FontColor = 'k';

                                %Extract grid size
                                app.gridX_2.Value = R2.CellExtentInLatitude;
                                app.gridY_2.Value = R2.CellExtentInLongitude;
                                app.gridX_2.FontColor = 'k';
                                app.gridY_2.FontColor = 'k';

                                %Update the X-axis limits in the 'General Layout' section
                                app.xcoord = [0:app.col1vel] * app.gridX_2.Value+app.LongitudeEditField_2.Value;
                                app.EastEditField_2.Value = max(app.xcoord);
                                app.WestEditField_2.Value = min(app.xcoord);

                                %Update the Y-axis limits in the 'General Layout' section
                                app.ycoord = [0:app.row1vel] * app.gridY_2.Value+app.LatitudeEditField_2.Value;
                                app.NorthEditField_2.Value = max(app.ycoord);
                                app.SouthEditField_2.Value = min(app.ycoord);

                                app.EastEditField_2.FontColor = 'k';
                                app.WestEditField_2.FontColor = 'k';
                                app.NorthEditField_2.FontColor = 'k';
                                app.SouthEditField_2.FontColor = 'k';
                                drawnow

                                if app.BathymetryButton.Value %If bathymetry is selected as the basemap
                                    app.FileTextArea_5.Value = app.DepthFileEditField_2.Value; %show the bathymetry filename in the textbox in 'Basemap and Overlays' section
                                else
                                    app.FileTextArea_5.Value = '';
                                end
                                drawnow

                            else %Only read the matrix for use as a basemap and for plotting bathymetry contours
                                app.bathymetrydata2 = readmatrix(app.bathymetryinputdata2);
                            end

                        end
                        %Enable the 'Land Color' dropdown list & the Plot Bathymetry Contours option in 'Basemap and Overlays' section
                        app.LandColor.Enable = "on";
                        app.PlotBathymetryContoursCheckBox.Enable = "on";
```

```
                end
            end

            %Check if the uploaded file has the required file format to log error
            if ~contains(bathymetryfile, '.txt') && ~contains(bathymetryfile, '.tif') && ~contains(bathymetryfile, 'mask_')
                app.bathymetrycheck = 1;
            end

        end

        % Callback function
        function ButtonGroup_13SelectionChanged(app, event)



        end

        % Value changed function: PlotindegreesCheckBox_2
        function PlotindegreesCheckBox_2ValueChanged(app, event)


        end

        % Callback function
        function Button_23Pushed(app, event)



        end

        % Value changed function: FileTextArea_5
        function FileTextArea_5ValueChanged(app, event)


        end

        % Selection changed function: ButtonGroup_12
        function ButtonGroup_12SelectionChanged(app, event)
            if app.PlotalldatainonefigureButton_2.Value
                app.mp4CheckBox_2.Enable = "off";
                app.FramerateEditFieldLabel.Visible = "off";
                app.FramerateEditField.Visible = "off";
                app.txtCheckBox_2.Enable  = "off";
                app.tifCheckBox.Enable = "off";
            elseif app.PlotseparatelyButton_3.Value
                app.mp4CheckBox_2.Enable = "on";
                app.txtCheckBox_2.Enable  = "on";
                app.tifCheckBox.Enable = "on";
                if  app.mp4CheckBox_2.Value %If it is left checked, show the frame rate option
                    app.FramerateEditFieldLabel.Visible = "on";
                    app.FramerateEditField.Visible = "on";
                else
                    app.FramerateEditFieldLabel.Visible = "off";
                    app.FramerateEditField.Visible = "off";
                end
            end
        end

        % Callback function
        function CurrentDirectionsCheckBoxValueChanged(app, event)

        end

        % Callback function
        function Button_24Pushed(app, event)
            %LOAD THE U VECTOR FILES
            [files, path] = uigetfile('*.*','Select the files','Code files','MultiSelect', 'on');

            %Collect all the files
            allfiles = string(path) + string(files);
            app.Uvectorfullfile = sort(allfiles);
            FileNAME = sort(string(files));
            app.FileTextArea_6.Value = string(FileNAME); %display the filename list
            app.FileTextArea_6.FontColor = 'k';
            app.FileTextArea_6.BackgroundColor = 'w';

            %Find the equivalent V Vector files
            vfiles = string(files);
            vfilenames = regexprep(vfiles,"u","v",'once');  %replaces only the first letter "u" in each of the filenames
            app.Vvectorfullfile = string(path) + vfilenames;
            app.FileTextArea_7.Value = vfilenames;
            app.FileTextArea_7.FontColor = 'k';
            app.FileTextArea_7.BackgroundColor = 'w';


            %Check if any file is loaded
            if matches(app.FileTextArea_3.Value,"0") || matches(app.FileTextArea_3.Value,"00")
                %do nothing
            else
                %READ THE FIRST U FILE TO SETUP MATRIX SIZE AND MAP BOUNDARY LIMIT
                f = readmatrix(app.Ufile(1));
                %SETUP OF THE COORDINATES
                [row,col] = size(f);
                col1 = col-1;
                row1 = row-1;
                app.xcoord = (0:col1) * app.gridX_2.Value+ app.LongitudeEditField_2.Value;
                app.ycoord = [0:row1] * app.gridY_2.Value+ app.LatitudeEditField_2.Value;

                %SET UP THE MAP BOUNDARY
                app.EastEditField_2.Value = max(app.xcoord);
                app.WestEditField_2.Value = min(app.xcoord);
                app.NorthEditField_2.Value = max(app.ycoord);
                app.SouthEditField_2.Value = min(app.ycoord);
            end

        end
        % Value changed function: FileTextArea_3
        function FileTextArea_3ValueChanged(app, event)


        end

        % Size changed function: VelocityMapTab
        function VelocityMapTabSizeChanged(app, event)


        end

        % Callback function
        function PlotColormapCheckBoxValueChanged(app, event)
            if app.PlotColormapCheckBox.Value
                app.BackgroundMapColorDropDown.Enable = "on";
                app.FlipCheckBox_4.Enable = "on";
                app.InterpolationDivisionEditField_2.Enable = "on";
                app.MaxBarValue.Enable = "on";
                app.MinBarValue.Enable = "on";
            else
                app.BackgroundMapColorDropDown.Enable = "off";
                app.FlipCheckBox_4.Enable = "off";
                app.InterpolationDivisionEditField_2.Enable = "off";
                app.MaxBarValue.Enable = "off";
                app.MinBarValue.Enable = "off";
            end

            if app.PlotColormapCheckBox.Value && app.etaButton.Value
                app.Button_23.Enable = "on";
            else
                app.Button_23.Enable = "off";
            end

            if app.BathymetryButton.Value
                app.Button_23.Enable = "off";
            end
        end

        % Value changed function: PlotBathymetryContoursCheckBox
        function PlotBathymetryContoursCheckBoxValueChanged(app, event)
            if app.PlotBathymetryContoursCheckBox.Value
                app.WidthEditField_2.Enable = "on";
                app.StyleDropDown_3.Enable = "on";
                app.IntervalEditField_6.Enable = "on";
                app.MaximumEditField_2.Enable = "on";
                app.MinimumEditField_2.Enable = "on";
                app.ColorDropDown_4.Enable = "on";
                app.AddLabelCheckBox_2.Enable = "on";
                if app.AddLabelCheckBox_2.Value
                    app.IntervalEditField_8.Enable = "on";
                    app.LabelSizeCont.Enable = "on";
                    app.SpacingEditField.Enable = "on";
                else
                    app.IntervalEditField_8.Enable = "off";
                    app.LabelSizeCont.Enable = "off";
                    app.SpacingEditField.Enable = "off";
                end
            else
                app.IntervalEditField_8.Enable = "off";
                app.WidthEditField_2.Enable = "off";
                app.StyleDropDown_3.Enable = "off";
                app.IntervalEditField_6.Enable = "off";
                app.MaximumEditField_2.Enable = "off";
                app.MinimumEditField_2.Enable = "off";
                app.ColorDropDown_4.Enable = "off";
                app.AddLabelCheckBox_2.Enable = "off";
                app.LabelSizeCont.Enable = "off";
                app.SpacingEditField.Enable = "off";
            end
        end

        % Size changed function: Panel_16
        function Panel_16SizeChanged(app, event)


        end

        % Size changed function: BasemapTab
        function BasemapTabSizeChanged(app, event)

        end

        % Value changed function: DepthFileEditField_2
        function DepthFileEditField_2ValueChanged(app, event)

        end

        % Value changed function: AutoSetCheckBox_3
        function AutoSetCheckBox_3ValueChanged(app, event)
            if app.AutoSetCheckBox_3.Value
                app.Height_3.Enable = "off";
                app.Width_3.Enable = "off";
            else
                app.Height_3.Enable = "on";
                app.Width_3.Enable = "on";
            end
        end

        % Size changed function: ButtonGroup_12
        function ButtonGroup_12SizeChanged(app, event)

        end

        % Button pushed function: Button_26
        function Button_26Pushed(app, event)
            workingfolder = uigetdir;
            Dir1 = fullfile(string(workingfolder), 'OUTPUT_FILES');
            app.WORKFOLDER3 = Dir1;
            FigureFolder = fullfile(Dir1, 'Figures');
            app.OutputDirectoryEditField_3.Value = FigureFolder;
            app.WORKFOLDER4 = fullfile(deblank(FigureFolder));
            app.OutputDirectoryEditField_3.FontColor = 'k';

            %Create the folders
            if ~exist(Dir1, 'dir')
                mkdir(Dir1);
            end
            if ~exist(FigureFolder, 'dir')
                mkdir(FigureFolder);
            end
        end

        % Value changed function: CoastlinecolourDropDown_7
        function CoastlinecolourDropDown_7ValueChanged(app, event)
            value = app.CoastlinecolourDropDown_7.Value;
            %List of values based on the cbrewer2 colormap
            validValues = {'blue', 'blue - green', 'blue - purple', 'green - blue', ...
                'greens', 'grays', 'oranges', 'orange - red', ...
                'purple - blue', 'purple - blue - green', 'purple - red', ...
                'purples', 'red - purple', 'reds', 'yellow - green', ...
                'yellow - green - blue', 'yellow - orange - brown', ...
                'yellow - orange - red', 'brown - teal', 'pink - light green', ...
                'purple - green', 'purple - orange', 'red - blue', ...
                'red - gray', 'red - yellow - blue', 'red - yellow - green', ...
                'spectral', 'accent', 'dark 2', 'paired', 'pastel 1', ...
                'pastel 2', 'set 1', 'set 2', 'set 3'};

            % If the selected value in the dropdown list is part of the cbrewer2 colormap, enable the color interpolation textbox
            if ismember(value, validValues)
                app.ColorinterpolationEditField_2.Enable = "on";
                app.DivisionLabel.Enable = "on";
            else
                app.ColorinterpolationEditField_2.Enable = "off";
                app.DivisionLabel.Enable = "off";
            end
        end

        % Value changed function: BackgroundMapColorDropDown
        function BackgroundMapColorDropDownValueChanged(app, event)
            cbrewercolors = {'blue', 'blue - green', 'blue - purple', 'green - blue', 'greens', ...
                'grays', 'oranges', 'orange - red', 'purple - blue', 'purple - blue - green', ...
                'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', ...
                'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', ...
                'brown - teal', 'pink - light green', 'purple - green', 'purple - orange', ...
                'red - blue', 'red - gray', 'red - yellow - blue', 'red - yellow - green', ...
                'spectral', 'accent', 'dark 2', 'paired', 'pastel 1', 'pastel 2', ...
                'set 1', 'set 2', 'set 3'};

            %Activate the interpolation division when the selected colormap is of cbrewer type
            if ismember(app.BackgroundMapColorDropDown.Value, cbrewercolors)
```

```
                app.InterpolationDivisionEditField_2.Enable = "on";
                app.InterpolationDivisionEditFieldLabel_2.Enable = "on";
            else
                app.InterpolationDivisionEditField_2.Enable = "off";
                app.InterpolationDivisionEditFieldLabel_2.Enable = "off";
            end



        end

        % Drop down opening function: BackgroundMapColorDropDown
        function BackgroundMapColorDropDownOpening(app, event)
            app.IntervalEditField_6.FontColor = 'k';
        end

        % Value changed function: IntervalEditField_6
        function IntervalEditField_6ValueChanged(app, event)
            app.IntervalEditField_6.FontColor = 'k';
            app.IntervalEditField_8.Value = app.IntervalEditField_6.Value;
        end

        % Button pushed function: SettoDefaultButton
        function SettoDefaultButtonPushed(app, event)
            %Restore the boundary extent using input from the 'Input Data' section

            %Extract row and column dimensions from the first imported file
            f = readmatrix(app.Uvectorfullfile(1));

            %Determine the coordinates
            [row,col] = size(f);
            col1 = col-1;
            row1 = row-1;
            app.xcoord = [0:col1] * app.gridX_2.Value+ app.LongitudeEditField_2.Value;
            app.ycoord = [0:row1] * app.gridY_2.Value+ app.LatitudeEditField_2.Value;

            %Show the map boundary in the textboxes in 'General Layout'
            app.EastEditField_2.Value = max(app.xcoord);
            app.WestEditField_2.Value = min(app.xcoord);
            app.NorthEditField_2.Value = max(app.ycoord);
            app.SouthEditField_2.Value = min(app.ycoord);

            app.EastEditField_2.FontColor = 'k';
            app.WestEditField_2.FontColor = 'k';
            app.NorthEditField_2.FontColor = 'k';
            app.SouthEditField_2.FontColor = 'k';
            drawnow

        end

        % Button pushed function: SettoDefaultButton_2
        function SettoDefaultButton_2Pushed(app, event)
            %Restore the boundary extent using input from the 'Input Data' section

            %Extract row and column dimensions from the first imported file
            f = readmatrix(app.FileINPUT(1));
            [row,col] = size(f);

            %Determine the coordinates
            col1 = col-1;
            row1 = row-1;
            app.x = app.LongitudeEditField.Value + [0:col1] * app.gridX.Value;
            app.y = app.LatitudeEditField.Value + [0:row1] * app.gridY.Value;


            %Show the map boundary in the textboxes in 'General Layout'
            app.EastEditField.Value = max(app.x);
            app.WestEditField.Value = min(app.x);
            app.NorthEditField.Value = max(app.y);
            app.SouthEditField.Value = min(app.y);
            drawnow

            app.EastEditField.FontColor = 'k';
            app.WestEditField.FontColor = 'k';
            app.NorthEditField.FontColor = 'k';
            app.SouthEditField.FontColor = 'k';

        end

        % Size changed function: Panel_15
        function Panel_15SizeChanged(app, event)


        end

        % Value changed function: PlotGaugesCheckBox
        function PlotGaugesCheckBoxValueChanged(app, event)
            if app.PlotGaugesCheckBox.Value
                app.FileEditField_2.Enable = "on";
                app.Button_27.Enable = "on";
                app.ColorDropDown_7.Enable = "on";
                app.SizeEditField_2.Enable = "on";
                app.gaugemarkerVelocityTab.Enable = "on";
                app.AddLabelCheckBox.Enable = "on";
                if app.AddLabelCheckBox.Value
                    app.GLspacing_2.Enable = "on";
                    app.GLspacing_2.Editable = "on";
                else
                    app.GLspacing_2.Editable = "off";
                    app.GLspacingLabel_2.Enable = "off";
                end

            else
                app.FileEditField_2.Enable = "off";
                app.Button_27.Enable = "off";
                app.ColorDropDown_7.Enable = "off";
                app.SizeEditField_2.Enable = "off";
                app.gaugemarkerVelocityTab.Enable = "off";
                app.AddLabelCheckBox.Enable = "off";
            end


        end

        % Value changed function: AddLabelCheckBox
        function AddLabelCheckBoxValueChanged(app, event)
            if app.AddLabelCheckBox.Value
                app.FontSizeEditField_2.Enable = "on";
                app.FontSizeEditField_2.Editable = "on";
                app.CoastlinecolourDropDown_11.Enable = "on";
                app.CoastlinecolourDropDown_10.Enable = "on";
                app.GLspacing_2.Enable = "on";
                app.GLspacing_2.Editable = "on";
                app.GLspacingLabel_2.Enable = "on";
                app.FontSizeEditField_2Label.Enable = "on";
                app.AlignmentLabel_2.Enable = "on";
            else
                app.FontSizeEditField_2.Enable = "off";
                app.FontSizeEditField_2.Editable = "off";
                app.CoastlinecolourDropDown_11.Enable = "off";
                app.CoastlinecolourDropDown_10.Enable = "off";
                app.GLspacing_2.Enable = "off";
                app.GLspacingLabel_2.Enable = "off";
                app.FontSizeEditField_2Label.Enable = "off";
                app.AlignmentLabel_2.Enable = "off";
            end

        end

        % Callback function
        function AddLabelsCheckBox_2ValueChanged(app, event)
            if app.AddLabelsCheckBox_2.Value
                app.IntervalEditField_3.Enable = "on";
                app.TextLabelSize.Enable = "on";
                app.LabelSpacingEditField.Enable = "on";
                app.SpacingEditField_4Label_4.Enable = "on";
                app.LabelSpacingEditFieldLabel.Enable = "on";
                app.LabelSpacingEditFieldLabel_2.Enable = "on";
            else
                app.IntervalEditField_3.Enable = "off";
                app.TextLabelSize.Enable = "off";
                app.LabelSpacingEditField.Enable = "off";
                app.SpacingEditField_4Label_4.Enable = "off";
                app.LabelSpacingEditFieldLabel.Enable = "off";
                app.LabelSpacingEditFieldLabel_2.Enable = "off";
            end

        end

        % Callback function
        function AddLabelsCheckBox_3ValueChanged(app, event)
            if app.AddLabelsCheckBox_3.Value
                app.TextLabelSize_2.Enable = "on";
                app.wh_interval.Enable = "on";
                app.LabelSpacingEditField_2.Enable = "on";
                app.SpacingEditField_4Label_5.Enable = "on";
                app.LabelSpacingEditField_2Label_2.Enable = "on";
                app.LabelSpacingEditField_2Label.Enable = "on";
            else
                app.TextLabelSize_2.Enable = "off";
                app.wh_interval.Enable = "off";
                app.LabelSpacingEditField_2.Enable = "off";
                app.SpacingEditField_4Label_5.Enable =  "off";
                app.LabelSpacingEditField_2Label_2.Enable =  "off";
                app.LabelSpacingEditField_2Label.Enable =  "off";
            end
        end

        % Callback function
        function AddLabelsCheckBox_4ValueChanged(app, event)
            if app.AddLabelsCheckBox_4.Value
                app.CoastlinecolourDropDown_5.Enable = "on";
                app.CoastlinecolourDropDown_4.Enable = "on";
                app.SizeEditField_3Label.Enable = "on";
                app.AlignmentLabel.Enable = "on";
                app.SizeEditField_3.Enable = "on";
                app.SizeEditField_3.Editable = "on";
                app.GLspacing.Enable = "on";
                app.GLspacing.Editable = "on";
                app.GLspacingLabel.Enable = "on";
                app.FontSizeEditField_2Label.Enable = "on";
                app.AlignmentLabel_2.Enable = "on";
            else
                app.CoastlinecolourDropDown_5.Enable = "off";
                app.CoastlinecolourDropDown_4.Enable = "off";
                app.SizeEditField_3Label.Enable = "off";
                app.AlignmentLabel.Enable = "off";
                app.SizeEditField_3.Enable = "off";
                app.SizeEditField_3.Editable = "off";
                app.GLspacing.Enable = "off";
                app.GLspacing.Editable = "off";
                app.GLspacingLabel.Enable = "off";
                app.FontSizeEditField_2Label.Enable = "off";
                app.AlignmentLabel_2.Enable = "off";
            end
        end

        % Size changed function: GaugesTab_2
        function GaugesTab_2SizeChanged(app, event)


        end

        % Button pushed function: Button_27
        function Button_27Pushed(app, event)
            [filename3,path3] = uigetfile('*.txt;*.shp');
            fullname = fullfile(path3,filename3);

            [~, name, ext] = fileparts(filename3);
            if ~(ismcc || isdeployed)
                addpath(genpath(string(path3)));
            end

            app.GaugeFname3 = name;
            app.FileEditField_2.Value = filename3;
            app.GAUGEFILE3= filename3;
            app.FileEditField_2.FontColor = 'k';
            app.FileEditField_2.BackgroundColor = 'w';

            if strcmp(ext, '.shp')
                S = shaperead(fullname);
                latitudes= [S.Y]';
                longitudes= [S.X]';
                list = [round(latitudes, 4), round(longitudes, 4)];
                list = unique(list, 'rows');
                app.latGauge2 = list(:, 1);
                app.longGauge2 = list(:, 2);

            elseif strcmp(ext, '.txt')
                fileID = fopen(fullname, 'r');
                dataArray = textscan(fileID, '%f %f', 'Delimiter', 'whitespace'); %Read text file and ensure that it is tab delimited
                fclose(fileID);
                app.latGauge2  = dataArray{1};
                app.longGauge2 = dataArray{2};
            end

            app.FileEditField_2.FontColor = 'k';
            app.FileEditField_2.BackgroundColor = 'w';


        end

        % Value changed function: AddLabelCheckBox_2
        function AddLabelCheckBox_2ValueChanged(app, event)
            if app.AddLabelCheckBox_2.Value
                app.LabelSizeCont.Enable = "on";
                app.SpacingEditField.Enable = "on";
                app.IntervalEditField_8.Enable = "on";
            else
                app.LabelSizeCont.Enable = "off";
                app.SpacingEditField.Enable = "off";
                app.IntervalEditField_8.Enable = "off";
            end
        end

        % Value changed function: LongitudeEditField_2
        function LongitudeEditField_2ValueChanged(app, event)
            app.LongitudeEditField_2.FontColor = 'k';

            %Update the X-axis limits in the 'General Layout' section
            app.xcoord = [0:app.col1vel] * app.gridX_2.Value+ app.LongitudeEditField_2.Value;
            app.EastEditField_2.Value = max(app.xcoord);
            app.WestEditField_2.Value = min(app.xcoord);
            drawnow
        end

        % Value changed function: LatitudeEditField_2
        function LatitudeEditField_2ValueChanged(app, event)
            app.LatitudeEditField_2.FontColor = 'k';

            %Update the Y-axis limits in the 'General Layout' section
            app.ycoord = [0:app.row1vel] * app.gridY_2.Value+ app.LatitudeEditField_2.Value;
            app.NorthEditField_2.Value = max(app.ycoord);
            app.SouthEditField_2.Value = min(app.ycoord);
            drawnow
        end

        % Value changed function: gridX_2
        function gridX_2ValueChanged(app, event)
            app.gridX_2.FontColor = 'k';

            %Update the X-axis limits in the 'General Layout' section
            app.xcoord = [0:app.col1vel] * app.gridX_2.Value+ app.LongitudeEditField_2.Value;
            app.EastEditField_2.Value = max(app.xcoord);
            app.WestEditField_2.Value = min(app.xcoord);
            drawnow
        end

        % Value changed function: gridY_2
        function gridY_2ValueChanged(app, event)
            app.gridY_2.FontColor = 'k';

            %Update the Y-axis limits in the 'General Layout' section
            app.ycoord = [0:app.row1vel] * app.gridY_2.Value+ app.LatitudeEditField_2.Value;
            app.NorthEditField_2.Value = max(app.ycoord);
            app.SouthEditField_2.Value = min(app.ycoord);
            drawnow
        end

        % Value changed function: TotalSimuilationTimesecEditField_4
        function TotalSimuilationTimesecEditField_4ValueChanged(app, event)
            app.TotalSimuilationTimesecEditField_4.FontColor = 'k';

        end

        % Value changed function: arrowscale
        function arrowscaleValueChanged(app, event)
            app.arrowscale.FontColor = 'k';

        end

        % Value changed function: ArrowHeadSize
        function ArrowHeadSizeValueChanged(app, event)
            app.ArrowHeadSize.FontColor = 'k';

        end

        % Value changed function: ArrowThickness
        function ArrowThicknessValueChanged(app, event)
            app.ArrowThickness.FontColor = 'k';

        end

        % Callback function
        function ArrowSizeLegendValueChanged(app, event)
            app.ArrowSizeLegend.FontColor = 'k';

        end

        % Callback function
        function ColorbarTextSize_2ValueChanged(app, event)
            app.ColorbarTextSize_2.FontColor = 'k';

        end

        % Callback function
        function ArrowXLabelLocValueChanged(app, event)
            app.ArrowXLabelLoc.FontColor = 'k';

        end

        % Callback function
        function ArrowYLabelLocValueChanged(app, event)
            app.ArrowYLabelLoc.FontColor = 'k';

        end

        % Value changed function: MinimumEditField_2
        function MinimumEditField_2ValueChanged(app, event)
            app.MinimumEditField_2.FontColor = 'k';

        end

        % Value changed function: MaximumEditField_2
        function MaximumEditField_2ValueChanged(app, event)
            app.MaximumEditField_2.FontColor = 'k';

        end

        % Value changed function: WidthEditField_2
        function WidthEditField_2ValueChanged(app, event)
            app.WidthEditField_2.FontColor = 'k';

        end

        % Value changed function: SpacingEditField
        function SpacingEditFieldValueChanged(app, event)
            app.SpacingEditField.FontColor = 'k';

        end

        % Value changed function: LabelSizeCont
        function LabelSizeContValueChanged(app, event)
            app.LabelSizeCont.FontColor = 'k';

        end

        % Value changed function: IntervalEditField_8
        function IntervalEditField_8ValueChanged(app, event)
            app.IntervalEditField_8.FontColor = 'k';

        end

        % Value changed function: NorthEditField_2
        function NorthEditField_2ValueChanged(app, event)
            app.NorthEditField_2.FontColor = 'k';

        end

        % Value changed function: WestEditField_2
        function WestEditField_2ValueChanged(app, event)
            app.WestEditField_2.FontColor = 'k';

        end

        % Value changed function: EastEditField_2
        function EastEditField_2ValueChanged(app, event)
            app.EastEditField_2.FontColor = 'k';

        end

        % Value changed function: SouthEditField_2
        function SouthEditField_2ValueChanged(app, event)
            app.SouthEditField_2.FontColor = 'k';

        end

        % Value changed function: Width_3
        function Width_3ValueChanged(app, event)
            app.Width_3.FontColor = 'k';

        end

        % Value changed function: Height_3
        function Height_3ValueChanged(app, event)
            app.Height_3.FontColor = 'k';

        end

        % Value changed function: OutputDirectoryEditField_3
        function OutputDirectoryEditField_3ValueChanged(app, event)
            app.OutputDirectoryEditField_3.FontColor = 'k';

        end

        % Value changed function: FramerateEditField
        function FramerateEditFieldValueChanged(app, event)
            app.FramerateEditField.FontColor = 'k';

        end

        % Value changed function: maplabelsize
        function maplabelsizeValueChanged(app, event)
            app.maplabelsize.FontColor = 'k';

        end

        % Value changed function: MaxBarValue
        function MaxBarValueValueChanged(app, event)
            app.MaxBarValue.FontColor = 'k';

        end

        % Value changed function: MinBarValue
        function MinBarValueValueChanged(app, event)
```

```
                app.MinBarValue.FontColor = 'k';

        end

        % Value changed function: InterpolationDivisionEditField_2
        function InterpolationDivisionEditField_2ValueChanged(app, event)
            app.InterpolationDivisionEditField_2.FontColor = 'k';

        end

        % Value changed function: FontSizeEditField_2
        function FontSizeEditField_2ValueChanged(app, event)
            app.FontSizeEditField_2.FontColor = 'k';

        end

        % Value changed function: SizeEditField_2
        function SizeEditField_2ValueChanged(app, event)
            app.SizeEditField_2.FontColor = 'k';

        end

        % Value changed function: FileEditField_2
        function FileEditField_2ValueChanged(app, event)

        end

        % Size changed function: WaveHeightMapTab
        function WaveHeightMapTabSizeChanged(app, event)

        end

        % Button pushed function: CloseFiguresButton_2
        function CloseFiguresButton_2Pushed2(app, event)
            close all
        end

        % Value changed function: PlotindegreesCheckBox
        function PlotindegreesCheckBoxValueChanged(app, event)

        end

        % Value changed function: txtCheckBox
        function txtCheckBoxValueChanged(app, event)


        end

        % Value changed function: FlipCheckBox_2
        function FlipCheckBox_2ValueChanged(app, event)


        end

        % Value changed function: YaxisUseDataDropDown
        function YaxisUseDataDropDownValueChanged(app, event)
            % Determine the Y-axis label based on the dropdown value
            switch app.YaxisUseDataDropDown.Value
```

```
                case '1'
                    if ~app.CheckBox_4.Value
                        app.YAxisEditField.Value = "Time (sec)";
                    else %If 'Convert the Values' checkbox is ticked, remove the unit
                        app.YAxisEditField.Value = "Time";
                    end
                case '2'
                    if ~app.CheckBox_4.Value
                        app.YAxisEditField.Value = "\eta (m)";
                    else
                        app.XAxisEditField.Value = "\eta";
                    end
                case {'3', '4', '3,4,5'}
                    if ~app.CheckBox_4.Value
                        app.YAxisEditField.Value = "Velocity (m/s)";
                    else
                        app.YAxisEditField.Value = "Velocity";
                    end
            end

            %Change the font color of 'Axes Labels: Y axis' in 'Plot Style and Layout'
            if ~app.CheckBox_4.Value
                app.YAxisEditField.FontColor = "k";
            end

            %Update which dropdown list to show in 'Plot Properties: Color' in 'Plot Style and Layout'
            if length(app.GaugeDirectorylist) == 1 &&  ~strcmp(app.YaxisUseDataDropDown.Value, '3,4,5')
                app.GaugeLineColor.Visible = "on";
                app.GaugeLineColor.Enable = "on";
                app.GaugeLineColor_multiple.Visible = "off";
                app.GaugeLineColor_multiple.Enable = "off";
                app.FlipCheckBox_3.Visible = "off";
                app.FlipCheckBox_3.Enable = "off";
            else
                app.GaugeLineColor.Visible = "off";
                app.GaugeLineColor.Enable = "off";
                app.GaugeLineColor_multiple.Visible = "on";
                app.GaugeLineColor_multiple.Enable = "on";
                app.FlipCheckBox_3.Visible = "on";
                app.FlipCheckBox_3.Enable = "on";
            end
            drawnow
        end

        % Callback function
        function YCol2ValueChanged(app, event)
          app.YCol2.FontColor = 'k';

        end

        % Callback function
        function YCol3ValueChanged(app, event)
            app.YCol3.FontColor = 'k';

        end

        % Callback function
        function YCol4ValueChanged(app, event)
            app.YCol4.FontColor = 'k';

        end

        % Size changed function: GaugeRecordsTab
        function GaugeRecordsTabSizeChanged(app, event)


        end

        % Size changed function: Panel_11
        function Panel_11SizeChanged(app, event)

        end

        % Selection changed function: ButtonGroup_16
        function ButtonGroup_16SelectionChanged(app, event)
            % Enable parameters for the selected basemap option
            if ~isempty(app.FileTextArea_3.Value) %Check if there are u_ files loaded in the app
                if app.etaButton.Value
                    %Find the corresponding 'eta' file in the directory of the u_ file
                    matchingFiles = [];
                    for i = 1:length(app.ETAfilenames)
                        filename = app.ETAfilenames{i};
                        fullPath = fullfile(app.vectorpath, filename);
                        if exist(fullPath, 'file') == 2 %Append to the list
                            matchingFiles = [matchingFiles; filename];
                        end
                    end

                    if isempty(matchingFiles)
                        %No matching files found
                        app.FileTextArea_5.Value = 'No matching files found';

                        %Disable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'off';
                        end

                    else  %At least one matching file was found
                        app.FileTextArea_5.Value = strjoin(string(matchingFiles), newline);

                        %Enable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'on';
                        end
                        app.BackgroundMapColorDropDown.Value = 'red - blue';
                        app.FlipCheckBox_4.Value = true;
                    end
```

```matlab

                elseif app.hmaxButton.Value
                    %Find the corresponding 'hmax' file in the directory of the U_ file
                    matchingFiles = [];
                    for i = 1:length(app.HMAXfilenames)
                        filename = app.HMAXfilenames{i};
                        fullPath = fullfile(app.vectorpath, filename);
                        if exist(fullPath, 'file') == 2
                            matchingFiles = [matchingFiles; filename];
                        end
                    end

                    if isempty(matchingFiles)
                        % No matching files found
                        app.FileTextArea_5.Value = 'No matching files found';

                        %Disable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'off';
                        end
                    else
                        % At least one matching file was found
                        app.FileTextArea_5.Value = strjoin(string(matchingFiles), newline);

                        %Enable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'on';
                        end
                        app.BackgroundMapColorDropDown.Value = 'parula';
                        app.FlipCheckBox_4.Value = false;
                        app.InterpolationDivisionEditField_2.Enable = 'off';
                    end

                elseif app.BathymetryButton.Value
                    %Use bathymetry data as the basemap
                    if isempty(app.DepthFileEditField_2.Value) %If a bathymetry file is not loaded in the 'Input Data' section
                        % No matching files found
                        app.FileTextArea_5.Value = 'No bathymetry file found'; drawnow

                        %Disable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'off';
                        end
                        app.LandColor.Enable = 'off';
                    else
                        %Enable colormap and colorbar options
                        children = app.ThirdTabColorMapPanel.Children;
                        for i = 1:length(children)
                            children(i).Enable = 'on';
                        end
                         app.FileTextArea_5.Value = ' ';
                        app.BackgroundMapColorDropDown.Value = 'parula';
                        app.FlipCheckBox_4.Value = false;
                        app.InterpolationDivisionEditField_2.Enable = 'off';
                    end

                elseif app.VelocityButton.Value
                    children = app.ThirdTabColorMapPanel.Children;
                    if ~isempty(app.FileNAME5)
                        % Use the Z magnitude, calculated from the U and V vectors, as the basemap
                        %Enable colormap and colorbar options
                        for i = 1:length(children)
                            children(i).Enable = 'on';
                        end

                        %Change the default colormap and bar limits to match the damage index ranges in Lynett et al. (2022)
                        app.BackgroundMapColorDropDown.Value = 'blue - purple';
                        app.InterpolationDivisionEditField_2.Value = 4;
                        app.InterpolationDivisionEditField_2.Enable = "on";
                    else
                        for i = 1:length(children)
                            children(i).Enable = 'off';
                        end
                    end

                elseif app.VorticityButton.Value
                    children = app.ThirdTabColorMapPanel.Children;
                    if ~isempty(app.FileNAME5)
                        % Use the Z magnitude, calculated from the U and V vectors, as the basemap
                        %Enable colormap and colorbar options
                        for i = 1:length(children)
                            children(i).Enable = 'on';
                        end
                        app.BackgroundMapColorDropDown.Value = 'jet';
                        app.FlipCheckBox_4.Value = false;
                        app.InterpolationDivisionEditField_2.Enable = 'off';
                    else
                        for i = 1:length(children)
                            children(i).Enable = 'off';
                        end
                    end

                end

                ArrowSection = app.Panel_14.Children;
                if ~app.VorticityButton.Value
                    %Enable the Arrow Section
                    for i = 1:length(ArrowSection)
                        ArrowSection(i).Enable = 'on';
                    end
                else
                    %Disable the Arrow Section
                    for i = 1:length(ArrowSection)
                        ArrowSection(i).Enable = 'off';
                    end
```

```
                        app.PlotvectorsCheckBox.Value = false; %Uncheck the box
                    end

                    if contains(string(app.FileNAME5), 'umax')
                        for i = 1:length(ArrowSection)
                            ArrowSection(i).Enable = 'off';
                        end
                    end
                end

        end

        % Value changed function: LongitudeEditField
        function LongitudeEditFieldValueChanged(app, event)
            app.LongitudeEditField.FontColor = 'k';

            %Update the X-axis limits in the 'General Layout' section
            app.x = app.LongitudeEditField.Value + [0:app.col1] * app.gridX.Value;
            app.EastEditField.Value = max(app.x);
            app.WestEditField.Value = min(app.x);
            drawnow
        end

        % Value changed function: LatitudeEditField
        function LatitudeEditFieldValueChanged(app, event)
            app.LatitudeEditField.FontColor = 'k';

            %Update the Y-axis limits in the 'General Layout' section
            app.y = [0:app.row1] * app.gridY.Value+ app.LatitudeEditField.Value;
            app.NorthEditField.Value = max(app.y);
            app.SouthEditField.Value = min(app.y);
            drawnow
        end

        % Value changed function: gridX
        function gridXValueChanged(app, event)
            app.gridX.FontColor = 'k';

            %Update the X-axis limits in the 'General Layout' section
            app.x = app.LongitudeEditField.Value + [0:app.col1] * app.gridX.Value;
            app.EastEditField.Value = max(app.x);
            app.WestEditField.Value = min(app.x);
            drawnow

        end

        % Value changed function: gridY
        function gridYValueChanged(app, event)
            app.gridY.FontColor = 'k';

            %Update the Y-axis limits in the 'General Layout' section
            app.y = [0:app.row1] * app.gridY.Value+ app.LatitudeEditField.Value;
            app.NorthEditField.Value = max(app.y);
            app.SouthEditField.Value = min(app.y);
            drawnow
        end

        % Value changed function: TotalSimuilationTimesecEditField_3
        function TotalSimuilationTimesecEditField_3ValueChanged(app, event)
            app.TotalSimuilationTimesecEditField_3.FontColor = 'k';

        end

        % Value changed function: tifCheckBox_2
        function tifCheckBox_2ValueChanged(app, event)


        end

        % Callback function
        function ArrivalTimeCheckBoxValueChanged(app, event)


        end

        % Callback function
        function TimeFileButtonPushed(app, event)
            [timefile,path2] = uigetfile('*.*', 'Select the files', 'MultiSelect', 'on'); %uigetfile('*.txt;*.out'); %;*.tiff;*.tif;*.asdf;*.asc;*.grd;*.flt;*.grb;*.grid;*.grib2;*.dt0;*.dt1;*.dt2;*.ddf;*.dem;*.hgt;*.grd;*.grc;*.ers;*.dat;*.img;*.jp2'); uigetfile('*.txt;*.shp')
            app.FileEditField_3.Value = string(timefile);
            app.FileEditField_3.FontColor = 'k';

            if ~(ismcc || isdeployed)
                addpath(genpath(fullfile(string(path2))));
            end

            conditions = [
                ~strcmp(app.FileEditField_3.Value, "0"),
                ~strcmp(app.FileEditField_3.Value, "00")
                ];

            if app.FileEditField_3.Value == string(timefile)
                if any(conditions)
                    try
                        ArrivalTimeinputdata = string(path2) + string(timefile);
                        app.ArrivalTimename = string(timefile);
                        app.FileEditField_3.BackgroundColor = 'w';
                        app.FileEditField_3.FontColor =  'k';
                        ArrivalTimeDATA = readmatrix(ArrivalTimeinputdata);
                        app.ArrivalTimeConverted =  ArrivalTimeDATA ./ 60; % convert the values to minutes. Default threshold to read first wave arrival time in FUNWAVE is 0.001 m
                        app.MaximumdepthEditField_3.Value = max(max(app.ArrivalTimeConverted));
                        app.MaximumdepthEditField_3.FontColor = 'k';
                    end
                end
            end

        end

        % Callback function
        function AddLabelsCheckBox_5ValueChanged(app, event)
            if app.AddLabelsCheckBox_5.Value
                app.SpacingEditField_4Label_6.Enable = "on";
                app.TextLabelSize_3.Enable = "on";
                app.LabelSpacingEditField_2Label_3.Enable = "on";
                app.LabelSpacingEditField_4.Enable = "on";
                app.LabelSpacingEditField_2Label_4.Enable = "on";
                app.IntervalEditField_7.Enable = "on";
            else
                app.SpacingEditField_4Label_6.Enable = "off";
                app.TextLabelSize_3.Enable = "off";
                app.LabelSpacingEditField_2Label_3.Enable = "off";
                app.LabelSpacingEditField_4.Enable = "off";
                app.LabelSpacingEditField_2Label_4.Enable = "off";
                app.IntervalEditField_7.Enable = "off";
            end

        end

        % Value changed function: XaxisUseDataDropDown
        function XaxisUseDataDropDownValueChanged(app, event)
            % Determine the X-axis label based on the dropdown value
            switch app.XaxisUseDataDropDown.Value
                case '1'
                    if ~app.CheckBox_3.Value
                        app.XAxisEditField.Value = "Time (sec)";
                    else %If 'Convert the Values' checkbox is ticked, remove the unit
                        app.XAxisEditField.Value = "Time";
                    end
                case '2'
                    if ~app.CheckBox_3.Value
                        app.XAxisEditField.Value = "\eta (m)";
                    else
                        app.XAxisEditField.Value = "\eta";
                    end
                case {'3', '4'}  % Handles both '3' and '4'
                    if ~app.CheckBox_3.Value
                        app.XAxisEditField.Value = "Velocity (m/s)";
                    else
                        app.XAxisEditField.Value = "Velocity";
                    end
            end

            %Change the font color of 'Axes Labels: X axis' in 'Plot Style and Layout'
            if ~app.CheckBox_3.Value
                app.XAxisEditField.FontColor = "k";
            end


        end

        % Selection changed function: ButtonGroup_17
        function ButtonGroup_17SelectionChanged(app, event)
```

```
            %Display parameter options for each tab when selected
            if app.BathymetryButton_2.Value
                app.TabGroup2.SelectedTab = app.BathymetryTab;
            elseif app.WaveHeightButton.Value
                app.TabGroup2.SelectedTab = app.WaveHeightTab;
            elseif app.ArrivalTimeButton.Value
                app.TabGroup2.SelectedTab = app.ArrivalTimeTab;
            end

            tabs = {app.BathymetryTab, app.WaveHeightTab, app.ArrivalTimeTab};
            components1 = {app.LabelSpacingEditField, app.IntervalEditField_3, app.TextLabelSize,   app.LabelSpacingEditFieldLabel_2, app.LabelSpacingEditFieldLabel, app.SpacingEditField_4Label_4};
            components2 = {app.LabelSpacingEditField_2, app.wh_interval, app.TextLabelSize_2, app.LabelSpacingEditField_2Label_2, app.LabelSpacingEditField_2Label, app.SpacingEditField_4Label_5};
            components3 = {app.LabelSpacingEditField_2Label_3, app.LabelSpacingEditField_4, app.SpacingEditField_4Label_6, app.TextLabelSize_3, app.IntervalEditField_7, app.LabelSpacingEditField_2Label_4};

            for k = 1:length(tabs)
                children = tabs{k}.Children;
                for i = 1:length(children)
                    if app.NoneButton.Value   %Disable all the items when 'None' is selected
                        children(i).Enable = 'off';
                    else  %Enable all the items
                        children(i).Enable = 'on';

                        %For 'Add Labels' suboptions, enable only when the box is ticked
                        %Bathymetry Tab
                        for i = 1:length(components1)
                            if  app.AddLabelsCheckBox_2.Value
                                components1{i}.Enable = 'on';
                            else
                                components1{i}.Enable = 'off';
                            end
                        end

                        %Wave Height Tab
                        for i = 1:length(components2)
                            if app.AddLabelsCheckBox_3.Value
                                components2{i}.Enable = 'on';
                            else
                                components2{i}.Enable = 'off';
                            end
                        end

                        %Arrival Time Tab
                        for i = 1:length(components3)
                            if app.AddLabelsCheckBox_5.Value
                                components3{i}.Enable = 'on';
                            else
                                components3{i}.Enable = 'off';
                            end
                        end

                    end
                end
            end
        end

        % Size changed function: ButtonGroup_17
        function ButtonGroup_17SizeChanged(app, event)

        end

        % Callback function
        function LineintervalEditField_3ValueChanged(app, event)
            app.IntervalEditField_7.Value =   app.LineintervalEditField_3.Value;
            app.LineintervalEditField_3.FontColor = 'k';

        end

        % Callback function
        function LabelSpacingEditField_4ValueChanged(app, event)
            app.LabelSpacingEditField_4.FontColor = 'k';

        end

        % Callback function
        function TextLabelSize_3ValueChanged(app, event)
            app.TextLabelSize_3.FontColor = 'k';

        end

        % Callback function
        function ThicknessEditField_3ValueChanged(app, event)
            app.ThicknessEditField_3.FontColor = 'k';

        end

        % Callback function
        function MinimumdepthEditField_3ValueChanged(app, event)
            app.MinimumdepthEditField_3.FontColor = 'k';

        end

        % Callback function
        function MaximumdepthEditField_3ValueChanged(app, event)
            app.MaximumdepthEditField_3.FontColor = 'k';

        end

        % Drop down opening function: YaxisUseDataDropDown
        function YaxisUseDataDropDownOpening(app, event)

        end

        % Value changed function: ArrowSpacing
        function ArrowSpacingValueChanged(app, event)
            app.ArrowSpacing.FontColor = 'k';

        end
```

```matlab

        % Value changed function: StartTime
        function StartTimeValueChanged(app, event)
            app.StartTime.FontColor = 'k';
        end

        % Value changed function: ColorbarTextSize
        function ColorbarTextSizeValueChanged(app, event)
            app.ColorbarTextSize.FontColor = 'k';

        end

        % Value changed function: FlipCheckBox_5
        function FlipCheckBox_5ValueChanged(app, event)

        end

        % Value changed function: StartTime2
        function StartTime2ValueChanged(app, event)


        end

        % Value changed function: PlotvectorsCheckBox
        function PlotvectorsCheckBoxValueChanged(app, event)


        end

        % Value changed function: GLspacing_2
        function GLspacing_2ValueChanged(app, event)
       app.GLspacing_2.FontColor = 'k';

        end

        % Value changing function: XAxisEditField
        function XAxisEditFieldValueChanging(app, event)

        end

        % Value changed function: legendfirsttext
        function legendfirsttextValueChanged(app, event)
         app.legendfirsttext.FontColor = 'k';

        end

        % Value changed function: LineYThickness_2
        function LineYThickness_2ValueChanged(app, event)
         app.LineYThickness_2.FontColor = 'k';

        end

        % Value changed function: Width
        function WidthValueChanged(app, event)
           app.Width.FontColor = 'k';

        end

        % Value changed function: Height
        function HeightValueChanged(app, event)
            app.Height.FontColor = 'k';

        end

        % Value changed function: Width_2
        function Width_2ValueChanged(app, event)
            app.Width_2.FontColor = 'k';

        end

        % Value changed function: Height_2
        function Height_2ValueChanged(app, event)
            app.Height_2.FontColor = 'k';

        end

        % Value changed function: OutputDirectoryEditField_2
        function OutputDirectoryEditField_2ValueChanged(app, event)


        end

        % Value changed function: mp4CheckBox_2
        function mp4CheckBox_2ValueChanged(app, event)
            if app.mp4CheckBox_2.Value
                app.FramerateEditFieldLabel.Visible = "on";
                app.FramerateEditField.Visible = "on";
            else
                app.FramerateEditFieldLabel.Visible = "off";
                app.FramerateEditField.Visible = "off";
            end

        end

        % Callback function
        function IntervalEditField_7ValueChanged(app, event)
            app.IntervalEditField_7.FontColor = 'k';
        end

        % Button pushed function: TimeFileButton
        function TimeFileButtonPushed2(app, event)
            [timefile,path2] = uigetfile('*.*', 'Select the files',↙
'MultiSelect', 'on'); %uigetfile('*.txt;*.out'); %;*.tiff;*.tif;*.asdf;*.asc;*.grd;↙
*.flt;*.grb;*.grid;*.grib2;*.dt0;*.dt1;*.dt2;*.ddf;*.dem;*.hgt;*.grd;*.grc;*.ers;*.↙
dat;*.img;*.jp2'); uigetfile('*.txt;*.shp')
            app.FileEditField_3.Value = string(timefile);
            app.FileEditField_3.FontColor = 'k';

            if ~(ismcc || isdeployed)
                addpath(genpath(fullfile(string(path2))));
            end

            conditions = [
                ~strcmp(app.FileEditField_3.Value, "0"),
                ~strcmp(app.FileEditField_3.Value, "00")
                ];

            if app.FileEditField_3.Value == string(timefile)
                if any(conditions)
                    try
                        ArrivalTimeinputdata = string(path2) + string(timefile);
                        app.ArrivalTimename = string(timefile);
                        app.FileEditField_3.BackgroundColor = 'w';
                        app.FileEditField_3.FontColor =  'k';
                        ArrivalTimeDATA = readmatrix(ArrivalTimeinputdata);
                        app.ArrivalTimeConverted =  round(ArrivalTimeDATA ./60, 2); %convert the values to minutes. Default threshold to read first wave arrival time in FUNWAVE is 0.001 m
                        app.MaximumdepthEditField_3.Value = max(max(app.ArrivalTimeConverted));
                        app.MaximumdepthEditField_3.FontColor = 'k';
                    end
                end
            end

        end

        % Button pushed function: Button_8
        function Button_8Pushed2(app, event)

            [filename3,path3] = uigetfile('*.txt;*.shp');
            fullname = fullfile(path3,filename3);

            [~, name, ext] = fileparts(filename3);
            if ~(ismcc || isdeployed)
                addpath(genpath(string(path3)));
            end

            app.GaugeFname = name;
            app.GAUGEFILE = filename3;
            app.FileEditField.Value = filename3;
            app.FileEditField.FontColor = 'k';
            app.FileEditField.BackgroundColor = 'w';


            if strcmp(ext, '.shp')
                S = shaperead(fullname);
                latitudes= [S.Y]';
                longitudes= [S.X]';
                list = [round(latitudes, 4), round(longitudes, 4)];
                list = unique(list, 'rows');
                app.latGauge = list(:, 1);
                app.longGauge =  list(:, 2);
            elseif strcmp(ext, '.txt')
                fileID = fopen(fullname, 'r');
                dataArray = textscan(fileID, '%f %f', 'Delimiter', 'whitespace'); %Read text file and ensure that it is tab delimited
                fclose(fileID);
                app.STATION_FILE = string(filename3);
                app.latGauge  = dataArray{1};
                app.longGauge = dataArray{2};
            end

        end

        % Value changed function: AddLabelsCheckBox_4
        function AddLabelsCheckBox_4ValueChanged2(app, event)

               if app.AddLabelsCheckBox_4.Value
                app.CoastlinecolourDropDown_5.Enable = "on";
                app.CoastlinecolourDropDown_4.Enable = "on";
                app.SizeEditField_3Label.Enable = "on";
                app.AlignmentLabel.Enable = "on";
                app.SizeEditField_3.Enable = "on";
                app.SizeEditField_3.Editable = "on";
                app.GLspacing.Enable = "on";
                app.GLspacing.Editable = "on";
                app.GLspacingLabel.Enable = "on";
                app.FontSizeEditField_2Label.Enable = "on";
                app.AlignmentLabel_2.Enable = "on";
            else
                app.CoastlinecolourDropDown_5.Enable = "off";
                app.CoastlinecolourDropDown_4.Enable = "off";
                app.SizeEditField_3Label.Enable = "off";
                app.AlignmentLabel.Enable = "off";
                app.SizeEditField_3.Enable = "off";
                app.SizeEditField_3.Editable = "off";
                app.GLspacing.Enable = "off";
                app.GLspacing.Editable = "off";
                app.GLspacingLabel.Enable = "off";
                app.FontSizeEditField_2Label.Enable = "off";
                app.AlignmentLabel_2.Enable = "off";
            end
        end

        % Value changed function: AddLabelsCheckBox_5
        function AddLabelsCheckBox_5ValueChanged2(app, event)
            if app.AddLabelsCheckBox_5.Value
                app.SpacingEditField_4Label_6.Enable = "on";
                app.TextLabelSize_3.Enable = "on";
                app.LabelSpacingEditField_2Label_3.Enable = "on";
                app.LabelSpacingEditField_4.Enable = "on";
                app.LabelSpacingEditField_2Label_4.Enable = "on";
                app.IntervalEditField_7.Enable = "on";
            else
                app.SpacingEditField_4Label_6.Enable = "off";
                app.TextLabelSize_3.Enable = "off";
                app.LabelSpacingEditField_2Label_3.Enable = "off";
                app.LabelSpacingEditField_4.Enable = "off";
                app.LabelSpacingEditField_2Label_4.Enable = "off";
                app.IntervalEditField_7.Enable = "off";
            end
        end

        % Value changed function: AddLabelsCheckBox_3
        function AddLabelsCheckBox_3ValueChanged2(app, event)
            if app.AddLabelsCheckBox_3.Value
                app.TextLabelSize_2.Enable = "on";
                app.wh_interval.Enable = "on";
                app.LabelSpacingEditField_2.Enable = "on";
                app.SpacingEditField_4Label_5.Enable = "on";
                app.LabelSpacingEditField_2Label_2.Enable = "on";
                app.LabelSpacingEditField_2Label.Enable = "on";
            else
                app.TextLabelSize_2.Enable = "off";
                app.wh_interval.Enable = "off";
                app.LabelSpacingEditField_2.Enable = "off";
                app.SpacingEditField_4Label_5.Enable =  "off";
                app.LabelSpacingEditField_2Label_2.Enable =  "off";
                app.LabelSpacingEditField_2Label.Enable =  "off";
            end
        end

        % Value changed function: AddLabelsCheckBox_2
        function AddLabelsCheckBox_2ValueChanged2(app, event)
             if app.AddLabelsCheckBox_2.Value
                app.IntervalEditField_3.Enable = "on";
                app.TextLabelSize.Enable = "on";
                app.LabelSpacingEditField.Enable = "on";
                app.SpacingEditField_4Label_4.Enable = "on";
                app.LabelSpacingEditFieldLabel.Enable = "on";
                app.LabelSpacingEditFieldLabel_2.Enable = "on";
            else
                app.IntervalEditField_3.Enable = "off";
                app.TextLabelSize.Enable = "off";
                app.LabelSpacingEditField.Enable = "off";
                app.SpacingEditField_4Label_4.Enable = "off";
                app.LabelSpacingEditFieldLabel.Enable = "off";
                app.LabelSpacingEditFieldLabel_2.Enable = "off";
            end
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Position = [1 11 542 861];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.Scrollable = 'on';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.AutoResizeChildren = 'off';
            app.TabGroup.Position = [1 1 542 861];

            % Create WaveHeightMapTab
            app.WaveHeightMapTab = uitab(app.TabGroup);
            app.WaveHeightMapTab.AutoResizeChildren = 'off';
            app.WaveHeightMapTab.SizeChangedFcn = createCallbackFcn(app, ↙
@WaveHeightMapTabSizeChanged, true);
            app.WaveHeightMapTab.Title = 'Wave Height Map';
            app.WaveHeightMapTab.Scrollable = 'on';

            % Create Panel_3
            app.Panel_3 = uipanel(app.WaveHeightMapTab);
            app.Panel_3.AutoResizeChildren = 'off';
            app.Panel_3.ForegroundColor = [0.149 0.149 0.149];
            app.Panel_3.Position = [11 662 520 160];

            % Create FileTextArea
            app.FileTextArea = uitextarea(app.Panel_3);
            app.FileTextArea.ValueChangedFcn = createCallbackFcn(app, @FileTextAreaValueChanged, true);
            app.FileTextArea.Editable = 'off';
            app.FileTextArea.FontColor = [0.149 0.149 0.149];
            app.FileTextArea.Tooltip = {''};
            app.FileTextArea.Placeholder = 'eta_xxxx; hmax_xxxx;';
            app.FileTextArea.Position = [43 65 95 55];

            % Create Button_14
            app.Button_14 = uibutton(app.Panel_3, 'push');
            app.Button_14.ButtonPushedFcn = createCallbackFcn(app, @Button_14Pushed, true);
            app.Button_14.FontAngle = 'italic';
            app.Button_14.FontColor = [0.651 0.651 0.651];
            app.Button_14.Tooltip = {'Load the files'; ''; 'Files must start with ''eta'' or ''hmax'''};
            app.Button_14.Position = [146 101 19 19];
            app.Button_14.Text = '...';

            % Create SouthwestCornerLabel
            app.SouthwestCornerLabel = uilabel(app.Panel_3);
            app.SouthwestCornerLabel.FontWeight = 'bold';
            app.SouthwestCornerLabel.Tooltip = {'Coordinates of the southwest corner of the loaded files.'};
            app.SouthwestCornerLabel.Position = [232 121 110 22];
            app.SouthwestCornerLabel.Text = 'Southwest Corner';

            % Create LongitudeEditFieldLabel
            app.LongitudeEditFieldLabel = uilabel(app.Panel_3);
            app.LongitudeEditFieldLabel.HorizontalAlignment = 'right';
            app.LongitudeEditFieldLabel.WordWrap = 'on';
            app.LongitudeEditFieldLabel.Tooltip = {'West boundary'; ''; 'Unit: degrees / meters'};
            app.LongitudeEditFieldLabel.Position = [228 98 60 22];
            app.LongitudeEditFieldLabel.Text = 'Longitude';

            % Create LongitudeEditField
            app.LongitudeEditField = uieditfield(app.Panel_3, 'numeric');
            app.LongitudeEditField.ValueDisplayFormat = '%8.4f';
            app.LongitudeEditField.ValueChangedFcn = createCallbackFcn(app, @LongitudeEditFieldValueChanged, true);
            app.LongitudeEditField.FontColor = [0.651 0.651 0.651];
            app.LongitudeEditField.Tooltip = {''};
            app.LongitudeEditField.Position = [296 104 60 16];

            % Create LatitudeEditFieldLabel
            app.LatitudeEditFieldLabel = uilabel(app.Panel_3);
            app.LatitudeEditFieldLabel.HorizontalAlignment = 'right';
            app.LatitudeEditFieldLabel.WordWrap = 'on';
            app.LatitudeEditFieldLabel.Tooltip = {'South boundary'; ''; 'Unit:
```

```
degrees / meters'};
            app.LatitudeEditFieldLabel.Position = [234 81 44 17];
            app.LatitudeEditFieldLabel.Text = 'Latitude';

            % Create LatitudeEditField
            app.LatitudeEditField = uieditfield(app.Panel_3, 'numeric');
            app.LatitudeEditField.ValueDisplayFormat = '%8.4f';
            app.LatitudeEditField.ValueChangedFcn = createCallbackFcn(app, @LatitudeEditFieldValueChanged, true);
            app.LatitudeEditField.FontColor = [0.651 0.651 0.651];
            app.LatitudeEditField.Tooltip = {''};
            app.LatitudeEditField.Position = [296 82 60 16];

            % Create Button_15
            app.Button_15 = uibutton(app.Panel_3, 'push');
            app.Button_15.ButtonPushedFcn = createCallbackFcn(app, @Button_15Pushed, true);
            app.Button_15.FontAngle = 'italic';
            app.Button_15.FontColor = [0.651 0.651 0.651];
            app.Button_15.Tooltip = {'Load the file'; ''; 'Select the bathymetry used in the simulation, or the output ''mask_xxxx'' file.'; ''; 'Format accepted: .txt, .tif, mask_'};
            app.Button_15.Position = [146 11 19 19];
            app.Button_15.Text = '...';

            % Create DepthFileEditField
            app.DepthFileEditField = uieditfield(app.Panel_3, 'text');
            app.DepthFileEditField.ValueChangedFcn = createCallbackFcn(app, @DepthFileEditFieldValueChanged, true);
            app.DepthFileEditField.Editable = 'off';
            app.DepthFileEditField.FontColor = [0.149 0.149 0.149];
            app.DepthFileEditField.Tooltip = {''};
            app.DepthFileEditField.Placeholder = '.txt, .tif., mask_';
            app.DepthFileEditField.Position = [44 13 96 16];

            % Create FilesLabel
            app.FilesLabel = uilabel(app.Panel_3);
            app.FilesLabel.FontWeight = 'bold';
            app.FilesLabel.Tooltip = {'Load the files'; ''; 'Files must start with ''eta'' or ''hmax'''};
            app.FilesLabel.Position = [44 121 32 22];
            app.FilesLabel.Text = 'Files';

            % Create BathymetryLabel
            app.BathymetryLabel = uilabel(app.Panel_3);
            app.BathymetryLabel.FontWeight = 'bold';
            app.BathymetryLabel.Tooltip = {'Load the file'; ''; 'Select the bathymetry used in the simulation, or the output ''mask_xxxx'' file.'; ''; 'Format accepted: .txt, .tif, mask_'};
            app.BathymetryLabel.Position = [44 31 71 22];
            app.BathymetryLabel.Text = 'Bathymetry';

            % Create TotalSimuilationTimesecEditFieldLabel_3
            app.TotalSimuilationTimesecEditFieldLabel_3 = uilabel(app.Panel_3);
            app.TotalSimuilationTimesecEditFieldLabel_3.FontWeight = 'bold';
            app.TotalSimuilationTimesecEditFieldLabel_3.Tooltip = {'For values >60 seconds, the unit will be automatically converted to minutes or hours. The user
```

```
can verify whether the conversion is correct by double-checking the LOG.TXT file↙
generated after the simulation.'; ''; 'For example, in the line:'; '“PRINTING FILE↙
NO. 1  TIME/TOTAL: 3600.000 / 7200.000”'; 'the hmax or eta 00001 entry will be↙
labelled with a title corresponding to 1 hour.'};
            app.TotalSimuilationTimesecEditFieldLabel_3.Position = [232 31 98↙
22];
            app.TotalSimuilationTimesecEditFieldLabel_3.Text = 'Simulation↙
Time';

            % Create TotalSimuilationTimesecEditField_3
            app.TotalSimuilationTimesecEditField_3 = uieditfield(app.Panel_3,↙
'numeric');
            app.TotalSimuilationTimesecEditField_3.Limits = [1 Inf];
            app.TotalSimuilationTimesecEditField_3.ValueDisplayFormat = '%8.1↙
f';
            app.TotalSimuilationTimesecEditField_3.ValueChangedFcn =↙
createCallbackFcn(app, @TotalSimuilationTimesecEditField_3ValueChanged, true);
            app.TotalSimuilationTimesecEditField_3.FontColor = [0.651 0.651↙
0.651];
            app.TotalSimuilationTimesecEditField_3.Tooltip = {''};
            app.TotalSimuilationTimesecEditField_3.Position = [409 16 60 16];
            app.TotalSimuilationTimesecEditField_3.Value = 1;

            % Create IntervalLabel
            app.IntervalLabel = uilabel(app.Panel_3);
            app.IntervalLabel.HorizontalAlignment = 'right';
            app.IntervalLabel.Tooltip = {'Snapshot interval.'; ''; 'Same as↙
PLOT_INTV in input.txt'; ''; 'Unit: seconds'};
            app.IntervalLabel.Position = [358 13 45 22];
            app.IntervalLabel.Text = 'Interval';

            % Create StartTime
            app.StartTime = uieditfield(app.Panel_3, 'numeric');
            app.StartTime.Limits = [0 Inf];
            app.StartTime.ValueDisplayFormat = '%8.1f';
            app.StartTime.ValueChangedFcn = createCallbackFcn(app, ↙
@StartTimeValueChanged, true);
            app.StartTime.FontColor = [0.651 0.651 0.651];
            app.StartTime.Tooltip = {''};
            app.StartTime.Position = [296 16 47 16];

            % Create StartLabel
            app.StartLabel = uilabel(app.Panel_3);
            app.StartLabel.HorizontalAlignment = 'right';
            app.StartLabel.Tooltip = {'For imported initial tsunami files with↙
a non-zero start time, input the value here. Example: for a 30-second snapshot, ↙
input ''30''.'; ''; 'Unit: seconds'};
            app.StartLabel.Position = [255 13 31 22];
            app.StartLabel.Text = 'Start';

            % Create GridSizeLabel_2
            app.GridSizeLabel_2 = uilabel(app.Panel_3);
            app.GridSizeLabel_2.FontWeight = 'bold';
            app.GridSizeLabel_2.Tooltip = {'Resolution'; 'Unit: degrees /↙
meters'};
            app.GridSizeLabel_2.Position = [386 121 57 22];
            app.GridSizeLabel_2.Text = 'Grid Size';

            % Create XEditField_2Label
            app.XEditField_2Label = uilabel(app.Panel_3);
            app.XEditField_2Label.HorizontalAlignment = 'center';
            app.XEditField_2Label.Tooltip = {'X direction'; ''; 'Unit: degrees / meters'};
            app.XEditField_2Label.Position = [388 98 13 22];
            app.XEditField_2Label.Text = 'X ';

            % Create gridX
            app.gridX = uieditfield(app.Panel_3, 'numeric');
            app.gridX.Limits = [0 Inf];
            app.gridX.ValueDisplayFormat = '%8.5f';
            app.gridX.ValueChangedFcn = createCallbackFcn(app, @gridXValueChanged, true);
            app.gridX.FontColor = [0.651 0.651 0.651];
            app.gridX.Tooltip = {''};
            app.gridX.Position = [409 104 60 16];

            % Create YEditField_2Label
            app.YEditField_2Label = uilabel(app.Panel_3);
            app.YEditField_2Label.HorizontalAlignment = 'center';
            app.YEditField_2Label.Tooltip = {'Y direction'; ''; 'Unit: degrees / meters'};
            app.YEditField_2Label.Position = [386 79 13 21];
            app.YEditField_2Label.Text = 'Y';

            % Create gridY
            app.gridY = uieditfield(app.Panel_3, 'numeric');
            app.gridY.Limits = [0 Inf];
            app.gridY.ValueDisplayFormat = '%8.5f';
            app.gridY.ValueChangedFcn = createCallbackFcn(app, @gridYValueChanged, true);
            app.gridY.FontColor = [0.651 0.651 0.651];
            app.gridY.Tooltip = {''};
            app.gridY.Position = [409 82 60 16];

            % Create InputDataLabel
            app.InputDataLabel = uilabel(app.WaveHeightMapTab);
            app.InputDataLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.InputDataLabel.FontSize = 15;
            app.InputDataLabel.FontWeight = 'bold';
            app.InputDataLabel.FontColor = [0.0314 0.3686 0.6];
            app.InputDataLabel.Tooltip = {''};
            app.InputDataLabel.Position = [38 813 97 22];
            app.InputDataLabel.Text = '  Input Data  ';

            % Create Panel_7
            app.Panel_7 = uipanel(app.WaveHeightMapTab);
            app.Panel_7.AutoResizeChildren = 'off';
            app.Panel_7.TitlePosition = 'righttop';
            app.Panel_7.FontSize = 10;
            app.Panel_7.Position = [11 315 520 168];

            % Create TabGroup2
            app.TabGroup2 = uitabgroup(app.Panel_7);
            app.TabGroup2.AutoResizeChildren = 'off';
            app.TabGroup2.Position = [0 0 520 153];

            % Create BathymetryTab
            app.BathymetryTab = uitab(app.TabGroup2);
            app.BathymetryTab.AutoResizeChildren = 'off';
            app.BathymetryTab.Title = 'Bathymetry';

            % Create MaximumEditFieldLabel
            app.MaximumEditFieldLabel = uilabel(app.BathymetryTab);
            app.MaximumEditFieldLabel.Enable = 'off';
            app.MaximumEditFieldLabel.Tooltip = {'Higher value indicate deeper↙
waters'};
            app.MaximumEditFieldLabel.Position = [48 25 58 22];
            app.MaximumEditFieldLabel.Text = 'Maximum';

            % Create MaximumEditField
            app.MaximumEditField = uieditfield(app.BathymetryTab, 'numeric');
            app.MaximumEditField.Limits = [1 Inf];
            app.MaximumEditField.ValueDisplayFormat = '%8.1f';
            app.MaximumEditField.Enable = 'off';
            app.MaximumEditField.Tooltip = {''};
            app.MaximumEditField.Position = [107 28 50 16];
            app.MaximumEditField.Value = 1000;

            % Create MinimumEditFieldLabel
            app.MinimumEditFieldLabel = uilabel(app.BathymetryTab);
            app.MinimumEditFieldLabel.Enable = 'off';
            app.MinimumEditFieldLabel.Tooltip = {'0 value = coastline'; '';↙
'Lower values indicate shallower waters.'};
            app.MinimumEditFieldLabel.Position = [48 71 55 22];
            app.MinimumEditFieldLabel.Text = 'Minimum';

            % Create MinimumEditField
            app.MinimumEditField = uieditfield(app.BathymetryTab, 'numeric');
            app.MinimumEditField.Limits = [0 Inf];
            app.MinimumEditField.ValueDisplayFormat = '%8.1f';
            app.MinimumEditField.Enable = 'off';
            app.MinimumEditField.Tooltip = {''};
            app.MinimumEditField.Position = [107 74 50 16];

            % Create IntervalEditField_5Label
            app.IntervalEditField_5Label = uilabel(app.BathymetryTab);
            app.IntervalEditField_5Label.Enable = 'off';
            app.IntervalEditField_5Label.Tooltip = {'Interval between adjacent↙
contours'};
            app.IntervalEditField_5Label.Position = [48 48 45 22];
            app.IntervalEditField_5Label.Text = 'Interval';

            % Create IntervalEditField_5
            app.IntervalEditField_5 = uieditfield(app.BathymetryTab, ↙
'numeric');
            app.IntervalEditField_5.Limits = [0 Inf];
            app.IntervalEditField_5.ValueDisplayFormat = '%8.1f';
            app.IntervalEditField_5.Enable = 'off';
            app.IntervalEditField_5.Tooltip = {''};
            app.IntervalEditField_5.Position = [107 51 40 17];
            app.IntervalEditField_5.Value = 500;

            % Create DepthRangeLabel
            app.DepthRangeLabel = uilabel(app.BathymetryTab);
            app.DepthRangeLabel.FontWeight = 'bold';
            app.DepthRangeLabel.Enable = 'off';
            app.DepthRangeLabel.Position = [48 95 80 22];
            app.DepthRangeLabel.Text = 'Depth Range';

            % Create LabelSpacingEditFieldLabel_2
            app.LabelSpacingEditFieldLabel_2 = uilabel(app.BathymetryTab);
            app.LabelSpacingEditFieldLabel_2.WordWrap = 'on';
            app.LabelSpacingEditFieldLabel_2.Enable = 'off';
            app.LabelSpacingEditFieldLabel_2.Tooltip = {'Adjust the distance between the labels'; ''; 'Higher values reduce label crowding.'; ''; 'Unit: 1 = 1/72 inch'};
            app.LabelSpacingEditFieldLabel_2.Position = [377 25 58 22];
            app.LabelSpacingEditFieldLabel_2.Text = 'Spacing';

            % Create LabelSpacingEditField
            app.LabelSpacingEditField = uieditfield(app.BathymetryTab, 'numeric');
            app.LabelSpacingEditField.Limits = [0 Inf];
            app.LabelSpacingEditField.ValueDisplayFormat = '%8.1f';
            app.LabelSpacingEditField.Enable = 'off';
            app.LabelSpacingEditField.Tooltip = {''};
            app.LabelSpacingEditField.Position = [426 28 47 16];
            app.LabelSpacingEditField.Value = 1000;

            % Create WidthEditFieldLabel
            app.WidthEditFieldLabel = uilabel(app.BathymetryTab);
            app.WidthEditFieldLabel.Enable = 'off';
            app.WidthEditFieldLabel.Tooltip = {'Line thickness'};
            app.WidthEditFieldLabel.Position = [196 71 37 22];
            app.WidthEditFieldLabel.Text = 'Width';

            % Create WidthEditField
            app.WidthEditField = uieditfield(app.BathymetryTab, 'numeric');
            app.WidthEditField.Limits = [0 Inf];
            app.WidthEditField.ValueDisplayFormat = '%3.1f';
            app.WidthEditField.Enable = 'off';
            app.WidthEditField.Tooltip = {''};
            app.WidthEditField.Position = [236 74 35 16];
            app.WidthEditField.Value = 0.1;

            % Create LineLabel
            app.LineLabel = uilabel(app.BathymetryTab);
            app.LineLabel.FontWeight = 'bold';
            app.LineLabel.Enable = 'off';
            app.LineLabel.Position = [196 95 30 22];
            app.LineLabel.Text = 'Line';

            % Create StyleDropDownLabel
            app.StyleDropDownLabel = uilabel(app.BathymetryTab);
            app.StyleDropDownLabel.Enable = 'off';
            app.StyleDropDownLabel.Position = [196 49 32 22];
            app.StyleDropDownLabel.Text = 'Style';

            % Create StyleDropDown
            app.StyleDropDown = uidropdown(app.BathymetryTab);
            app.StyleDropDown.Items = {'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.StyleDropDown.Enable = 'off';
            app.StyleDropDown.Position = [236 52 104 16];
            app.StyleDropDown.Value = 'Solid';

            % Create ColorDropDownLabel
            app.ColorDropDownLabel = uilabel(app.BathymetryTab);
            app.ColorDropDownLabel.Enable = 'off';
            app.ColorDropDownLabel.Position = [196 25 35 22];
            app.ColorDropDownLabel.Text = 'Color';

            % Create ColorDropDown
            app.ColorDropDown = uidropdown(app.BathymetryTab);
            app.ColorDropDown.Items = {'Black', 'Dark gray', 'Medium gray',↙
'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown.Enable = 'off';
            app.ColorDropDown.Position = [236 28 104 16];
            app.ColorDropDown.Value = 'Dark gray';

            % Create AddLabelsCheckBox_2
            app.AddLabelsCheckBox_2 = uicheckbox(app.BathymetryTab);
            app.AddLabelsCheckBox_2.ValueChangedFcn = createCallbackFcn(app, ↙
@AddLabelsCheckBox_2ValueChanged2, true);
            app.AddLabelsCheckBox_2.Enable = 'off';
            app.AddLabelsCheckBox_2.Tooltip = {'Unit: meters'};
            app.AddLabelsCheckBox_2.Text = ' Add Labels';
            app.AddLabelsCheckBox_2.WordWrap = 'on';
            app.AddLabelsCheckBox_2.FontWeight = 'bold';
            app.AddLabelsCheckBox_2.Position = [377 95 93 22];

            % Create TextLabelSize
            app.TextLabelSize = uieditfield(app.BathymetryTab, 'numeric');
            app.TextLabelSize.Limits = [0 30];
            app.TextLabelSize.ValueDisplayFormat = '%8.0f';
            app.TextLabelSize.Enable = 'off';
            app.TextLabelSize.Tooltip = {''};
            app.TextLabelSize.Position = [426 74 30 16];
            app.TextLabelSize.Value = 8;

            % Create SpacingEditField_4Label_4
            app.SpacingEditField_4Label_4 = uilabel(app.BathymetryTab);
            app.SpacingEditField_4Label_4.WordWrap = 'on';
            app.SpacingEditField_4Label_4.Enable = 'off';
            app.SpacingEditField_4Label_4.Position = [377 71 40 22];
            app.SpacingEditField_4Label_4.Text = 'Size';

            % Create LabelSpacingEditFieldLabel
            app.LabelSpacingEditFieldLabel = uilabel(app.BathymetryTab);
            app.LabelSpacingEditFieldLabel.Enable = 'off';
            app.LabelSpacingEditFieldLabel.Tooltip = {'Add labels at specified↙
intervals.'; ''; 'Suggestion: Set at intervals divisible by the depth range↙
interval'};
            app.LabelSpacingEditFieldLabel.Position = [377 49 45 22];
            app.LabelSpacingEditFieldLabel.Text = 'Interval';

            % Create IntervalEditField_3
            app.IntervalEditField_3 = uieditfield(app.BathymetryTab, ↙
'numeric');
            app.IntervalEditField_3.Limits = [0.01 Inf];
```

```
        app.IntervalEditField_3.ValueDisplayFormat = '%8.1f';
        app.IntervalEditField_3.Enable = 'off';
        app.IntervalEditField_3.Tooltip = {''};
        app.IntervalEditField_3.Position = [426 52 47 16];
        app.IntervalEditField_3.Value = 500;

        % Create WaveHeightTab
        app.WaveHeightTab = uitab(app.TabGroup2);
        app.WaveHeightTab.AutoResizeChildren = 'off';
        app.WaveHeightTab.Title = 'Wave Height';

        % Create MinimumdepthEditField_2
        app.MinimumdepthEditField_2 = uieditfield(app.WaveHeightTab, ↙
'numeric');
        app.MinimumdepthEditField_2.ValueDisplayFormat = '%8.1f';
        app.MinimumdepthEditField_2.Tooltip = {'Minimum value of the wave↙
height'};
        app.MinimumdepthEditField_2.Position = [107 74 45 16];

        % Create MaximumdepthEditField_2
        app.MaximumdepthEditField_2 = uieditfield(app.WaveHeightTab, ↙
'numeric');
        app.MaximumdepthEditField_2.Limits = [1 Inf];
        app.MaximumdepthEditField_2.ValueDisplayFormat = '%8.1f';
        app.MaximumdepthEditField_2.Tooltip = {'Maximum value of the wave↙
height'};
        app.MaximumdepthEditField_2.Position = [107 29 45 16];
        app.MaximumdepthEditField_2.Value = 1;

        % Create LineintervalEditField_2
        app.LineintervalEditField_2 = uieditfield(app.WaveHeightTab, ↙
'numeric');
        app.LineintervalEditField_2.Limits = [0 Inf];
        app.LineintervalEditField_2.ValueDisplayFormat = '%8.2f';
        app.LineintervalEditField_2.HorizontalAlignment = 'center';
        app.LineintervalEditField_2.Tooltip = {'Interval between adjacent↙
contours'};
        app.LineintervalEditField_2.Position = [107 51 34 16];
        app.LineintervalEditField_2.Value = 0.5;

        % Create MaximumdepthEditFieldLabel_2
        app.MaximumdepthEditFieldLabel_2 = uilabel(app.WaveHeightTab);
        app.MaximumdepthEditFieldLabel_2.Position = [48 25 53 22];
        app.MaximumdepthEditFieldLabel_2.Text = 'Maximum';

        % Create MinimumdepthEditFieldLabel_2
        app.MinimumdepthEditFieldLabel_2 = uilabel(app.WaveHeightTab);
        app.MinimumdepthEditFieldLabel_2.Position = [48 71 51 22];
        app.MinimumdepthEditFieldLabel_2.Text = 'Minimum';

        % Create MaximumdepthEditFieldLabel_3
        app.MaximumdepthEditFieldLabel_3 = uilabel(app.WaveHeightTab);
        app.MaximumdepthEditFieldLabel_3.Position = [48 48 48 22];
        app.MaximumdepthEditFieldLabel_3.Text = 'Interval';

        % Create ContourRangeLabel_2
        app.ContourRangeLabel_2 = uilabel(app.WaveHeightTab);
        app.ContourRangeLabel_2.FontWeight = 'bold';
```

```
            app.ContourRangeLabel_2.Position = [48 95 92 22];
            app.ContourRangeLabel_2.Text = 'Contour Range';

            % Create LineLabel_2
            app.LineLabel_2 = uilabel(app.WaveHeightTab);
            app.LineLabel_2.FontWeight = 'bold';
            app.LineLabel_2.Position = [196 95 30 22];
            app.LineLabel_2.Text = 'Line';

            % Create ThicknessEditField_2Label
            app.ThicknessEditField_2Label = uilabel(app.WaveHeightTab);
            app.ThicknessEditField_2Label.Tooltip = {'Line thickness'};
            app.ThicknessEditField_2Label.Position = [196 71 37 22];
            app.ThicknessEditField_2Label.Text = 'Width';

            % Create ThicknessEditField_2
            app.ThicknessEditField_2 = uieditfield(app.WaveHeightTab, 'numeric');
            app.ThicknessEditField_2.Limits = [0 20];
            app.ThicknessEditField_2.ValueDisplayFormat = '%3.1f';
            app.ThicknessEditField_2.Position = [236 74 35 16];
            app.ThicknessEditField_2.Value = 0.1;

            % Create ColorDropDown_2Label
            app.ColorDropDown_2Label = uilabel(app.WaveHeightTab);
            app.ColorDropDown_2Label.Position = [196 25 35 22];
            app.ColorDropDown_2Label.Text = 'Color';

            % Create ColorDropDown_2
            app.ColorDropDown_2 = uidropdown(app.WaveHeightTab);
            app.ColorDropDown_2.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown_2.Position = [236 28 104 16];
            app.ColorDropDown_2.Value = 'Dark gray';

            % Create StyleDropDown_2Label
            app.StyleDropDown_2Label = uilabel(app.WaveHeightTab);
            app.StyleDropDown_2Label.Position = [196 49 32 22];
            app.StyleDropDown_2Label.Text = 'Style';

            % Create StyleDropDown_2
            app.StyleDropDown_2 = uidropdown(app.WaveHeightTab);
            app.StyleDropDown_2.Items = {'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.StyleDropDown_2.Position = [236 52 104 16];
            app.StyleDropDown_2.Value = 'Solid';

            % Create AddLabelsCheckBox_3
            app.AddLabelsCheckBox_3 = uicheckbox(app.WaveHeightTab);
            app.AddLabelsCheckBox_3.ValueChangedFcn = createCallbackFcn(app, @AddLabelsCheckBox_3ValueChanged2, true);
            app.AddLabelsCheckBox_3.Tooltip = {'Unit: meters'};
            app.AddLabelsCheckBox_3.Text = ' Add Labels';
            app.AddLabelsCheckBox_3.FontWeight = 'bold';
            app.AddLabelsCheckBox_3.Position = [377 95 90 22];

            % Create TextLabelSize_2
            app.TextLabelSize_2 = uieditfield(app.WaveHeightTab, 'numeric');
            app.TextLabelSize_2.Limits = [0 Inf];
            app.TextLabelSize_2.ValueDisplayFormat = '%8.0f';
            app.TextLabelSize_2.Enable = 'off';
            app.TextLabelSize_2.Position = [426 74 30 16];
            app.TextLabelSize_2.Value = 8;

            % Create SpacingEditField_4Label_5
            app.SpacingEditField_4Label_5 = uilabel(app.WaveHeightTab);
            app.SpacingEditField_4Label_5.WordWrap = 'on';
            app.SpacingEditField_4Label_5.Enable = 'off';
            app.SpacingEditField_4Label_5.Position = [377 71 40 22];
            app.SpacingEditField_4Label_5.Text = 'Size';

            % Create LabelSpacingEditField_2Label_2
            app.LabelSpacingEditField_2Label_2 = uilabel(app.WaveHeightTab);
            app.LabelSpacingEditField_2Label_2.WordWrap = 'on';
            app.LabelSpacingEditField_2Label_2.Enable = 'off';
            app.LabelSpacingEditField_2Label_2.Tooltip = {'Adjust the distance between the labels'; ''; 'Higher values reduce label crowding.'; ''; 'Unit: 1 = 1/72 inch'};
            app.LabelSpacingEditField_2Label_2.Position = [377 27 46 18];
            app.LabelSpacingEditField_2Label_2.Text = 'Spacing';

            % Create LabelSpacingEditField_2
            app.LabelSpacingEditField_2 = uieditfield(app.WaveHeightTab, 'numeric');
            app.LabelSpacingEditField_2.Limits = [0 Inf];
            app.LabelSpacingEditField_2.ValueDisplayFormat = '%8.1f';
            app.LabelSpacingEditField_2.Enable = 'off';
            app.LabelSpacingEditField_2.Tooltip = {''};
            app.LabelSpacingEditField_2.Position = [426 28 47 16];
            app.LabelSpacingEditField_2.Value = 1000;

            % Create LabelSpacingEditField_2Label
            app.LabelSpacingEditField_2Label = uilabel(app.WaveHeightTab);
            app.LabelSpacingEditField_2Label.Enable = 'off';
            app.LabelSpacingEditField_2Label.Tooltip = {'Add labels at specified intervals'; ''; 'Suggestion: Set at intervals divisible by the contour range interval'};
            app.LabelSpacingEditField_2Label.Position = [377 49 45 22];
            app.LabelSpacingEditField_2Label.Text = 'Interval';

            % Create wh_interval
            app.wh_interval = uieditfield(app.WaveHeightTab, 'numeric');
            app.wh_interval.Limits = [0.01 Inf];
            app.wh_interval.ValueDisplayFormat = '%8.1f';
            app.wh_interval.Enable = 'off';
            app.wh_interval.Tooltip = {''};
            app.wh_interval.Position = [426 52 47 16];
            app.wh_interval.Value = 1;

            % Create ArrivalTimeTab
            app.ArrivalTimeTab = uitab(app.TabGroup2);
            app.ArrivalTimeTab.AutoResizeChildren = 'off';
            app.ArrivalTimeTab.Title = 'Arrival Time';
            app.ArrivalTimeTab.ForegroundColor = [0.9412 0.9412 0.9412];

            % Create TimeFileButton
            app.TimeFileButton = uibutton(app.ArrivalTimeTab, 'push');
            app.TimeFileButton.ButtonPushedFcn = createCallbackFcn(app, @TimeFileButtonPushed2, true);
            app.TimeFileButton.Tooltip = {'Load the file.'; ''; 'File extension must be .shp or .txt'; ''; 'Unit: automatically converted to minutes. '; ''; ''; 'The default threshold for detecting the first wave arrival time in FUNWAVE is 0.001 meters.'};
            app.TimeFileButton.Position = [211 105 19 19];
            app.TimeFileButton.Text = '...';

            % Create FileEditField_3Label
            app.FileEditField_3Label = uilabel(app.ArrivalTimeTab);
            app.FileEditField_3Label.FontWeight = 'bold';
            app.FileEditField_3Label.Position = [48 103 29 22];
            app.FileEditField_3Label.Text = 'File:';

            % Create FileEditField_3
            app.FileEditField_3 = uieditfield(app.ArrivalTimeTab, 'text');
            app.FileEditField_3.Editable = 'off';
            app.FileEditField_3.Tooltip = {'Unit: automatically converted to meters '; ''; 'Default threshold for the first wave arrival time in FUNWAVE is 0.001 m.'};
            app.FileEditField_3.Placeholder = 'time_xxxxx';
            app.FileEditField_3.Position = [80 106 125 16];

            % Create ContourRangeLabel
            app.ContourRangeLabel = uilabel(app.ArrivalTimeTab);
            app.ContourRangeLabel.FontWeight = 'bold';
            app.ContourRangeLabel.Tooltip = {'Unit: minutes'};
            app.ContourRangeLabel.Position = [49 78 92 22];
            app.ContourRangeLabel.Text = 'Contour Range';

            % Create MaximumdepthEditField_3
            app.MaximumdepthEditField_3 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.MaximumdepthEditField_3.Limits = [1 Inf];
            app.MaximumdepthEditField_3.ValueDisplayFormat = '%8.1f';
            app.MaximumdepthEditField_3.Tooltip = {'Unit: minutes'};
            app.MaximumdepthEditField_3.Position = [106 18 55 16];
            app.MaximumdepthEditField_3.Value = 1;

            % Create MaximumdepthEditFieldLabel_9
            app.MaximumdepthEditFieldLabel_9 = uilabel(app.ArrivalTimeTab);
            app.MaximumdepthEditFieldLabel_9.Tooltip = {'Unit: minutes'};
            app.MaximumdepthEditFieldLabel_9.Position = [49 15 53 22];
            app.MaximumdepthEditFieldLabel_9.Text = 'Maximum';

            % Create LineintervalEditField_3
            app.LineintervalEditField_3 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.LineintervalEditField_3.Limits = [0 Inf];
            app.LineintervalEditField_3.ValueDisplayFormat = '%8.1f';
            app.LineintervalEditField_3.Tooltip = {'Interval between adjacent contours.'; ''; 'Unit: minutes'};
            app.LineintervalEditField_3.Position = [106 38 55 16];
            app.LineintervalEditField_3.Value = 100;

            % Create MaximumdepthEditFieldLabel_10
            app.MaximumdepthEditFieldLabel_10 = uilabel(app.ArrivalTimeTab);
            app.MaximumdepthEditFieldLabel_10.Tooltip = {'Unit: minutes'};
            app.MaximumdepthEditFieldLabel_10.Position = [49 35 48 22];
            app.MaximumdepthEditFieldLabel_10.Text = 'Interval';

            % Create MinimumdepthEditField_3
            app.MinimumdepthEditField_3 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.MinimumdepthEditField_3.Limits = [0 Inf];
            app.MinimumdepthEditField_3.ValueDisplayFormat = '%8.1f';
            app.MinimumdepthEditField_3.Tooltip = {'Unit: minutes'};
            app.MinimumdepthEditField_3.Position = [106 59 55 16];
            app.MinimumdepthEditField_3.Value = 10;

            % Create MinimumdepthEditFieldLabel_5
            app.MinimumdepthEditFieldLabel_5 = uilabel(app.ArrivalTimeTab);
            app.MinimumdepthEditFieldLabel_5.Tooltip = {'Unit: minutes'};
            app.MinimumdepthEditFieldLabel_5.Position = [49 56 51 22];
            app.MinimumdepthEditFieldLabel_5.Text = 'Minimum';

            % Create LabelSpacingEditField_2Label_4
            app.LabelSpacingEditField_2Label_4 = uilabel(app.ArrivalTimeTab);
            app.LabelSpacingEditField_2Label_4.Enable = 'off';
            app.LabelSpacingEditField_2Label_4.Tooltip = {'Add labels at specified intervals.'; ''; 'Suggestion: Set at intervals divisible by the contour range interval'; ''; 'Unit: meters'};
            app.LabelSpacingEditField_2Label_4.Position = [392 35 45 22];
            app.LabelSpacingEditField_2Label_4.Text = 'Interval';

            % Create IntervalEditField_7
            app.IntervalEditField_7 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.IntervalEditField_7.Limits = [0.01 Inf];
            app.IntervalEditField_7.ValueDisplayFormat = '%8.1f';
            app.IntervalEditField_7.Enable = 'off';
            app.IntervalEditField_7.Tooltip = {''};
            app.IntervalEditField_7.Position = [441 38 41 16];
            app.IntervalEditField_7.Value = 1;

            % Create TextLabelSize_3
            app.TextLabelSize_3 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.TextLabelSize_3.Limits = [0 Inf];
            app.TextLabelSize_3.ValueDisplayFormat = '%8.0f';
            app.TextLabelSize_3.Enable = 'off';
            app.TextLabelSize_3.Tooltip = {''};
            app.TextLabelSize_3.Position = [441 59 30 16];
            app.TextLabelSize_3.Value = 8;

            % Create SpacingEditField_4Label_6
            app.SpacingEditField_4Label_6 = uilabel(app.ArrivalTimeTab);
            app.SpacingEditField_4Label_6.WordWrap = 'on';
            app.SpacingEditField_4Label_6.Enable = 'off';
            app.SpacingEditField_4Label_6.Tooltip = {'Unit: meters'};
            app.SpacingEditField_4Label_6.Position = [392 59 36 17];
            app.SpacingEditField_4Label_6.Text = 'Size';

            % Create AddLabelsCheckBox_5
            app.AddLabelsCheckBox_5 = uicheckbox(app.ArrivalTimeTab);
            app.AddLabelsCheckBox_5.ValueChangedFcn = createCallbackFcn(app, @AddLabelsCheckBox_5ValueChanged2, true);
            app.AddLabelsCheckBox_5.Tooltip = {'Unit: meters'};
            app.AddLabelsCheckBox_5.Text = 'Add Labels';
            app.AddLabelsCheckBox_5.FontWeight = 'bold';
            app.AddLabelsCheckBox_5.Position = [392 78 86 22];

            % Create LineLabel_5
            app.LineLabel_5 = uilabel(app.ArrivalTimeTab);
            app.LineLabel_5.FontWeight = 'bold';
            app.LineLabel_5.Position = [201 78 30 22];
            app.LineLabel_5.Text = 'Line';

            % Create ThicknessEditField_3
            app.ThicknessEditField_3 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.ThicknessEditField_3.Limits = [0 20];
            app.ThicknessEditField_3.ValueDisplayFormat = '%3.1f';
            app.ThicknessEditField_3.Position = [239 59 35 16];
            app.ThicknessEditField_3.Value = 0.1;

            % Create ThicknessEditField_2Label_2
            app.ThicknessEditField_2Label_2 = uilabel(app.ArrivalTimeTab);
            app.ThicknessEditField_2Label_2.Position = [201 56 39 22];
            app.ThicknessEditField_2Label_2.Text = 'Width';

            % Create ColorDropDown_6Label
            app.ColorDropDown_6Label = uilabel(app.ArrivalTimeTab);
            app.ColorDropDown_6Label.Position = [200 15 35 22];
            app.ColorDropDown_6Label.Text = 'Color';

            % Create ColorDropDown_6
            app.ColorDropDown_6 = uidropdown(app.ArrivalTimeTab);
            app.ColorDropDown_6.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown_6.Position = [239 18 104 16];
            app.ColorDropDown_6.Value = 'Dark gray';

            % Create StyleDropDown_4Label_2
            app.StyleDropDown_4Label_2 = uilabel(app.ArrivalTimeTab);
            app.StyleDropDown_4Label_2.Position = [201 35 32 22];
            app.StyleDropDown_4Label_2.Text = 'Style';

            % Create StyleDropDown_4
            app.StyleDropDown_4 = uidropdown(app.ArrivalTimeTab);
            app.StyleDropDown_4.Items = {'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.StyleDropDown_4.Position = [239 38 104 16];
            app.StyleDropDown_4.Value = 'Solid';

            % Create LabelSpacingEditField_4
            app.LabelSpacingEditField_4 = uieditfield(app.ArrivalTimeTab, 'numeric');
            app.LabelSpacingEditField_4.Limits = [0 Inf];
            app.LabelSpacingEditField_4.ValueDisplayFormat = '%8.0f';
            app.LabelSpacingEditField_4.Enable = 'off';
            app.LabelSpacingEditField_4.Tooltip = {''};
            app.LabelSpacingEditField_4.Position = [441 18 41 16];
            app.LabelSpacingEditField_4.Value = 1000;

            % Create LabelSpacingEditField_2Label_3
            app.LabelSpacingEditField_2Label_3 = uilabel(app.ArrivalTimeTab);
            app.LabelSpacingEditField_2Label_3.WordWrap = 'on';
            app.LabelSpacingEditField_2Label_3.Enable = 'off';
            app.LabelSpacingEditField_2Label_3.Tooltip = {'Adjust the distance between the labels'; ''; 'Higher values reduce label crowding.'; ''; 'Unit: 1 = 1/72 inch'};
            app.LabelSpacingEditField_2Label_3.Position = [392 12 45 29];
            app.LabelSpacingEditField_2Label_3.Text = 'Spacing';

            % Create GaugesTab
            app.GaugesTab = uitab(app.TabGroup2);
            app.GaugesTab.AutoResizeChildren = 'off';
            app.GaugesTab.Title = 'Gauges';

            % Create Button_8
            app.Button_8 = uibutton(app.GaugesTab, 'push');
            app.Button_8.ButtonPushedFcn = createCallbackFcn(app, @Button_8Pushed2, true);
            app.Button_8.Tooltip = {'Load the file'; ''; 'File extension must be .shp or .txt (tab delimited)'};
            app.Button_8.Position = [209 96 19 19];
            app.Button_8.Text = '...';

            % Create GaugeFileLabel
            app.GaugeFileLabel = uilabel(app.GaugesTab);
            app.GaugeFileLabel.FontWeight = 'bold';
            app.GaugeFileLabel.Tooltip = {'File extension must be .shp or .txt'};
            app.GaugeFileLabel.Position = [38 94 26 22];
            app.GaugeFileLabel.Text = 'File';

            % Create FileEditField
            app.FileEditField = uieditfield(app.GaugesTab, 'text');
            app.FileEditField.Editable = 'off';
            app.FileEditField.Tooltip = {''};
            app.FileEditField.Placeholder = '.shp or .txt';
            app.FileEditField.Position = [77 97 125 16];

            % Create ColorDropDown_3Label
            app.ColorDropDown_3Label = uilabel(app.GaugesTab);
            app.ColorDropDown_3Label.Tooltip = {'Marker color'};
            app.ColorDropDown_3Label.Position = [38 24 35 22];
            app.ColorDropDown_3Label.Text = 'Color';

            % Create ColorDropDown_3
            app.ColorDropDown_3 = uidropdown(app.GaugesTab);
            app.ColorDropDown_3.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown_3.Position = [77 27 103 16];
            app.ColorDropDown_3.Value = 'Black';

            % Create CoastlinecolourDropDownLabel_3
            app.CoastlinecolourDropDownLabel_3 = uilabel(app.GaugesTab);
            app.CoastlinecolourDropDownLabel_3.Tooltip = {'Marker style'};
            app.CoastlinecolourDropDownLabel_3.Position = [38 46 32 22];
            app.CoastlinecolourDropDownLabel_3.Text = 'Style';

            % Create CoastlinecolourDropDown_3
            app.CoastlinecolourDropDown_3 = uidropdown(app.GaugesTab);
            app.CoastlinecolourDropDown_3.Items = {'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'};
            app.CoastlinecolourDropDown_3.Tooltip = {''};
            app.CoastlinecolourDropDown_3.Position = [79 49 92 16];
            app.CoastlinecolourDropDown_3.Value = 'o';

            % Create SizeEditFieldLabel
            app.SizeEditFieldLabel = uilabel(app.GaugesTab);
            app.SizeEditFieldLabel.Tooltip = {'Marker size'};
            app.SizeEditFieldLabel.Position = [38 69 28 22];
            app.SizeEditFieldLabel.Text = 'Size';

            % Create SizeEditField
            app.SizeEditField = uieditfield(app.GaugesTab, 'numeric');
            app.SizeEditField.Limits = [0 50];
            app.SizeEditField.ValueDisplayFormat = '%3.0f';
            app.SizeEditField.HorizontalAlignment = 'left';
            app.SizeEditField.Tooltip = {''};
            app.SizeEditField.Position = [79 72 40 16];
            app.SizeEditField.Value = 5;

            % Create AddLabelsCheckBox_4
            app.AddLabelsCheckBox_4 = uicheckbox(app.GaugesTab);
            app.AddLabelsCheckBox_4.ValueChangedFcn = createCallbackFcn(app, @AddLabelsCheckBox_4ValueChanged2, true);
            app.AddLabelsCheckBox_4.Tooltip = {'The label follows the sequence of latitude-longitude combinations from the loaded file.'; ''; 'The labels correspond to the numeric values in the sta_ files.'};
            app.AddLabelsCheckBox_4.Text = ' Add Labels';
            app.AddLabelsCheckBox_4.WordWrap = 'on';
            app.AddLabelsCheckBox_4.FontWeight = 'bold';
            app.AddLabelsCheckBox_4.Position = [267 91 96 28];

            % Create AlignmentLabel
            app.AlignmentLabel = uilabel(app.GaugesTab);
            app.AlignmentLabel.Enable = 'off';
            app.AlignmentLabel.Tooltip = {'Horizontal and vertical alignments relative to the marker location'};
            app.AlignmentLabel.Position = [267 24 59 22];
            app.AlignmentLabel.Text = 'Alignment';

            % Create CoastlinecolourDropDown_4
            app.CoastlinecolourDropDown_4 = uidropdown(app.GaugesTab);
            app.CoastlinecolourDropDown_4.Items = {'Centre', 'Right', 'Left'};
            app.CoastlinecolourDropDown_4.Enable = 'off';
            app.CoastlinecolourDropDown_4.Tooltip = {''};
            app.CoastlinecolourDropDown_4.Position = [326 27 74 16];
            app.CoastlinecolourDropDown_4.Value = 'Right';

            % Create CoastlinecolourDropDown_5
            app.CoastlinecolourDropDown_5 = uidropdown(app.GaugesTab);
            app.CoastlinecolourDropDown_5.Items = {'Centre', 'Top', 'Bottom'};
            app.CoastlinecolourDropDown_5.Enable = 'off';
            app.CoastlinecolourDropDown_5.Tooltip = {''};
            app.CoastlinecolourDropDown_5.Position = [406 27 74 16];
            app.CoastlinecolourDropDown_5.Value = 'Centre';

            % Create SizeEditField_3Label
            app.SizeEditField_3Label = uilabel(app.GaugesTab);
            app.SizeEditField_3Label.Enable = 'off';
            app.SizeEditField_3Label.Position = [267 69 30 22];
            app.SizeEditField_3Label.Text = 'Size';

            % Create SizeEditField_3
            app.SizeEditField_3 = uieditfield(app.GaugesTab, 'numeric');
            app.SizeEditField_3.Limits = [0 100];
            app.SizeEditField_3.ValueDisplayFormat = '%8.0f';
            app.SizeEditField_3.Enable = 'off';
            app.SizeEditField_3.Position = [326 72 40 16];
            app.SizeEditField_3.Value = 12;

            % Create GLspacingLabel
            app.GLspacingLabel = uilabel(app.GaugesTab);
            app.GLspacingLabel.Enable = 'off';
            app.GLspacingLabel.Tooltip = {'Horizontal distance between the point and the label.'; ''; ' A value of 0.5 means that a label for a point at 120°E will be placed at 120.5°E.'; ' '; 'Unit: degrees / meters'};
            app.GLspacingLabel.Position = [267 46 50 22];
            app.GLspacingLabel.Text = 'Spacing';

            % Create GLspacing
            app.GLspacing = uieditfield(app.GaugesTab, 'numeric');
            app.GLspacing.ValueDisplayFormat = '%8.3f';
            app.GLspacing.Enable = 'off';
            app.GLspacing.Tooltip = {''};
            app.GLspacing.Position = [326 49 40 16];
            app.GLspacing.Value = 1;

            % Create Tab
            app.Tab = uitab(app.TabGroup2);
            app.Tab.AutoResizeChildren = 'off';
            app.Tab.Title = 'Tab';

            % Create ButtonGroup_17
            app.ButtonGroup_17 = uibuttongroup(app.Panel_7);
            app.ButtonGroup_17.AutoResizeChildren = 'off';
            app.ButtonGroup_17.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_17SelectionChanged, true);
            app.ButtonGroup_17.BorderType = 'none';
            app.ButtonGroup_17.SizeChangedFcn = createCallbackFcn(app, @ButtonGroup_17SizeChanged, true);
            app.ButtonGroup_17.Position = [1 126 518 37];

            % Create BathymetryButton_2
            app.BathymetryButton_2 = uiradiobutton(app.ButtonGroup_17);
            app.BathymetryButton_2.Tooltip = {''};
            app.BathymetryButton_2.Text = 'Bathymetry';
            app.BathymetryButton_2.Position = [44 3 84 22];

            % Create WaveHeightButton
            app.WaveHeightButton = uiradiobutton(app.ButtonGroup_17);
            app.WaveHeightButton.Tooltip = {'This is based on the uploaded
```

```
hmax/eta files'};
            app.WaveHeightButton.Text = 'Wave Height';
            app.WaveHeightButton.Position = [139 3 90 22];

            % Create ArrivalTimeButton
            app.ArrivalTimeButton = uiradiobutton(app.ButtonGroup_17);
            app.ArrivalTimeButton.Tooltip = {'Arrival time threshold set by FUNWAVE: 0.001 m'};
            app.ArrivalTimeButton.Text = 'Arrival Time';
            app.ArrivalTimeButton.Position = [239 3 85 22];

            % Create NoneButton
            app.NoneButton = uiradiobutton(app.ButtonGroup_17);
            app.NoneButton.Text = 'None';
            app.NoneButton.Position = [338 3 51 22];
            app.NoneButton.Value = true;

            % Create GaugesCheckBox
            app.GaugesCheckBox = uicheckbox(app.Panel_7);
            app.GaugesCheckBox.ValueChangedFcn = createCallbackFcn(app, @GaugesCheckBoxValueChanged, true);
            app.GaugesCheckBox.Tooltip = {''};
            app.GaugesCheckBox.Text = 'Gauges';
            app.GaugesCheckBox.Position = [419 129 64 22];

            % Create OverlayFeaturesLabel
            app.OverlayFeaturesLabel = uilabel(app.WaveHeightMapTab);
            app.OverlayFeaturesLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.OverlayFeaturesLabel.FontSize = 15;
            app.OverlayFeaturesLabel.FontWeight = 'bold';
            app.OverlayFeaturesLabel.FontColor = [0.0314 0.3686 0.6];
            app.OverlayFeaturesLabel.Position = [38 471 143 22];
            app.OverlayFeaturesLabel.Text = '  Overlay Features  ';

            % Create Panel_5
            app.Panel_5 = uipanel(app.WaveHeightMapTab);
            app.Panel_5.AutoResizeChildren = 'off';
            app.Panel_5.Position = [11 494 520 157];

            % Create DivisionEditFieldLabel
            app.DivisionEditFieldLabel = uilabel(app.Panel_5);
            app.DivisionEditFieldLabel.FontSize = 11;
            app.DivisionEditFieldLabel.Enable = 'off';
            app.DivisionEditFieldLabel.Tooltip = {'Defines the color distribution:'; ''; 'Low value:'; 'Distinct color boundaries'; ''; 'Higher value:'; 'Smoother color transition'};
            app.DivisionEditFieldLabel.Position = [179 85 44 22];
            app.DivisionEditFieldLabel.Text = 'Division';

            % Create DivisionEditField
            app.DivisionEditField = uieditfield(app.Panel_5, 'numeric');
            app.DivisionEditField.Limits = [0 200];
            app.DivisionEditField.ValueDisplayFormat = '%3.0f';
            app.DivisionEditField.ValueChangedFcn = createCallbackFcn(app, @DivisionEditFieldValueChanged, true);
            app.DivisionEditField.FontColor = [0.651 0.651 0.651];
            app.DivisionEditField.Enable = 'off';
            app.DivisionEditField.Tooltip = {''};
            app.DivisionEditField.Position = [223 88 30 16];
            app.DivisionEditField.Value = 100;

            % Create FlipCheckBox_5
            app.FlipCheckBox_5 = uicheckbox(app.Panel_5);
            app.FlipCheckBox_5.ValueChangedFcn = createCallbackFcn(app, @FlipCheckBox_5ValueChanged, true);
            app.FlipCheckBox_5.Enable = 'off';
            app.FlipCheckBox_5.Tooltip = {'Reverse the color sequence'};
            app.FlipCheckBox_5.Text = 'Flip';
            app.FlipCheckBox_5.FontSize = 11;
            app.FlipCheckBox_5.Position = [112 85 40 22];

            % Create CoastlinecolourDropDown_8
            app.CoastlinecolourDropDown_8 = uidropdown(app.Panel_5);
            app.CoastlinecolourDropDown_8.Items = {'--- MATLAB default ----', 'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', 'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', '--- CBREWER 2 ---', '< sequential >', 'blue', 'blue - green', 'blue - purple', 'green - blue', 'greens', 'grays', 'oranges', 'orange - red', 'purple - blue', 'purple - blue - green', 'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', 'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', '< divergent >', 'brown - teal', 'pink - light green', 'purple - green', 'purple - orange', 'red - blue', 'red - gray', 'red - yellow - blue', 'red - yellow - green', 'spectral', '< qualitative >', 'accent', 'dark 2', 'paired', 'pastel 1', 'pastel 2', 'set 1', 'set 2', 'set 3'};
            app.CoastlinecolourDropDown_8.ValueChangedFcn = createCallbackFcn(app, @CoastlinecolourDropDown_6ValueChanged, true);
            app.CoastlinecolourDropDown_8.Enable = 'off';
            app.CoastlinecolourDropDown_8.Tooltip = {''};
            app.CoastlinecolourDropDown_8.Position = [112 106 141 17];
            app.CoastlinecolourDropDown_8.Value = 'parula';

            % Create CoastlinecolourDropDownLabel_8
            app.CoastlinecolourDropDownLabel_8 = uilabel(app.Panel_5);
            app.CoastlinecolourDropDownLabel_8.FontWeight = 'bold';
            app.CoastlinecolourDropDownLabel_8.Position = [26 104 62 22];
            app.CoastlinecolourDropDownLabel_8.Text = 'Colormap';

            % Create hmaxLabel
            app.hmaxLabel = uilabel(app.Panel_5);
            app.hmaxLabel.WordWrap = 'on';
            app.hmaxLabel.FontWeight = 'bold';
            app.hmaxLabel.FontAngle = 'italic';
            app.hmaxLabel.Tooltip = {'Maximum wave height'};
            app.hmaxLabel.Position = [112 122 60 22];
            app.hmaxLabel.Text = ' hmax';

            % Create MinEditField
            app.MinEditField = uieditfield(app.Panel_5, 'numeric');
            app.MinEditField.Limits = [0 Inf];
            app.MinEditField.ValueDisplayFormat = '%8.1f';
            app.MinEditField.ValueChangedFcn = createCallbackFcn(app, @MinEditFieldValueChanged, true);
            app.MinEditField.FontColor = [0.651 0.651 0.651];
            app.MinEditField.Enable = 'off';
            app.MinEditField.Tooltip = {''};
            app.MinEditField.Position = [112 62 40 16];
```

```matlab

            % Create toEditField
            app.toEditField = uieditfield(app.Panel_5, 'numeric');
            app.toEditField.Limits = [0.1 Inf];
            app.toEditField.ValueDisplayFormat = '%.1f';
            app.toEditField.ValueChangedFcn = createCallbackFcn(app, @toEditFieldValueChanged, true);
            app.toEditField.FontColor = [0.651 0.651 0.651];
            app.toEditField.Enable = 'off';
            app.toEditField.Tooltip = {''};
            app.toEditField.Position = [177 62 40 16];
            app.toEditField.Value = 5;

            % Create CoastlinecolourDropDownLabel_10
            app.CoastlinecolourDropDownLabel_10 = uilabel(app.Panel_5);
            app.CoastlinecolourDropDownLabel_10.WordWrap = 'on';
            app.CoastlinecolourDropDownLabel_10.FontWeight = 'bold';
            app.CoastlinecolourDropDownLabel_10.Tooltip = {'Limit of the ave height values to display on the map'};
            app.CoastlinecolourDropDownLabel_10.Position = [25 55 67 30];
            app.CoastlinecolourDropDownLabel_10.Text = 'Colorbar';

            % Create toLabel
            app.toLabel = uilabel(app.Panel_5);
            app.toLabel.Position = [159 59 16 22];
            app.toLabel.Text = 'to';

            % Create etaLabel
            app.etaLabel = uilabel(app.Panel_5);
            app.etaLabel.WordWrap = 'on';
            app.etaLabel.FontWeight = 'bold';
            app.etaLabel.FontAngle = 'italic';
            app.etaLabel.Tooltip = {'Sea surface displacement at time T'};
            app.etaLabel.Position = [311 122 57 22];
            app.etaLabel.Text = 'eta';

            % Create CoastlinecolourDropDown_7
            app.CoastlinecolourDropDown_7 = uidropdown(app.Panel_5);
            app.CoastlinecolourDropDown_7.Items = {'--- MATLAB default ----', 'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', 'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', '--- CBREWER 2 ---', '< sequential >', 'blue', 'blue - green', 'blue - purple', 'green - blue', 'greens', 'grays', 'oranges', 'orange - red', 'purple - blue', 'purple - blue - green', 'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', 'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', '< divergent >', 'brown - teal', 'pink - light green', 'purple - green', 'purple - orange', 'red - blue', 'red - gray', 'red - yellow - blue', 'red - yellow - green', 'spectral', '< qualitative >', 'accent', 'dark 2', 'paired', 'pastel 1', 'pastel 2', 'set 1', 'set 2', 'set 3'};
            app.CoastlinecolourDropDown_7.ValueChangedFcn = createCallbackFcn(app, @CoastlinecolourDropDown_7ValueChanged, true);
            app.CoastlinecolourDropDown_7.Enable = 'off';
            app.CoastlinecolourDropDown_7.Tooltip = {''};
            app.CoastlinecolourDropDown_7.Position = [311 106 141 17];
            app.CoastlinecolourDropDown_7.Value = 'red - blue';

            % Create FlipCheckBox_2
            app.FlipCheckBox_2 = uicheckbox(app.Panel_5);
            app.FlipCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @FlipCheckBox_2ValueChanged, true);
            app.FlipCheckBox_2.Enable = 'off';
            app.FlipCheckBox_2.Tooltip = {'Reverse the color sequence'};
            app.FlipCheckBox_2.Text = 'Flip';
            app.FlipCheckBox_2.FontSize = 11;
            app.FlipCheckBox_2.Position = [311 85 40 22];
            app.FlipCheckBox_2.Value = true;

            % Create ColorinterpolationEditField_2
            app.ColorinterpolationEditField_2 = uieditfield(app.Panel_5, 'numeric');
            app.ColorinterpolationEditField_2.Limits = [0 200];
            app.ColorinterpolationEditField_2.ValueDisplayFormat = '%3.0f';
            app.ColorinterpolationEditField_2.ValueChangedFcn = createCallbackFcn(app, @ColorinterpolationEditField_2ValueChanged, true);
            app.ColorinterpolationEditField_2.FontColor = [0.651 0.651 0.651];
            app.ColorinterpolationEditField_2.Enable = 'off';
            app.ColorinterpolationEditField_2.Tooltip = {''};
            app.ColorinterpolationEditField_2.Position = [422 88 30 16];
            app.ColorinterpolationEditField_2.Value = 100;

            % Create DivisionLabel
            app.DivisionLabel = uilabel(app.Panel_5);
            app.DivisionLabel.FontSize = 11;
            app.DivisionLabel.Enable = 'off';
            app.DivisionLabel.Tooltip = {'Defines the color distribution:'; ''; 'Low value:'; 'Distinct color boundaries'; ''; 'Higher value:'; 'Smoother color transition'};
            app.DivisionLabel.Position = [377 85 44 22];
            app.DivisionLabel.Text = 'Division';

            % Create MaxEditField_2
            app.MaxEditField_2 = uieditfield(app.Panel_5, 'numeric');
            app.MaxEditField_2.Limits = [0 Inf];
            app.MaxEditField_2.ValueDisplayFormat = '%.1f';
            app.MaxEditField_2.ValueChangedFcn = createCallbackFcn(app, @MaxEditField_2ValueChanged, true);
            app.MaxEditField_2.FontColor = [0.651 0.651 0.651];
            app.MaxEditField_2.Enable = 'off';
            app.MaxEditField_2.Tooltip = {'Suggestion: '; 'Set this value as the (+) of the min value to place the zero mark in the middle of the colorbar'};
            app.MaxEditField_2.Position = [376 62 40 16];
            app.MaxEditField_2.Value = 0.5;

            % Create MinEditField_2
            app.MinEditField_2 = uieditfield(app.Panel_5, 'numeric');
            app.MinEditField_2.ValueDisplayFormat = '%8.1f';
            app.MinEditField_2.ValueChangedFcn = createCallbackFcn(app, @MinEditField_2ValueChanged, true);
            app.MinEditField_2.FontColor = [0.651 0.651 0.651];
            app.MinEditField_2.Enable = 'off';
            app.MinEditField_2.Tooltip = {'Suggestion: '; 'Set this value as the (-) of the max value to place the zero mark in the middle of the colorbar'};
            app.MinEditField_2.Position = [311 62 41 16];
            app.MinEditField_2.Value = -0.5;

            % Create toLabel_2
            app.toLabel_2 = uilabel(app.Panel_5);
            app.toLabel_2.Position = [358 59 16 22];
            app.toLabel_2.Text = 'to';

            % Create mLabel
            app.mLabel = uilabel(app.Panel_5);
            app.mLabel.Position = [223 59 25 22];
            app.mLabel.Text = 'm';

            % Create mLabel_2
            app.mLabel_2 = uilabel(app.Panel_5);
            app.mLabel_2.Position = [423 59 25 22];
            app.mLabel_2.Text = 'm';

            % Create CoastlinecolourDropDownLabel_9
            app.CoastlinecolourDropDownLabel_9 = uilabel(app.Panel_5);
            app.CoastlinecolourDropDownLabel_9.FontWeight = 'bold';
            app.CoastlinecolourDropDownLabel_9.Tooltip = {'Required input:'; 'Bathymetry file'};
            app.CoastlinecolourDropDownLabel_9.Position = [307 28 69 22];
            app.CoastlinecolourDropDownLabel_9.Text = 'Land Color';

            % Create HorizontalCheckBox
            app.HorizontalCheckBox = uicheckbox(app.Panel_5);
            app.HorizontalCheckBox.Text = 'Horizontal';
            app.HorizontalCheckBox.Position = [119 17 76 22];

            % Create VerticalCheckBox
            app.VerticalCheckBox = uicheckbox(app.Panel_5);
            app.VerticalCheckBox.Text = 'Vertical';
            app.VerticalCheckBox.Position = [199 17 61 22];

            % Create FlipBasemapLabel_2
            app.FlipBasemapLabel_2 = uilabel(app.Panel_5);
            app.FlipBasemapLabel_2.FontWeight = 'bold';
            app.FlipBasemapLabel_2.Tooltip = {'Flips the basemap.'; ''; 'Horizontally (left-right)'; 'Vertically (up-side down)'};
            app.FlipBasemapLabel_2.Position = [25 17 87 22];
            app.FlipBasemapLabel_2.Text = 'Flip Basemap';

            % Create CoastlinecolourDropDown_2
            app.CoastlinecolourDropDown_2 = uidropdown(app.Panel_5);
            app.CoastlinecolourDropDown_2.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.CoastlinecolourDropDown_2.ValueChangedFcn = createCallbackFcn(app, @CoastlinecolourDropDown_2ValueChanged, true);
            app.CoastlinecolourDropDown_2.Enable = 'off';
            app.CoastlinecolourDropDown_2.Tooltip = {''};
            app.CoastlinecolourDropDown_2.Position = [381 31 106 17];
            app.CoastlinecolourDropDown_2.Value = 'Dark gray';

            % Create TextSizeLabel_2
            app.TextSizeLabel_2 = uilabel(app.Panel_5);
            app.TextSizeLabel_2.WordWrap = 'on';
            app.TextSizeLabel_2.FontWeight = 'bold';
            app.TextSizeLabel_2.Tooltip = {'Font size of the colorbar labels'};
            app.TextSizeLabel_2.Position = [307 3 60 28];
            app.TextSizeLabel_2.Text = 'Text Size';

            % Create ColorbarTextSize
            app.ColorbarTextSize = uieditfield(app.Panel_5, 'numeric');
            app.ColorbarTextSize.Limits = [1 50];
            app.ColorbarTextSize.ValueDisplayFormat = '%2.0f';
            app.ColorbarTextSize.ValueChangedFcn = createCallbackFcn(app, @ColorbarTextSizeValueChanged, true);
            app.ColorbarTextSize.FontColor = [0.651 0.651 0.651];
            app.ColorbarTextSize.Enable = 'off';
            app.ColorbarTextSize.Tooltip = {''};
            app.ColorbarTextSize.Position = [381 9 29 16];
            app.ColorbarTextSize.Value = 11;

            % Create BasemapLabel
            app.BasemapLabel = uilabel(app.WaveHeightMapTab);
            app.BasemapLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.BasemapLabel.FontSize = 15;
            app.BasemapLabel.FontWeight = 'bold';
            app.BasemapLabel.FontColor = [0.0314 0.3686 0.6];
            app.BasemapLabel.Tooltip = {'Modify the visuals of the imported file/s (eta/hmax)'};
            app.BasemapLabel.Position = [38 639 89 22];
            app.BasemapLabel.Text = '  Basemap  ';

            % Create Panel_6
            app.Panel_6 = uipanel(app.WaveHeightMapTab);
            app.Panel_6.AutoResizeChildren = 'off';
            app.Panel_6.Position = [11 39 520 77];

            % Create pngCheckBox
            app.pngCheckBox = uicheckbox(app.Panel_6);
            app.pngCheckBox.Tooltip = {'Resolution is set to 300 DPI'};
            app.pngCheckBox.Text = '.png';
            app.pngCheckBox.Position = [191 12 46 22];

            % Create FileFormatLabel
            app.FileFormatLabel = uilabel(app.Panel_6);
            app.FileFormatLabel.Position = [69 12 66 22];
            app.FileFormatLabel.Text = 'File Format';

            % Create jpgCheckBox
            app.jpgCheckBox = uicheckbox(app.Panel_6);
            app.jpgCheckBox.Tooltip = {'Resolution is set to 300 DPI'};
            app.jpgCheckBox.Text = '.jpg';
            app.jpgCheckBox.Position = [139 12 42 22];

            % Create tifCheckBox_2
            app.tifCheckBox_2 = uicheckbox(app.Panel_6);
            app.tifCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @tifCheckBox_2ValueChanged, true);
            app.tifCheckBox_2.Enable = 'off';
            app.tifCheckBox_2.Tooltip = {'Georeferenced raster file.'; ''; 'Enabled when "Plot separately" is selected.'};
            app.tifCheckBox_2.Text = '.tif';
            app.tifCheckBox_2.Position = [246 12 35 22];

            % Create Button_2
            app.Button_2 = uibutton(app.Panel_6, 'push');
            app.Button_2.ButtonPushedFcn = createCallbackFcn(app, @Button_2Pushed, true);
            app.Button_2.FontSize = 10;
            app.Button_2.FontAngle = 'italic';
            app.Button_2.Tooltip = {'Load the preferred directory.'; ''; '"OUTPUT_FILES/Figures"  folder will be created in the selected directory.'};
            app.Button_2.Position = [403 42 19 19];
            app.Button_2.Text = '...';

            % Create OutputDirectoryLabel
            app.OutputDirectoryLabel = uilabel(app.Panel_6);
            app.OutputDirectoryLabel.Tooltip = {'Load the preferred directory.'; ''; '''OUTPUT_FILES/Figures'' folder will be created in the selected directory.'};
            app.OutputDirectoryLabel.Position = [40 40 94 22];
            app.OutputDirectoryLabel.Text = 'Output Directory';

            % Create OutputDirectoryEditField
            app.OutputDirectoryEditField = uieditfield(app.Panel_6, 'text');
            app.OutputDirectoryEditField.Editable = 'off';
            app.OutputDirectoryEditField.FontColor = [0 0 1];
            app.OutputDirectoryEditField.Tooltip = {'Load the preferred directory.'; ''; '''OUTPUT_FILES/Figures'' folder will be created in the selected directory.'};
            app.OutputDirectoryEditField.Placeholder = 'Default: Desktop';
            app.OutputDirectoryEditField.Position = [139 43 257 16];

            % Create FramerateEditField_2Label
            app.FramerateEditField_2Label = uilabel(app.Panel_6);
            app.FramerateEditField_2Label.WordWrap = 'on';
            app.FramerateEditField_2Label.Visible = 'off';
            app.FramerateEditField_2Label.Tooltip = {'Frames per second.'; ''; 'Lower value = slower animation.'};
            app.FramerateEditField_2Label.Position = [346 10 68 27];
            app.FramerateEditField_2Label.Text = 'Frame rate';

            % Create FramerateEditField_2
            app.FramerateEditField_2 = uieditfield(app.Panel_6, 'numeric');
            app.FramerateEditField_2.Limits = [0 100];
            app.FramerateEditField_2.ValueDisplayFormat = '%.0f';
            app.FramerateEditField_2.FontColor = [0.651 0.651 0.651];
            app.FramerateEditField_2.Visible = 'off';
            app.FramerateEditField_2.Tooltip = {''};
            app.FramerateEditField_2.Position = [412 15 30 17];
            app.FramerateEditField_2.Value = 5;

            % Create mp4CheckBox
            app.mp4CheckBox = uicheckbox(app.Panel_6);
            app.mp4CheckBox.ValueChangedFcn = createCallbackFcn(app, @mp4CheckBoxValueChanged, true);
            app.mp4CheckBox.Enable = 'off';
            app.mp4CheckBox.Tooltip = {'Create an animation.'; ''; 'File is saved as "animation.mp4."'; ''; 'Enabled when "Plot separately" is selected.'};
            app.mp4CheckBox.Text = '.mp4';
            app.mp4CheckBox.Position = [290 12 49 22];

            % Create SaveMapLabel
            app.SaveMapLabel = uilabel(app.WaveHeightMapTab);
            app.SaveMapLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.SaveMapLabel.FontSize = 15;
            app.SaveMapLabel.FontWeight = 'bold';
            app.SaveMapLabel.FontColor = [0.0314 0.3686 0.6];
            app.SaveMapLabel.Position = [38 105 92 22];
            app.SaveMapLabel.Text = '  Save Map  ';

            % Create GENERATEButton
            app.GENERATEButton = uibutton(app.WaveHeightMapTab, 'push');
            app.GENERATEButton.ButtonPushedFcn = createCallbackFcn(app, @GENERATEButtonPushed, true);
            app.GENERATEButton.FontSize = 14;
            app.GENERATEButton.FontWeight = 'bold';
            app.GENERATEButton.Position = [179 6 183 25];
            app.GENERATEButton.Text = 'GENERATE';

            % Create Panel_19
            app.Panel_19 = uipanel(app.WaveHeightMapTab);
            app.Panel_19.AutoResizeChildren = 'off';
            app.Panel_19.Position = [11 128 520 172];

            % Create BoundaryLimitsLabel
            app.BoundaryLimitsLabel = uilabel(app.Panel_19);
            app.BoundaryLimitsLabel.FontWeight = 'bold';
            app.BoundaryLimitsLabel.Tooltip = {'When all textboxes are left unchanged (all zeroes), the values are automatically set based on the extent of the uploaded file and the input southwest corner coordinates'};
            app.BoundaryLimitsLabel.Position = [25 91 99 22];
            app.BoundaryLimitsLabel.Text = 'Boundary Limits';

            % Create PlotindegreesCheckBox_2
            app.PlotindegreesCheckBox_2 = uicheckbox(app.Panel_19);
            app.PlotindegreesCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @PlotindegreesCheckBox_2ValueChanged, true);
            app.PlotindegreesCheckBox_2.Tooltip = {'Change the label format into geographic coordinates'};
            app.PlotindegreesCheckBox_2.Text = 'Plot in degrees';
            app.PlotindegreesCheckBox_2.FontSize = 11;
            app.PlotindegreesCheckBox_2.Position = [403 56 96 22];

            % Create EastEditFieldLabel
            app.EastEditFieldLabel = uilabel(app.Panel_19);
            app.EastEditFieldLabel.Position = [331 62 29 22];
            app.EastEditFieldLabel.Text = 'East';

            % Create EastEditField
            app.EastEditField = uieditfield(app.Panel_19, 'numeric');
            app.EastEditField.ValueDisplayFormat = '%8.4f';
            app.EastEditField.ValueChangedFcn = createCallbackFcn(app, @EastEditFieldValueChanged, true);
            app.EastEditField.FontColor = [0.651 0.651 0.651];
            app.EastEditField.Position = [265 65 60 16];

            % Create WestEditFieldLabel
            app.WestEditFieldLabel = uilabel(app.Panel_19);
            app.WestEditFieldLabel.HorizontalAlignment = 'right';
            app.WestEditFieldLabel.Position = [87 62 32 22];
            app.WestEditFieldLabel.Text = 'West';

            % Create WestEditField
            app.WestEditField = uieditfield(app.Panel_19, 'numeric');
            app.WestEditField.ValueDisplayFormat = '%8.4f';
            app.WestEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@WestEditFieldValueChanged, true);
            app.WestEditField.FontColor = [0.651 0.651 0.651];
            app.WestEditField.Position = [125 65 60 16];

            % Create NorthEditFieldLabel
            app.NorthEditFieldLabel = uilabel(app.Panel_19);
            app.NorthEditFieldLabel.HorizontalAlignment = 'right';
            app.NorthEditFieldLabel.Position = [206 91 35 22];
            app.NorthEditFieldLabel.Text = 'North';

            % Create NorthEditField
            app.NorthEditField = uieditfield(app.Panel_19, 'numeric');
            app.NorthEditField.ValueDisplayFormat = '%8.4f';
            app.NorthEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@NorthEditFieldValueChanged, true);
            app.NorthEditField.FontColor = [0.651 0.651 0.651];
            app.NorthEditField.Position = [195 77 60 16];

            % Create SouthEditFieldLabel
            app.SouthEditFieldLabel = uilabel(app.Panel_19);
            app.SouthEditFieldLabel.HorizontalAlignment = 'right';
            app.SouthEditFieldLabel.Position = [205 34 37 22];
            app.SouthEditFieldLabel.Text = 'South';

            % Create SouthEditField
            app.SouthEditField = uieditfield(app.Panel_19, 'numeric');
            app.SouthEditField.ValueDisplayFormat = '%8.4f';
            app.SouthEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@SouthEditFieldValueChanged, true);
            app.SouthEditField.FontColor = [0.651 0.651 0.651];
            app.SouthEditField.Position = [195 53 60 16];

            % Create Height_2
            app.Height_2 = uieditfield(app.Panel_19, 'numeric');
            app.Height_2.Limits = [1 Inf];
            app.Height_2.ValueDisplayFormat = '%2.0f';
            app.Height_2.ValueChangedFcn = createCallbackFcn(app, ↙
@Height_2ValueChanged, true);
            app.Height_2.FontColor = [0.651 0.651 0.651];
            app.Height_2.Enable = 'off';
            app.Height_2.Tooltip = {''};
            app.Height_2.Position = [330 12 40 16];
            app.Height_2.Value = 11;

            % Create ThicknessLabel_8
            app.ThicknessLabel_8 = uilabel(app.Panel_19);
            app.ThicknessLabel_8.HorizontalAlignment = 'right';
            app.ThicknessLabel_8.Position = [283 9 41 22];
            app.ThicknessLabel_8.Text = 'Height';

            % Create AutoSetCheckBox_2
            app.AutoSetCheckBox_2 = uicheckbox(app.Panel_19);
            app.AutoSetCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @AutoSetCheckBox_2ValueChanged, true);
            app.AutoSetCheckBox_2.Text = ' Auto Set';
            app.AutoSetCheckBox_2.FontSize = 11;
            app.AutoSetCheckBox_2.Position = [102 9 68 22];
            app.AutoSetCheckBox_2.Value = true;

            % Create Width_2
            app.Width_2 = uieditfield(app.Panel_19, 'numeric');
            app.Width_2.Limits = [1 Inf];
            app.Width_2.ValueDisplayFormat = '%2.0f';
            app.Width_2.ValueChangedFcn = createCallbackFcn(app, @Width_2ValueChanged, true);
            app.Width_2.FontColor = [0.651 0.651 0.651];
            app.Width_2.Enable = 'off';
            app.Width_2.Tooltip = {''};
            app.Width_2.Position = [225 12 41 16];
            app.Width_2.Value = 8;

            % Create ThicknessLabel_9
            app.ThicknessLabel_9 = uilabel(app.Panel_19);
            app.ThicknessLabel_9.Tooltip = {'Unit: inches'};
            app.ThicknessLabel_9.Position = [188 9 37 22];
            app.ThicknessLabel_9.Text = 'Width';

            % Create FigureSizeLabel_2
            app.FigureSizeLabel_2 = uilabel(app.Panel_19);
            app.FigureSizeLabel_2.FontWeight = 'bold';
            app.FigureSizeLabel_2.Position = [25 9 69 22];
            app.FigureSizeLabel_2.Text = 'Figure Size';

            % Create CloseFiguresButton_4
            app.CloseFiguresButton_4 = uibutton(app.Panel_19, 'push');
            app.CloseFiguresButton_4.ButtonPushedFcn = createCallbackFcn(app, @CloseFiguresButton_4Pushed2, true);
            app.CloseFiguresButton_4.Tooltip = {''};
            app.CloseFiguresButton_4.Position = [403 10 89 20];
            app.CloseFiguresButton_4.Text = 'Close Figures';

            % Create SettoDefaultButton_2
            app.SettoDefaultButton_2 = uibutton(app.Panel_19, 'push');
            app.SettoDefaultButton_2.ButtonPushedFcn = createCallbackFcn(app, @SettoDefaultButton_2Pushed, true);
            app.SettoDefaultButton_2.Tooltip = {'Use the boundary limits of the eta/hmax file'};
            app.SettoDefaultButton_2.Position = [403 76 84 20];
            app.SettoDefaultButton_2.Text = 'Set to Default';

            % Create ButtonGroup_11
            app.ButtonGroup_11 = uibuttongroup(app.Panel_19);
            app.ButtonGroup_11.AutoResizeChildren = 'off';
            app.ButtonGroup_11.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_11SelectionChanged, true);
            app.ButtonGroup_11.Position = [106 126 311 30];

            % Create PlotalldatainonefigureButton
            app.PlotalldatainonefigureButton = uiradiobutton(app.
ButtonGroup_11);
            app.PlotalldatainonefigureButton.Tooltip = {'All maps are displayed as subplots in one window'};
            app.PlotalldatainonefigureButton.Text = 'Plot all data in one figure';
            app.PlotalldatainonefigureButton.FontWeight = 'bold';
            app.PlotalldatainonefigureButton.Position = [13 4 164 22];
            app.PlotalldatainonefigureButton.Value = true;

            % Create PlotseparatelyButton_2
            app.PlotseparatelyButton_2 = uiradiobutton(app.ButtonGroup_11);
            app.PlotseparatelyButton_2.Tooltip = {'One map per window'};
            app.PlotseparatelyButton_2.Text = 'Plot separately';
            app.PlotseparatelyButton_2.FontWeight = 'bold';
            app.PlotseparatelyButton_2.Position = [193 4 108 22];

            % Create GeneralLayoutLabel_3
            app.GeneralLayoutLabel_3 = uilabel(app.WaveHeightMapTab);
            app.GeneralLayoutLabel_3.BackgroundColor = [0.9412 0.9412 0.9412];
            app.GeneralLayoutLabel_3.FontSize = 15;
            app.GeneralLayoutLabel_3.FontWeight = 'bold';
            app.GeneralLayoutLabel_3.FontColor = [0.0314 0.3686 0.6];
            app.GeneralLayoutLabel_3.Position = [38 290 131 22];
            app.GeneralLayoutLabel_3.Text = '  General Layout  ';

            % Create GaugeRecordsTab
            app.GaugeRecordsTab = uitab(app.TabGroup);
            app.GaugeRecordsTab.AutoResizeChildren = 'off';
            app.GaugeRecordsTab.Tooltip = {'When ''3 & 4'' is used for Y data, default values:'; ''; 'Column 3: Dashed-dotted lines'; 'Column 4: Dotted lines'};
            app.GaugeRecordsTab.SizeChangedFcn = createCallbackFcn(app, @GaugeRecordsTabSizeChanged, true);
            app.GaugeRecordsTab.Title = 'Gauge Records';
            app.GaugeRecordsTab.Scrollable = 'on';

            % Create Panel_10
            app.Panel_10 = uipanel(app.GaugeRecordsTab);
            app.Panel_10.AutoResizeChildren = 'off';
            app.Panel_10.Interruptible = 'off';
            app.Panel_10.Position = [11 129 520 562];

            % Create PlotPropertiesLabel
            app.PlotPropertiesLabel = uilabel(app.Panel_10);
            app.PlotPropertiesLabel.FontWeight = 'bold';
            app.PlotPropertiesLabel.Position = [50 179 91 22];
            app.PlotPropertiesLabel.Text = 'Plot Properties';

            % Create ColorDropDownLabel_2
            app.ColorDropDownLabel_2 = uilabel(app.Panel_10);
            app.ColorDropDownLabel_2.Tooltip = {''; 'When column ''3,4,5'''' is selected for Y axis, the default colors are: '; ''; '3 - purple'; '4 - green'; '5 - black'};
            app.ColorDropDownLabel_2.Position = [50 85 35 22];
            app.ColorDropDownLabel_2.Text = 'Color';

            % Create GaugeLineColor
            app.GaugeLineColor = uidropdown(app.Panel_10);
            app.GaugeLineColor.Items = {'Black', 'Dark gray', 'Medium gray',
'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta'};
            app.GaugeLineColor.Visible = 'off';
            app.GaugeLineColor.Position = [108 87 89 17];
            app.GaugeLineColor.Value = 'Black';

            % Create LineStyleLabel
            app.LineStyleLabel = uilabel(app.Panel_10);
            app.LineStyleLabel.Tooltip = {'Line style of the plotted data.'; ''; 'When column ''3,4,5'''' is selected for Y axis, the default styles are:'; ''; '3, 4 - dashed lines'; '5 - solid line'};
            app.LineStyleLabel.Position = [50 131 28 22];
            app.LineStyleLabel.Text = 'Line';

            % Create GaugeLineStyle
            app.GaugeLineStyle = uidropdown(app.Panel_10);
            app.GaugeLineStyle.Items = {'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.GaugeLineStyle.Tooltip = {''};
            app.GaugeLineStyle.Position = [107 134 91 17];
            app.GaugeLineStyle.Value = 'Solid';

            % Create ButtonGroup_10
            app.ButtonGroup_10 = uibuttongroup(app.Panel_10);
            app.ButtonGroup_10.AutoResizeChildren = 'off';
            app.ButtonGroup_10.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_10SelectionChanged, true);
            app.ButtonGroup_10.FontWeight = 'bold';
            app.ButtonGroup_10.Position = [104 510 311 30];

            % Create PlotalldatainonegraphButton
            app.PlotalldatainonegraphButton = uiradiobutton(app.ButtonGroup_10);
            app.PlotalldatainonegraphButton.Tooltip = {'Alll data are summarized in one graph'};
            app.PlotalldatainonegraphButton.Text = 'Plot all data in one graph';
            app.PlotalldatainonegraphButton.FontWeight = 'bold';
            app.PlotalldatainonegraphButton.Position = [13 4 165 22];
            app.PlotalldatainonegraphButton.Value = true;

            % Create PlotseparatelyButton_4
            app.PlotseparatelyButton_4 = uiradiobutton(app.ButtonGroup_10);
            app.PlotseparatelyButton_4.Tooltip = {'Data from each ''sta_'' file is plotted in separate windows'};
            app.PlotseparatelyButton_4.Text = 'Plot separately';
            app.PlotseparatelyButton_4.FontWeight = 'bold';
            app.PlotseparatelyButton_4.Position = [195 4 108 22];

            % Create DataLineWidth
            app.DataLineWidth = uieditfield(app.Panel_10, 'numeric');
            app.DataLineWidth.Limits = [0.1 Inf];
            app.DataLineWidth.ValueDisplayFormat = '%8.1f';
            app.DataLineWidth.ValueChangedFcn = createCallbackFcn(app, @DataLineWidthValueChanged, true);
            app.DataLineWidth.FontColor = [0.651 0.651 0.651];
            app.DataLineWidth.Tooltip = {''};
            app.DataLineWidth.Position = [107 111 29 16];
            app.DataLineWidth.Value = 1;

            % Create WidthLabel
            app.WidthLabel = uilabel(app.Panel_10);
            app.WidthLabel.Position = [50 108 37 22];
            app.WidthLabel.Text = 'Width';

            % Create AxesLabels
            app.AxesLabels = uilabel(app.Panel_10);
            app.AxesLabels.BackgroundColor = [0.9412 0.9412 0.9412];
            app.AxesLabels.FontWeight = 'bold';
            app.AxesLabels.Position = [50 297 75 22];
            app.AxesLabels.Text = 'Axes Labels';

            % Create TextArea2_4
            app.TextArea2_4 = uitextarea(app.Panel_10);
            app.TextArea2_4.Editable = 'off';
            app.TextArea2_4.BackgroundColor = [0.9412 0.9412 0.9412];
            app.TextArea2_4.Enable = 'off';
            app.TextArea2_4.Position = [173 356 222 109];

            % Create xMinLimit
            app.xMinLimit = uieditfield(app.Panel_10, 'numeric');
            app.xMinLimit.ValueDisplayFormat = '%8.1f';
            app.xMinLimit.ValueChangedFcn = createCallbackFcn(app, ↙
@xMinLimitValueChanged, true);
            app.xMinLimit.FontColor = [0.651 0.651 0.651];
            app.xMinLimit.Enable = 'off';
            app.xMinLimit.Tooltip = {'X-axis minimum value'};
            app.xMinLimit.Position = [217 356 45 16];

            % Create MinimumdepthEditFieldLabel_4
            app.MinimumdepthEditFieldLabel_4 = uilabel(app.Panel_10);
            app.MinimumdepthEditFieldLabel_4.HorizontalAlignment = 'center';
            app.MinimumdepthEditFieldLabel_4.Position = [213 336 51 22];
            app.MinimumdepthEditFieldLabel_4.Text = 'X min';

            % Create xMaxLimit
            app.xMaxLimit = uieditfield(app.Panel_10, 'numeric');
            app.xMaxLimit.ValueDisplayFormat = '%8.1f';
            app.xMaxLimit.ValueChangedFcn = createCallbackFcn(app, ↙
@xMaxLimitValueChanged, true);
            app.xMaxLimit.FontColor = [0.651 0.651 0.651];
            app.xMaxLimit.Enable = 'off';
            app.xMaxLimit.Tooltip = {'X-axis maximum value'};
            app.xMaxLimit.Position = [340 356 55 16];
            app.xMaxLimit.Value = 3600;

            % Create MaximumdepthEditFieldLabel_5
            app.MaximumdepthEditFieldLabel_5 = uilabel(app.Panel_10);
            app.MaximumdepthEditFieldLabel_5.HorizontalAlignment = 'center';
            app.MaximumdepthEditFieldLabel_5.Position = [343 336 53 22];
            app.MaximumdepthEditFieldLabel_5.Text = 'X max';

            % Create xLimInterval
            app.xLimInterval = uieditfield(app.Panel_10, 'numeric');
            app.xLimInterval.Limits = [0 Inf];
            app.xLimInterval.ValueDisplayFormat = '%8.1f';
            app.xLimInterval.ValueChangedFcn = createCallbackFcn(app, ↙
@xLimIntervalValueChanged, true);
            app.xLimInterval.FontColor = [0.651 0.651 0.651];
            app.xLimInterval.Enable = 'off';
            app.xLimInterval.Tooltip = {'X-axis tick interval'};
            app.xLimInterval.Position = [275 356 51 16];
            app.xLimInterval.Value = 1000;

            % Create MaximumdepthEditFieldLabel_6
            app.MaximumdepthEditFieldLabel_6 = uilabel(app.Panel_10);
            app.MaximumdepthEditFieldLabel_6.HorizontalAlignment = 'center';
            app.MaximumdepthEditFieldLabel_6.Position = [278 336 45 22];
            app.MaximumdepthEditFieldLabel_6.Text = 'Interval';

            % Create CheckBox_2
            app.CheckBox_2 = uicheckbox(app.Panel_10);
            app.CheckBox_2.ValueChangedFcn = createCallbackFcn(app, ↙
@CheckBox_2ValueChanged, true);
            app.CheckBox_2.Text = '';
            app.CheckBox_2.WordWrap = 'on';
            app.CheckBox_2.FontSize = 10;
            app.CheckBox_2.Position = [173 353 21 22];
            app.CheckBox_2.Value = true;

            % Create MaximumdepthEditFieldLabel_4
            app.MaximumdepthEditFieldLabel_4 = uilabel(app.Panel_10);
            app.MaximumdepthEditFieldLabel_4.HorizontalAlignment = 'right';
            app.MaximumdepthEditFieldLabel_4.Position = [110 446 53 22];
            app.MaximumdepthEditFieldLabel_4.Text = 'Y max';

            % Create yLimInterval
            app.yLimInterval = uieditfield(app.Panel_10, 'numeric');
            app.yLimInterval.Limits = [0 Inf];
            app.yLimInterval.ValueDisplayFormat = '%8.1f';
            app.yLimInterval.ValueChangedFcn = createCallbackFcn(app, ↙
@yLimIntervalValueChanged, true);
            app.yLimInterval.FontColor = [0.651 0.651 0.651];
            app.yLimInterval.Enable = 'off';
            app.yLimInterval.Tooltip = {'Y-axis tick interval'};
            app.yLimInterval.Position = [173 428 36 16];
            app.yLimInterval.Value = 0.5;

            % Create MaximumdepthEditFieldLabel_8
            app.MaximumdepthEditFieldLabel_8 = uilabel(app.Panel_10);
            app.MaximumdepthEditFieldLabel_8.HorizontalAlignment = 'right';
            app.MaximumdepthEditFieldLabel_8.Position = [110 425 53 22];
            app.MaximumdepthEditFieldLabel_8.Text = 'Interval';

            % Create yMinLimit
            app.yMinLimit = uieditfield(app.Panel_10, 'numeric');
            app.yMinLimit.ValueDisplayFormat = '%8.1f';
            app.yMinLimit.ValueChangedFcn = createCallbackFcn(app, ↙
@yMinLimitValueChanged, true);
            app.yMinLimit.FontColor = [0.651 0.651 0.651];
            app.yMinLimit.Enable = 'off';
            app.yMinLimit.Tooltip = {'Y-axis maximum value'};
            app.yMinLimit.Position = [173 407 45 16];
            app.yMinLimit.Value = -1;

            % Create MinimumdepthEditFieldLabel_3
            app.MinimumdepthEditFieldLabel_3 = uilabel(app.Panel_10);
            app.MinimumdepthEditFieldLabel_3.HorizontalAlignment = 'right';
            app.MinimumdepthEditFieldLabel_3.Position = [112 404 51 22];
            app.MinimumdepthEditFieldLabel_3.Text = 'Y min';

            % Create yMaxLimit
            app.yMaxLimit = uieditfield(app.Panel_10, 'numeric');
            app.yMaxLimit.Limits = [0 Inf];
            app.yMaxLimit.ValueDisplayFormat = '%8.1f';
            app.yMaxLimit.ValueChangedFcn = createCallbackFcn(app, @yMaxLimitValueChanged, true);
            app.yMaxLimit.FontColor = [0.651 0.651 0.651];
            app.yMaxLimit.Enable = 'off';
            app.yMaxLimit.Tooltip = {'Y-axis maximum value'};
            app.yMaxLimit.Position = [173 449 45 16];
            app.yMaxLimit.Value = 1;

            % Create CheckBox
            app.CheckBox = uicheckbox(app.Panel_10);
            app.CheckBox.ValueChangedFcn = createCallbackFcn(app, @CheckBoxValueChanged, true);
            app.CheckBox.Text = '';
            app.CheckBox.WordWrap = 'on';
            app.CheckBox.FontSize = 8;
            app.CheckBox.FontWeight = 'bold';
            app.CheckBox.Position = [173 370 16 31];
            app.CheckBox.Value = true;

            % Create YAutoSetLabel
            app.YAutoSetLabel = uilabel(app.Panel_10);
            app.YAutoSetLabel.HorizontalAlignment = 'right';
            app.YAutoSetLabel.FontSize = 10;
            app.YAutoSetLabel.Tooltip = {'Automatically set the Y axis limits'};
            app.YAutoSetLabel.Position = [110 374 53 22];
            app.YAutoSetLabel.Text = 'Y Auto Set';

            % Create XAutoSetLabel
            app.XAutoSetLabel = uilabel(app.Panel_10);
            app.XAutoSetLabel.HorizontalAlignment = 'right';
            app.XAutoSetLabel.FontSize = 10;
            app.XAutoSetLabel.Tooltip = {'Automatically set the X axis limits'};
            app.XAutoSetLabel.Position = [110 353 53 22];
            app.XAutoSetLabel.Text = 'X Auto Set';

            % Create LabelEditFieldLabel
            app.LabelEditFieldLabel = uilabel(app.Panel_10);
            app.LabelEditFieldLabel.Position = [50 252 39 22];
            app.LabelEditFieldLabel.Text = 'Y Axis';

            % Create YAxisEditField
            app.YAxisEditField = uieditfield(app.Panel_10, 'text');
            app.YAxisEditField.ValueChangedFcn = createCallbackFcn(app, @YAxisEditFieldValueChanged, true);
            app.YAxisEditField.FontColor = [0.651 0.651 0.651];
            app.YAxisEditField.Position = [105 255 108 16];
```

```matlab
            app.YAxisEditField.Value = '\eta (m)';

            % Create XandYLimitsLabel_2
            app.XandYLimitsLabel_2 = uilabel(app.Panel_10);
            app.XandYLimitsLabel_2.BackgroundColor = [0.9412 0.9412 0.9412];
            app.XandYLimitsLabel_2.FontWeight = 'bold';
            app.XandYLimitsLabel_2.Position = [241 467 91 22];
            app.XandYLimitsLabel_2.Text = 'X and Y Limits:';

            % Create YAxisLabelEditField_2Label
            app.YAxisLabelEditField_2Label = uilabel(app.Panel_10);
            app.YAxisLabelEditField_2Label.Position = [50 276 39 22];
            app.YAxisLabelEditField_2Label.Text = 'X Axis';

            % Create XAxisEditField
            app.XAxisEditField = uieditfield(app.Panel_10, 'text');
            app.XAxisEditField.ValueChangedFcn = createCallbackFcn(app, @XAxisEditFieldValueChanged, true);
            app.XAxisEditField.ValueChangingFcn = createCallbackFcn(app, @XAxisEditFieldValueChanging, true);
            app.XAxisEditField.FontColor = [0.651 0.651 0.651];
            app.XAxisEditField.Position = [105 279 108 16];
            app.XAxisEditField.Value = 'Time (sec)';

            % Create BackgroundGridLabel
            app.BackgroundGridLabel = uilabel(app.Panel_10);
            app.BackgroundGridLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.BackgroundGridLabel.FontWeight = 'bold';
            app.BackgroundGridLabel.Position = [288 179 103 22];
            app.BackgroundGridLabel.Text = 'Background Grid';

            % Create StyleDropDown_4Label
            app.StyleDropDown_4Label = uilabel(app.Panel_10);
            app.StyleDropDown_4Label.Tooltip = {'Line style of the grid lines'};
            app.StyleDropDown_4Label.Position = [289 155 32 22];
            app.StyleDropDown_4Label.Text = 'Style';

            % Create GridStyle
            app.GridStyle = uidropdown(app.Panel_10);
            app.GridStyle.Items = {'None', 'X axis only', 'Y axis only', 'Both - major lines', 'Both - with minor lines'};
            app.GridStyle.DropDownOpeningFcn = createCallbackFcn(app, @GridStyleDropDownOpening, true);
            app.GridStyle.ValueChangedFcn = createCallbackFcn(app, @GridStyleValueChanged, true);
            app.GridStyle.Tooltip = {''};
            app.GridStyle.Position = [364 157 100 17];
            app.GridStyle.Value = 'None';

            % Create Lineaty0DropDownLabel
            app.Lineaty0DropDownLabel = uilabel(app.Panel_10);
            app.Lineaty0DropDownLabel.WordWrap = 'on';
            app.Lineaty0DropDownLabel.Tooltip = {'Add a horizontal line at y = 0'};
            app.Lineaty0DropDownLabel.Position = [289 108 69 22];
            app.Lineaty0DropDownLabel.Text = 'Line at y=0';

            % Create Lineaty0DropDown
            app.Lineaty0DropDown = uidropdown(app.Panel_10);
            app.Lineaty0DropDown.Items = {'None', 'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.Lineaty0DropDown.ValueChangedFcn = createCallbackFcn(app, @Lineaty0DropDownValueChanged, true);
            app.Lineaty0DropDown.Tooltip = {''};
            app.Lineaty0DropDown.Position = [364 110 100 17];
            app.Lineaty0DropDown.Value = 'Dotted';

            % Create LineYThickness
            app.LineYThickness = uieditfield(app.Panel_10, 'numeric');
            app.LineYThickness.Limits = [0.1 Inf];
            app.LineYThickness.ValueDisplayFormat = '%2.1f';
            app.LineYThickness.ValueChangedFcn = createCallbackFcn(app, @LineYThicknessValueChanged, true);
            app.LineYThickness.HorizontalAlignment = 'left';
            app.LineYThickness.FontColor = [0.651 0.651 0.651];
            app.LineYThickness.Tooltip = {''};
            app.LineYThickness.Position = [364 87 29 16];
            app.LineYThickness.Value = 0.5;

            % Create ThicknessLabel_3
            app.ThicknessLabel_3 = uilabel(app.Panel_10);
            app.ThicknessLabel_3.WordWrap = 'on';
            app.ThicknessLabel_3.Position = [289 82 71 27];
            app.ThicknessLabel_3.Text = 'Width at y=0';

            % Create ThicknessLabel_5
            app.ThicknessLabel_5 = uilabel(app.Panel_10);
            app.ThicknessLabel_5.Position = [50 227 53 22];
            app.ThicknessLabel_5.Text = 'Text Size';

            % Create GridLabelSize
            app.GridLabelSize = uieditfield(app.Panel_10, 'numeric');
            app.GridLabelSize.Limits = [1 Inf];
            app.GridLabelSize.ValueDisplayFormat = '%2.0f';
            app.GridLabelSize.ValueChangedFcn = createCallbackFcn(app, @GridLabelSizeValueChanged, true);
            app.GridLabelSize.FontColor = [0.651 0.651 0.651];
            app.GridLabelSize.Position = [106 230 29 16];
            app.GridLabelSize.Value = 12;

            % Create GaugeLineColor_multiple
            app.GaugeLineColor_multiple = uidropdown(app.Panel_10);
            app.GaugeLineColor_multiple.Items = {'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', 'lines', 'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter'};
            app.GaugeLineColor_multiple.ValueChangedFcn = createCallbackFcn(app, @GaugeLineColor_multipleValueChanged, true);
            app.GaugeLineColor_multiple.Position = [108 87 89 17];
            app.GaugeLineColor_multiple.Value = 'lines';

            % Create FlipCheckBox_3
            app.FlipCheckBox_3 = uicheckbox(app.Panel_10);
            app.FlipCheckBox_3.Tooltip = {'Reverse the color sequence'};
            app.FlipCheckBox_3.Text = 'Flip';
            app.FlipCheckBox_3.FontSize = 11;
            app.FlipCheckBox_3.Position = [207 85 40 22];

            % Create Height
            app.Height = uieditfield(app.Panel_10, 'numeric');
            app.Height.Limits = [1 Inf];
            app.Height.ValueDisplayFormat = '%2.0f';
            app.Height.ValueChangedFcn = createCallbackFcn(app, @HeightValueChanged, true);
            app.Height.FontColor = [0.651 0.651 0.651];
            app.Height.Enable = 'off';
            app.Height.Tooltip = {''};
            app.Height.Position = [283 27 40 16];
            app.Height.Value = 8;

            % Create ThicknessLabel_6
            app.ThicknessLabel_6 = uilabel(app.Panel_10);
            app.ThicknessLabel_6.Tooltip = {'Unit: inches'};
            app.ThicknessLabel_6.Position = [239 24 41 22];
            app.ThicknessLabel_6.Text = 'Height';

            % Create FigureSizeLabel
            app.FigureSizeLabel = uilabel(app.Panel_10);
            app.FigureSizeLabel.FontWeight = 'bold';
            app.FigureSizeLabel.Position = [50 43 69 22];
            app.FigureSizeLabel.Text = 'Figure Size';

            % Create Width
            app.Width = uieditfield(app.Panel_10, 'numeric');
            app.Width.Limits = [1 Inf];
            app.Width.ValueDisplayFormat = '%2.0f';
            app.Width.ValueChangedFcn = createCallbackFcn(app, @WidthValueChanged, true);
            app.Width.FontColor = [0.651 0.651 0.651];
            app.Width.Enable = 'off';
            app.Width.Tooltip = {''};
            app.Width.Position = [182 27 41 16];
            app.Width.Value = 11;

            % Create ThicknessLabel_7
            app.ThicknessLabel_7 = uilabel(app.Panel_10);
            app.ThicknessLabel_7.Tooltip = {'Unit: inches'};
            app.ThicknessLabel_7.Position = [139 24 37 22];
            app.ThicknessLabel_7.Text = 'Width';

            % Create AutoSetCheckBox
            app.AutoSetCheckBox = uicheckbox(app.Panel_10);
            app.AutoSetCheckBox.ValueChangedFcn = createCallbackFcn(app, @AutoSetCheckBoxValueChanged, true);
            app.AutoSetCheckBox.Text = 'Auto Set';
            app.AutoSetCheckBox.Position = [50 24 68 22];
            app.AutoSetCheckBox.Value = true;

            % Create CloseFiguresButton_2
            app.CloseFiguresButton_2 = uibutton(app.Panel_10, 'push');
            app.CloseFiguresButton_2.ButtonPushedFcn = createCallbackFcn(app, @CloseFiguresButton_2Pushed2, true);
            app.CloseFiguresButton_2.Tooltip = {''};
            app.CloseFiguresButton_2.Position = [364 24 89 20];
            app.CloseFiguresButton_2.Text = 'Close Figures';

            % Create LineYThickness_2
            app.LineYThickness_2 = uieditfield(app.Panel_10, 'numeric');
            app.LineYThickness_2.Limits = [0.1 Inf];
            app.LineYThickness_2.ValueDisplayFormat = '%2.1f';
            app.LineYThickness_2.ValueChangedFcn = createCallbackFcn(app, @LineYThickness_2ValueChanged, true);
            app.LineYThickness_2.Editable = 'off';
            app.LineYThickness_2.FontColor = [0.651 0.651 0.651];
            app.LineYThickness_2.Position = [364 134 29 16];
            app.LineYThickness_2.Value = 0.5;

            % Create ThicknessLabel_12
            app.ThicknessLabel_12 = uilabel(app.Panel_10);
            app.ThicknessLabel_12.Position = [289 131 37 22];
            app.ThicknessLabel_12.Text = 'Width';

            % Create LegendLabel
            app.LegendLabel = uilabel(app.Panel_10);
            app.LegendLabel.FontWeight = 'bold';
            app.LegendLabel.Position = [287 297 48 22];
            app.LegendLabel.Text = 'Legend';

            % Create legendfirsttextlabel
            app.legendfirsttextlabel = uilabel(app.Panel_10);
            app.legendfirsttextlabel.Tooltip = {'Text added to data labels, for example: Station 1, Station 2'};
            app.legendfirsttextlabel.Position = [288 276 35 22];

            % Create legendfirsttext
            app.legendfirsttext = uieditfield(app.Panel_10, 'text');
            app.legendfirsttext.ValueChangedFcn = createCallbackFcn(app, @legendfirsttextValueChanged, true);
            app.legendfirsttext.FontColor = [0.651 0.651 0.651];
            app.legendfirsttext.Tooltip = {''};
            app.legendfirsttext.Position = [345 279 104 16];
            app.legendfirsttext.Value = 'Station';

            % Create LocationDropDownLabel
            app.LocationDropDownLabel = uilabel(app.Panel_10);
            app.LocationDropDownLabel.Position = [288 227 52 22];
            app.LocationDropDownLabel.Text = 'Location';

            % Create LocationDropDown
            app.LocationDropDown = uidropdown(app.Panel_10);
            app.LocationDropDown.Items = {'North', 'South', 'East', 'West', 'Northeast', 'Northwest', 'Southeast', 'Southwest', 'Northoutside', 'Southoutside', 'Eastoutside', 'Westoutside', 'Northeastoutside', 'Northwestoutside', 'Southeastoutside', 'Southwestoutside', 'Best', 'Bestoutside', 'None'};
            app.LocationDropDown.ValueChangedFcn = createCallbackFcn(app, @LocationDropDownValueChanged, true);
            app.LocationDropDown.Tooltip = {''};
            app.LocationDropDown.Position = [345 230 104 17];
            app.LocationDropDown.Value = 'Best';

            % Create LegendSize
            app.LegendSize = uieditfield(app.Panel_10, 'numeric');
```

```matlab
            app.LegendSize.Limits = [1 Inf];
            app.LegendSize.ValueDisplayFormat = '%2.0f';
            app.LegendSize.ValueChangedFcn = createCallbackFcn(app, @LegendSizeValueChanged, true);
            app.LegendSize.FontColor = [0.651 0.651 0.651];
            app.LegendSize.Position = [345 255 29 16];
            app.LegendSize.Value = 12;

            % Create ThicknessLabel_4
            app.ThicknessLabel_4 = uilabel(app.Panel_10);
            app.ThicknessLabel_4.Position = [288 252 53 22];
            app.ThicknessLabel_4.Text = 'Text Size';

            % Create MarkerDropDownLabel
            app.MarkerDropDownLabel = uilabel(app.Panel_10);
            app.MarkerDropDownLabel.Position = [50 155 43 22];
            app.MarkerDropDownLabel.Text = 'Marker';

            % Create MarkerDropDown
            app.MarkerDropDown = uidropdown(app.Panel_10);
            app.MarkerDropDown.Items = {'none', 'o', '+', '*', '.', 'x', '-', '|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'};
            app.MarkerDropDown.Tooltip = {'none'; 'o'; '+'; '*'};
            app.MarkerDropDown.Position = [106 157 92 17];
            app.MarkerDropDown.Value = 'none';

            % Create PLOTButton
            app.PLOTButton = uibutton(app.GaugeRecordsTab, 'push');
            app.PLOTButton.ButtonPushedFcn = createCallbackFcn(app, @PLOTButtonPushed, true);
            app.PLOTButton.FontSize = 14;
            app.PLOTButton.FontWeight = 'bold';
            app.PLOTButton.Position = [179 6 183 25];
            app.PLOTButton.Text = 'PLOT';

            % Create Panel_11
            app.Panel_11 = uipanel(app.GaugeRecordsTab);
            app.Panel_11.AutoResizeChildren = 'off';
            app.Panel_11.SizeChangedFcn = createCallbackFcn(app, @Panel_11SizeChanged, true);
            app.Panel_11.Position = [11 703 520 119];

            % Create FilesLabel_2
            app.FilesLabel_2 = uilabel(app.Panel_11);
            app.FilesLabel_2.FontWeight = 'bold';
            app.FilesLabel_2.Tooltip = {'Load the files.'; ''; 'Files must start with ''sta_'''};
            app.FilesLabel_2.Position = [44 80 32 22];
            app.FilesLabel_2.Text = 'Files';

            % Create FileTextArea_2
            app.FileTextArea_2 = uitextarea(app.Panel_11);
            app.FileTextArea_2.ValueChangedFcn = createCallbackFcn(app, @FileTextArea_2ValueChanged, true);
            app.FileTextArea_2.Editable = 'off';
            app.FileTextArea_2.FontColor = [0.651 0.651 0.651];
            app.FileTextArea_2.Tooltip = {''};
            app.FileTextArea_2.Placeholder = 'sta_xxxx';
            app.FileTextArea_2.Position = [44 19 95 60];

            % Create Button_16
            app.Button_16 = uibutton(app.Panel_11, 'push');
            app.Button_16.ButtonPushedFcn = createCallbackFcn(app, @Button_16Pushed, true);
            app.Button_16.FontAngle = 'italic';
            app.Button_16.FontColor = [0.651 0.651 0.651];
            app.Button_16.Tooltip = {'Load the files.'; ''; 'Files must start with ''sta_'''};
            app.Button_16.Position = [146 60 19 19];
            app.Button_16.Text = '...';

            % Create XaxisLabel
            app.XaxisLabel = uilabel(app.Panel_11);
            app.XaxisLabel.Tooltip = {'Select column to use for plotting.'; ''; 'Unit for each column: '; '1: time (s)'; '2: eta (m)'; '3: u vector (m/s)'; '4: v vector (m/s)'; '5: z vector (m/s)'};
            app.XaxisLabel.Position = [188 57 44 22];
            app.XaxisLabel.Text = 'X axis: ';

            % Create SelectColumnLabel
            app.SelectColumnLabel = uilabel(app.Panel_11);
            app.SelectColumnLabel.FontWeight = 'bold';
            app.SelectColumnLabel.Tooltip = {'Select column to use for plotting.'; ''; 'Unit for each column: '; '1: time (s)'; '2: eta (m)'; '3: u vector (m/s)'; '4: v vector (m/s)'; '5: z vector (m/s)'};
            app.SelectColumnLabel.Position = [188 80 89 22];
            app.SelectColumnLabel.Text = 'Select Column';

            % Create XaxisdataLabel
            app.XaxisdataLabel = uilabel(app.Panel_11);
            app.XaxisdataLabel.Tooltip = {'Example:'; 'Convert column 1 data from seconds to minutes. '};
            app.XaxisdataLabel.Position = [357 56 65 22];
            app.XaxisdataLabel.Text = 'X axis data';

            % Create ConvertLabel
            app.ConvertLabel = uilabel(app.Panel_11);
            app.ConvertLabel.FontWeight = 'bold';
            app.ConvertLabel.Tooltip = {'Examples:'; 'Convert column 1 data from seconds to minutes. '; ''; 'Convert column 2 data from meters to centimeters.'};
            app.ConvertLabel.Position = [333 80 54 22];
            app.ConvertLabel.Text = 'Convert:';

            % Create numeratorY
            app.numeratorY = uieditfield(app.Panel_11, 'numeric');
            app.numeratorY.Limits = [1 Inf];
            app.numeratorY.ValueDisplayFormat = '%8.0f';
            app.numeratorY.HorizontalAlignment = 'center';
            app.numeratorY.FontColor = [0.651 0.651 0.651];
            app.numeratorY.Enable = 'off';
            app.numeratorY.Position = [445 26 36 16];
            app.numeratorY.Value = 100;

            % Create denomeratorY
            app.denomeratorY = uieditfield(app.Panel_11, 'numeric');
            app.denomeratorY.Limits = [1 Inf];
            app.denomeratorY.ValueDisplayFormat = '%8.0f';
            app.denomeratorY.ValueChangedFcn = createCallbackFcn(app, @denomeratorYValueChanged, true);
            app.denomeratorY.HorizontalAlignment = 'center';
            app.denomeratorY.FontColor = [0.651 0.651 0.651];
            app.denomeratorY.Enable = 'off';
            app.denomeratorY.Position = [445 11 36 16];
            app.denomeratorY.Value = 1;

            % Create Label_9
            app.Label_9 = uilabel(app.Panel_11);
            app.Label_9.Position = [424 19 13 22];
            app.Label_9.Text = '×';

            % Create YaxisEditFieldLabel
            app.YaxisEditFieldLabel = uilabel(app.Panel_11);
            app.YaxisEditFieldLabel.Tooltip = {'Example: Convert column 2 data from meters to centimeters.'};
            app.YaxisEditFieldLabel.Position = [357 19 65 22];
            app.YaxisEditFieldLabel.Text = 'Y axis data';

            % Create CheckBox_3
            app.CheckBox_3 = uicheckbox(app.Panel_11);
            app.CheckBox_3.ValueChangedFcn = createCallbackFcn(app, @CheckBox_3ValueChanged, true);
            app.CheckBox_3.Text = '';
            app.CheckBox_3.Position = [333 56 25 22];

            % Create CheckBox_4
            app.CheckBox_4 = uicheckbox(app.Panel_11);
            app.CheckBox_4.ValueChangedFcn = createCallbackFcn(app, @CheckBox_4ValueChanged, true);
            app.CheckBox_4.Tooltip = {'Only the input in the first textbox under ''Y axis: Use data from column'' will be converted'};
            app.CheckBox_4.Text = '';
            app.CheckBox_4.Position = [333 19 25 22];

            % Create numeratorX
            app.numeratorX = uieditfield(app.Panel_11, 'numeric');
            app.numeratorX.Limits = [1 Inf];
            app.numeratorX.ValueDisplayFormat = '%8.0f';
            app.numeratorX.HorizontalAlignment = 'center';
            app.numeratorX.FontColor = [0.651 0.651 0.651];
            app.numeratorX.Enable = 'off';
            app.numeratorX.Position = [445 64 36 16];
            app.numeratorX.Value = 1;

            % Create denomeratorX
            app.denomeratorX = uieditfield(app.Panel_11, 'numeric');
            app.denomeratorX.Limits = [1 Inf];
            app.denomeratorX.ValueDisplayFormat = '%8.0f';
            app.denomeratorX.HorizontalAlignment = 'center';
            app.denomeratorX.FontColor = [0.651 0.651 0.651];
            app.denomeratorX.Enable = 'off';
            app.denomeratorX.Position = [445 49 36 16];
            app.denomeratorX.Value = 60;

            % Create Label_5
            app.Label_5 = uilabel(app.Panel_11);
            app.Label_5.Position = [424 56 13 22];
            app.Label_5.Text = '×';

            % Create Label_6
            app.Label_6 = uilabel(app.Panel_11);
            app.Label_6.FontSize = 22;
            app.Label_6.Position = [435 51 10 29];
            app.Label_6.Text = '(';

            % Create Label_7
            app.Label_7 = uilabel(app.Panel_11);
            app.Label_7.FontSize = 22;
            app.Label_7.Position = [487 52 10 29];
            app.Label_7.Text = ')';

            % Create Label_10
            app.Label_10 = uilabel(app.Panel_11);
            app.Label_10.FontSize = 22;
            app.Label_10.Position = [435 12 10 29];
            app.Label_10.Text = '(';

            % Create Label_11
            app.Label_11 = uilabel(app.Panel_11);
            app.Label_11.FontSize = 22;
            app.Label_11.Position = [487 13 10 29];
            app.Label_11.Text = ')';

            % Create YaxisSetcolumncountLabel
            app.YaxisSetcolumncountLabel = uilabel(app.Panel_11);
            app.YaxisSetcolumncountLabel.Tooltip = {'Select column to use for plotting.'; ''; 'Unit for each column: '; '1: time (s)'; '2: eta (m)'; '3: u vector (m/s)'; '4: v vector (m/s)'; '5: z vector (m/s)'};
            app.YaxisSetcolumncountLabel.Position = [188 33 45 22];
            app.YaxisSetcolumncountLabel.Text = 'Y axis: ';

            % Create YaxisUseDataDropDown
            app.YaxisUseDataDropDown = uidropdown(app.Panel_11);
            app.YaxisUseDataDropDown.Items = {'1', '2', '3', '4', '5', '3,4,5'};
            app.YaxisUseDataDropDown.DropDownOpeningFcn = createCallbackFcn(app, @YaxisUseDataDropDownOpening, true);
            app.YaxisUseDataDropDown.ValueChangedFcn = createCallbackFcn(app, @YaxisUseDataDropDownValueChanged, true);
            app.YaxisUseDataDropDown.Tooltip = {''};
            app.YaxisUseDataDropDown.Position = [231 36 61 16];
            app.YaxisUseDataDropDown.Value = '2';

            % Create XaxisUseDataDropDown
            app.XaxisUseDataDropDown = uidropdown(app.Panel_11);
            app.XaxisUseDataDropDown.Items = {'1', '2', '3', '4'};
            app.XaxisUseDataDropDown.ValueChangedFcn = createCallbackFcn(app, @XaxisUseDataDropDownValueChanged, true);
            app.XaxisUseDataDropDown.Tooltip = {''};
            app.XaxisUseDataDropDown.Position = [231 61 41 16];
            app.XaxisUseDataDropDown.Value = '1';

            % Create InputDataLabel_3
            app.InputDataLabel_3 = uilabel(app.GaugeRecordsTab);
            app.InputDataLabel_3.BackgroundColor = [0.9412 0.9412 0.9412];
            app.InputDataLabel_3.FontSize = 15;
            app.InputDataLabel_3.FontWeight = 'bold';
            app.InputDataLabel_3.FontColor = [0.0314 0.3686 0.6];
            app.InputDataLabel_3.Position = [38 813 97 22];
            app.InputDataLabel_3.Text = '  Input Data  ';

            % Create Panel_13
            app.Panel_13 = uipanel(app.GaugeRecordsTab);
            app.Panel_13.AutoResizeChildren = 'off';
            app.Panel_13.Position = [11 39 520 77];

            % Create pngCheckBox_2
            app.pngCheckBox_2 = uicheckbox(app.Panel_13);
            app.pngCheckBox_2.Tooltip = {'Resolution is set to 300 DPI'};
            app.pngCheckBox_2.Text = '.png';
            app.pngCheckBox_2.Position = [192 12 46 22];

            % Create FileFormatLabel_2
            app.FileFormatLabel_2 = uilabel(app.Panel_13);
            app.FileFormatLabel_2.Position = [68 12 66 22];
            app.FileFormatLabel_2.Text = 'File Format';

            % Create jpgCheckBox_2
            app.jpgCheckBox_2 = uicheckbox(app.Panel_13);
            app.jpgCheckBox_2.Tooltip = {'Resolution is set to 300 DPI'};
            app.jpgCheckBox_2.Text = '.jpg';
            app.jpgCheckBox_2.Position = [139 12 42 22];

            % Create pdfCheckBox_2
            app.pdfCheckBox_2 = uicheckbox(app.Panel_13);
            app.pdfCheckBox_2.Tooltip = {'Vector-type pdf file'};
            app.pdfCheckBox_2.Text = '.pdf';
            app.pdfCheckBox_2.Position = [249 12 43 22];

            % Create Button_18
            app.Button_18 = uibutton(app.Panel_13, 'push');
            app.Button_18.ButtonPushedFcn = createCallbackFcn(app, @Button_18Pushed, true);
            app.Button_18.FontSize = 10;
            app.Button_18.FontAngle = 'italic';
            app.Button_18.Tooltip = {'Load the preferred directory.'; ''; '"OUTPUT_FILES/Figures"  folder will be created in the selected directory.'};
            app.Button_18.Position = [403 42 19 19];
            app.Button_18.Text = '...';

            % Create OutputDirectoryLabel_3
            app.OutputDirectoryLabel_3 = uilabel(app.Panel_13);
            app.OutputDirectoryLabel_3.Position = [40 40 94 22];
            app.OutputDirectoryLabel_3.Text = 'Output Directory';

            % Create OutputDirectoryEditField_2
            app.OutputDirectoryEditField_2 = uieditfield(app.Panel_13, 'text');
            app.OutputDirectoryEditField_2.ValueChangedFcn = createCallbackFcn(app, @OutputDirectoryEditField_2ValueChanged, true);
            app.OutputDirectoryEditField_2.Editable = 'off';
            app.OutputDirectoryEditField_2.FontColor = [0 0 1];
            app.OutputDirectoryEditField_2.Tooltip = {'Load the preferred directory.'; ''; '''OUTPUT_FILES/Figures'' folder will be created in the selected directory.'};
            app.OutputDirectoryEditField_2.Placeholder = 'Default: Desktop';
            app.OutputDirectoryEditField_2.Position = [139 43 257 16];

            % Create txtCheckBox
            app.txtCheckBox = uicheckbox(app.Panel_13);
            app.txtCheckBox.ValueChangedFcn = createCallbackFcn(app, @txtCheckBoxValueChanged, true);
            app.txtCheckBox.Enable = 'off';
            app.txtCheckBox.Tooltip = {'Save raw data from ''sta_'' files as tab-delimited texts '; ''; 'Functionality enabled when "Plot separately" is selected.'};
            app.txtCheckBox.Text = '.txt';
            app.txtCheckBox.Position = [356 12 39 22];

            % Create epsCheckBox
            app.epsCheckBox = uicheckbox(app.Panel_13);
            app.epsCheckBox.Tooltip = {'Vector graphic file'};
            app.epsCheckBox.Text = '.eps';
            app.epsCheckBox.Position = [303 12 45 22];

            % Create SavePlotLabel
            app.SavePlotLabel = uilabel(app.GaugeRecordsTab);
            app.SavePlotLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.SavePlotLabel.FontSize = 15;
            app.SavePlotLabel.FontWeight = 'bold';
            app.SavePlotLabel.FontColor = [0.0314 0.3686 0.6];
            app.SavePlotLabel.Position = [38 105 92 22];
            app.SavePlotLabel.Text = '  Save Plot  ';

            % Create PlotStyleandLayoutLabel
            app.PlotStyleandLayoutLabel = uilabel(app.GaugeRecordsTab);
            app.PlotStyleandLayoutLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.PlotStyleandLayoutLabel.FontSize = 15;
            app.PlotStyleandLayoutLabel.FontWeight = 'bold';
            app.PlotStyleandLayoutLabel.FontColor = [0.0314 0.3686 0.6];
            app.PlotStyleandLayoutLabel.Position = [34 681 174 22];
            app.PlotStyleandLayoutLabel.Text = '  Plot Style and Layout  ';

            % Create VelocityMapTab
            app.VelocityMapTab = uitab(app.TabGroup);
            app.VelocityMapTab.AutoResizeChildren = 'off';
            app.VelocityMapTab.SizeChangedFcn = createCallbackFcn(app, @VelocityMapTabSizeChanged, true);
            app.VelocityMapTab.Title = 'Velocity Map';

            % Create GENERATEButton_2
            app.GENERATEButton_2 = uibutton(app.VelocityMapTab, 'push');
            app.GENERATEButton_2.ButtonPushedFcn = createCallbackFcn(app, @GENERATEButton_2Pushed, true);
            app.GENERATEButton_2.FontSize = 14;
            app.GENERATEButton_2.FontWeight = 'bold';
            app.GENERATEButton_2.Position = [179 6 183 25];
```

```matlab
            app.GENERATEButton_2.Text = 'GENERATE';

            % Create Panel_14
            app.Panel_14 = uipanel(app.VelocityMapTab);
            app.Panel_14.AutoResizeChildren = 'off';
            app.Panel_14.Position = [11 550 520 65];

            % Create QuiverColorDropDown
            app.QuiverColorDropDown = uidropdown(app.Panel_14);
            app.QuiverColorDropDown.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.QuiverColorDropDown.ValueChangedFcn = createCallbackFcn(app, @QuiverColorDropDownValueChanged, true);
            app.QuiverColorDropDown.Position = [396 12 87 16];
            app.QuiverColorDropDown.Value = 'Black';

            % Create XEditField_2Label_3
            app.XEditField_2Label_3 = uilabel(app.Panel_14);
            app.XEditField_2Label_3.Tooltip = {'Size scaling'};
            app.XEditField_2Label_3.Position = [131 9 67 22];
            app.XEditField_2Label_3.Text = 'Body Scale';

            % Create arrowscale
            app.arrowscale = uieditfield(app.Panel_14, 'numeric');
            app.arrowscale.Limits = [0 Inf];
            app.arrowscale.ValueDisplayFormat = '%8.1f';
            app.arrowscale.ValueChangedFcn = createCallbackFcn(app, @arrowscaleValueChanged, true);
            app.arrowscale.FontColor = [0.651 0.651 0.651];
            app.arrowscale.Tooltip = {'Scaling factor. '; ''; 'Set to 0 to disable auto scaling. '};
            app.arrowscale.Position = [228 12 35 16];
            app.arrowscale.Value = 10;

            % Create ArrowThicknessLabel
            app.ArrowThicknessLabel = uilabel(app.Panel_14);
            app.ArrowThicknessLabel.HorizontalAlignment = 'right';
            app.ArrowThicknessLabel.Tooltip = {'Line thickness'};
            app.ArrowThicknessLabel.Position = [281 9 37 22];
            app.ArrowThicknessLabel.Text = 'Width';

            % Create ArrowThickness
            app.ArrowThickness = uieditfield(app.Panel_14, 'numeric');
            app.ArrowThickness.Limits = [0 Inf];
            app.ArrowThickness.ValueDisplayFormat = '%.1f';
            app.ArrowThickness.ValueChangedFcn = createCallbackFcn(app, @ArrowThicknessValueChanged, true);
            app.ArrowThickness.FontColor = [0.651 0.651 0.651];
            app.ArrowThickness.Tooltip = {''};
            app.ArrowThickness.Position = [335 12 35 16];
            app.ArrowThickness.Value = 1;

            % Create ColorLabel_2
            app.ColorLabel_2 = uilabel(app.Panel_14);
            app.ColorLabel_2.Position = [397 30 35 22];
            app.ColorLabel_2.Text = 'Color';

            % Create ArrowSizeLabel
```

```
        app.ArrowSizeLabel = uilabel(app.Panel_14);
        app.ArrowSizeLabel.WordWrap = 'on';
        app.ArrowSizeLabel.Tooltip = {'Arrowhead size relative to the body size'};
        app.ArrowSizeLabel.Position = [131 27 85 29];
        app.ArrowSizeLabel.Text = 'Arrowhead Size';

        % Create ArrowHeadSize
        app.ArrowHeadSize = uieditfield(app.Panel_14, 'numeric');
        app.ArrowHeadSize.Limits = [0 1];
        app.ArrowHeadSize.ValueDisplayFormat = '%.1f';
        app.ArrowHeadSize.ValueChangedFcn = createCallbackFcn(app, @ArrowHeadSizeValueChanged, true);
        app.ArrowHeadSize.FontColor = [0.651 0.651 0.651];
        app.ArrowHeadSize.Tooltip = {''};
        app.ArrowHeadSize.Position = [228 33 35 16];
        app.ArrowHeadSize.Value = 0.2;

        % Create PlotvectorsCheckBox
        app.PlotvectorsCheckBox = uicheckbox(app.Panel_14);
        app.PlotvectorsCheckBox.ValueChangedFcn = createCallbackFcn(app, @PlotvectorsCheckBoxValueChanged, true);
        app.PlotvectorsCheckBox.Tooltip = {'Overlay vectors on the map. '; ''; 'Applicable when U and V vectors are loaded.'};
        app.PlotvectorsCheckBox.Text = 'Plot vectors';
        app.PlotvectorsCheckBox.FontWeight = 'bold';
        app.PlotvectorsCheckBox.Position = [31 19 91 22];

        % Create SpacingLabel
        app.SpacingLabel = uilabel(app.Panel_14);
        app.SpacingLabel.HorizontalAlignment = 'right';
        app.SpacingLabel.Tooltip = {'Spacing between arrows to avoid clustering on the map'};
        app.SpacingLabel.Position = [278 30 50 22];
        app.SpacingLabel.Text = 'Spacing';

        % Create ArrowSpacing
        app.ArrowSpacing = uieditfield(app.Panel_14, 'numeric');
        app.ArrowSpacing.Limits = [0.001 Inf];
        app.ArrowSpacing.ValueDisplayFormat = '%11.2f';
        app.ArrowSpacing.ValueChangedFcn = createCallbackFcn(app, @ArrowSpacingValueChanged, true);
        app.ArrowSpacing.FontColor = [0.651 0.651 0.651];
        app.ArrowSpacing.Position = [337 33 35 16];
        app.ArrowSpacing.Value = 1;

        % Create Panel_15
        app.Panel_15 = uipanel(app.VelocityMapTab);
        app.Panel_15.AutoResizeChildren = 'off';
        app.Panel_15.SizeChangedFcn = createCallbackFcn(app, @Panel_15SizeChanged, true);
        app.Panel_15.Position = [11 626 520 196];

        % Create FileTextArea_3
        app.FileTextArea_3 = uitextarea(app.Panel_15);
        app.FileTextArea_3.ValueChangedFcn = createCallbackFcn(app, @FileTextArea_3ValueChanged, true);
        app.FileTextArea_3.Editable = 'off';
```

```
            app.FileTextArea_3.Tooltip = {''};
            app.FileTextArea_3.Placeholder = 'u_xxxx; umean_xxxx; umax_xxxx';
            app.FileTextArea_3.Position = [32 77 84 74];

            % Create UvectorsLabel
            app.UvectorsLabel = uilabel(app.Panel_15);
            app.UvectorsLabel.WordWrap = 'on';
            app.UvectorsLabel.FontWeight = 'bold';
            app.UvectorsLabel.Tooltip = {'Load files with filenames starting with ''u'''};
            app.UvectorsLabel.Position = [33 157 61 22];
            app.UvectorsLabel.Text = 'U vectors';

            % Create DepthFileEditField_2
            app.DepthFileEditField_2 = uieditfield(app.Panel_15, 'text');
            app.DepthFileEditField_2.ValueChangedFcn = createCallbackFcn(app, @DepthFileEditField_2ValueChanged, true);
            app.DepthFileEditField_2.Editable = 'off';
            app.DepthFileEditField_2.Tooltip = {'Load the bathymetry file used in the simulation'; ''; 'Format accepted: .txt, .tif'};
            app.DepthFileEditField_2.Placeholder = '.txt, .tif., mask_';
            app.DepthFileEditField_2.Position = [33 22 96 16];

            % Create BathymetryLabel_2
            app.BathymetryLabel_2 = uilabel(app.Panel_15);
            app.BathymetryLabel_2.FontWeight = 'bold';
            app.BathymetryLabel_2.Tooltip = {'Load the file'; ''; 'Select the bathymetry used in the simulation'; ''; 'Format accepted: .txt, .tif'};
            app.BathymetryLabel_2.Position = [33 38 71 22];
            app.BathymetryLabel_2.Text = 'Bathymetry';

            % Create Button_22
            app.Button_22 = uibutton(app.Panel_15, 'push');
            app.Button_22.ButtonPushedFcn = createCallbackFcn(app, @Button_22Pushed, true);
            app.Button_22.Tooltip = {'Load the file'; ''; 'Select the bathymetry used in the simulation'; ''; 'Format accepted: .txt, .tif'};
            app.Button_22.Position = [136 20 19 19];
            app.Button_22.Text = '...';

            % Create SouthwestCornerLabel_2
            app.SouthwestCornerLabel_2 = uilabel(app.Panel_15);
            app.SouthwestCornerLabel_2.FontWeight = 'bold';
            app.SouthwestCornerLabel_2.Tooltip = {'Coordinates of the southwest corner of the loaded files.'};
            app.SouthwestCornerLabel_2.Position = [244 134 110 22];
            app.SouthwestCornerLabel_2.Text = 'Southwest Corner';

            % Create LongitudeEditField_2Label
            app.LongitudeEditField_2Label = uilabel(app.Panel_15);
            app.LongitudeEditField_2Label.HorizontalAlignment = 'right';
            app.LongitudeEditField_2Label.WordWrap = 'on';
            app.LongitudeEditField_2Label.Tooltip = {'West boundary'; ''; 'Unit: degrees / meters'};
            app.LongitudeEditField_2Label.Position = [246 114 60 22];
            app.LongitudeEditField_2Label.Text = 'Longitude';

            % Create LongitudeEditField_2
            app.LongitudeEditField_2 = uieditfield(app.Panel_15, 'numeric');
            app.LongitudeEditField_2.ValueDisplayFormat = '%8.4f';
            app.LongitudeEditField_2.ValueChangedFcn = createCallbackFcn(app, @LongitudeEditField_2ValueChanged, true);
            app.LongitudeEditField_2.FontColor = [0.651 0.651 0.651];
            app.LongitudeEditField_2.Tooltip = {''};
            app.LongitudeEditField_2.Position = [314 117 60 16];

            % Create LatitudeEditField_2Label
            app.LatitudeEditField_2Label = uilabel(app.Panel_15);
            app.LatitudeEditField_2Label.HorizontalAlignment = 'right';
            app.LatitudeEditField_2Label.WordWrap = 'on';
            app.LatitudeEditField_2Label.Tooltip = {'South boundary'; ''; 'Unit: degrees / meters'};
            app.LatitudeEditField_2Label.Position = [251 94 44 17];
            app.LatitudeEditField_2Label.Text = 'Latitude';

            % Create LatitudeEditField_2
            app.LatitudeEditField_2 = uieditfield(app.Panel_15, 'numeric');
            app.LatitudeEditField_2.ValueDisplayFormat = '%8.4f';
            app.LatitudeEditField_2.ValueChangedFcn = createCallbackFcn(app, @LatitudeEditField_2ValueChanged, true);
            app.LatitudeEditField_2.FontColor = [0.651 0.651 0.651];
            app.LatitudeEditField_2.Tooltip = {''};
            app.LatitudeEditField_2.Position = [314 95 60 16];

            % Create GridSizeLabel
            app.GridSizeLabel = uilabel(app.Panel_15);
            app.GridSizeLabel.FontWeight = 'bold';
            app.GridSizeLabel.Tooltip = {'Resolution'; 'Unit: degrees / meters'};
            app.GridSizeLabel.Position = [401 134 57 22];
            app.GridSizeLabel.Text = 'Grid Size';

            % Create YEditField_2Label_2
            app.YEditField_2Label_2 = uilabel(app.Panel_15);
            app.YEditField_2Label_2.HorizontalAlignment = 'right';
            app.YEditField_2Label_2.Tooltip = {'Y direction'; ''; 'Unit: degrees / meters'};
            app.YEditField_2Label_2.Position = [403 92 13 21];
            app.YEditField_2Label_2.Text = 'Y';

            % Create gridY_2
            app.gridY_2 = uieditfield(app.Panel_15, 'numeric');
            app.gridY_2.Limits = [0 Inf];
            app.gridY_2.ValueDisplayFormat = '%8.5f';
            app.gridY_2.ValueChangedFcn = createCallbackFcn(app, @gridY_2ValueChanged, true);
            app.gridY_2.FontColor = [0.651 0.651 0.651];
            app.gridY_2.Tooltip = {''};
            app.gridY_2.Position = [427 94 60 16];

            % Create TotalSimuilationTimesecEditFieldLabel_4
            app.TotalSimuilationTimesecEditFieldLabel_4 = uilabel(app.Panel_15);
            app.TotalSimuilationTimesecEditFieldLabel_4.FontWeight = 'bold';
            app.TotalSimuilationTimesecEditFieldLabel_4.Position = [244 38 98 22];
            app.TotalSimuilationTimesecEditFieldLabel_4.Text = 'Simulation Time';

            % Create StartTime2
            app.StartTime2 = uieditfield(app.Panel_15, 'numeric');
            app.StartTime2.Limits = [0 Inf];
            app.StartTime2.ValueDisplayFormat = '%8.1f';
            app.StartTime2.ValueChangedFcn = createCallbackFcn(app, @StartTime2ValueChanged, true);
            app.StartTime2.FontColor = [0.651 0.651 0.651];
            app.StartTime2.Tooltip = {''};
            app.StartTime2.Position = [314 22 47 16];

            % Create StartLabel_2
            app.StartLabel_2 = uilabel(app.Panel_15);
            app.StartLabel_2.HorizontalAlignment = 'right';
            app.StartLabel_2.Tooltip = {'For imported initial tsunami files with a non-zero start time, input the value here. Example: for a 30-second snapshot, input ''30''.'; ''; 'Unit: seconds'};
            app.StartLabel_2.Position = [275 19 31 22];
            app.StartLabel_2.Text = 'Start';

            % Create IntervalLabel_2
            app.IntervalLabel_2 = uilabel(app.Panel_15);
            app.IntervalLabel_2.HorizontalAlignment = 'right';
            app.IntervalLabel_2.Tooltip = {'Snapshot interval.'; ''; 'Same as PLOT_INTV in input.txt'; ''; 'Unit: seconds'};
            app.IntervalLabel_2.Position = [372 19 45 22];
            app.IntervalLabel_2.Text = 'Interval';

            % Create TotalSimuilationTimesecEditField_4
            app.TotalSimuilationTimesecEditField_4 = uieditfield(app.Panel_15, 'numeric');
            app.TotalSimuilationTimesecEditField_4.Limits = [1 Inf];
            app.TotalSimuilationTimesecEditField_4.ValueDisplayFormat = '%8.1f';
            app.TotalSimuilationTimesecEditField_4.ValueChangedFcn = createCallbackFcn(app, @TotalSimuilationTimesecEditField_4ValueChanged, true);
            app.TotalSimuilationTimesecEditField_4.FontColor = [0.651 0.651 0.651];
            app.TotalSimuilationTimesecEditField_4.Tooltip = {''};
            app.TotalSimuilationTimesecEditField_4.Position = [427 22 60 16];
            app.TotalSimuilationTimesecEditField_4.Value = 1;

            % Create XEditField_2Label_2
            app.XEditField_2Label_2 = uilabel(app.Panel_15);
            app.XEditField_2Label_2.HorizontalAlignment = 'right';
            app.XEditField_2Label_2.Tooltip = {'X direction'; ''; 'Unit: degrees / meters'};
            app.XEditField_2Label_2.Position = [406 114 13 22];
            app.XEditField_2Label_2.Text = 'X ';

            % Create gridX_2
            app.gridX_2 = uieditfield(app.Panel_15, 'numeric');
            app.gridX_2.Limits = [0 Inf];
            app.gridX_2.ValueDisplayFormat = '%8.5f';
            app.gridX_2.ValueChangedFcn = createCallbackFcn(app, @gridX_2ValueChanged, true);
            app.gridX_2.FontColor = [0.651 0.651 0.651];
            app.gridX_2.Tooltip = {''};
            app.gridX_2.Position = [427 117 60 16];

            % Create FileTextArea_4
            app.FileTextArea_4 = uitextarea(app.Panel_15);
            app.FileTextArea_4.Editable = 'off';
            app.FileTextArea_4.Tooltip = {'The V-vector file is automatically searched in the same directory as the corresponding U-vector file.'};
            app.FileTextArea_4.Placeholder = '(Auto finds) v_xxxx; vmean_xxxx;';
            app.FileTextArea_4.Position = [127 77 84 74];

            % Create Button_19
            app.Button_19 = uibutton(app.Panel_15, 'push');
            app.Button_19.ButtonPushedFcn = createCallbackFcn(app, @Button_19Pushed, true);
            app.Button_19.FontAngle = 'italic';
            app.Button_19.FontColor = [0.651 0.651 0.651];
            app.Button_19.Tooltip = {'Load files with filenames starting with ''u'''};
            app.Button_19.Position = [93 159 19 19];
            app.Button_19.Text = '...';

            % Create InputDataLabel_4
            app.InputDataLabel_4 = uilabel(app.VelocityMapTab);
            app.InputDataLabel_4.BackgroundColor = [0.9412 0.9412 0.9412];
            app.InputDataLabel_4.FontSize = 15;
            app.InputDataLabel_4.FontWeight = 'bold';
            app.InputDataLabel_4.FontColor = [0.0314 0.3686 0.6];
            app.InputDataLabel_4.Position = [38 813 97 22];
            app.InputDataLabel_4.Text = '  Input Data  ';

            % Create ArrowsLabel
            app.ArrowsLabel = uilabel(app.VelocityMapTab);
            app.ArrowsLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.ArrowsLabel.FontSize = 15;
            app.ArrowsLabel.FontWeight = 'bold';
            app.ArrowsLabel.FontColor = [0.0314 0.3686 0.6];
            app.ArrowsLabel.Position = [38 603 73 22];
            app.ArrowsLabel.Text = '  Arrows  ';

            % Create Panel_16
            app.Panel_16 = uipanel(app.VelocityMapTab);
            app.Panel_16.AutoResizeChildren = 'off';
            app.Panel_16.SizeChangedFcn = createCallbackFcn(app, @Panel_16SizeChanged, true);
            app.Panel_16.Position = [11 315 520 222];

            % Create TabGroup3
            app.TabGroup3 = uitabgroup(app.Panel_16);
            app.TabGroup3.AutoResizeChildren = 'off';
            app.TabGroup3.Position = [0 -1 520 204];

            % Create BasemapTab
            app.BasemapTab = uitab(app.TabGroup3);
            app.BasemapTab.AutoResizeChildren = 'off';
            app.BasemapTab.SizeChangedFcn = createCallbackFcn(app,
@BasemapTabSizeChanged, true);
            app.BasemapTab.Title = 'Basemap';

            % Create ButtonGroup_16
            app.ButtonGroup_16 = uibuttongroup(app.BasemapTab);
            app.ButtonGroup_16.AutoResizeChildren = 'off';
            app.ButtonGroup_16.SelectionChangedFcn = createCallbackFcn(app, ↙
@ButtonGroup_16SelectionChanged, true);
            app.ButtonGroup_16.BorderType = 'none';
            app.ButtonGroup_16.FontWeight = 'bold';
            app.ButtonGroup_16.Position = [14 54 117 120];

            % Create VelocityButton
            app.VelocityButton = uiradiobutton(app.ButtonGroup_16);
            app.VelocityButton.Tooltip = {'Z component of the velocity'};
            app.VelocityButton.Text = 'Z Velocity';
            app.VelocityButton.FontWeight = 'bold';
            app.VelocityButton.Position = [11 98 78 22];
            app.VelocityButton.Value = true;

            % Create etaButton
            app.etaButton = uiradiobutton(app.ButtonGroup_16);
            app.etaButton.Tooltip = {'Sea surface displacement at  specific↙
time snapshot.'; ''; 'The eta files are automatically loaded. '; ''; 'Ensure the↙
eta files are in the same folder as the U vector files.'};
            app.etaButton.Text = 'eta';
            app.etaButton.FontWeight = 'bold';
            app.etaButton.Position = [10 74 40 22];

            % Create hmaxButton
            app.hmaxButton = uiradiobutton(app.ButtonGroup_16);
            app.hmaxButton.Tooltip = {'Maximum wave height.'; ''; ''; 'The↙
hmax files are automatically loaded. '; ''; 'Ensure the hmax files are in the same↙
folder as the U vector files.'};
            app.hmaxButton.Text = 'hmax';
            app.hmaxButton.FontWeight = 'bold';
            app.hmaxButton.Position = [9 50 53 22];

            % Create BathymetryButton
            app.BathymetryButton = uiradiobutton(app.ButtonGroup_16);
            app.BathymetryButton.Text = 'Bathymetry';
            app.BathymetryButton.FontWeight = 'bold';
            app.BathymetryButton.Position = [8 26 88 22];

            % Create VorticityButton
            app.VorticityButton = uiradiobutton(app.ButtonGroup_16);
            app.VorticityButton.Text = 'Vorticity';
            app.VorticityButton.FontWeight = 'bold';
            app.VorticityButton.Position = [7 3 69 22];

            % Create ThirdTabColorMapPanel
            app.ThirdTabColorMapPanel = uipanel(app.BasemapTab);
            app.ThirdTabColorMapPanel.AutoResizeChildren = 'off';
            app.ThirdTabColorMapPanel.BorderType = 'none';
            app.ThirdTabColorMapPanel.Position = [265 32 203 113];

            % Create maplabelsize
            app.maplabelsize = uieditfield(app.ThirdTabColorMapPanel, ↙
'numeric');
            app.maplabelsize.Limits = [0 50];
            app.maplabelsize.ValueDisplayFormat = '%3.0f';
            app.maplabelsize.ValueChangedFcn = createCallbackFcn(app, @maplabelsizeValueChanged, true);
            app.maplabelsize.FontColor = [0.651 0.651 0.651];
            app.maplabelsize.Tooltip = {''};
            app.maplabelsize.Position = [70 11 35 16];
            app.maplabelsize.Value = 10;

            % Create BackgroundMapColorDropDown
            app.BackgroundMapColorDropDown = uidropdown(app.ThirdTabColorMapPanel);
            app.BackgroundMapColorDropDown.Items = {'--- MATLAB default ----', 'autumn', 'bone', 'colorcube', 'cool', 'copper', 'flag', 'gray', 'hot', 'hsv', 'jet', 'parula', 'pink', 'prism', 'spring', 'summer', 'turbo', 'winter', '--- CBREWER 2 ---', '< sequential >', 'blue', 'blue - green', 'blue - purple', 'green - blue', 'greens', 'grays', 'oranges', 'orange - red', 'purple - blue', 'purple - blue - green', 'purple - red', 'purples', 'red - purple', 'reds', 'yellow - green', 'yellow - green - blue', 'yellow - orange - brown', 'yellow - orange - red', '< divergent >', 'brown - teal', 'pink - light green', 'purple - green', 'purple - orange', 'red - blue', 'red - gray', 'red - yellow - blue', 'red - yellow - green', 'spectral', '< qualitative >', 'accent', 'dark 2', 'paired', 'pastel 1', 'pastel 2', 'set 1', 'set 2', 'set 3'};
            app.BackgroundMapColorDropDown.DropDownOpeningFcn = createCallbackFcn(app, @BackgroundMapColorDropDownOpening, true);
            app.BackgroundMapColorDropDown.ValueChangedFcn = createCallbackFcn(app, @BackgroundMapColorDropDownValueChanged, true);
            app.BackgroundMapColorDropDown.Position = [8 92 174 16];
            app.BackgroundMapColorDropDown.Value = 'blue - purple';

            % Create FlipCheckBox_4
            app.FlipCheckBox_4 = uicheckbox(app.ThirdTabColorMapPanel);
            app.FlipCheckBox_4.Tooltip = {'Reverse the color sequence'};
            app.FlipCheckBox_4.Text = 'Flip';
            app.FlipCheckBox_4.FontSize = 11;
            app.FlipCheckBox_4.Position = [8 67 40 22];

            % Create InterpolationDivisionEditFieldLabel_2
            app.InterpolationDivisionEditFieldLabel_2 = uilabel(app.ThirdTabColorMapPanel);
            app.InterpolationDivisionEditFieldLabel_2.HorizontalAlignment = 'right';
            app.InterpolationDivisionEditFieldLabel_2.FontSize = 11;
            app.InterpolationDivisionEditFieldLabel_2.Tooltip = {'Defines the color distribution:'; ''; 'Low value:'; 'Distinct color boundaries'; ''; 'Higher value:'; 'Smoother color transition'};
            app.InterpolationDivisionEditFieldLabel_2.Position = [93 68 48 22];
            app.InterpolationDivisionEditFieldLabel_2.Text = 'Division';

            % Create InterpolationDivisionEditField_2
            app.InterpolationDivisionEditField_2 = uieditfield(app.ThirdTabColorMapPanel, 'numeric');
            app.InterpolationDivisionEditField_2.Limits = [0 Inf];
            app.InterpolationDivisionEditField_2.RoundFractionalValues = 'on';
            app.InterpolationDivisionEditField_2.ValueDisplayFormat = '%3.0f';
            app.InterpolationDivisionEditField_2.ValueChangedFcn =
createCallbackFcn(app, @InterpolationDivisionEditField_2ValueChanged, true);
            app.InterpolationDivisionEditField_2.FontColor = [0.651 0.651 0.651];
            app.InterpolationDivisionEditField_2.Tooltip = {''};
            app.InterpolationDivisionEditField_2.Position = [146 72 36 16];
            app.InterpolationDivisionEditField_2.Value = 4;

            % Create MinBarValue
            app.MinBarValue = uieditfield(app.ThirdTabColorMapPanel, 'numeric');
            app.MinBarValue.ValueDisplayFormat = '%.2f';
            app.MinBarValue.ValueChangedFcn = createCallbackFcn(app, @MinBarValueValueChanged, true);
            app.MinBarValue.FontColor = [0.651 0.651 0.651];
            app.MinBarValue.Tooltip = {''};
            app.MinBarValue.Position = [50 34 55 16];

            % Create toLabel_3
            app.toLabel_3 = uilabel(app.ThirdTabColorMapPanel);
            app.toLabel_3.Tooltip = {'Define the minimum value to display in the map.'; ''; 'Unit: meters'};
            app.toLabel_3.Position = [111 31 25 22];
            app.toLabel_3.Text = 'to';

            % Create mLabel_3
            app.mLabel_3 = uilabel(app.ThirdTabColorMapPanel);
            app.mLabel_3.Tooltip = {'Define the maximum value to display in the map.'; ''; 'Unit: meters'};
            app.mLabel_3.Position = [186 31 25 22];
            app.mLabel_3.Text = 'm';

            % Create MaxBarValue
            app.MaxBarValue = uieditfield(app.ThirdTabColorMapPanel, 'numeric');
            app.MaxBarValue.ValueDisplayFormat = '%.2f';
            app.MaxBarValue.ValueChangedFcn = createCallbackFcn(app, @MaxBarValueValueChanged, true);
            app.MaxBarValue.FontColor = [0.651 0.651 0.651];
            app.MaxBarValue.Tooltip = {''};
            app.MaxBarValue.Position = [127 34 55 16];
            app.MaxBarValue.Value = 6;

            % Create LimitsLabel
            app.LimitsLabel = uilabel(app.ThirdTabColorMapPanel);
            app.LimitsLabel.Tooltip = {'Define the maximum value to display in the map.'; ''; 'Unit: meters'};
            app.LimitsLabel.Position = [8 31 38 22];
            app.LimitsLabel.Text = 'Limits';

            % Create TextSizeLabel
            app.TextSizeLabel = uilabel(app.ThirdTabColorMapPanel);
            app.TextSizeLabel.Tooltip = {'Define the maximum value to display in the map.'; ''; 'Unit: meters'};
            app.TextSizeLabel.Position = [8 8 53 22];
            app.TextSizeLabel.Text = 'Text Size';

            % Create ColormapLabel
            app.ColormapLabel = uilabel(app.BasemapTab);
            app.ColormapLabel.FontWeight = 'bold';
            app.ColormapLabel.Position = [195 120 62 22];
            app.ColormapLabel.Text = 'Colormap';

            % Create ColorbarLabel
            app.ColorbarLabel = uilabel(app.BasemapTab);
            app.ColorbarLabel.FontWeight = 'bold';
            app.ColorbarLabel.Position = [195 62 56 22];
            app.ColorbarLabel.Text = 'Colorbar';

            % Create FileTextArea_5
            app.FileTextArea_5 = uitextarea(app.BasemapTab);
            app.FileTextArea_5.ValueChangedFcn = createCallbackFcn(app, @FileTextArea_5ValueChanged, true);
            app.FileTextArea_5.Editable = 'off';
            app.FileTextArea_5.FontColor = [0.149 0.149 0.149];
            app.FileTextArea_5.BackgroundColor = [0.9412 0.9412 0.9412];
            app.FileTextArea_5.Tooltip = {'File preview. '; ''; 'Ensure that the listed files are in the same folder as the U vector files.'};
            app.FileTextArea_5.Placeholder = '(Filename preview)';
            app.FileTextArea_5.Position = [21 16 114 35];

            % Create HorizontalCheckBox_2
            app.HorizontalCheckBox_2 = uicheckbox(app.BasemapTab);
            app.HorizontalCheckBox_2.Text = 'Horizontal';
            app.HorizontalCheckBox_2.Position = [302 148 76 22];

            % Create VerticalCheckBox_2
            app.VerticalCheckBox_2 = uicheckbox(app.BasemapTab);
            app.VerticalCheckBox_2.Text = 'Vertical';
            app.VerticalCheckBox_2.Position = [388 148 61 22];

            % Create FlipBasemapLabel
            app.FlipBasemapLabel = uilabel(app.BasemapTab);
            app.FlipBasemapLabel.FontWeight = 'bold';
            app.FlipBasemapLabel.Tooltip = {'Flips the basemap.'; ''; 'Horizontally (left-right)'; 'Vertically (up-side down)'};
            app.FlipBasemapLabel.Position = [195 148 83 22];
            app.FlipBasemapLabel.Text = 'Flip Basemap';

            % Create CoastlinecolourDropDownLabel_12
            app.CoastlinecolourDropDownLabel_12 = uilabel(app.BasemapTab);
            app.CoastlinecolourDropDownLabel_12.FontWeight = 'bold';
            app.CoastlinecolourDropDownLabel_12.Tooltip = {'Required input:'; 'Bathymetry file'};
            app.CoastlinecolourDropDownLabel_12.Position = [195 7 69 22];
            app.CoastlinecolourDropDownLabel_12.Text = 'Land Color';

            % Create LandColor
            app.LandColor = uidropdown(app.BasemapTab);
            app.LandColor.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.LandColor.Enable = 'off';
            app.LandColor.Tooltip = {''};
            app.LandColor.Position = [273 10 105 16];
            app.LandColor.Value = 'Dark gray';

            % Create BathymetryContoursTab_2
            app.BathymetryContoursTab_2 = uitab(app.TabGroup3);
            app.BathymetryContoursTab_2.AutoResizeChildren = 'off';
            app.BathymetryContoursTab_2.Tooltip = {'Enabled when bathymetry file is uploaded'};
            app.BathymetryContoursTab_2.Title = 'Bathymetry Contours';

            % Create RangeLabel
            app.RangeLabel = uilabel(app.BathymetryContoursTab_2);
            app.RangeLabel.FontWeight = 'bold';
            app.RangeLabel.Position = [45 109 42 22];
            app.RangeLabel.Text = 'Range';

            % Create PlotBathymetryContoursCheckBox
            app.PlotBathymetryContoursCheckBox = uicheckbox(app.BathymetryContoursTab_2);
            app.PlotBathymetryContoursCheckBox.ValueChangedFcn = createCallbackFcn(app, @PlotBathymetryContoursCheckBoxValueChanged, true);
            app.PlotBathymetryContoursCheckBox.Enable = 'off';
            app.PlotBathymetryContoursCheckBox.Text = 'Plot Bathymetry Contours';
            app.PlotBathymetryContoursCheckBox.FontWeight = 'bold';
            app.PlotBathymetryContoursCheckBox.Position = [45 141 170 22];

            % Create MinimumEditField_2Label
            app.MinimumEditField_2Label = uilabel(app.BathymetryContoursTab_2);
            app.MinimumEditField_2Label.Position = [45 86 55 22];
            app.MinimumEditField_2Label.Text = 'Minimum';

            % Create MinimumEditField_2
            app.MinimumEditField_2 = uieditfield(app.BathymetryContoursTab_2, 'numeric');
            app.MinimumEditField_2.Limits = [0 Inf];
            app.MinimumEditField_2.ValueDisplayFormat = '%8.1f';
            app.MinimumEditField_2.ValueChangedFcn = createCallbackFcn(app, @MinimumEditField_2ValueChanged, true);
            app.MinimumEditField_2.FontColor = [0.651 0.651 0.651];
            app.MinimumEditField_2.Enable = 'off';
            app.MinimumEditField_2.Tooltip = {'0 value = coastline'; ''; 'Lower values indicate shallower waters.'};
            app.MinimumEditField_2.Position = [102 89 57 16];

            % Create MaximumEditField_2Label
            app.MaximumEditField_2Label = uilabel(app.BathymetryContoursTab_2);
            app.MaximumEditField_2Label.Position = [45 32 58 22];
            app.MaximumEditField_2Label.Text = 'Maximum';

            % Create MaximumEditField_2
            app.MaximumEditField_2 = uieditfield(app.BathymetryContoursTab_2, 'numeric');
            app.MaximumEditField_2.Limits = [0.1 Inf];
            app.MaximumEditField_2.ValueDisplayFormat = '%8.1f';
            app.MaximumEditField_2.ValueChangedFcn = createCallbackFcn(app, @MaximumEditField_2ValueChanged, true);
            app.MaximumEditField_2.FontColor = [0.651 0.651 0.651];
            app.MaximumEditField_2.Enable = 'off';
            app.MaximumEditField_2.Tooltip = {'Higher value indicate deeper
```

```
waters'};
            app.MaximumEditField_2.Position = [102 35 57 16];
            app.MaximumEditField_2.Value = 10000;

            % Create IntervalEditField_6Label
            app.IntervalEditField_6Label = uilabel(app.BathymetryContoursTab_2);
            app.IntervalEditField_6Label.Position = [45 59 45 22];
            app.IntervalEditField_6Label.Text = 'Interval';

            % Create IntervalEditField_6
            app.IntervalEditField_6 = uieditfield(app.BathymetryContoursTab_2, 'numeric');
            app.IntervalEditField_6.Limits = [0 Inf];
            app.IntervalEditField_6.ValueDisplayFormat = '%8.1f';
            app.IntervalEditField_6.ValueChangedFcn = createCallbackFcn(app, @IntervalEditField_6ValueChanged, true);
            app.IntervalEditField_6.FontColor = [0.651 0.651 0.651];
            app.IntervalEditField_6.Enable = 'off';
            app.IntervalEditField_6.Tooltip = {'Interval between adjacent contours'};
            app.IntervalEditField_6.Position = [102 62 48 16];
            app.IntervalEditField_6.Value = 500;

            % Create LineLabel_4
            app.LineLabel_4 = uilabel(app.BathymetryContoursTab_2);
            app.LineLabel_4.FontWeight = 'bold';
            app.LineLabel_4.Position = [196 109 30 22];
            app.LineLabel_4.Text = 'Line';

            % Create AddLabelCheckBox_2
            app.AddLabelCheckBox_2 = uicheckbox(app.BathymetryContoursTab_2);
            app.AddLabelCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @AddLabelCheckBox_2ValueChanged, true);
            app.AddLabelCheckBox_2.Enable = 'off';
            app.AddLabelCheckBox_2.Text = 'Add Label';
            app.AddLabelCheckBox_2.FontWeight = 'bold';
            app.AddLabelCheckBox_2.Position = [375 109 80 22];

            % Create ColorDropDown_4Label
            app.ColorDropDown_4Label = uilabel(app.BathymetryContoursTab_2);
            app.ColorDropDown_4Label.Position = [196 32 35 22];
            app.ColorDropDown_4Label.Text = 'Color';

            % Create ColorDropDown_4
            app.ColorDropDown_4 = uidropdown(app.BathymetryContoursTab_2);
            app.ColorDropDown_4.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown_4.Enable = 'off';
            app.ColorDropDown_4.Position = [234 35 104 16];
            app.ColorDropDown_4.Value = 'Black';

            % Create StyleDropDown_3Label_2
            app.StyleDropDown_3Label_2 = uilabel(app.BathymetryContoursTab_2);
            app.StyleDropDown_3Label_2.Position = [196 59 32 22];
            app.StyleDropDown_3Label_2.Text = 'Style';

            % Create StyleDropDown_3
            app.StyleDropDown_3 = uidropdown(app.BathymetryContoursTab_2);
            app.StyleDropDown_3.Items = {'Solid', 'Dashed', 'Dotted', 'Dash-dotted'};
            app.StyleDropDown_3.Enable = 'off';
            app.StyleDropDown_3.Position = [234 62 104 16];
            app.StyleDropDown_3.Value = 'Solid';

            % Create WidthEditField_2Label
            app.WidthEditField_2Label = uilabel(app.BathymetryContoursTab_2);
            app.WidthEditField_2Label.Position = [196 86 37 22];
            app.WidthEditField_2Label.Text = 'Width';

            % Create WidthEditField_2
            app.WidthEditField_2 = uieditfield(app.BathymetryContoursTab_2, 'numeric');
            app.WidthEditField_2.Limits = [0 Inf];
            app.WidthEditField_2.ValueDisplayFormat = '%3.1f';
            app.WidthEditField_2.ValueChangedFcn = createCallbackFcn(app, @WidthEditField_2ValueChanged, true);
            app.WidthEditField_2.FontColor = [0.651 0.651 0.651];
            app.WidthEditField_2.Enable = 'off';
            app.WidthEditField_2.Tooltip = {'Line thickness'};
            app.WidthEditField_2.Position = [234 89 33 16];
            app.WidthEditField_2.Value = 0.1;

            % Create IntervalEditField_8Label
            app.IntervalEditField_8Label = uilabel(app.BathymetryContoursTab_2);
            app.IntervalEditField_8Label.Tooltip = {'Add labels at specified intervals.'; ''; 'Suggestion: Set at intervals divisible by the contour range interval'; ''; ''; 'Suggestion: Set at intervals divisible by the depth range interval'};
            app.IntervalEditField_8Label.Position = [375 59 45 22];
            app.IntervalEditField_8Label.Text = 'Interval';

            % Create LabelSizeCont
            app.LabelSizeCont = uieditfield(app.BathymetryContoursTab_2, 'numeric');
            app.LabelSizeCont.Limits = [0 30];
            app.LabelSizeCont.ValueDisplayFormat = '%8.0f';
            app.LabelSizeCont.ValueChangedFcn = createCallbackFcn(app, @LabelSizeContValueChanged, true);
            app.LabelSizeCont.FontColor = [0.651 0.651 0.651];
            app.LabelSizeCont.Enable = 'off';
            app.LabelSizeCont.Tooltip = {''};
            app.LabelSizeCont.Position = [426 89 40 16];
            app.LabelSizeCont.Value = 8;

            % Create SpacingEditField_4Label_3
            app.SpacingEditField_4Label_3 = uilabel(app.BathymetryContoursTab_2);
            app.SpacingEditField_4Label_3.Tooltip = {'Font size'};
            app.SpacingEditField_4Label_3.Position = [375 86 28 22];
            app.SpacingEditField_4Label_3.Text = 'Size';

            % Create SpacingEditField_4Label_2
            app.SpacingEditField_4Label_2 = uilabel(app.BathymetryContoursTab_2);
            app.SpacingEditField_4Label_2.WordWrap = 'on';
            app.SpacingEditField_4Label_2.Tooltip = {'Adjust the distance↙
between the labels'; ''; 'Higher values reduce label crowding'};
            app.SpacingEditField_4Label_2.Position = [375 29 77 29];
            app.SpacingEditField_4Label_2.Text = 'Spacing';

            % Create SpacingEditField
            app.SpacingEditField = uieditfield(app.BathymetryContoursTab_2,↙
'numeric');
            app.SpacingEditField.Limits = [0 Inf];
            app.SpacingEditField.ValueDisplayFormat = '%8.1f';
            app.SpacingEditField.ValueChangedFcn = createCallbackFcn(app,↙
@SpacingEditFieldValueChanged, true);
            app.SpacingEditField.FontColor = [0.651 0.651 0.651];
            app.SpacingEditField.Enable = 'off';
            app.SpacingEditField.Tooltip = {''};
            app.SpacingEditField.Position = [426 35 47 16];
            app.SpacingEditField.Value = 100;

            % Create IntervalEditField_8
            app.IntervalEditField_8 = uieditfield(app.BathymetryContoursTab_2,↙
'numeric');
            app.IntervalEditField_8.Limits = [0 Inf];
            app.IntervalEditField_8.ValueDisplayFormat = '%8.1f';
            app.IntervalEditField_8.ValueChangedFcn = createCallbackFcn(app,↙
@IntervalEditField_8ValueChanged, true);
            app.IntervalEditField_8.FontColor = [0.651 0.651 0.651];
            app.IntervalEditField_8.Enable = 'off';
            app.IntervalEditField_8.Tooltip = {''};
            app.IntervalEditField_8.Position = [426 62 47 16];
            app.IntervalEditField_8.Value = 100;

            % Create GaugesTab_2
            app.GaugesTab_2 = uitab(app.TabGroup3);
            app.GaugesTab_2.AutoResizeChildren = 'off';
            app.GaugesTab_2.SizeChangedFcn = createCallbackFcn(app,↙
@GaugesTab_2SizeChanged, true);
            app.GaugesTab_2.Title = 'Gauges';

            % Create CoastlinecolourDropDownLabel_19
            app.CoastlinecolourDropDownLabel_19 = uilabel(app.GaugesTab_2);
            app.CoastlinecolourDropDownLabel_19.Position = [45 57 32 22];
            app.CoastlinecolourDropDownLabel_19.Text = 'Style';

            % Create gaugemarkerVelocityTab
            app.gaugemarkerVelocityTab = uidropdown(app.GaugesTab_2);
            app.gaugemarkerVelocityTab.Items = {'o', '+', '*', '.', 'x', '-',↙
'|', '^', 'v', '>', '<', 'diamond', 'hexagram', 'pentagram', 'square'};
            app.gaugemarkerVelocityTab.Enable = 'off';
            app.gaugemarkerVelocityTab.Tooltip = {'Marker style'};
            app.gaugemarkerVelocityTab.Position = [81 60 92 16];
            app.gaugemarkerVelocityTab.Value = 'o';

            % Create ColorDropDown_7Label
            app.ColorDropDown_7Label = uilabel(app.GaugesTab_2);
            app.ColorDropDown_7Label.Position = [45 30 35 22];
            app.ColorDropDown_7Label.Text = 'Color';

            % Create ColorDropDown_7
            app.ColorDropDown_7 = uidropdown(app.GaugesTab_2);
            app.ColorDropDown_7.Items = {'Black', 'Dark gray', 'Medium gray', 'Light gray', 'Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'White'};
            app.ColorDropDown_7.Enable = 'off';
            app.ColorDropDown_7.Position = [81 33 103 16];
            app.ColorDropDown_7.Value = 'Black';

            % Create Button_27
            app.Button_27 = uibutton(app.GaugesTab_2, 'push');
            app.Button_27.ButtonPushedFcn = createCallbackFcn(app, @Button_27Pushed, true);
            app.Button_27.Enable = 'off';
            app.Button_27.Tooltip = {'Load the file'; ''; 'File extension must be .shp or .txt (tab delimited)'};
            app.Button_27.Position = [211 116 19 19];
            app.Button_27.Text = '...';

            % Create GaugeFileLabel_2
            app.GaugeFileLabel_2 = uilabel(app.GaugesTab_2);
            app.GaugeFileLabel_2.FontWeight = 'bold';
            app.GaugeFileLabel_2.Position = [45 114 26 22];
            app.GaugeFileLabel_2.Text = 'File';

            % Create FileEditField_2
            app.FileEditField_2 = uieditfield(app.GaugesTab_2, 'text');
            app.FileEditField_2.ValueChangedFcn = createCallbackFcn(app, @FileEditField_2ValueChanged, true);
            app.FileEditField_2.Editable = 'off';
            app.FileEditField_2.Enable = 'off';
            app.FileEditField_2.Tooltip = {'File extension must be .shp or .txt'};
            app.FileEditField_2.Placeholder = '.shp or .txt';
            app.FileEditField_2.Position = [74 117 130 16];

            % Create PlotGaugesCheckBox
            app.PlotGaugesCheckBox = uicheckbox(app.GaugesTab_2);
            app.PlotGaugesCheckBox.ValueChangedFcn = createCallbackFcn(app, @PlotGaugesCheckBoxValueChanged, true);
            app.PlotGaugesCheckBox.Text = 'Plot Gauges';
            app.PlotGaugesCheckBox.FontWeight = 'bold';
            app.PlotGaugesCheckBox.Position = [45 141 92 22];

            % Create AddLabelCheckBox
            app.AddLabelCheckBox = uicheckbox(app.GaugesTab_2);
            app.AddLabelCheckBox.ValueChangedFcn = createCallbackFcn(app, @AddLabelCheckBoxValueChanged, true);
            app.AddLabelCheckBox.Enable = 'off';
            app.AddLabelCheckBox.Tooltip = {'The label follows the sequence of latitude-longitude combinations from the loaded file.'; ''; 'The labels correspond to the numeric values in the sta_ files.'};
            app.AddLabelCheckBox.Text = '  Add Label';
            app.AddLabelCheckBox.FontWeight = 'bold';
            app.AddLabelCheckBox.Position = [254 84 86 22];

            % Create FontSizeEditField_2Label
            app.FontSizeEditField_2Label = uilabel(app.GaugesTab_2);
            app.FontSizeEditField_2Label.Enable = 'off';
            app.FontSizeEditField_2Label.Tooltip = {'Font size'};
            app.FontSizeEditField_2Label.Position = [255 57 33 22];
            app.FontSizeEditField_2Label.Text = 'Size';

            % Create FontSizeEditField_2
            app.FontSizeEditField_2 = uieditfield(app.GaugesTab_2, 'numeric');
            app.FontSizeEditField_2.Limits = [0 100];
            app.FontSizeEditField_2.ValueDisplayFormat = '%8.0f';
            app.FontSizeEditField_2.ValueChangedFcn = createCallbackFcn(app, @FontSizeEditField_2ValueChanged, true);
            app.FontSizeEditField_2.FontColor = [0.651 0.651 0.651];
            app.FontSizeEditField_2.Enable = 'off';
            app.FontSizeEditField_2.Tooltip = {'Font size'};
            app.FontSizeEditField_2.Position = [290 60 30 16];
            app.FontSizeEditField_2.Value = 12;

            % Create GLspacing_2
            app.GLspacing_2 = uieditfield(app.GaugesTab_2, 'numeric');
            app.GLspacing_2.Limits = [0 100];
            app.GLspacing_2.ValueDisplayFormat = '%8.3f';
            app.GLspacing_2.ValueChangedFcn = createCallbackFcn(app, @GLspacing_2ValueChanged, true);
            app.GLspacing_2.FontColor = [0.651 0.651 0.651];
            app.GLspacing_2.Enable = 'off';
            app.GLspacing_2.Tooltip = {''};
            app.GLspacing_2.Position = [394 60 40 16];
            app.GLspacing_2.Value = 0.5;

            % Create GLspacingLabel_2
            app.GLspacingLabel_2 = uilabel(app.GaugesTab_2);
            app.GLspacingLabel_2.Enable = 'off';
            app.GLspacingLabel_2.Tooltip = {'Horizontal distance between the point and the label. '; ' '; ' A value of 0.5 means that a label for a point at 120°E will be placed at 120.5°E.'; ''; 'Unit: degrees / meters'};
            app.GLspacingLabel_2.Position = [339 57 50 22];
            app.GLspacingLabel_2.Text = 'Spacing';

            % Create AlignmentLabel_2
            app.AlignmentLabel_2 = uilabel(app.GaugesTab_2);
            app.AlignmentLabel_2.Enable = 'off';
            app.AlignmentLabel_2.Tooltip = {'Horizontal and vertical alignments relative to the marker location'};
            app.AlignmentLabel_2.Position = [255 30 59 22];
            app.AlignmentLabel_2.Text = 'Alignment';

            % Create CoastlinecolourDropDown_10
            app.CoastlinecolourDropDown_10 = uidropdown(app.GaugesTab_2);
            app.CoastlinecolourDropDown_10.Items = {'Centre', 'Right', 'Left'};
            app.CoastlinecolourDropDown_10.Enable = 'off';
            app.CoastlinecolourDropDown_10.Tooltip = {''};
            app.CoastlinecolourDropDown_10.Position = [314 33 70 16];
            app.CoastlinecolourDropDown_10.Value = 'Right';

            % Create CoastlinecolourDropDown_11
            app.CoastlinecolourDropDown_11 = uidropdown(app.GaugesTab_2);
            app.CoastlinecolourDropDown_11.Items = {'Centre', 'Top', 'Bottom'};
            app.CoastlinecolourDropDown_11.Enable = 'off';
            app.CoastlinecolourDropDown_11.Tooltip = {''};
            app.CoastlinecolourDropDown_11.Position = [389 33 70 16];
            app.CoastlinecolourDropDown_11.Value = 'Centre';

            % Create SizeEditField_2Label
            app.SizeEditField_2Label = uilabel(app.GaugesTab_2);
            app.SizeEditField_2Label.Position = [45 84 28 22];
            app.SizeEditField_2Label.Text = 'Size';

            % Create SizeEditField_2
            app.SizeEditField_2 = uieditfield(app.GaugesTab_2, 'numeric');
            app.SizeEditField_2.Limits = [0 100];
            app.SizeEditField_2.ValueDisplayFormat = '%3.0f';
            app.SizeEditField_2.ValueChangedFcn = createCallbackFcn(app, ↙
@SizeEditField_2ValueChanged, true);
            app.SizeEditField_2.FontColor = [0.651 0.651 0.651];
            app.SizeEditField_2.Enable = 'off';
            app.SizeEditField_2.Position = [81 87 30 16];
            app.SizeEditField_2.Value = 5;

            % Create Panel_17
            app.Panel_17 = uipanel(app.VelocityMapTab);
            app.Panel_17.AutoResizeChildren = 'off';
            app.Panel_17.Position = [11 39 520 77];

            % Create pngCheckBox_3
            app.pngCheckBox_3 = uicheckbox(app.Panel_17);
            app.pngCheckBox_3.Tooltip = {'Resolution is set to 300 DPI'};
            app.pngCheckBox_3.Text = '.png';
            app.pngCheckBox_3.Position = [186 12 46 22];

            % Create FileFormatLabel_3
            app.FileFormatLabel_3 = uilabel(app.Panel_17);
            app.FileFormatLabel_3.Position = [69 12 66 22];
            app.FileFormatLabel_3.Text = 'File Format';

            % Create jpgCheckBox_3
            app.jpgCheckBox_3 = uicheckbox(app.Panel_17);
            app.jpgCheckBox_3.Tooltip = {'Resolution is set to 300 DPI'};
            app.jpgCheckBox_3.Text = '.jpg';
            app.jpgCheckBox_3.Position = [138 12 42 22];

            % Create Button_26
            app.Button_26 = uibutton(app.Panel_17, 'push');
            app.Button_26.ButtonPushedFcn = createCallbackFcn(app, ↙
@Button_26Pushed, true);
            app.Button_26.FontSize = 10;
            app.Button_26.FontAngle = 'italic';
            app.Button_26.Tooltip = {'Load the preferred directory.'; '';↙
'"OUTPUT_FILES/Figures"  folder will be created in the selected directory.'};
            app.Button_26.Position = [403 42 19 19];
            app.Button_26.Text = '...';

            % Create FramerateEditFieldLabel
            app.FramerateEditFieldLabel = uilabel(app.Panel_17);
            app.FramerateEditFieldLabel.WordWrap = 'on';
            app.FramerateEditFieldLabel.Visible = 'off';
            app.FramerateEditFieldLabel.Tooltip = {'Frames per second.'; ''; 'Lower value = slower animation.'};
            app.FramerateEditFieldLabel.Position = [383 7 77 33];
            app.FramerateEditFieldLabel.Text = 'Frame rate';

            % Create FramerateEditField
            app.FramerateEditField = uieditfield(app.Panel_17, 'numeric');
            app.FramerateEditField.Limits = [0 100];
            app.FramerateEditField.ValueDisplayFormat = '%.0f';
            app.FramerateEditField.ValueChangedFcn = createCallbackFcn(app, @FramerateEditFieldValueChanged, true);
            app.FramerateEditField.FontColor = [0.651 0.651 0.651];
            app.FramerateEditField.Visible = 'off';
            app.FramerateEditField.Tooltip = {''};
            app.FramerateEditField.Position = [448 15 30 17];
            app.FramerateEditField.Value = 5;

            % Create mp4CheckBox_2
            app.mp4CheckBox_2 = uicheckbox(app.Panel_17);
            app.mp4CheckBox_2.ValueChangedFcn = createCallbackFcn(app, @mp4CheckBox_2ValueChanged, true);
            app.mp4CheckBox_2.Enable = 'off';
            app.mp4CheckBox_2.Tooltip = {'Create an animation.'; ''; 'File is saved as "animation.mp4."'; ''; 'Enabled when "Plot separately" is selected.'};
            app.mp4CheckBox_2.Text = '.mp4';
            app.mp4CheckBox_2.Position = [324 12 49 22];

            % Create OutputDirectoryLabel_2
            app.OutputDirectoryLabel_2 = uilabel(app.Panel_17);
            app.OutputDirectoryLabel_2.Tooltip = {'Load the preferred directory.'; ''; '''OUTPUT_FILES/Figures'' folder will be created in the selected directory.'};
            app.OutputDirectoryLabel_2.Position = [40 40 94 22];
            app.OutputDirectoryLabel_2.Text = 'Output Directory';

            % Create OutputDirectoryEditField_3
            app.OutputDirectoryEditField_3 = uieditfield(app.Panel_17, 'text');
            app.OutputDirectoryEditField_3.ValueChangedFcn = createCallbackFcn(app, @OutputDirectoryEditField_3ValueChanged, true);
            app.OutputDirectoryEditField_3.Editable = 'off';
            app.OutputDirectoryEditField_3.FontColor = [0 0 1];
            app.OutputDirectoryEditField_3.Tooltip = {''};
            app.OutputDirectoryEditField_3.Placeholder = 'Default: Desktop';
            app.OutputDirectoryEditField_3.Position = [139 43 257 16];

            % Create txtCheckBox_2
            app.txtCheckBox_2 = uicheckbox(app.Panel_17);
            app.txtCheckBox_2.Enable = 'off';
            app.txtCheckBox_2.Tooltip = {'Save raw data of U and V vector files as tab-delimited texts.'; ''; 'It is enabled when "Plot separately" is selected.'};
            app.txtCheckBox_2.Text = '.txt';
            app.txtCheckBox_2.Position = [238 12 39 22];

            % Create tifCheckBox
            app.tifCheckBox = uicheckbox(app.Panel_17);
            app.tifCheckBox.Enable = 'off';
            app.tifCheckBox.Tooltip = {'Georeferenced raster file.'; ''; 'Enabled when "Plot separately" is selected.'};
            app.tifCheckBox.Text = '.tif';
            app.tifCheckBox.Position = [283 12 35 22];

            % Create SaveMapLabel_2
            app.SaveMapLabel_2 = uilabel(app.VelocityMapTab);
            app.SaveMapLabel_2.BackgroundColor = [0.9412 0.9412 0.9412];
            app.SaveMapLabel_2.FontSize = 15;
            app.SaveMapLabel_2.FontWeight = 'bold';
            app.SaveMapLabel_2.FontColor = [0.0314 0.3686 0.6];
            app.SaveMapLabel_2.Position = [38 105 92 22];
            app.SaveMapLabel_2.Text = '  Save Map  ';

            % Create BasemapandOverlaysLabel
            app.BasemapandOverlaysLabel = uilabel(app.VelocityMapTab);
            app.BasemapandOverlaysLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.BasemapandOverlaysLabel.FontSize = 15;
            app.BasemapandOverlaysLabel.FontWeight = 'bold';
            app.BasemapandOverlaysLabel.FontColor = [0.0314 0.3686 0.6];
            app.BasemapandOverlaysLabel.Position = [38 527 187 22];
            app.BasemapandOverlaysLabel.Text = '  Basemap and Overlays  ';

            % Create Panel_18
            app.Panel_18 = uipanel(app.VelocityMapTab);
            app.Panel_18.AutoResizeChildren = 'off';
            app.Panel_18.Position = [11 128 520 172];

            % Create BoundaryLimitsLabel_2
            app.BoundaryLimitsLabel_2 = uilabel(app.Panel_18);
            app.BoundaryLimitsLabel_2.FontWeight = 'bold';
            app.BoundaryLimitsLabel_2.Tooltip = {'When all textboxes are left unchanged (all zeroes), the values are automatically set based on the extent of the uploaded file and the input southwest corner coordinates'};
            app.BoundaryLimitsLabel_2.Position = [25 91 99 22];
            app.BoundaryLimitsLabel_2.Text = 'Boundary Limits';

            % Create PlotindegreesCheckBox
            app.PlotindegreesCheckBox = uicheckbox(app.Panel_18);
            app.PlotindegreesCheckBox.ValueChangedFcn = createCallbackFcn(app, @PlotindegreesCheckBoxValueChanged, true);
            app.PlotindegreesCheckBox.Tooltip = {'Change the label format into geographic coordinates'};
            app.PlotindegreesCheckBox.Text = 'Plot in degrees';
            app.PlotindegreesCheckBox.FontSize = 11;
            app.PlotindegreesCheckBox.Position = [403 56 96 22];

            % Create EastEditField_2Label
            app.EastEditField_2Label = uilabel(app.Panel_18);
            app.EastEditField_2Label.Position = [331 62 29 22];
            app.EastEditField_2Label.Text = 'East';

            % Create EastEditField_2
            app.EastEditField_2 = uieditfield(app.Panel_18, 'numeric');
            app.EastEditField_2.ValueDisplayFormat = '%8.4f';
            app.EastEditField_2.ValueChangedFcn = createCallbackFcn(app, @EastEditField_2ValueChanged, true);
            app.EastEditField_2.FontColor = [0.651 0.651 0.651];
            app.EastEditField_2.Position = [265 65 60 16];

            % Create WestEditField_2Label
            app.WestEditField_2Label = uilabel(app.Panel_18);
            app.WestEditField_2Label.HorizontalAlignment = 'right';
            app.WestEditField_2Label.Position = [87 62 32 22];
            app.WestEditField_2Label.Text = 'West';

            % Create WestEditField_2
            app.WestEditField_2 = uieditfield(app.Panel_18, 'numeric');
            app.WestEditField_2.ValueDisplayFormat = '%8.4f';
            app.WestEditField_2.ValueChangedFcn = createCallbackFcn(app, ↙
@WestEditField_2ValueChanged, true);
            app.WestEditField_2.FontColor = [0.651 0.651 0.651];
            app.WestEditField_2.Position = [125 65 60 16];

            % Create NorthEditField_2Label
            app.NorthEditField_2Label = uilabel(app.Panel_18);
            app.NorthEditField_2Label.HorizontalAlignment = 'right';
            app.NorthEditField_2Label.Position = [206 91 35 22];
            app.NorthEditField_2Label.Text = 'North';

            % Create NorthEditField_2
            app.NorthEditField_2 = uieditfield(app.Panel_18, 'numeric');
            app.NorthEditField_2.ValueDisplayFormat = '%8.4f';
            app.NorthEditField_2.ValueChangedFcn = createCallbackFcn(app, ↙
@NorthEditField_2ValueChanged, true);
            app.NorthEditField_2.FontColor = [0.651 0.651 0.651];
            app.NorthEditField_2.Position = [195 77 60 16];

            % Create SouthEditField_2Label
            app.SouthEditField_2Label = uilabel(app.Panel_18);
            app.SouthEditField_2Label.HorizontalAlignment = 'right';
            app.SouthEditField_2Label.Position = [205 34 37 22];
            app.SouthEditField_2Label.Text = 'South';

            % Create SouthEditField_2
            app.SouthEditField_2 = uieditfield(app.Panel_18, 'numeric');
            app.SouthEditField_2.ValueDisplayFormat = '%8.4f';
            app.SouthEditField_2.ValueChangedFcn = createCallbackFcn(app, ↙
@SouthEditField_2ValueChanged, true);
            app.SouthEditField_2.FontColor = [0.651 0.651 0.651];
            app.SouthEditField_2.Position = [195 53 60 16];

            % Create Height_3
            app.Height_3 = uieditfield(app.Panel_18, 'numeric');
            app.Height_3.Limits = [1 Inf];
            app.Height_3.ValueDisplayFormat = '%2.0f';
            app.Height_3.ValueChangedFcn = createCallbackFcn(app, ↙
@Height_3ValueChanged, true);
            app.Height_3.FontColor = [0.651 0.651 0.651];
            app.Height_3.Enable = 'off';
            app.Height_3.Tooltip = {''};
            app.Height_3.Position = [330 12 40 16];
            app.Height_3.Value = 8;

            % Create ThicknessLabel_10
            app.ThicknessLabel_10 = uilabel(app.Panel_18);
            app.ThicknessLabel_10.Tooltip = {'Unit: inches'};
            app.ThicknessLabel_10.Position = [283 9 41 22];
            app.ThicknessLabel_10.Text = 'Height';

            % Create AutoSetCheckBox_3
            app.AutoSetCheckBox_3 = uicheckbox(app.Panel_18);
            app.AutoSetCheckBox_3.ValueChangedFcn = createCallbackFcn(app, @AutoSetCheckBox_3ValueChanged, true);
            app.AutoSetCheckBox_3.Text = ' Auto Set';
            app.AutoSetCheckBox_3.FontSize = 11;
            app.AutoSetCheckBox_3.Position = [102 9 68 22];
            app.AutoSetCheckBox_3.Value = true;

            % Create Width_3
            app.Width_3 = uieditfield(app.Panel_18, 'numeric');
            app.Width_3.Limits = [1 Inf];
            app.Width_3.ValueDisplayFormat = '%2.0f';
            app.Width_3.ValueChangedFcn = createCallbackFcn(app, @Width_3ValueChanged, true);
            app.Width_3.FontColor = [0.651 0.651 0.651];
            app.Width_3.Enable = 'off';
            app.Width_3.Tooltip = {''};
            app.Width_3.Position = [225 12 41 16];
            app.Width_3.Value = 11;

            % Create ThicknessLabel_11
            app.ThicknessLabel_11 = uilabel(app.Panel_18);
            app.ThicknessLabel_11.Tooltip = {'Unit: inches'};
            app.ThicknessLabel_11.Position = [188 9 37 22];
            app.ThicknessLabel_11.Text = 'Width';

            % Create FigureSizeLabel_3
            app.FigureSizeLabel_3 = uilabel(app.Panel_18);
            app.FigureSizeLabel_3.FontWeight = 'bold';
            app.FigureSizeLabel_3.Position = [25 9 69 22];
            app.FigureSizeLabel_3.Text = 'Figure Size';

            % Create CloseFiguresButton_3
            app.CloseFiguresButton_3 = uibutton(app.Panel_18, 'push');
            app.CloseFiguresButton_3.ButtonPushedFcn = createCallbackFcn(app, @CloseFiguresButton_3Pushed, true);
            app.CloseFiguresButton_3.Tooltip = {''};
            app.CloseFiguresButton_3.Position = [403 10 89 20];
            app.CloseFiguresButton_3.Text = 'Close Figures';

            % Create SettoDefaultButton
            app.SettoDefaultButton = uibutton(app.Panel_18, 'push');
            app.SettoDefaultButton.ButtonPushedFcn = createCallbackFcn(app, @SettoDefaultButtonPushed, true);
            app.SettoDefaultButton.Tooltip = {'Use the boundary limits of the U-vector file'};
            app.SettoDefaultButton.Position = [403 76 84 20];
            app.SettoDefaultButton.Text = 'Set to Default';

            % Create ButtonGroup_12
            app.ButtonGroup_12 = uibuttongroup(app.Panel_18);
            app.ButtonGroup_12.AutoResizeChildren = 'off';
            app.ButtonGroup_12.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_12SelectionChanged, true);
            app.ButtonGroup_12.SizeChangedFcn = createCallbackFcn(app, @ButtonGroup_12SizeChanged, true);
            app.ButtonGroup_12.Position = [106 126 311 30];

            % Create PlotalldatainonefigureButton_2
            app.PlotalldatainonefigureButton_2 = uiradiobutton(app.ButtonGroup_12);
            app.PlotalldatainonefigureButton_2.Tooltip = {'All maps are displayed as subplots in one window'};
            app.PlotalldatainonefigureButton_2.Text = 'Plot all data in one figure';
            app.PlotalldatainonefigureButton_2.FontWeight = 'bold';
            app.PlotalldatainonefigureButton_2.Position = [13 4 164 22];
            app.PlotalldatainonefigureButton_2.Value = true;

            % Create PlotseparatelyButton_3
            app.PlotseparatelyButton_3 = uiradiobutton(app.ButtonGroup_12);
            app.PlotseparatelyButton_3.Tooltip = {'One map per window'};
            app.PlotseparatelyButton_3.Text = 'Plot separately';
            app.PlotseparatelyButton_3.FontWeight = 'bold';
            app.PlotseparatelyButton_3.Position = [193 4 108 22];

            % Create GeneralLayoutLabel_2
            app.GeneralLayoutLabel_2 = uilabel(app.VelocityMapTab);
            app.GeneralLayoutLabel_2.BackgroundColor = [0.9412 0.9412 0.9412];
            app.GeneralLayoutLabel_2.FontSize = 15;
            app.GeneralLayoutLabel_2.FontWeight = 'bold';
            app.GeneralLayoutLabel_2.FontColor = [0.0314 0.3686 0.6];
            app.GeneralLayoutLabel_2.Position = [38 290 131 22];
            app.GeneralLayoutLabel_2.Text = '  General Layout  ';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = FUNMAP_Output

            runningApp = getRunningApp(app);

            % Check for running singleton app
            if isempty(runningApp)

                % Create UIFigure and components
                createComponents(app)

                % Register the app with App Designer
                registerApp(app, app.UIFigure)

                % Execute the startup function
                runStartupFcn(app, @startupFcn)
            else
```

```

                % Focus the running singleton app
                figure(runningApp.UIFigure)

                app = runningApp;
            end

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```